

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“DESARROLLO DE SOFTWARE BASADO EN
METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL
PROBLEMA DE BALANCEO DE MALLAS CURRICULARES
DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL
ACADÉMICO PARA ALUMNOS DE INFORMÁTICA DE LA
UTFSM CC”

DIEGO AGUILERA MANCILLA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: María Cristina Riff
Profesor Correferente: Pedro Godoy

Diciembre - 2024

DEDICATORIA

A mis padres, Alberto y Jessica. Sin su apoyo durante este proceso nada de esto sería posible. Su dedicación y esfuerzo constante para asegurarme una educación son un regalo que valoro más allá de las palabras.

AGRADECIMIENTOS

Un sincero agradecimiento a mi profesora guía María Cristina Riff, por apoyarme durante este proceso y trabajo. Una excelente docente cuya tutoría me permitió desarrollarme y encontrar mi llamado en la inteligencia artificial.

A mi profesor correferente, Pedro Godoy. Muchísimas gracias por guiarme y darme su consejo durante este largo proceso, sin su ayuda jamás hubiera podido terminar este trabajo en el tiempo necesario.

A Erika Guzman, gracias por su disposición a ayudar y su comunicación con el estudiantado.

Y por supuesto, un enorme agradecimiento a todos mis amigos y compañeros, gracias por su compañía y apoyo en los momentos de estrés y alegría, a los que me acompañaron en la salas de clases, los que terminaron, los que se cambiaron, los que buscaron otros caminos a seguir. Siempre estaré eternamente agradecido de su compañía.

RESUMEN

Resumen— Todas las universidades durante el proceso de desarrollo de un plan de estudios deben enfrentarse a la resolución del problema de balanceo de malla curricular considerando una serie de factores que determinarán la formación del estudiante, la duración de la carrera y los conocimientos que se consideran necesarios para el campo laboral de la disciplina, sin embargo, esta malla en general no considera el historial académico ni preferencias que puedan tener los alumnos. Es por esto que se propone un software, el cual de manera dinámica a través del uso de metaheurísticas y la resolución del Problema de Balanceo de Mallas Curriculares pueda reestructurar una malla académica en base a múltiples factores, generando en lo posible una malla que satisfaga las necesidades y expectativas de los estudiantes de Ingeniería Civil Informática de la Universidad Técnica Federico Santa María Casa Central.

Palabras Clave— Metaheurísticas; *BACP*; *CSP*

ABSTRACT

Abstract— All universities, during the process of developing a curriculum, must face the challenge of balancing the course schedule, considering a series of factors that will determine the student's education, the duration of the program, and the knowledge deemed necessary for the relevant job market. However, this schedule does not consider the academic history or preferences that students may have. Therefore, a software is proposed which, dynamically through the use of metaheuristics and the solution of the Balanced Academic Curriculum Problem, can restructure a curriculum based on multiple factors, generating a solution schedule that satisfies the needs and expectations of Computer Science Engineering students at Universidad Técnica Federico Santa María, Casa Central.

Keywords— Metaheuristics; *BACP*; *CSP*

GLOSARIO

BACP: Balanced Academic Curriculum Problem, Problema de Balanceo de Malla Curricular.

DI: Departamento de Informática.

UTFSM: Universidad Técnica Federico Santa María.

CSP: Constraint satisfaction problem, Problema de Satisfacción de Restricciones.

EAC: Error Absoluto Cuadrático.

CP: Constraint Programming, Programación con Restricciones.

TS: Tabu Search, Búsqueda Tabú.

GA: Genetic Algorithm, Algoritmos genéticos.

RHC: Randomized Hill Climbing.

SA: Simulated Annealing.

FA: Fireflies Algorithm, Algoritmo basado en luciérnagas.

DTS: Dynamic Tabu Search, Búsqueda Tabú Dinámica.

SD: Steepest Descent, Descenso más profundo.

FD: First Descent. Primer descenso.

B&B: Branch & Bound.

SUS: System Usability Scale.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS	VIII
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Definición inicial del problema	3
1.2 Problemas detectados	5
1.3 Alcance	6
1.4 Motivación	7
1.5 Objetivos	7
1.5.1 Objetivo general	7
1.5.2 Objetivos específicos	7
CAPÍTULO 2: MARCO CONCEPTUAL	8
2.1 El Problema de Balanceo de Malla Curricular	8
2.2 Estado del arte	9
2.2.1 Solución vía Programación con Restricciones	9
2.2.2 Solución vía Ant Colony Optimization (ACO)	12
2.2.3 Solución vía Algoritmos Genéticos (GA)	13
2.2.4 Solución vía Técnicas de búsqueda local híbridas: Steepest Descent (SD), First Descent (FD), Radomized Hill Climbing (RHC), Simulated Annealing (SA), Tabu Search (TS) y Dynamic Tabu Search (DTS)	16
2.2.5 Solución vía Algoritmo de Luciérnagas (FA)	17
2.3 Resumen del capítulo	18
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	20
3.1 Formulación Matemática del problema	20
3.2 Metaheurística propuesta : <i>Tabu Search</i>	22
3.3 Representación de la Solución	23
3.3.1 Representación matricial de la solución	23
3.3.2 Representación vectorial de la solución	24
3.4 Descripción del algoritmo - Solución inicial	25
3.4.1 Ciclo principal - Elección de asignaturas y semestres	31
3.4.2 Verificación de Restricciones	33
3.4.3 Actualización de rangos	34

3.4.4	Diagrama de flujo - Solución Inicial	34
3.5	Soluciones simétricas y óptimos locales: Ejemplos	36
3.6	Ventajas de la Búsqueda Tabú	38
3.7	Elección de movimiento	39
3.8	Programa Propuesto	42
3.8.1	Malla interactiva	43
3.8.2	Desarrollo e Integración	46
3.8.3	Interfaz usuaria - Ejemplos	48
3.9	Propuesta de Solución: Conclusiones	53
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		54
4.1	Especificaciones del equipo de trabajo	54
4.2	Instancias de prueba - Benchmarks	54
4.2.1	8 semestres - 46 asignaturas	55
4.2.2	10 Semestres - 42 asignaturas	56
4.2.3	12 Semestres - 66 asignaturas	58
4.2.4	Resumen Comparativo - Instancias de Benchmark	59
4.3	Instancias de prueba reales - Malla Informática USM (2023)	60
4.3.1	11 semestres - 59 asignaturas	60
4.3.2	10 semestres - 59 asignaturas	61
4.3.3	7 semestres, 35 asignaturas	63
4.3.4	Resumen comparativo - Instancias reales	68
4.4	Validación de la Interfaz	68
4.4.1	System Usability Scale	68
4.4.2	Resultados	69
4.5	Validación de Solución: Conclusiones	70
CAPÍTULO 5: CONCLUSIONES		71
REFERENCIAS BIBLIOGRÁFICAS		73

