Solución al UCSP (university class scheduling problem) con satisfacción de restricciones

Carlos Andrés Páez Chitiva, Tomás Trejos Gil

Proyecto de Inteligencia Artificial

Diciembre de 2024

Resumen

Este informe presenta el desarrollo de un modelo de inteligencia artificial con el propósito de programar de manera automática los horarios de los cursos dados en una institución educativa. Para lo anterior utilizando un modelo de satisfacción de restricciones y la técnica de 'Local Search', además se evaluaran los resultados para el potencial uso de esto modelo un entorno de producción

Palabras clave: CSP, Local Search, IA, UCSP.

1. Introducción

Se plantea la realización de un algoritmo de inteligencia artificial que resuelva un UCSP modelado como un CSP. Este es un problema de asignación multidimensional que toda institución educativa debe resolver según sus necesidades. Lo que se busca lograr es asignar a cada curso: El o los días en que se da el curso, el horario, el profesor y el salón correspondiente. Para la asignación de lo antes mencionado se deben satisfacer ciertas restricciones, como que ningún profesor este asignado más de un curso al mismo tiempo, que ninguna salón se asigne a más de una curso al mismo tiempo, etc.

En la práctica, este problema puede comprender miles de cursos, miles de estudiantes, cientos de profesores, cientos de salones y otros recursos asociados según la necesidad de la institución. Además, es importante tener en cuenta que este problema se cataloga como un NP completo (es decir, no se conoce ningún algoritmo de tiempo polinomial para resolver el problema), lo que es una razón para determinar que este es un problema de ingeniería complejo.

Es un problema importante a abordar debido a que es una actividad administrativa importante, regular y compleja en la mayoría de las instituciones educativas, que toma bastante tiempo y se deben tener en cuenta una gran cantidad de variables. Automatizar este proceso dentro de una institución elimina la necesidad de contratar personal para realizar estas tareas o aligerar la carga del personal ya existente dentro de la institución, reduciendo costos y eliminando del proceso la posibilidad de un error humano que puede tener repercusiones notables para los miembros de la institución, especialmente para los estudiantes.

Abordamos el problema usando IA puesto que es un problema que en la actualidad es resuelto por ciertas capacidades de razonamiento del ser humano que son replicables haciendo uso de algoritmos de IA, en este caso el problema encaja dentro del modelo general de un problema de satisfacción de restricciones, lo que nos permite partir una base teórica sobre la cual realizar una formulación del problema.

2. Formulación del Problema

Desde la perspectiva de la Inteligencia Artificial, es importante definir lo que el usuario final proporciona (Entradas) y la respuesta que este recibe (Salida). Las entradas corresponde a una la lista de cursos para los cuales se quiere resolver la asignación de horarios, de cada curso es necesaria la siguiente información:

- **Nombre del curso:** Corresponde a un identificador único.
- **Asignatura:** Nombre de la materia según el plan de estudios.
- **Semestre:** Concierne al semestre en el cual el curso se debería ver según el plan de estudios.
- **Cantidad de cupos:** Se refiere a la máxima cantidad de estudiantes que pueden tomar el curso.
- **Profesores:** Compete a una lista de profesores, de manera tal que cada profesor tenga un identificador único.
- **Jornada:** Define la hora del día en que la materia debe darse (Diurna o nocturna) si es que se desea restringir.
- **Sesiones:** Concuerda con la cantidad de veces que se ve el curso a la semana, por cuanto tiempo y el tipo de salón en que se debe ver la sesión (Ej: Laboratorio, salón de informática).

Además de los cursos, se debe ingresar:

- Lista de profesores: Cada profesor debe tener su identificador único junto con el horario en que esta disponible para dar clase.

- Lista de salones: Cada salón debe tener un identificador único, el tipo y el aforo del mismo.
- Lista de jornadas académicas con sus respectivos horarios.
- Conjunto de los días disponibles para la asignación.
- conjunto de horas en las que se puede dar clase para cualquier día.

