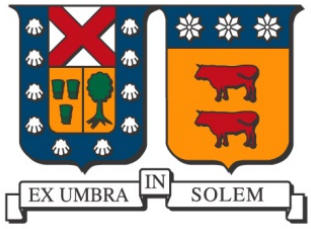
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“FREE TIME POST ENROLMENT COURSE TIMETABLING PROBLEM”

CRISTIAN SORIANO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Elizabeth Montero
Profesor Correferente: María Cristina Riff

Octubre - 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo:

Free Time Post Enrolment Course Timetabling Problem

Nombre del candidato(a):

Cristian Felipe Soriano Fuentes

Carrera / Grado:

Ingeniería Civil Informática

Campus: Casa Central, Valparaíso **Departamento:** Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Elizabeth Montero, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 23 de octubre de 2026 **Firma:** _____

Estudiante o Candidato(a):

Fecha: 23 de octubre, 2026 **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

Dedico este trabajo a mis padres, Viviana de las Mercedes Fuentes Peralta y Cristian Hernan Soriano Manquian, quienes me han dado apoyo incondicional a lo largo de toda la carrera universitaria. Sin ellos no hubiera llegado hasta este momento.

AGRADECIMIENTOS

Agradezco a todos los amigos que he hecho a lo largo de este camino y a quienes conozco desde antes por su presencia en mi vida y por todo el amor y cariño que me han brindado a través de los años.

RESUMEN

Resumen—

En este trabajo se estudia una variante multiobjetivo del *Post Enrolment Course Timetabling Problem* (PECTP), un problema NP-duro que consiste en calendarizar las actividades académicas de una institución de educación superior en un conjunto de salones y bloques horarios, considerando que los estudiantes realizan una preinscripción de los cursos a los que desean asistir. A esta formulación se añade que cada estudiante declara además un conjunto de bloques horarios que preferiría tener libres, por lo que el problema exige optimizar simultáneamente dos objetivos en conflicto: la cobertura de los cursos preinscritos y la satisfacción de las preferencias de tiempo libre. Su alta complejidad combinatoria –derivada de las restricciones institucionales y de los topes de horario de cada estudiante– vuelve impracticable la resolución exacta en casos de tamaño realista. Para abordarlo se propone un algoritmo NSGA-II especializado. Se implementa un generador de instancias que simula casos de la Universidad Técnica Federico Santa María con distintos tamaños, niveles de ocupación del horario y distribuciones de la demanda. Los frentes obtenidos se contrastan con los de un método exacto que resuelve el mismo modelo mediante AMPL y el solver Gurobi, midiendo la calidad con el indicador de hipervolumen y un test de Wilcoxon. Los resultados evidencian un claro compromiso entre calidad y costo computacional: el método exacto alcanza un hipervolumen superior de forma estadísticamente significativa, pero NSGA-II obtiene sus soluciones en un tiempo órdenes de magnitud menor y con una brecha de calidad que se reduce al aumentar el tamaño del problema, llegando a un 92–94 % del hipervolumen del método exacto en las instancias grandes, donde este último degrada la diversidad de su frente. Esto posiciona al algoritmo propuesto como una alternativa práctica para una calendarización centrada en el estudiante a escalas realistas.

Palabras Clave— Post Enrolment Course Timetabling Problem, Problemas Multiobjetivo, Satisfacción Estudiantil, NSGA-II

ABSTRACT

Abstract— This work studies a multiobjective variant of the *Post Enrolment Course Timetabling Problem* (PECTP), an NP-hard problem that consists of scheduling the academic activities of a higher education institution across a set of rooms and time blocks, given that students pre-enrol in the courses they wish to attend. This formulation is extended by letting each student declare a set of time blocks they would prefer to keep free, so

that the problem requires the simultaneous optimization of two conflicting objectives: the coverage of the pre-enrolled courses and the satisfaction of the free-time preferences. Its high combinatorial complexity –arising from the institutional constraints and the schedule clashes of each student– makes exact resolution impractical for realistically sized cases. To address it, a specialized NSGA-II algorithm is proposed. An instance generator is implemented to simulate cases of the *Universidad Técnica Federico Santa María* with varying sizes, timetable occupancy levels and demand distributions. The resulting fronts are contrasted with those of an exact method that solves the same model through AMPL and the Gurobi solver, measuring quality with the hypervolume indicator and a Wilcoxon test. The results reveal a clear trade-off between quality and computational cost: the exact method attains a statistically significantly higher hypervolume, but NSGA-II obtains its solutions in orders of magnitude less time and with a quality gap that narrows as the problem size grows, reaching 92–94 % of the exact method's hypervolume on the large instances, where the latter degrades the diversity of its front. This positions the proposed algorithm as a practical alternative for student-centered timetabling at realistic scales.

Keywords— Post Enrolment Course Timetabling, Multiobjective Problems, Student Satisfaction, NSGA-II

GLOSARIO

Aquí se deben colocar las siglas mencionadas en el trabajo y su explicación, por orden alfabético. Por ejemplo:

AG: Algoritmo Genético.
AMPL: *A Mathematical Programming Language*; lenguaje de modelado matemático empleado junto al solver Gurobi.
HV: Hipervolumen; indicador de calidad para problemas de optimización multiobjetivo.
ITC: *International Timetabling Competition*.
LS: *Local Search* (búsqueda local).
MC: *Maximum Clique* (clique máxima).
MOEA: *Multi-objective Evolutionary Algorithm* (algoritmo evolutivo multiobjetivo).
MOEA/D: *Multi-objective Evolutionary Algorithm based on Decomposition*.
NP: *Nondeterministic Polynomial time*; clase de complejidad computacional (un problema NP-duro es al menos tan difícil como cualquier problema de NP).
NSGA-II: *Non-dominated Sorting Genetic Algorithm II*.
NSGA-III: *Non-dominated Sorting Genetic Algorithm III*.
PATAT: *International Conference on the Practice and Theory of Automated Timetabling*.
PECTP: *Post Enrolment Course Timetabling Problem*.
PLE: Programación Lineal Entera.
RWS: *Roulette Wheel Selection* (selección por ruleta).
SA: *Simulated Annealing* (recocido simulado).
SAR: *Simulated Annealing* con Recalentamiento.
TS: *Tabu Search* (búsqueda tabú).
TSSP: *Tabu Search* con Muestreo y Perturbación.
UTFSM: Universidad Técnica Federico Santa María.

ÍNDICE DE CONTENIDOS

RESUMEN iv

ABSTRACT iv

GLOSARIO vi

ÍNDICE DE FIGURAS ix

ÍNDICE DE TABLAS x

INTRODUCCIÓN 1

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA 3

1.1 Problemas multi-objetivo . . . 4

1.1.1 Dominancia de Pareto . . . 5

1.1.2 Puntos de referencia . . . 6

1.1.3 Medidas de calidad multi-objetivo . . . 7

1.1.4 Métodos de resolución . . . 8

1.2 Modelo matemático . . . 11

1.2.1 Conjuntos y Parámetros . . . 11

1.2.2 Variables . . . 12

1.2.3 Funciones objetivo . . . 13

1.2.4 Restricciones . . . 13

CAPÍTULO 2: MARCO CONCEPTUAL 16

2.1 Estado del Arte . . . 16

2.1.1 Enfoques Completos . . . 17

2.1.2 Búsqueda Local . . . 18

2.1.3 Enfoques Evolutivos . . . 20

2.1.4 Acercamientos multi-objetivo . . . 21

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN 25

3.1 Non-dominated Sorting Genetic Algorithm II . . . 25

3.2 Representación del cromosoma . . . 28

3.3 Inicialización . . . 28

3.4 Evaluación . . . 29

3.4.1 Heurística de inscripción de estudiantes . . . 30

3.4.2 Procedimiento de evaluación . . . 31

3.4.3 Factibilidad . . . 31

3.5 Operadores de transformación . . . 33

3.5.1 Cruzamiento . . . 33

3.5.2 Mutación . . . 33

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN **36**
4.1 Experimentos 36
4.1.1 Casos de prueba 38
4.1.2 Obtención de frentes usando AMPL 39
4.1.3 Parámetros del algoritmo 41
4.2 Resultados 42
4.3 Discusión 44

CAPÍTULO 5: CONCLUSIONES **46**
5.1 Conclusiones sobre los resultados 46
5.2 Trabajo futuro 47

REFERENCIAS BIBLIOGRÁFICAS **49**

ÍNDICE DE FIGURAS

1 Relación entre el espacio de decisión y el espacio objetivo para un problema con dos variables de decisión (x_1, x_2) y dos objetivos (f_1, f_2). Fuente: Elaboración propia. 5

2 Frente de Pareto para un problema de minimización de dos objetivos (f_1 y f_2) arbitrarios. Fuente: Elaboración Propia. 6

3 Ejemplo de hipervolumen en el espacio objetivo para dos objetivos f_1 y f_2. Fuente: Elaboración propia. 9

4 Selección de individuos para la generación $g + 1$. Fuente: Elaboración Propia 26

5 Representación del cromosoma de un individuo arbitrario. Fuente: Elaboración propia 28

6 Ejemplo de subgrafo de preferencias de cursos para un estudiante que solicita los cursos 1 a 5. El algoritmo de clique máxima inscribirá al estudiante en los cursos $\{1, 2, 3, 4\}$ porque forman un grafo completo, es decir, no tienen actividades que colisionen. 30

7 Ejemplo de cruzamiento. Fuente: Elaboración propia 34

8 Posible resultado de distribución de demanda y su transformación con la función (24) con cuatro cifras significativas. Fuente: Elaboración propia. 37

9 Distribución de la demanda estudiantil en los cursos de cada caso de prueba. Fuente: Elaboración propia. 40

10 Distribución de la cantidad de peticiones de bloque libre por estudiante para cada instancia. Fuente: Elaboración propia. 41

11 Comparación de frentes de Pareto para las diez instancias. El eje y muestra la insatisfacción en inscripción de cursos mientras que el eje x muestra insatisfacción de bloques libres. 43

ÍNDICE DE TABLAS

1 Características de los diez casos de prueba generados con el algoritmo probabilístico. 38
2 Parámetros de ejecución de NSGA-II. 42
3 Hipervolumen por instancia. Para NSGA-II se reporta el del frente agregado (diez semillas) y el promedio desagregado sobre las semillas. El punto de referencia es el nadir de AMPL para cada instancia. 44
4 Tiempos de ejecución por instancia. Para AMPL se reporta el tiempo total y para NSGA-II el promedio sobre las diez semillas. 45
5 Test de Wilcoxon de rangos con signo sobre el hipervolumen del frente agregado, pareado entre las diez instancias. 45

INTRODUCCIÓN

La satisfacción estudiantil se ha consolidado como uno de los indicadores clave del desempeño de las instituciones de educación superior. En un contexto global cada vez más competitivo, la satisfacción de los estudiantes no solo refleja la calidad percibida de los servicios educativos, sino que incide directamente en la reputación institucional, en la lealtad y retención de los estudiantes e incluso en la asignación de recursos y financiamiento [Wong y Chapman, 2023]. Por este motivo, comprender y mejorar los factores que determinan la satisfacción estudiantil se ha vuelto una prioridad para las universidades.

Entre estos factores, uno que suele subestimarse es la forma en que se organizan los horarios de clases. Diversos estudios muestran que decisiones de calendarización aparentemente operativas —como la hora de inicio de las clases o la cantidad de días a la semana en que se dictan— tienen un efecto estadísticamente significativo tanto en los resultados de aprendizaje como en la satisfacción de los estudiantes [Loveland y Bland, 2013]. La calendarización, por lo tanto, no es un mero problema logístico, sino una palanca concreta sobre un indicador estratégico de las instituciones.

A partir de esta evidencia ha surgido el interés por una calendarización centrada en el estudiante, que integre explícitamente sus preferencias en la planificación de los cursos. Este enfoque busca alinear los recursos institucionales con la demanda real de los estudiantes, reduciendo problemas como la sobredemanda, las listas de espera y los choques de horario, y mejorando así la experiencia académica [Luna, 2025]. Poner al estudiante en el centro del proceso de calendarización motiva el desarrollo de algoritmos capaces de considerar sus preferencias como un objetivo de primer orden.

Este trabajo se enmarca en el *Post Enrolment Course Timetabling Problem* (PECTP) [McCollum *et al.*, 2010], la variante de calendarización universitaria en la que la demanda de los estudiantes se conoce de antemano mediante una preinscripción de cursos. Es conocido por pertenecer a la clase de problemas $\mathcal{NP}$-duros [Cooper y Kingston, 1996], es decir, es un problema complicado de resolver manualmente en casos realistas donde la escala de demandas es grande, por lo que enfoques heurísticos para aproximar una solución son bastante comunes. Sobre esta base se propone una variante multi-objetivo que incorpora, además de la cobertura de los cursos preinscritos, la satisfacción de las preferencias de tiempo libre de los estudiantes. En particular, estas preferencias se modelan a partir de un esquema de demandas proveniente de [Müller y Murray, 2010], donde cada estudiante crea una lista de preferencias que pueden ser por algún curso o petición de tiempo libre. Problemas similares han sido abordados por [Davison *et al.*, 2024] al incorporar formatos de enseñanza remota –donde una clase puede ser realizada de manera online– e híbrida, con algunos estudiantes asistiendo de manera remota mientras otros asisten de manera presencial. Dado que la cobertura de cursos y la satisfacción de preferencias horarias son objetivos que, en general, entran en conflicto, el problema adquiere naturaleza multi-objetivo y su resolución produce un conjunto de soluciones de compromiso.

En este contexto, los objetivos principales de este trabajo son:

- Proponer una variante multi-objetivo del PECTP que incorpore la optimización de las preferencias estudiantiles.
- Diseñar un algoritmo NSGA-II especializado para resolver instancias que simulan casos particulares de la *Universidad Técnica Federico Santa María* (UTFSM).
- Comparar los frentes obtenidos por NSGA-II con los obtenidos mediante AMPL (resueltos con Gurobi).

Para alcanzar estos objetivos se implementa un generador de instancias que reproduce distintas condiciones de demanda estudiantil y de ocupación del horario, simulando así casos sintéticos de demanda que asemejan el modelo de calendarización para la UTFSM. El problema se resuelve mediante un método exacto para obtener el frente de Pareto referencial y luego con un algoritmo multi-objetivo y se contrastan ambos enfoques mediante un indicador de calidad multi-objetivo.

El resto del documento se organiza de la siguiente manera. En el capítulo de **definición del problema** se formaliza el PECTP y se extienden los conceptos base de la programación multi-objetivo para presentar el modelo matemático. En el **marco conceptual** se revisa el estado del arte en calendarización educacional, considerando estudios actuales y aplicaciones multi-objetivo con requerimientos similares a los abordados en este trabajo. La **propuesta de solución** describe el algoritmo NSGA-II especializado y sus operadores. La **validación de la solución** detalla los casos de prueba y los resultados experimentales. Finalmente, la **conclusión** resume los hallazgos y plantea líneas de trabajo futuro.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

Los problemas de *timetabling* (o de calendarización) conforman una familia de problemas de optimización combinatoria que consisten, en términos generales, en asignar un conjunto de eventos a un número limitado de recursos distribuidos en el tiempo, de manera que se respeten ciertas condiciones de obligatorio cumplimiento y, simultáneamente, se optimice un conjunto de criterios de calidad.

De forma más precisa, una instancia de un problema de timetabling puede describirse mediante una tupla $(E, P, \mathcal{R}, \mathcal{C})$, donde:

- E: conjunto de eventos que deben ser programados.
- P: conjunto de periodos de tiempo disponibles.
- $\mathcal{R}$: conjunto de recursos que los eventos requieren para llevarse a cabo (por ejemplo, espacios físicos, personas o equipamiento).
- $\mathcal{C}$: conjunto de restricciones que toda asignación debe considerar.

Una solución corresponde a una asignación $\pi : E \to P \times \mathcal{R}$ que ubica cada evento en un periodo y le asocia los recursos necesarios. El conjunto de restricciones $\mathcal{C}$ suele dividirse en *restricciones duras*, cuyo incumplimiento vuelve inviable una solución –como que dos eventos ocupen el mismo recurso en el mismo periodo–, y *restricciones blandas*, que expresan preferencias deseables pero no obligatorias y cuyo grado de satisfacción determina la calidad de la solución. Se dice que una asignación es *factible* si satisface todas las restricciones duras. El objetivo es entonces encontrar, dentro del conjunto de soluciones factibles, aquellas que satisfagan de la mejor manera posible las restricciones blandas.

Dentro de esta familia, los problemas de *timetabling* educacional abordan la calendarización de actividades en instituciones de enseñanza. En particular, el *timetabling* universitario se ocupa de la programación de las actividades académicas de una institución de educación superior. En este contexto, los eventos corresponden a las actividades de los cursos (clases, laboratorios o ayudantías), los periodos a los bloques horarios en que se divide la semana, y los recursos a los salones en los que dichas actividades pueden dictarse. La particularidad de este escenario es que las restricciones no provienen únicamente de la disponibilidad de salones y bloques, sino también de los estudiantes. En este sentido, un mismo estudiante puede inscribirse en varios cursos, por lo que una calendarización adecuada debe evitar que las actividades de los cursos que cursa un estudiante se solapen en el tiempo.

El problema estudiado en esta memoria corresponde a la variante conocida como *Post Enrolment Course Timetabling Problem* (PECTP) [McCollum *et al.*, 2010]. Su rasgo distintivo es que

la demanda de cursos por parte de los estudiantes se conoce de antemano. Cada estudiante realiza una preinscripción de cursos a los que desea asistir y, a partir de esa información, debe construirse una calendarización que le permita cursar la mayor cantidad posible de dichos cursos sin incurrir en topes de horario. Sobre esta base, en este trabajo se incorporan las preferencias de horarios. Cada estudiante declara un conjunto de bloques horarios en los que preferiría no tener clases. De este modo, el problema deja de perseguir un único criterio de calidad y pasa a tener naturaleza multiobjetivo, pues la satisfacción de las preferencias horarias y la cobertura de los cursos preinscritos son criterios que, en general, entran en conflicto y no pueden maximizarse de manera independiente.

Esta correspondencia permite situar la formulación propuesta dentro del marco general descrito. Los eventos del problema son el conjunto de actividades académicas A; los periodos, el conjunto de bloques horarios T; y los recursos, el conjunto de salones R, cada uno de los cuales es adecuado solo para un subconjunto de actividades. A esta estructura se suma una capa centrada en el estudiante –el conjunto S de estudiantes, el conjunto C de cursos y, para cada estudiante, sus cursos preinscritos C_s y sus preferencias de bloques libres T_s. Estos últimos dan origen a los dos criterios de calidad que se buscan optimizar conjuntamente.

