Análisis y Resolución de Algoritmos para University Course Timetabling Problem (UCTTP)

Raimel Daniel Romaguera Puig
Grupo C412

Abstract

El University Course Timetabling Problem (UCTTP) es un problema NP-Completo de optimización combinatoria con una gran relevancia práctica en la gestión académica . Este trabajo aborda el problema desde una doble perspectiva: teórica y práctica. En primer lugar, se presenta una demostración formal de su NP-Completitud mediante una reducción desde el problema de coloración de grafos, y se analiza la inexistencia de algoritmos de aproximación con garantías estrictas para su variante general. Estos resultados teóricos justifican la necesidad de enfoques aproximados. En consecuencia, se estudian y aplican métodos heurísticos y metaheurísticos para su resolución práctica. Se evalúan algoritmos de coloración de grafos (Greedy, DSatur, RLF) en grafos aleatorios, donde DSatur demostró un rendimiento superior. Asimismo, se implementan y comparan las metaheurísticas GRASP y Algoritmo Genético en instancias representativas del UCTTP. Los resultados experimentales indican que ambas son efectivas, con GRASP mostrando una robustez y velocidad notables, mientras que el Algoritmo Genético puede alcanzar soluciones de menor costo en algunos casos, aunque con un mayor tiempo de ejecución. La investigación confirma la inherente complejidad computacional del UCTTP y valida el uso de métodos heurísticos y metaheurísticos como herramientas viables para generar horarios factibles y de calidad en entornos educativos reales.

1. Introducción

El University Course Timetabling Problem (UCTTP) es un problema de optimización combinatoria fundamental en la gestión académica, que consiste en asignar cursos [2], [3], [4], profesores y estudiantes a franjas horarias y aulas disponibles, respetando un conjunto de restricciones tanto duras (obligatorias) como blandas (deseables). Su relevancia práctica es indiscutible, ya que un horario bien diseñado optimiza el uso de recursos, mejora la experiencia educativa y reduce conflictos operativos en instituciones de educación superior.

Sin embargo, el UCTTP está clasificado como un problema NP-Completo [3], lo que implica que no existe un algoritmo exacto capaz de resolverlo en tiempo polinomial para instancias de tamaño realista. Esta complejidad computacional ha motivado la búsqueda de enfoques alternativos, como métodos heurísticos [1], [4], [8] y metaheurísticos [2], [3], [6], que ofrecen soluciones factibles y de alta calidad en tiempos razonables, aunque sin garantías de optimalidad.

En este proyecto, se aborda el UCTTP desde una perspectiva teórico-práctica integral. En primer lugar, se presenta una demostración formal de su NP-Completitud, basada en una reducción desde el problema de coloración de grafos, estableciendo así su dificultad computacional inherente. Además, se explora la inexistencia de algoritmos de aproximación con garantías estrictas para ciertas variantes del problema [5], lo que refuerza la necesidad de métodos aproximados y heurísticos.

Como respuesta a esta complejidad, se estudian y aplican enfoques heurísticos, en particular aquellos basados en la teoría de grafos y coloración, así como metaheurísticas como GRASP y algoritmos genéticos, las cuales permiten una exploración eficiente del espacio de soluciones. Finalmente, se incluye un análisis experimental que evalúa el rendimiento de estos algoritmos en instancias representativas del problema, utilizando métricas como factibilidad, calidad de la solución y tiempo de ejecución.

Esta investigación no solo contribuye a la comprensión teórica del UCTTP, sino que también ofrece herramientas prácticas para su resolución, equilibrando rigor académico y aplicabilidad en contextos reales.

2. Definición del Problema

El University Course Timetabling Problem (UCTTP) es un problema de optimización combinatoria con el objetivo principal es asignar un conjunto de eventos académicos (cursos, clases, laboratorios) a franjas horarias y aulas predefinidas, cumpliendo un conjunto de restricciones que garantizan la factibilidad y calidad del horario generado.

2.1 Componentes Principales

- **Eventos (E):** Actividades a programar (clases, exámenes, tutorías).
- **Franjas Horarias (T):** Intervalos de tiempo discretos disponibles.
- **Recursos (R):** Aulas, profesores, equipamiento necesario.
- **Personas (P):** Estudiantes y profesores involucrados.
- **Restricciones (C):** Reglas clasificadas en duras (obligatorias) y blandas (deseables).