ÍNDICE DE FIGURAS

1	Malla curricular Ing. Civil Informática UTFSM Casa Central (2023)	4
2	Diagrama de causas y efectos	6
3	Resultados obtenidos al variar la variable de ordenamiento en Oz	12
4	Convergencia de GA	15
5	Representación Matricial	23
6	Representación Vectorial	24
7	Rangos Matemáticos 1	25
8	Rangos Matemáticos 2	26
9	Rangos Matemáticos 3	26
10	Posibles asignaciones de todas las matemáticas	27
11	Diagrama de precedencia (2023).	28
12	Diagrama de flujo - Solución inicial	35
13	Solución en tres semestres con créditos	36
14	Solución en tres semestres con créditos	37
15	Ejemplo de lista tabu 1	38
16	Ejemplo de lista tabu 2	39
17	Ejemplo de movimiento 1	40
18	Ejemplo de movimiento 2	41
19	Malla Interactiva Original	44
20	Malla Interactiva: Selección de asignaturas aprobadas	45
21	Ejemplo de formato JSON para un semestre particular	47
22	Diagrama de flujo, Malla Interactiva - BACP	48
23	BACP - 59 asignaturas, 11 semestre, 30 créditos máx.	49
24	BACP - Selección dinámica de asignaturas aprobadas en base a la malla BACP	51
25	Malla solución resultante selección dinámica - 7 semestres, 35 créditos máx.	52
26	TS - Iteraciones vs EAC	56
27	TS - Iteraciones vs EAC	57
28	TS - Iteraciones vs EAC	59
29	TS - Iteraciones vs EAC	62
30	Malla Curricular - Instancia de prueba	64
31	TS - Iteraciones vs EAC	66
32	Semestre resultado - 7 semestres, 35 asignaturas	67
33	Grafico Resultados SUS	70

ÍNDICE DE TABLAS

1	Resumen comparativo de rendimiento programación con restricciones vs programación en enteros	11
2	Resumen comparativo de rendimiento - Instancias de referencia, ACO	13
3	Resultados utilizando GA + CP	15
4	Resultados para las instancias de CSPLib	17

5	Síntesis de resultados - Firefly Algorithm	18
6	Resultados Solución inicial (CP): 8 semestres, 46 asignaturas	
	Fuente: Elaboración Propia.	55
7	Resultados Solución Final (TS): 8 semestres, 46 asignaturas	
	Fuente: Elaboración Propia.	55
8	Resultados Solución inicial (CP): 10 semestres, 42 asignaturas	
	Fuente: Elaboración Propia.	56
9	Resultados Solución final (TS): 10 semestres, 42 asignaturas	
	Fuente: Elaboración Propia.	57
10	Resultados Solución inicial (CP): 12 semestres, 66 asignaturas	
	Fuente: Elaboración Propia.	58
11	Resultados Solución final (TS): 12 semestres, 66 asignaturas	
	Fuente: Elaboración Propia.	58
12	Resumen Comparativo - Instancias Benchmark	60
13	Resultados Solución inicial (CP): 11 semestres, 59 asignaturas	
	Fuente: Elaboración Propia.	60
14	Resultados Solución final (TS): 11 semestres, 59 asignaturas	
	Fuente: Elaboración Propia.	61
15	Resultados Solución inicial (CP): 10 semestres, 59 asignaturas	
	Fuente: Elaboración Propia.	61
16	Resultados Solución final (TS): 10 semestres, 59 asignaturas	
	Fuente: Elaboración Propia.	62
17	Resultados Solución inicial (CP): 7 semestres, 35 asignaturas	65
18	Resultados Solución final (TS): 7 semestres, 35 asignaturas	65
19	Resumen Comparativo - Instancias Reales	68
20	Tabla Resultados SUS	69

INTRODUCCIÓN

Contexto General En el contexto educativo universitario actual, con el considerable aumento de la cantidad de matrículas anuales durante las últimas décadas, son cada vez más evidentes los desafíos que los estudiantes deben enfrentar durante el transcurso de su carrera universitaria. Uno de los problemas más comunes para los regímenes semiflexibles y flexibles es la distribución desbalanceada de asignaturas semestrales, la que se puede originar debido a una mala estructura organizativa por parte de los departamentos académicos y docentes, o bien, por voluntad propia del estudiante al querer cumplir sus propios objetivos, metas o preferencias. Este desbalanceo puede suponer una sobrecarga académica en ciertos semestres, provocando comúnmente estrés y/o ansiedad, lo cual se puede traducir en un aumento en las probabilidades de reprobar una o más asignatura(s).

El Balanced Academic Curricular Problem (BACP) desde el punto de vista de la universidad surge como un problema combinatorio cuya resolución busca optimizar la distribución de asignaturas en períodos académicos, garantizando un equilibrio entre la carga académica y las restricciones curriculares, como los prerrequisitos y la cantidad máxima de créditos. Este problema, ampliamente estudiado a nivel internacional, es de particular relevancia en las universidades, donde las carreras suelen estar estructuradas de manera rígida y los estudiantes no poseen una herramienta de estructuración directa.

Resolver el BACP con un visión estudiantil no solo tiene implicaciones teóricas en el ámbito de la optimización combinatoria, sino que también ofrece beneficios prácticos tangibles para los alumnos. Una solución adecuada a la medida para regímenes semiflexibles permite diseñar mallas que distribuyan equitativamente la carga académica, reduciendo el riesgo de sobrecarga en semestres específicos y permitiendo a los estudiantes avanzar de manera más eficiente en sus estudios.

En este trabajo, se presenta el desarrollo de un software el cual utiliza técnicas avanzadas en la resolución del BACP estudiantil para generar planes académicos con una distribución equilibrada de asignaturas. Este enfoque busca otorgar a futuros estudiantes de ing. civil informática pertenecientes a la UTFSM una herramienta capaz de ayudarlos durante la estructuración de su malla académica, facilitando la toma de decisiones durante la inscripción de asignaturas.

Estructura del documento La estructura del documento se divide en cinco partes principales; La Definición del Problema en la cual se explicarán los principales problemas detectados, asociados a la falta de una herramienta que ayude al estudiante promedio a organizar su currículum académico, además del alcance, la motivación y los objetivos principales abordados en este documento. En segundo lugar, el Estado del Arte en el que se presentarán brevemente estudios previos del BACP con el fin de otorgar información relevante y comprender en profundidad los diferentes métodos propuestos como solución en las últimas décadas. En la

Propuesta de Solución se explica en detalle la construcción de la solución, desde la formulación matemática y la representación de la solución hasta la propuesta de software y su uso esperado. En la Validación de Solución se compara el rendimiento de la solución en base a las principales instancias de referencia además de algunas instancias reales obtenidas de la malla curricular de ing. civil informática UTFSM (2023). Finalmente, se plantean las Conclusiones del trabajo en base a los resultados obtenidos, el análisis de técnica utilizada, posibles mejoras y trabajo futuro.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

En el siguiente capítulo se definirá el problema a abordar y las dificultades que presenta junto con los problemas detectados. Se plantea además el alcance que tiene la elaboración de la propuesta, la motivación detrás del trabajo y los objetivos principales.