Dado que no existe una única solución que optimice ambos criterios a la vez, el resultado del problema no es una solución individual sino un conjunto de soluciones de compromiso. En la siguiente sección se describen los conceptos básicos que conforman los problemas multiobjetivo y se formaliza la descripción del problema mediante un modelo de programación lineal entera mixta, detallando sus conjuntos, parámetros, variables de decisión, las dos funciones objetivo que componen el vector $\vec{z}$ y el conjunto de restricciones que delimitan las calendarizaciones admisibles.

1.1. Problemas multi-objetivo

Un problema de optimización multi-objetivo trata de optimizar más de una función objetivo simultáneamente. Su formulación es similar a la de un problema mono-objetivo, pero en vez de minimizar una única función, se minimiza un vector de funciones [Hao *et al.*, 2024]:

$$\begin{aligned} \text{mín} \quad & \vec{F}(x) = [f_1(x), f_2(x), \ldots, f_k(x)]^T \\ \text{sujeto a} \quad & g_i(x) \leq 0 \quad \forall i = 1, 2, \ldots, L \\ & h_j(x) = 0 \quad \forall j = 1, 2, \ldots, E \\ & x = [x_1, x_2, \ldots, x_D]^T \in S \end{aligned}$$

Donde k es la cantidad de objetivos o criterios a considerar, g_i son las restricciones de desigualdad (con un total de L restricciones), h_j las restricciones de igualdad (E restricciones) y x el vector de variables de tamaño D con dominio en S. Notar que, para este caso, se busca minimizar el vector completo debido a que generalmente se suelen plantear como problemas de minimización. Para maximizar funciones, se puede definir una equivalencia en minimización optimizando $-\vec{F}(x)$ en la ecuación anterior.

En contraste con problemas de optimización mono-objetivo se introduce la noción del *espacio de decisión*, que es el espacio donde viven las variables de decisión S. Consecuentemente el *espacio de decisión factible* es un subconjunto de éste, $\mathcal{S} \subseteq S$ el cual representa las soluciones que respetan todas las restricciones g_i y h_j. A su vez, el *espacio objetivo* $Z \subseteq \mathbb{R}^k$ es la imagen del espacio de decisión, es decir, $Z = \{\vec{F}(\vec{x}) | \vec{x} \in S\} = \vec{F}(S)$. Nos interesan los elementos del espacio objetivo donde sus soluciones están en $\mathcal{S}$, es decir $\mathcal{Z} \subseteq Z = \vec{F}(\mathcal{S})$. La figura 1 muestra cómo interactúan estos espacios y el mapeo entre el espacio de decisión y el espacio objetivo para dos variables de decisión x_1 y x_2 y un vector objetivo de dos funciones f_1 y f_2. El espacio de decisión completo está delimitado por las líneas punteadas, mientras que el espacio de decisión factible se muestra de un color más oscuro. Como muestra la figura, la región del espacio objetivo $\mathcal{Z}$ se obtiene al evaluar cada $x \in \mathcal{S}$ en $\vec{F}$, formando así el conjunto completo $\mathcal{Z} = \vec{F}(\mathcal{S})$.

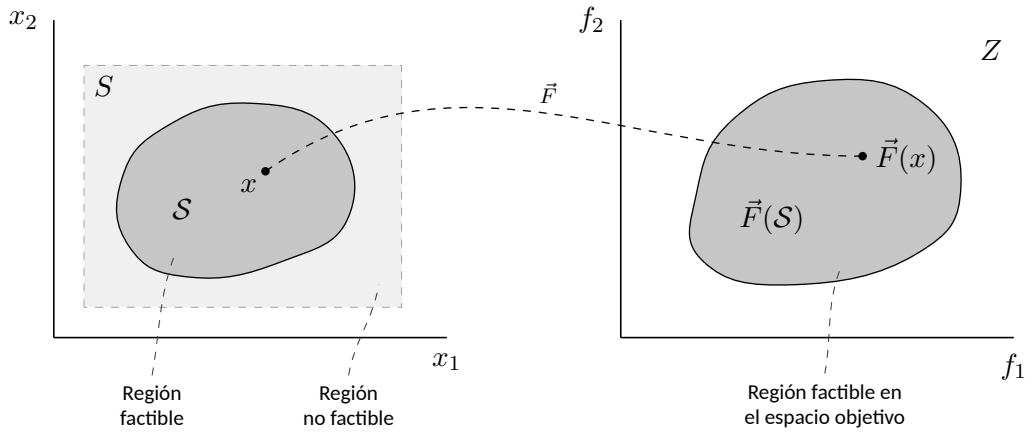


Figura 1: Relación entre el espacio de decisión y el espacio objetivo para un problema con dos variables de decisión (x_1, x_2) y dos objetivos (f_1, f_2). Fuente: Elaboración propia.

1.1.1. Dominancia de Pareto

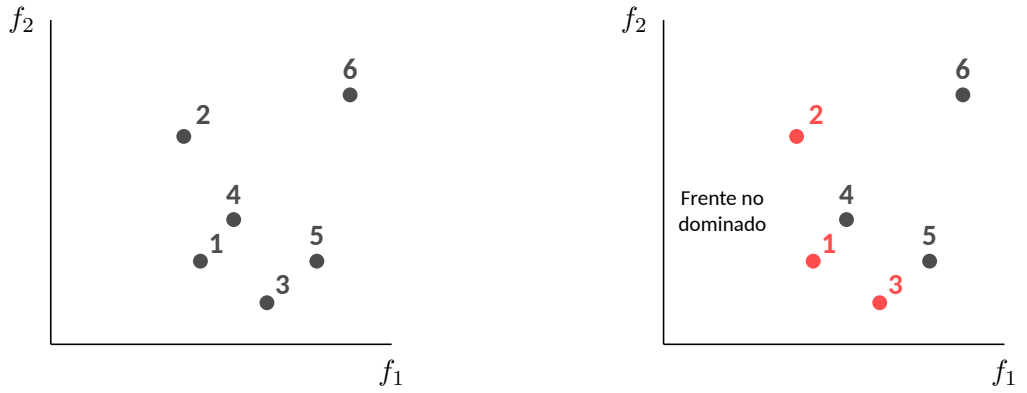
Para comparar soluciones de un problema multi-objetivo, se define la siguiente relación de dominancia [Deb, 2008]:

[Dominancia de Pareto] Una solución $x^{(1)} \in \mathcal{S}$ domina a otra solución $x^{(2)} \in \mathcal{S}$ si las siguientes condiciones son verdaderas:

- La solución $x^{(1)}$ no es peor que $x^{(2)}$ para todos los objetivos. Para el caso de minimización, esto quiere decir que se debe cumplir que $f_i(x^{(1)}) \leq f_i(x^{(2)}), \forall i = \{1, 2, \ldots, k\}$.
- La solución $x^{(1)}$ es estrictamente mejor que la solución $x^{(2)}$ en al menos uno de los objetivos. Para el caso de minimización, la desigualdad anterior se reescribe como $\exists i \in \{1, 2, \ldots, k\} : f_i(x^{(1)}) < f_i(x^{(2)})$.

El conjunto de soluciones factibles no dominadas define el *conjunto óptimo de Pareto*. Cada solución dentro de este conjunto es una *solución óptima de Pareto*. La imagen del conjunto óptimo de Pareto en el espacio objetivo forma el *frente de Pareto*. Una propiedad interesante de los puntos en el conjunto óptimo de Pareto es que la ganancia en uno de los objetivos de un punto a otro sucede al costo o sacrificio en al menos uno de los otros objetivos.

En la figura 2a se presentan las soluciones factibles en el espacio objetivo $\mathcal{Z}$ de un problema arbitrario, mientras que la figura 2b marca en rojo aquellas soluciones que conforman el frente de Pareto para el caso de minimización de los objetivos.



(a) Conjunto de soluciones factibles $\mathcal{Z}$ (b) Soluciones que forman el frente de Pareto

Figura 2: Frente de Pareto para un problema de minimización de dos objetivos (f_1 y f_2) arbitrarios. Fuente: Elaboración Propia.

1.1.2. Puntos de referencia

Cuando el frente de Pareto real es desconocido (como lo es en el caso de NSGA-II y el método de ponderación), generalmente se quiere hacer una comparación entre el frente obtenido y algún punto de referencia que tenga valores específicos para los objetivos. Estos puntos de referencia se pueden obtener artificialmente y son importantes para definir las métricas de calidad multi-objetivo que se verán a continuación (ver sección 1.1.3). Los puntos más importantes a considerar son el *punto ideal* o *utópico*, el *punto de Nadir* y los *puntos extremos*.

[Punto utópico] El punto utópico (o punto ideal) es el punto en el espacio objetivo que tiene los mejores valores (o los más deseables) para todos los objetivos. En la práctica suele ser un punto inalcanzable.

[Punto de Nadir] Al contrario del punto utópico, el punto de Nadir es el punto en el espacio objetivo que tiene los peores valores para todos los objetivos.

[Puntos extremos] Los puntos extremos son aquellos que tienen el mejor valor posible pero solo para uno de los objetivos. Estos puntos si se suelen obtener después de la ejecución

de alguno de los métodos vistos, de hecho una metodología bastante común es usar orden lexicográfico con k iteraciones con el orden de importancia igual al orden original de los objetivos.

1.1.3. Medidas de calidad multi-objetivo

Existen varias medidas de calidad multi-objetivo en la literatura. En [Okabe *et al.*, 2003], se agrupan estas medidas en tres grupos:

- **Medidas de Cardinalidad**: Este tipo de medidas se basan en la cantidad de soluciones que tenga el frente de Pareto. Ejemplos de estas métricas son el *Overall Non-dominated Vector Generation* [Schott, 1995] que mide el total de soluciones del frente. Por otro lado, *Overall Non-dominated Vector Generation Ratio* mide el ratio entre el total de soluciones encontradas y las soluciones del frente real [Van Veldhuizen y Lamont, 2000].

- **Medidas de Precisión**: Miden la convergencia de un frente obtenido hacia el frente real. Esta categoría se divide en métricas de distancia y métricas de volumen. Cuando el frente real es desconocido, se usa un conjunto de referencia al que se quiere acercar. Algunas métricas de distancia incluyen la *Distancia Generacional* [Van Veldhuizen *et al.*, 1998] que mide la distancia Euclidiana promedio entre las soluciones del frente obtenido y las soluciones del frente de referencia; además de ésta tambien existe la *Distancia Promedio desde el Conjunto de Referencia* [Czyżżak y Jaszkiewicz, 1998] la cual en vez de usar la distancia Euclidiana, la distancia es el mayor rango de separación entre objetivos del frente de referencia y el frente encontrado, ponderados por el rango de cada objetivo en el frente de referencia. Dentro del mismo trabajo se propone la *Peor Distancia desde el Conjunto de Referencia*, la cual calcula la distancia más grande que hay entre una solución del frente de referencia y el frente encontrado. Dentro de las métricas de volumen, [Zitzler y Thiele, 1999] propuso el *Tamaño del Espacio Cubierto*, referenciado en la literatura como el *hiperárea* o *hipervolúmen*, que mide el área (o volumen) que es dominada por el frente obtenido desde un punto de referencia (que generalmente es el punto de nadir 1.1.2). Por otro lado, [Wu y Azarm, 2001] propone una *Diferencia de Hiperárea* que mide la diferencia entre el espacio (área o volumen) dominado por el frente obtenido y el espacio dominado por el frente de referencia. A diferencia del hipervolumen clásico, interesa que la diferencia de hiperárea sea lo más pequeña posible. Una desventaja de estas métricas es que son sensibles al punto de referencia que se considere para calcularlas.

- **Medidas de Diversidad**: Miden la distribución y que tan espaciadas son las soluciones del frente obtenido. Estas medidas tienden a centrarse en la varianza del frente de Pareto y en la distancia entre soluciones. Ejemplos de métricas de distribución son el *Espaciamiento* [Schott, 1995], que mide la desviación de las distancias del frente

encontrado, donde la medida de distancia se calcula a través de la suma de las diferencias absolutas entre soluciones. En [Zitzler, 1999] se propone el índice $\mathcal{M}_2^*$ que mide la distribución del frente obtenido dentro de un vecindario acotado por un radio σ^*, lo que significa que mientras más grande sea menos soluciones van a haber en un radio apropiado por lo que las soluciones están más dispersas. Las medidas de espaciamiento por otro lado buscan medir el rango de las funciones objetivo del frente aproximado. La *Distribución General de Pareto* [Wu y Azarm, 2001] cuantifica que tan extenso se esparce el frente aproximado en el espacio objetivo cuando se consideran todas las funciones objetivo, por lo que involucra el cálculo del ratio entre el el hipervolumen considerando los puntos extremos y hipervolumen usando los puntos utópico y de nadir; mientras mayor sea esta métrica, más espaciado es el frente. Esta medida se puede considerar como un agregado de todos los objetivos, por lo que los autores también proponen calcular una distribución por objetivo individual para describir cuantitativamente el rango de cada uno. En [Zitzler, 1999] también se propone la métrica $\mathcal{M}_3^*$ que mide la máxima extensión de cada objetivo para estimar el rango en el que el conjunto no dominado se esparce.

En este trabajo se usará el hipervolumen para poder comparar las soluciones del método exacto y de NSGA-II. Se usa esta medida debido a su amplio uso en la comparación para problemas de dos objetivos [Riquelme *et al.*, 2015, Laszczyk y Myszkowski, 2019].

El hipervolumen corresponde a la medida de un espacio no dominado usando como referencia el punto de Nadir (definición 1.1.2) y el frente de Pareto aproximado. Permite medir la convergencia al frente de referencia, la uniformidad de las soluciones y qué tan dispersas son. Puede ser calculado mediante:

$$HV(PF) = \Lambda(\bigcup_{s \in PF} \{s' | s \prec s' \prec s^{nadir}\}) \tag{1}$$

En la ecuación 1, Λ es la medida de Lebesgue (para dos objetivos representa el área encerrada, para tres el volumen y así sucesivamente), $\prec$ es la relación de dominancia de Pareto, s es un punto en el frente de Pareto y s^{nadir} corresponde al punto de Nadir. La figura 3 muestra un ejemplo de esta métrica para dos objetivos considerando un caso de minimización y usando el punto de nadir como referencia, el cual se puede ver en la parte superior derecha del gráfico. El área sombreada representa el valor de la medida de Lebesgue Λ dado el frente definido por los puntos morados.

1.1.4. Métodos de resolución

Métodos Completos En [Miettinen, 2008] se categorizan métodos numéricos clásicos para resolver problemas de optimización multi-objetivo. En esta sección se explican los más convencionales y usados a lo largo de la literatura relacionada a estos problemas.

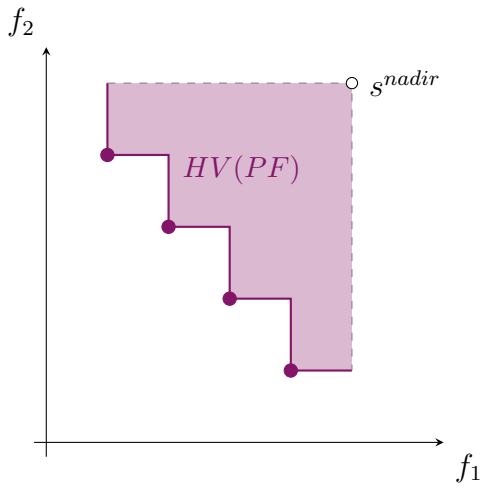


Figura 3: Ejemplo de hipervolumen en el espacio objetivo para dos objetivos f_1 y f_2. Fuente: Elaboración propia.

Métodos de ponderación En estos métodos se busca resolver el siguiente problema:

$$\begin{aligned} \text{mín} \quad & \sum_{i=1}^{k} w_i f_i(x) \\ \text{sujeto a} \quad & x \in S \end{aligned}$$

Donde $w_i \geq 0 \quad \forall i = 1, \ldots, k$ y generalmente $\sum_{i=1}^{k} w_i = 1$. Notar que las restricciones g_i y h_j siguen siendo parte de la definición del problema, pero se omiten por simplificación. Las soluciones obtenidas mediante este método son Pareto óptimas si se tiene que $w_i > 0 \quad \forall i = 1, \ldots, k$. Un punto a tomar en cuenta es que estos métodos pueden encontrar soluciones que son óptimas de Pareto alternando los pesos solamente si el problema es convexo. Por lo que el método tiene esta desventaja de que para problemas no convexos no se pueden encontrar soluciones sin importar cómo se seleccionen los pesos. Incluso para problemas lineales puede ser que no se obtenga un comportamiento esperado, ya que al alternar pesos el método salta de un vértice a otro sin detectar soluciones adyacentes (esto depende también del algoritmo usado por el solver comercial) [Marler y Arora, 2010].

Métodos ε-restrictos En los métodos ε-restrictos, se selecciona un objetivo $\ell \in \{1, \ldots, k\}$ a ser optimizado y los otros pasan a ser restricciones, el problema pasa a tomar la siguiente forma:

$$\begin{aligned} \text{mín} \quad & f_\ell(x) \\ \text{sujeto a} \quad & f_j(x) \leq \varepsilon_j \quad \forall j = 1, \ldots, k : j \neq \ell \\ & x \in S \end{aligned}$$

En esta formulación, ε_j son los límites superiores que pueden tomar los otros objetivos dado que se está optimizando f_ℓ. Una solución $x^* \in S$ es Pareto óptima si resuelve la ecuación para todo $\ell = 1, \ldots, k$ con $\varepsilon_j = f_j(x^*)$ para $j = 1, \ldots, k : j \neq \ell$. Una ventaja que tiene este método en comparación con el método de ponderación es que no necesita convexidad, por lo que funciona para problemas convexos y no convexos. Su desventaja principal es que, si existe una buena solución que está cerca de los límites pero en el lado infactible, nunca será encontrada, por lo que la optimalidad del método depende fuertemente de los límites iniciales especificados, los cuales son difíciles de encontrar en la práctica [Laumanns *et al.*, 2005].

Orden Lexicográfico Este método requiere que se ordenen los objetivos según su importancia absoluta. Una vez ordenadas, cada función objetivo es optimizada de tal manera que cada problema subsecuente no degrade las funciones objetivo obtenidas anteriormente, por lo que en cada iteración se resuelve un problema de la siguiente forma:

$$\begin{aligned} \text{mín} \quad & f_i(x) \\ \text{sujeto a} \quad & f_j(x) \leq f_j(x_j^*) \quad \forall j = 1, \ldots, i-1 : i > 1 \\ & x \in S \end{aligned}$$

En este caso, i representa el índice de las funciones objetivo según el orden definido previamente y x_j^* es la solución óptima encontrada para el objetivo j en su respectiva iteración. Las soluciones obtenidas por este método son altamente sensibles al orden definido, por lo que hacer ajustes en este orden permite explorar diferentes secciones del frente de Pareto [Stewart *et al.*, 2021].

