Estado del arte de planificadores en Inteligencia Artificial

Edgar Uriel Domínguez Espinoza*

10 de marzo de 2022

Resumen

Este artículo presenta brevemente la historia de los planificadores y reseña cuatro planificadores que participaron en la competencia IPC 2018. Además se realiza un ejercicio de ejecución de dichos planificadores y se brinda el resultado considerando que las versiones probadas tienen cuatro años de antigüedad.

Las evidencias completas del trabajo están disponibles libremente en la carpeta `planners` del siguiente repositorio: `https://gitlab.com/genomorro/unir/-/tree/RPA-01/`.

Palabras clave: planificadores, MAPlan, Scopion, Metis, IBaCoP

1. Introducción

La planificación autónoma, también adjetivada *automática*, trata de resolver un problema en el cual deben enunciarse una serie de acciones secuenciales que permitirán ir de un estado inicial a un estado objetivo al momento de realizarse (Tapia García, 2017). En el libro *Automated Planning: Theory and Practice* de Ghallab et al. (2004, cap. 1) como primera definición se escribe: «La planificación es la parte racional de actuar. Es un proceso de deliberación abstracto y explícito que elige y organiza acciones anticipando sus resultados esperados. Esta deliberación tiene como propósito lograr lo mejor posible algunos objetivos prestablecidos. La planificación automatizada es un área de Inteligencia Artificial (AI) que estudia este proceso de deliberación computacionalmente»[1].

Los primeros planificadores se basaban en demostraciones de teoremas y lógica de predicados, eran conocidos como *General Problem Solver* (GPS). Estos planificadores consideraban entornos finitos, deterministas, estáticos para generar planes secuenciales que no tienen en cuenta el tiempo, con una fase deliberativa y otra de ejecución. (Tapia García, 2017)

En 1971, Richard Fikes y Nils Nilsson crean el planificador STRIPS, basado en lógica de primer orden. Actualmente, STRIPS es también el nombre del lenguaje que este planificador usa para definir un problema concreto a resolver. Este lenguaje es usado por programas como *Graphplan*, el cual construye un grafo con acciones y predicados para obtener un plan. (Tapia García, 2017)

Posteriormente se crea la técnica HTN, donde se descompone un sistema en distintas tareas jerarquizadas hasta llegar a primitivas que posteriormente se usarán para crear el plan. También se crean los llamados *SAT solvers*; en esta lógica, se asignan al problema valores de verdad en una expresión proposicional, el planificador asegura el resultado al crear un plan cuyo resultado tenga un valor verda-

*Universidad de la Rioja en México

[1]Traducción libre.

dero. (Tapia García, 2017)

Los planificadores heurísticos tomaron popularidad porque mejoraban notablemente el tiempo respecto a Graphplan. Estos se centran en una función heurística independiente del dominio por lo que pueden ser de propósito general y resolver problemas más complejos. (Tapia García, 2017)

Actualmente, destacan planificadores basados en las técnicas antes mencionadas y otras como: *Madagascar*, basada en SAT, y *Fast Downward*, un planificador heurístico con una fuerte conexión con SAT. De hecho, es posible decir que se han creado portafolios de planificadores para resolver problemas, es decir, ejecutar varios planificadores en paralelo. En este sentido se toman en cuenta las características de cada planificador para conseguir un mejor rendimiento en distintos problemas. (Tapia García, 2017)

En el presente texto se estudian algunos de los planificadores presentes en la competición IPC 2018, se sintetiza su funcionamiento y se ejemplifica su ejecución, con el propósito de entenderlos *grosso modo*.

2. Marco teórico

2.1. Scopion

El planificador Scopion (Seipp, 2018) utiliza la técnica Fast Downward. Se complementa con un componente heurístico de abstracción de componentes que se combinan con una partición saturada de costos. El componente heurístico tendrá variaciones dependiendo si la tarea tiene efectos condicionales.