1.1. Definición inicial del problema

El proceso de enseñanza superior se basa en una malla académica que define los cursos necesarios para la obtención de un título universitario en un área específica. Esta malla se crea considerando las necesidades del campo laboral, la calidad de la formación y una organización de cursos, créditos y prerrequisitos en un número limitado de semestres.

Actualmente las instituciones universitarias (considerando los correspondientes departamentos) son las que definen la malla curricular para cada carrera. Esta planificación suele realizarse de manera manual o mediante sistemas, los cuales no siempre optimizan adecuadamente la distribución de las asignaturas debido a que en un entorno real, es necesario considerar otras problemáticas como; asignación de salas, disponibilidad de profesores, organización de horarios académicos, entre otros.

En el caso de los regímenes semiflexibles, los estudiantes pueden alterar el orden de la malla curricular establecida en función de sus objetivos o situaciones de avance en la carrera. Ellos no poseen una herramienta que les facilite la distribución equitativa de carga académica. Por lo general, los estudiantes recurren a estrategias o recomendaciones de pares con la finalidad de ordenar sus períodos académicos, lo cual tiende a generar problemas a largo plazo, dado que una mala modificación del orden al plan de estudios puede resultar en una extensión innecesaria de la duración de la carrera, afectando su experiencia universitaria de manera negativa. Es por esto que en esta memoria se desea proponer una herramienta de apoyo a la toma de decisiones del estudiante con indicadores cuantitativos.

DESARROLLO DE SOFTWARE BASADO EN METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE
BALANCEO DE MALLAS CURRICULARES DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL ACADÉMICO
PARA ALUMNOS DE INFORMÁTICA DE LA UTFSM CC

AÑO 1			AÑO 2			AÑO 3			AÑO 4			AÑO 5		AÑO 5 1/2
SEMESTRE I	SEMESTRE II	SEMESTRE III	SEMESTRE II	SEMESTRE III	SEMESTRE IV	SEMESTRE V	SEMESTRE VI	SEMESTRE VII	SEMESTRE VIII	SEMESTRE IX	SEMESTRE X	SEMESTRE IX	SEMESTRE X	
INF-131 Programación 3 5	QUI-010 Química y Sociedad 3 5	INF-134 Estructuras de Datos 1 3 5	INF-253 Lenguajes de Programación 12 3 5	MAT-021 Matemáticas I 5 8	MAT-022 Matemáticas II 2 3 5 7	FIS-100 Introducción a la Física 3 6	FIS-110 Física General I 2 3 5 8	FIS-130 Física General III 7 8 4 8	FIS-120 Física General II 7 8 4 8	INF-239 Bases de Datos 12 3 5	INF-236 Análisis y Diseño de Software 24 18 3 5	INF-225 Ingeniería de Software 36 3 5	INF-322 Diseño Interfaces Usuarios 42 3 5	INF-302 Electivo Informática II 3 5
MAT-021 Matemáticas I 5 8	MAT-022 Matemáticas II 2 3 5 7	MAT-023 Matemáticas III 7 4 7	MAT-024 Matemáticas IV 13 4 6	FIS-100 Introducción a la Física 3 6	FIS-110 Física General I 2 3 5 8	FIS-130 Física General III 7 8 4 8	FIS-120 Física General II 7 8 4 8	FIS-140 Física General IV 14 20 4 8	INF-276 Ingeniería, Informática y Sociedad 28 3 5	INF-245 Arquitectura y Organización de Computadores 12 3 5	INF-246 Sistemas Operativos 25 3 5	INF-256 Redes de Computadores 31 3 5	INF-343 Sistemas Distribuidos 43 3 5	
FIS-100 Introducción a la Física 3 6	FIS-110 Física General I 2 3 5 8	FIS-130 Física General III 7 8 4 8	FIS-120 Física General II 7 8 4 8	INF-101 Introducción a la Ingeniería 2 3	INF-152 Estructuras Discretas 1 2 3 5	INF-155 Informática Teórica 12 15 3 5	INF-170 Economía I A 22 3 5	INF-280 Estadística Computacional 1 13 3 5	INF-221 Algoritmos y Complejidad 15 18 3 5	INF-285 Computación Científica 10 33 3 5	INF-293 Investigación de Operaciones 24 3 6	INF-295 Inteligencia Artificial 34 33 3 5	INF-312 Electivo II 3 5	INF-312 Electivo II 3 5
HRW-132 Humanístico I 2 3	HRW-133 Humanístico II 2 3	INF-260 Teoría de Sistemas 9 3 5	INF-170 Economía I A 22 3 5	INF-101 Introducción a la Ingeniería 2 3	INF-152 Estructuras Discretas 1 2 3 5	INF-155 Informática Teórica 12 15 3 5	INF-170 Economía I A 22 3 5	INF-280 Estadística Computacional 1 13 3 5	INF-221 Algoritmos y Complejidad 15 18 3 5	INF-285 Computación Científica 10 33 3 5	INF-293 Investigación de Operaciones 24 3 6	INF-295 Inteligencia Artificial 34 33 3 5	INF-312 Electivo II 3 5	
DEW-100 Educación Física I 1 2	DEW-101 Educación Física II 1 2	INF-1 Libre 1/ Actividad co-curricular 1 2	INF-2 Libre 2/ Actividad co-curricular 1 2	DEW-100 Educación Física I 1 2	DEW-101 Educación Física II 1 2	INF-1 Libre 1/ Actividad co-curricular 1 2	INF-2 Libre 2/ Actividad co-curricular 1 2	INF-3 Libre 3/ Actividad co-curricular 1 2	INF-4 Libre 4/ Actividad co-curricular 1 2	INF-5 Libre 5/ Actividad co-curricular 1 2	INF-6 Libre 6/ Actividad co-curricular 1 2	INF-7 Libre 7/ Actividad co-curricular 1 2	INF-309 Trabajo de Título 1 1 2	INF-310 Trabajo de Título 2 12 20
14 24	18 28	18 32	18 31	14 24	18 28	18 32	18 31	17 30	16 27	16 28	16 27	16 27	16 27	12 20
BACHILLER EN CIENCIAS DE LA INGENIERÍA				LICENCIATURA EN CIENCIAS DE LA ING. INFORMÁTICA				INGENIERÍA CIVIL INFORMÁTICA						

Figura 1: Malla curricular Ing. Civil Informática UTFSM Casa Central (2023)

Fuente: Departamento de Informática UTFSM

1.2. Problemas detectados

A continuación se listan algunos de los problemas detectados a los que pueden enfrentarse los estudiantes :

- Dificultad en la equitativa distribución de carga académica en los distintos semestres a cursar por el estudiante durante la personalización de su malla curricular.
- sobrecarga académica provocada por una mal asignación curricular.
- Desconocimiento sobre los prerrequisitos y carga académica de algunas asignaturas.
- No existe una herramienta que facilite a los estudiantes una reestructuración de su malla curricular frente a la necesidad (o voluntad) de cambiar el orden de ésta.
- Una mala organización de la malla curricular puede causar una extensión considerable de la duración de la carrera y/o una mala experiencia universitaria para el estudiante.
- Una mala estructura curricular puede generar insatisfacción y/o una percepción de dificultad por parte del estudiante, optando por un cambio de carrera, universidad o deserción.
- La sobrecarga académica puede derivar en estrés y problemas de salud mental a corto y largo plazo.