En la literatura también existen enfoques meta-heurísticos (tanto deterministas como estocásticos) para resolver un problema multi-objetivo. En [Sharma y Kumar, 2022] se revisan las técnicas basadas en población más populares y usadas a lo largo de los años para problemas multi-objetivo. Esta clase de técnicas de resolución se pueden dividir en dos categorías principales, por un lado se tienen técnicas evolutivas y por el otro técnicas basadas en comportamientos de colmenas. Ambas se pueden combinar para crear un híbrido (conocidos en la literatura como algoritmos híbridos). Los algoritmos evolutivos usan conceptos basados en la teoría de la evolución natural de Darwin para poder encontrar un conjunto de soluciones. De manera similar, las técnicas basadas en colmenas imitan el comportamiento natural de sistemas biológicos descentralizados y auto organizados, tales como las hormigas, abejas, u otras especies que exhiben estos comportamientos (en [Chakraborty y Kar, 2017] se hace una revisión profunda de algunas variantes de estos tipos de algoritmos).

Algoritmos Evolutivos Multi-objetivo Este trabajo de memoria se centra en el uso de técnicas meta-heurísticas basadas en poblaciones, particularmente algoritmos evolutivos para aproximar el frente de Pareto. Los algoritmos evolutivos multi-objetivo (MOEAs) se dividen en tres clases principales: basados en dominancia, basados en descomposición y basados en indicadores [Schütze y Hernández, 2021]. Los algoritmos basados en dominancia usan la dominancia de Pareto para la selección de individuos. Algunos ejemplos

son MOGA [Fonseca *et al.*, 1993], PAES [Knowles y Corne, 2000], SPEA2 [Zitzler *et al.*, 2001] y NSGA-II [Deb *et al.*, 2002]. Por otro lado, los algoritmos basados en descomposición dividen el problema en sub-problemas más simples usando funciones de escalarización y después los resuelven de manera colaborativa. Algoritmos propuestos en la literatura son MOEA/D [Zhang y Li, 2007], VEGA [Mishra y Singh, 2016] y NSGA-III [Deb y Jain, 2014]. Finalmente, en los algoritmos basados en indicadores, el valor del indicador de calidad determina la selección o ranking de individuos, por lo que la idea es optimizar el indicador a través del proceso de evolución. El algoritmo más representativo de esta clase es el SMS-EMOEA [Beume *et al.*, 2007], el cual usa el hipervolumen como indicador a optimizar.

En este trabajo se usa *Nondominated Sorting Genetic Algorithm II* (NSGA-II) propuesto por [Deb *et al.*, 2002]. En cada generación el algoritmo selecciona soluciones basadas en su grado de no dominancia. Una descripción completa del algoritmo se detalla en la sección 3.1.

1.2. Modelo matemático

En esta sección se presenta un modelo de programación multi-objetivo en donde se consideran, además de los cursos preinscritos por cada estudiante, los horarios en los que preferiría no tener clases. En este contexto, cada curso está asociado a actividades que el estudiante debe tener en su calendarización para inscribirse en el curso. Cada curso exige ciertas funcionalidades de los salones para sus clases, por lo que también se debe considerar que no todos los espacios son adecuados para llevar a cabo las actividades de cada curso.

1.2.1. Conjuntos y Parámetros

Se consideran los siguientes conjuntos:

- S: Conjunto de estudiantes
- C: Conjunto de cursos
- A: Conjunto de actividades
- A_c: Actividades a las que se debe asistir para inscribirse en el curso $c \in C$, con $A_c \subset A$.
- T: Conjunto de bloques horarios disponibles
- R: Conjunto de salones
- R_a: Conjunto de salones adecuados para la actividad $a \in A$
- C_s: Conjunto de cursos en los que el estudiante $s \in S$ quiere inscribirse, con $C_s \subset C$.

- T_s: Conjunto de bloques horarios que el estudiante $s \in S$ prefiere tener libres, con $T_s \subset T$.

Un estudiante $s \in S$ preinscribe $|C_s|$ cursos, por lo que se debe cumplir que $|T_s| \leq |T| - |\bigcup_{c \in C_s} A_c|$. Esto quiere decir que el estudiante no puede elegir más bloques de tiempo libre que el total de actividades a las que asistiría dadas sus preferencias y el conteo de bloques totales. Esto asegura la cobertura de todas las actividades a las que debe asistir, de otro modo no se podrían satisfacer todas sus preferencias por ramos.

Además, se consideran los siguientes parámetros:

- ρ_r: Capacidad del salón $r \in R$.
- σ_c: Límite de inscripciones para el curso $c \in C$.
- $k^s_{\text{mín}}$: Cantidad mínima de ramos que debe tomar el estudiante $s \in S$.
- $k^s_{\text{máx}}$: Cantidad máxima de ramos que puede tomar el estudiante $s \in S$.

Aparte de estos parámetros, en algunas restricciones se usa el término M como un valor lo suficientemente grande para asegurar la activación/desactivación de las restricciones correspondientes.

1.2.2. Variables

Se definen las siguientes variables de decisión:

$$x_{a,r,t} = \begin{cases} 1, & \text{si la actividad } a \in A \text{ se realiza en el salón } r \in R \text{ en el bloque } t \in T \\ 0, & \text{en otro caso} \end{cases}$$

$$y^R_{a,r} = \begin{cases} 1, & \text{si la actividad } a \in A \text{ se realiza en el salón } r \in R \\ 0, & \text{en otro caso} \end{cases}$$

$$y^T_{a,t} = \begin{cases} 1, & \text{si la actividad } a \in A \text{ se realiza en el bloque } t \in T \\ 0, & \text{en otro caso} \end{cases}$$

$$y^A_{s,a} = \begin{cases} 1, & \text{si el estudiante } s \in S \text{ asiste a la clase } a \in A \\ 0, & \text{en otro caso} \end{cases}$$

$$y^C_{s,c} = \begin{cases} 1, & \text{si el estudiante } s \in S \text{ inscribe el curso } c \in C \\ 0, & \text{en otro caso} \end{cases}$$

$$\tau_{s,a,t} = \begin{cases} 1, & \text{si el estudiante } s \in S \text{ asiste a la actividad } a \in A \text{ en el bloque } t \in T \\ 0, & \text{en otro caso} \end{cases}$$

1.2.3. Funciones objetivo

Primero se plantea la función objetivo que minimiza la planificación de actividades en los horarios que los estudiantes prefieren libres. La expresión se presenta en la ecuación 2.

$$z_1 = \frac{1}{|S|} \sum_{s \in S} \frac{\sum_{a \in A} \sum_{t \in T_s} \tau_{s,a,t}}{|T_s|} \tag{2}$$

Lo que mide realmente es la tasa promedio de clases que los estudiantes tienen en bloques en los que pidieron tiempo libre. La suma interior mide el promedio para cada estudiante individual y la exterior promedia a todos los estudiantes, considerando el factor $1/|S|$. Está normalizada entre 0 y 1, por lo que se obtiene una estimación teórica del porcentaje de insatisfacción media en preferencias de tiempo de la solución actual.

La segunda función objetivo hace algo similar con la proporción de cursos que la solución no pudo asignar a cada estudiante, definida en la ecuación 3.

$$z_2 = \frac{1}{|S|} \sum_{s \in S} 1 - \frac{\sum_{c \in C_s} y^C_{s,c}}{|C_s|} \tag{3}$$

Al igual que en el caso anterior, la suma interna mide la proporción de cursos que al estudiante sí se le pudieron asignar y que habían sido preinscritos (la cual estará entre 0 y 1), por lo que al calcular su complemento –marcado por la diferencia entre 1 y la proporción calculada– es posible saber la insatisfacción de ese estudiante con respecto al calendario generado. Finalmente se considera la insatisfacción promedio de todos los estudiantes.

1.2.4. Restricciones

A continuación se presentan y explican las restricciones del problema. Una actividad puede llevarse a cabo en un único salón y bloque.

$$\sum_{r \in R} \sum_{t \in T} x_{a,r,t} = 1 \quad \forall a \in A \tag{4}$$

Una actividad no puede ser asignada en un salón que no sea adecuado para ella.

$$\sum_{t \in T} x_{a,r,t} = 0 \quad \forall a \in A, r \in R \setminus R_a \tag{5}$$

En una combinación salón-bloque solo puede llevarse a cabo a lo más una actividad en cada instante de tiempo.

$$\sum_{a \in A} x_{a,r,t} \leq 1 \quad \forall t \in T, r \in R \tag{6}$$

Una actividad $a \in A$ se realiza en el salón $r \in R$ si para algún bloque de tiempo la actividad se realiza en ese salón:

$$y^R_{a,r} = \sum_{t \in T} x_{a,r,t} \quad \forall a \in A, r \in R \tag{7}$$

Una actividad $a \in A$ sucede en el bloque de tiempo $t \in T$ si la actividad está asignada a algún salón en ese bloque de tiempo:

$$y^T_{a,t} = \sum_{r \in R} x_{a,r,t} \quad \forall a \in A, t \in T \tag{8}$$

Una actividad solo puede suceder en un bloque de tiempo y un salón a la vez:

$$\sum_{t \in T} y^T_{a,t} = 1 \quad \forall a \in A \tag{9}$$

$$\sum_{r \in R} y^R_{a,r} = 1 \quad \forall a \in A \tag{10}$$

A un estudiante no se le inscribe en cursos en los que no se preinscribió y tampoco asiste a las actividades de dichos cursos.

$$y^C_{s,c} = 0 \quad \forall s \in S, c \in C \setminus C_s \tag{11}$$

$$\sum_{a \in A_c} y^A_{s,a} = 0 \quad \forall s \in S, c \in C \setminus C_s \tag{12}$$

Un estudiante debe inscribir al menos $k^s_{\text{mín}}$ y a lo más $k^s_{\text{máx}}$ cursos:

$$\sum_{c \in C_s} y^C_{s,c} \geq k^s_{\text{mín}} \quad \forall s \in S \tag{13}$$

$$\sum_{c \in C_s} y^C_{s,c} \leq k^s_{\text{máx}} \quad \forall s \in S \tag{14}$$

Un estudiante asiste a un curso si asiste a todas las actividades asociadas al curso. La ecuación 15 asegura la activación de la variable $y^C_{s,c}$ si el estudiante asiste a todas las actividades del curso. La ecuación 16 fuerza al estudiante a asistir a todas las actividades del curso.

$$|A_c| + y^C_{s,c} > \sum_{a \in A_c} y^A_{s,a} \quad \forall s \in S, c \in C_s \tag{15}$$

$$y^C_{s,c} \leq y^A_{s,a} \quad \forall s \in S, c \in C_s, a \in A_c \tag{16}$$

Las ecuaciones 17 a 19 controlan los bloques de tiempo en los que un estudiante tiene que ir a clases.

$$\tau_{s,a,t} \geq y^A_{s,a} + y^T_{a,t} - 1 \quad \forall s \in S, c \in C, a \in A_c, t \in T \tag{17}$$

$$\tau_{s,a,t} \leq y^A_{s,a} \quad \forall s \in S, c \in C_s, a \in A_c, t \in T \tag{18}$$

$$\tau_{s,a,t} \leq y^T_{a,t} \quad \forall s \in S, c \in C_s, a \in A_c, t \in T \tag{19}$$

Un estudiante puede asistir a lo más a una actividad en cada bloque de tiempo.

$$\sum_{a \in A_c} \tau_{s,a,t} \leq 1 \quad \forall s \in S, c \in C_s, t \in T \tag{20}$$

Un estudiante puede ir a una actividad en a lo más un bloque de tiempo.

$$\sum_{t \in T} \tau_{s,a,t} \leq 1 \quad \forall s \in S, c \in C_s, a \in A_c \tag{21}$$

No exceder capacidades de salas ni de cupos de los cursos.

$$\sum_{s \in S} y^A_{s,a} \leq \rho_r + M(1 - y^R_{a,r}) \quad \forall r \in R, a \in A \tag{22}$$

$$\sum_{s \in S} y^C_{s,c} \leq \sigma_c \quad \forall c \in C \tag{23}$$

El modelo anterior delimita de forma precisa el espacio de calendarizaciones admisibles y los dos criterios de calidad que se buscan optimizar, pero no prescribe un mecanismo para resolverlo. Dada su naturaleza multi-objetivo y la complejidad combinatoria del problema, obtener el frente de Pareto de manera exacta resulta impracticable para instancias de tamaño realista, lo que motiva recurrir a métodos de optimización aproximados. En el siguiente capítulo se revisa el estado del arte en calendarización universitaria y en optimización multiobjetivo, con el fin de situar la propuesta de esta memoria respecto de los enfoques y técnicas empleados en la literatura.

CAPÍTULO 2
MARCO CONCEPTUAL

A lo largo de los años se han propuesto múltiples formulaciones del problema adaptando los modelos adecuadamente para las circunstancias de instituciones específicas. Por esto, resulta natural que desde diversas áreas de investigación se trate de encontrar una forma de resolver el problema evitando el uso de métodos de optimización matemática clásicos. En este capítulo se presenta una revisión del estado del arte sobre lo que se ha hecho con respecto al problema y qué formulaciones se han tomado en la actualidad.

2.1. Estado del Arte

En [Schaerf, 1999] se clasifican las principales variantes del problema de elaboración de horarios educacionales (del inglés *educational timetabling problem*) que han sido propuestas en la literatura en tres categorías, las cuales se diferencian entre sí dependiendo del tipo de institución y del tipo de restricciones por aplicar. Estas categorías son las siguientes:

- **Elaboración de horarios escolares**: Su objetivo es obtener una calendarización semanal para todas las clases de un colegio, evitando choques horarios para los profesores.
- **Programación de cursos**: Se quiere obtener una calendarización semanal para todas las actividades de un conjunto de cursos de una universidad, minimizando colisiones entre actividades de cursos que tienen estudiantes en común.
- **Programación de exámenes**: Obtener una calendarización para los exámenes de un conjunto de cursos de una universidad, evitando colisiones de exámenes de cursos que tienen estudiantes en común.

Para el contexto de este estudio, el problema de programación de cursos posterior a las inscripciones se considera una extensión de la categoría de programación de cursos. Debido a la naturaleza multi-objetivo expuesta en la sección anterior, en la siguiente sección se hace una revisión del estado del arte del problema enfocada en los estudios más recientes. Pasando por técnicas usadas en *benchmarks* actuales y casos reales.

Las competencias han sido un eje principal de motivación en la investigación y el desarrollo en problemas de programación de horarios. La *International Conference on the Practice and Theory of Automated Timetabling* (PATAT)[1] ha financiado varias de estas competencias

[1] https://patatconference.org/index.html

a lo largo de los años. De estas competencias, las que más se relacionan con el problema a abordar son las *International Timetabling Competition* (ITC) las cuales hasta su cuarta edición han tratado de problemas relacionados con elaboración de horarios educacionales [McCollum *et al.*, 2010, Post *et al.*, 2016, Müller *et al.*, 2024]. La última de estas, organizada el año 2019 por [Müller *et al.*, 2024] consideró durante su duración múltiples instancias de casos de universidades reales que se generalizaron a treinta casos disponibles en el sitio de la competencia[2]. Las principales instituciones con las que se trabajó para colectar estos datos fueron las universidades de Masaryk, Purdue y la AGH de Cracovia; sin embargo, casos de otras universidades también fueron considerados, tales como la universidad de Maryville en Estados Unidos y la universidad Yachay Tech en Ecuador. Según el formato de la competencia, un curso ofrece un conjunto limitado de configuraciones en las cuales los estudiantes deben ser seccionados. Un estudiante pidiendo un curso debe asistir a una única configuración de ese curso, y para asistir a la configuración debe asistir a una sola clase de cada una de las sub-partes de la configuración. Además de esto las clases pueden tener relaciones padre-hijo, lo que significa que si se asiste a una clase de una sub-parte que tiene una referencia a otra clase padre de otra sub-parte, entonces se debe asistir a la clase padre también. Las instancias pueden venir acompañadas de un conjunto dinámico de restricciones de distribución de distintos tipos, las cuales se aplican sobre un conjunto de clases y pueden afectar en configuraciones de tiempos factibles, días, semanas y salones posibles. Estas restricciones están curadas específicamente para representar las necesidades de una institución, lo que permite acoplar restricciones impuestas por el personal docente. Las instancias están caracterizadas por consistir en grandes volúmenes de datos y una alta cantidad de restricciones duras. En la siguiente sección se revisarán los enfoques completos que ganaron los primeros dos lugares de la competencia, mientras que en la sección 2.1.2 se revisa la solución del tercer lugar. Además de esto, también se cubren enfoques en casos reales de otras universidades que no están ligados a la competencia pero que son interesantes de abordar debido a su similitud con este trabajo.

2.1.1. Enfoques Completos

La propuesta matheurística de [Rappos *et al.*, 2022] ganó el segundo lugar. Debido al gran volumen de datos de las instancias, la matheurística consiste en dos etapas: obtención de una solución factible (cumpliendo con restricciones duras) y mejora de la solución mediante búsqueda local que minimiza las penalizaciones definidas por cada instancia en bloques de tiempo, salones, distribución y topes de horarios de los estudiantes. Se usan solvers comerciales (Gurobi e IBM CPLEX) de manera intercalada. La mayoría de las instancias se resolvieron usando un único solver, pero se encontró que si se llegaba hasta el punto donde ya no había mejoras, cambiar el solver ayudaba a llegar a mejores soluciones. Se lograron resolver veintinueve de treinta instancias. Para instancias de tamaño pequeño y mediano lograron llegar a soluciones factibles en dos horas o menos, y la segunda etapa convergía rápidamente a límites casi óptimos.

[2] https://www.itc2019.org/home

Los ganadores de la competencia [Holm *et al.*, 2022] proponen un modelo basado en grafos e hicieron una reducción de datos redundantes mediante vértices fijos (clases con un único par salón-tiempo factible) y cliques que ayudan a eliminar asignaciones infactibles cuando múltiples clases entran en conflicto. Posteriormente se construyen los grafos de conflictos para las restricciones clase-tiempo (topes horarios), clase-salón (topes de lugar físico) y clase-tiempo-salón (topes de tiempo de traslado entre clases). Luego, en vez de modelar estos conflictos en pares, los autores agregan los conflictos en restricciones tipo clique –representando un conjunto mutuamente excluyente de elecciones de clase-tiempo o clase-salón– y restricciones estrella que representan restricciones de distribución suaves. Para modelar conflictos en los estudiantes, se filtran los pares de clases que son lógicamente imposibles según las reglas de la competencia. La solución propuesta logró reducir el total de restricciones en un 77.18 % en promedio y el total de variables en un 42.96 %. A través de distintas ejecuciones usando Gurobi, la formulación logró encontrar límites superiores e inferiores más ajustados que los de modelos básicos.

