Heurística y Optimización

Grado de Ingeniería en Informática. Curso 2022-2023

Práctica 2. Satisfacción de Restricciones y Búsqueda Heurística

Link al repositorio de GitHub de la práctica:
Práctica 2 - 100451170 - 100451258 - Repositorio de Github

Práctica realizada por los alumnos:
Carlos Iborra Llopis: 100451170@alumnos.uc3m.es
Miguel Domínguez Gómez: 100451258@alumnos.uc3m.es

1. Tabla de Contenidos

2. Introducción

La memoria de la práctica 2 de Heurística y optimización se ha decidido orientar de la siguiente manera:
Primero, explicaremos nuestras implementaciones y observaciones sobre ambos modelos:

- En el modelo de la parte 1 se hablará sobre nuestra interpretación del problema y sobre las variables, restricciones y dominios que se han conseguido sacar a partir de esta interpretación.
- En el modelo de la parte 2, se hará hincapié en nuestro planteamiento del problema (estados, costes y restricciones), las distintas distintas heurísticas implementadas y en el diseño de nuestra implementación.

Tras los modelos, tendrá lugar el análisis de los resultados obtenidos, dónde se explicarán los distintos resultados extraídos de las ejecuciones, el por qué de estos resultados, si son los esperados y una comparación de las heurísticas (en el apartado de la parte 2), comparándolas entre ellas para poder sacar unas conclusiones fehacientes sobre sus distintos rendimientos y las razones de estos.

Por último, se sacarán conclusiones sobre esta práctica dónde comentaremos si nos ha ayudado a comprender mejor lo visto en clase, y haremos alguna sugerencia y/o crítica sobre el planteamiento de esta.

3. Modelo Parte 1

Para empezar con la modelización, primero se han de obtener los datos más relevantes presentes en el enunciado (tanto implícitamente como explícitamente). Se han destacado los siguientes:

Se dispone de un tablero que representa los **asientos de un autobús**, estando estos numerados **del 1 al 32**:

- Los **asientos 1, 2, 3, 4, 13, 14, 15, 16, 17, 18, 19 y 20** están **destinados a alumnos con movilidad reducida**.
- Existe un pasillo separando en dos el autobús, dejando dos filas de asientos a la derecha y dos a la izquierda.
  Nota: este pasillo no estará explícitamente definido en los dominios o variables, pero se tendrá en cuenta su existencia a la hora de elaborar las restricciones y el código.
- Al igual que con el pasillo, existirán espacios del autobús reservados para las puertas del medio del autobús (que marcan el cambio de ciclo), delante a la izquierda, un espacio reservado para el chófer y a la derecha otra puerta. Nota: estos se tendrán en cuenta a la hora de elaborar las restricciones y el código.
- Los **asientos** de los alumnos del **primer ciclo** van del número **1 al 16**.
- Los **asientos** destinados a los alumnos del **segundo ciclo** van del número **17 al 32**.

Por otro lado, se dispone de unos **alumnos** que poseen **características** diferenciadoras:

- **Ciclo**: alumnos del **primer ciclo o** alumnos del **segundo ciclo**
- **Movilidad**: alumnos con **movilidad reducida o no**
- **Conflictividad**: alumnos **conflictivos o no**
- **Hermanos**: **pareja de alumnos** que son hermanos el uno del otro

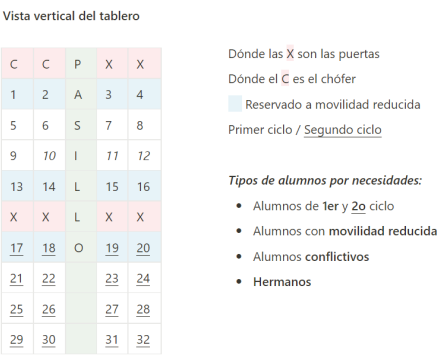
Además de lo anterior, se no dan ciertos **requisitos** que se han de cumplir:

1. Ha de haber **un solo alumno por asiento**.
2. Si hay alumnos con **movilidad reducida**, estos tendrán que sentarse en los **asientos reservados** para los alumnos de movilidad reducida, **siendo imposible ocupar** el **asiento** inmediatamente **a su lado**.
3. **Si no están ocupados** los **asientos reservados** para alumnos con movilidad reducida, estos asientos podrán ser **ocupados por otros alumnos**.
4. Los alumnos **conflictivos no** pueden sentarse en la **periferia de otros alumnos conflictivos ni con movilidad reducida**. *Por ejemplo: un alumno conflictivo en el asiento 23 implica que no se podrá sentar en los asientos 18, 19, 20, 22, 23, 26, 27, y 28 ningún otro alumno conflictivo.*
5. Los alumnos del **primer ciclo** se deben sentar en la **parte delantera** del autobús (1 al 16) y los de **segundo ciclo** en la **parte trasera** del autobús (17 al 32)
6. Si dos alumnos son **hermanos**, deberán sentarse **uno inmediatamente al lado del otro** (**no les puede separar un pasillo**). Además, el ser hermanos, implica ciertas **excepciones sobre los requisitos anteriores**:
   - Si los **hermanos pertenecen a ciclos distintos**, **ambos** se sentarán en los asientos del **ciclo 1**. En este caso, el **hermano mayor ocupará el asiento más cercano al pasillo**.
   - Si **dos hermanos son conflictivos**, han de **sentarse juntos** pero se conserva la propiedad de que ningún otro conflictivo (o alumno con movilidad reducida) pueda sentarse alrededor de ellos.
   - Si **hay un hermano con movilidad reducida**, entonces **no** haría falta que los hermanos se sentasen **juntos**, **pero sí** en la **misma parte del autobús** (delantera para el ciclo 1, trasera para el ciclo 2).

3.1. Interpretación del problema

Tras hacer un análisis de lo expuesto anteriormente, se puede inferir que **los alumnos serán las variables** y que, además, las diferentes **características** de cada alumno **serán parte de esas variables**.
También podemos inferir que, lo que se nos presenta como **requisitos**, serán necesariamente **restricciones** a nuestro problema.

Aclaración: para hacer más intuitivo el tablero o esquema de los asientos del autobús, se ha decidido representarlo verticalmente (como se muestra en la imágen de la derecha), aunque posteriormente a nivel de codificación, se parsearán los resultados para que estos tengan finalmente la distribución presentada en el enunciado.



3.2. Variables

Tras lo expuesto previamente, se ha determinado que lo ideal sería tomar por variables los distintos alumnos, que tendrán el nombre compuesto de la siguiente manera: ***'[A][1,2][X,C][X,R][B]'***, dónde el primer dígito (*A*) indica el número identificativo (ID) del alumno, el segundo dígito indica si el alumno pertenece al ciclo 1 o al ciclo 2, el tercer dígito es una letra que indica si el alumno es conflictivo (*C*) o no (*X*), el cuarto dígito indica si el alumno tiene movilidad reducida (*R*) o no *(X*), y el último dígito (*B*) es un número que hace referencia al número identificativo del hermano (si no lo tuviese, su valor sería de 0).

Por tanto, para poder representar todos estos datos como una sola variable, alumnos (X) en CSP, lo haremos de la siguiente manera:

$X_{i,j,k,l,m}$, dónde:

- $i \in \{1, 2, 3, 4, 5, 6, 7 \dots 32\}$ representa el número identificativo del alumno
- $j \in \{1, 2\}$ representa a que ciclo pertenece el alumno
- $k \in \{X, C\}$ representa si el alumno es conflictivo (C) o no (X)
- $l \in \{X, R\}$ representa si el alumno es de movilidad reducida (R) o no (X)
- $m \in \{0, 1, 2, 3, 4, 5, 6, 7 \dots 32\}$ representa el número identificativo del hermano (0 si no tiene hermano)

Además, como se podrá intuir, los valores que cogerán estas variables serán las **coordenadas** de su asiento en el **eje x e y** (para poder facilitar el uso de las restricciones), que al final se transformarán para poder obtener el *número real de asiento* de la variable X en el autobús.