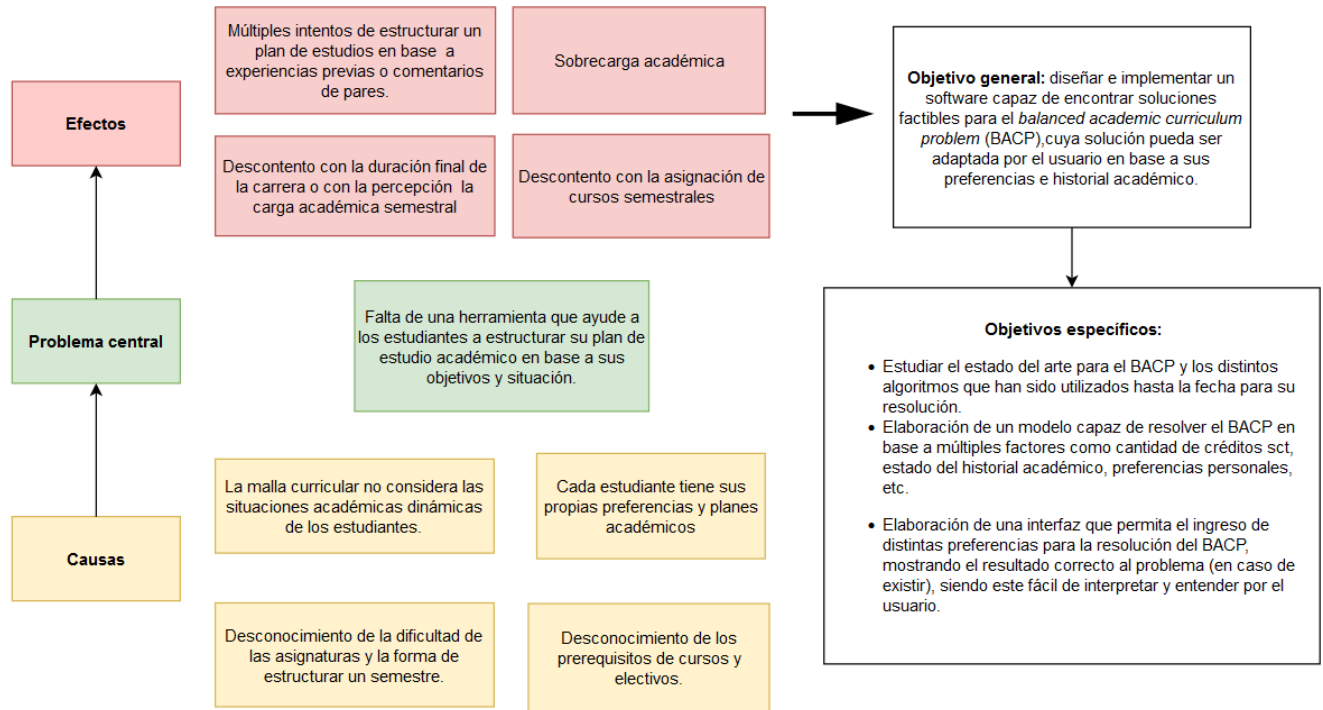


Figura 2: Diagrama de causas y efectos
Fuente: Elaboración propia

1.3. Alcance

El desarrollo de este trabajo se centra en la malla curricular(2023) del departamento de ing. civil informática de la Universidad Técnica Federico Santa María Casa Central, esto debido a la familiaridad por parte del autor con la universidad y la carrera, además del conocimiento de la relación entre asignaturas, carga académica y dificultades presentadas en el transcurso de la carrera. Esto permite un mejor acercamiento y análisis en cuanto a los resultados obtenidos.

Se medirán indicadores de éxito como el resultado obtenido en base a la aplicación de la solución, considerando la malla curricular (2023) y recepción del uso del software por parte de estudiantes y/o exalumnos de ing. civil informática UTFSM Casa Central.

Al elegir este enfoque, se prioriza trabajar con datos conocidos, mejorando el desarrollo del proyecto dentro de los tiempos establecidos y logrando un sentido de propósito en la resolución de una herramienta que podría ayudar a futuros estudiantes.

1.4. Motivación

La motivación para la realización de este documento se basa en el aprendizaje del desarrollo de modelos, como estos pueden variar dependiendo las necesidades de quien lo modela y el aprendizaje de la resolución de estos en base a distintas técnicas y algoritmos (centradas en la aplicación de metaheurísticas).

También es importante destacar la importancia del BACP en el trasfondo universitario-académico, dado que desde su planteamiento hasta el día de hoy se ha intentado resolver con múltiples técnicas y enfoques, tratando de proponer una solución a la distribución académica de diversos departamentos universitarios. Además de tener la oportunidad de plantear una solución que pueda ser utilizada por estudiantes futuros en el transcurso de su carrera universitaria. Dado esto y la complejidad que conlleva, es que se vuelve un importante caso de investigación en el área de la inteligencia artificial.

1.5. Objetivos

1.5.1. Objetivo general

Diseñar e implementar un software capaz de encontrar soluciones factibles para el balanced academic curriculum problem (BACP) estudiantil, cuya solución pueda ser adaptada por el usuario en base a sus preferencias e historial académico.

1.5.2. Objetivos específicos

- Estudiar el estado del arte para el BACP universitario y los distintos algoritmos que han sido utilizados hasta la fecha para su resolución.
- Elaboración de un modelo capaz de resolver el BACP estudiantil en base a múltiples factores como cantidad de créditos sct, historial académico y cantidad deseable de semestres.
- Elaboración de una interfaz que permita el ingreso de distintas preferencias para la resolución del BACP, generando una solución cercana al óptimo (en caso de existir) y sea esta presentada al usuario a través de la visualización de una malla curricular.

CAPÍTULO 2

MARCO CONCEPTUAL

En el siguiente capítulo se definirá el problema además de presentar el estado del arte con la revisión de las principales propuestas de la comunidad científica para disponer de una mejor visión y comparativa moderna del problema, resumiendo los métodos utilizados y sus resultados.