2.1.2. Búsqueda Local

En [Goh *et al.*, 2017], un enfoque de dos etapas es propuesto para resolver el PECTP. En la primera etapa, se usa Tabu Search con Muestreo y Perturbación (TSSP en inglés) para generar soluciones factibles. En la segunda etapa, se emplea una variante de Simulated Annealing con Recalentamiento (SAR en inglés) para mejorar la calidad de las soluciones obtenidas durante la primera etapa. Su representación es un mapa de los pares salón-bloque de tiempo hacia los eventos asignados en cada par. La función de evaluación consta del tamaño del conjunto de eventos no asignados más el ratio de topes horarios entre eventos. El proceso de construcción del horario de clases comienza con asignar todos los eventos del conjunto de eventos no asignados con TSSP, luego, si todos los eventos se pudieron calendarizar, se aplican mejoras con SAR. En cada iteración de TSSP se selecciona una muestra del conjunto no asignado y para cada evento en la muestra se busca la mejor asignación de salón-tiempo, luego se agregan a la lista tabú aquellos eventos que tienen conflictos con el mejor. Finalmente, si se cumple un criterio de iteraciones, se perturba la solución actual con movimientos de intercambio (swap) y cadenas de Kempe– se construyen cadenas de eventos que se topan entre sí para intercambiar entre dos bloques horarios –, reiniciando la lista tabú. Este proceso se repite hasta que se asignen todos los eventos y no se supere el tiempo máximo de ejecución. Una vez obtenida la solución inicial, se mejora la función de evaluación con SAR. El algoritmo propuesto está diseñado con detección de estancamiento y una estructura de vecindario variante dependiendo del conjunto de datos a usar. El vecindario se genera con movimientos de transferencia de un evento hacia otro bloque salón-tiempo y los mismos intercambios y cadenas de Kempe mencionados anteriormente (con distintas proporciones del vecindario dependiendo de la instancia). La temperatura se inicializa con un factor de la función de evaluación inicial. Cuando se llega al punto de estancamiento, se vuelve a subir dependiendo de la función de evaluación de la solución actual y de un factor de calor. El efecto de esto es que el algoritmo explora más cuando el costo es alto y explota más cuando

es bajo. El conjunto de datos usado consta de las competencias ITC 2002, ITC 2007 y el conjunto propuesto por [Socha *et al.*, 2002]. En todas las instancias, TSSP logra encontrar una solución inicial factible con mejor tiempo total que TS normal y, combinada con SAR, logra ser bastante competitiva con solvers del estado del arte.

Los autores de [Song *et al.*, 2018] proponen un enfoque de búsqueda local iterativa. Su representación consta de un conjunto de arreglos de cada uno de los períodos de tiempo del tamaño de los salones. Cada celda del arreglo tiene como valores los eventos asignados al salón. Su método de inicialización, al contrario del estudio de [Goh *et al.*, 2017], permite obtener eventos que no se asignan a un par salón-tiempo, para lo cual la fase de búsqueda local intenta reparar. El algortítmo de inicialización recorre cada período secuencialmente, selecciona un subconjunto de eventos no asignados que tenga la menor cantidad de topes posibles y se asignan a algún salón adecuado. La función de evaluación es la cantidad de eventos no asignados a la calendarización. En cada iteración se hace una búsqueda local con SA y se hace una mutación a la solución encontrada. Si la solución mutada es mejor que la solución actual, entonces la actual pasa a ser la solución mejorada. El proceso se repite hasta que se encuentra el mínimo o se alcanza el tiempo de ejecución máximo impuesto. El movimiento que genera la estructura del vecindario es uno de reconstrucción. Se selecciona aleatoriamente un evento asignado y se elimina. Esto, para después seleccionar eventos del conjunto no asignado que no generen topes horarios con los otros eventos. Solamente aquellos eventos a los que se les pudieron asignar salones permanecen en la representación. El criterio de aceptación de una solución considera la diferencia entre la cantidad de eventos asignados. Si no hay diferencias en la cantidad de eventos asignados, se considera la diferencia entre la cantidad de estudiantes que atienden a eventos y el grado de colisiones entre eventos de cada solución. Una vez que SA encuentra el mínimo local, se aplica una perturbación que busca mejorar la solución. Esta modificación consiste en hacer intercambios factibles de eventos pertenecientes a dos periodos seleccionados al azar. Estos intercambios pueden ser considerando un único evento, todos los periodos (y en consecuencia todos los eventos) o bien con una fracción de los períodos. La idea es que esta perturbación explore nuevas soluciones en el espacio de búsqueda para las siguientes iteraciones. Se usaron sesenta instancias propuestas por [Lewis y Paechter, 2007] que fueron clasificadas en instancias pequeñas ($200 \leq |A| \leq 225$, $|R| \in \{5, 6\}$), medianas ($390 \leq |A| \leq 425$, $|R| \in \{10, 11\}$) y grandes ($1000 \leq |A| \leq 1075$, $25 \leq |R| \leq 28$) El algoritmo logró encontrar soluciones factibles en todas las instancias pequeñas y medianas, mientras que no pudo encontrar soluciones factibles para dos de las instancias grandes. Pruebas de Wilcoxon confirmaron que el algoritmo es estadísticamente superior a otros métodos y midiendo el efecto del uso de las tres posibilidades de perturbación, encontraron que la técnica de perturbación que ofrece el mejor balance entre tiempo de cómputo y exploración fue usando una fracción de los períodos.

Por último, [Sylejmani *et al.*, 2023] proponen un algoritmo para la ITC 2019 que logró obtener el tercer lugar de la competencia. Se usa SA clásico agregando penalizaciones por restricciones rotas. La representación usada consiste en tres vectores. Un vector mapea cada clase con el salón al que está asignada, otro vector mapea las clases con el tiempo asignado

y, finalmente, un vector que mapea las clases con la cantidad de estudiantes inscritos en ellas. Para mantener en memoria las asignaciones de los estudiantes, se tiene un vector del tamaño de su listado de cursos que indica en qué configuración del curso va el estudiante. Además de esto, se tiene otro listado que indica a qué subpartes de la configuración asiste y a qué clase queda inscrita cada subparte. Para evaluar restricciones de distribución, se usa una tupla que contiene la penalización de restricciones duras y blandas. La solución inicial se obtiene determinísticamente asignando a cada clase el primer salón y el primer bloque de tiempo, ordenados de manera ascendente según la penalización que tengan para cada clase. Para seccionar a los estudiantes sin romper relaciones padre-hijo, se crean cadenas de clases que respetan esas relaciones. Luego cada estudiante es asignado a una cadena en orden creciente de las cadenas. La función de evaluación consta de tres tipos de penalizaciones: violación de restricciones duras y suaves, sobre la asignación de estudiantes a una clase y penalizaciones de búsqueda. La función es normalizada para aplanar valores muy altos. El vecindario se construye aplicando aleatoriamente alguna de las tres posibles modificaciones: cambiar aleatoriamente el índice de tiempo de una clase, cambiar aleatoriamente el valor del salón de una clase a otro o cambiar la configuración de un estudiante a una nueva. Cuando el algoritmo se queda estancado y se llega al máximo de tiempo por ciclo, se guardan en memoria las características de la solución actual y se reinicia la búsqueda con una temperatura baja. El algoritmo propuesto logró encontrar soluciones factibles para todas las instancias y mostró ser bastante competente en comparación con las técnicas completas vistas en la sección anterior.

2.1.3. Enfoques Evolutivos

En [Wong *et al.*, 2022] se propone un algoritmo genético (AG) para el Campus Internacional Sabah-Labuan de la Universidad de Malasia (abreviada del inglés como UMS-LIC). El objetivo es minimizar la violación de restricciones suaves: (1) un estudiante no va solamente a una clase en un día ni (2) atiende a más de dos clases consecutivas, ni (3) debe asistir a una clase en el último período del día. La representación del cromosoma consta del mismo mapa de pares salón-bloque hacia el evento asignado para cada par propuesto por [Goh *et al.*, 2017], por lo que se asegura el cumplimiento de restricciones duras. La población inicial se obtiene mediante TSSP. La condición de término del algoritmo es encontrar el óptimo (una solución que no viole ninguna restricción suave) y la solución con mejor ajuste es la que se retorna como la solución final. Se implementan dos métodos de selección: dos individuos con los mejores valores de la función de evaluación son seleccionados automáticamente como padres; además de esto, también se incorpora selección por ruleta. El operador de cruzamiento intercambia eventos entre los padres y el punto de corte se selecciona aleatoriamente. Se implementan dos operadores de mutación que generan dos individuos nuevos a partir de uno. El primero es un movimiento de transferencia que intenta reasignar un evento a un bloque vacío mientras que el segundo hace un intercambio entre dos eventos. El conjunto de datos consiste en el proceso de inscripción de la universidad para el primer semestre del año 2018, el cual tiene las siguientes características: $|S| = 1993$, $|C| = 144$, $|T| = 35$ y

$|R| = 24$. Los resultados muestran que el primer método de selección logra mejores valores para la función objetivo que la selección por ruleta; además, se encontró que el algoritmo en general también supera la calendarización generada manualmente.

Las técnicas de coloreo de grafos son ampliamente usadas para resolver estos tipos de problemas. Recientemente, [Jain *et al.*, 2025] propone un algoritmo híbrido que agrega una componente de búsqueda local a las soluciones obtenidas mediante un AG. La representación del grafo considera como colores los bloques de tiempo disponibles; los vértices son las clases y las aristas representan conflictos entre clases. Para el AG, un cromosoma se representa con un arreglo de las clases donde cada celda lleva un par profesor-bloque de tiempo. La función de evaluación es la cantidad de restricciones suaves no respetadas. Una vez obtenida la población de soluciones iniciales, se incorpora un proceso de SA para mutarlas a mejores valores de la función de evaluación. La representación consta de una matriz de asignación de profesores y salones a bloques de tiempo. Cada celda contiene la clase asignada a un profesor y a un salón. Esta representación es muy útil para detectar rápidamente qué tan factible es la solución y facilita movimientos de intercambios entre clases o intercambios de columnas enteras (que son equivalentes a un intercambio de bloques horarios). Sin embargo, su validación experimental es limitada, pues no transparenta el origen del conjunto de datos usado y las simulaciones realizadas solo son una prueba de concepto que considera de cincuenta a setenta clases para poder evaluar costos a medida que el tamaño del problema aumenta.

2.1.4. Acercamientos multi-objetivo

Diversos enfoques multi-objetivo se han presentado en la literatura [Thepphakorn y Pongcharoen, 2023, Gülcü y Akkan, 2020, Zhu *et al.*, 2026]. En esta sección se revisan estudios actuales que buscan solucionar el problema en contextos específicos reales usando algunas de las técnicas mencionadas en la sección 1.1.4.

En el trabajo realizado por [Tung Ngo *et al.*, 2021] se propone un framework de programación de cursos centrado en el estudiante. Dentro de su contexto, la Universidad FTP de Hanoi, Vietnam toma en cuenta las preferencias de tiempos de estudio del estudiante y su nivel de prioridad a la hora de inscribirlo en cursos. Si un estudiante repite un curso, su nivel de prioridad baja. Los cursos se abren de manera dinámica basándose en la demanda estudiantil y en que esta se encuentre en sus rangos de viabilidad. Cada clase abierta tiene que ser seccionada de manera tal que se mantenga un rango estricto de tamaños de grupos, desde quince hasta treinta estudiantes para balancear costos de enseñanza y calidad de educación. El problema se modela mediante variables enteras (para cursos abiertos y prioridades estudiantiles) y binarias (para el seccionamiento de los estudiantes). El estudio considera cinco objetivos: maximizar la inscripción en cursos prioritarios, maximizar las prioridades de tiempo de los estudiantes, balancear los tamaños de las secciones (minimizando la varianza de las secciones de un mismo curso), maximizar un puntaje de *partes del día* que penaliza programaciones que obliguen a los estudiantes a realizar múltiples viajes dentro del campus en

un mismo día y minimizar el total de cursos abiertos para prevenir cuellos de botella administrativos. Para prevenir que quienes toman decisiones deban evaluar y seleccionar frentes de Pareto de manera manual, se reescala el problema mediante programación por compromisos, escalando el vector objetivo a una distancia L_2 relativa al punto utópico. La métrica se minimiza mediante un algoritmo genético, donde la representación del cromosoma consiste en una estructura de tres partes, dos matrices y un vector. La primera estructura es una matriz de los salones y bloques horarios donde cada celda contiene el curso asignado al par salón-bloque. Luego, una cuadrícula de estudiantes y secciones que mapea a qué sección de cada curso queda inscrito cada estudiante y finalmente un vector binario indicando si un curso se ofrece o no. Debido a que los operadores genéticos generan individuos no factibles, se incorpora un operador de reparación que reasigna de manera dinámica a los estudiantes desde secciones que no cumplen el mínimo viable (quince estudiantes) a secciones viables (menos de 30 estudiantes) o bien desinscribe a los estudiantes cuando no hay bloques libres de topes. Los experimentos se realizan sobre conjuntos de datos reales de la universidad FTP, consistiendo de datos seleccionados de más de 3000 estudiantes en el departamento de Tecnologías de la Información (TI) el cual tiene casi 150 cursos. Se experimentó en dos escalas: primero con los datos del departamento completo y segundo con un subconjunto de 500 estudiantes y 5 cursos.Se hicieron quince ejecuciones independientes del algoritmo en ambos casos de prueba y la evaluación de estos experimentos muestra que el algoritmo converge en alrededor de 700 generaciones para el subconjunto más pequeño mientras que para el más grande tarda unas 900 generaciones. Además de esto el tiempo total de ejecución rondaba un promedio de 2 minutos consistentemente a través de las distintas ejecuciones.

El trabajo de [Dunke y Nickel, 2023] toma en consideración un esquema de horarios estudiantiles que contiene las preferencias horarias de los estudiantes y requerimientos operacionales modernos como lo es la implementación de clases en línea o híbridas (parte en línea, parte presencial), lo que en combinación con la alta demanda hace que el problema sea altamente complejo. Para resolverlo, se plantea una jerarquía de tres niveles que primero evalúa la disponibilidad esperada de los cursos con sus restricciones de tiempo, luego genera la programación de esos cursos en salones y bloques de tiempo para finalmente obtener una programación individual para cada estudiante basándose en sus demandas y preferencias de tiempo. Al igual que [Tung Ngo *et al.*, 2021], un problema de cinco criterios es optimizado en las siguientes métricas: puntaje del curso (basado en los puntajes de tiempos definidos por la facultad), balance de trabajos didácticos, puntaje de los estudiantes (medido por sus preferencias de tiempo cumplidas), penalizaciones de ventanas de tiempo muy largas y evitar días inactivos (penalizando días que tienen muy pocas clases). Para resolverlo, se implementan tres modelos de programación lineal entera (PLE) para primero resolver la planificación de las asignaturas, luego la calendarización de las clases y finalmente los horarios de los estudiantes. Estos modelos son mejorados iterativamente en un algoritmo genético que optimiza el vector objetivo simultáneamente. Como ejecutar el solver PLE repetidamente es computacionalmente costoso, una red neuronal es incorporada como un filtro de preselección que estima los vectores objetivo para las configuraciones generadas, rechazando candidatos inferiores antes de seguir con la ejecución del solver. La solución fue validada en datos reales del Instituto de Tecnologías de Karlsruhe (KIT) y veinte benchmarks sintéticos. Para la instancia

real, se consideran 4 programas de estudio con 452, 122, 413 y 102 estudiantes respectivamente (haciendo un total de 1089 estudiantes). La malla curricular consiste de 22 cursos que requieren 41 actividades académicas, 9 de los cuales ofrecen tutorías que requieren 170 clases en total. Se consideran 30 bloques de tiempo (6 bloques por cada uno de los 5 días de la semana). Hay 106 salones disponibles con capacidades entre 105 y 734 asientos, de los cuales solo 30 pueden acomodar más de 450 estudiantes. Para las tutorías hay 279 salones con capacidades de 24 a 30 asientos. Los profesores y estudiantes proveen preferencias para cada bloque de tiempo con un puntaje de $\{1, 2, 3, 4, 5\}$. Las instancias generadas, por otro lado, varían entre $\{2, 4, 6, 8\}$ programas de estudio que tienen una cantidad promedio de estudiantes posibles de $\{50, 100, 200, 400, 800\}$ para cada programa. De esta forma se pueden generar 20 instancias de tamaño variable. La cantidad de salones para clases es el doble del total de programas de estudio mientras que para las tutorías se agrega un factor de la cantidad de estudiantes del programa. Para cada instancia puede haber un total de 10 a 15 bloques de tiempo elegidos aleatoriamente. Los resultados muestran que para la instancia real, el promedio de los puntajes de las preferencias de los alumnos y profesores fueron superiores a 4, mientras que para los casos artificiales entre 3 y 4 puntos. También se logró mantener ventanas de tiempo cortas (con un promedio de 2 ventanas por alumno) y el promedio de días vacíos por debajo de un 0.5 para cada estudiante. La red neuronal si bien no muestra tener un efecto considerable en encontrar mejores soluciones, amplifican la diversidad de soluciones del frente de Pareto y además implica un ahorro de evaluaciones costosas debido a que descarta configuraciones poco prometedoras. Resolver el problema de calendarización de los estudiantes es la parte más compleja del método y los autores lo identifican como el cuello de botella, pues para programas que tengan más de 800 estudiantes resolverlo con métodos exactos se vuelve prohibitivo.