3.3. Dominios

Como ya se ha comentado anteriormente, los dominios representarán el tablero en posición vertical, de tal forma que la posición de los asientos será de la forma (x, y), dónde $x \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ representa el número de fila del asiento e $y \in \{0, 1, 2, 3\}$, representa las columnas del autobús. Para poder simplificar el problema lo máximo posible, usaremos solo los dominios estrictamente necesarios:

1. Dominio de los asientos de los **alumnos de movilidad reducida**:

$$D_{X_{i,j,k,R,m}} = \{(x, y)\},\ \forall x \in \{0, 3, 4\},\ \forall y \in \{0, 1, 2, 3\}$$

Con dominio surge un problema y es que no podemos operar sobre este este (si en la codificación), así pues, para simplificar, se dividirá este dominio en **dos nuevos dominios**:

1.1. Dominio de movilidad reducida ciclo 1

$$D_{X_{1,j,k,R,m}} = \{(x, y)\},\ \forall x \in \{0, 3\},\ \forall y \in \{0, 1, 2, 3\}$$

1.2. Dominio movilidad reducida ciclo 2

$$D_{X_{2,j,k,l,R,m}} = \{(x, y)\},\ \forall x \in \{4\},\ \forall y \in \{0, 1, 2, 3\}$$

Nota: el dominio de los alumnos sin movilidad reducida será todo el tablero, ya que siguiendo el requisito tercero, si no hay alumnos de movilidad reducida, el resto de alumnos podrán tomar sus asientos, siempre que estén en su mismo ciclo.

2. Dominio de los asientos de los **alumnos del primer ciclo**:

$$D_{X_{1,j,k,l,m}} = \{(x,y)\},\ \forall x \in \{0,1,2,3\},\ \forall y \in \{0,1,2,3\}$$

3. Dominio de los asientos de los **alumnos del segundo ciclo**:

$$D_{X_{2,j,k,l,m}} = \{(x,y)\},\ \forall x \in \{4,5,6,7\},\ \forall y \in \{0,1,2,3\}$$

3.4. Restricciones

Para poder ver y comprender mejor las restricciones, lo mejor será primero formalizarlas y luego verlas en funcionamiento con un par de ejemplos debido a su complejidad. Nota: para todas las restricciones se asume que los valores X1 y X2 tendrán como dominios todas las posiciones que les corresponde según sus atributos, a no ser, que se indique lo contrario (restricción 6).

1. Los alumnos con número identificativos distintos (todos) han de ocupar asientos distintos.

$$X1_{i,j,k,l,m} \neq X2_{i',j',k',l',m'} \mid i \neq i'$$

2. Si hay un alumno de movilidad reducida sentado, no se podrá sentar nadie en su asiento de al lado (dos restricciones, una para cada posición de cada par de asientos a cada lado del pasillo)

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\} \mid x = x';\ y' \neq \{y, y+1\};\ y \in \{0,2\};\ l = R$$

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\} \mid x = x';\ y' \neq \{y, y-1\};\ y \in \{1,3\};\ l = R$$

3. Si hay un **asiento de persona con movilidad reducida no ocupado**, este pueden **ocuparlo el resto de alumnos**
Esta restricción no hace falta formularla ya que el propio dominio de las personas sin movilidad reducida abarca también los asientos de las personas con movilidad reducida (dominios 3 y 4 previamente expuestos).

4. Los **alumnos conflictivos no pueden sentarse cerca de** ningún **otro alumno conflictivo ni** de **movilidad reducida**

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\} \mid x' \neq \{x, x+1, x-1\};\ y' \neq \{y, y+1, y-1\};\ k' = R \vee l' = C;\ l = C$$

5. Dos **alumnos hermanos han de sentarse juntos (incluso siendo conflictivos),** siempre y cuando **no sean de movilidad reducida** (dos restricciones, una para cada posición de cada par de asientos a cada lado del pasillo, ambos hermanos en el mismo ciclo).

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\} \mid j = j';\ x = x';\ y' = y+1;\ y \in \{0,2\};\ m' = i,\ m = i';\ l = l' = X$$

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\} \mid j = j';\ x = x';\ y' = y-1;\ y \in \{1,3\};\ m' = i,\ m = i';\ l = l' = X$$

6. Si al menos hubiese un **hermano con movilidad reducida**, estos deberían ir al **mismo ciclo pero no en asientos contiguos** (por tanto no se restringe, ya que ya lo hace la restricción 2). Incluso si pertenecieran a ciclos distintos.

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\} \mid j = j';\ l \vee l' = R;\ m' = i,\ m = i'$$

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\};\ X2_{i',j',k',l',m'} = D_{X_{1,j,k,l,m}} \mid j < j';\ m' = i,\ m = i';\ l \vee l' = R$$

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l,m'} = \{(x',y')\};\ X1_{i,j,k,l,m} = D_{X_{1,j,k,l,m}} \mid j > j';\ m' = i,\ m = i';\ l \vee l' = R$$

7. Esta restricción es similar a la restricción 5, pero en este caso los **hermanos** (de no movilidad reducida) pertenecen a **distintos ciclos**. Por tanto se forzaría que **fuesen al ciclo 1** (haciendo que el que estuviese en el ciclo 2 tome por valores los del dominio del ciclo 1) y que el **hermano mayor (de ciclo 2)** vaya en el asiento **pegado al pasillo**.

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\};\ X2_{i',j',k',l',m'} = D_{X_{1,j,k,l,m}} \mid j < j';\ x = x';\ y' = y-1;\ y = 3;\ l = l' = X;\ m' = i,\ m = i'$$

$$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y')\};\ X2_{i',j',k',l',m'} = D_{X_{1,j,k,l,m}} \mid j < j';\ x = x';\ y' = y+1;\ y = 0;\ l = l' = X;\ m' = i,\ m = i'$$

$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y');\ X1_{i,j,k,l,m} = D_{X_{1,j,k,l,m}} \mid j > j';\ x = x';\ y' = y - 1;\ y = 1;\ l = l' = X;\ m' = i,\ m = i'$

$X1_{i,j,k,l,m} = \{(x,y)\},\ X2_{i',j',k',l',m'} = \{(x',y');\ X1_{i,j,k,l,m} = D_{X_{1,j,k,l,m}} \mid j > j';\ x = x';\ y' = y + 1;\ y = 2;\ l = l' = X;\ m' = i,\ m = i'$

Para poder comprender bien cómo funcionan nuestras restricciones, haremos un par de ejemplos que abarquen varias restricciones a la vez y así observar detalladamente su funcionamiento.

1. Hermanos conflictivos y de ciclos distintos

$X1_{1,1,C,X,2} = D_{X_{1,j,k,l,m}}$; $X2_{2,2,C,X,1} = D_{X_{2,j,k,l,m}}$, *aplicando las restricciones 1, 3 y 7 (que es una extensión de la restricción 5) obtenemos:*

$R_{X1,\,X2} = \{((0,0),(0,1)),((1,0),(1,1)),\ ((2,0),(2,1)),\ ((3,0),(3,1)),\ ((0,3),(0,2)),\ ((1,3),(1,2)),\ ((2,3),(2,2)),\ ((3,3),(3,2))\}$

Dónde se ve como los hermanos conflictivos deben ir juntos (moviendo el alumno X2 al ciclo 1) y además, al estar en ciclos distintos, el hermano mayor (X2) tomará los asientos más cercanos al pasillo.

1. Hermanos de movilidad reducida y normal

$X1_{1,1,X,R,2} = D_{X_{1,j,k,l,m}}$; $X2_{2,2,X,R,1} = D_{X_{2,j,k,l,m}}$, *aplicando las restricciones 1, 2, 3 y 6 obtenemos:*

$R_{X1,\,X2} = \{((0,0),(1,0)),((0,1),(0,3)),\ ((0,2),(2,1)),\ ((0,3),(0,0)),\ ((3,0),(0,2)),\ ((3,1),(3,2)),\ ((3,2),(2,2)),\ ((3,3),(1,3)),\ldots\}$

Dónde se puede observar (sobretodo si se comprueban las 112 posibles soluciones), que al ser hermanos pero uno teniendo movilidad reducida (estando en ciclos distintos), ambos irán al ciclo 1, pero se respetarán las necesidades del hermano con movilidad reducida y por tanto no se sentarán juntos (ya que el asiento contiguo a un alumno de movilidad reducida debe quedar vacío).

