Resolución Laboratorio: Planificación de STRIPS

Contenido

[DESCRIPCIÓN DE LA ACTIVIDAD 1](#_Toc43799041)

[INFORME 1](#_Toc43799042)

[DESCRIPCIÓN DE LOS ESCENARIOS 3](#_Toc43799043)

[RESULTADOS PLANES 5](#_Toc43799044)

[DIFICULTADES ENCONTRADAS 10](#_Toc43799045)

[BIBLIOGRAFÍA 11](#_Toc43799046)

[ANEXOS 12](#_Toc43799047)

DESCRIPCIÓN DE LA ACTIVIDAD

En una habitación hay un mono, una caja y un plátano, tal como indica la figura (situación inicial). El objetivo del mono es tener el plátano. El mono puede:

* Ir de una posición a otra.
* Empujar la caja de una posición a otra si está en la misma posición que ella y no está sobre ella.
* Subirse a la caja si está en la misma posición que ella.
* Coger el plátano si está en encima de la caja.

**Opción 2.**Realizar el modelado del escenario "habitación-mono-platano" en PDDL (dominio.pddl y problema.pddl - 3 problemas) y ejecutarlo con 3 (tres) planificadores. Realizar el informe detallado y adjuntar el código fuente.

INFORME

Se desarrolló la solución utilizando el lenguaje PDDL, se crearon varios archivos con el código fuente y se describen a continuación:

El archivo domain.pddl contiene la información del domino cuyo nombre es **dominio-mono**. En la sección requirements, utilizada para la definición del dominio, es en donde se indicará cuales características de PPDL se utilizará en este(Haslum et al., 2019), se incluye los elementos:

* **strips:** El cual indica que el dominio está en la forma simple. Corresponde a un conjunto de entradas y salidas generalmente bien definidas(Haslum et al., 2019).
* **typing:** Para el uso de objetos manejo de herencias y representaciones de clases(Haslum et al., 2019).
* **negative-preconditions:** Para el uso de predicados tipos falso en las acciones (Haslum et al., 2019).

En la sección types se identificarán los diferentes tipos de clases a los que pertenecen las entidades que se encuentran en el mundo. Se utilizan nombres auto descriptivos como son:

* **posicion:** Utilizada para referenciar la posición donde se encuentran los objetos en el mundo.
* **entidadMono:** Clase utilizada para representar los monos.
* **entidadCaja:** Clase utilizada pare representar las cajas.
* **entidadBanana:** Clase utilizada para representar las bananas.

En la sección predicados se realizará la definición del dominio que contienen generalmente la lista de estado del modelo. Se establecerá predicados con una representación de variables binarias, que representan hechos que son verdaderos o falsos(Haslum et al., 2019).

Este caso se utilizarán los siguientes predicados para determinar la posición de las entidades en el mundo con su respectiva posición.

    (localizacion ?ent - entidadMono ?pos - posicion)

    (localizacionCaja ?ent - entidadCaja ?pos - posicion)

    (localizacionBanana ?ent - entidadBanana ?pos - posicion)

El siguiente predicado representa la navegación entre las diferentes posiciones.

    (inc ?x - posicion ?y - posicion)

Se crean predicados para determinar el estado de las entidades en el mundo al encontrar las cajas y buscar recoger las bananas.

    (concaja ?ent - entidadMono)

    (encimacaja ?ent - entidadMono ?caja - entidadCaja)

    (recojebanana ?ent - entidadMono ?banana - entidadBanana)

Se crearon las siguientes acciones que serán ejecutas por las entidades en el mundo.

* **mover-mono:** Acción utilizada para mover el mono de una posición a otra.
* **encontrar-mono-caja:** Acción utilizada para informar cuando el mono ha encontrado una caja en alguna posición.
* **subir-mono-caja:** Acción utilizada para que el mono suba a la caja.
* **bajar-mono-caja:** Acción utilizada para que el mono baje de la caja.
* **recoger-mono-banana:** Acción del mono al encontrar las bananas y estar encima de la caja.
* **empujar-mono-caja:** Acción utilizada para mover el mono en conjunto con la caja de una posición a otra.

Los archivos con el formato problem-nombre-problema tienen la información de los diferentes problemas que harán uso del dominio(dominio-mono). En la sección objects, se declaran los objetos que interactuaran en el mundo, en donde los nombres son auto descriptivos y haremos uso de posiciones en donde se encontrarán las diferentes entidades que son representadas por monos, banas y cajas.

En la sección init, se inicializan los predicados y condiciones iniciales del mundo, los cuales serán análogos para los diferentes problemas que serán planteados. Por ejemplo:

(inc p1 p2)

(inc p2 p3)

(inc p3 p2)

(inc p2 p1)

(localizacion m1 p1)

(localizacionCaja c1 p3)

(localizacionBanana b1 p1)

En la sección goal, se inicializan los objetivos meta que se desea alcanzar por las entidades en el mundo, los cuales serán análogos para los diferentes problemas que serán planteados. Por ejemplo:

(recojebanana m1 b1)

DESCRIPCIÓN DE LOS ESCENARIOS

A continuación, se realiza una descripción gráfica de los diferentes escenarios(problemas), que en adelante serán identificados como escenario1, escenario2 y escenario3.

* Para el problema problem-base.pddl(escenario1), en este escenario el mono de la posición 3 agarrará las bananas en la posición 2.

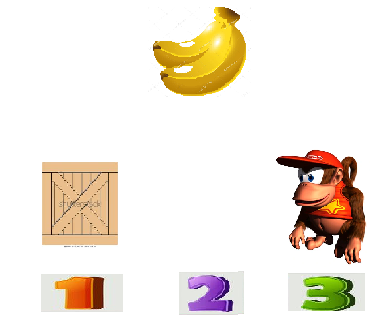


Figura situación inicial(escenario1)

* Para el problema problem-multiple-mono-banana.pddl (escenario2), en este escenario el mono de la posición 1 agarrará las bananas de las posiciones 1 y 4. Por otro lado, el mono de la posición 2 agarrará las bananas de la poción 3.

