

Actividad 1: Artículo científico de planificadores del estado del arte

Aldo Alberto Bernal Castillo
13/03/2022

Razonamiento y Planificación Automática

Abstract

El presente trabajo muestra la ejecución y entendimiento de 4 planificadores que fueron mostrados en la conferencia ICAPS 2018. El trabajo enseña a detalle desde la instalación de Ubuntu 20.04.4 LTS en una máquina virtual Oracle VirtualBox 6.1, instalación de Singularity, así como la obtención y ejecución de las planificadoras DecStar, Delfi1, MSP y Scorpion.

Introducción

La planificación y programación automatizadas, a veces denominada simplemente planificación de IA, es una rama de la inteligencia artificial que se ocupa de la realización de estrategias o secuencias de acción, normalmente para que las ejecuten agentes inteligentes, robots autónomos y vehículos no tripulados. A diferencia de los problemas clásicos de control y clasificación, las soluciones son complejas y deben descubrirse y optimizarse en un espacio multidimensional. La planificación también está relacionada con la teoría de la decisión.

En entornos conocidos con modelos disponibles, la planificación se puede realizar sin conexión. Las soluciones se pueden encontrar y evaluar antes de la ejecución. En entornos dinámicamente desconocidos, la estrategia a menudo debe revisarse en línea. Los modelos y las políticas deben adaptarse. Las soluciones suelen recurrir a procesos iterativos de prueba y error comúnmente vistos en inteligencia artificial. Estos

procesos iterativos de prueba y error comúnmente vistos en inteligencia artificial. Estos incluyen programación dinámica, aprendizaje por refuerzo y optimización combinatoria. Los lenguajes utilizados para describir la planificación y la programación a menudo se denominan lenguajes de acción.

En la planificación de IA, los planificadores suelen introducir un modelo de dominio (una descripción de un conjunto de acciones posibles que modelan el dominio), así como el problema específico que se resolverá especificado por el estado inicial y el objetivo, en contraste con aquellos en los que no hay dominio de entrada especificado.

Dichos planificadores se denominan "independientes del dominio" para enfatizar el hecho de que pueden resolver problemas de planificación de una amplia gama de dominios.

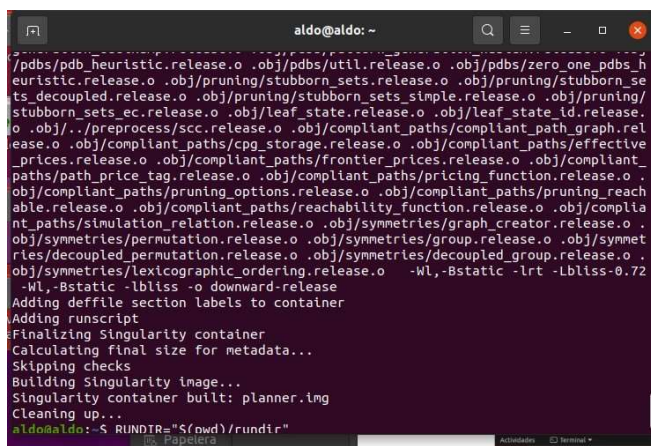
Los ejemplos típicos de dominios son el apilamiento de bloques, la logística, la gestión del flujo de trabajo y la planificación de tareas de robots. Por lo tanto, se puede utilizar un solo planificador independiente del dominio para resolver problemas de planificación en todos estos diversos dominios. Por otro lado, un planificador de rutas es típico de un planificador de dominio específico.

Desarrollo

Planificador DecStar

DecStar es una técnica recientemente introducida en la planificación de IA. Explota la independencia entre los componentes de una tarea de planificación para reducir el tamaño de la espacio-estado de la representación. Dividiendo las variables de estado en componentes, tal que la interacción entre estos toma la forma de una **topología estrella**, la búsqueda desacoplada solo se busca sobre secuencias de acción afectando el componente central de la topología, y enumera asignaciones accesibles para cada componente por separado.