La investigación de [Hafsa *et al.*, 2021] aborda el complejo problema de calendarización automática en entornos de capacitación corporativos. A diferencia de una calendarización de cursos tradicional que se enfoca principalmente en horarios semestrales, el trabajo se realiza en torno al problema de calendarización en la academia de Mandarine, la cual se dedica a ofrecer cursos de capacitación para distintas herramientas en contextos empresariales. El problema se aborda desde el enfoque de una tarea de optimización multi-objetivo que considera cinco criterios: maximizar sesiones de capacitación calendarizadas, minimizar penalizaciones por restricciones suaves (tales como configuración de salones y discordancia de profesores), minimizar el uso de profesores externos, balancear la distribución de trabajo de los profesores y minimizar el uso de salones físicos. Se usan NSGA-II y NSGA-III para optimizar el vector objetivo de manera simultánea, con el objetivo de comparar la eficacia de ambos algoritmos para este contexto. Se usa una representación jerárquica por niveles, donde el nivel más bajo consiste en una instancia de sesión para un evento– una tupla que contiene los recursos asignados, día asignado, horas de inicio y fin, ubicación geográfica y número de secuencia–. Cada curso tiene un conjunto de sesiones y la solución consta de la colección de sesiones para cada curso. Para generar horarios iniciales factibles, una heurística constructiva basada en prioridades asigna recursos mediante un filtro de memoria basado en Tabu Search. El operador de cruzamiento hace un intercambio de clases entre ambos padres. Como los hijos resultantes pueden ser infactibles, pues los eventos pueden no tener la misma

cantidad de sesiones, por lo que la solución del hijo puede quedar con sesiones donde ciertos recursos no están disponibles debido al cambio en el orden de eventos. Se aplican operadores de reparación que intentan cambiar los recursos y tiempos de las sesiones afectadas. El operador de mutación selecciona una clase aleatoria y altera el salón asignado, el profesor o el bloque de tiempo según las reglas de la institución. El conjunto de pruebas consta de 60 instancias pequeñas (100 sesiones, 20 cursos, 60 días), medianas (500 sesiones, 50 cursos, 152 días) y grandes (1050 sesiones, 70 cursos, 244 días) y cada una tiene a su disposición 269 profesores, 230 salones y 14 dispositivos de enseñanza (computadores, tablets, entre otros). Evaluaciones empíricas a través de todas las instancias demuestran que NSGA-II supera a NSGA-III estadísticamente en la convergencia del hipervolumen y en la diversidad de soluciones a medida que la escala del problema aumenta.

La revisión anterior muestra que los trabajos más recientes convergen en dos ideas centrales para el problema abordado: incorporar explícitamente las preferencias de los estudiantes como un criterio de optimización de primer orden, y emplear algoritmos evolutivos multi-objetivo para producir conjuntos de soluciones de compromiso en instancias de gran tamaño, donde los métodos exactos dejan de ser viables. Estos antecedentes fundamentan las decisiones de diseño adoptadas en esta memoria. En el siguiente capítulo se presenta la propuesta de solución: un algoritmo NSGA-II especializado para la variante multi-objetivo del PECTP, detallando la representación del cromosoma, la inicialización, el proceso de evaluación y los operadores de transformación.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

Este capítulo presenta la propuesta de solución para la variante multi-objetivo del PECTP descrita en el capítulo anterior: un algoritmo NSGA-II especializado. Primero se expone el funcionamiento general de NSGA-II, que constituye la base del método, y a continuación se detallan las adaptaciones diseñadas para este problema en particular: la representación del cromosoma, la estrategia de inicialización, el proceso de evaluación de las dos funciones objetivo y los operadores de cruzamiento y mutación. Cada uno de estos componentes se orienta a manejar simultáneamente la cobertura de los cursos preinscritos y la satisfacción de las preferencias de tiempo libre de los estudiantes.

En la siguiente sección se profundiza en el funcionamiento de NSGA-II [Deb *et al.*, 2002], uno de los algoritmos más utilizados en la literatura. Al 11 de octubre de 2026, NSGA-II tiene 44562 citas en artículos. Es conocido debido a su buen rendimiento en problemas de dos o tres objetivos y su flexibilidad y facilidad de implementación para distintos tipos de problemas, existiendo implementaciones del algoritmo específicas para PECTP (véase [Leite *et al.*, 2015] y [Thepphakorn *et al.*, 2016]).

3.1. Non-dominated Sorting Genetic Algorithm II

El contraste principal entre NSGA-II y su precursor [Srinivas y Deb, 1994] radica en la complejidad total del algoritmo y mejoras en el rendimiento de éste mediante el elitismo. Si se tiene un problema de M objetivos y una población de N soluciones, NSGA tiene una complejidad de tiempo $O(MN^3)$, mientras que NSGA-II es $O(MN^2)$. Además de esto, la diversidad de la población para la siguiente generación dependía de un radio σ_{share} que se requería como entrada del algoritmo, algo que también cambió con esta nueva versión.

El algoritmo 1 muestra la estructura principal de NSGA-II. Lo primero que hace el algoritmo es inicializar la población de tamaño p en la línea 2 y se crea la población de hijos usando los mecanismos de selección, cruzamiento y mutación en la línea 3. En cada generación, se combina la población de padres con la población de hijos (línea 5) en R_g para luego clasificar esos individuos en frentes no dominados en la línea 6 para obtener la población en la siguiente generación (línea 7). Luego en la línea 8 se obtiene la población de hijos de la generación nueva y se repite el proceso hasta llegar al máximo de generaciones. Una vez obtenida la población final y sus descendentes, se repite el proceso de combinación y clasificación y el algoritmo retorna el primer frente no dominado.

El proceso de clasificación y selección de individuos para la generación siguiente se ilustra en la figura 4 y consta de dos partes: ordenamiento de frentes no dominados y ordenamiento por distancia de hacinamiento. El ordenamiento por no dominancia itera secuencialmente

Algorithm 1 Estructura de NSGA-II

Input: $p, MaxGen, p_c, p_m, p_m^{ts}, p_m^{sc}$

```
g ← 0;
Generate initial population P_g(p);
Generate offspring Q_g(p) (selection, crossover and mutation);
while g < MaxGen do
    R_g = P_g ∪ Q_g;
    Classify non-dominated fronts F = (F_1, F_2, ...);
    Obtain new population P_{g+1}(p);
    Generate offspring Q_{g+1}(p) (selection, crossover, mutation);
    g ← g + 1
end while
R_g = P_g ∪ Q_g;
Classify non-dominated fronts F = (F_1, F_2, ...);
Return F_1
```

en las soluciones de la población combinada R_g, calcula el número de soluciones n_s que dominan a la solución $s \in R_g$ y el conjunto de soluciones dominadas por ella (S_s), obteniendo distintos frentes de Pareto en esa población de tal manera que el primer frente son los individuos no dominados, el segundo frente los que son dominados solo por el primero y así sucesivamente. Esta numeración de frentes define el rango (*rank*) de un individuo.

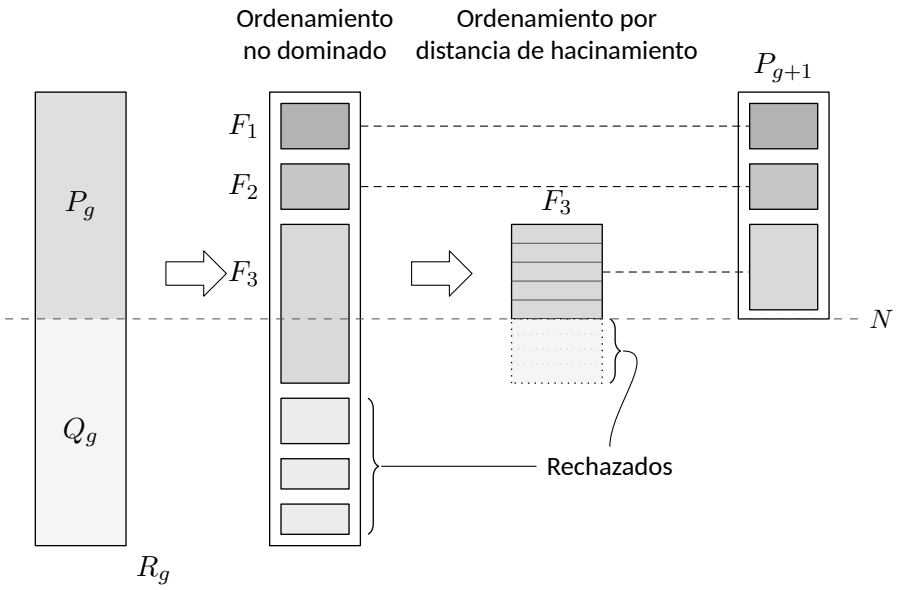


Figura 4: Selección de individuos para la generación $g + 1$. Fuente: Elaboración Propia

En la primera versión del algoritmo propuesta en [Srinivas y Deb, 1994], la diversidad de la población dependía de σ_{share}, un parámetro que funcionaba como radio de distancia fenotípica entre individuos, esto para poder controlar qué tan diversa era la población resultante

del proceso de clasificación. En NSGA-II se define una métrica de estimación de la densidad de soluciones alrededor de alguna solución de la población y se calcula la distancia promedio entre dos puntos en cualquiera de los lados de la solución, definida como distancia de hacinamiento. Esto evita el uso de parámetros externos para controlar diversidad. Una vez que todos los individuos tienen asignada una distancia de hacinamiento, se seleccionan aquellos con mayor valor. El algoritmo 2 muestra el cálculo de la distancia de hacinamiento para cada individuo de una población P. Este proceso viene después de hacer el ordenamiento no dominado en el bloque de clasificación.

Algorithm 2 Distancia de Hacinamiento [Deb *et al.*, 2002]

Require: Población $\mathcal{P}$
  $l = |\mathcal{P}|$
  for all i, set $\mathcal{P}[i]_{distance} = 0$ **do**
  end for
  for all obj m **do**
    $\mathcal{P} = \text{sort}(\mathcal{P}, m)$
    $\mathcal{P}[1]_{distance} = \mathcal{P}[l]_{distance} = \infty$
    for $i = 2$ to $l - 1$ **do**
      $\mathcal{P}[i]_{distance} = \mathcal{P}[i]_{distance} + (\mathcal{P}[i+1].m - \mathcal{P}[i-1].m)/(f_m^{max} - f_m^{min})$
    end for
  end for

Donde f_m^{max} y f_m^{min} son los valores máximos y mínimos para el objetivo m y sirven para normalizar la distancia de hacinamiento para cada individuo. La operación sort$(\mathcal{P}, m)$ de la línea 5 ordena de manera ascendente la población con respecto al objetivo m.

Al finalizar el cómputo del *rank* y *distance*, se usa un operador *crowded-comparison* para saber qué individuos sobreviven para obtener la población P_{g+1} que se ve en la figura 4. Este operador (simbolizado en el artículo como $\prec_n$) guía el proceso de selección usando ambos atributos de los individuos (*rank* y *distancia*). El operador de orden parcial $i \prec_n j$ se define de la siguiente manera:

[Operador $\prec_n$] Un individuo i es preferible sobre un individuo j, notado como $i \prec_n j$, si y solo si:

$$i \prec_n j \iff (i_{rank} < j_{rank}) \vee ((i_{rank} = j_{rank}) \wedge (i_{distancia} > j_{distancia}))$$

Esto quiere decir que, entre soluciones que tengan distinto *rank*, se elige la que tenga el menor (pues pertenece a un frente menos dominado). Si ambas pertenecen al mismo frente (igual *rank*), se elige la que tenga menos densidad de soluciones a su alrededor.

La población combinada $R_g = P_g \cup Q_g$ de la figura 4 es de tamaño $2N$ (con N el tamaño de la población). Se realiza el ordenamiento no dominado y se obtienen los frentes F_1, F_2,

F_3 y así sucesivamente–estos frentes pueden incluir soluciones tanto de P_g como de Q_g, por ende este ordenamiento implica el elitismo del algoritmo–. Una vez ordenados por no dominancia, se prioriza el paso de las soluciones en F_1 a la siguiente población P_{g+1}. Si $|F_1| < N$, todas las soluciones pasan a la siguiente población. Se hace sucesivamente con los frentes F_2, F_3 hasta F_l. Si no todos los individuos del frente F_l—en el caso ilustrado, F_3—logran pasar a la siguiente debido a la restricción de N soluciones por población, entonces son ordenados usando la regla de comparación 3.1 de manera descendente y elegimos las mejores soluciones para rellenar los espacios que quedan.

La complejidad total del algoritmo es de $O(MN^2)$ donde M es la cantidad de objetivos y N la cantidad de soluciones de cada población. Es una optimización considerable y escalable en el tamaño de la población con respecto a la de la primera versión, la cual tiene una complejidad de $O(MN^3)$.

3.2. Representación del cromosoma

La representación del cromosoma consiste en una matriz de $|R| \times |T|$ en la que cada celda $c_{i,j}$, con $i = \{1, \ldots, |R|\}$, $j = \{1, \ldots, |T|\}$, almacena el identificador de una actividad académica $a \in A$ que se programa en el salón i y el bloque horario j. Se inserta una actividad vacía especial cuando no hay actividades programadas. La figura 5 muestra un ejemplo de representación del cromosoma para cinco salones, cinco bloques horarios y ocho actividades académicas. Esta representación permite asegurar que las restricciones (4) y (6) hasta (10) se satisfacen, ya que solo una actividad puede asignarse a un par salón–bloque horario, y las actividades no se repiten en la matriz.

	T1	T2	T3	T4	T5
R1	A2			A7	
R2	A5			A3	A6
R3			A1		
R4		A8			
R5		A4			

Figura 5: Representación del cromosoma de un individuo arbitrario. Fuente: Elaboración propia

3.3. Inicialización

Se utiliza un algoritmo voraz para llenar la matriz de modo que las actividades que pertenecen a un mismo curso no ocupen el mismo bloque horario. Se muestrea un par aleatorio de salón–bloque adecuado desde R_a y T para garantizar que solo se elijan salones factibles

al programar una actividad académica. El algoritmo 3 muestra el pseudocódigo de este proceso. Las líneas 1 a 4 inicializan una solución vacía. Luego, para cada curso $c \in C$ y cada actividad $a \in A_c$, se elige de forma aleatoria un salón y un bloque horario factibles desde R_a y T (líneas 9 y 10). Si el salón no está disponible en el bloque horario, o hay otra actividad del mismo curso en el mismo bloque, la actividad no se asigna y el proceso se repite hasta lograrlo (esto se representa mediante el ciclo *while* que comienza en la línea 8 y termina en la línea 15). En caso contrario, la actividad se programa y el bloque horario se marca como ocupado (líneas 12 y 13). Cabe destacar que las marcas de bloques horarios son independientes para cada curso, por lo que actividades de cursos distintos pueden compartir bloques horarios, pero actividades de un mismo curso no.

Algorithm 3 Inicialización del cromosoma

Input: individual

```
1:  for r ∈ R do
2:      for t ∈ T do
3:          individual.gene[r, t] ← ∅
4:      end for
5:  end for
6:  for all c ∈ C do
7:      for all a ∈ A_c do
8:          while a is not assigned do
9:              r ← choice(R_a)
10:             t ← choice(T)
11:             if individual.gene[r, t] is ∅ and t is free then
12:                 individual.gene[r, t] ← a
13:                 mark t as occupied
14:             end if
15:         end while
16:     end for
17: end for
```

3.4. Evaluación

Para evaluar las soluciones, primero se debe inscribir a los estudiantes en sus respectivos cursos, asegurando que las restricciones (6), (11), (12) y (14) a (23) se respeten. Este problema se aborda mediante un proceso de evaluación del cromosoma que (1) inscribe a los estudiantes, (2) evalúa cada objetivo y luego (3) verifica cualquier inviabilidad en las restricciones restantes.

3.4.1. Heurística de inscripción de estudiantes

Como se describió anteriormente, el primer paso del proceso de evaluación es conocer, para cada estudiante $s \in S$, los cursos que cursa y que pertenecen a su respectivo conjunto de preferencias de cursos, C_s. Además, se incluye en la estructura del individuo un arreglo de arreglos `student_courses`, donde cada entrada representa una asignación binaria de un estudiante a un curso de C_s. La heurística consiste en resolver un problema de *clique máxima* (MC) [Cormen *et al.*, 2022, Pardalos y Xue, 1994] para cada subgrafo de no-conflicto de las preferencias de cursos. El subgrafo se construye de manera que dos cursos quedan conectados por una arista si **no** tienen actividades que colisionen en el mismo bloque horario. Durante el procedimiento de inicialización, las actividades de un mismo curso no colisionan, y los operadores también están diseñados para no romper esta estructura. Por lo tanto, no existen colisiones internas dentro de un curso. El problema de clique consiste en encontrar un subconjunto de vértices tal que formen un subgrafo completo. El problema de clique máxima (MC) busca encontrar el mayor de estos subconjuntos. Para este caso, esto significa inscribir al estudiante en la mayor cantidad de cursos posible. La figura 6 presenta un ejemplo para un estudiante que busca inscribir cinco cursos (cinco nodos). Se puede observar que, para este grafo, MC inscribirá al estudiante en los cursos 1 a 4, dejando el curso 5 fuera de las inscripciones, ya que tiene conflictos con todos los cursos excepto el curso 4.

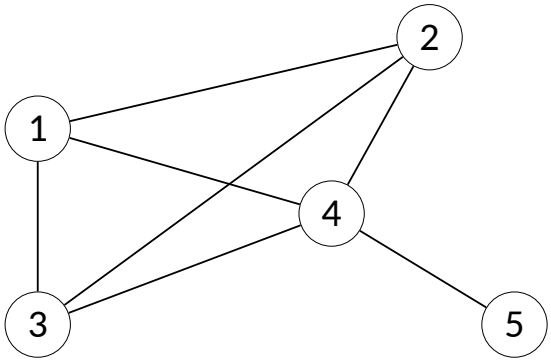


Figura 6: Ejemplo de subgrafo de preferencias de cursos para un estudiante que solicita los cursos 1 a 5. El algoritmo de clique máxima inscribirá al estudiante en los cursos $\{1, 2, 3, 4\}$ porque forman un grafo completo, es decir, no tienen actividades que colisionen.

El problema inmediato que surge al usar un algoritmo directo de MC, como Bron-Kerbosch [Bron y Kerbosch, 1973], es que estos algoritmos tienen complejidad exponencial, lo que vuelve el proceso computacionalmente ineficiente para toda la población, considerando que la inscripción se ejecuta para cada estudiante y que debe tener en cuenta varias restricciones. Por esto, se implementa el algoritmo 4, que retorna de forma indirecta la MC que puede encontrarse dadas las restricciones espaciales, temporales y de regularización. El arreglo course_fill (línea 1) lleva la cuenta de cuántos estudiantes se han inscrito en cada curso. Luego, para todos los estudiantes, crea un arreglo que registre los bloques horarios en los que asisten a actividades académicas (línea 3). Para cada una de sus preferencias de curso, los inscribe (i) si el curso no está a capacidad completa (líneas 5 a 7), (ii) si no están

asistiendo a actividades de otros cursos en los bloques horarios donde hay actividades programadas, (iii) si hay suficiente espacio en el salón de cada actividad vinculada al curso (líneas 8 a 16) y (iv) si la suma de cursos inscritos es menor que el máximo permitido (líneas 17 a 20). Al inscribir a los estudiantes, el algoritmo actualiza la estructura `student_courses`, el contador course_fill y marca los bloques horarios en los que se programa cada actividad académica del curso (líneas 21 a 26). Tanto la función find_timeslot como la función find_room, que aparecen en las líneas 10, 11 y 22, se implementan como un mapa en memoria previo a este procedimiento, de modo que la búsqueda de pares actividad–bloque horario y actividad–salón pueda realizarse en tiempo constante. Este algoritmo asegura que, para cada estudiante $s \in S$, solo se inscriban cursos pertenecientes a C_s y, dado que no tienen colisiones de bloques horarios, se cumplen las restricciones (11), (12) y (15) a (21). También asegura que las restricciones (14) y (23) sean respetadas, ya que un estudiante no se inscribe en un curso si el salón de una de sus actividades ya está lleno o si ya ha inscrito el máximo número de cursos permitido. Para cada estudiante $s \in S$, la complejidad de la parte de inscripción es $O(|C_s|\,|\bigcup_{c\in C_s} A_c|)$, lo que daría una complejidad temporal total de $O(p \times |S| \times |\bigcup_{s\in S} C_s| \times |\bigcup_{c\in C_s} A_c|)$ para toda la población.