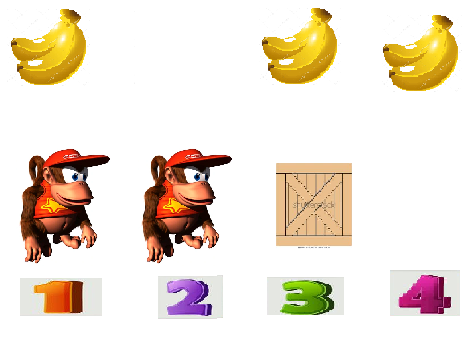


Figura problema múltiple escenario2

* Para el problema problem-multiple-mono-banana-caja.pddl(escenario3), en el escenario uno de los monos de la posición 1 agarrará las bananas de la posición 4; adicionalmente, otro mono en la misma posición agarrara las bananas de las posiciones 2 y 4 . Finalmente, el mono de la posición 6 agarrará las bananas de la poción 1.

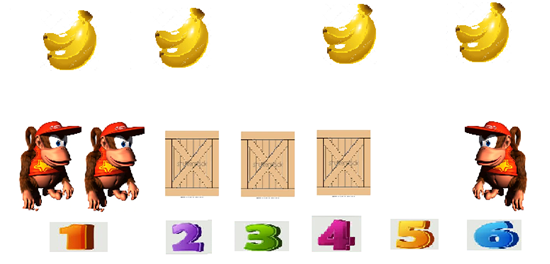


Figura problema múltiple escenario3

RESULTADOS PLANES

1. **Fast-Forward(FF). -** es un sistema de planificación independiente del dominio desarrollado por Joerg. FF puede manejar tareas de planificación ADL clásicas, así como ADL a escala completa(Joerg, s. f.-a).