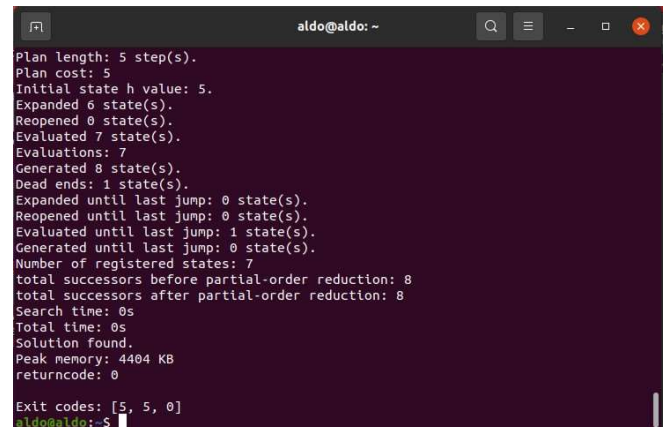
Esto puede conducir a una reducción exponencial del tamaño de la búsqueda-espacio. No siempre es fácil encontrar una partición para la tarea planificada dada, aun así, amplía el algoritmo STD (topología estrella) por una opción, que ejecuta una búsqueda estándar siempre que no se pudo encontrar ninguna partición (buena).



```
aldo@aldo: ~  
/pdds/pdb_heuristic.release.o .obj/pdds/util.release.o .obj/pdds/zero_one_pdds_h  
euristic.release.o .obj/pruning/stubborn_sets.release.o .obj/pruning/stubborn_se  
ts_decoupled.release.o .obj/pruning/stubborn_sets_simple.release.o .obj/pruning/  
stubborn_sets_ec.release.o .obj/leaf_state.release.o .obj/leaf_state_id.release.  
o .obj/./preprocess/scc.release.o .obj/compliant_paths/compliant_path_graph.rel  
ease.o .obj/compliant_paths/cpg_storage.release.o .obj/compliant_paths/effective  
_prices.release.o .obj/compliant_paths/frontier_prices.release.o .obj/compliant  
_paths/path_price_tag.release.o .obj/compliant_paths/pricing_function.release.o .  
obj/compliant_paths/pruning_options.release.o .obj/compliant_paths/pruning_reach  
able.release.o .obj/compliant_paths/reachability_function.release.o .obj/complia  
nt_paths/simulation_relation.release.o .obj/symmetries/graph_creator.release.o .  
obj/symmetries/permutation.release.o .obj/symmetries/group.release.o .obj/symmet  
ries/decoupled_permutation.release.o .obj/symmetries/decoupled_group.release.o .  
obj/symmetries/lexicographic_ordering.release.o -Wl,-Bstatic -lrt -lbtiss-0.72  
-Wl,-Bstatic -lbtiss -o downward-release  
Adding deffile section labels to container  
Adding runscript  
Finalizing Singularity container  
Calculating final size for metadata...  
Skipping checks  
Building Singularity image...  
Singularity container built: planner.img  
Cleaning up...  
aldo@aldo: ~$ RUNDIR="$(pwd)/rundir"  
Papetera
```

Figura 1.1

En la figura 1.1 podemos observar como en la terminal de Ubuntu 20.04.4 LTS, obtenemos la imagen del planificador para poder correr el algoritmo, así mismo de obtienen el dominio y el problema en cuestión para hacer las pruebas correctas.



```
aldo@aldo: ~  
Plan length: 5 step(s).  
Plan cost: 5  
Initial state h value: 5.  
Expanded 6 state(s).  
Reopened 0 state(s).  
Evaluated 7 state(s).  
Evaluations: 7  
Generated 8 state(s).  
Dead ends: 1 state(s).  
Expanded until last jump: 0 state(s).  
Reopened until last jump: 0 state(s).  
Evaluated until last jump: 1 state(s).  
Generated until last jump: 0 state(s).  
Number of registered states: 7  
total successors before partial-order reduction: 8  
total successors after partial-order reduction: 8  
Search time: 0s  
Total time: 0s  
Solution found.  
Peak memory: 4404 KB  
returncode: 0  
Exit codes: [5, 5, 0]  
aldo@aldo: ~$
```

Figura 1.2

En la figura 1.2 podemos comprobar que se corrió el planner de manera correcta.