A estas restricciones hay que sumarle que cuando se aplique tanto la arco-consistencia como la camino-consistencia, nos llevarán a un resultado válido de forma global entre todos los distintos alumnos a colocar en el autobús, restringiéndose entre ellos las distintas soluciones posibles y acotando dichos resultados.

4. Modelo Parte 2

Tras leer el enunciado de la parte 2, se ha destacado la siguiente información relevante:

- Los **alumnos con movilidad reducida** tardan **3 veces más** que el resto de alumnos.
- **No** es posible tener a **2 alumnos con movilidad reducida seguidos.**
- **No** es posible tener a un **alumno con movilidad reducida al final de la cola.**
- El **coste** del alumno que ayuda a subir al **alumno con movilidad reducida** es de **cero**.
- Los **costes** de los alumnos inmediatamente al lado de un alumno **conflictivo** se **duplican** (incluso cuando un alumno conflictivo ayuda a subir a un alumno de movilidad reducida, siendo su coste de 6 unidades)**.**
- Los **costes** de **los alumnos detrás** de un alumno **conflictivo** en la **cola**, con asientos de una **numeración mayor** al del conflictivo (es decir, que se sientan por detrás de él en el autobús), serán **duplicados.**

Nota: la duplicidad de los costes debido a un alumno conflictivo es acumulable. Esto es, un alumno puede ver duplicado su coste varias veces debido a varios alumnos conflictivos que se sientan por delante de él en el autobús.

4.1. Planteamiento del problema de búsqueda

En base a los datos expuestos en el enunciado de la parte 2 se han podido obtener los siguientes datos y características útiles para el planteamiento del problema:

4.1.1. Nodos (estados)

Los nodos/estados de nuestro problema van a estar basados principalmente en la situación en la que se encuentra la cola del autobús y los alumnos que faltan por añadir a esta, mediante los siguientes atributos:

- Un estado **padre** del que proviene el estado actual
- La **cola del autobús** en ese estado, compuesta por listas de alumnos, siendo identificados los alumnos, a su vez, por una lista de dos posiciones, donde la primera posición es la identificación del alumno y la segunda posición, el asiento que ocupa ese alumno en el autobús. *Un ejemplo de lo que sería una cola del autobús en el problema sería: cola_bus = [['1XR', 18], ['5XX', 14]].*
- Una **cola restante**, que contendrá los alumnos que todavía no han sido colocados en la cola del autobús, siguiendo el mismo formato de lista que en la **cola del autobús**.

- Un **coste g**, que será el coste, teniendo en cuenta lo expuesto en el enunciado, que va desde el nodo inicial hasta el estado actual, calculado mediante la función *coste*.
- Un **coste h**, que será el coste heurístico del nodo actual calculado en base a las funciones heurísticas creadas.
- Un **coste f,** que será el coste **total** del nodo actual, siendo este la suma del **coste g** y del **coste h.**

Nota: se ha considerado que dos estados son iguales si las colas del autobús de ambos estados son iguales y con un mismo orden, esto es, que la situación del problema en la que nos encontramos sea exactamente la misma. Por como está formulado este problema, no se repetirán nodos en un mismo camino de búsqueda pero, al tener que implementar A* fielmente, se ha mantenido esta comprobación.

4.1.2. Costes

Los costes se han obtenido en base a la información relevante mencionada en el apartado introductorio y se han realizado ciertas interpretaciones:

- El **coste** de un alumno de **movilidad reducida** es de **3 unidades**, es decir, tres veces mayor que el de los alumnos **conflictivos** y los alumnos **normales**, teniendo estos dos últimos grupos un **coste base** de **1 unidad.**
- El **coste** de un alumno que se encuentra **inmediatamente delante o detrás** de una persona conflictivo es **duplicado**, al igual que el **coste** de cualquier alumno que se encuentre **detrás de un alumno conflictivo en la cola** y que tenga una **posición** asignada en el autobús **mayor** que la del alumno **conflictivo**.
- El **coste** de un alumno **inmediatamente después en la cola** que un alumno de **movilidad reducida** es de **cero**.
- A su vez, si ciertas condiciones anteriormente mencionadas son cumplidas, los **costes** de los alumnos afectados son **duplicados**. *A modo de ejemplo, si hubiesen 3 alumnos conflictivos delante de un alumno con movilidad reducida con una posición mayor en el autobús, el coste de esta última persona asciende a **24** (2*2*2*3). Debido a este fenómeno, se ha tenido que idear un multiplicador que guarda la cantidad de alumnos conflictivos y la posición de estos en el autobús.*

4.1.3. Restricciones

Las restricciones del problema de búsqueda, al igual que en el caso de los costes, han sido extraídas de la información que nos aporta el enunciado. Se han elaborado las siguientes restricciones:

- **No** puede haber un alumno con **movilidad reducida** al **final** de la **cola del autobús**. Esto es debido a que un alumno, ya sea o no conflictivo , debe ayudar al de movilidad reducida a subir.
- **No** pueden haber 2 alumnos con **movilidad reducida juntos en la cola** debido a lo mencionado en el anterior punto.

4.2. Heurísticas

Tras el planteamiento del problema en de búsqueda, las **funciones heurísticas** nos van a permitir optimizar la **obtención del resultado relajando ciertas restricciones** o considerando **casos ideales** en nuestra cola, disminuyendo así el *número de estados creados y expandidos* y, como consecuencia, obteniendo mejores tiempos de ejecución con respecto al problema inicial sin el uso de heurísticas. Para ello se han creado **4 funciones heurísticas** y una quinta que en sí misma no puede ser considerada una heurística, ya que no calcula ningún valor heurístico. Siempre debemos de tener en cuenta que una heurística es **admisible** siempre y cuando sea menor que el coste g del nodo actual, n, al nodo final, f. Es decir, que **nunca sobreestima el coste real**.

4.2.1. Heurística 1: fuerza bruta

Cómo se mencionaba anteriormente, esta función *"heurística"* no se podría aceptar como tal, puesto que **no está nada informada**. Lo que hace esta *"heurística"* es **asignar un coste heurístico 1** a **todos** los estados **menos** al **último** que tendrá **coste 0** (cuando la cola de alumnos por meter esté vacía). Esta heurística se ha creado para poder ver cómo mejorarían el resto de heurísticas anteriores frente a una búsqueda no informada. Es decir, esta heurística es considerada en nuestro programa solo como **modo de evaluación** de las próximas heurísticas.

4.2.2. Heurística 2: alumnos que faltan por meter en la cola del autobús

Esta primera heurística ayuda a A* a pronosticar el estado más cercano a la meta mediante el cálculo de la **longitud de la cola restante**, atributo de los estados que ya se ha descrito anteriormente. Esta heurística es admisible ya que nunca sobreestima el coste ideal que le queda para llegar a la meta. En el caso ideal de que todos los alumnos no fuesen ni conflictivos ni tuviesen movilidad reducida, **el coste heurístico nunca sería mayor que el coste para llegar a la meta**, puesto que cada alumno cuesta una unidad y la heurística calcularía la unidad por el número de alumnos, *que daría el mismo resultado*. En ese caso, el coste heurístico sería igual al coste real.

Nota: se han realizado heurísticas que minimizan el coste de los conflictivos ya que se acerca más al caso ideal. Si se dejasen los conflictivos al principio de la lista, el coste total se vería muy afectado exponencialmente por las cualidades de los alumnos conflictivos ejemplificadas anteriormente.