**Fast-Forward(FF) con JAVA**

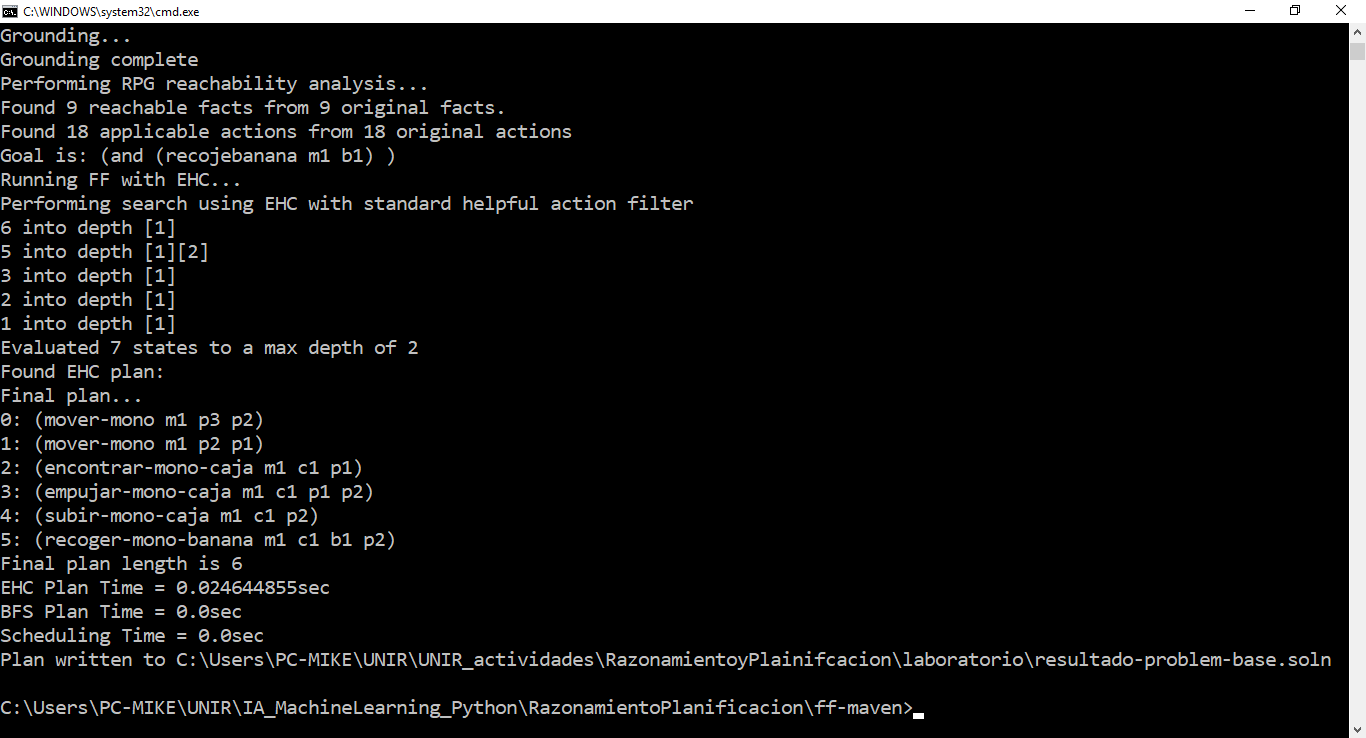


Figura resultados FF escenario1

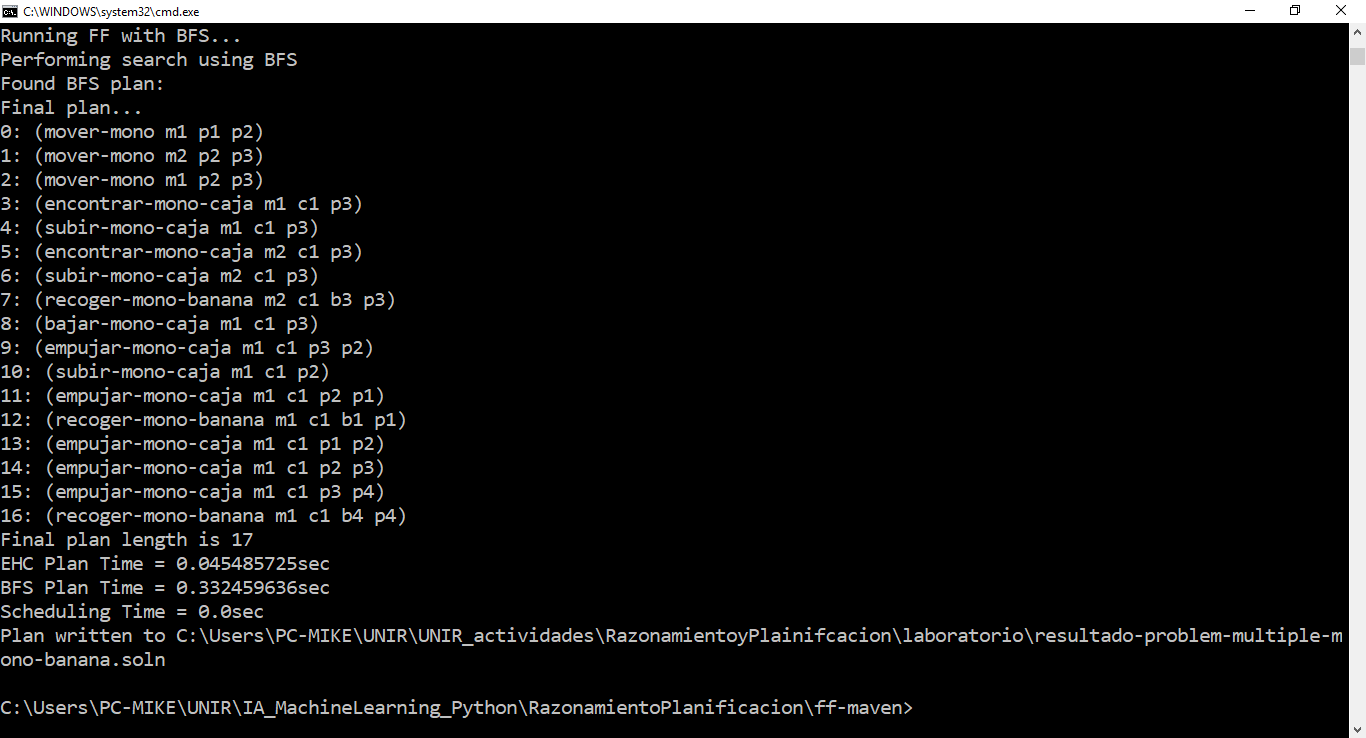


Figura resultados FF escenario2

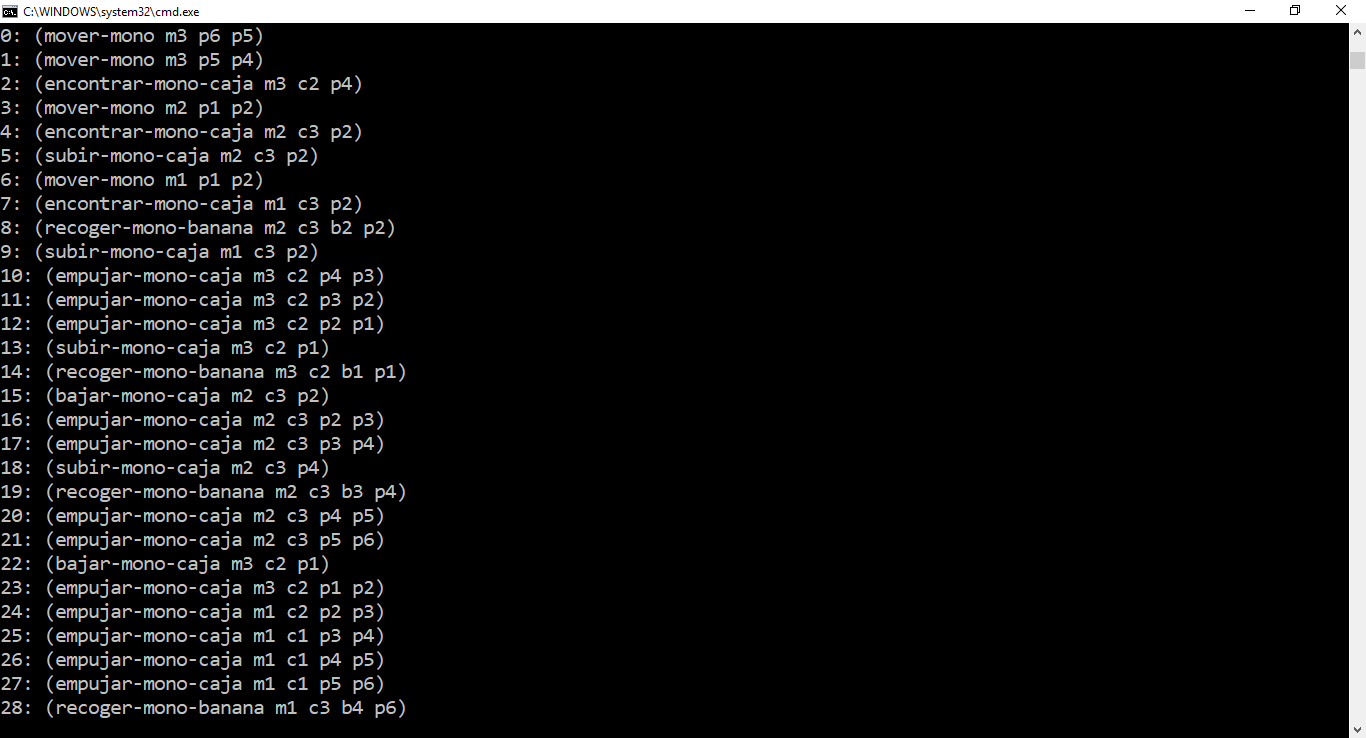


Figura resultados FF escenario3