2.2 Tipos de Restricciones

- **Restricciones Duras:** Deben satisfacerse completamente. Ejemplo de algunas de estas restricciones son:
  - Un profesor no puede impartir dos clases simultáneamente.
  - Un aula no puede albergar más de un evento a la vez.
  - La capacidad del aula debe ser suficiente para los estudiantes.
- **Restricciones Blandas:** Su cumplimiento mejora la calidad. Entre las que se encuentran:
  - Preferencias de horario de profesores y estudiantes.
  - Distribución uniforme de clases en la semana.
  - Evitar horarios consecutivos para un mismo estudiante.

2.3 Variantes del UCTTP

- **Curriculum-Based (CB-UCTTP):** [3] Basado en planes de estudio fijos (currículos) que agrupan cursos con estudiantes comunes. El objetivo es evitar solapamientos entre cursos de un mismo currículo.
- **Post-Enrolment (PE-UCTTP):** [3] Se construye después de la matrícula, utilizando listas reales de inscripción para minimizar conflictos individuales.

2.4 Objetivo General

Encontrar una asignación completa que:

1. Cumpla todas las restricciones duras (solución factible).
2. Minimice la violación de restricciones blandas (solución de calidad).

3. Modelo Matemático del Problema

Se presenta un modelo matemático para la variante **CB-UCTTP**, formulado como un problema de programación entera binaria.

3.1 Conjuntos

- C: Conjunto de cursos.
- Q: Conjunto de currículos. Cada currículo $q \in Q$ es un subconjunto de cursos $C_q \subseteq C$.
- P: Conjunto de períodos de tiempo disponibles.
- R: Conjunto de aulas.
- T: Conjunto de profesores.
- F: Conjunto de preferencias (restricciones blandas).

3.2 Parámetros

- $a(c)$: Número de estudiantes inscritos en el curso $c \in C$.
- $\text{cap}(r)$: Capacidad (estudiantes) del aula $r \in R$.
- $C(t)$: Conjunto de cursos para los que el profesor $t \in T$ está calificado.
- $\text{disp}_t(p) \in \{0, 1\}$: Disponibilidad del profesor t en el período $p \in P$.
- $\text{disp}_r(p) \in \{0, 1\}$: Disponibilidad del aula r en el período $p \in P$.

3.3 Variables de Decisión

Para cada combinación factible de curso, período, aula y profesor, se define una variable binaria de asignación:

- $x_{c,p,r,t} \in \{0, 1\}$, definida para todo $c \in C$, $p \in P$, $r \in R$, $t \in T$ que satisfacen $\text{cap}(r) \geq a(c)$ y $c \in C(t)$.

La variable toma el valor 1 si el curso c se asigna al período p, en el aula r y es impartido por el profesor t; y 0 en caso contrario. Las condiciones $\text{cap}(r) \geq a(c)$ y $c \in C(t)$ garantizan que solo se consideren combinaciones factibles en términos de capacidad del aula y calificación del profesor.

3.4 Restricciones Duras

Las siguientes restricciones deben cumplirse en toda solución factible del problema:

$$\sum_{p \in P} \sum_{r \in R} \sum_{t \in T} x_{c,p,r,t} = 1, \qquad \forall c \in C, \tag{1}$$

$$\sum_{c \in C} \sum_{r \in R} x_{c,p,r,t} \leq 1, \qquad \forall t \in T, \forall p \in P, \tag{2}$$

$$\sum_{c \in C_q} \sum_{r \in R} \sum_{t \in T} x_{c,p,r,t} \leq 1, \qquad \forall q \in Q, \forall p \in P, \tag{3}$$

$$\sum_{c \in C} \sum_{t \in T} x_{c,p,r,t} \leq 1, \qquad \forall r \in R, \forall p \in P, \tag{4}$$

$$x_{c,p,r,t} \leq \text{disp}_t(p), \qquad \forall c \in C, \forall p \in P, \forall r \in R, \forall t \in T, \tag{5}$$

$$x_{c,p,r,t} \leq \text{disp}_r(p), \qquad \forall c \in C, \forall p \in P, \forall r \in R, \forall t \in T. \tag{6}$$

Explicación de las restricciones:

- (1): Cada curso debe asignarse exactamente a un período, un aula y un profesor.
- (2): Un profesor no puede impartir más de un curso simultáneamente en un mismo período.