Finalmente, el usuario tiene la opción de ingresar unas condiciones iniciales, es decir, una asignación parcial del horario para algunos de los cursos ingresados, y una lista de duplas que indique que cursos no se pueden cruzar, cada dupla se compone por dos cursos (Sus identificadores únicos), que son los que no se pueden cruzar.

Luego de procesar los datos y dar solución al problema, se obtendrá una salida para cada curso, la cual corresponde a una lista, donde cada elemento de la lista representa una sesión del curso y tiene la siguiente estructura:

- **Día:** Día de la semana en que se da el curso.
- **Hora de inicio:** Hora en formato militar en que inicia la clase.
- **Hora de fin:** Hora en formato militar en que finaliza la clase.
- **Salón:** Aula en que se da la clase.
- **Profesor:** Profesor asignado para dar el curso.

Teniendo en cuenta las entradas y las salidas del problema, se ha identificado que este puede ser modelado como un problema de satisfacción de restricciones, en el cual se definen las variables, el dominio, las restricciones y la solución.

2.1. Variables

Las variables en un problema de satisfacción de restricciones son aquellas que representan los elementos desconocidos del problema y que deben ser asignados a un valor dentro de un dominio definido. Estas variables están sujetas a un conjunto de restricciones que especifican las condiciones que deben cumplirse para que la solución sea válida.

Para este problema en concreto, las variables conciernen a las sesiones de todos los cursos, y cada sesión tiene unas sub-variables: día, hora de inicio, hora de fin, salón y profesor.

2.2. Dominio

El dominio hace alusión a los posibles valores que se le pueden asignar a una variable. Para este caso, el dominio de una sesión se define como el producto cartesiano de los dominios de las sub-variables:

- Hora de inicio: Rango de horas de clase del día definidas por el problema, es decir, desde que hora se empieza a dar clase y cual es la hora en que ya no se dan clases.
- Hora de fin: Rango de horas de clase del día definidas por el problema, es decir, desde que hora se empieza a dar clase y cual es la hora en que ya no se dan clases.
- Salón: Conjunto de salones disponibles definidos por el problema.
- Profesor: Conjunto de profesores definidos por el problema.

2.3. Restricciones

Hacen referencia a todas limitaciones y condiciones que el problema debe cumplir para considerarse resuelto, para este problema, las restricciones son las siguientes:

- Para la asignación de una sesión la combinación de las sub-variables día, hora de inicio y hora de fin debe pertenecer al conjunto de horarios disponibles del profesor asignado.
- Para las asignaciones de dos sesiones del mismo curso, la sub-variable día debe ser diferente.
- Para la asignación de una sesión, la sub-variable salón debe ser del tipo de salón del curso, si aplica.
- Para la asignación de una sesión, la sub-variable salón debe tener capacidad mayor o igual a la cantidad de cupos del curso.
- Para la asignación de una sesión, el rango entre la hora de inicio y la hora de fin debe estar dentro del rango de la jornada del curso, si tiene.
- Para la asignación de una sesión, el profesor asignado debe ser el mismo para todas las sesiones del mismo curso.
- Si para las asignaciones de dos sesiones de cursos sobre diferentes asignaturas que se ven en el mismo semestre la sub-variable día es igual, entonces se debe cumplir que el rango entre la hora de inicio y la de fin, de cada sesión, no se puede cruzar si es que no existen mas cursos de esas asignaturas que no se cruzen.
- Si para las asignaciones de dos sesiones de cursos sobre diferentes asignaturas que no se pueden cruzar de acuerdo a la definición del problema la sub-variable día es igual, entonces se debe cumplir que el rango entre la hora de inicio y la de fin, de cada sesión, no se puede cruzar.
- Si para las asignaciones de dos sesiones de cursos la combinación de las sub-variables día y profesor son iguales, entonces se debe cumplir que el rango entre la hora de inicio y la de fin, de cada sesión, no se puede cruzar.

- Si para las asignaciones de dos sesiones de cursos la combinación de las sub-variables día, hora de inicio y hora de fin son iguales, entonces se debe cumplir que el salón sea diferente.

2.4. Solución

La solución se define como todas las asignaciones a las variables que satisfacen el conjunto de restricciones.