**Fast-Forward(FF) con LINUX**

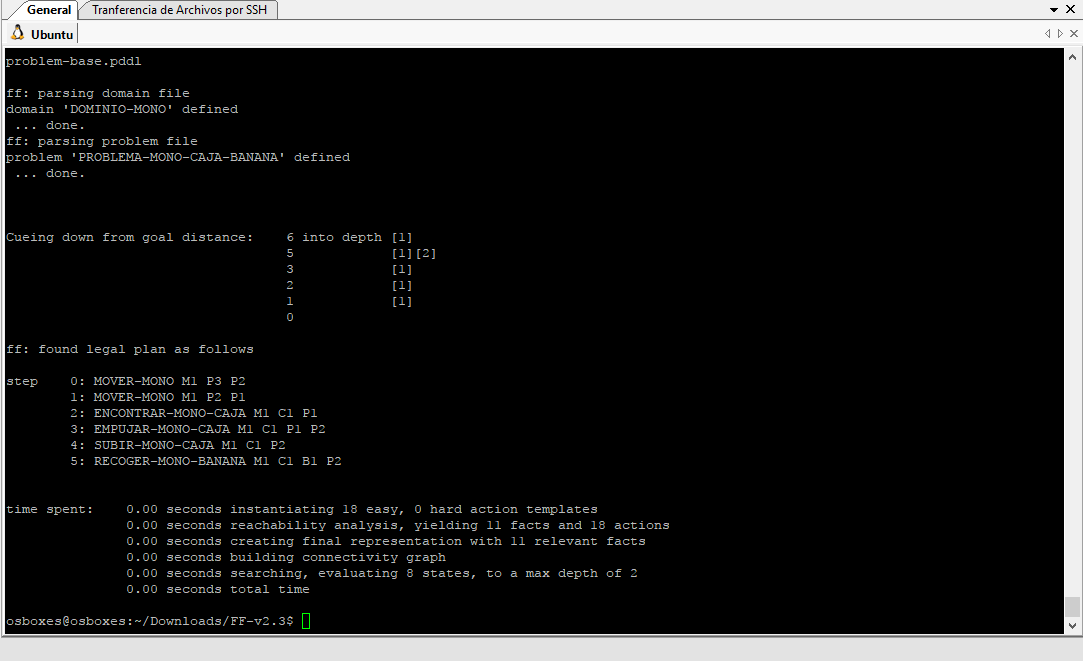
****

Figura resultados FF Linux escenario1

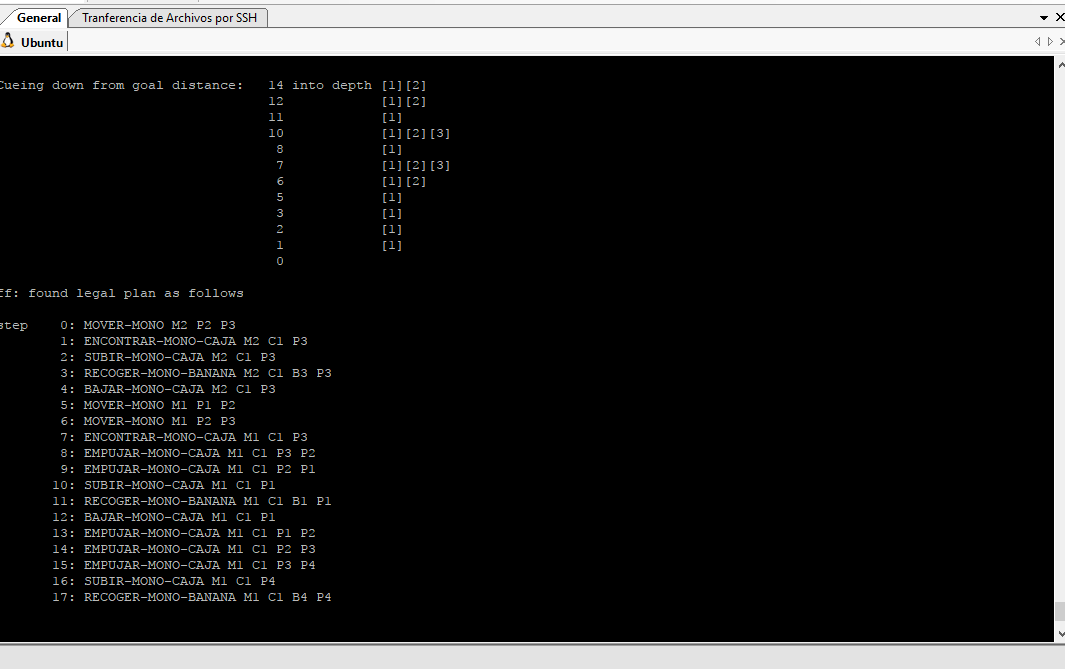
****

Figura resultados FF Linux escenario2

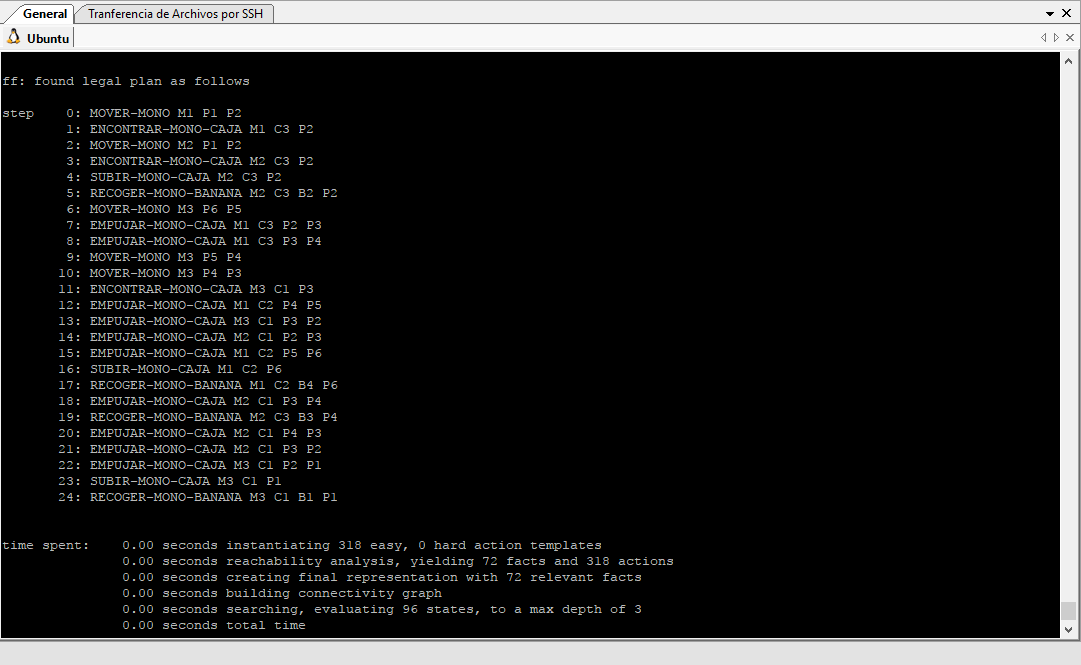


Figura resultados FF Linux escenario3

1. **LPG-td.-** es una nueva versión de LPG que mejora la versión 1.2, y que participó en la 4ta Competencia Internacional de Planificación, 2004(LPG Team, s. f.). LPG-td es una extensión de LPG para manejar las nuevas características de la planificación estándar lenguaje de descripción de dominio PDDL(Gerevini & Serina, 2002).