Cuando se afrontan tareas con efectos condicionales se crea una base de datos sistemáticamente con patrones de distintos tamaños, esta generación está limitada en tiempo debido a que es una labor que puede tardar mucho tiempo para algunas tareas.

En el caso de tareas sin efectos condicionales se usa una combinación de distintas técnicas: Abstracciones cartesianas (CARS), base de datos con algoritmo de búsqueda escalada simple (HC) y base de datos para patrones sistemáticos.

El resultado de estas componentes heurísticas es procesado por la partición saturada de costos. Esta asigna iterativamente a cada heurística los costos necesarios para justificar sus estimaciones y así ahorrar costos para las heurísticas posteriores, así se distribuyen los costos.

Finalmente, este planificador somete la información generada a un proceso de *pruning* o poda, en el cual se eliminan datos redundantes, reduce la complejidad y mejora la precisión del resultado.

2.2. MAPlan

El planificador multiagente MAPlan (Fiser and Komenda, 2018) usa una búsqueda heurística basada en estados con un traductor de PDDL a STRIPS y posteriormente a una representación de dominio finito. El traductor trata de reducir el problema usando algoritmos de exclusión mutua (mutex) en procesos heurísticos de regresión y progresión. Este planificador tiene dos configuraciones distintas, aunque solo difieren en la heurística: maplan-1 utiliza la heurística LM-Cut; maplan-2 utiliza una heurística de abstracción simplificada.

El algoritmo de grupos mutex se implementa en el planificador MAPlan utilizando el solucionador CPLEX, principalmente se usa en la traducción STRIPS. Por otra parte, hay grupos mutex alternativos llamados fam-groups, que constituyen uno de los métodos usados para reducir los insumos del problema de planificación. Otra de las aplicaciones de estos grupos es su participación en el proceso de desambiguación que «es un proceso simple que extiende un conjunto de hechos donde un hecho es la

única posibilidad dada el conjunto, que fue formado exactamente con un hecho de cada estado posible»[2] (Fiser and Komenda, 2018).

En la búsqueda heurística se calcula un grafo de accesibilidad con hechos individuales y hechos en pares. Es un método ampliamente conocido, sin embargo, el costo de operación aumenta exponencialmente, por lo que solo se utiliza en potencias 1 y 2. Dicho grafo se someterá a un proceso de poda que funciona a su vez en progresión y regresión, elimina elementos inalcanzables hasta llegar a la estabilidad.

El planificador admite diversos espacios de mejora, tanto en procesamiento como en optimización de recursos, por lo que se considera un trabajo en proceso. La versión entregada para IPC, si bien es estable y funcional, podría obtener mejores estimaciones en el futuro, al tiempo que requiera menos recursos computacionales.

2.3. Metis

Metis (Sievers and Katz, 2018) es una reimplementación de un planificador que participó en el IPC 2014. Sus componentes básicos son un proceso heurístico y dos procesos de poda.

El artículo original trata sobre las diferencias en la implementación de ambas versiones de Metis, por lo que no ofrece gran detalle sobre su funcionamiento actual. Es posible mencionar que la versión actual usa la técnica Fast Downward que mejora su eficiencia respecto a la versión original. Al igual que MAPlan, este planificador usa algoritmos de exclusión mutua para un proceso de poda.

De forma más particular, Metis usa un algoritmo de heurística admisible LM-cut y una heurística histórica con un método LAMA; un proceso de poda basado en simetrías estructurales[3] y otro proceso de poda basado en la reducción parcial del orden. El proceso de poda se desactiva después de las primeras mil expansiones si menos del uno por ciento de los estados han sido podados en este punto.

2.4. IBaCoP

El documento de IBaCoP (Cenamor et al., 2018) describe dos portafolios de planificadores. Los portafolios fueron integrados gracias a la técnica de preselección, la cual toma en cuenta varios criterios como por ejemplo: el tiempo de la primera solución y la calidad de la mejor solución. Se modela el rendimiento del planificador con el propósito de predecir el comportamiento del portafolio en función de las necesidades y funciones del problema.

