Práctica de Planificación

Josep Maria Olivé Fernández, Pol Monroig, Yaiza Cano

31 de Mayo del 2020

Índice

1 El problema

Nuestro cliente *CoachingPotato*, después del maravilloso sistema de recomendación de ejercicios que le construimos en CLIPS, decidió que quería que le proporcionáramos una herramienta sencilla que, basándose en los ejercicios que el usuario está realizando en este momento y su nivel de dificultad, les haga un plan de entrenamiento físico quincenal con el objetivo de conseguir subir de nivel en un conjunto de ejercicios escogidos.

El plan de entrenamiento ha de tener en cuenta las siguientes restricciones:

- Ejercicios precursores: existen ejercicios que son predecesores de otros, es decir, se deben realizar justo antes de al que preceden, sin ningún ejercicio intercalado entre ellos.
- Ejercicios preparadores: existen ejercicios que se deben realizar antes de un cierto ejercicio en el mismo día pero no es necesario que sea inmediatamente antes, es decir, pueden haber ejercicios intercalados entre ellos.
- Ejercicios nivel + 1: modela la evolución en el nivel de dificultad de un ejercicio según pasan los días. Para cada ejercicio podremos subir un nivel N + 1 solo si ese ejercicios se ha realizado en algún día anterior en el nivel N.

2 Modulación

2.1 Nivel básico

El nivel básico de nuestro programa se encarga de asignar los ejercicios a diferentes días de la rutina de manera que en como mucho 15 días se realizaran los objetivos de dificultad de los ejercicios seleccionados por el usuario. Además, se contempla que un ejercicio dado pueda tener un ejercicio preparador.

2.2 Extensión 1

La primera extensión del programa consiste en poner que los ejercicios puedan tener más de un ejercicio preparador.

2.3 Extensión 2

Esta segunda extensión abarca toda la primera y le añade la funcionalidad de que un ejercicio pueda tener un ejercicio precursor.

2.4 Extensión 3

Esta versión del programa tiene en cuenta la que como mucho se pueden poner 6 ejercicios en un mismo día.

2.5 Extensión 4

La última versión no es compatible con la tercera ya que tiene una función similar, sin embargo, en vez de limitar la cantidad de ejercicios en un día, se limita el tiempo total del día (i.e. el sumatorio del tiempo de todos los ejercicios).

3 Dominio

3.1 Variables

Para realizar nos que se nos propone el problema, hemos creado varias variables:

- ejercicio: tipo que indica que el objeto es un ejercicio.
- nivel: tipo que indica que el objeto es el nivel con el que se realizará un ejercicio.
- día: tipo que indica que el objeto es un día.

3.2 Predicados

- (preparador ?x - ejercicio ?y - ejercicio): el ejercicio x es preparador del ejercicio y.
- (precursor ?x - ejercicio ?y - ejercicio): el ejercicio x es precursor del ejercicio y.
- (dificultad ?d - nivel ?x - ejercicio): d es el nivel con el que se realizó por última vez el ejercicio x.
- (objetivo ?n - nivel ?x - ejercicio): n es el nivel objetivo del ejercicio x.
- (asig ?x - ejercicio ?d - dia): d es uno de los días en el que se realizará el ejercicio x.
- (conseguido ?x - ejercicio): se ha alcanzado el nivel objetivo deseado con el ejercicio x.
- (sig ?n1 - nivel ?n2 - nivel): *n2* es el siguiente nivel de *n1*.
- (ant ?d1 - dia ?d2 - dia): *d1* es día anterior a *d2*.
- (primer_dia ?d - dia): d es el primer día del plan de entrenamiento.

3.3 Funciones

- (nivel ?e - ejercicio): tiene el nivel del ejercicio e.
- (objetivo ?e - ejercicio): tiene el nivel objetivo de e.
- (ultimo_dia ?e - ejercicio): tiene el último día que se ha realizado el ejercicio e.
- (orden_dia ?d - dia): tiene el orden de los días.
- (dia_actual ?d - dia): tiene el día actual.
- (ultimo_ejercicio): tiene el último ejercicio que se ha realizado.
- (num_ej ?e - ejercicio): tiene el número del ejercicio e.
- (capacidad_dia ?d - dia): tiene la cantidad de ejercicios asignados en el día d.

- (minutos_dia ?d - dia): tiene la duración de todos los ejercicios en el día *d*.
- (minutos_ejercicio ?e - ejercicio): tiene la duración del ejercicio *e*.