- (3): Cursos pertenecientes a un mismo currículo no pueden solaparse en un mismo período.
- (4): Un aula no puede albergar más de un curso por período.
- (5): Un profesor solo puede ser asignado a un curso en períodos en los que esté disponible.
- (6): Un aula solo puede utilizarse para un curso en períodos en los que esté disponible.

Las restricciones de capacidad del aula y calificación del profesor se incorporan implícitamente en el dominio de definición de las variables, por lo que no requieren ecuaciones explícitas adicionales.

3.5 Función Objetivo

Maximizar el cumplimiento de preferencias:

$$\text{máx} \sum_{f \in F} x_{c_f, p_f, r_f, t_f} \tag{7}$$

donde cada preferencia $f \in F$ se representa como una tupla (c_f, p_f, r_f, t_f) que indica una asignación deseada.

4. Complejidad Computacional: Demostración de NP-Completitud

El University Course Timetabling Problem (UCTTP) pertenece a la clase de problemas NP-Completos [2], [3] [4]. Esta clasificación implica que no se conoce ningún algoritmo que lo resuelva de manera exacta en tiempo polinomial para todas las instancias. En esta sección se demuestra formalmente la NP-Completitud del UCTTP mediante una reducción desde el problema de coloración de grafos, el cual a su vez se demuestra NP-Completo a partir del problema 3-SAT.

4.1 Preliminares de Complejidad Computacional

- **Clase P:** Problemas de decisión resolubles en tiempo polinomial por una máquina de Turing determinista.
- **Clase NP:** Problemas de decisión cuyas soluciones pueden verificarse en tiempo polinomial.
- **Problema NP-Completo:** Un problema está en esta clase si pertenece a NP y todo problema en NP puede reducirse a él en tiempo polinomial (es NP-Difícil).

4.2 El Problema de Coloración de Grafos es NP-Completo

El problema K-Coloring consiste en determinar si un grafo no dirigido $G = (V, E)$ puede colorearse con a lo sumo K colores, de modo que vértices adyacentes tengan colores distintos.

4.2.1 K-Coloring está en NP

Dado un grafo G y un entero K, un certificado es una asignación de colores. Verificar que cada arista conecte vértices de colores distintos toma tiempo $O(|E|)$, por lo que el problema está en NP.

4.2.2 Reducción de 3-SAT a 3-Coloring (NP-Dificultad)

Para demostrar que 3-Coloring es NP-Difícil, se reduce desde 3-SAT, un problema NP-Completo conocido. Dada una fórmula booleana ϕ en CNF con n variables y m cláusulas, se construye un grafo G_ϕ que es 3-coloreable si y solo si ϕ es satisfacible.

Construcción de G_ϕ:

1. **Vértices de verdad:** Se crean tres vértices T (verdadero), F (falso) y B (base), conectados en triángulo.
2. **Variables:** Para cada variable x_i, se crean vértices x_i y $\neg x_i$. Se añaden aristas $(x_i, \neg x_i)$, (x_i, B) y $(\neg x_i, B)$.
3. **Cláusulas:** Para cada cláusula $C_j = (l_{j1} \vee l_{j2} \vee l_{j3})$, se añade un gadget de 6 vértices $\{u_j, v_j, w_j, a_j, b_j, c_j\}$ con aristas que conectan triángulos superiores/inferiores, literales, T y entre sí.

Prueba de equivalencia:

- Si ϕ es satisfacible, existe una asignación de verdad que permite colorear G_ϕ con 3 colores.
- Si G_ϕ es 3-coloreable, la coloración induce una asignación de verdad que satisface ϕ.

Dado que 3-SAT es NP-Difícil, 3-Coloring también lo es. Esta demostración se extiende a K-Coloring para $K \geq 3$.