IBaCoP es producto de una configuración estática llamada preselección de Pareto, mientras que IBaCoP2 es el resultado de una configuración dinámica, los datos de la primera versión se utilizaron como parte de los datos de capacitación y se incluyeron nuevos planificadores. Los planificadores elegidos como parte de IBaCoP son: Jasper, mercurio, BFS(F), SIW, FDSS-2, sonda, yashp2-mt, lama-2011, lamar y arvand.

El portafolio usa el algoritmo Rotation Forrest para entrenar un modelo predictivo de clasificación que trata de codificar que planificador resolverá el problema. Si hay más de un candidato, se creará una lista de cinco planificadores base con la mejor confianza de predicción. Si los planificadores listados fallan los planificadores lama-2011, lamar y arvand son llamados para cubrir el tiempo disponible entre cada planificador.

3. Marco referencial

Para probar los planificadores revisados en este texto se usó una distribución *rolling-release* de un

[2] Traducción libre.

[3] Suponemos que aquí se usan los algoritmos mutex.

sistema operativo GNU/Linux. El proceso de instalación puede verse de forma detallada en el Handbook de Gentoo. La información general del sistema operativo se puede observar con el siguiente comando:

```
uname -a ; grep MemTotal /proc/meminfo
```

```
Linux genomorro-16t90p 5.15.23-gentoo-dist #1 SMP Tue Feb 15 02:38:43 CST 2022 x86_64 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz GenuineIntel GNU/Linux
MemTotal:       16182448 kB
```

La instalación de *Singularity* se llevó a cabo mediante el gestor de paquetes Portage, se usó el código fuente de la versión 3.8.5, tal como se muestra enseguida:

```
emerge -avt sys-cluster/singularity
```

```
These are the packages that would be merged, in reverse order:

Calculating dependencies  ... done!
[ebuild   R    ] sys-cluster/singularity-3.8.5::gentoo  USE="network suid -examples" 7714 KiB

Total: 1 package (1 reinstall), Size of downloads: 7714 KiB
```

4. Metodología

Se ejecutaron los planificadores uno a uno, se usó el mismo procedimiento y los mismos archivos para el dominio y el problema. Se escribió un script general que permite variar con facilidad los parámetros de ejecución de los distintos planificadores para aquellos casos que requieran mayor experimentación. En seguida se muestran las opciones generales de dicho script.

```
./frontend.bash -h
```

```
Help message
-P  Set Plan [output] file
-c  Set costbound, only if planner need it
-d  Set domain file, default domain.pddl
-i  Set image file, default singularity.img
-m  Set memory available for Singularity un bytes, default 12582912 (12GB)
-p  Set problem file, default problem.pddl
-r  Set runtime directory, default .
-s  Set Singularity file script, default Singularity
-t  Set max execution time in seconds, default 7200 (2hr)
```

Para fines prácticos, el script, los archivos *pddl* con el dominio y el problema se encontrarán en el mismo directorio junto con los cuatro archivos de definición de Singularity (Sylabs Inc. and Project Contributors, 2021, sec. Interact with images). Dentro de este directorio se crearán, eventualmente, los archivos de trabajo de cada planificador. El script volcará `stdout` y `stderr` a dos archivos para una consulta detallada. A continuación se presenta la estructura inicial de directorios:

```
ls -R
```

```
.:
total 16K
4.0K ibacop/  4.0K maplan/  4.0K metis/  4.0K scorpion/

./ibacop:
total 44K
 16K domain.pddl  4.0K frontend.bash*  4.0K ibacop1  4.0K ibacop2   16K problem.pddl

./maplan:
total 44K
 16K domain.pddl  4.0K frontend.bash*  4.0K maplan-1  4.0K maplan-2   16K problem.pddl

./metis:
total 40K
 16K domain.pddl  4.0K frontend.bash*  4.0K metis   16K problem.pddl

./scorpion:
```

```
total 40K
 16K domain.pddl  4.0K frontend.bash*   16K
     problem.pddl  4.0K scorpion
```

La ejecución más básica es:

```
time ./frontend.bash
```

El comando `time` permitirá saber con exactitud cuanto tiempo estuvo en ejecución el script, incluyendo la creación de la imagen del contenedor.