3.4.2. Procedimiento de evaluación

La satisfacción de las preferencias de tiempo libre se evalúa primero mediante un enfoque de insatisfacción, en el que se cuenta cada actividad a la que el estudiante asiste en un bloque horario que prefiere tener libre. Para ello, se define α_s como la razón entre las actividades asignadas a preferencias de tiempo libre y el total de preferencias de tiempo libre ($|T_s|$) del estudiante $s \in S$. Así, la insatisfacción media se calcula como $\bar{\alpha} = \frac{1}{|S|}\sum_{s\in S} \alpha_s$.

Para calcular el objetivo de satisfacción de cursos, primero se agrega la contribución β_s que cada estudiante $s \in S$ aporta al objetivo, sumando su respectivo `student_courses`$[s]$ y dividiendo por $|C_s|$. Luego, la insatisfacción media para ese estudiante es $1 - \beta_s$, por lo que la insatisfacción media total por estudiante $\bar{\beta}$ se calcula como $\bar{\beta} = \frac{1}{|S|}\sum_{s\in S}(1 - \beta_s)$.

3.4.3. Factibilidad

El último paso del proceso de evaluación es evaluar la factibilidad de las restricciones restantes. Como se verá en la sección 3.5, los operadores de mutación y cruzamiento conservan la consistencia establecida durante el proceso de inicialización, y las actividades solo se programan en salones factibles (respetando la restricción (5)). Esto significa que el propio algoritmo asegura que casi todas las restricciones se satisfacen, excepto la restricción (13). El procedimiento para verificar la inscripción mínima por estudiante se implementa mediante el algoritmo 5, donde constr_violation afecta la dominancia del individuo cuando este no es factible.

Algorithm 4 Inscripción de estudiantes

Input: individual

course_fill $\leftarrow$ zeros$(|C|)$
for all $s \in S$ **do**
    for all $c \in C_s$ **do**
        if course_fill $\geq \sigma_c$ **then**
            continue
        end if
        can_enroll $\leftarrow 1$
        for all $a \in A_c$ **do**
            $t \leftarrow$ find_timeslot(individual.gene, a)
            $r \leftarrow$ find_room(individual.gene, a)
            if s is busy in t **or** course_fill$[c] \geq \rho_r$ **then**
                can_enroll $\leftarrow 0$
                break
            end if
        end for
        can_enroll $\leftarrow$ sum(individual.student_courses$[s]) \leq k^s_{\text{máx}}$ **and** can_enroll
        if not can_enroll **then**
            continue
        end if
        individual.student_courses$[s, c] \leftarrow 1$
        course_fill$[c] \leftarrow$ course_fill$[c] + 1$
        for all $a \in A_c$ **do**
            $t \leftarrow$ find_timeslot(individual.gene, a)
            mark t as busy for s
        end for
    end for
end for

Algorithm 5 Verificación de inscripción mínima

Input: individual

for all $s \in S$ **do**
    $k \leftarrow$ sum(individual.student_courses$[s]$)
    if $k < k^s_{\text{mín}}$ **then**
        individual.constr_violation $\leftarrow$ individual.constr_violation -1
    end if
end for

3.5. Operadores de transformación

Los operadores de transformación buscan mantener la estructura establecida por la inicialización del cromosoma, de modo que las actividades de un mismo curso no se programen en el mismo bloque horario. Para ello, la granularidad del operador de cruzamiento debe permitirle heredar la calendarización de los cursos, y la mutación debe usarse para mejorar el nivel de satisfacción. Además, introducimos un factor de satisfacción en el cruzamiento para ayudar a la convergencia del algoritmo.

3.5.1. Cruzamiento

El operador de cruzamiento busca heredar la calendarización de los cursos con mayores porcentajes de inscripción. Comienza seleccionando aleatoriamente un punto de corte p en el conjunto de cursos, luego los ordena según la demanda satisfecha en orden descendente (de modo que se seleccionan primero los cursos con mayor porcentaje de satisfacción de inscripción) y selecciona los primeros p cursos para que los herede el descendiente correspondiente (para el primer padre, el primer hijo; y para el segundo padre, el segundo hijo). Los cursos restantes que no fueron heredados del padre correspondiente se heredan del otro padre, asegurando que sus actividades no reemplacen a las de cursos ya calendarizados. En caso de que una actividad proveniente del segundo padre comparta la celda salón–bloque horario de una actividad heredada del primer padre, primero se programa en un salón distinto pero en el mismo bloque horario; de lo contrario, en el mismo salón pero en un bloque horario distinto; y luego, si no puede heredarse por completo, se asigna en la primera celda factible disponible. El operador es muy preciso y es consistente con la estructura que provee la inicialización. La figura 7 muestra un ejemplo de cruzamiento para un problema de cinco cursos. Los cursos escritos en negrita se heredan del padre 1, mientras que los escritos en cursiva se heredan del padre 2. Se selecciona el punto aleatorio $p = 3$. Los cursos ya están ordenados con fines explicativos. Como se puede observar, el operador está diseñado de modo que es posible que uno de los descendientes herede los mejores niveles de porcentaje de inscripción entre los padres. En este caso, el hijo 1 logra el mejor porcentaje promedio de inscripción. El hijo 2 todavía conserva información importante, ya que su porcentaje de inscripción para el curso 2 es mejor que el del primer hijo. Este escenario asume que todas las actividades pudieron asignarse sin romper ninguna estructura proveniente de los padres, por lo que se preserva cada par actividad–bloque horario. En caso de que no lo hagan–habiendo topes horarios al heredar de ambos calendarios–estos porcentajes de satisfacción pueden estar sujetos a cambios al momento de evaluar ambos hijos.

3.5.2. Mutación

Se implementan tres operadores de mutación: **timeslot_swap**, **schedule_change** y **activity_swap**. El objetivo general de la mutación es lograr menos conflictos de bloques horarios

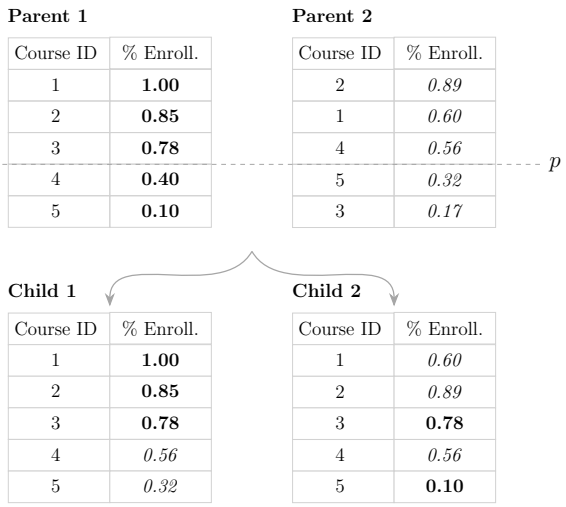


Figura 7: Ejemplo de cruzamiento. Fuente: Elaboración propia

entre cursos y apoyar el objetivo de preferencias de tiempo libre. Los dos primeros operadores dependen de una selección de los peores y mejores bloques horarios, medidos según cuántas preferencias de tiempo libre satisfacen. Para evitar seleccionar siempre los mismos bloques horarios para cada individuo, los `n_timeslot` peores y mejores bloques horarios se seleccionan mediante selección por ruleta (RWS), donde `n_timeslot` es un hiperparámetro fijado en cinco.

El operador de mutación **timeslot_swap** realiza un intercambio directo entre los bloques horarios seleccionados. Esto se hace para que la carga de actividades se distribuya en bloques horarios menos densos en demanda de tiempo libre, y que aquellos con mayor demanda queden menos ocupados de lo que estaban.

El operador **schedule_change** inserta una actividad del peor bloque horario en un salón factible disponible del mejor bloque horario. Si está lleno (es decir, todos los salones factibles ya están ocupados) o hay actividades del mismo curso ya programadas en el bloque horario seleccionado, entonces se selecciona otro mejor bloque horario mediante RWS hasta que pueda moverse. Ambos operadores, **timeslot_swap** y **schedule_change**, exploran el espacio de búsqueda guiados por las preferencias de tiempo libre de los estudiantes. Por lo tanto, se espera tener más actividades programadas en bloques horarios con menos demanda.

El operador **activity_swap** consiste en una búsqueda local (LS) con un enfoque de primera mejora que minimiza el conflicto horario de los estudiantes. El vecindario se genera usando operadores de intercambio entre las celdas de actividades. El intercambio se realiza solo si es factible. Esto significa que si se va a realizar un intercambio entre $a \in A$ y $b \in A$, y r_a y r_b son sus salones correspondientes, el intercambio puede hacerse si y solo si $r_a \in R_b$ y $r_b \in R_a$, y si intercambiar sus bloques horarios no genera un conflicto con otras actividades de sus respectivos cursos. Esta última verificación asegura que el cromosoma mantenga la consistencia con la estrategia de inicialización para maximizar las inscripciones de cursos. Dado

que solo se hacen intercambios entre celdas disponibles y ninguna actividad se intercambia a una celda vacía, este operador explota el espacio de búsqueda.

Los operadores **timeslot_swap** y **schedule_change** tienen probabilidades p_m^{ts} y p_m^{sc} de ocurrir durante una mutación. En caso que ninguna de las dos suceda, la mutación por defecto es **activity_swap**.

Con la representación del cromosoma, la estrategia de inicialización, el proceso de evaluación y los operadores de cruzamiento y mutación queda completamente especificado el algoritmo NSGA-II propuesto. En el siguiente capítulo se valida esta propuesta de forma experimental: se describen los casos de prueba generados, se contrastan los frentes obtenidos con los de un método exacto resuelto mediante AMPL y Gurobi, y se evalúa la calidad de las soluciones mediante indicadores de calidad multi-objetivo.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

En este capítulo se valida experimentalmente la propuesta de solución. El objetivo es evaluar la calidad de los frentes obtenidos por el algoritmo NSGA-II especializado y contrastarlos con los de un método exacto que resuelve el mismo modelo mediante AMPL y el solver Gurobi. El capítulo se organiza en tres partes: primero se describe el conjunto de experimentos –el generador de instancias, los casos de prueba y los parámetros de ejecución–; luego se reportan los resultados en términos de hipervolúmen y tiempos de ejecución; y finalmente se discuten estos resultados, incluyendo un test estadístico que compara el desempeño de ambos métodos.

4.1. Experimentos

Para construir instancias de prueba, se ha implementado un generador de casos simulados con distintas distribuciones de demanda que buscan asemejar la realidad de la Universidad Técnica Federico Santa María.

Los bloques horarios siguen una nomenclatura definida. Un bloque de tiempo tiene una duración de treinta y cinco minutos y un bloque horario factible para una actividad consta de dos bloques de tiempo seguidos. Además, se considera que los posibles días para realizar clases son de lunes a viernes, por lo que se codifican los tiempos disponibles para realizar una clase siguiendo el formato `día_bt1_bt2`, donde `día` es un número entero en el rango $[1, 5]$, `bt1` es el primer bloque de tiempo del horario y `bt2` el segundo, donde $\texttt{bt2} = \texttt{bt1} + 1$. A modo de ejemplo, con veinte bloques de tiempo los horarios disponibles para el día lunes quedan codificados como `1_1_2`, `1_3_4`, ..., `1_19_20`; para el día martes `2_1_2`, ..., `2_19_20` y así sucesivamente hasta llegar al día viernes.

En este trabajo se implementa un generador de instancias probabilístico cuyo objetivo es producir casos de prueba que pongan a prueba el enfoque propuesto desde el punto de vista de la capacidad del horario y la distribución de la demanda estudiantil. La capacidad del horario queda determinada por la cantidad de salones y de bloques horarios disponibles, es decir, por el total de combinaciones sala-bloque en las que se puede programar una actividad académica. La distribución de la demanda modela cómo se reparten entre los estudiantes las preferencias de cursos y de bloques libres. De esta forma se enfrenta al algoritmo a estructuras distintas tanto en términos de ocupación del horario como de concentración de la demanda.

El generador define la *capacidad del horario* como el total de combinaciones sala–bloque disponibles, $|R| \cdot |T|$, y sobre ella se controla la *ocupación*, esto es, la fracción de esas celdas que requieren las actividades académicas. Cada curso se asocia a exactamente dos activida-

des y, en las instancias pequeñas, todos los salones son adecuados para toda actividad. La demanda estudiantil se genera haciendo que cada estudiante preinscriba entre cinco y siete cursos, elegidos según una de dos distribuciones. En la distribución uniforme cada curso tiene la misma probabilidad de ser elegido, $\frac{1}{|C|}$, y lo mismo ocurre con los bloques horarios. En la distribución multinomial se modela una presión mayoritaria hacia ciertos cursos: se lanza un dado de $|C|$ caras $|S|$ veces, obteniendo una demanda bruta por curso que luego se normaliza a una probabilidad en $[0, 1]$ mediante la frecuencia relativa de la demanda de cada curso:

$$\text{freq}(c_i) = \frac{d_i}{\sum_{j=1}^{|C|} d_j} \tag{24}$$

Donde $\text{freq}(c_i)$ nos dice la probabilidad de elegir el curso i, d_i es la demanda del curso i y d es el vector de todas las demandas, por lo que su suma da $|S|$. A modo de ejemplo, un posible resultado del lanzamiento de dados puede ser el que se ve en la figura 8:

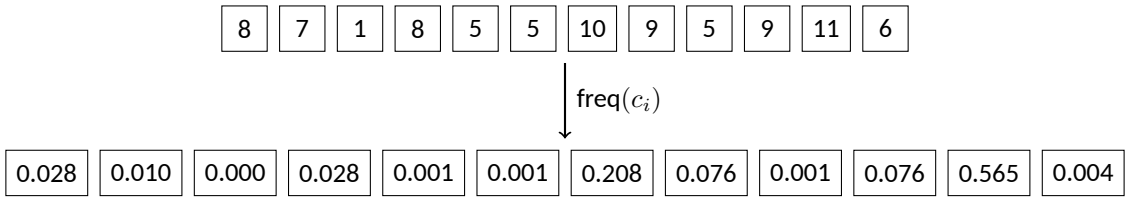


Figura 8: Posible resultado de distribución de demanda y su transformación con la función (24) con cuatro cifras significativas. Fuente: Elaboración propia.

La transformada hace notar la diferencia proporcional de los cursos con respecto a su demanda, haciendo que el curso tres tenga una muy baja probabilidad de ser elegido en comparación con el curso once (el cual tiene una demanda de once estudiantes).

Con la distribución multinomial aparece además la posibilidad de que algún curso no reciba demanda; en tal caso el curso se descarta, de modo que no se generan calendarizaciones para cursos sin estudiantes.

Sobre esta base se varían tres aspectos para construir el conjunto de casos de prueba. El primero es el tamaño y la ocupación de la infraestructura: las instancias pequeñas usan cinco salones y diez bloques horarios (capacidad $5 \cdot 10 = 50$), con ocupaciones del 50 % (doce cursos, veinticuatro actividades y ochenta y cuatro estudiantes) y del 80 % (veinte cursos, cuarenta actividades y ciento cuarenta estudiantes); las instancias grandes usan cincuenta salones y cincuenta bloques horarios (capacidad $50 \cdot 50 = 2500$) con una ocupación del 50 % (seiscientos veinticinco cursos, mil doscientas cincuenta actividades y mil quinientos estudiantes), y en ellas cada actividad admite entre uno y seis salones adecuados elegidos al azar. El segundo aspecto es la distribución de la demanda de cursos (uniforme o multinomial) ya descrita. El tercero es el tipo de petición de bloques libres: una variante básica, en la que

cada estudiante solicita entre dos y cuatro bloques libres, y una variante escalada, en la que la cantidad de bloques solicitados crece con la cantidad de cursos del estudiante (hasta $2|C_s|$ bloques, acotado por $|T|$).

Por simplicidad, el cupo de cada curso se fija igual a su demanda efectiva y la capacidad de cada salón se deja lo suficientemente holgada como para no restringir las asignaciones. Para cada estudiante, la cantidad mínima de cursos a inscribir se fija en $k^s_{\text{mín}} = \lfloor 0{,}5\,|C_s| \rfloor$ y la máxima en $k^s_{\text{máx}} = |C_s|$. Todos estos parámetros se escriben en el archivo de la instancia correspondiente.

4.1.1. Casos de prueba

Del generador probabilístico se obtienen diez casos de prueba, etiquetados de I1 a I10, cuyos tamaños de conjunto y características se resumen en la tabla 1. Los primeros ocho corresponden a la infraestructura pequeña (cinco salones, diez bloques horarios) y combinan las dos ocupaciones (50 % y 80 %), las dos distribuciones de demanda de cursos (uniforme y multinomial) y los dos tipos de petición de bloques libres (básica y escalada). Las dos últimas (I9 e I10) corresponden a la infraestructura grande con ocupación del 50 % y petición escalada, variando únicamente la distribución de la demanda de cursos. En el caso multinomial de la instancia grande (I10) el descarte de cursos sin demanda reduce levemente $|C|$ (y, por lo tanto, $|A|$) respecto del valor nominal.

Caso	$\|S\|$	$\|C\|$	$\|A\|$	$\|T\|$	$\|R\|$	Demanda cursos	Pet. libre	Ocup.
I1	84	12	24	10	5	Uniforme	Básica	50 %
I2	84	12	24	10	5	Uniforme	Escalada	50 %
I3	84	12	24	10	5	Multinomial	Básica	50 %
I4	84	12	24	10	5	Multinomial	Escalada	50 %
I5	140	20	40	10	5	Uniforme	Básica	80 %
I6	140	20	40	10	5	Uniforme	Escalada	80 %
I7	140	20	40	10	5	Multinomial	Básica	80 %
I8	140	20	40	10	5	Multinomial	Escalada	80 %
I9	1500	625	1250	50	50	Uniforme	Escalada	50 %
I10	1500	568	1136	50	50	Multinomial	Escalada	50 %

Tabla 1: Características de los diez casos de prueba generados con el algoritmo probabilístico.