4.3 El Problema de Course Timetabling (CT) es NP-Completo

Se define la versión de decisión del CT, equivalente al UCTTP: dada una instancia con cursos, períodos, aulas y conflictos, determinar si existe una asignación factible.

4.3.1 CT está en NP

Dada una asignación propuesta, verificar todas las restricciones duras (no superposición, capacidad, disponibilidad) se realiza en tiempo polinomial. Por tanto, CT $\in$ NP.

4.3.2 Reducción de Graph Coloring a CT (NP-Dificultad)

Se construye una reducción polinomial desde el problema de Coloración (GC, Graph Coloring) hacia CT.

Dada una instancia de GC: grafo $G = (V, E)$ y entero p, se construye una instancia de CT:

- **Cursos:** Un curso K_i por cada vértice $v_i \in V$, con una clase.
- **Períodos:** p períodos (equivalentes a colores).
- **Aulas:** Capacidad ilimitada por período ($l_k = |V|$).
- **Conflictos:** Para cada arista $(v_i, v_j) \in E$, los cursos K_i y K_j están en conflicto.
- **Restricciones adicionales:** Sin preasignaciones ni restricciones de disponibilidad.

Prueba de equivalencia:

- Si G es p-coloreable, la coloración define una asignación factible de cursos a períodos.
- Si existe una solución factible para CT, la asignación de períodos a cursos define una p-coloración válida para G.

Por lo tanto, CT es NP-Difícil.

4.4 Conclusión de NP-Completitud

Dado que:

1. El problema de decisión Course Timetabling (CT) está en NP, y
2. CT es NP-Difícil (por reducción desde GC, un problema NP-Completo),

se concluye que el Course Timetabling Problem (y por extensión el UCTTP) es NP-Completo.

Conocer que este problema es NP-Completo implica que, bajo el supuesto $P \neq NP$, no existe un algoritmo de tiempo polinomial que resuelva óptimamente todas las instancias del UCTTP. Además, la complejidad inherente valida el uso de métodos aproximados (heurísticas, metaheurísticas, híbridos) para obtener soluciones de calidad práctica en tiempos razonables.

5. Inexistencia de Algoritmos de Aproximación

La complejidad NP-Completa del University Course Timetabling Problem (UCTTP) no solo implica la inexistencia de algoritmos exactos de tiempo polinomial, sino que también establece límites fuertes sobre la posibilidad de diseñar algoritmos de aproximación con garantías teóricas estrictas. Utilizando el Teorema PCP (Probabilistically Checkable Proofs) y otros resultados de complejidad que se mencionan en [5], se demuestra que aproximar el número cromático de un grafo (y por extensión el UCTTP) es aún más difícil.

Teorema de Inaproximabilidad del Número Cromático [5] : Asumiendo que $NP \not\subseteq ZPP$ (más fuerte que $P \neq NP$), no existe un algoritmo de tiempo polinomial que aproxime $\chi(G)$ dentro de un factor $n^{1-\varepsilon}$ para cualquier constante $\varepsilon > 0$, donde n es el número de vértices del grafo.

Interpretación: No se puede garantizar un algoritmo que devuelva una coloración usando a lo sumo $n^{1-\varepsilon} \cdot \chi(G)$ colores, a menos que $P = NP$. Dado que el UCTTP se reduce directamente al problema de coloración, el mismo límite se aplica: **no existe un algoritmo de aproximación en tiempo polinomial que garantice un factor polinomialmente cercano al óptimo**.

El problema de coloración de grafos cuenta con algoritmos de aproximación clásicos, pero sus factores de aproximación son elevados. Para los casos más generales, los mejores algoritmos conocidos alcanzan únicamente ratios de la forma $n/\text{polilog}(n)$ [9], lo que representa una mejora marginal sobre la aproximación trivial de n.

Esta limitación es fundamental: no existe un PTAS (Esquema de Aproximación en Tiempo Polinomial), y mucho menos un FPTAS, para el problema general de coloración, a menos que $P = NP$ [9]. Esta dureza de aproximación se traslada directamente al UCTTP, restringiendo severamente la garantía de rendimiento que pueden ofrecer los algoritmos prácticos.