Planificador Delfi1

La planificación rentable no ha visto tenido muchos enfoques o acercamientos exitosos que funcionen bien en todos los dominios. Algunos planificadores de costo óptimo sobresalen de algunos dominios, mientras exhiben desempeño menos emocionante en otros.

Para un dominio en partícula, sin embargo, a menudo existe un planificador rentable que funciona extremadamente bien, por esta razón, las técnicas basadas en **portafolios** se han popularizado recientemente. Estos deciden fuera de línea sobre un esquema en partícula de asignación de recursos para una colección dada de planificadores o tratar de realizar una clasificación en línea de una determinada tarea de planificación para seleccionar un planificador que se aplicara para resolver la tarea en cuetion.

El planificador Delfi es un planificador de **portafolio online**. En contraste con otras técnicas existentes, Delfi explota técnicas de **Deep learning** para aprender un modelo que predice cual de los planificadores en el portafolio puede resolver una tarea planificada dada, dentro de los limites de tiempo y memoria dada. Delfi usa representación graficas de una tarea planificada la cual permite explotar herramientas existentes para la convolución de la imagen.

```
ald@ald: ~  
Removing libisl15:amd64 (0.16.1-1) ...  
Removing libitm1:amd64 (5.4.0-6ubuntu1-16.04.12) ...  
Removing liblsan0:amd64 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libmpc3:amd64 (1.0.3-1) ...  
Removing libmpx0:amd64 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libquadmath0:amd64 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libtsan0:amd64 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libubsan0:amd64 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libx32asan2 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libx32atomic1 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libx32gcc1 (1:6.0.1-0ubuntu1) ...  
Removing libx32gomp1 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libx32ltm1 (5.4.0-6ubuntu1-16.04.12) ...  
Removing libc6-x32 (2.23-0ubuntu11.3) ...  
Processing triggers for libc-bin (2.23-0ubuntu11.3) ...  
Adding deffile section labels to container  
Adding runscript  
Finalizing Singularity container  
Calculating final size for metadata...  
Skipping checks  
Building Singularity image...  
Singularity container built: planner.img  
Cleaning up...  
ald@ald:~$
```

Figura 2.1

En la figura 2.1, podemos observar como en la terminal de Ubuntu 20.04.4 LTS, obtenemos la imagen del planificador para poder correr el algoritmo, así mismo de obtienen el dominio y el problema en cuestión para hacer las pruebas correctas.

```
ald@ald: ~  
nt: 215  
pattern: [80, 107, 108, 113, 114, 116, 117, 118, 119, 120, 121, 122] - improve  
nt: 120  
pattern: [80, 107, 108, 113, 115, 116, 117, 118, 119, 120, 121, 122] - improve  
nt: 44  
pattern: [80, 85, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improvemen  
t: 18  
pattern: [80, 86, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improvemen  
t: 12  
pattern: [80, 87, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improvemen  
t: 6  
pattern: [80, 88, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improvemen  
t: 5  
pattern: [80, 89, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improvemen  
t: 28  
pattern: [80, 97, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improvemen  
t: 6  
pattern: [80, 99, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improvemen  
t: 8  
pattern: [80, 100, 107, 108, 113, 116, 117, 118, 119, 120, 121, 122] - improve  
nt: 1  
Found a better pattern with improvement 215  
pattern: [80, 107, 108, 111, 113, 116, 117, 118, 119, 120, 121, 122]
```

Figura 2.2

En la figura 2.2 podemos comprobar que se corrió el planificador de manera correcta. Cabe destacar que este planificador tardo más de 2 hrs en obtener los resultados deseados.

MSP

(Meta-Search-Planner) es un sistema de meta-razonamiento que busca a través del espacio de planificadores, representaciones y heurísticas problema por problema. Dado un problema de planificación, MSP realiza dos fases: fase de meta-búsqueda y fase de resolución de problemas.