4.2.3. Heurística 3: 0.5C + 2R + 1N

Esta heurística en principio es mucho **más informada** que la anterior, ya que va a guiar de una manera más eficiente a nuestro algoritmo hacia el resultado óptimo. En esta heurística se relaja la condición de que el coste de la combinación de una persona con movilidad reducida y el ayudante de esa persona tengan coste 3+0, siendo ahora un **coste de 2+1**, es decir, un alumno de **movilidad reducida** tendrá **coste 2** y su **acompañante** de **1**. De esta forma, conseguimos obtener una fórmula matemática que nos permite obtener el caso ideal de este problema, dejando a todos los alumnos **conflictivos al final de la cola**. Esto se consigue gracias a la **relajación del coste de una persona conflictiva**, que pasará de ser de 1 a ser de **0.5** (además no se tiene en cuenta el coste que los conflictivos ocasionan sobre el resto de alumnos). La primera relajación nos permite calcular con mayor precisión los costes de cada persona de la cola, pudiendo de esa forma asumir que los **operadores** de la fórmula matemática del título son los **alumnos de esa índole** que quedan por meter a la cola. A modo de resumen, esta heurística nos permite obtener el coste óptimo al premiar que se posicionen los alumnos conflictivos al final, evitando que estos puedan duplicar los costes de las personas con movilidad reducida o normales, reduciendo así el coste significativamente.
*Esta heurística es admisible puesto que estamos forzando el crear una cola como la siguiente forma de ejemplo: [RNRNCC]. El coste real de la cola siendo alumnos restantes por asignar, en el mejor caso, sería de 3+3+2+2 (10), en cambio, el valor heurístico de este problema sería de 0.5*2 + 2*2 + 1*2 = 6 y como se puede apreciar, no sobreestima el coste real.*

4.2.4. Heurística 4: 0.1C + 2R + 1N

La heurística 4 es una **ampliación** del planteamiento de la **heurística 3**. En esta heurística se consigue **colocar** a los alumnos **conflictivos** de una forma **más agresiva** al **final** de la cola del autobús, pero siendo **menos informada que la heurística 3** al estar más alejado del coste real de una persona conflictiva. Por tanto, esta heurística se alejará a un ritmo mucho mayor del coste real según aumenten los alumnos conflictivos.

4.2.5. Heurística 5: 0.9C + 2R + 1N

La heurística 5, al igual que la heurística 4, se podría definir como una **extensión de la heurística 3**. Esta heurística consigue ser **más informada que la heurística 2** al aproximar el valor asignado a los alumnos conflictivos al coste real que supondría colocar al conflictivo, siempre siendo su **coste (0.9) menor de 1** para así incentivar que estos conflictivos se coloquen al final de la cola por detrás de los alumnos normales que tienen coste 1, lo que es lo **óptimo** en la **práctica totalidad** de los **casos**.

4.3. Diseño de la implementación

Nuestro código ha sido debidamente **modularizado**, contando con una serie de archivos que serán descritos a continuación:

readParse.py
Nos permite **transformar** el **resultado** obtenido de la **parte 1**. Con la función de *parse_result* leemos el archivo y obtenemos un string del diccionario de los alumnos y su posición. Tras esto se llama función *parse_input* que parsea el string recibido por la función anterior a un diccionario de Python. Por último se llama a la función *parse_list* que nos transforma el diccionario anterior a una lista usable, la *cola_total* de todos los alumnos por meter en el autobús en base al formato que buscamos: [[‘4XR’, 2], [‘3XR’, 16]], es decir, [[id,pos],[id,pos]]. Además, está también la función inversa de esta última, *parse_dict*, que nos permite obtener un diccionario en base a una lista, lo cual será útil para escribir el resultado obtenido con el formato adecuado.

writeResult.py
Este archivo contiene las funciones que nos permiten **escribir** un **archivo** de **salida** con el **formato** de diccionario pedido en el **enunciado**, al igual que un archivo de estadísticas generales de la ejecución de A*. Las funciones encargadas de realizar estas actividades son *write_solution* y *write_statistics,* respectivamente.

heuristicas.py
Nos permite **seleccionar** la **heurística** a **usar** en **A*** junto con las implementaciones de dichas heurísticas, ya comentadas en el anterior punto. Cabe destacar que en la heurística 3, 4 y 5 iteramos por todos los alumnos de la cola restante para sacar el valor de la fórmula matemática extraída, lo cuál cogerá importancia al comparar el rendimiento de las heurísticas más adelante.

restricciones.py
Implementa una serie de funciones que van a ser de **vital importancia** para la **posible expansión** del **A***. Las funciones de *mov_reducida_final* y *mov_reducida_seguidos* materializan las **restricciones** halladas en el **enunciado**. *mov_reducida_seguidos* comprueba si el último alumno metido en la cola y el siguiente a meter (ya decido al expandir) son ambos de movilidad reducida. Por otro lado, *mov_reducida_final* comprueba que si el último alumno a añadir es de movilidad reducida y si la cola de alumnos restantes por meter está vacía, entonces vamos a meter como último alumno uno de movilidad reducida. Con que una de estas dos restricciones sea positiva, el nodo hijo a expandir será desechado por el incumplimiento de las restricciones.

comprobaciones.py
Realiza una serie de **comprobaciones** del **estado**. Con la función *comprobar_estado*, se valora si la **heurística es válida o no**. Antes de nada se comprueba que los costes f sean la suma de los coste g más los costes heurísticos. Posteriormente, comprobará que h(n) > h(n') para toda la ruta. Esta función sólo se ha utilizado durante el **proceso de creación de las heurísticas**.

costeG.py
Posee una función *coste* que determina el **coste real** en base a las condiciones que se mencionaron en la modelización del **problema de búsqueda**. Esta función se encarga de sumar los costes de los alumnos de una cola de alumnos dada, comprobando los costes de estos alumnos y los efectos que tienen sus posiciones con respecto al resto. Para ello se evaluarán teniendo en cuenta los **tipos de alumnos en el orden predeterminado de la lista** que recibe como parámetro. Además, se hará uso de una **lista** para almacenar los **costes** de cada alumno (ya que un coste de un alumno puede verse afectado por los alumnos posteriores a este) y de una **lista** para almacenar los alumnos **conflictivos** (para saber cuantos conflictivos hay por delante de un alumno que le puedan afectar a su coste).

utils.py
Aporta **funciones complementarias** al manejo de **A***. La función ***restantes*** nos permite obtener el atributo *cola_restante* en base a la *cola_total* (mencionada en **readParse.py**) y al atributo *cola_bus*. ***is_goal*** nos permite saber si se ha llegado al estado meta (meta cuando h=0 en un estado). Finalmente, la función ***esta_en_lista*** nos permite, comprobar si un estado está en una lista (lista abierta o lista cerrada) por la se itera haciendo uso de un bucle for.