Finalmente, la inexistencia de algoritmos de aproximación con garantías estrictas para el UCTTP bajo suposiciones estándar de complejidad computacional ($P \neq NP$ y $NP \not\subseteq ZPP$) subraya su naturaleza intrínsecamente difícil. Esto justifica el uso de enfoques prácticos como metaheurísticas, algoritmos híbridos y métodos basados en aprendizaje automático, los cuales, aunque carecen de cotas formales de aproximación, han demostrado ser efectivos en la generación de horarios de alta calidad en escenarios reales.

6. Resolución por Heurísticas basadas en Grafos

El University Course Timetabling Problem (UCTTP) puede modelarse eficazmente como un problema de coloración de grafos [1], [8], dada la equivalencia estructural entre ambos. En este enfoque, los eventos (cursos, exámenes) se representan como vértices, los conflictos entre eventos (debidos a recursos compartidos como profesores, estudiantes o aulas) se modelan como aristas, y los intervalos de tiempo disponibles corresponden a colores. Así, asignar horarios sin conflictos equivale a colorear el grafo con el mínimo número de colores posible.

6.1 Algoritmos Heurísticos de Coloración de Grafos

Debido a la NP-Completitud del problema de coloración, se utilizan algoritmos heurísticos que encuentran soluciones factibles en tiempo razonable, aunque no garantizan optimalidad. A continuación, se describen tres algoritmos clave.

6.1.1 Algoritmo Greedy

El algoritmo Greedy [8] procesa los vértices en un orden determinado (por ejemplo, aleatorio) y asigna

a cada vértice el primer color disponible que no cause conflictos con sus vecinos ya coloreados. Aunque su simplicidad permite obtener soluciones rápidamente, la calidad de la solución depende críticamente del orden de procesamiento. En el peor caso (grafo completo K_n), su complejidad temporal es $O(n^2)$.

6.1.2 Algoritmo DSatur (Degree Saturation)

DSatur [1] mejora al Greedy mediante una ordenación dinámica de los vértices basada en su grado de saturación, definido como el número de colores diferentes asignados a sus vecinos. En cada paso, se selecciona el vértice con mayor grado de saturación (y en caso de empate, mayor grado) para colorearlo. Esta estrategia prioriza los vértices más restringidos, reduciendo el número de colores necesarios. DSatur tiene complejidad $O(n^2)$ y es exacto para grafos bipartitos, ciclos y grafos circulares.

6.1.3 Algoritmo RLF (Recursive Largest First)

RLF [8] opera coloreando conjuntos independientes máximos en cada iteración, asignando un color por iteración. En lugar de colorear vértices de uno en uno, construye recursivamente el conjunto independiente más grande posible, lo elimina del grafo y repite el proceso con el subgrafo restante. Aunque su complejidad es mayor ($O(n^3)$), tiende a producir soluciones de mayor calidad al explorar de manera más estructurada el espacio de soluciones.

6.2 Evaluación Experimental en Grafos Aleatorios

Para comparar el rendimiento de estos algoritmos, se generaron 50 grafos aleatorios para cada tamaño $n \in \{20, 40, 60, 80, 100\}$ utilizando el modelo Erdös-Rényi $G_{n,p}$ con $p = 0.5$ [8]. Los resultados (promedio μ y desviación estándar σ del número de colores) se resumen en la Tabla 1.

Cuadro 1: Resultados promedio y desviación estándar del número de colores para cada heurística.

n	Greedy	DSatur	RLF
20	7.12 ± 0.65	6.04 ± 0.53	6.90 ± 0.64
40	11.24 ± 0.81	9.64 ± 0.71	11.42 ± 0.60
60	14.82 ± 0.93	12.62 ± 0.72	15.50 ± 0.73
80	18.28 ± 0.92	15.70 ± 0.81	19.38 ± 0.75
100	21.12 ± 0.93	18.24 ± 0.76	23.08 ± 0.87

6.2.1 Análisis de Resultados

En los experimentos realizados, el algoritmo DSatur logró superar a los algoritmos Greedy y RLF al producir soluciones de mayor calidad.

