PLANIFICACIÓN AUTOMÁTICA

Práctica 1: Planificación con PDDL

Grado en Ingeniería Informática
Universidad de Alcalá

Andrés Corbacho Sánchez
Gonzalo González Silverio
Javier Roma Rodríguez

Curso 2024-2025

PARTE 1

EJERCICIO 1.1: LOGÍSTICA DE SERVICIO DE EMERGENCIAS, VERSIÓN INICIAL

MODELADO DEL DOMINIO

El dominio se ha modelado utilizando la sintaxis STRIPS en PDDL, aprovechando la capacidad de tipado para definir las entidades involucradas, como las personas, las cajas, los tipos de contenido (comida, medicina), las localizaciones y el dron. Se ha optado por un diseño genérico en el que el contenido de las cajas se representa mediante el predicado (crate-content), lo que permite que, en caso de que se deseen introducir nuevos contenidos, no sea necesario modificar el dominio, sino únicamente la especificación del problema.

El dominio ha sido creado pensando en la posibilidad de escalar el problema hacia configuraciones que involucren planificación paralela con múltiples drones en las próximas partes de la práctica. Para ello, se ha modelado el dron como un objeto más, dotándolo de dos brazos independientes, permitiendo que el dron pueda cargar dos cajas simultáneamente, una en cada brazo. Las acciones de recoger y entregar cajas se definen de modo que se requiera que tanto el dron como la caja se encuentren en la misma localización, y en el caso de la entrega, se verifica además que la persona necesitada esté presente en esa ubicación y que el contenido de la caja corresponda con su necesidad. Al entregar la caja, el estado del dron se actualiza liberando el brazo ocupado y se marca a la persona como beneficiada del contenido necesario, eliminando así la necesidad previamente registrada.

Para evitar el uso de precondiciones negativas, se han empleado predicados contrarios que expresan de manera positiva el estado de los elementos. En lugar de indicar directamente la ausencia de disponibilidad, se utilizan los predicados arm-free y arm-load para gestionar los brazos del dron, y person-needs junto con person-has para gestionar las necesidades y la satisfacción de las personas.

CONSTRUCCIÓN DE PROBLEMAS

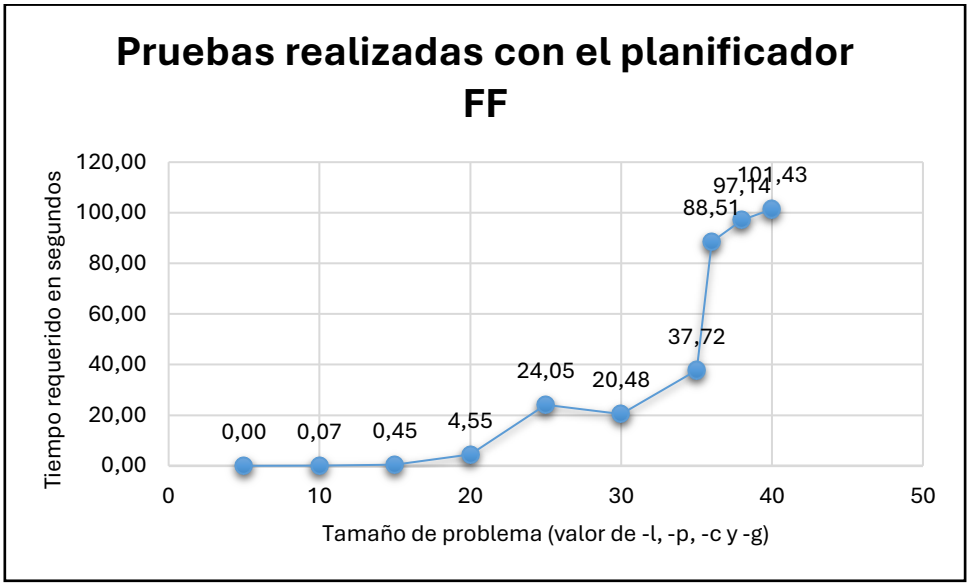
El primer problema se ha construido para representar una situación sencilla en la que se involucra un único dron, una única caja y una sola persona. En este escenario, tanto el dron como la caja se encuentran inicialmente en el depósito, mientras que la persona está situada en otra localización. La caja contiene comida, que es justamente lo que la persona necesita. La solución a este problema implica que el dron debe partir desde el depósito, recoger la caja mediante uno de sus brazos (siendo irrelevante cuál de los dos, ya que ambos están inicialmente libres), y trasladarla a la localización en la que se encuentra la persona. Una vez allí, el dron entrega la caja, cumpliendo con la condición de que el contenido de la caja (comida) satisface la necesidad de la persona. Finalmente, se exige que el dron vuelva al depósito.