2.1. El Problema de Balanceo de Malla Curricular

El *Balanced Academic Curriculum Problem* (BACP) es un problema de optimización combinatoria que surge en la planificación de currículos académicos. Su objetivo es asignar un conjunto de asignaturas a un número fijo de períodos académicos de manera que se respeten ciertas restricciones y se optimice el balance de la carga académica.

Las principales restricciones del problema incluyen:

1. Cada asignatura debe ser asignada a un único semestre (o período académico) dentro de un rango permitido.
2. Deben cumplirse las relaciones de precedencia entre asignaturas, es decir, una asignatura no puede asignarse antes que su(s) prerrequisito(s).
3. El número total de créditos o carga académica en cada período debe mantenerse dentro de un rango permitido.
4. La solución debe minimizar la diferencia entre las cargas académicas de los períodos, logrando un balance óptimo.

El BACP es un problema NP-completo[Chiarandini *et al.*, 2012], además de ser un problema interesante dado que es semejante a varios tipos de problemas, como el *bin packing*, problemas de planificación (*scheduling problem*) y problemas de balanceo (*balancing problem*) [Monette *et al.*, 2007].

En la formulación original propuesta y resuelta por [Castro y Manzano, 2001], se consideran los siguientes conceptos y variables, las cuales son la base para la resolución general del problema:

- **Curriculum académico:** Un curriculum académico es definido por un conjunto de cursos y prerrequisitos relacionados entre sí.

- Número de períodos : Los cursos deben ser asignados dentro de un máximo número de periodos académicos.
- Carga académica : Cada curso tiene asociado un número de créditos o unidades que representan el esfuerzo teórico necesario para satisfactoriamente aprobarlo.
- Prerrequisitos: Algunos cursos pueden tener uno o más cursos como prerrequisito(s) obligatorio(s), es decir, algunas asignaturas deben ser cursadas antes que otras.
- Mínima carga académica: Se requiere una cantidad mínima de créditos académicos por periodo para ser considerado un estudiante a tiempo completo (alumno regular).
- Máxima carga académica: Se permite una cantidad máxima de créditos académicos por periodo para evitar la sobrecarga académica.
- Mínimo número de cursos: Se requiere una cantidad mínima de cursos por periodo para ser considerado un estudiante a tiempo completo (alumno regular).
- Máximo número de cursos: Se permite una cantidad máxima de cursos por periodo para evitar sobrecarga académica.

2.2. Estado del arte

En la siguiente sección se revisarán brevemente las principales formas de resolución del BACP propuestas por la comunidad científica en las últimas décadas, con la finalidad de lograr un mejor entendimiento del problema, su alcance, las formas de resolución y resultados.

2.2.1. Solución vía Programación con Restricciones

[Castro y Manzano, 2001] sientan las bases de lo que sería la estructura principal del problema, sus parámetros, objetivos, principios de resolución e instancias de prueba, discutiendo además la importancia del papel que juegan las heurísticas en la resolución de problemas con un acercamiento basado en restricciones y el cómo estas técnicas permiten resolver de manera eficiente problemas de optimización combinatoria demasiado difíciles para las técnicas de programación de enteros.

Este trabajo buscó resolver las mallas académicas de las tres carreras de informática ofrecidas por la Universidad Técnica Federico Santa María Casa Central durante el año 2001 (8,10 y 12 semestres), lo que resulta de particular interés para el desarrollo de este documento dado su alcance.

Para la resolución y evaluación de resultados se utilizan dos acercamientos:

1. Programación en enteros: Se utiliza la técnica más exitosa conocida hasta el momento para la resolución de modelos basados en programación de enteros, además de estar incluida en casi todos los paquetes de resolución matemática, denominada *Branch & Bound* (B&B) [Wolsey y Nemhauser, 1999]. La resolución fue llevada a cabo a través de *lp_solve*¹ para instancias de 8, 10 y 12 semestres con 46, 42 y 66 asignaturas respectivamente (estas instancias son las consideradas *benchmark* ² [Gent y Walsh, 1999]).
2. Programación con restricciones: Para esta solución se utilizó el lenguaje de programación Oz³, la idea es (asumiendo que se trata de un problema de minimización) utilizar un límite superior que represente la mejor solución posible obtenida hasta el momento para luego resolver una serie de CSP's, cada uno de los cuales proporciona una mejor solución con respecto a la función de optimización. Específicamente, se computa una solución α al problema P y se añade la restricción $f < \alpha(f)$ la cual restringe el conjunto de posibles soluciones al conjunto con mejores valores de solución, de esta forma si al añadir una última restricción la solución se vuelve inviable se considera como mejor solución la solución previa. Para este caso también se utilizan las tres instancias de referencia de CSPLib².

Los resultados son comparados directamente en cuanto a tiempo de resolución y, en el caso de la programación con restricciones, seccionada en base al método y heurística utilizada, esto resumido en la siguiente tabla:

¹*lp_solve* es un solver de programación lineal entera mixta (MILP). <https://web.mit.edu/lpsolve/doc/>

²Instancias de referencia, estas contienen los problemas principales de testeo con sus parámetros correspondientes <https://www.csplib.org/Problems/prob030/data/>

³Oz es un lenguaje de programación multiparadigma desarrollado originalmente en el laboratorio de Programación de Sistemas en la Universidad de Saarland por Gert Smolka y sus estudiantes a comienzos de 1990 <http://mozart2.org/>

8-period problem					
	lp_solve	Oz			
		grouped by course		grouped by period	
		naive	naive	naive	naive
Time in seconds for first solution (worst quality)	1.7	0.1	0.1	0.2	0.1
	∞	∞	0.1	∞	0.1
10-period problem					
Time in seconds for first solution (worst quality)	5.9	0.2	0.1	0.2	0.1
	∞	∞	3.6	∞	0.1
12-period problem					
Time in seconds for first solution (worst quality)	∞	0.4	0.5	0.3	0.3
	∞	∞	∞	∞	0.3

Tabla 1: Resumen comparativo de rendimiento programación con restricciones vs programación en enteros

Fuente: [Castro y Manzano, 2001], Table 3, pág. 11

Los valores solución óptimos encontrados en estas instancias serán los valores de referencia a conseguir durante la demostración de todos los algoritmos de BACP presentados en esta sección, estos son:

- Problema de 8 periodos: 17 créditos
- Problema de 10 períodos: 14 créditos
- Problema de 12 períodos: 17 créditos

Se puede apreciar claramente cómo la programación con restricciones y una heurística de ordenamiento inverso de variables obtiene el resultado óptimo para las tres instancias propuestas, mientras que la programación en enteros no logra optimizar (dentro del tiempo medible del experimento) las instancias de 10 y 12 semestres, mientras que para el semestre que sí pudo mejorar, este tomó una cantidad substancialmente mayor de tiempo.