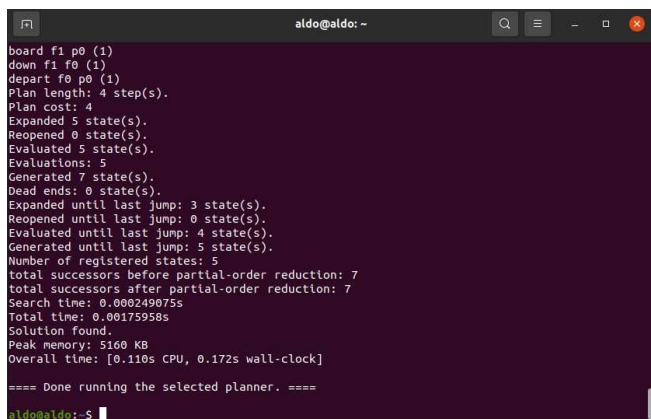
La fase de meta-búsqueda es un proceso de búsqueda que genera una combinación adecuada para resolver de manera optima el problema en específico. La fase de resolución de problema es básicamente una llamada al planificador seleccionado con otros elementos de la combinación seleccionada por el algoritmo.

Cabe señalar que en el trabajo presentado por MSP para su lectura, no describe una correcta estructura del algoritmo usado para la solución del problema ni el dominio usado para el planificador.

```
ald@ald: ~  
Exploding layer: sha256:fb15d46c38dcd1ea0b199006c3366ecd10c79d374f341687eb2cb23  
a2c8672e.tar.gz  
Exploding layer: sha256:c6a9ef4b9995d615851d7786fbc2fe72f72321bee1a87d66919b881a  
0336525a.tar.gz  
User defined %runscript found! Taking priority.  
Running post scriptlet  
+ apt update  
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [99.8 kB]  
Get:2 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]  
Get:3 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [205  
1 kB]  
Get:4 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [99.8 kB]  
Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [97.4 kB]  
Get:6 http://archive.ubuntu.com/ubuntu xenial/main amd64 Packages [1558 kB]  
Get:7 http://security.ubuntu.com/ubuntu xenial-security/restricted amd64 Packag  
es [15.9 kB]  
Get:8 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 Packages  
[984 kB]  
Get:9 http://archive.ubuntu.com/ubuntu xenial/restricted amd64 Packages [14.1 kB  
]  
Get:10 http://archive.ubuntu.com/ubuntu xenial/universe amd64 Packages [9827 kB]  
Get:11 http://security.ubuntu.com/ubuntu xenial-security/multiverse amd64 Packag  
es [8820 B]  
71% [10 Packages 8118 kB/9827 kB 83%] 454 kB/s 13s
```

Figura 3.1

En la figura 3.1, podemos observar como en la terminal de Ubuntu 20.04.4 LTS, obtenemos la imagen del planificador para poder correr el algoritmo, así mismo de obtienen el dominio y el problema en cuestión para hacer las pruebas correctas.



```
board f1 p0 (1)
down f1 f0 (1)
depart f0 p0 (1)
Plan length: 4 step(s).
Plan cost: 4
Expanded 5 state(s).
Reopened 0 state(s).
Evaluated 5 state(s).
Generated 7 state(s).
Dead ends: 0 state(s).
Expanded until last jump: 3 state(s).
Reopened until last jump: 0 state(s).
Evaluated until last jump: 4 state(s).
Generated until last jump: 5 state(s).
Number of registered states: 5
total successors before partial-order reduction: 7
total successors after partial-order reduction: 7
Search time: 0.00249075s
Total time: 0.00175958s
Solution found.
Peak memory: 5160 KB
Overall time: [0.110s CPU, 0.172s wall-clock]

==== Done running the selected planner. ====
aldo@aldo:~$
```