El segundo problema introduce mayor complejidad al incluir tres cajas, dos personas y tres localizaciones distintas. En esta formulación, todas las cajas parten del depósito, pero se diferencian por su contenido: una caja contiene comida y las otras dos contienen medicina. Las necesidades se distribuyen de forma que la primera persona requiere tanto comida como medicina, mientras que la segunda persona necesita medicina. El plan generado dependerá del planificador que se emplee y de si este aprovecha ambos brazos del robot en lugar de llevar las cajas una a una usando un solo brazo, como ocurre al usar un planificador BFWS. El plan óptimo, devuelto por planificadores más avanzados como Delfi, consiste en recoger simultáneamente la caja con comida y una caja con medicina, trasladarse a la localización de la primera persona y efectuar las entregas correspondientes. Posteriormente, el dron debe regresar al depósito para recoger la caja restante con medicina y proceder a entregar dicho recurso a la segunda persona, asegurándose de que ambas personas reciban el contenido requerido. En todo momento se mantiene la restricción de que el dron, después de completar las entregas, debe volver al depósito.

EJERCICIO 1.2: GENERADOR DE PROBLEMAS EN PYTHON

En este ejercicio, primero se completó el esqueleto del generador modificando las secciones marcadas con TODO, adaptando la declaración de objetos para que coincida con los tipos del dominio (drone, location, crate, contents, person y carrier) y configurando el estado inicial para que cada dron y caja se ubiquen en el depósito, asignando ubicaciones aleatorias a las personas y estableciendo sus necesidades mediante (person-needs). Además, se generan y distribuyen los contenidos de las cajas de forma que cada tipo tenga al menos una asignación, y en la sección de metas se garantiza que todos los drones retornen al depósito y que cada persona reciba el contenido que necesita (representado con (person-has)).

Una vez comprobado el correcto funcionamiento del programa en Python, se generaron problemas de complejidad creciente manteniendo constantes los parámetros -d 1 -r 0 y variando los valores de -l, -p, -c y -g. Con cada tamaño de problema generado se utilizó el planificador FF para medir el tiempo necesario para encontrar una solución, el cual aumentó progresivamente con el tamaño del problema. Los resultados indican que para problemas pequeños (por ejemplo, 5 o 10) el tiempo es prácticamente nulo, mientras que al aumentar la complejidad los tiempos crecen de forma notable; se observa que a partir de un valor de 35 los tiempos superan el minuto. Cabe destacar que se produjo una fluctuación anómala al probar con el valor 30, donde el tiempo fue inferior al obtenido con el valor 25. A continuación se muestra una gráfica que resume los resultados obtenidos:



EJERCICIO 1.3: COMPARATIVA RENDIMIENTO DE PLANIFICADORES

1. Basado en el análisis experimental, se puede concluir lo siguiente:
    - **Capacidad de procesar problemas de mayor tamaño:**
      LPG-TD es capaz de resolver instancias de mayor tamaño dentro del límite de 1 minuto. Según los resultados (tomados de la tabla generada a partir de casi 2500 filas de datos), LPG-TD resolvió hasta un problema de tamaño 15, mientras que FF y SGPLAN40 alcanzaron respectivamente tamaños máximos de 12 y 13.
    - **Tiempo de resolución:**
      LPG-TD no solo resolvió instancias más grandes, sino que además lo hizo en menos tiempo (por ejemplo, 40.3 s para el tamaño máximo alcanzado) en comparación con FF (55.2 s) y SGPLAN40 (58.7 s).
    - **Número de acciones del plan:**

Aunque los planes generados por los tres planificadores tienen una longitud similar (entre 38 y 42 acciones), esta métrica no diferencia significativamente la calidad de la solución entre ellos.

- **Conclusión final:**
  LPG-TD es el mejor planificador en términos de capacidad para procesar problemas de mayor tamaño dentro de un tiempo límite de 1 minuto, ya que alcanza instancias más complejas y lo hace de manera más rápida, sin sacrificar la calidad del plan.