Debido a que las preferencias de los estudiantes son distintas para cada caso de prueba, tener una idea visual de ellas es más adecuado. En la figura 9 se ponen uno al lado del otro los histogramas con la frecuencia relativa de estudiantes que asume cada curso. En el eje de las ordenadas se usa la frecuencia relativa de estudiantes que se preinscriben en cada curso, para poder comparar instancias de distinto tamaño. En los casos uniformes (I1, I2, I5, I6, I9)

la demanda queda aproximadamente plana, mientras que en los multinomiales (I3, I4, I7, I8, I10) la normalización por frecuencia relativa produce el efecto esperado de concentrar la demanda en unos pocos cursos.

En la figura 10 se disponen los histogramas de la proporción de la cantidad de peticiones de tiempo libre de los estudiantes, relativos a la cantidad de estudiantes por cada instancia para comparar las diferencias en la distribución. En los casos con petición *básica* (I1, I3, I5, I7) los estudiantes solicitan a lo más cuatro bloques, mientras que en los de petición *escalada* (I2, I4, I6, I8, I9, I10) el rango se amplía, llegando a más de diez bloques en las instancias grandes.

4.1.2. Obtención de frentes usando AMPL

Se usa `AMPL`[3], un lenguaje de modelado que permite implementar rápidamente el modelo matemático en una interfaz de programación y resolverlo con el solver `Gurobi`[4]. Se usa el método de ponderación (ver sección 1.1.4) para resolver el problema, normalizando con respecto a los peores y mejores valores para cada objetivo (MV_i y PV_i respectivamente, con $i = \{1, 2\}$) y con ponderaciones (o pesos) equiespaciados y normalizados (es decir, su suma da uno). El problema se reformula de la siguiente manera:

$$w_1 \cdot \frac{MV_1 - z_1}{MV_1 - PV_1} + w_2 \cdot \frac{MV_2 - z_2}{MV_2 - PV_2} \tag{25}$$

Con $w_1 + w_2 = 1$, MV_i y PV_i son el mejor y peor valor obtenido para el objetivo i. Se hacen un total de once ejecuciones cambiando los valores de w_1 y w_2 en cada iteración y para obtener los valores de MV_i y PV_i se ejecuta el modelo previamente para cada uno de los objetivos por separado.

[3]Página de AMPL: https://ampl.com/, último acceso: 14 de marzo, 2026
[4]Página de Gurobi: https://www.gurobi.com/, último acceso: 14 de marzo, 2026

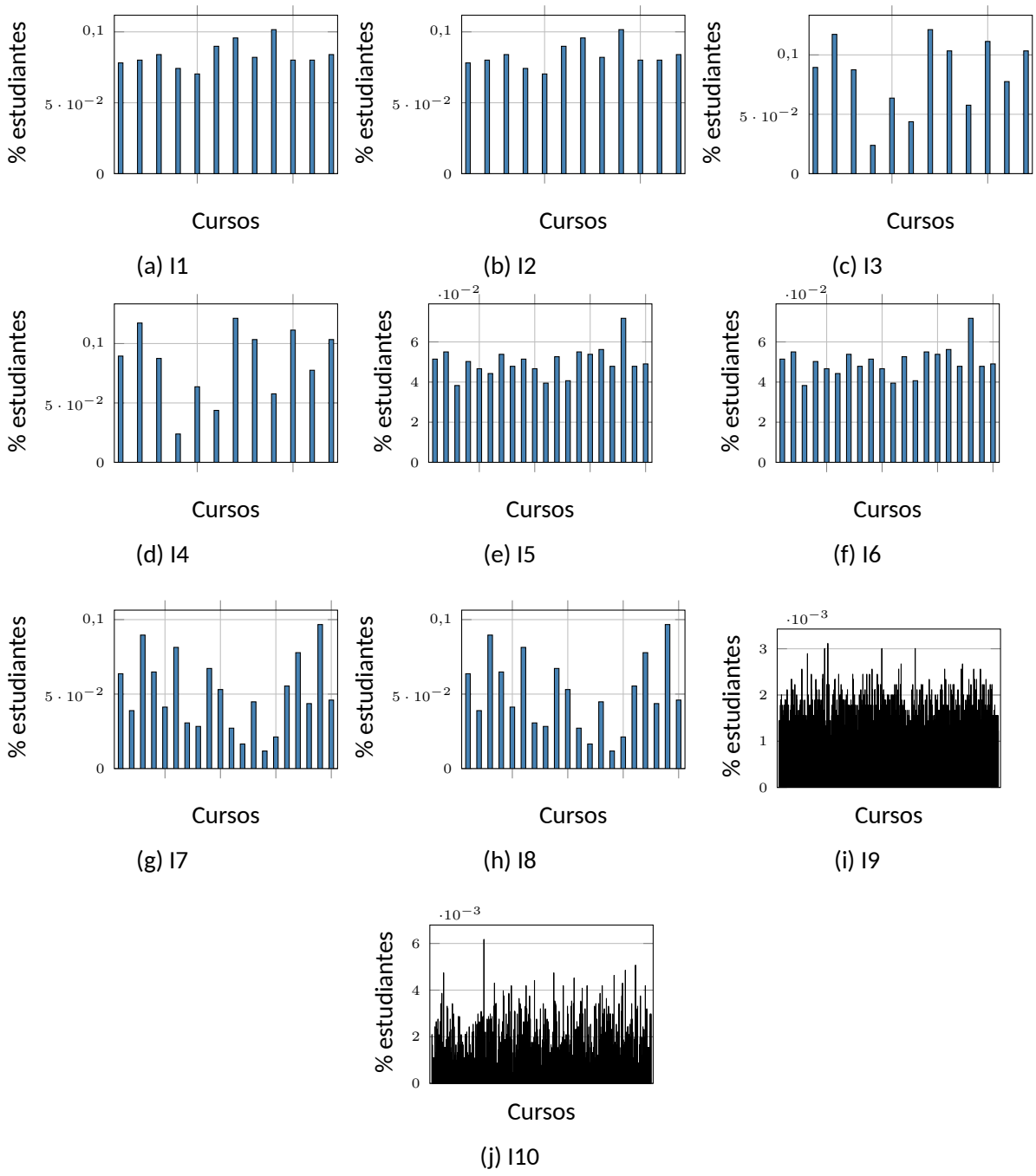


Figura 9: Distribución de la demanda estudiantil en los cursos de cada caso de prueba. Fuente: Elaboración propia.

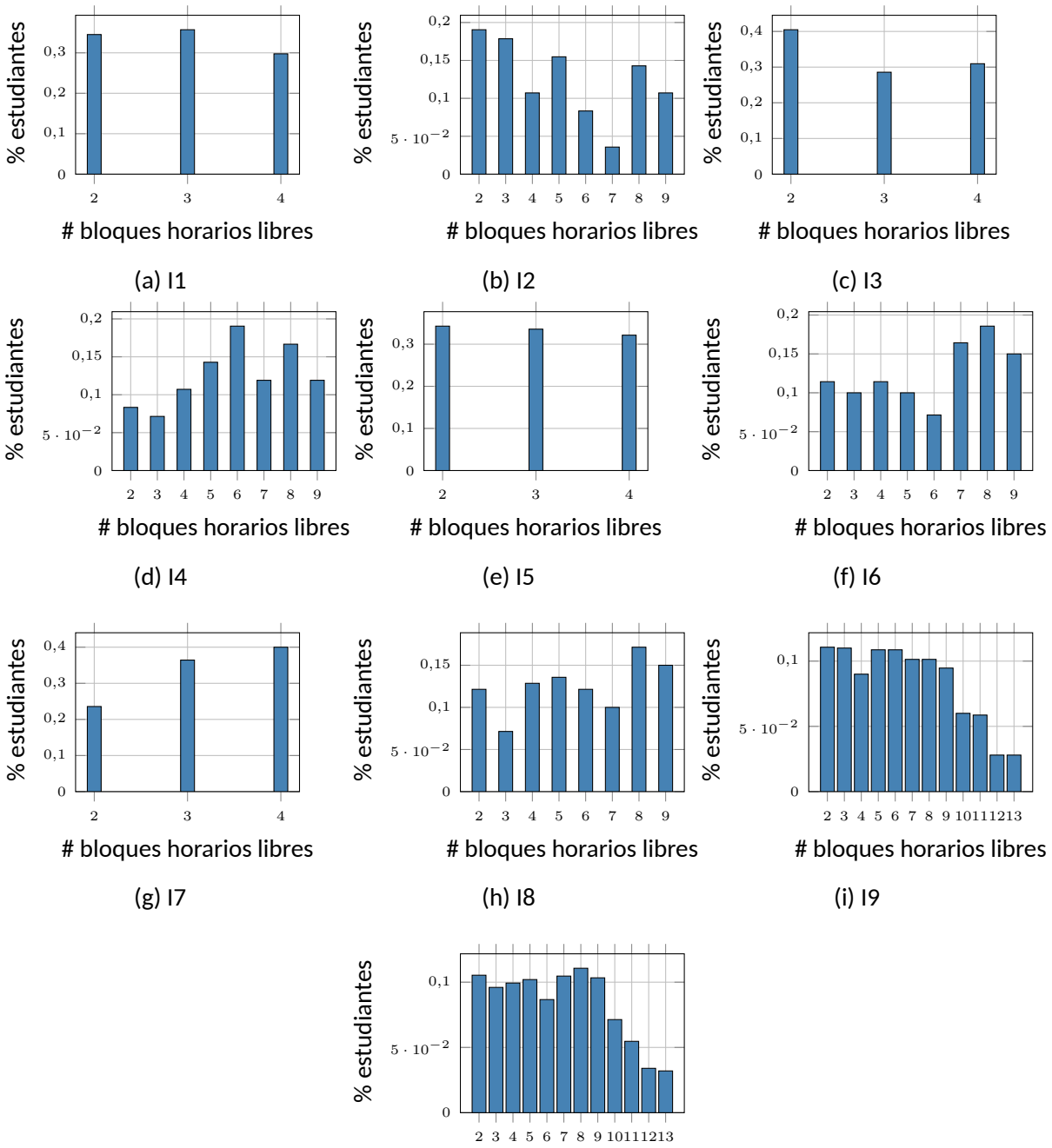


Figura 10: Distribución de la cantidad de peticiones de bloque libre por estudiante para cada instancia. Fuente: Elaboración propia.

4.1.3. Parámetros del algoritmo

Para todas las instancias se utilizó la misma configuración de ejecución de NSGA-II, resumida en la tabla 2. Cada instancia se resolvió con diez semillas distintas.

Parámetro	Valor
p	500
$MaxGen$	$10\,000$
p_c	$0{,}9$
p_m	$0{,}1$
p_m^{ts}	$0{,}6$
p_m^{sc}	$0{,}6$

Tabla 2: Parámetros de ejecución de NSGA-II.

4.2. Resultados

Tanto el algoritmo como el modelo exacto se ejecutaron en un mismo servidor, equipado con dos procesadores Intel Xeon Gold 5416S a 2.0 GHz y 31 GiB de memoria RAM. Para cada instancia, NSGA-II se ejecuta con diez semillas distintas, de modo que cada caso de prueba cuenta con diez frentes independientes. A partir de ellos se construye el frente agregado, que resulta de unir las soluciones de las diez ejecuciones y filtrar el conjunto no dominado. La calidad de los frentes se mide con el hipervolumen, usando como punto de referencia el punto de nadir entregado por AMPL para cada instancia, de modo que ambos algoritmos se comparan sobre la misma región de referencia. Para NSGA-II se reporta tanto el hipervolumen del frente agregado como el promedio (media $\pm$ desviación estándar) sobre las diez semillas; para AMPL, al tratarse de una solución única por instancia, no se aplican medidas de dispersión.

Como se puede ver en la figura 11, las soluciones de AMPL dominan por completo a las de NSGA-II, por lo que es esperable que el hipervolumen del solver sea mayor que el del algoritmo propuesto. Las métricas calculadas para cada algoritmo se muestran en la tabla 3, los tiempos de ejecución en la tabla 4 y el contraste estadístico sobre el hipervolumen en la tabla 5.

Para determinar si la diferencia de hipervolumen entre ambos algoritmos es estadísticamente significativa se aplica el test de Wilcoxon de rangos con signo, pareado entre instancias: para cada caso de prueba se compara el hipervolumen del *frente agregado* de NSGA-II con el de AMPL, y el test se evalúa sobre las diferencias entre los diez pares resultantes. El resultado se resume en la tabla 5.

Con un p-value de $0{,}00195$, se rechaza la hipótesis nula de igualdad: AMPL supera a NSGA-II en hipervolumen de forma estadísticamente significativa en el conjunto de instancias evaluadas.

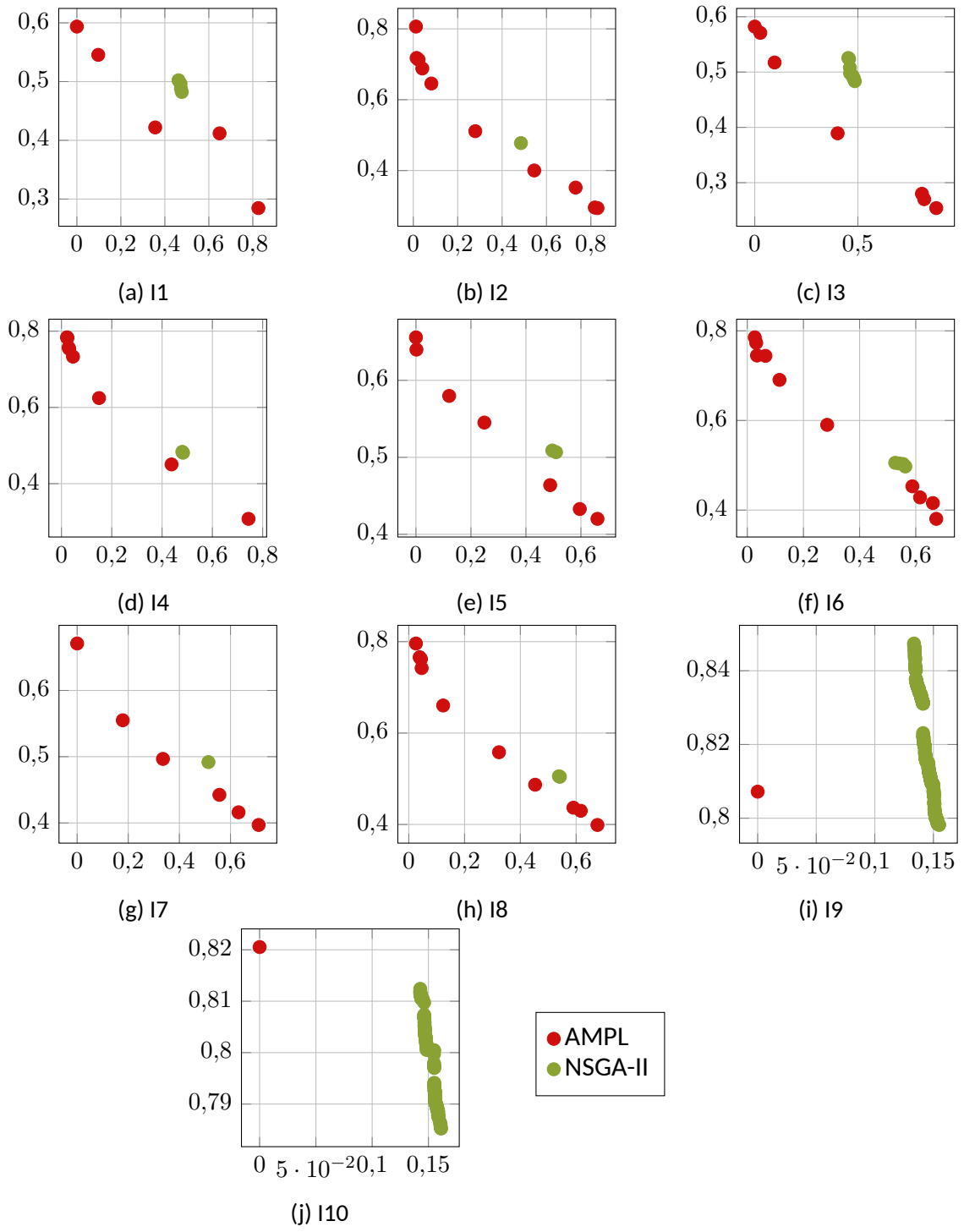


Figura 11: Comparación de frentes de Pareto para las diez instancias. El eje y muestra la insatisfacción en inscripción de cursos mientras que el eje x muestra insatisfacción de bloques libres.

Caso	HV AMPL	HV NSGA-II (agregado)	HV NSGA-II ($\mu \pm \sigma$)
I1	2.1801	1.4591	1.4269 ± 0.0192
I2	2.0370	1.3600	1.3268 ± 0.0252
I3	2.3594	1.5813	1.5397 ± 0.0189
I4	2.1598	1.5078	1.4558 ± 0.0263
I5	1.7334	1.1179	1.0982 ± 0.0102
I6	1.7286	1.1289	1.0943 ± 0.0235
I7	1.7949	1.1516	1.1148 ± 0.0252
I8	1.7567	1.1300	1.1078 ± 0.0163
I9	1.2934	1.1917	1.1745 ± 0.0068
I10	1.2779	1.2059	1.1896 ± 0.0086

Tabla 3: Hipervolumen por instancia. Para NSGA-II se reporta el del frente agregado (diez semillas) y el promedio desagregado sobre las semillas. El punto de referencia es el nadir de AMPL para cada instancia.

4.3. Discusión

En esta sección se interpretan los resultados a la luz de las características de cada caso de prueba. A continuación se desarrollan los ejes de análisis, partiendo de las observaciones cuantitativas de las tablas 3 y 4.

En las instancias pequeñas y medianas (I1–I8), NSGA-II alcanza aproximadamente dos tercios del hipervolumen de AMPL (entre un 64 % y un 67 %; por ejemplo, en I3 obtiene $1{,}58$ frente a $2{,}36$). El test de Wilcoxon confirma que la diferencia es estadísticamente significativa ($p = 0{,}00195$). Al pasar de un 50 % (I1–I4) a un 80 % (I5–I8) de ocupación, el hipervolumen de ambos métodos disminuye (AMPL de $\sim 2{,}2$ a $\sim 1{,}75$; NSGA-II de $\sim 1{,}45$ a $\sim 1{,}12$). Al comparar los casos uniformes (I1, I2, I5, I6) con los multinomiales (I3, I4, I7, I8) se observa que los multinomiales tienden a alcanzar un hipervolumen algo mayor (p. ej. I3 frente a I1). En las instancias grandes (I9, I10) NSGA-II se mantiene sorprendentemente competitivo: alcanza un 92 % y un 94 % del hipervolumen de AMPL ($1{,}19$ frente a $1{,}29$ en I9, y $1{,}21$ frente a $1{,}28$ en I10), una brecha relativa bastante menor que el ~ 65 % observado en las instancias pequeñas y medianas. Esto se explica, en parte, porque en estos tamaños el método exacto entrega muy pocas soluciones (un único punto en el frente), mientras que NSGA-II conserva un frente con mayor dispersión. NSGA-II es órdenes de magnitud más rápido que AMPL: en las instancias pequeñas y medianas emplea ~ 230–420 segundos frente a las ~ 10 horas del solver, y en las grandes $\sim 37{,}000$–$39{,}000$ segundos frente a las ~ 65–82 horas de AMPL.