**LPG-td con LINUX**

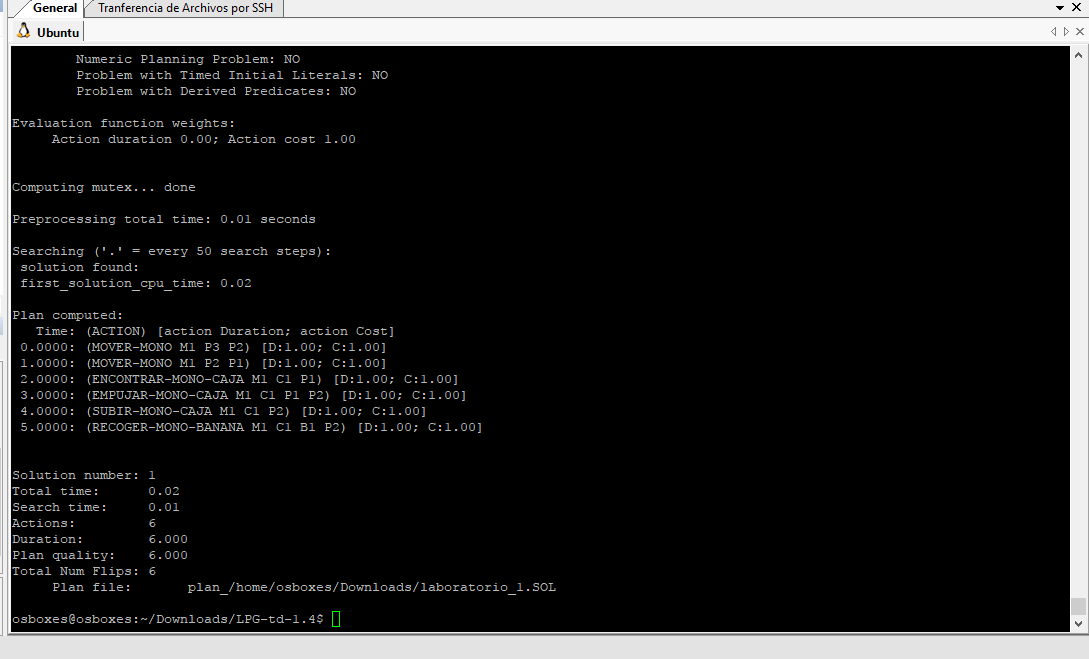


Figura resultados LPG Linux escenario1

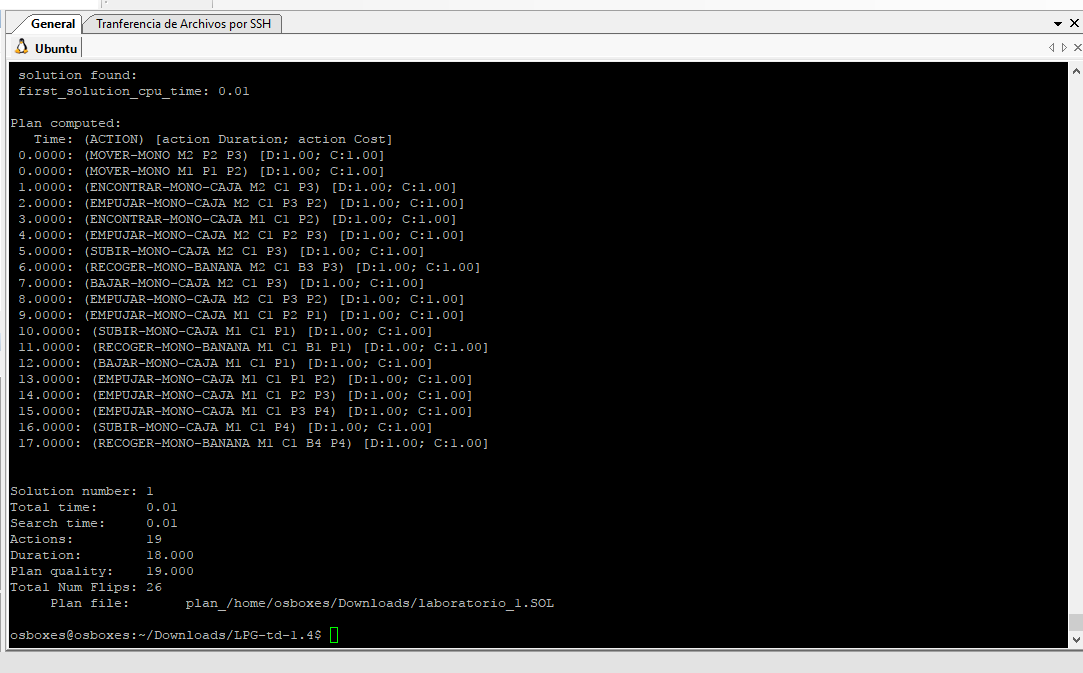


Figura resultados LPG Linux escenario2

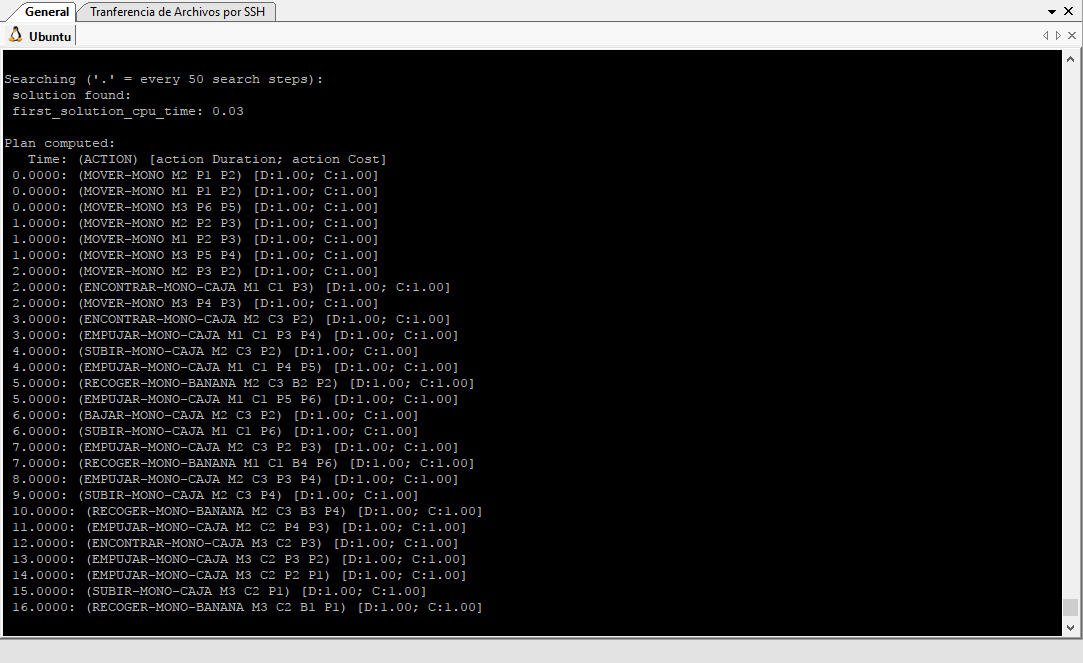
****

Figura resultados LPG Linux escenario3

1. **Planning domains(PD). -** Es un componente planning.domains que consiste en un planificador y validador automatizado en la nube. Puede invocar el software enviando enlaces a los archivos PDDL o enviando contenido PDDL en formato JSON directamente para recuperar o validar un plan (Andrew Coles, s. f.-b). Se utilizará la conexión a la URl http://solver.planning.domains/solve.