Finalmente, si los archivos argumentos del script están en una carpeta distinta, es necesario proporcionar rutas absolutas de los mismos.

5. Resultado

En esta sección se muestran algunos mensajes destacados al ejecutar los planificadores inspeccionados en este documento. Debido a la longitud de los archivos y mensajes producidos es imposible mostrar todo su contenido, así como mencionarlos exhaustivamente uno a uno, sin embargo, se hacen comentarios pertinentes que brindan un mejor contexto.

Si se desea verificar los resultados completamente, cada uno de los *logs* están resguardados en la carpeta `planners` del repositorio del proyecto.

5.1. Scorpion

La ejecución del planificador Scorpion produce un archivo `output.sas`.

```
head output.sas
```

```
begin_version
3
end_version
begin_metric
1
end_metric
96
begin_variable
var0
-1
```

El planificador falló en su ejecución debido a falta de memoria, `stdout` termina con el siguiente mensaje:

```
f = 2585 [172375196 evaluated, 114159035
    expanded, t=1971.26s, 9228416 KB]
f = 2586 [178752564 evaluated, 116580139
    expanded, t=2031.4s, 9493964 KB]
Failed to allocate memory.
Memory limit has been reached.
Peak memory: 10706644 KB
exitcode: 6

Command '['run-portfolio', '/planner/driver/
    portfolios/seq_opt_scorpion.py']' returned
    non-zero exit status 6
```

El tiempo de ejecución del planificador fue la mitad del límite establecido en el script mediante el comando `ulimit`.

```
real    44m36.331s
user    47m53.699s
sys     0m34.293s
```

5.2. MAPlan

Tal como se mencionó en la sección 2.2, este planificador tiene como dependencia un producto de IBM llamado CPLEX, sin embargo, no es software libre ni de distribución gratuita. Como consecuencia, la imagen de Singularity no puede crearse. En seguida se presenta el final de `stderr`:

```
INFO:    Copying cplex_studio12
   .7.1.linux-x86-64.bin to /third-
   party/
FATAL:   While performing build:
   unable to copy files from host
   to container fs: while copying [
   cplex_studio12.7.1.linux-x86-64.
   bin] to /tmp/build-temp
   -3960486677/rootfs/third-party/:
```

```
     [-fLr cplex_studio12.7.1.linux-
    x86-64.bin /tmp/build-temp
    -3960486677/rootfs/third-party
    /]: /bin/cp: no se puede
    efectuar `stat' sobre '
    cplex_studio12.7.1.linux-x86-64.
    bin': No existe el fichero o el
    directorio
FATAL:   could not open image /home
    /genomorro/School/UNIR/
    RazonamientoPlanificacion/
    planners/maplan/singularity.img:
     failed to retrieve path for /
    home/genomorro/School/UNIR/
    RazonamientoPlanificacion/
    planners/maplan/singularity.img:
     lstat /home/genomorro/School/
    UNIR/RazonamientoPlanificacion/
    planners/maplan/singularity.img:
     no such file or directory
```

Al día en que se ha redactado este artículo (10 de marzo de 2022), en la página de IBM esta disponible, previo registro, el archivo de instalación de CPLEX: `cos_installer_preview-20.1.0.0.R1-CC8CWML-linux-x86-64.bin`. Se trató de buscar dentro de la instalación de CPLEX el archivo solicitado, también se intentó proporcionar el instalador completo, en ambos casos se obtuvo el mismo resultado:

```
cc -O3 -Wall -pedantic -std=gnu99 -ffast-math -I
    . -I/opt/ibm/ILOG/CPLEX_Studio1271/cplex/
    include -c -o .objs/lp-cplex.o src/lp-cplex.
    c
Makefile:239: recipe for target '.objs/lp-cplex.
    o' failed
make: Leaving directory '/planner/boruvka'
```