En síntesis, los resultados revelan un compromiso claro entre calidad y costo computacional: AMPL alcanza un hipervolumen superior de forma estadísticamente significativa, pero NSGA-II lo consigue en un tiempo órdenes de magnitud menor y se vuelve notablemente más competitivo a medida que crece el tamaño de las instancias, justo donde el método exacto degrada la diversidad de su frente. En el siguiente capítulo se recogen estas observaciones

Caso	Tiempo AMPL [s]	Tiempo NSGA-II [s] ($\mu \pm \sigma$)
I1	36093	351.56 ± 78.35
I2	36115	312.64 ± 59.94
I3	36136	250.91 ± 40.70
I4	39615	229.47 ± 23.63
I5	36023	395.49 ± 36.71
I6	39618	411.16 ± 38.30
I7	36022	416.05 ± 52.87
I8	39619	417.70 ± 56.39
I9	293377	39312.55 ± 850.41
I10	234748	37552.22 ± 797.60

Tabla 4: Tiempos de ejecución por instancia. Para AMPL se reporta el tiempo total y para NSGA-II el promedio sobre las diez semillas.

Comparación	Estadístico W	p-value
HV agregado (NSGA-II vs AMPL)	0	$0{,}00195$

Tabla 5: Test de Wilcoxon de rangos con signo sobre el hipervolumen del frente agregado, pareado entre las diez instancias.

para discutir el cumplimiento de los objetivos planteados, las contribuciones y limitaciones del trabajo, y las líneas de investigación futuras.

CAPÍTULO 5
CONCLUSIONES

Este trabajo abordó una variante multi-objetivo del *Post Enrolment Course Timetabling Problem* (PECTP) que, además de la cobertura de los cursos preinscritos, incorpora la satisfacción de las preferencias de tiempo libre de los estudiantes como un objetivo de primer orden. Para resolverla se diseñó un algoritmo NSGA-II especializado y se contrastó su desempeño con el de un método exacto que resuelve el mismo modelo mediante AMPL y el solver Gurobi.

La formulación del problema considera la minimización de la insatisfacción media de los estudiantes con respecto a los cursos que el método logró inscribir y bloques de tiempo libre respetados, considerando restricciones de tiempo que no permiten a los estudiantes asistir a clases que suceden al mismo tiempo y además verifican que la asignación de recursos sea adecuada.

Se generaron diez instancias que simulan casos de la UTFSM, variando tamaño, ocupación y distribución de la demanda. Estas se abordaron mediante el algoritmo NSGA-II especializado para la formulación del problema y se realizó una evaluación experimental con un frente de referencia obtenido mediante AMPL.

5.1. Conclusiones sobre los resultados

Los tres objetivos planteados se cumplieron: se formuló la variante multi-objetivo del PECTP, se diseñó un NSGA-II capaz de resolver las instancias simuladas y se compararon sus frentes con los del método exacto. El resultado central es un claro compromiso entre calidad de solución y costo computacional. AMPL alcanza un hipervolumen superior de forma estadísticamente significativa (Wilcoxon, $p = 0{,}00195$), pero NSGA-II obtiene sus soluciones en un tiempo órdenes de magnitud menor. La brecha de calidad no se agranda al escalar, sino que se reduce: de alcanzar cerca de dos tercios del hipervolumen de AMPL en las instancias pequeñas y medianas (I1–I8, ~ 64–$67\,\%$), NSGA-II llega a un 92–$94\,\%$ en las grandes (I9, I10). Estos resultados pueden ser indicio de que en contextos más realistas, NSGA-II es más competitivo pues otorga una mayor variabilidad de soluciones que están más dispersas. Al aumentar la ocupación de un 50 % a un 80 %, el hipervolumen se reduce con ambos métodos y en general las demandas que siguen una distribución multinomial tienden a valores mayores que las uniformes.

En términos prácticos, el algoritmo propuesto es una alternativa factible en relación tamaño/cómputo, pues el método completo deja de ser viable puesto que la diversidad de las soluciones es mucho menor.

Las limitaciones principales de la metodología son que los parámetros usados no fueron ele-

gidos mediante algún afinamiento final con herramientas como ParamILS, lo que podría ayudar a estrechar la brecha en instancias pequeñas y medianas. Además de esto, la validación se realizó sobre instancias simuladas que podrían no ser tan representativas de casos reales, por lo que las conclusiones respecto a la propuesta se circunscriben a las condiciones impuestas por el generador.

5.2. Trabajo futuro

El trabajo pendiente consiste en realizar el afinamiento de parámetros para volver a evaluar la brecha de calidad de manera que sea justa para NSGA-II. Además de esto poder validar con datos reales de la UTFSM o con instancias benchmark de la ITC 2019. Además es interesante investigar por qué NSGA-II presenta una ventaja relativa a medida que crece el tamaño del problema.

Otras lineas de investigación futura son la integración de esta formulación dentro de los sistemas de calendarización de la universidad.

A partir de este trabajo se desprenden varias líneas de continuación.

En primer lugar, queda pendiente completar el afinamiento de parámetros del algoritmo para volver a evaluar la brecha de calidad en condiciones justas para NSGA-II, dado que los resultados reportados se obtuvieron con la configuración arbitraria y no con una configuración ajustada específicamente para el problema. Un ajuste fino de las probabilidades de los operadores y del tamaño de la población podría estrechar la diferencia observada frente al método exacto, sobre todo en las instancias pequeñas y medianas.

En segundo lugar, resulta necesario validar la propuesta sobre datos reales de la Universidad Técnica Federico Santa María –mallas curriculares, preinscripciones y preferencias efectivas de los estudiantes– o, en su defecto, sobre instancias de referencia reconocidas como las de la ITC 2019, de modo de contrastar el comportamiento del algoritmo más allá de las condiciones reproducidas por el generador de instancias.

En tercer lugar, es interesante investigar con mayor profundidad por qué la ventaja relativa de NSGA-II aumenta a medida que crece el tamaño del problema. En particular, conviene caracterizar a partir de qué punto el método exacto deja de entregar frentes con dispersión y determinar si ese comportamiento responde a la dificultad intrínseca del modelo o a los límites de tiempo impuestos al solver.

En el plano algorítmico, también es posible explorar otras metaheurísticas multi-objetivo –como MOEA/D– y complementar el hipervolumen con indicadores de calidad adicionales, así como aprovechar la naturaleza paralelizable de los algoritmos evolutivos para reducir aún más los tiempos de ejecución.

Finalmente, desde una perspectiva aplicada, una línea de mayor alcance es la integración

de esta formulación dentro de los sistemas de calendarización de la universidad, de manera que las preferencias de tiempo libre de los estudiantes se consideren de forma efectiva en la planificación académica. En esta dirección cobran relevancia los mecanismos de apoyo a la decisión, que permiten seleccionar automáticamente una solución representativa del frente de Pareto sin requerir la inspección manual de todas las alternativas.

REFERENCIAS BIBLIOGRÁFICAS

[Beume *et al.*, 2007] Beume, N., Naujoks, B., y Emmerich, M. (2007). Sms-emoa: Multiobjective selection based on dominated hypervolume. *European journal of operational research*, 181(3):1653–1669.

[Bron y Kerbosch, 1973] Bron, C. y Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577.

[Chakraborty y Kar, 2017] Chakraborty, A. y Kar, A. K. (2017). *Swarm Intelligence: A Review of Algorithms*, pp. 475–494. Springer International Publishing, Cham.

[Cooper y Kingston, 1996] Cooper, T. B. y Kingston, J. H. (1996). The complexity of timetable construction problems. En Burke, E. y Ross, P., editores, *Practice and Theory of Automated Timetabling*, pp. 281–295, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Cormen *et al.*, 2022] Cormen, T., Leiserson, C., Rivest, R., y Stein, C. (2022). *Introduction to Algorithms, fourth edition*, pp. 1081–1084. MIT Press.

[Czyżżak y Jaszkiewicz, 1998] Czyżżak, P. y Jaszkiewicz, A. (1998). Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47.

[Davison *et al.*, 2024] Davison, M., Kheiri, A., y Zografos, K. G. (2024). Modelling and solving the university course timetabling problem with hybrid teaching considerations. *Journal of Scheduling*.

[Deb, 2008] Deb, K. (2008). *Introduction to Evolutionary Multiobjective Optimization*, pp. 59–96. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Deb y Jain, 2014] Deb, K. y Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18:577–601.

[Deb *et al.*, 2002] Deb, K., Pratap, A., Agarwal, S., y Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Dunke y Nickel, 2023] Dunke, F. y Nickel, S. (2023). A matheuristic for customized multilevel multi-criteria university timetabling. *Annals of Operations Research*, 328(2):1313–1348.

[Fonseca *et al.*, 1993] Fonseca, Carlos M and Fleming, Peter J and others (1993). Genetic algorithms for multiobjective optimization: formulationdiscussion and generalization. En *Icga*, volumen 93, pp. 416–423.

[Goh *et al.*, 2017] Goh, S. L., Kendall, G., y Sabar, N. R. (2017). Improved local search approaches to solve the post enrolment course timetabling problem. *European Journal of Operational Research*, 261(1):17–29.

[Gülcü y Akkan, 2020] Gülcü, A. y Akkan, C. (2020). Robust university course timetabling problem subject to single and multiple disruptions. *European Journal of Operational Research*, 283(2):630–646.

[Hafsa *et al.*, 2021] Hafsa, M., Wattebled, P., Jacques, J., y Jourdan, L. (2021). A multi-objective evolutionary approach to professional course timetabling: A real-world case study. En *2021 IEEE Congress on Evolutionary Computation (CEC)*, pp. 997–1004.

[Hao *et al.*, 2024] Hao, Y., Zhao, C., Zhang, Y., Cao, Y., y Li, Z. (2024). Constrained multi-objective optimization problems: Methodologies, algorithms and applications. *Knowledge-Based Systems*, 299:111998.

[Holm *et al.*, 2022] Holm, D. S., Mikkelsen, R. Ø., Sørensen, M., y Stidsen, T. J. R. (2022). A graph-based mip formulation of the international timetabling competition 2019. *Journal of Scheduling*, 25(4):405–428.

[Jain *et al.*, 2025] Jain, M., Raikoti, S., Bala, S. J., Jagadish Kumar, V., Kamal Sulaiman, S., y Vijayalakshmi, B. (2025). Conflict-free school timetabling using graph coloring and hybrid metaheuristic algorithms. En *2025 International Conference on Information, Implementation, and Innovation in Technology (I2ITCON)*, pp. 1–5.

[Knowles y Corne, 2000] Knowles, J. D. y Corne, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2):149–172.

[Laszczyk y Myszkowski, 2019] Laszczyk, M. y Myszkowski, P. B. (2019). Survey of quality measures for multi-objective optimization: Construction of complementary set of multi-objective quality measures. *Swarm and Evolutionary Computation*, 48:109–133.

[Laumanns *et al.*, 2005] Laumanns, M., Thiele, L., y Zitzler, E. (2005). An adaptive scheme to generate the pareto front based on the epsilon-constraint method. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[Leite *et al.*, 2015] Leite, N., Neves, R., Horta, N., Melício, F., y Rosa, A. C. (2015). Solving a capacitated exam timetabling problem instance using a bi-objective nsga-ii. *Computational Intelligence*, pp. 115–129.

[Lewis y Paechter, 2007] Lewis, R. y Paechter, B. (2007). Finding feasible timetables using group-based operators. *IEEE Transactions on Evolutionary Computation*, 11(3):397–413.

[Loveland y Bland, 2013] Loveland, K. A. y Bland, E. (2013). Impact of course scheduling formats on student learning and satisfaction. *Atlantic Economic Journal*, 41(2):191–192.

[Luna, 2025] Luna, J. (2025). Student-centric scheduling: Integrating student pre ferences into university course planning. Master's thesis, California Polytechnic State University, San Luis O bispo.

[Marler y Arora, 2010] Marler, R. T. y Arora, J. S. (2010). The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41(6):853–862.

[McCollum *et al.*, 2010] McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., y Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *Informs Journal on computing*, 22(1):120–130.

[Miettinen, 2008] Miettinen, K. (2008). *Introduction to Multiobjective Optimization: Noninteractive Approaches*, pp. 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Mishra y Singh, 2016] Mishra, V. y Singh, V. (2016). Vector evaluated genetic algorithm-based distributed query plan generation in distributed database. En Afzalpulkar, N., Srivastava, V., Singh, G., y Bhatnagar, D., editores, *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing*, pp. 325–337, New Delhi. Springer India.

[Müller y Murray, 2010] Müller, T. y Murray, K. (2010). Comprehensive approach to student sectioning. *Annals of Operations Research*, 181(1):249–269.

[Müller *et al.*, 2024] Müller, T., Rudová, H., y Müllerová, Z. (2024). Real-world university course timetabling at the international timetabling competition 2019. *Journal of Scheduling*.

[Okabe *et al.*, 2003] Okabe, T., Jin, Y., y Sendhoff, B. (2003). A critical survey of performance indices for multi-objective optimisation. En *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volumen 2, pp. 878–885 Vol.2.

[Pardalos y Xue, 1994] Pardalos, P. M. y Xue, J. (1994). The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328.

[Post *et al.*, 2016] Post, G., Di Gaspero, L., Kingston, J. H., McCollum, B., y Schaerf, A. (2016). The third international timetabling competition. *Annals of Operations Research*, 239(1):69–75.

[Rappos *et al.*, 2022] Rappos, E., Thiémard, E., Robert, S., y Hêche, J.-F. (2022). A mixed-integer programming approach for solving university course timetabling problems. *Journal of Scheduling*, 25(4):391–404.

[Riquelme *et al.*, 2015] Riquelme, N., Von Lücken, C., y Baran, B. (2015). Performance metrics in multi-objective optimization. En *2015 Latin American Computing Conference (CLEI)*, pp. 1–11.

[Schaerf, 1999] Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127.

[Schott, 1995] Schott, J. R. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Tesis doctoral, Massachusetts Institute of Technology.

[Schütze y Hernández, 2021] Schütze, O. y Hernández, C. (2021). *Archiving in Evolutionary Multi-objective Optimization: A Short Overview*, pp. 17–20. Springer International Publishing, Cham.

[Sharma y Kumar, 2022] Sharma, S. y Kumar, V. (2022). A comprehensive review on multi-objective optimization techniques: Past, present and future. *Archives of Computational Methods in Engineering*, 29(7):5605–5633.

[Socha *et al.*, 2002] Socha, K., Knowles, J., y Sampels, M. (2002). A max-min ant system for the university course timetabling problem. En Dorigo, M., Di Caro, G., y Sampels, M., editores, *Ant Algorithms*, pp. 1–13, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Song *et al.*, 2018] Song, T., Liu, S., Tang, X., Peng, X., y Chen, M. (2018). An iterated local search algorithm for the university course timetabling problem. *Applied Soft Computing*, 68:597–608.

[Srinivas y Deb, 1994] Srinivas, N. y Deb, K. (1994). Muiltiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

[Stewart *et al.*, 2021] Stewart, R. H., Palmer, T. S., y DuPont, B. (2021). A survey of multi-objective optimization methods and their applications for nuclear scientists and engineers. *Progress in Nuclear Energy*, 138:103830.

[Sylejmani *et al.*, 2023] Sylejmani, K., Gashi, E., y Ymeri, A. (2023). Simulated annealing with penalization for university course timetabling. *Journal of Scheduling*, 26(5):497–517.

[Thepphakorn y Pongcharoen, 2023] Thepphakorn, T. y Pongcharoen, P. (2023). Modified and hybridised bi-objective firefly algorithms for university course scheduling. *Soft Computing*, 27(14):9735–9772.

[Thepphakorn *et al.*, 2016] Thepphakorn, T., Pongcharoen, P., y Vitayasak, S. (2016). A new multiple objective cuckoo search for university course timetabling problem. *Multi-disciplinary Trends in Artificial Intelligence*, pp. 196–207.

[Tung Ngo *et al.*, 2021] Tung Ngo, S., Jafreezal, J., Hoang Nguyen, G., y Ngoc Bui, A. (2021). A genetic algorithm for multi-objective optimization in complex course timetabling. En *Proceedings of the 2021 10th International Conference on Software and Computer Applications*, ICSCA '21, p. 229–237, New York, NY, USA. Association for Computing Machinery.

[Van Veldhuizen y Lamont, 2000] Van Veldhuizen, D. y Lamont, G. (2000). On measuring multiobjective evolutionary algorithm performance. En *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volumen 1, pp. 204–211 vol.1.

[Van Veldhuizen *et al.*, 1998] Van Veldhuizen, David A and Lamont, Gary B and others (1998). Evolutionary computation and convergence to a pareto front. En *Late breaking papers at the genetic programming 1998 conference*, pp. 221–228.

[Wong *et al.*, 2022] Wong, C. H., Goh, S. L., y Likoh, J. (2022). A genetic algorithm for the real-world university course timetabling problem. En *2022 IEEE 18th International Colloquium on Signal Processing Applications (CSPA)*, pp. 46–50.

[Wong y Chapman, 2023] Wong, W. H. y Chapman, E. (2023). Student satisfaction and interaction in higher education. *Higher Education*, 85(5):957–978.

[Wu y Azarm, 2001] Wu, J. y Azarm, S. (2001). Metrics for quality assessment of a multiobjective design optimization solution set. *J. Mech. Des.*, 123(1):18–25.

[Zhang y Li, 2007] Zhang, Q. y Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731.

[Zhu *et al.*, 2026] Zhu, Z., Zhou, Y., Zhou, G., Luo, Q., y Huang, H. (2026). Q-learning-based multi-objective hyper-heuristic algorithm for multi-campus university course scheduling. *Swarm and Evolutionary Computation*, 102:102338.

[Zitzler, 1999] Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volumen 63. Shaker Ithaca.

[Zitzler *et al.*, 2001] Zitzler, E., Laumanns, M., y Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. *TIK report*, 103.

[Zitzler y Thiele, 1999] Zitzler, E. y Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.