2. Se seleccionó como instancia la de mayor tamaño que el planificador FF puede resolver en menos de 1 minuto. A este mismo problema se le ejecutó con LPG-TD (usando la opción –n 1) y SGPLAN40. Para cada planificador se capturó un fragmento representativo de la salida por consola, y se midió el tiempo de ejecución (en segundos) y el número de pasos (acciones) del plan generado.
   Tabla de resultados:

Planificador	Tiempo(s)	Acciones	Tamaño
LPG-TD	34.14	506	127
SGPLAN40	45.54	426	94
FF	11.62	132	34

- Tiempo de generación de solución:
  - FF es el más rápido, generando una solución en 11.62 s.
  - Sin embargo, esta eficiencia en tiempo se alcanza a costa de resolver un problema de muy baja complejidad (tamaño 34).
- Capacidad para resolver problemas de mayor tamaño:
  - LPG-TD es capaz de resolver instancias mucho más complejas (tamaño 127) dentro del límite de 1 minuto.
  - SGPLAN40 también resuelve problemas de mayor tamaño que FF (tamaño 94), pero su tiempo es superior a LPG-TD.
- Tamaño de la solución (número de pasos):
  - FF genera planes compactos (132 pasos), lo cual podría interpretarse como una solución más óptima en términos de compacidad.
  - LPG-TD y SGPLAN40 generan planes considerablemente más largos (506 y 426 pasos, respectivamente), lo que refleja la mayor complejidad del problema resuelto.
- Por tanto, concluimos que:
  - Según el tiempo de generación de la solución: LPG-TD es el mejor, ya que resuelve el problema de mayor tamaño en 34.14 s, superando a SGPLAN40 (45.54 s) y, aunque FF es más rápido, solo puede resolver problemas muy simples.
  - Según el tamaño de la solución (número de pasos): Si bien FF genera planes mucho más cortos (132 pasos), esto se debe a que solo puede abordar problemas de baja complejidad. En términos absolutos, los planes generados por LPG-TD y SGPLAN40 son más extensos, reflejando la capacidad para resolver problemas complejos.
    Si se valora la compacidad del plan, FF sería el mejor; sin embargo, si el criterio es la capacidad para procesar problemas complejos (lo que generalmente se espera en entornos de planificación), LPG-TD y SGPLAN40 son superiores, con LPG-TD siendo el líder dado su menor tiempo.

PARTE 2

EJERCICIO 2.1: SERVICIO DE EMERGENCIAS, TRANSPORTADORES

Se ha verificado experimentalmente que, en el dominio modificado que incorpora transportadores, los problemas generados pueden ser resueltos por los planificadores de la parte 1. Sin embargo, se observa que:

- LPG-TD es el único que incorpora en sus planes las nuevas acciones relacionadas con el transportador (por ejemplo, MOVER-TRANSPORTADOR), lo cual le permite explotar la capacidad de transportar múltiples cajas simultáneamente.
- FF y SGPLAN40 continúan resolviendo las instancias utilizando únicamente las acciones clásicas (COGER-CAJA, MOVER_DRON, ENTREGAR-CAJA), sin hacer uso de las nuevas operaciones que facilitan el transporte.

Estos resultados indican que, a pesar de que el dominio y los problemas se han adaptado correctamente para incluir el mecanismo de transportador, solo LPG-TD ha modificado su estrategia de búsqueda para beneficiarse de este nuevo mecanismo. Esto es relevante para la evaluación de la eficiencia y escalabilidad del planificador, ya que el uso de transportadores puede reducir el número de viajes y, en problemas de mayor complejidad, mejorar el rendimiento.

Se incluyen fragmentos de la ejecución de los planificadores:

Planificador FF:

ff: parsing domain file

domain 'DRON-STRIPS' defined ... done.

ff: parsing problem file

problem 'DRONE_PROBLEM_D1_R1_L10_P10_C10_G10_CT2' defined ... done.

...

ff: found legal plan as follows

step 0: COGER-CAJA DRON1 CAJA1 DEPOSITO

step 1: MOVER_DRON DRON1 DEPOSITO LOC7

step 2: ENTREGAR-CAJA CAJA1 DRON1 LOC7 COMIDA PERSONA8

...

step 38: ENTREGAR-CAJA CAJA2 DRON1 LOC7 MEDICINA PERSONA8

...

time spent: 0.31 seconds total time

Planificador LPG-TD:

NUMERIC_THREATS_MODE: 0

...

Parsing domain file: domain 'DRON-STRIPS' defined ... done.

Parsing problem file: problem 'DRONE_PROBLEM_D1_R1_L10_P10_C10_G10_CT2' defined ... done.