5.3. Metis

Al igual que en el planificador Scorpion, el presente genera un archivo `output.sas`, enseguida se presenta el final de dicho archivo.

```
tail output.sas
```

```
89 5718
91 12483
90 12423
92 12525
93 41061
3
11 1
22 1
39 1
end_CG
```

Por su parte, la salida de `stdout` deja saber que ha fallado la ejecución, igualmente por alcanzar el límite de memoria.

```
Level 40 completed.
h^m landmarks computed.
Removed 0 reasonable or obedient reasonable
    orders
Calculating achievers.
Landmarks generation time: 1.1888s
Discovered 46 landmarks, of which 0 are
    disjunctive and 0 are conjunctive
119 edges
Adding simple landmarks
Adding disjunctive landmarks
Adding orderings
Removed 0 reasonable or obedient reasonable
    orders
Landmarks generation time: 2.5835s
Discovered 83 landmarks, of which 0 are
    disjunctive and 0 are conjunctive
831 edges
Initializing symmetries (eager search)
Initializing symmetries
Using Bliss to find group generators
Done initializing symmetries: 6.20247s
Number of generators: 12
Number of identity generators (on states, not on
     operators): 29518
Order of generators: [2, 2, 2, 2, 2, 2, 2, 2, 2,
     2, 2, 2]
Setting group in registry for DKS search
Conducting best first search with reopening
    closed nodes, (real) bound = 2147483647
43 initial landmarks, 1 goal landmarks
New best heuristic value for lmcount(lm_factory
    = lm_merged(list(lm_rhw, lm_hm(m = 1))),
    admissible = true, transform =
    multiply_out_conditional_effects): 395
New best heuristic value for celmcut: 395
```

```
[g=0, 1 evaluated, 0 expanded, t=9.17426s, 157580 KB]
f = 395 [1 evaluated, 0 expanded, t=9.17426s, 157580 KB]
Initial heuristic value for lmcount(lm_factory = lm_merged(list(lm_rhw, lm_hm(m = 1))), admissible = true, transform = multiply_out_conditional_effects): 395
Initial heuristic value for celmcut: 395
Failed to allocate memory.
Memory limit has been reached.
Peak memory: 7929624 KB
```

El `stderr` solo tiene información de la compilación, nada que brinde indicios sobre este problema recurrente de memoria. Es importante notar que este planificador se ejecuta muy pocos minutos.

```
real    6m2.266s
user    8m59.197s
sys     0m23.746s
```

5.4. IBaCoP

En el caso de este portafolio se obtienen varios archivos temporales de gran tamaño. Podemos suponer que la creación de estos archivos corresponden a los diversos planificadores que IBaCoP ejecuta. Es de relevancia el archivo `outputModel` y `listPlanner`, en ellos se puede ver como el portafolio ha seleccionado cinco planificadores. El archivo `listPlanner` contiene la lista de los planificadores cuya columna "predicted" tiene valor "True". el contenido integro de `outputModel` es el siguiente:

```
cat outputModel
```

```
=== Predictions on test data ===

 inst#     actual  predicted error prediction (planner)
     1     1:True     1:True       0.714 (fdss-2)
     2     1:True     1:True       0.788 (probe)
     3     1:True    2:False   +   0.517 (yahsp2-mt)
     4     1:True     1:True       0.559 (mercury)
     5     1:True    2:False   +   0.504 (jasper)
     6     1:True     1:True       0.865 (siw)
     7     1:True     1:True       0.765 (bfs-f)
```