Modelar este problema como un problema de satisfacción de restricciones nos permitirá hacer uso de algoritmos y técnicas conocidas para la resolución de los mismos.

3. Solución

Para solucionar este problema se utilizo un enfoque orientado a objetos para modelar el CSP; las clases configuradas para guardar la información ingresada por el usuario fueron: 'Professor', 'Classroom', 'Course' y 'Session', y para solucionar el CSP fueron: 'CSP', 'UCSP', 'CSPSolver' y 'LocalSearch', todas estas abstracciones a fin de generalizar la solución y reutilizar código.

La técnica utilizada fue Búsqueda Local, que consiste en lo siguiente:

1. Tomar una asignación inicial cualquiera de manera aleatoria.
2. Mientras el problema no este resuelto realizar lo siguiente:
    - Seleccionar si es posible la variable que incumpla la mayor cantidad de restricciones, de lo contrario escoger una de manera aleatoria que incumpla por lo menos una restricción.
    - Elegir un valor que viole la menor cantidad de restricciones. Para esto se pueden tener en cuenta todas las posibles asignación y elegir la que menos cantidad de restricciones incumpla o plantear una heurística que determine esto.
    - Revisar si el problema ya esta resuelto, sino, repetir el proceso desde la selección.

Una de las principales razones para la adopción de esta técnica en este proyecto es debido a que en la literatura consultada, esta es recurrentemente utilizada para solucionar este tipo de problema en especifico, esto a razón de que el algoritmo es mucho más rápido en comparación a otros, como lo es backtracking y sus distintas variaciones, pero hay que tener en cuenta que esta técnica es subóptima en muchas de las ocasiones, ya que de acuerdo a la asignación inicial aleatoria obtenida puede ser imposible hallar una solución.

La implementación de este algoritmo tuvo varias complicaciones, lo que llevo a implementar diferentes estrategias y técnicas para que este lograra encontrar una solución. En un principio se plantean las siguientes estrategias para adaptar el algoritmo de búsqueda local a este tipo de problemas en concreto:

- Al momento de seleccionar una variable, a cada una de estas se le verifica la cantidad de restricciones que incumple, esto a fin de organizar las variables según la cantidad de restricciones que incumple. La finalidad del ordenamiento es que si la nueva asignación de la variable que más incumple restricciones no mejora la solución, se pueda intentar con la variable que le sigue.
- Teniendo en cuenta el anterior item, se tuvo en cuenta que ocurre si ninguna variable mejora, si esto ocurre, a la mitad de las variables se les da una asignación de manera aleatoria, con el propósito de realizar un cambio que pueda ser significativo para hallar una solución. Esta idea surge a partir de los algoritmos genéticos.

Una vez implementadas dichas estrategias, y tras realizar algunas pruebas con unas entradas previamente establecidas y con una solución posible, encontramos dos casos extremo que impedían encontrar una solución con la implementación planteada hasta el momento. Estos fueron:

- La primera situación que impedía la resolución del problema se presentaba debido a que el algoritmo entraba en un bucle en el cual siempre mejoraba alguna variable, pero no lograba una solución completa, lo que impedía que se ejecutara la parte del código que tras no encontrar variable que mejorara reasignaba de forma aleatoria una mitad de las variables escogidas de forma aleatoria.
- Otra situación que impide que se solucione el problema se da cuando la nueva asignación de una variable aumenta las restricciones de otra y luego esta es escogida en la siguiente iteración entrando en un bucle infinito en el que las dos variables se afectan la una a la otra.

Como solución parcial a estos casos extremos, se decidió que al momento de darle una asignación a la variable seleccionada, si esta no puede mejorar entonces se elige otra asignación diferente a la que tenia antes pero con igual cantidad de restricciones, esto a fin de aumentar la exploración del algoritmo y reducir la posibilidad de quedar atascado en bucles.