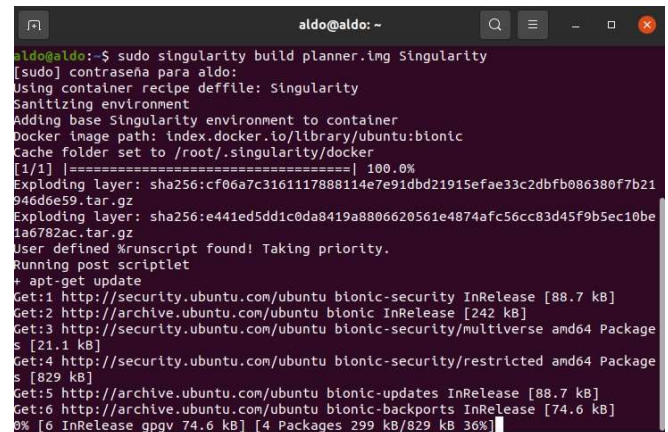
Figura 3.2

En la figura 3.2 podemos comprobar que se corrió el planificador de manera correcta.

Scorpion

El planificador esta implementado en el Sistema de planificación Fast Downward (Helmert 2006), Utiliza A*(Hart,Nilsson, and Raphael 1968) con la heurística admisible para crear planes óptimos. La heurística en general esta basada en componentes abstractos heurísticos que son combinados mediante la partición de costos saturados (Seipp y Helmert 2008) . En el trabajo que ellos presentan enlistan componentes del planificador y la configuración que ellos utilizaron para ellos.

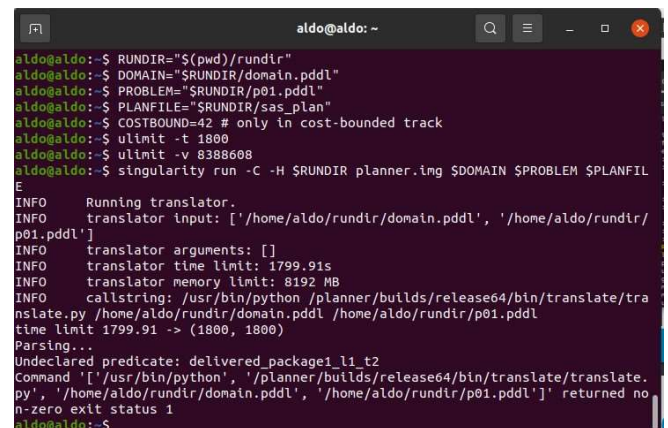
Cabe señalar que en el trabajo presentado por Scorpion para su lectura, no describe una correcta estructura del algoritmo usado para la solución del problema ni el dominio usado para el planificador.



```
aldo@aldo:~$ sudo singularity build planner.img Singularity
[sudo] contraseña para aldo:
Using container recipe deffile: Singularity
Sanitizing environment
Adding base Singularity environment to container
Docker image path: index.docker.io/library/ubuntu:bionic
Cache folder set to /root/.singularity/docker
[1/1] |=====| 100.0%
Exploting layer: sha256:cf06a7c3161117888114e7e91dbd21915efae33c2dbfb086380f7b21
946d6e59.tar.gz
Exploting layer: sha256:e441ed5dd1c0da8419a8806620561e4874afc56cc83d45f9b5ec10be
1a6782ac.tar.gz
User defined %runscript found! Taking priority.
Running post scriptlet
+ apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Package
s [21.1 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Package
s [829 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
0% [6 InRelease gpgv 74.6 kB] [4 Packages 299 kB/829 kB 36%]
```