...

Plan computed:

Time: (ACTION) [action Duration; action Cost]

0.0000: (COGER-CAJA DRON1 CAJA1 DEPOSITO) [D:1.00; C:1.00]

1.0000: (MOVER_DRON DRON1 DEPOSITO LOC2) [D:1.00; C:1.00]

2.0000: (MOVER_DRON DRON1 LOC2 LOC7) [D:1.00; C:1.00]

3.0000: (ENTREGAR-CAJA CAJA1 DRON1 LOC7 COMIDA PERSONA8) [D:1.00; C:1.00]

4.0000: (MOVER_DRON DRON1 LOC7 DEPOSITO) [D:1.00; C:1.00]

5.0000: (MOVER-TRANSPORTADOR TRANSPORTADOR1 DRON1 LOC4 DEPOSITO) [D:1.00; C:1.00]

6.0000: (MOVER_DRON DRON1 LOC4 DEPOSITO) [D:1.00; C:1.00]

...

time spent: 19.04 seconds total time

Planificador SGPLAN40:

; Time 0.02

; ParsingTime 0.00

; NrActions 45

...

0.010: (COGER-CAJA DRON1 CAJA1 DEPOSITO)[0.000]

0.020: (MOVER_DRON DRON1 DEPOSITO LOC7)[0.000]

0.030: (ENTREGAR-CAJA CAJA1 DRON1 LOC7 COMIDA PERSONA8)[0.000]

0.040: (MOVER_DRON DRON1 LOC7 DEPOSITO)[0.000]

0.050: (COGER-CAJA DRON1 CAJA10 DEPOSITO)[0.000]

0.060: (MOVER_DRON DRON1 DEPOSITO LOC2)[0.000]

0.070: (ENTREGAR-CAJA CAJA10 DRON1 LOC2 MEDICINA PERSONA1)[0.000]

0.080: (MOVER_DRON DRON1 DEPOSITO LOC6)[0.000]

...

EJERCICIO 2.2: SERVICIO DE EMERGENCIAS, COSTES DE ACCIÓN

Para que los planificadores puedan aprovechar eficazmente el uso de transportadores, es necesario asignar costes específicos a cada acción. Esto es porque evaluar una solución únicamente por la cantidad de acciones no refleja necesariamente si es mejor o peor, ya que no representa ningún coste real asociado. Por esta razón, hemos incorporado el requisito :action-costs, que nos permite definir un coste individual y realista para cada acción del dominio.

Hemos introducido dos funciones clave: total-cost, que acumula el coste total y que se utilizará como métrica a minimizar, y fly-cost, que especifica explícitamente el coste de volar entre dos localizaciones concretas. Las acciones de vuelo, tanto con transportador como sin él, tienen asignado un coste significativamente más alto en comparación con el resto de acciones, como recoger o entregar cajas. De esta forma, aquellos planificadores que gestionen adecuadamente los costes priorizarán el uso del transportador, reduciendo así los vuelos innecesarios y costosos.

En cuanto al generador de problemas, se ha modificado para que inicialice correctamente ambas funciones de coste, asignando valores específicos a cada par de localizaciones mediante la función flight_cost(). Además, se ha definido explícitamente la métrica a minimizar (el coste total) en la especificación del problema generado.

Finalmente, se presenta a continuación una tabla comparativa del rendimiento obtenido por los diferentes planificadores analizados.

Parámetros	Metric-FF	FD(LAMA)	FD(BJOLP)	FD (FDSS2)
5	0.00s, coste 1437	0.00s, coste 1352	4.86s, coste 725	0.79s, coste 725
10	0.05s, coste 3393	0.00s, coste 3040	ERROR	ERROR(Tiempo Limite)
20	ERROR	0.03s, coste 5868	-	-
50	-	1.59s, coste 13940	-	-
100	-	33.31s, coste 35052	-	-
105	-	Tarda más de 60s	-	-

Metric-FF proporciona rápidamente una solución inicial, pero con una minimización del coste bastante pobre. Además, cuando los parámetros de los problemas superan el valor 10, Metric-FF presenta errores debido a su incapacidad para manejar problemas tan grandes. Por otro lado, Fast Downward en sus versiones BJOLP y FDSS2 obtiene soluciones con costes significativamente más bajos que otros planificadores. En particular, FDSS2 consigue esto en tiempos menores que BJOLP, aunque ambos empiezan a fallar al superar problemas con valor superior a 5 en sus parámetros.