7. Resolución usando Metaheurísticas

Dada la NP-Completitud del UCTTP, las metaheurísticas emergen como herramientas poderosas para encontrar soluciones de alta calidad en tiempos razonables, aunque sin garantía de optimalidad [7]. Estas técnicas equilibran exploración (diversificación) y explotación (intensificación) del espacio de búsqueda, manejando eficazmente restricciones complejas.

7.1 Metaheurísticas Aplicadas al UCTTP

Entre las metaheurísticas más utilizadas para el UCTP se encuentran los Algoritmos Genéticos (GA) ([2] [3]) y GRASP (Greedy Randomized Adaptive Search Procedure) ([2] [3] [6]), ambas evaluadas experimentalmente en este trabajo.

7.1.1 Algoritmo Genético (GA)

Los algoritmos genéticos simulan el proceso de evolución natural. Mantienen una población de soluciones candidatas (individuos) que se combinan y mutan a lo largo de generaciones. La selección favorece a los individuos con mayor *fitness* (calidad de solución), permitiendo que las mejores características se propaguen. Los operadores principales son:

- **Selección**: Elección de padres basada en fitness (ej. ruleta, torneo).
- **Cruzamiento (crossover)**: Combinación de dos padres para generar descendencia.
- **Mutación**: Alteración aleatoria de genes para mantener diversidad.

Aunque los GA son robustos y aplicables a diversos problemas, requieren ajuste cuidadoso de parámetros (tamaño de población, tasas de cruce y mutación) y pueden converger prematuramente a óptimos locales.

7.1.2 GRASP (Greedy Randomized Adaptive Search Procedure)

[6] GRASP es una metaheurística iterativa de dos fases:

1. **Construcción Greedy Aleatorizada (Greedy-Randomized)**: Genera una solución factible mediante un algoritmo greedy con aleatoriedad controlada (usando una Lista Restringida de Candidatos, RCL).
2. **Búsqueda Local**: Mejora la solución construida explotando su vecindario hasta alcanzar un óptimo local.

GRASP destaca por su simplicidad, facilidad de implementación y capacidad para producir soluciones de alta calidad con mínima intervención en parámetros. Estrategias avanzadas como GRASP Reactivo ajustan dinámicamente el balance entre greediness y aleatoriedad basándose en el desempeño de iteraciones anteriores.

7.2 Evaluación Experimental en Instancias de UCTTP

Se evaluaron los algoritmos GRASP y el Algoritmo Genético (GA) en un conjunto de cinco instancias representativas del problema UCTTP, seleccionadas por sus características distintivas: tamaño (pequeño, mediano, grande), ratio capacidad de aula/estudiantes, y niveles de conflicto (altos conflictos de curriculum y baja disponibilidad de recursos).

Cada experimento se repitió 20 veces para garantizar robustez estadística. Las métricas evaluadas incluyeron la factibilidad (cumplimiento de todas las restricciones duras), el costo de la solución (valor de la función objetivo que penaliza restricciones blandas) y el tiempo de ejecución en segundos.

7.2.1 Resultados Clave

Los resultados promedio del costo de solución y tiempo de ejecución para cada combinación de instancia y algoritmo se presentan en la Tabla **??**.

Instancia	Algoritmo	Promedio	Tiempo
1	GRASP	0.0	7.7
1	GA	0.0	9.5
2	GRASP	10.4	2.3
2	GA	7.5	3.6
3	GRASP	57.6	13.2
3	GA	52.7	16.9
4	GRASP	32.5	5.2
4	GA	39.6	9.7
5	GRASP	31.8	3.2
5	GA	27.3	5.1

Cuadro 2: Resultados comparativos de GRASP y el Algoritmo Genético en las instancias de prueba. El costo promedio refleja la penalización por violación de restricciones blandas; un valor de 0 indica una solución factible y óptima en términos de dichas restricciones.

- **Instancia 1:** Alto Conflicto
- **Instancia 2:** Disponibilidad Alta
- **Instancia 3:** Grande (12 Cursos)
- **Instancia 4:** Mediana (10 Cursos)
- **Instancia 5:** Pequeña (8 Cursos)

Ambas metaheurísticas demostraron ser efectivas para resolver el UCTTP, ofreciendo un balance adecuado entre calidad de solución y tiempo de ejecución. GRASP mostró un desempeño particularmente robusto y consistente, con tiempos de ejecución generalmente menores y una variabilidad reducida entre repeticiones en la mayoría de los casos. Por otro lado, el Algoritmo Genético requirió un mayor tiempo de cómputo pero logró mejores costos promedio en algunas instancias, como en la instancia pequeña (5) y la de disponibilidad alta (2).