**VSCode PD con Windows**

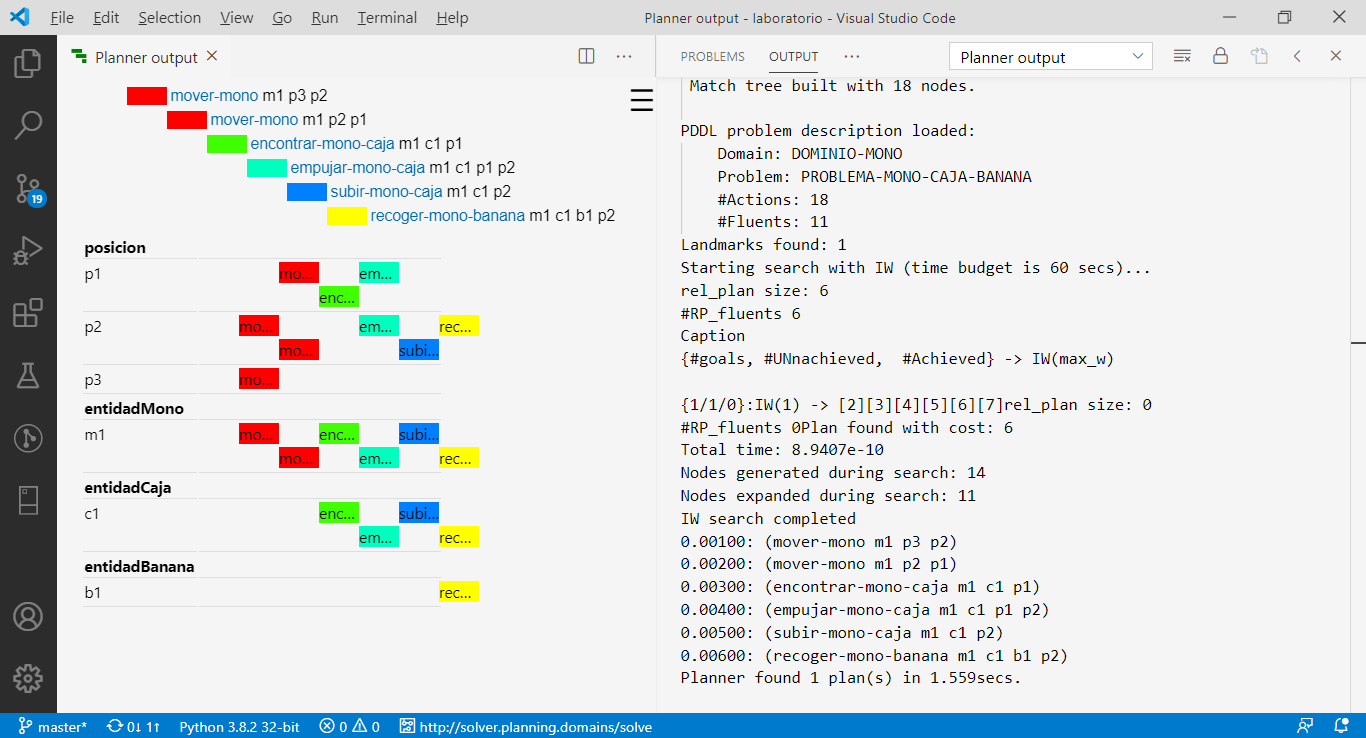


Figura resultados PD escenario1

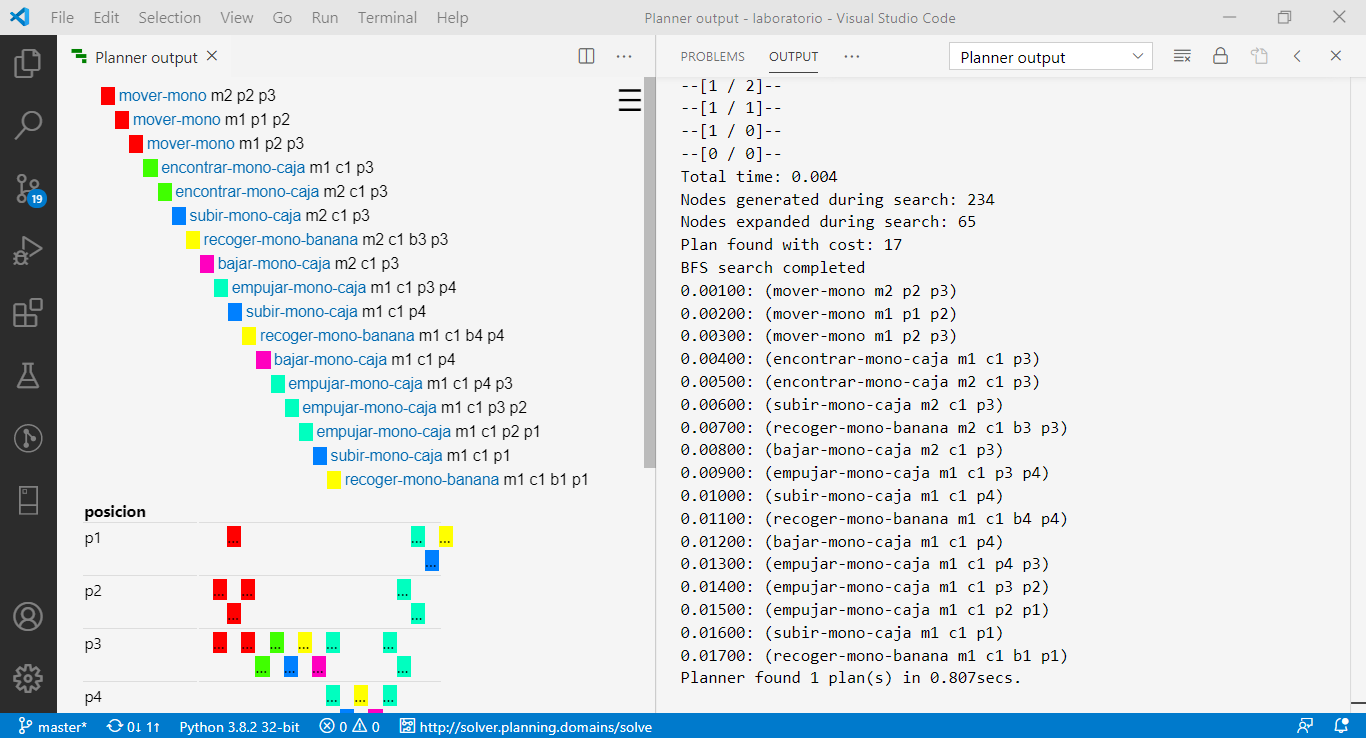


Figura resultados PD escenario2

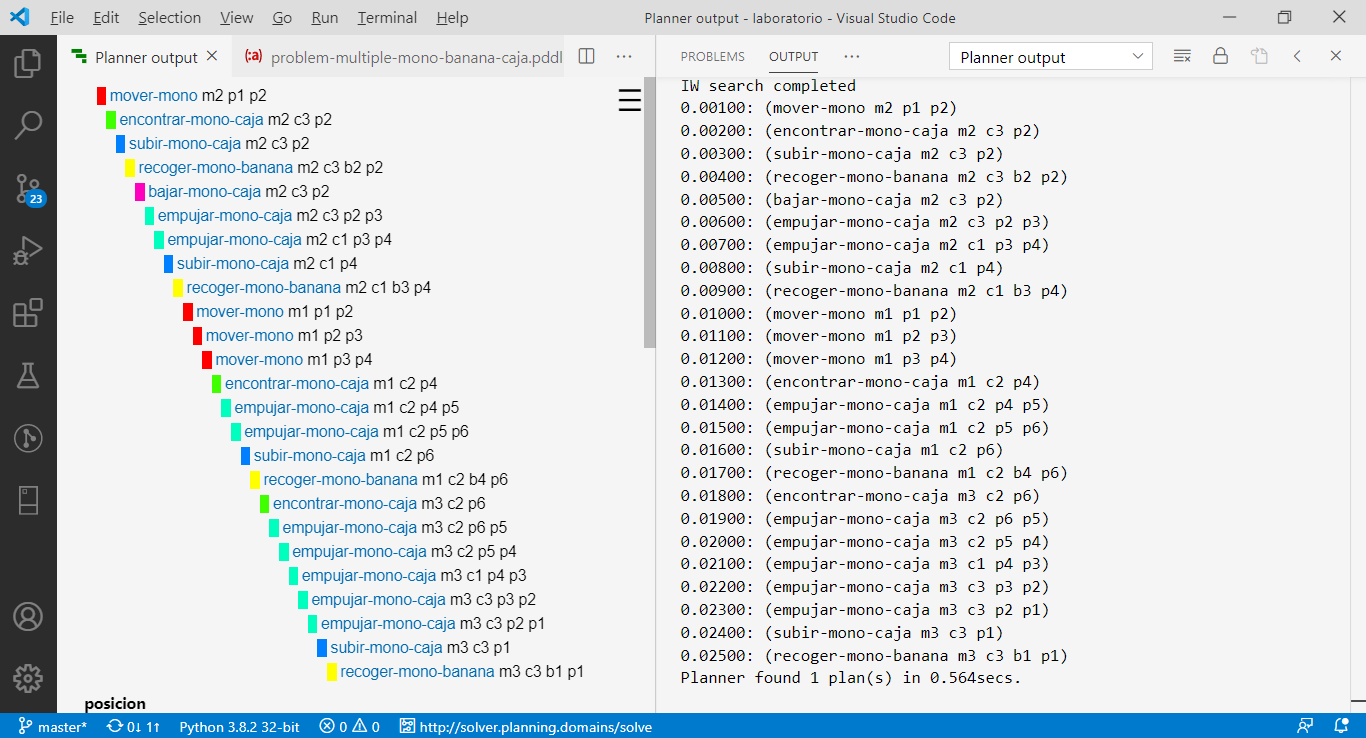


Figura resultados PD escenario3

DIFICULTADES ENCONTRADAS

* Respecto de la codificación en el lenguaje pddl, es un muy diferente de los lenguajes de programación tradicionales. Para el desarrollo del ejercicio se tomó la sugerencia en clase relacionada con el ir creando pequeños micro objetivos e irlos testeando. Por ejemplo, primero probar la acción moverse, luego probar la acción encontrar caja, luego probar la acción subir caja y así sucesivamente con la consecución de las diferentes micro-metas se logró realizar un mejor entendimiento del lenguaje. Finalmente se hizo un refuerzo con un libro para entender de forma práctica cómo funciona el lenguaje.
* Respecto del uso de Linux para la compilación de los planificadores fue necesario realizar varias modificaciones en los archivos de rutas de recursos del sistema operativo. Luego de descargar e instalar una máquina de virtual, distribuidas por <https://www.osboxes.org/> para Oracle VM VirtualBox(Umair, s. f.), fue necesario la instalación de varias librerías necesarias para la compilación de los planificadores. Finalmente, se procedió a seguir los pasos para la compilación vistos en clase.