ASTARColaBus.py
Es el programa principal de la Parte 2. En él implementamos las **funciones características del A*** junto con el **algoritmo de A***. Ya que es el programa donde se ejecutará la **función *main***, se ha implementado una función *command_prompt* que nos permite obtener los argumentos introducidos por el terminal (o bash) a la hora de ejecutar A* (aportando además el path del .prob y número de la heurística a usar). Como *método* de la clase ***Estado*** (comentada en la modelización del problema), tenemos *__eq__*, que nos permite **comparar** dos **estados** usando el operador "=".
Otra función fundamental para el manejo de A* es ***expandir***, la cual recibe un **estado** y devuelve una **lista** de todos sus posibles **hijos**. Los **hijos** van a ser **creados** en **base** a la **cola restante** del estado a expandir ya que, en este problema de búsqueda, se va a poder meter en la cola del autobús cualquier alumno, siempre y cuando se cumplan las dos restricciones mencionadas (que no existan dos alumnos de movilidad reducida juntos o que no se coloque un alumno de movilidad reducida al final). Si ninguna de las restricciones ha sido violada, se procederá a **crear** un **hijo** (este proceso se repetirá tantas veces como alumnos haya en la cola restante). Este **hijo** va a tener una **cola del autobús** igual a la del estado de expansión solo que con la adición del elemento seleccionado de la cola restante en esa iteración del bucle.
Por otro lado, la cola restante se creará con la función **restantes** comentada en **utils.py**, el **padre** será el **estado de expansión** y el coste se asignará cuando se cree el estado. Finalmente, el **hijo** será **añadido** al final de la **lista de hijos** que va a ser devuelta.
En cuanto al algoritmo de A* implementado en la función ***a_estrella***, se han creado *variables para controlar los tiempos de ejecución y control de las estadísticas del programa*, que tendrán que ser redirigidas al archivo de salida de las soluciones y estadísticas. En lo que se refiere al algoritmo en sí, **no** se han realizado **grandes modificaciones con respecto al pseudocódigo suministrado**. Para detallar, cuando se comprueba si el *estado actual es meta*, la **variable de control** de estado final es **actualizada**, **terminando** la **ejecución** de A*. Cuando no es meta, el **estado actual** es insertado en la **lista cerrada** y es **expandido** gracias a la función *expandir*. Si ha devuelto hijos la función anterior, **iteramos** por los **hijos** para ir comprobando si se encuentra tanto en la lista cerrada como en la abierta. Si el **hijo** se encuentra en la **lista cerrada**, **pasamos** al siguiente hijo por estar ya expandido ese estado. Si **no** se encuentra en la **lista cerrada ni** tampoco en la **abierta**, se añade a la **lista abierta**. Finalmente, si **sí** que se encuentra en la **lista abierta** el estado actual, se **busca** el otro **estado igual** (coincidente) por toda la lista abierta. Al encontrarlo, si el **coste** del **hijo** es **menor** que el del **estado** de la **lista abierta**, **se sustituye**. Como otro comentario a cerca de la implementación de A*, al terminar la expansión del estado actual, se efectúa una **ordenación de la lista abierta de A*** mediante una función lambda por el menor **coste f**; en caso de empate se pondrá por delante el estado con un menor **coste h.** Posteriormente, **si** se ha llegado a una **solución**, se procederá a hacer **backtracking** por los **padres** para obtener la **ruta** seguida. Finalmente, se exportan los resultados en los archivos de salida creados usando las funciones del módulo **writeResult.py**. En la función ***main***, se obtiene la ruta y la heurística seleccionada por el usuario gracias a la función *command_prompt*, explicada al principio. A parte de la heurística seleccionada, como parámetros de la función *a_estrella,* se le añade la *cola_total* obtenida de la función *parse_result* de **readParse.py** y un **estado inicial** en donde la **cola del autobús** se encuentra **vacía** donde el padre es None ya que es el nodo inicial y se asigna la ***cola_total*** (cola con los alumnos a introducir en la cola del autobús) como **cola restante** del estado inicial. Finalmente, se ejecuta la función *a_estrella* para generar los **resultados y las estadísticas** del **problema de búsqueda.**

5. Análisis de los Resultados

De cara al análisis de los resultados, vamos a comentar los distintos **casos de prueba** que se han llevado a cabo y la razón de estos tests. Además, en la parte 2 se compararán los **resultados** de las **diferentes heurísticas implementadas**. Para hacer esta comparación lo más sencilla y efectiva posible, haremos uso de una **tabla** con los **resultados** de las ejecuciones (que hayan tenido un resultado válido).

5.1. Parte 1

En la parte 1 se han elaborado un total de **34 test unitarios** para poder asegurar de forma prácticamente inequívoca que nuestras variables, dominios, y sobre todo, restricciones, funcionan cumpliendo lo expuesto en el enunciado. Para poder organizar de forma eficiente las pruebas, estas se han dividido en los siguientes grupos (siguiendo el orden expuesto en *ASTAR-calls.sh*):

5.1.1. Pruebas con 1 solo alumno

Estas pruebas nos servirán para comprobar que, tanto un alumno normal como uno conflictivo, puede sentarse en los 16 sitios disponibles, pudiendo ser del ciclo 1 o del ciclo 2 del autobús (contando con los asientos de las personas de movilidad reducida).
alumnos00*: un alumno normal en el ciclo 1*, se comprobará que este pueda coger los **16 asientos posibles**.
alumnos01*: un alumno normal en el ciclo 2*, se comprobará que este pueda coger los **16 asientos posibles**.
alumnos02*: un alumno conflictivo en el ciclo 1*, se comprobará que este pueda coger los **16 asientos posibles**.
alumnos03*: un alumno conflictivo en el ciclo 2*, se comprobará que este pueda coger los **16 asientos posibles**.

5.1.2. Caso excepcional

alumnos04*:* se comprueba que *dos alumnos con un mismo número de identificación*, lleva a **error** del **sistema**.

5.1.3. Pruebas de alumnos conflictivos

Estas pruebas nos ayudarán a comprobar el correcto funcionamiento de las **restricciones** de los espacios que hay que mantener entre los alumnos **conflictivos** junto con los alumnos de movilidad reducida.
alumnos05*: cinco alumnos conflictivos en el ciclo 1*. Esto resulta en **0 soluciones válidas** ya que excede el número de alumnos conflictivos (sin ser hermanos) que caben en un solo ciclo.
alumnos06*: cinco alumnos conflictivos y de movilidad reducida ('CR')*. Caso similar al anterior, **no tiene soluciones**.
alumnos07*: mezcla de cinco alumnos conflictivos y de movilidad reducida*. Caso similar al anterior, **no tiene soluciones**.
alumnos08*: cuatro alumnos conflictivos*. Tiene soluciones válidas, ya que es el **máximo número de alumnos conflictivos** (sin ser hermanos) que caben en un ciclo.
alumnos09*: cuatro alumnos conflictivos y de movilidad reducida en un mismo ciclo*. Caso similar al anterior, solo que estos conflictivos al tener también **movilidad reducida** han de estar en los **asientos reservados para ellos**. Soluciones válidas.
alumnos10*: dos alumnos conflictivos y dos de movilidad reducida*, *mismo ciclo*. Similar a los dos anteriores, soluciones válidas.

5.1.4. Pruebas de alumnos de movilidad reducida

Con estas pruebas se ha comprobado que las **restricciones** que afectan directamente a los **alumnos de movilidad reducida** se cumplen (los asientos que toman, los números máximos de alumnos de este tipo por ciclo, su interacción con otros alumnos…).
alumnos11*: cuatro alumnos de movilidad reducida en el ciclo 1*. Esta tiene soluciones válidas ya que el ciclo 1 dispone de 8 asientos de movilidad reducida (de los cuales los alumnos de movilidad reducida ocupan uno y su contiguo ya que necesitan espacio). Por tanto siendo **4** el **número máximo de alumnos de movilidad reducida** que caben en el **ciclo 1**.
alumnos12*: dos alumnos de movilidad reducida en el ciclo 2*. Similar al anterior pese a que en el **ciclo 2** solo se dispone de 4 asientos especiales y por tanto como máximo cabrán **2 alumnos de movilidad reducida**. Teniendo este test **8 soluciones válidas**.
alumnos13*: cinco alumnos de movilidad reducida, ciclo 1*. Por lo explicado en *alumnos11*, este test no tendrá soluciones válidas.
alumnos14*: tres alumnos de movilidad reducida, ciclo 2*. Por lo explicado en *alumnos12*, este test no tendrá soluciones válidas
alumnos15*: dos alumnos de movilidad reducida y uno normal en el ciclo 2*. Este test simplemente comprueba la **interacción** de un alumno normal con los de **movilidad reducida** (no ocupará ninguno de los 4 asientos reservados a movilidad reducida).