8. Conclusiones

Este trabajo abordó integralmente el University Course Timetabling Problem (UCTTP), confirmando su naturaleza NP-Completa y estableciendo límites para la existencia de algoritmos de aproximación con garantías estrictas. Esta complejidad inherente justificó la exploración de métodos heurísticos y metaheurísticos en la práctica. Los experimentos demostraron que, entre los algoritmos de coloración de grafos, DSatur supera en calidad a las heurísticas Greedy y RLF. Asimismo, la evaluación de las metaheurísticas GRASP y Algoritmo Genético en instancias representativas reveló que ambas son herramientas efectivas para la resolución del UCTTP, con GRASP mostrando un desempeño robusto y eficiente en tiempo.

9. Recomendaciones

1. **Profundizar en Metaheurísticas Híbridas y Avanzadas:** Se recomienda investigar el diseño e implementación de metaheurísticas híbridas que combinen las fortalezas de GRASP (construcción rápida y búsqueda local) con las de los Algoritmos Genéticos (exploración poblacional). También se sugiere explorar otras técnicas como el Recocido Simulado, la Búsqueda Tabú o algoritmos basados en colonias de hormigas, para comparar su desempeño en un conjunto más amplio de instancias benchmark.

2. **Incorporar Restricciones del Mundo Real y Escalabilidad:** Para una transferencia tecnológica efectiva, es necesario extender los modelos y algoritmos para incorporar un conjunto más rico de restricciones. Asimismo, se debe evaluar el rendimiento de los algoritmos propuestos en instancias de gran escala, propias de universidades de tamaño considerable, optimizando para tiempo de ejecución y uso de memoria.

3. **Desarrollar un Marco de Trabajo Software Modular:** Se recomienda el desarrollo de una plataforma software modular y configurable para la generación de horarios. Esta herramienta permitiría a las instituciones académicas definir sus propias restricciones (duras y blandas) y parametrizar los diferentes algoritmos (heurísticos y metaheurísticos) aquí estudiados, facilitando la adopción práctica y la experimentación continua.

4. **Explorar Enfoques de Aprendizaje Automático (ML):** Dada la naturaleza combinatoria y la gran cantidad de datos históricos disponibles en muchas instituciones, se propone explorar la aplicación de técnicas de Machine Learning, como el Aprendizaje por Refuerzo o las Redes Neuronales, para guiar la búsqueda de soluciones, aprender de horarios previos exitosos o para predecir conflictos, potenciando así a las metaheurísticas tradicionales.

Referencias

[1] A Study on Course Timetable Scheduling using Graph Coloring Approach. International Journal of Computational and Applied Mathematics. ISSN 1819-4966 Volume 12, Number 2 (2017), pp. 469-485

[2] A Survey of Approaches for University Course Timetabling Problem: Perspectives, Trends and Opportunities. Computers & Industrial Engineering. Volume 86, August 2015, Pages 43-59

[3] A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. in IEEE Access, vol. 9, pp. 106515-106529, 2021, doi: 10.1109/ACCESS.2021.3100613.

[4] Constraint-based Timetabling. Ph.D. Thesis. Tomás Muller. Charles University in Prague. Faculty of Mathematics and Physics

[5] Zero Knowledge and the Chromatic Number. Journal of Computer and System Sciences. Volume 57, Issue 2, October 1998, Pages 187-199

[6] GRASP: Greedy Randomized Adaptive Search Procedures. Mauricio G. C. Resende, Ricardo M. A. Silva. February 17, 2009. Revised September 9, 2009

[7] Metaheuristics from Design to Implementation. El-Ghazali Talbi. University of Lille -CNRS - INRIA

[8] El Problema de Coloracion de Grafos. Universidad de Santiago de Compostela. Sergio Pena Seijas

[9] Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. THEORY OF COMPUTING, Volume 3 (2007), pp. 103–128