Figura 4.1

En la figura 4.1, podemos observar como en la terminal de Ubuntu 20.04.4 LTS, obtenemos la imagen del planificador para poder correr el algoritmo, así mismo de obtienen el dominio y el problema en cuestión para hacer las pruebas correctas.



```
aldo@aldo:~$ RUNDIR="$(pwd)/rundir"
aldo@aldo:~$ DOMAIN="$RUNDIR/domain.pddl"
aldo@aldo:~$ PROBLEM="$RUNDIR/p01.pddl"
aldo@aldo:~$ PLANFILE="$RUNDIR/sas_plan"
aldo@aldo:~$ COSTBOUND=42 # only in cost-bounded track
aldo@aldo:~$ ulimit -t 1800
aldo@aldo:~$ ulimit -v 8388608
aldo@aldo:~$ singularity run -C -H $RUNDIR planner.img $DOMAIN $PROBLEM $PLANFIL
E
INFO      Running translator.
INFO      translator input: ['/home/aldo/rundir/domain.pddl', '/home/aldo/rundir/
p01.pddl']
INFO      translator arguments: []
INFO      translator time limit: 1799.91s
INFO      translator memory limit: 8192 MB
INFO      callstring: /usr/bin/python /planner/builds/release64/bin/translate/tra
nslate.py /home/aldo/rundir/domain.pddl /home/aldo/rundir/p01.pddl
time limit 1799.91 -> (1800, 1800)
Parsing...
Undeclared predicate: delivered_package1_l1 t2
Command '['/usr/bin/python', '/planner/builds/release64/bin/translate/translate.
py', '/home/aldo/rundir/domain.pddl', '/home/aldo/rundir/p01.pddl']' returned no
n-zero exit status 1
aldo@aldo:~$
```

Figura 4.2

En la figura 4.2 podemos comprobar que se corrió el planificador de manera correcta.

Conclusión

Este trabajo muestra los resultados obtenidos de ejecutar 4 planificadores obtenidos de (<http://icaps-conference.org/index.php/Main/Competitions>) para sus competencias y conferencias hechas en 2018. Todos los planificadores fueron ejecutados de forma correcta, con la sorpresa que algunos de ellos tardaban horas en crear la imagen del planificador, o incluso realizar la tarea de búsqueda con el dominio y el problema.

Cada uno de los planificadores usan algoritmos (aunque algunos trabajos no describen el algoritmo usado, pero si su uso en específico) diferentes para la resolución de problemas creados por la misma ICAPS para su competencia, desde algoritmo con topología estrella (haciendo referencia a lo parecido que es con redes al momento de graficarlo), hasta el uso de portafolios, donde se almacenan diferentes planeadores y el agente de IA decide cual es el más correcto para su solución después de analizar el problema obtenido.

Anexos

Instalación de Singularity

Se obtuvo singularity usando la herramienta:

https://neuro.debian.net/install_pkg.html?p=singularity-container

donde facilita todo para la creación del comando para la consola de Linux. (Figura A). Una vez decidido se empiezan a instalar repositorios (Figura B), descargados los repositorios empezamos a bajar los archivos de la aplicación Singularity (Figura C)