La versión LAMA de Fast Downward genera múltiples soluciones iterativas rápidamente (planificador tipo anytime), encontrando la primera solución en un tiempo muy corto, aunque esta solución inicial suele tener un coste más alto en comparación con las soluciones proporcionadas por otros planificadores. Aun así, LAMA destaca especialmente por resolver con facilidad problemas de mayor tamaño.

En definitiva, cada planificador se comporta según la estrategia para la cual fue diseñado: LAMA devuelve rápidamente soluciones sucesivas (anytime), BJOLP devuelve la solución con el menor coste posible (óptimo), y FDSS2 combina rapidez y calidad (portfolio).

PARTE 3

EJERCICIO 3.1: ACCIONES CONCURRENTES EN GESTIÓN DE EMERGENCIAS

En este apartado se analiza, acción por acción, la posibilidad de realizar cada una de ellas en paralelo con otras.

La primera acción, recoger-caja, puede realizarse simultáneamente por varios drones, siempre y cuando cada dron recoja una caja distinta. No sería posible que varios drones recogieran la misma caja al mismo tiempo.

En cuanto a la acción meter-caja-transportador, es posible que múltiples drones coloquen cajas simultáneamente, tanto en distintos transportadores como en un mismo transportador. Sin embargo, no se permite introducir la misma caja en más de un transportador al mismo tiempo.

Respecto a las acciones relacionadas con desplazamientos (volar y volar-con-transportador), ambas pueden ejecutarse simultáneamente sin interferencia, salvo que en la acción volar-con-transportador dos drones intenten transportar el mismo transportador al mismo tiempo, lo cual no es permitido.

Finalmente, la acción sacar-caja-transportador sigue una lógica similar a la acción de meter cajas, permitiendo sacar distintas cajas del mismo o de diferentes transportadores de forma simultánea, pero impidiendo que una misma caja sea retirada por múltiples drones al mismo tiempo. En cuanto a entregar-caja, no es posible que una misma persona reciba más de una caja simultáneamente, ya que una persona no podría recoger ambas cajas a la vez. Asimismo, tampoco se permite entregar simultáneamente una misma caja a más de una persona.

EJERCICIO 3.2: IMPLEMENTACIÓN Y PRUEBAS DE CONCURRENCIA

Una vez adaptado el dominio mediante la creación de durative-actions para permitir planes concurrentes, se procede a realizar diversas pruebas utilizando problemas con diferentes tamaños y parámetros.

Tras modificar el archivo generate-problem.py, se generan múltiples problemas para evaluar el rendimiento de la concurrencia. Dada la complejidad que implican estos planes concurrentes, comienzo probando con un problema pequeño compuesto por 2 drones, 1 transportador, y 2 personas, localizaciones, cajas y metas. Para este problema, el planificador OPTIC encuentra una primera solución en tan solo 0.02 segundos, con un coste temporal de 415 segundos.

Sin embargo, la solución obtenida no utiliza ningún transportador. Esto ocurre porque OPTIC aprovecha la posibilidad de paralelizar las acciones entre los dos drones disponibles, distribuyendo las cajas equitativamente entre ambos y resultando en un plan más rápido que el que podría obtenerse utilizando un único dron y el transportador. Al aumentar la cantidad de cajas y metas a 3, se esperaría que el planificador hiciera uso del transportador para optimizar tiempos.

Sin embargo, incluso con estos parámetros ampliados, OPTIC continúa generando soluciones sin utilizar el transportador. Este comportamiento se debe probablemente al carácter anytime del planificador OPTIC, es decir, la primera solución devuelta no necesariamente es óptima, aunque sea muy rápida (0.02 segundos, coste temporal de 611 segundos). En teoría, al esperar una segunda solución más eficiente, debería encontrarse un plan que aproveche el transportador. Pero en la práctica, OPTIC no genera una segunda solución ni siquiera después de más de 30 minutos de ejecución, lo que evidencia la gran complejidad computacional del problema.

Al evaluar el mismo problema con el planificador Temporal Fast Downward, tampoco encontramos soluciones que hagan uso del transportador entre los planes devueltos.

En conclusión, aunque la concurrencia en planificación tiene un potencial significativo, su incorporación aumenta considerablemente la complejidad computacional, llevando a los planificadores a tiempos de ejecución excesivos y, en muchos casos, a soluciones subóptimas o desaprovechando los recursos disponibles, como sucede aquí con el transportador.