3.4 Acciones

- sin-preparador:
  - Parámetros: un ejercicio, un día y dos niveles.
  - Pre-condiciones: dicho ejercicio no tiene ningún preparador, el día anterior se realizó el mismo ejercicio con dificultad -1 a la actual y no se ha alcanzado aún el nivel objetivo deseado con dicho ejercicio.
  - Efecto: se ha asignado dicho ejercicio a un día y con un nivel específicos.
- con-preparador: Ésta es igual que la acción *sin-preparador* pero se comprueba además que el ejercicio del parámetro tiene un preparador y que este se ha realizado con anterioridad el mismo día.
- alcanzado:
  - Parámetros: un ejercicio y un nivel.
  - Pre-condiciones: la última vez que se realizó el ejercicio, se hizo con la dificultad nivel objetivo impuesto por el usuario.
  - Efecto: se ha conseguido alcanzar el nivel deseado en dicho ejercicio.
- completado: Esta acción es equivalente a la anterior y la sustituye a partir de la implementación de la extensión 2.
  - Parámetros: un ejercicio.
  - Pre-condiciones: el nivel del ejercicio es el mismo que el impuesto por el usuario (nivel objetivo).
  - Efecto: se ha alcanzado el nivel objetivo de dicho ejercicio.
- asignar: Esta acción se ve modificada según la versión que estemos utilizando. El objetivo principal es asignar un ejercicio a un día e incrementarle el nivel con el que éste se realiza. Se tiene en cuenta que dicho ejercicio no se haya ya asignado a ese mismo día, que los preparadores y predecesores cumplan las restricciones y que no se haya conseguido ya, por supuesto, el nivel objetivo del ejercicio en cuestión.
- dia_siguiente: Esta acción se ve ligeramente modificada según la versión que estemos utilizando. Así pues, el objetivo principal de ésta es simplemente estético en el momento de leer la salida del programa. Hace que el output esté ordenado según los días en orden ascendiente y no, por ejemplo, por ejercicios como nos sale en las versiones *nivel básico* y *extensión 1*.

4 Resolución

4.1 Objetos

Para los objetos de la resolución tenemos que todos los problemas siempre tendrán los mismos objetos. Los objetos serán un conjuntos de ejercicios $e_1...e_n$, un conjunto de niveles $n_1..n_{10}$ y un conjunto de días $d_1..d_{15}$. La única diferencia que puede haber entre los problemas es la cantidad de ejercicios ya que para todo problema siempre habrán 10 posibles dificultades y 15 posibles días.

4.2 Estado inicial

El estado inicial tiene una parte comuna para todas las extensiones y una aplicación. Para empezar, según las necesidades de la extensión se añade el *precursor* y los *preparador* de cada ejercicio. Después de añade la *dificultad* de cada ejercicio. También se añade para ciertos ejercicios un nivel *objetivo*. Para poder establecer el orden de días se declaran predicados *ant* y para saber el orden de los niveles de dificultad se declaran predicados *sig*, también se declara el predicado *primer_día* con el fin de establecer cual es el primer día de la rutina. Por otra parte tenemos nuevos predicados para las extensiones 3 y 4. Para la extensión 3 tenemos que tener en cuenta la cantidad de ejercicio que se realizan durante el día, es poder que usando FF-Metrics inicializamos predicados de *capacidad* con un valor de 0 que tendrá que ir aumentado con las asignaciones. Por otro lado, en la extensión 4 tenemos que tener en cuenta la duración en vez de la cantidad de ejercicios y para poder resolver el problema hemos declarado dos predicados distintos, uno seria equivalente al de la extensión 3, en el cual inicializamos los minutos que hay en un día con el predicado *minutos_dia*, además, por cada ejercicio añadimos el predicado *minutos_ejercicio* con el fin de guardar la duración de dicho ejercicio.

4.3 Estado final

El estado final, al igual que los objetos tiene la misma forma en todos los problemas, en esta solo ponemos una clausula *and* con varios predicados *conseguido* e_i donde e_i es el ejercicio que se quiere mejorar, esto para varias i.

5 Desarrollo

Para el desarrollo de los distintos problemas hemos optado por un diseño incremental basado en prototipos siguiendo el guión del enunciado para las extensiones 1 y 2; en cuanto a la 3 y la 4, las dos parten del diseño del básico + ext2, pero son incompatibles entre sí.

A parte de los modelos en PDDL hemos desarrollado un generador de problemas capaz de hacer problemas con características aleatorias a partir de los parámetros proporcionados. Para generar un problema se han de introducir los siguientes parámetros:

- *outFile*: indica el nombre del archivo, donde se debe guardar el problema.
- *seed*: indica el seed aleatorio para la generación de números aleatorios del problema.
- *domain*: nombre del dominio con el que se esta trabajando.
- *nExercices*: total de ejercicios a generar como objetos.
- *maxPredecesors*: el máximo numero de predecesores que puede tener un ejercicio.
- *maxPrecursors*: el máximo numero de precursores que puede tener un ejercicio.
- *duration*: valor booleano que indica si el ejercicio no tiene duración, si es de tipo ext3 o si es de tipo ext4.