Finalmente, luego de implementar las modificaciones planteadas y tras realizar algunas pruebas con una entrada de datos previamente establecida, el algoritmo es capaz de dar solución al UCSP de manera satisfactoria, es decir, asignando la totalidad de las variables de manera tal que satisfagan las restricciones establecidas a partir de la entrada de datos. Más sin embargo, el algoritmo aún puede fallar en encontrar solución cuando cae en el segundo caso extremo mencionado anteriormente. Con esto en mente, se añadió un numero máximo de iteraciones a el algoritmo de búsqueda local, por lo que el algoritmo indica que no puede encontrar

la solución luego de llegar a la máxima iteración con una asignación de variables que no cumple con todas las restricciones.

En el siguiente repositorio de github puedes encontrar el código que implementa todo lo mencionado durante esta sección.

4. Análisis de Resultados y Aplicaciones

Una vez implementadas correctamente las estrategias arriba mencionadas, se prosiguió con una fase de pruebas de efectividad en la cual se hacen uso de tres entradas de datos distintas preparadas con anterioridad, las cuales corresponden a:

- La primera entrada corresponde a un problema en el que se tienen un total de 6 variables o sesiones de cursos a asignar para dos profesores distintos con horarios cortos y restrictivos.
- Para la segunda entrada se tienen 27 sesiones a asignar (variables), 8 sesiones fijas (asignaciones iniciales), y un total de 7 profesores con horarios completamente disponibles, es decir en todos los días de la semana y en la mayor cantidad posible de horarios en un día, lo que da lugar a que el problema tenga solución trivial.
- Finalmente, se reutilizan las sesiones y los profesores de la anterior entrada de datos pero restringiendo los horarios de manera tal que el problema no tenga una solución trivial para el ser humano.

Como medida de evaluación tomamos en cuenta la corrección del algoritmo, entendiéndose corrección como la frecuencia con la que el algoritmo encuentra una solución dado un máximo número de iteraciones. Para medir la corrección efectuamos un total de 1000 ejecuciones para cada una de las entradas de datos mencionadas anteriormente, con un máximo de 100 iteraciones. Los resultados se visualizan en la tabla 1,

Datos N°	**N° de ejecuciones satisfactorias / Total**
1	371 / 1000
2	984/1000
3	378 / 1000

Tabla 1: Resultados

Aunque al mirar a los números los resultados pueden parecer pesimistas, en la practica una corrección del 37 % es decente en cuanto a que con que el algoritmo sea capaz de encontrar una sola solución es suficiente, y si al ejecutarlo diez veces en al menos tres de ellas obtenemos una solución, dado la baja complejidad temporal del algoritmo con respecto a otros, es completamente factible el uso de este algoritmo para una situación real. Lo anterior sin tener en cuenta que esta es solo una prueba de concepto y que aun hay muchas optimizaciones en términos del uso inteligente de la memoria (estructuras de datos) y determinación de heuristicas que pueden disminuir drásticamente el tiempo que toma encontrar una solución.

5. Conclusiones

Tras llevar a cabo el desarrollo de esta prueba de concepto, y observar los resultados obtenidos, podemos evidenciar que es posible dar solución a un UCSP abordándolo desde el campo de la inteligencia artificial, con una cantidad considerable de variables y restricciones, lo que hace que valga la pena seguir investigando y trabajando sobre este desarrollo de software ya que resulta de gran utilidad para instituciones educativas de toda índole.

También evidenciamos, que la formulación del problema y la solución en este informe pueden ser desarrolladas con mayor profundidad, en especial la solución con enfoque en el algoritmo de búsqueda local, a la cual es posible hacerle cambios tanto en las estrategias que puedan ayudar de manera heuristica a aumentar la eficiencia o la corrección, y en cuanto a detalles de implementación como los relacionados al manejo de la memoria y el uso inteligente de estructuras de datos que contribuyen a reducir el tiempo que toma encontrar una solución, esto pensando en la posibilidad de llevar este algoritmo a producción para que una o más instituciones puedan beneficiarse de el.

Finalmente, como investigadores y desarrolladores de software nos complace hacer uso de técnicas de diseño e implementación de software del campo de la inteligencia artificial para plantear y dar solución a problemas que no solo afectan a otras personas, sino que directamente nos afectan como estudiantes universitarios.