Los valores resultados agrupados por heurística utilizada pueden verse en el siguiente gráfico:

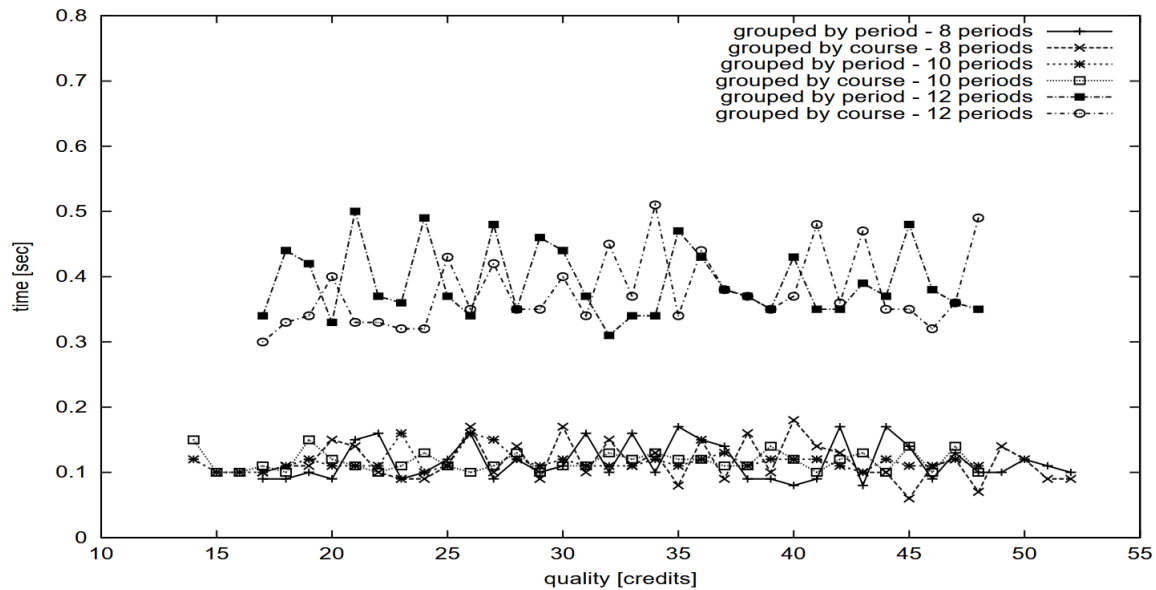


Figura 3: Resultados obtenidos al variar la variable de ordenamiento en Oz
[Castro y Manzano, 2001], Fig. 4, pág. 11

2.2.2. Solución vía Ant Colony Optimization (ACO)

La siguiente solución propuesta por [Rubio *et al.*, 2013], explora una resolución a través del algoritmo de optimización basado en colonia de hormigas a través del modelo conocido como *Beast-Worst Ant System* (BWAS). [Cordon *et al.*, 2000]

ACO es un algoritmo metaheurístico inspirado en el comportamiento de las hormigas reales al buscar alimento. Estas depositan feromonas en los caminos, y otras hormigas tienden a seguir rutas con mayor concentración de feromonas, lo que genera un proceso de retro-alimentación positiva. En ACO, esta ruta a seguir es interpretada como un grafo donde el movimiento de las hormigas es una serie de decisiones probabilísticas basadas en la cantidad de feromonas y la heurística local de cada ruta; estas rutas son entonces las soluciones obtenidas.

BWAS tiene tres componentes principales:

1. **Regla de actualización de feromonas Best-Worst:** Esta regla refuerza los bordes contenidos en soluciones buenas y penaliza todos los bordes contenidos en la peor solución actual.
2. **Mutación del rastro de feromonas:** Los rastros de feromonas se mutan, con cierta probabilidad, para introducir diversidad en el proceso de búsqueda. Los rastros de

feromonas se mutan sumando o restando la misma cantidad de feromonas.

3. **Reinicio del proceso de búsqueda:** Cuando los rastros de feromonas relacionados con los bordes de las mejores soluciones son muy altos y los restantes están cerca de cero (estancamiento), la matriz de feromonas se restablece a su valor inicial.

Las principales diferencias que se presentan entre ACO base y ACO BWAS radica en el manejo mismo de las feromonas. Mientras que en ACO todas las hormigas aportan a la actualización de las feromonas, en BWAS solo dos hormigas contribuyen a su actualización; La mejor hormiga (mejor resultado de solución durante la iteración actual) y la peor hormiga (peor resultado de solución durante la iteración actual), esto debido a que en BWAS existe un fuerte enfoque en el mejoramiento de la solución global penalizando el peor resultado obtenido.

Instance	BWAS Best	BWAS Worst	BWAS Average	σ	lp-solve Best	lp-solve Worst	Optimal
Bacp8	17	18	17.3	0.48	17	54	17
Bacp10	14	15	14.2	0.42	24	48	14
Bacp12	18	20	18.4	0.69	∞	∞	18

Tabla 2: Resumen comparativo de rendimiento - Instancias de referencia, ACO

Fuente: [Rubio *et al.*, 2013], Table 3, pág. 5

En la figura anterior se presentan los valores mínimos obtenidos para la distribución de créditos, donde se puede notar que BWAS Best iguala los valores óptimos esperados.

2.2.3. Solución vía Algoritmos Genéticos (GA)

El algoritmo propuesto por [Villalobos-Cid *et al.*, 2019] utiliza un acercamiento genético para abordar el gran espacio de búsqueda que presenta el problema y utiliza las ventajas intrínsecas (como lo son la definición de mutaciones y cruces) de GA para llegar a soluciones óptimas.

Los algoritmos genéticos son una técnica metaheurística inspirada en los principios de la evolución biológica, como la selección natural, mutación, cruce (crossover) y supervivencia del más apto. Se utilizan para resolver problemas de optimización, especialmente aquellos con grandes espacios de búsqueda. La estructura clásica de un algoritmo genético [Holland, 1975] posee la siguiente estructura, parámetros y conceptos.

1. **Población inicial:** Se genera una población inicial de soluciones candidatas (individuos), normalmente representadas como cadenas binarias o cromosomas.

2. **Evaluación:** Cada individuo se evalúa mediante una función objetivo para determinar su calidad (*fitness*).
3. **Selección:** Se eligen los individuos para reproducirse, basándose en su *fitness* (por ejemplo, selección por torneo o ruleta).
4. **Cruce (crossover):** Se combinan los cromosomas de dos padres para crear nuevos individuos (hijos).
5. **Mutación:** Se realizan pequeños cambios aleatorios en los cromosomas para mantener la diversidad genética.
6. **Reemplazo:** La nueva generación reemplaza parcial o totalmente a la población actual.
7. **Iteración:** El proceso se repite hasta que se cumpla un criterio de parada, como alcanzar un número máximo de iteraciones o encontrar una solución óptima.

