Estructures de dades i algorismes

Algorisme principal generador d'horaris

Descripció

Hem remodelat l'algorisme per la segona entrega:

L'algorisme segueix estant compost de 3 funcions principals. La primera (**generateSchedule**) prepara les dades per a generar l'horari. Per a fer-ho, comença assignant totes les possibles aules, hores i dies possibles per a cada Lecture (un grup pot tenir múltiples lecture), tenint en compte les restriccions unàries, de manera que eliminem qualsevol combinació que no cumpleixi les restriccions unàries. Anomenem a aquest procés **"primer filtrat"**, que estalviarà feina al forward checking. En aquesta segona entrega, hem millorat en gran mesura l'**heurística** que utilitzàvem per optimitzar la selecció d'assignacions: durant el primer filtrat mantenim el recompte total de possibles assignacions que té cada Lecture i a més també contem, per cada combinació de dia + hora + aula, amb el número de Lectures que consideren aquella combinació com una possible assignació. D'aquesta manera, el que farem serà **a cada pas inserir la Lecture que té menys possibles assignacions, és a dir, la que té menys possibilitats de ser inserida en un futur**. I aquesta Lecture la inserirem en la combinació de dia + hora + aula que està menys referenciada, és a dir, la combinació que menys Lectures consideren com a una possible assignació. D'aquesta manera, **per a cada inserció minimitzem el nombre de possibilitats que restem a la resta de Lectures**. Al fer això, augmentem en gran mesura la probabilitat de no haver de tirar enrere en un futur.

Un cop generades les possibles assignacions per a cada Lecture, les passem a la segona funció (**backjumping**). Aquesta, com hem dit, agafa la Lecture que té menys possibles assignacions i la insereix a l'horari en el dia hora i aula que menys lectures estan considerant com a possibilitat. Un cop fet això, actualitzem les possibles assignacions que tenen la resta de Lectures encara no inserides, mitjançant **forward checking**. El que farem serà eliminar totes les possibles assignacions de cada Lecture que no compleixin les restriccions un cop feta aquesta última inserció. Conforme anem eliminant, guardem totes les dades esborrades, de manera que si no trobem un horari factible, podrem desfer els canvis i intentar una altra combinació (**backtracking**). Mentres borrem opcions que no compleixen les restriccions, mirem que no hi hagi cap Lecture que es quedi sense cap possible assignació, ja que llavors seria impossible inserir-la a l'horari. Si això passa, parem el forward checking, restaurem els canvis, fem backtracking i intentem una altra combinació.

Si després del forward checking totes les Lectures segueixen tenint possibilitats de ser inserides, continuarem fins a generar l'horari o, en cas de que no es pugui generar, esgotar les possibilitats (de fet en aquesta segona entrega hem instaurat un timeout que para al cap de 30 segons de intentar generar l'horari sense èxit).

En totes aquestes crides, per a emmagatzemar les possibilitats de cada lecture ho fem en un Map que té com a key la lecture i com a value un objecte de classe posAssig, que és un Map que per cada dia té un Map de hores que té un Set de totes les aules possibles a cada hora.
El numero de possibilitats total de cada lecture és un Map que té com a key l'identificador de la lecture i com a value el número de possibilitats.
Pel número de referències de cada dia + hora + aula també utilitzem un Map, on l'identificador és una String en un format que hem triat (hem creat funcions per codificar i descodificar en aquest format), i el value el número de lectures que consideren aquella combinació com a possible.

Cal dir que en aquesta segona entrega, ja no passem tota la informació del Ctrl Domini pels paràmetres a la funció generadora d'horaris. Hem creat una classe Environment que els emmagatzema, i la funció generadora en crida una instància per referència.

Altres estructures de dades

L'estructura de dades que guarda l'**horari** és un mapa de clau un String, que fa referència a una Aula, i de valor una matriu, que representa els 5 dies de la setmana i les 12 hores possibles, de manera que guardem l'horari de cada aula.

Per a guardar les Aules, Assignatures, Grups i Lectures a la classe Environment, utilitzem Maps que tenen com a key l'identificador de cada objecte, i el valor és el propi objecte. Les assignatures es relacionen amb grups i aquests amb lectures utilitzant Strings que els identifiquen. Així **evitem redundància** a memòria.

Ens basem en el mateix per a guardar les restriccions. Les unàries es guarden en un Map on la key és el Grup i el valor un Set de restriccions, d'aquesta manera podem accedir directament al es de cada grup en **temps constant** (encara no hem trobat cap escenari on el Hash code es repeteixi per algun objecte, de manera que **no tenim col·lisions** en cap dels Maps). Les nàries segueixen la mateixa lògica.