El problema generado tiene un porcentaje de ejercicios con objetivo por defecto del 15%, sin embargo este porcentaje se puede cambiar fácilmente. El generador esta escrito en C++, hemos escogido este lenguaje simplemente por comodidad.

6 Juegos de Prueba

En esta sección evaluamos los diferentes juegos de prueba que hemos utilizado para probar nuestro programa, cada prueba esta enfocada a una versión distinta del programa, según las extensiones especificadas. Cada problema esta dentro de la carpeta *jp*. La traza de resolución de los problemas se ha separado en archivos correspondientes en la carpeta *log*. Para cada juego de pruebas hay una traza distinta.

6.1 Prueba 1

- Versión: Básico
- Ubicación problema: jp/basic_1.pddl
- Ubicación traza: log/basic_1.log
- Objetivo: El objetivo de este primer juego de pruebas es comprobar el funcionamiento general del sistema con la versión básica de la aplicación. Por cada ejercicio que ha pedido el usuario se ha realizado la acción de realizado, lo que nos indica que todos los objetivos se han cumplido.

6.2 Prueba 2

- Versión: Básico
- Ubicación problema: jp/basic_2.pddl
- Ubicación traza: log/basic_2.log
- Objetivo: Este segundo juego de pruebas se encarga es más sencillo ya que no hay ningún ejercicio preparador. En el resultado de este juego de pruebas se puede verificar que no se hace ninguna vez la acción encargada de asignar ejercicios con preparadores. Ya que por la naturaleza del problema es incompatible. Sin embargo, al igual que la prueba anterior se dan los resultados esperados.

6.3 Prueba 3

- Versión: Básico
- Ubicación problema: jp/basic_3.pddl
- Ubicación traza: log/basic_3.log
- Objetivo: En este caso hacemos una prueba extrema ya que hemos puesto que cada ejercicio i depende del anterior $i-1$, también hemos puesto que cada ejercicio empieza en el nivel 1 y hemos quitado ejercicios con objetivos, sin embargo como era previsto, el programa nos ha demostrado que no se encuentra ninguna solución con los parámetros proporcionados, es normal en este caso ya que le hemos puesto muchas restricciones.

6.4 Prueba 4

- Versión: Extensión 1
- Ubicación problema: jp/ext1_1.pddl
- Ubicación traza: log/ext1_1.log
- Objetivo: Para este juego de pruebas queremos verifica simplemente que el programa funciona cuando un ejercicio tiene mas de un preparador, para mostrar un caso extremo, lo que hemos hecho es hacer que un ejercicio específico necesite a todos los demás como preparadores

6.5 Prueba 5

- Versión: Extensión 2
- Ubicación problema: jp/ext2_1.pddl
- Ubicación traza: log/ext2_1.log
- Objetivo: Esta prueba comprueba que efectivamente si añadimos como mucho 1 predecesor el planning generador por nuestro sistema se genera adecuadamente. En este caso solo añadimos un único preparador.

6.6 Prueba 6

- Versión: Extensión 3
- Ubicación problema: jp/ext3_1.pddl
- Ubicación traza: log/ext3_1.log
- Objetivo: Para poder comprobar la nueva extensión, hemos utilizado una prueba anterior y la hemos adaptado para que el operador de acciones pueda tener en cuenta la cantidad de ejercicios que se hacen diariamente.

6.7 Prueba 7

- Versión: Extensión 4
- Ubicación problema: jp/ext4_1.pddl
- Ubicación traza: log/ext4_1.log
- Objetivo: La ultima prueba que hemos hecho tiene una gran similitud a la anterior pero con un nivel de dificultad mayor, es por eso que podemos ver en la traza de salida que el tiempo de búsqueda es superior, sin embargo da resultados esperados, equilibrando cada día y limitando el tiempo que dedica la rutina diariamente.

7 Conclusiones

Durante esta práctica hemos aprendido otra manera de ayudar a nuestros fieles clientes, *CoachingPotato*, nos hemos familiarizado con el comportamiento de *Fast Forward* y, para nuestra sorpresa, nos ha sorprendido lo similiar que es *PDDL* al lenguaje que aprendemos en la asignatura *LI*, *Prolog*.

Además, las veces que se nos ha quedado colgada la terminal a causa de la terrible implementación que habíamos realizado, nos ha servido de gran ayuda recordar el funcionamiento del algoritmo de *Hill Climbing* para encontrar como solventar el problema.