Para este caso particular se tienen las siguientes definiciones:

- **Población inicial:** Se construye en base a una interpretación matricial, donde la primera población solución P construida debe respetar las restricciones, luego se genera una segunda población Q en base a operadores genéticos.
- **Algoritmos de Cruce:** Se eligen dos padres (S_1 y S_2) de la población P utilizando un torneo binario estocástico basado en una tasa probabilista, luego se comparan S_1 y S_2 , si se encuentran diferencias en los cursos se selecciona aleatoriamente uno de estos cursos como pivote. Una vez alcanzada la posición del pivote la solución cambia y se genera una nueva solución a partir de esta posición, las mejores son asignadas a Q
- **Operador de Mutación:** Cuando se selecciona una solución, se identifican los períodos con la carga académica más alta y se selecciona uno de ellos de forma aleatoria. Luego, un curso se mueve a otro período con menos carga académica, de acuerdo con las restricciones.

Similar a casos anteriores, se resuelven las instancias de referencias postuladas por [Castro y Manzano, 2001], obteniendo los resultados óptimos de 18, 17 y 14 para 12, 8 y 10 semestres respectivamente:

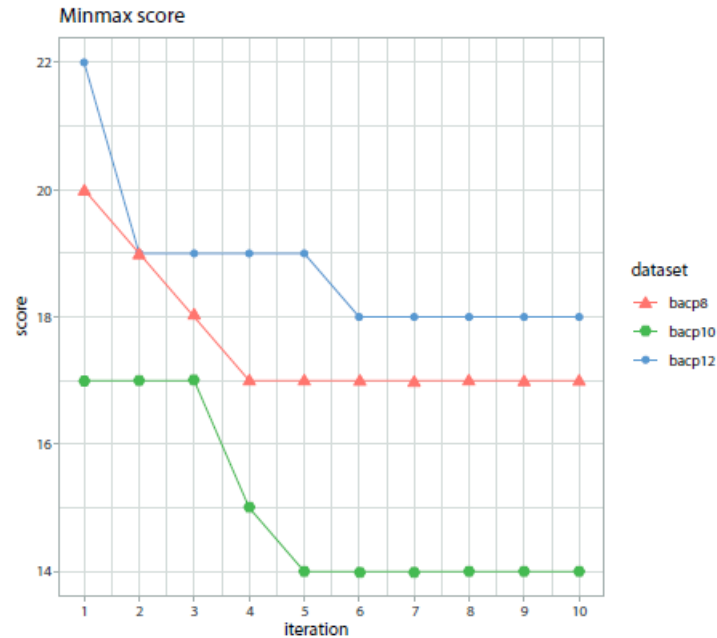


Figura 4: Convergencia de GA

Fuente: [Villalobos-Cid *et al.*, 2019], Figura 3, pág. 5

La literatura disponible no presenta mucha información respecto a tiempos de ejecución para el caso de algoritmos genéticos, sin embargo, [Lambert *et al.*, 2006] resuelve las instancias de referencias con una hibridación entre algoritmo genético y propagación con restricciones. Sus resultados son los siguientes:

Sol quality	bacp 8 [s]	bacp 10 [s]	bacp 12 [s]
24	0.47	4.71	2.34
23	0.54	4.67	2.40
22	0.61	3.68	2.48
21	0.61	4.36	2.76
20	0.69	4.63	3.20
19	0.83	4.95	4.25
18	1.20	5.13	35.20
17	15.05	5.60	
16		6.39	
15		8.53	
14		34.84	

Tabla 3: Resultados utilizando GA + CP

Fuente: [Lambert *et al.*, 2006] pág. 416, fig. 4

2.2.4. Solución vía Técnicas de búsqueda local híbridas: Steepest Descent (SD), First Descent (FD), Radomized Hill Climbing (RHC), Simulated Annealing (SA), Tabu Search (TS) y Dynamic Tabu Search (DTS)

En esta sección se presentan técnicas de búsqueda local híbridas propuestas por [Di Gaspero y Schaerf, 2008] con la finalidad de comparar el rendimiento de cada una frente a las instancias de referencia, el rango de estas estrategias van desde muy triviales hasta metaheurísticas más complejas. En detalle, se consideran las siguientes:

- **Steepest Descent (SD):** En cada paso de la búsqueda, se explora toda la vecindad y se realiza el mejor movimiento de mejora. La búsqueda se detiene cuando se alcanza un mínimo local.
- **First Descent (FD):** A diferencia de SD, esta técnica interrumpe la exploración de la vecindad tan pronto como se encuentra el primer vecino que mejora. Al igual que SD, la búsqueda se detiene cuando se alcanza un mínimo local.
- **Radomized Hill Climbing (RHC):** En cada paso de la búsqueda, se selecciona un vecino aleatorio. El movimiento se realiza solo si el costo del vecino es menor o igual que el costo de la solución actual.
- **Simulated Annealing (SA):** De forma similar a RHC, se selecciona un vecino aleatorio en cada paso. El movimiento se realiza si es uno de mejora o de acuerdo con una probabilidad exponencial que disminuye con el tiempo. En detalle, si el costo del movimiento es $\Delta f > 0$, el movimiento se acepta con una probabilidad $e^{-\Delta f/T}$, donde T es un parámetro que disminuye con el tiempo llamado temperatura. En cada nivel de temperatura, se muestrean σ_N vecinos de la solución actual y la nueva solución se acepta de acuerdo con la distribución de probabilidad mencionada anteriormente. Posteriormente, el valor de T se modifica utilizando un programa geométrico, es decir, $T' = \beta \cdot T$, en el que el parámetro $\beta < 1$ se llama tasa de enfriamiento.
- **Tabu Search (TS):** En cada paso, se explora un subconjunto de la vecindad y el vecino que proporciona el valor de costo mínimo se convierte en la nueva solución independientemente del hecho de que su valor de costo sea mejor o peor que el actual. El subconjunto está inducido por la lista tabú, es decir, una lista de los movimientos realizados recientemente, que están actualmente prohibidos y, por lo tanto, excluidos de la exploración.
- **Dynamic Tabu Search (DTS):** La estrategia de búsqueda es la misma que TS; sin embargo, esta variante del algoritmo está equipada con un mecanismo que cambia adaptativamente la forma de la función de costo. En detalle, las restricciones que se satisfacen durante un número determinado de iteraciones se relajarán (el peso se reduce en un factor $\gamma > 1$, es decir, $w' = w/\gamma$) para permitir la exploración de soluciones donde esas restricciones no se cumplen. Por el contrario, si alguna restricción no se satisface, se ajusta ($w' = w \cdot \gamma$) con el objetivo de dirigir la búsqueda hacia su satisfacción.