5.1.5. Pruebas de hermanos

Estas pruebas comprueban los casos posibles a ocurrir entre **hermanos** (que ocurre cuando están en el mismo ciclo, en ciclo distinto, cuando son hermanos con distintas características…) y que el resultado sea acorde con el enunciado.
alumnos16*: dos hermanos normales en un mismo ciclo 1*. Este test tendrá **16 soluciones válidas** ya que hay 4 filas a cada lado del pasillo, los hermanos deben ir juntos y el lado en el que se sientan los alumnos es intercambiable.
alumnos17*: dos hermanos normales en un mismo ciclo 2*. Este test es exactamente igual al anterior, **16 soluciones válidas**.

alumnos18: *dos hermanos normales en diferentes ciclos*. Ambos irán al **ciclo 1** y el hermano mayor (en este caso el de id 1) irá pegado al pasillo. Por tanto serán **8 soluciones válidas** (4 filas de 2 asientos a cada lado del pasillo).
alumnos19: Este test es igual al anterior solo que en este caso el **hermano mayor** es el que tiene **ID 2**, 8 soluciones válidas.
alumnos20: *dos hermanos conflictivos en el mismo ciclo*. Podrían ir juntos y por tanto al igual que en el test 16 y 17, tendría 16 soluciones válidas.
alumnos21: *dos hermanos conflictivos en diferentes ciclos*. Interactúan entre ellos de igual manera que los alumnos del test 18 y 19, dando como resultado **8 soluciones válidas** (hermano mayor con ID 1, pegado al pasillo).
alumnos22: igual que el test anterior solo que el **alumno mayor** es el de **ID 2**, yendo pegado al pasillo. **8 soluciones válidas**.
alumnos23: *un hermano conflictivo y uno normal*. Comportamiento igual al de los tests 16, 17 y 20. **16 soluciones válidas**.
alumnos24: *dos hermanos y tres alumnos, conflictivos*. Tener ahora **5 alumnos conflictivos es posible** ya que dos de ellos son *hermanos* y por tanto han de ir **juntos**, reduciendo la cantidad de espacio que ocupan en el autobús. Test con soluciones válidas.
alumnos25: *dos hermanos conflictivos y cuatro alumnos conflictivos*. Al solo tener dos alumnos hermanos conflictivos, el meter 6 conflictivos en un mismo ciclo sigue sin ser posible (5 como máximo siendo de ellos 2 hermanos), por tanto este test no tiene soluciones válidas.
alumnos26: *un hermano de movilidad reducida y el otro normal en el mismo ciclo.* Este test es válido pero además podemos comprobar que el alumno de movilidad reducida está en un asiento reservado para él, **no estando los hermanos juntos**.
alumnos27: *un hermano de movilidad reducida y el otro normal en distintos ciclos.* En este test se observa como ambos hermanos se van al ciclo 1, pero **sin estar juntos**, ya que el de movilidad reducida ha de estar en un **asiento reservado**.
alumnos28: dos hermanos de movilidad reducida y conflictivos. Funciona similar a dos hermanos de movilidad reducida (donde estos han de estar en el mismo ciclo pero en sus asientos reservados), con el **añadido** de que **también restringen** un **espacio** del autobús a otros conflictivos y de movilidad reducida (al ser también conflictivos).
alumnos29: esta es una prueba válida que consta de un **problema con varios hermanos** (8) con **distintos atributos**, siendo este una *mezcla* entre las distintas interacciones comentadas en los test previos.

5.1.6. Pruebas más extensas

Este grupo de 4 test (del test **alumnos30** al test **alumnos33**) pretende abarcar mediante test con resultados válidos lo visto en todas las anteriores pruebas (varios hermanos, alumnos de movilidad reducida, alumnos normales, conflictivos, conflictivos y de movilidad reducida…) en unos **test de longitud más extensa a los anteriores** y así comprobar el buen funcionamiento de nuestro programa a una **mayor escala** (y que los resultados a esta escala son coherentes). También se ha ejecutado el test que aparecía de ejemplo en el enunciado de este ejercicio.

Nota: los nombres de los tests en rojo indican que esos test no tienen soluciones válidas ya que han sido elaborados a propósito para comprobar que no se obtienen soluciones cuando se excede algún parámetro.

5.2. Parte 2

5.2.1. Descripción de las pruebas

Para la comprobación del problema de búsqueda implementado usando el algoritmo A*, se han realizado **16 pruebas** que comprueban tanto el **correcto funcionamiento de las heurísticas** como los **errores** arrojados por el salto de restricciones y la correcta asignación de los costes en cada caso.

Los tests realizados han sido los siguientes:
alumnos00: *un alumno normal.*
alumnos01: *un alumno de movilidad reducida*. Error por insertar un alumno de movilidad reducida al final.
alumnos02: *dos alumnos de movilidad reducida*. Error por poner dos alumnos de movilidad reducida seguidos y uno al final.
alumnos03: *dos alumnos de movilidad reducida y uno normal*. Error por insertar un alumno de movilidad reducida al final.
alumnos04: *dos alumnos normales y uno de movilidad reducida*.
alumnos05: *tres alumnos de movilidad reducida y dos conflictivos*. Error por insertar un alumno de movilidad reducida al final.
alumnos06: *un alumno de movilidad reducida y uno normal*.
alumnos07: *un alumno de movilidad reducida y uno conflictivo*.
alumnos08: *un alumno conflictivo*.
alumnos09: *dos alumnos conflictivos*.
alumnos10: *tres alumnos conflictivos*.
alumnos11: *cuatro alumnos conflictivos*.
alumnos12: *todo alumnos normales*.
alumnos13: *ocho alumnos tanto de movilidad reducida como normales con ids de 2 dígitos*.
alumnos14: *ocho alumnos de cualquier tipo*.
alumnos15: *ocho alumnos de cualquier tipo* (confirmar correcto funcionamiento de A*).

Se han realizado tests específicamente para comprobar el efecto de la conflictividad (del *alumnos08* al *alumnos11*) puesto que **nuestras heurísticas 3, 4 y 5 relajan** los **costes de los conflictivos** para colocarlos así al final. Esto resulta en un arma de doble filo para nuestras heurísticas ya que si hay muchos alumnos conflictivos, estos se juntarán al final, dónde los conflictivos tomarán coste 0.5 por igual, resultando en una búsqueda parecida a nuestra heurística 2 (número de alumnos por meter en la cola del autobús) con el coste computacional añadido de que iteramos por todos los alumnos restantes para saber a qué grupo pertenecen y así asignarles un coste. Por tanto, podemos intuir que **a mayor número de alumnos conflictivos, mayor coste computacional tendrán nuestras heurísticas** (3, 4 y 5) **y por tanto, un peor rendimiento**.

5.2.2. Resultados

Los resultados de los tests han sido los siguientes:

Test	Heurística	Nodos expandidos	Longitud del plan	Coste total	Cola del bus	Tiempo
Alumnos00	1	1	1	1	{'1XX': 14}	0,003297328948974600
	2	1	1	1	{'1XX': 14}	0,000564813613891601
	3	1	1	1	{'1XX': 14}	0,000323534011840820
	4	1	1	1	{'1XX': 14}	0,000361919403076171
	5	1	1	1	{'1XX': 14}	0,000412225723266601
Alumnos04	1	7	10	4	{'1XR': 14, '3XX': 7, '4XX': 9}	0,002304792404174800
	2	6	8	4	{'3XX': 7, '1XR': 14, '4XX': 9}	0,002045869827270500
	3	6	8	4	{'3XX': 7, '1XR': 14, '4XX': 9}	0,002938270568847650
	4	6	8	4	{'3XX': 7, '1XR': 14, '4XX': 9}	0,001526355743408200
	5	6	8	4	{'3XX': 7, '1XR': 14, '4XX': 9}	0,004219532012939450
Alumnos06	1	3	3	3	{'1XR': 14, '5XX': 28}	0,001312255859375000
	2	3	3	3	{'1XR': 14, '5XX': 28}	0,000882148742675781
	3	3	3	3	{'1XR': 14, '5XX': 28}	0,000636816024780273
	4	3	3	3	{'1XR': 14, '5XX': 28}	0,001097679138183590
	5	3	3	3	{'1XR': 14, '5XX': 28}	0,002263069152832030
Alumnos07	1	3	3	6	{'1XR': 14, '6CX': 25}	0,000472307205200195
	2	3	3	6	{'1XR': 14, '6CX': 25}	0,000760316848754882
	3	3	3	6	{'1XR': 14, '6CX': 25}	0,000518083572387695
	4	3	3	6	{'1XR': 14, '6CX': 25}	0,000352621078491210
	5	3	3	6	{'1XR': 14, '6CX': 25}	0,000695228576660156
Alumnos08	1	1	1	1	{'6CX': 25}	0,000436782836914062
	2	1	1	1	{'6CX': 25}	0,000683307647705078
	3	1	1	1	{'6CX': 25}	0,000463247299194335
	4	1	1	1	{'6CX': 25}	0,000683069229125976
	5	1	1	1	{'6CX': 25}	0,001701831817626950
Alumnos09	1	3	4	4	{'6CX': 25, '9CX': 9}	0,001125335693359370
	2	3	4	4	{'6CX': 25, '9CX': 9}	0,000806093215942382
	3	3	4	4	{'6CX': 25, '9CX': 9}	0,000736474990844726
	4	3	4	4	{'6CX': 25, '9CX': 9}	0,000446557998657226
	5	3	4	4	{'6CX': 25, '9CX': 9}	0,001323938369750970
Alumnos10	1	10	15	8	{'6CX': 25, '12CX': 12, '9CX': 9}	0,003263473510742180
	2	10	15	8	{'6CX': 25, '12CX': 12, '9CX': 9}	0,003789186477661130
	3	10	15	8	{'6CX': 25, '12CX': 12, '9CX': 9}	0,002632379531860350
	4	10	15	8	{'6CX': 25, '12CX': 12, '9CX': 9}	0,003393650054931640
	5	10	15	8	{'6CX': 25, '12CX': 12, '9CX': 9}	0,002399206161499020
Alumnos11	1	25	48	12	{'8CX': 32, '6CX': 25, '12CX': 12, '9CX': 9}	0,011441230773925700
	2	25	48	12	{'8CX': 32, '6CX': 25, '12CX': 12, '9CX': 9}	0,010143995285034100
	3	25	48	12	{'8CX': 32, '6CX': 25, '12CX': 12, '9CX': 9}	0,007790327072143550
	4	25	48	12	{'8CX': 32, '6CX': 25, '12CX': 12, '9CX': 9}	0,012052059173583900
	5	25	48	12	{'8CX': 32, '6CX': 25, '12CX': 12, '9CX': 9}	0,012879133224487300
Alumnos12	1	18	41	4	{'3XX': 7, '4XX': 9, '5XX': 28, '7XX': 30}	0,004202842712402340
	2	4	10	4	{'3XX': 7, '4XX': 9, '5XX': 28, '7XX': 30}	0,002639532089233390
	3	4	10	4	{'3XX': 7, '4XX': 9, '5XX': 28, '7XX': 30}	0,001710653305053710
	4	4	10	4	{'3XX': 7, '4XX': 9, '5XX': 28, '7XX': 30}	0,001808166503906250
	5	4	10	4	{'3XX': 7, '4XX': 9, '5XX': 28, '7XX': 30}	0,002099990844726560
Alumnos13	1	20637	22940	12	{'5XR': 12, '4XX': 2, '6XR': 13, '3XX': 16, '1XR': 20, '2XX': 18, '11XR': 27, '10XX': 29}	52,143829345703100000
	2	16053	18764	12	{'5XR': 12, '4XX': 2, '6XR': 13, '3XX': 16, '1XR': 20, '2XX': 18, '11XR': 27, '10XX': 29}	28,025301933288500000
	3	7560	8882	12	{'5XR': 12, '4XX': 2, '6XR': 13, '3XX': 16, '1XR': 20, '2XX': 18, '11XR': 27, '10XX': 29}	7,047176837921140000
	4	7560	8882	12	{'5XR': 12, '4XX': 2, '6XR': 13, '3XX': 16, '1XR': 20, '2XX': 18, '11XR': 27, '10XX': 29}	8,179629325866700000
	5	7560	8882	12	{'5XR': 12, '4XX': 2, '6XR': 13, '3XX': 16, '1XR': 20, '2XX': 18, '11XR': 27, '10XX': 29}	7,304058313369750000
Alumnos14	1	2485	8680	24	{'9CX': 30, '5CR': 18, '3XX': 15, '6CX': 12, '8CX': 11, '4CX': 5, '1CX': 4, '2CX': 3}	8,696328163146970000
	2	1921	6873	24	{'9CX': 30, '5CR': 18, '3XX': 15, '6CX': 12, '8CX': 11, '4CX': 5, '1CX': 4, '2CX': 3}	5,323535919189450000
	3	2300	7961	24	{'9CX': 30, '5CR': 18, '3XX': 15, '6CX': 12, '8CX': 11, '4CX': 5, '1CX': 4, '2CX': 3}	8,428626775741570000
	4	2681	9182	24	{'9CX': 30, '5CR': 18, '3XX': 15, '6CX': 12, '8CX': 11, '4CX': 5, '1CX': 4, '2CX': 3}	11,111175060272200000
	5	2180	7546	24	{'9CX': 30, '5CR': 18, '3XX': 15, '6CX': 12, '8CX': 11, '4CX': 5, '1CX': 4, '2CX': 3}	8,999287128448480000
Alumnos15	1	3182	10563	24	{'9CX': 27, '5CR': 20, '3XX': 11, '2CX': 14, '1CX': 13, '4CX': 5, '6CX': 4, '8CX': 3}	12,771411657333300000
	2	2483	8473	24	{'9CX': 27, '5CR': 20, '3XX': 11, '2CX': 14, '1CX': 13, '4CX': 5, '6CX': 4, '8CX': 3}	8,488305568695060000
	3	3008	9884	24	{'9CX': 27, '5CR': 20, '3XX': 11, '2CX': 14, '1CX': 13, '4CX': 5, '6CX': 4, '8CX': 3}	16,128100872039700000
	4	3399	11079	24	{'9CX': 27, '5CR': 20, '3XX': 11, '2CX': 14, '1CX': 13, '4CX': 5, '6CX': 4, '8CX': 3}	17,171511650085400000
	5	2822	9289	24	{'9CX': 27, '5CR': 20, '3XX': 11, '2CX': 14, '1CX': 13, '4CX': 5, '6CX': 4, '8CX': 3}	12,940318346023500000

Nota: La tabla con los datos se ha desarrollado para fines meramente orientativos, ya que por cualquier proceso del sistema (pese a haber usado una distribución de Linux para ejecutarlos) puede que el tiempo obtenido en algún test desvaríe.

5.2.3 Conclusiones acerca de los resultados obtenidos

Como se puede apreciar en la tabla, se ha **saltado** del test *alumnos00* al *alumnos04* y se ha omitido también el test *alumnos05* puesto que son tests específicamente preparados para que nos arrojasen **errores** de **violación** de las **restricciones**, ya sea por que hubiesen *dos alumnos de movilidad reducida seguidos* o *uno al final de la cola del autobús*, como ya se comentó anteriormente.

Los **primeros tests**, del test *alumnos00* al test *alumnos07*, **comprueban** principalmente **2 cosas**, que los **costes** son **generados correctamente** dependiendo de la situación y que las **heurísticas** encuentran el **mejor camino** sin tener que visitar todos los nodos como sí haría la heurística 1 (fuerza bruta). Al ser **tests** tan **pequeños** en cuanto a volumen de alumnos, los **tiempos** empleados son **bastante similares**, al igual que la cantidad de nodos expandidos y la longitud del plan.
Cabe recalcar un caso interesante obtenido en el test *alumnos04*. En este test, se puede apreciar que la **cola final del autobús es distinta** que la de las heurísticas creadas. Esto es debido a que se puede obtener el **mismo coste de 2 formas**. Tanto si se coloca el alumno de movilidad reducida al principio de la cola como en el medio, el coste total seguirá siendo de 4.

Del test *alumnos08* al test *alumnos11*, ya que nuestras heurísticas son más susceptibles a la cantidad de alumnos conflictivos, se ha querido **evaluar** si suponía un **aumento de tiempo sustancial**. Como se puede apreciar en los resultados, el **aumento** de tiempo que se obtiene al ir pasando de 1 alumno conflictivo a 4 alumnos conflictivos es bastante **llamativo**, confirmando las hipótesis formuladas antes de realizar las pruebas de rendimiento. Por otro lado, los costes y los resultados de las ejecuciones siguen siendo iguales a la heurística 1, por lo que nuestras heurísticas siguen encontrando el camino óptimo.