* Respecto del planificador FF en JAVA, el cual hace referencia al trabajo de David Pattison(Pattison, 2015),al navegar hacia los links, al parecer los recursos no existen, entonces se optó por utilizar el código de la siguiente página <https://github.com/dpattiso/javaff> y generar un nuevo jar(dpattiso, 2017/2020). Luego, se creó un proyecto en formato MAVEN, el cual genera un JAR con sus dependencias.

BIBLIOGRAFÍA

Andrew Coles. (s. f.-b). *Solver.planning.domains*. Recuperado 23 de junio de 2020, de http://solver.planning.domains/

Gerevini, A., & Serina, I. (2002). LPG: A Planner Based on Local Search for Planning Graphs with Action Costs. *AIPS*, *2*, 281–290.

Haslum, P., Lipovetzky, N., Magazzeni, D., & Muise, C. (2019). *An introduction to the planning domain definition language* (Vol. 13). Morgan & Claypool Publishers.

Joerg. (s. f.-a). *FF Homepage*. Recuperado 23 de junio de 2020, de https://fai.cs.uni-saarland.de/hoffmann/ff.html

Pattison, D. T. (2015). *A new heuristic-based model of goal recognition without libraries* [PhD Thesis]. University of Strathclyde.

Umair. (s. f.). *OSBoxes—Virtual Machines for VirtualBox & VMware*. OSBoxes - Virtual Machines. Recuperado 23 de junio de 2020, de https://www.osboxes.org/

ANEXOS

Se incluye como anexos los siguientes archivos:

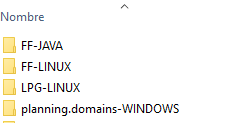
Dominio:

* domain.pddl

Problemas:

* problem-base.pddl
* problem-multiple-mono-banana.pddl
* problem-multiple-mono-banana-caja.pddl

Los resultados(resultadosplanificadores.rar) se encuentran en las siguientes carpetas:



Resultados:

* resultado-problem-base.soln
* resultado-problem-multiple-mono-banana.soln
* resultado-problem-multiple-mono-banana-caja.soln