Adicionalmente, es importante leer de `stdout` que el planificador falla, en esta ocasión por un problema propio del programa y no de la memoria del sistema:

```
*********************************************
*** Planner_path: /planner/src/siw/plan TimeOut: 838 ***
***************************************** --- OK.
 Match tree built with 0 nodes.

PDDL problem description loaded:
        Domain: AGRICOLA
        Problem: SAT21-3-10
        #Actions: 0
        #Fluents: 214
Landmarks found: 1
Starting search with IW (time budget is 60 secs) ...
Try allocate size: 0.00167188 MB
Try allocate size: 0.00167188 MB

Caption
{#goals, #UNnachieved,  #Achieved} -> IW(max_w)

{1/1/0}:IW(1) -> ;; NOT I-REACHABLE ;;
Total time: 1.79928e-05
Nodes generated during search: 1
Nodes expanded during search: 1
IW search completed in 1.79928e-05 secs
********


Name: /home/genomorro/School/UNIR/RazonamientoPlanificacion/planners/ibacop/cleaned_result.result
ERROR: Plan /home/genomorro/School/UNIR/RazonamientoPlanificacion/planners/ibacop/cleaned_result.result is not valid or the plan cost is equal to -1, therefore we remove it
Planner /planner/src/siw/plan run 5 seconds

Main portfolio plus default planner run 1822.0 seconds (in total)

Main portfolio runs 1822.0 seconds

-------------------- END --------------------
```

Este portafolio de planificadores se ha ejecutado por cerca de una hora:

```
real    59m46.981s
user    51m28.854s
sys     1m21.011s
```

6. Conclusión

En este artículo se hizo una breve reseña sobre el desarrollo de la planificación autómata, desde su definición hasta distintos planificadores y técnicas empleadas a lo largo de su historia. Posteriormente se estudiaron cuatro planificadores que fueron parte de la competencia IPC 2018.

Los mismos planificadores fueron ejecutados bajo condiciones iguales y se reportan los resultados del ejercicio. Lamentablemente, ninguno de los planificadores seleccionados terminó exitosamente su ejecución. Esto puede deberse a los archivos de entrada seleccionados, los cuales pudieron ser escogidos de manera inadecuada (al azar). Este punto de vista es cuestionable porque ninguno de los planificadores contiene una documentación o archivo README adecuado que ilustre su ejecución, aunque sea idealizada. Si bien se ejecutaron usando un contenedor Singularity, lo cual garantiza un entorno de ejecución correcto, también se sabe debido a los artículos leídos que no son desarrollos terminados, por lo cual pueden contener errores.

Un error grave en el planificador MAPlan es que tiene dependencias de compilación que el usuario debe proporcionar y, a cuatro años de distancia, ya no están disponibles pues se trata de software privativo. Este problema elimina la ventaja del uso de un contenedor Singularity que pretende replicar un entorno adecuado de ejecución. En este sentido se considera que el código fuente probado de este planificador es obsoleto.

Será necesario, como trabajo a futuro, revisar el código fuente y los cambios que han tenido los planificadores, con el propósito de saber si versiones más actuales tienen ejecuciones exitosas. Asimismo, será necesario variar los parámetros de ejecución de los contenedores, tanto en tiempo como en asignación de memoria; es posible que con los parámetros adecuados el planificador logré terminar exitosamente su proceso de ejecución.

Referencias

Cenamor, I., de la Rosa, T., and Fernández, F. (2018). IBaCoP-2018 and IBaCoP2-2018. *Ninth International Planning Competition (IPC 2018)*.

Fiser, D. and Komenda, A. (2018). MAPlan: Reductions with fact-alternating mutex groups and hm heuristics (planner abstract). *Ninth International Planning Competition (IPC 2018)*.

Gentoo Foundation, I. (2021). Manual:página principal.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Elsevier.

Seipp, J. (2018). Scorpion (planner abstract). *Ninth International Planning Competition (IPC 2018)*.

Sievers, S. and Katz, M. (2018). Metis. *Ninth International Planning Competition (IPC 2018)*.

Sylabs Inc. and Project Contributors (2021). Singularityce user guide — singularityce user guide 3.8 documentation.

Tapia García, C. (2017). *Diseño e implementación de un planificador para un agente autónomo*. PhD thesis, E.T.S.I. Industriales (UPM).