La figura D muestra Singularity instalado

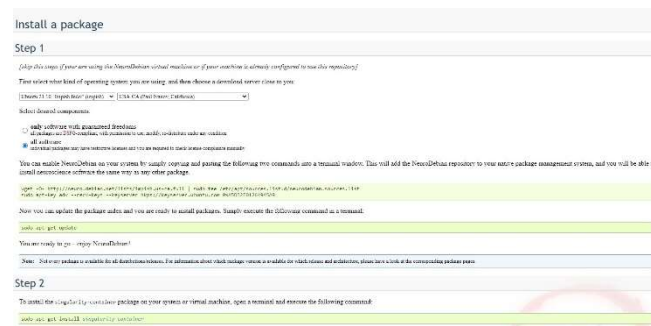


Figura A

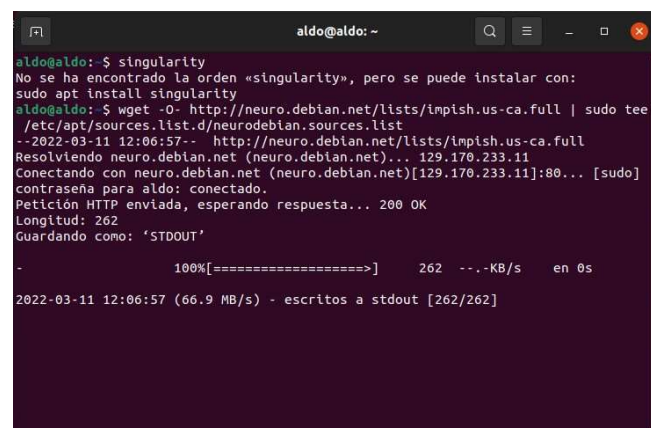


Figura B

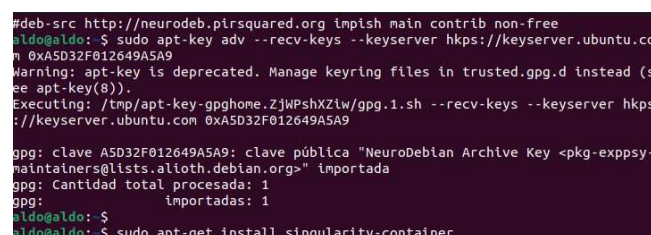


Figura C

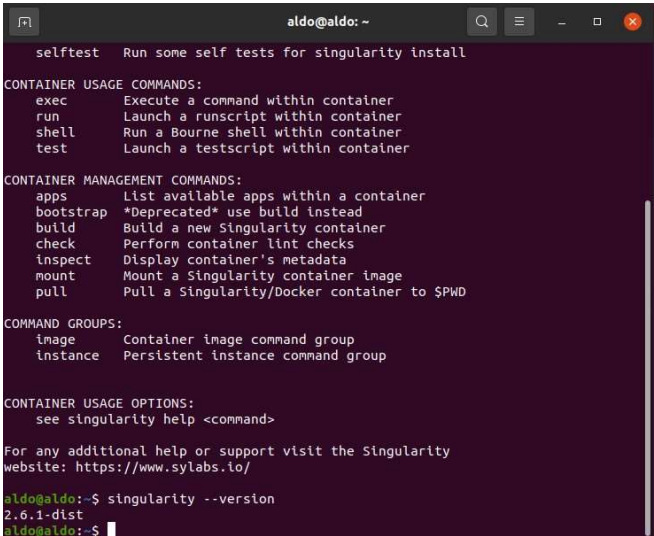


Figura D

Instalando Ubuntu 20.04.4 LTS y VirtualBox 6.1

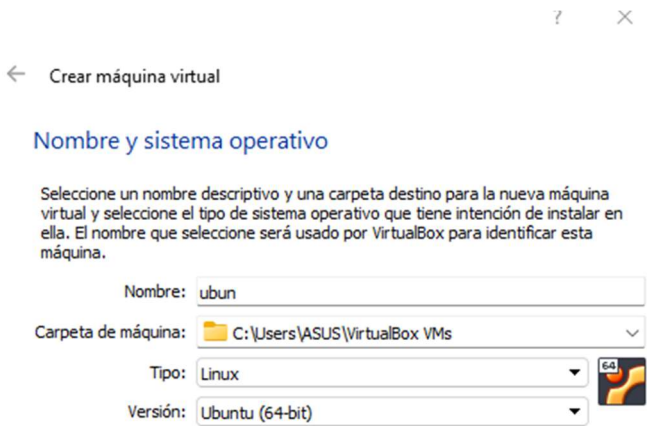


Figura E

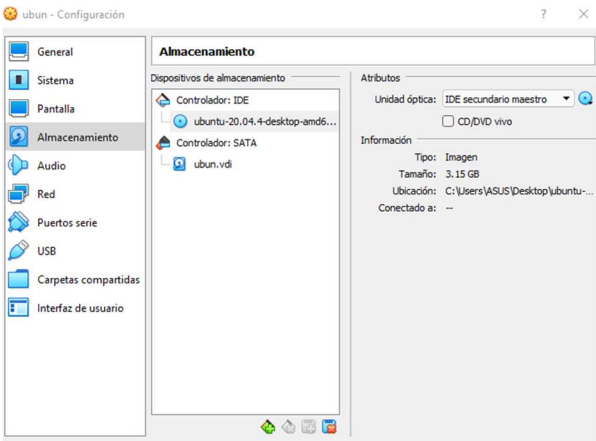


Figura F

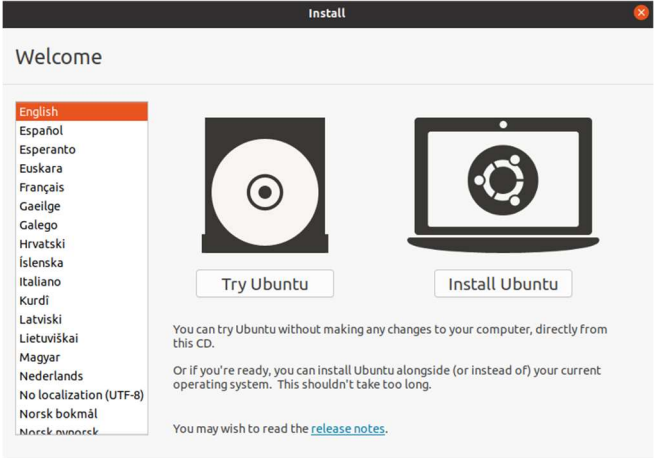


Figura G

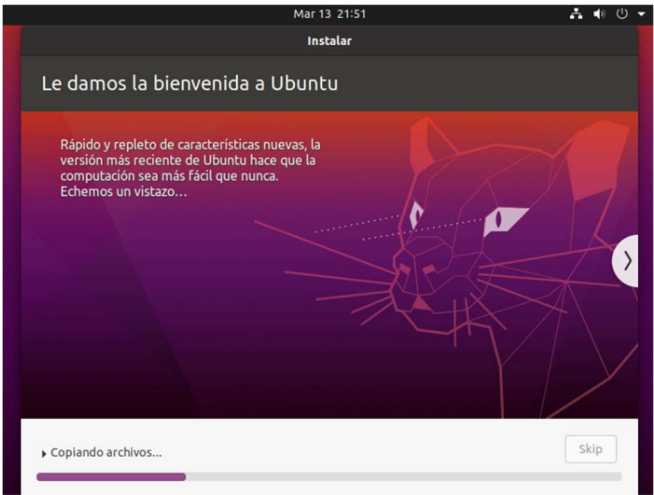


Figura H

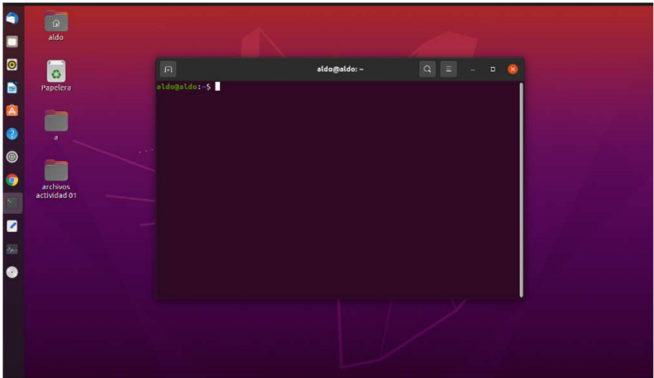


Figura I

Referencias:

Wikipedia contributors. (2022, 2 marzo). Automated planning and scheduling. Wikipedia. https://en.wikipedia.org/wiki/Automated_planning_and_scheduling

Tema: Planificación en Inteligencia Artificial. Agentes Planificadores. (s. f.). Facultad: Ingeniería Escuela: Computación Asignatura: Sistemas Expertos e Inteligencia Artificial. Recuperado 13 de marzo de 2022, de <https://docplayer.es/91429912-Tema-planificacion-en-inteligencia-artificial-agentes-planificadores.html>

Javier Vázquez-Salceda. (2011). Inteligencia Artificial. Facultad de Informática de Barcelona - Agentes planificadores. <https://www.fib.upc.edu/es/estudios/grados/grado-en-ingenieria-informatica/plan-de-estudios/asignaturas/IA>