Proyecto Final del Curso

Algoritmos Avanzados de Búsqueda y Optimización

Profesores: Michel Pedrera y Pío Dos Santos

Noviembre 2025

Alexia Aurrecochea, Mercedes Barrutia, Sofía Craigdallie y Paula Blasco

Primera propuesta: Optimización de horarios de clases.

Descripción del problema

Generar horarios semanales que asigne para cada curso una franja horaria, salón y docente, cumpliendo con restricciones a determinar.

Algunas de las restricciones discutidas implican; la minimización del tiempo total presencial del estudiante, y la limitación de turnos con una carga diaria máxima. Se considera regular el tiempo de transición entre salones (o la penalización si hay cambio de edificio). Se podrían consultar las preferencias y disponibilidades de los docentes y evitar superposición entre materias que se cursan en el mismo semestre.

Motivación y relevancia

Este problema es considerado de alto valor académico por el equipo, ya que la planificación óptima de horarios es un desafío real en instituciones educativas que puede ser remodelado y aplicado a otros entornos.

La motivación surge de la posibilidad de aplicar y comparar distintos enfoques de optimización, desde algoritmos de búsqueda exacta hasta heurísticas y metaheurísticas, evaluando su desempeño en términos de eficiencia y calidad de las soluciones obtenidas en un fenómeno tangible. Representa una oportunidad de aprendizaje aplicada, con potencial de uso en futuro administrativo de optimización de recursos.

Primer modelo formal y definición de criterios de éxito

Para el primer modelo formal es necesario establecer varios conjuntos:

- C (Conjunto de cursos)
- P (Conjunto de Profesores)
- R (Conjunto de aulas)
- T (Conjunto de franjas horarias discretas, día y hora)
- G (Conjunto de grupos de estudiantes)

Los parámetros a asignar:

- dura(c) $\in$ Z+ : número de franjas consecutivas que va a ocupar el curso C
- disp(p,t) $\in$ {0,1} : disponibilidad del profesor **p** en franja **t**
- cap(r) : capacidad del aula **r**
- grupos(c) $\subseteq$G: grupos que asisten al curso **c**
- igual(r1,r2) $\in$ {0,1}: da 1 como resultado si r1, r2 están en el mismo edificio
- cargamax(g): carga máxima de franjas por día para el grupo **G**
- **w1,w2,w3:** son los pesos para componentes del objetivo (definidos por el equipo y ajustables)

Variables de decisión:

- x(c,t,r,p) $\in$ {0,1}: da 1 si el curso **c** se programa empezando en la franja **t**, con profesor **p** y en el aula **r.** Si dura(c) > 1, x implica ocupación de **t**, **t+1, …, t + dura(c) - 1**.

Restricciones (obligatorias):

1. Asignación: cada curso debe ocupar exactamente una posición(inicio): ∑t,r,pxc,t,r,p=1 $\forall$c.
2. Disponibilidad profesor: si x(c,t,r,p) = 1 entonces disp(p,t')=1 para todas las franjas **t'** que ocupa **c.**

3. Aula válida: no permitir $x(c,t,r,p)=1$ si $cap(r) <$ alumnos de **c** o si **r** no cumple requisitos técnicos de **c.**
4. No solapamiento aula: para cada **r** y franja **t′**, a lo sumo un curso ocupa **r**.
5. No solapamiento profesor: para cada **p** y **t′**, a lo sumo un curso impartido por **p**.
6. No solapamiento por grupo: para cada $g \in G$ y franja **t′**, un grupo no puede tener más de una clase simultánea.
7. Carga diaria máxima: para cada **g** y día **d**, número de franjas ocupadas ≤ cargamax(g)

Restricciones (penalizables):

1. Cambio de edificio entre las clases consecutivas de un mismo grupo: introducir variable **y(g,t)** $\in \{0,1\} = 1$ si hay cambio problemático en la frontera **t** → **t+1.**
2. Huecos(ventanas) en la jornada de un grupo: contar los huecos y penalizarlos.
3. Preferencias docentes no cumplidas: penalización proporcional

Función objetivo:

Minimizar una suma ponderada de la calidad de horario y penalizaciones:

min (w1 x TiempoPresencialPromedio + w2 x PenalizaciónCambiosEdificio + w3 x Ventanas)

Este primer modelo proporciona una base sólida donde se aborda el problema de la optimización de horarios con un enfoque preciso y flexible. Permite integrar restricciones reales del entorno académico y poder adaptar fácilmente las nuevas condiciones sin rediseñar el sistema. La formulación aplicada facilita la comparación entre métodos heurísticos y exactos.

Criterios de éxito:

El éxito del modelo y de los algoritmos implementados se evaluará en base a:

1. Calidad de la solución: Número total de conflictos (superposiciones, violaciones de disponibilidad), tiempo total presencial promedio por estudiante, penalización total por cambio de edificio.
2. Eficiencia computacional: Tiempo de ejecución y convergencia del algoritmo y escalabilidad con el número de cursos y docentes.
3. Flexibilidad y robustez: Capacidad del modelo de adaptarse a cambios en la disponibilidad de docentes o en la cantidad de cursos sin rediseñar toda la solución y posibilidad de incorporar preferencias o restricciones adicionales sin pérdida de estabilidad.
4. Comparación de enfoques: Se evaluarán distintos métodos de búsqueda y optimización comparando desempeño y calidad de las soluciones obtenidas.

Una solución será considerada exitosa si logra: Asignar todos los cursos sin conflictos, mantener el tiempo total presencial dentro de los límites definidos, reducir significativamente la penalización por cambios de edificio y ejecutarse en tiempos razonables para una instancia representativa.