El test *alumnos12* se ha creado para ser comparado con el test *alumnos11*. Estos dos tests contienen **4 alumnos**, siendo los de *alumnos11* **conflictivos** y los de alumnos12, **normales**. No se ha creado un test lleno de alumnos de movilidad reducida puesto que nos arrojaría un error debido a las restricciones del problema. Como suponíamos, los costes temporales de *alumnos11* son mayores que los de *alumnos12*. Esto es debido a que, como los alumnos no conflictivos no poseen la característica de duplicar los valores de los alumnos inmediatamente delante y detrás (además de multiplicar por 2 el coste de los que están por detrás de ellos en la cola y en el autobús), *alumnos12* tendrá una **ejecución más lineal** que la de alumnos11, no teniendo que comprobar a los alumnos ya pertenecientes a la cola o a los alumnos que sean inmediatamente añadidos. finales, siendo **3 veces inferior a la heurística 2** y más de **5 veces inferior a la heurística 1**.

Realmente, donde se pueden sacar conclusiones más precisas es a partir de *alumnos13* hasta *alumnos15*.

Al aumentar el número de alumnos, el número de nodos expandidos y la longitud del plan **aumentan de forma exponencial**. En cambio, los **costes finales** siguen siendo relativamente **bajos**. El test *alumnos13* no posee alumnos conflictivos, haciendo que sea un **caso mucho más favorable para nuestras heurísticas 3, 4 y 5**. Al fin y al cabo, la heurística 2 es una heurística poco informada que solo va calculando la longitud de la cola de los alumnos que quedan por meter, por lo que va a tender más al rendimiento de la heurística 1 que a las demás. En este test, tanto los **nodos expandidos** como la **longitud del plan** de las **heurísticas 3, 4 y 5** son casi **dos veces inferiores** a los de la **heurística 1 y 2**. Los tiempos también son significativos y muy favorables para las 3 heurísticas

Este mismo patrón de rendimiento se ha podido observar en numerosos otros tests que se han realizado, dónde se observaba que cuando el **número de alumnos conflictivos** era **bajo**, las **heurísticas 3, 4 y 5** tenían **rendimientos muy superiores** al resto. *Un ejemplo de estos otros test extra implementados sería el test alumnos16, dónde se prueba con la ejecución de un test compuesto por 8 alumnos normales del ciclo 1. En este test, la heurística 1 (fuerza bruta) tarda alrededor de 250 segundos (con una longitud del plan de casi 70 mil nodos y de estos, casi 30 mil expandidos), mientras que la heurística 2 tarda 0.03 segundos y las heurísticas 3, 4 y 5 rondan los 0.025 segundos de media (con una longitud del plan de 36 nodos y 8 nodos expandidos).*
Este último test (alumnos16) no se ha añadido a los test anteriores debido a su similitud con el test alumnos12, solo que a una mayor escala (de 4 a 8 alumnos normales) y por tanto con un peor rendimiento para hacer las pruebas.

Al igual que contamos con test favorables para nuestras heurísticas, también contamos con los **tests menos favorables**. En los tests *alumnos14 y alumnos15* se ha incluído una **gran cantidad de alumnos conflictivos** para observar qué tan dañina es para estas heurísticas la adición de muchos alumnos conflictivos. Aunque tanto los *costes como la cola final* sean **óptimos**, los *tiempos* necesitados para ejecutar el algoritmo A*, *los nodos expandidos* y la *longitud del plan* son **peores** respecto a la heurística 1. Esto se debe entre otras cosas a que, en contraste con la búsqueda no informada (o de fuerza bruta) y con la heurística de la longitud de la cola restante, en las heurísticas 3, 4 y 5 se requiere de mucha más capacidad de cómputo al haber **muchos alumnos conflictivos juntos**, estas heurísticas **pierden mucho tiempo haciendo cálculos y asignando costes** (mismo coste por la cantidad de alumnos conflictivos restantes) recorriendo la lista de alumnos varias veces para identificar el tipo de alumno, lo que las hace **muy costosas si se incrementan mucho el número de alumnos conflictivos** para al final ser muy poco informadas cuando hay solo alumnos conflictivos.

*Por ejemplo, poner 3 alumnos conflictivos seguidos tiene un **coste real** (mínimo, excluyendo los costes adicionales sobre los alumnos posteriores en la cola con una posición más atrás en el autobús) **de 8** mientras que con nuestra heurística tendrían un **coste fijo de 1.5**.*

Esto repercute mucho en las heurísticas 3, 4 y 5 como se ha visto. De hecho, si comparamos las tres heurísticas la que **mejor rendimiento** tiene es la **quinta**, ya que asigna un coste de 0.9 por cada alumno conflictivo, por lo que es un poco **más informada** que la tercera que les asigna un coste de 0.5 y mucho más que la cuarta que les asigna un coste de 0.1. Esta hipótesis se puede corroborar fácilmente observando los resultados de test *alumnos14 y especialmente en alumnos15* adjuntados en la tabla, dónde se puede ver fácilmente como la peor de todas las heurísticas es la cuarta, es decir, la que asignaba un coste de 0.1 a los conflictivos y la mejor la quinta, que asignaba un coste de 0.9 por alumno conflictivo.

Es por esta misma razón que, lo **mejor** sería **quedarse** con la **heurística 5** (*de entre las heurísticas 3, 4 y 5*, en caso de que aumenten los alumnos conflictivos), ya que es la **más informada de las tres** en lo referente a los alumnos conflictivos y la que mejores rendimientos tiene (es importante el hecho de que este coste de los conflictivos sea menor que 1 (0.9) ya que así incentivamos que nuestro algoritmo coloque en la parte de atrás de la cola a los alumnos conflictivos, que pueden ocasionar grandes costes al ser colocados en medio o delante de la cola.

Por lo tanto, en caso de que haya *pocos alumnos conflictivos*, lo **mejor** por gran diferencia será quedarse con la **heurística 3** o la **heurística 5** (preferiblemente lo comentado anteriormente), en cambio, según **aumenta la cantidad de conflictivos**, lo mejor será escoger la **heurística 2**.

6. Conclusiones

Esta práctica ha sido bastante útil para nosotros de cara al asentamiento de los conocimientos adquiridos durante el tema de ***"satisfacción de restricciones"* (CSP)**, pero sobretodo, hemos notado que nos ha sido de gran ayuda con el tema de las ***búsquedas heurísticas*** en particular, ya que para poder haber terminado la **parte 2** de esta segunda práctica, debíamos adquirir unos conocimientos bastante amplios sobre el funcionamiento y aplicación de **A*** junto a la creación e implementación de unas buenas **heurísticas**.

Al igual que con el último trabajo, esto nos ha sido muy útil para poder asentar de **forma práctica** y **visual** los **conocimientos teóricos** vistos en clase, lo que particularmente a nosotros, nos ayuda mucho de cara a la correcta comprensión de esta asignatura.

Pese a todo lo mencionado, sí que hemos visto ciertos fallos en el enunciado como que por ejemplo se exponía que se debía de empezar desde una posición de la cola del autobús ya completa para la parte 2, por suerte, **todo esto se aclaró** en clase y **no ocasionó mayores problemas**.

No sabemos si es por que esta práctica nos ha resultado más interesante o intuitiva que la anterior por lo que hemos notado una **menor dificultad** en esta última, pero sin lugar a dudas, **nos ha gustado más esta**.

Comparativamente, la **segunda parte** de esta entrega nos ha llevado **mucho más tiempo**, ya que era **más extensa** y había que razonar muy bien qué heurísticas utilizar y cuáles no, con el añadido de que habíamos hecho **muchos menos ejercicios de este tema en clase** en comparación con "*satisfacción de restricciones"*.

En resumen, esta práctica, nos ha ayudado gratamente de cara a comprender la implementación de CSP en el lenguaje de programación **Python** junto con el el funcionamiento de las heurísticas y del algoritmo de A*, lo que intuimos que nos será de **enorme utilidad de cara al exámen final**.