Algoritmos preliminares a considerar

Para resolver este problema se podría aplicar algoritmos de búsqueda y optimización combinatoria, utilizando técnicas como Backtracking, Genetic Algorithm, Simulated Annealing o incluso una estrategia que combine ambos enfoques. En este caso el problema puede dividirse en dos etapas: una búsqueda exacta para asignar docentes, aulas y franjas horarias cumpliendo todas las restricciones, y una optimización heurística para mejorar la calidad del horario.

Segunda propuesta: Juego del gato y el ratón.

Descripción del problema

En un tablero limitado por obstáculos se encuentran dos agentes: gato y ratón. En cada turno se moverán alternadamente:

- el ratón, que busca evadir al gato y alcanzar una meta o sobrevivir durante un número limitado de turnos.
- el gato, que intenta capturar al ratón en el menor tiempo posible, reduciendo la distancia entre ambos hasta ocupar la misma celda/posición.

Motivación y relevancia

El problema del gato y el ratón constituye un entorno de alta complejidad estratégica que combina elementos de búsqueda, planificación, y toma de decisiones en escenarios adversos. Además, su estructura modular facilita la extensión hacia escenarios más complejos, como juegos multiagente (con más personajes), entornos parcialmente observables (cómo en videojuegos) o simulaciones con aprendizaje por refuerzo (en el que cada personaje posea distinta “skills”); de acuerdo al nivel de exigencia que el equipo docente sugiera.

Es un modelo desafiante por su complejidad, pero al mismo tiempo fácil de visualizar y evaluar por sus trayectorias, tiempo de captura y tiempo de escape. El enfrentarse a este modelo representa una prueba de comprensión de los algoritmos de búsqueda avanzada trabajados en el aula, y su implementación en escenarios competitivos y dinámicos. Así como resulta ser un proyecto de portfolio personal que brindará mayor comprensión de los mecanismos de decisión y dificultad en videojuegos y otros fenómenos gamificados.

Primer modelo formal y definición de criterios de éxito

Primer modelo formal:

El tablero se modelará en un entorno discreto positivo como una cuadrícula bidimensional $Z = (X, Y)$, donde cada vértice $x_i \in X$ presentará una celda, las mismas podrán ser navegables o representarán un obstáculo, y cada arista $y_i \in Y$ conectará celdas adyacentes.

El estado del juego será representado en cada turno $f(x)$, y será definido como $f(turno) = [(pos_{gato\_x}, pos_{gato\_y}), (pos_{ratón\_x}, pos_{ratón\_y})]$. Asimismo, las acciones disponibles para ambos actores serán $A = \{arriba, abajo, izquierda, derecha\}$, las cuales modifican la posición del agente activo según las restricciones del tablero. Y se alternarán turnos entre ambos personajes para la ejecución de las acciones.

El objetivo del ratón es maximizar la distancia al gato, o alcanzar una zona de escape, mientras que el gato busca minimizar esa distancia y capturarlo, es decir, lograr que $pos_{gato} = pos_{ratón}$. El juego concluirá cuando se cumpla alguna condición de: captura, escape, o incluso agotamiento del número máximo de turnos (en caso de establecer uno).

Criterios de éxito:

El desempeño de los algoritmos se evaluará mediante métricas cuantitativas:

1. Tasa de captura: porcentaje de partidas en las que el gato logra capturar al ratón.
2. Tiempo promedio de captura: número medio de pasos hasta la intersección de ambos jugadores.
3. Distancia promedio mantenida por el ratón: mide la eficiencia evasiva.
4. Eficiencia computacional: tiempo de ejecución y número de nodos explorados.
5. Comportamiento estratégico: coherencia y realismo de las trayectorias generadas.

Algoritmos preliminares a considerar

Con el fin de resolver esta dinámica se planea implementar múltiples algoritmos de búsqueda. En una primera etapa se aplicarán métodos clásicos como BFS, DFS, Dijkstra y A*, que permitirán analizar trayectorias óptimas y el impacto de las heurísticas en la eficiencia de búsqueda.

En etapas posteriores, se incorporarán enfoques multiagente y adversariales como Minimax, Argmax y Monte Carlo Tree Search (MCTS):

En el caso del Minimax, el gato y el ratón se representarán como agentes racionales que alternan movimientos dentro de un árbol de decisiones. Cada nodo del árbol corresponderá a un estado posible del tablero y las ramas representarán las acciones legales disponibles. Las funciones de evaluación estimarán el valor de cada estado según la distancia relativa entre los agentes, el número de obstáculos intermedios o el tiempo restante antes de la captura o escape. Así como la poda Alpha-Beta podrá emplearse para reducir la cantidad de estados explorados, mejorando la eficiencia sin sacrificar la calidad de las decisiones.

En el caso de Argmax, el modelo permitirá simplificar la toma de decisiones local, priorizando acciones que maximicen instantáneamente una heurística de utilidad, como la disminución o aumento de la distancia Manhattan entre los agentes.

Por su parte, Monte Carlo Tree Search (MCTS) proporciona un marco estocástico para la exploración del espacio de estados a través de simulaciones aleatorias (rollouts). En este contexto, el gato buscará maximizar la probabilidad de captura, mientras que el ratón intentará maximizar la supervivencia o el tiempo de evasión.

Se considera que, desde el momento cero, será necesario separar el fenómeno en dos problemas de búsqueda, uno para cada agente. Así como se considera de gran interés la integración de herramientas visuales para mejor comprensión de los estados del juego.