Los resultados experimentales pueden ser apreciados en la siguiente tabla:

Instance	SD		FD		RHC		SA		TS		DTS	
	%succ	time	%succ	time	%succ	time	%succ	time	%succ	time	%succ	time
bacp8	28.3	.0006	19.6	.0008	49.5	.1450	100.0	.0042	100.0	.0023	100.0	.0026
bacp10	6.2	.0009	2.2	.0012	33.3	.1906	100.0	.0429	100.0	.0046	100.0	.0060
bacp12	1.3	.0033	0.2	.0040	9.0	.2775	97.9	.1764	98.9	.0459	96.7	.08426

Tabla 4: Resultados para las instancias de CSPLib
Fuente: [Di Gaspero y Schaerf, 2008], Table 3, pág. 154

Aquí es importante notar que las metaheurísticas (a partir de SA) tienen un desempeño muchísimo mejor que las estrategias triviales como SD o FD y, dentro de estas metaheurísticas todas tienen un gran rendimiento tanto temporal como en cuanto a su probabilidad de éxito, demostrando que cualquiera de estas metaheurísticas puede resolver satisfactoriamente las instancias de referencia de CSPLib.

2.2.5. Solución vía Algoritmo de Luciérnagas (FA)

A continuación [Rubio *et al.*, 2018] proponen la resolución del BACP con una técnica de optimización basada en la atracción de las luciérnagas.

La optimización basada en luciérnagas es un algoritmo de optimización metaheurístico inspirado en el comportamiento de las luciérnagas, específicamente en su capacidad de emitir destellos de luz para atraer a otras luciérnagas. Este algoritmo, propuesto por [Yang y He, 2013], se utiliza para resolver problemas complejos de optimización, especialmente aquellos que son no lineales, multimodales y de alta dimensión.

Este acercamiento tiene los siguientes principios básicos:

- **Atracción por la luz:** Las luciérnagas se atraen unas a otras dependiendo de la intensidad de la luz que emiten. Las luciérnagas más brillantes atraen a las más débiles. En el algoritmo, la calidad de una solución (o la "luz") se evalúa mediante una función objetivo, y las soluciones mejores (más cercanas al óptimo) atraen a las peores.
- **Movimiento de las luciérnagas:** Cada luciérnaga se mueve hacia otras que son más brillantes (mejores soluciones), y la velocidad de su movimiento depende de la diferencia en la intensidad de la luz. Si no hay luciérnagas más brillantes, la luciérnaga se mueve aleatoriamente.
- **Actualización de posiciones:** La posición de cada luciérnaga en el espacio de soluciones se actualiza en función de la atracción hacia las luciérnagas más brillantes. La ecua-

ción de actualización considera la distancia entre luciérnagas y la diferencia en brillo, junto con un término aleatorio para mantener la exploración del espacio.

Para esta resolución se utilizó una representación matricial de los datos dado que es más fácil de implementar y se puede decir con seguridad que el algoritmo de luciérnagas se adapta perfectamente a una representación binaria la cual propone una estructura matricial.

Solución	BACP8	BACP10	BACP12	REAL8	REAL10	REAL12
Mejor	17	14	18	17	13	17
Media	17.5	14.9	18.8	18.1	13.9	19.0
Peor	18	15	20	19	15	20
σ	.509	.305	.484	.681	.860	.831
Óptimo	17	14	18	13	13	18

Tabla 5: Síntesis de resultados - Firefly Algorithm
Fuente: [Rubio *et al.*, 2018], Figura tabla 2, pág. 111

La tabla anterior muestra los resultados para las instancias de prueba referenciales, además de 3 instancias extras basadas en asignaciones semestrales reales.

2.3. Resumen del capítulo

En este capítulo se presentó la definición e introducción general del *Balanced Academic Curricular Problem*, al igual que una serie de métodos con los cuales se encontró una solución al problema junto a sus correspondientes resultados. Esto con la finalidad de comprender mejor el planteamiento del problema y los distintos acercamientos que han sido propuestos por la comunidad científica durante las últimas décadas además de tener un entendimiento general de los resultados esperados y la comparativa entre estos.

El *Balanced Academic Curricular Problem* (BACP) es un problema bien documentado en la literatura académica, principalmente como un desafío en la investigación en optimización combinatoria. Sin embargo, su aplicación directa en instituciones educativas o como herramienta práctica no es tan comúnmente reportada en publicaciones específicas. Las instituciones educativas suelen abordar problemas similares relacionados con la planificación de currículos, pero a menudo utilizan software comercial o sistemas personalizados, que no siempre mencionan explícitamente el uso de formulaciones como el BACP.

Algunas posibles áreas donde podría haber registros del uso del BACP o enfoques similares son:

1. **Aplicaciones de investigaciones académicas:** Una plétora de documentos de investigación han resuelto el BACP y demostrado su validez a través de la resolución de curriculums reales, lo cual indica hasta cierta medida su utilidad en el desarrollo de herramientas o reestructuraciones de curriculums académicos actuales.
2. **Herramientas de software para la planificación académica:**
 - Sistemas de gestión académica como *Banner*, *PeopleSoft* o plataformas similares, podrían usar algoritmos que aborden aspectos del BACP, aunque no necesariamente bajo ese nombre.
 - Herramientas especializadas en programación académica, como *UniTime*, podrían incorporar enfoques de optimización para equilibrar la carga académica.

En el siguiente capítulo se presenta la propuesta de solución.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

En este capítulo se planteará el problema estudiantil a resolver y la propuesta para el BACP, así como la base algorítmica y matemática del mismo, la metaheurística utilizada para encontrar mejores soluciones, los motivos de su elección y aplicación de las soluciones en forma de herramienta de software.

3.1. Formulación Matemática del problema

El modelo propuesto originalmente en [Castro y Manzano, 2001] utiliza el siguiente modelo de programación en enteros, el cual servirá de base para el entendimiento general del problema y el desarrollo de la solución inicial.

■ Parámetros

- m : Numero de asignaturas.
- n : Numero de semestres
- α_i : Numero de créditos de la asignatura i ; $\forall i = 1 \dots m$
- β : Mínima carga académica permitida por semestre.
- γ : Máxima carga académica permitida por semestre.
- δ : Cantidad mínima de asignaturas a tomar por semestre.
- ε : Máxima cantidad de asignaturas a tomar por semestre.

■ Variables de decisión

- $x_{ij} = \begin{cases} 1 & \text{si la asignatura } i \text{ pertenece al semestre } j \\ 0 & \text{en caso contrario} \end{cases}$
- c_j : Carga académica del semestre j , $\forall j = 1, \dots, n$

$$c_j = \sum_{i=1}^m \alpha_i \cdot x_{ij} \quad \forall j = 1, \dots, n \quad (1)$$

Función objetivo

$$\text{Min } c = \text{máx}\{c_1, \dots, c_n\} \quad (2)$$

Restricciones

- Todas las asignaturas i deben ser asignadas a un periodo j :

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, \dots, m \quad (3)$$

- Si la asignatura b tiene como **prerrequisito** a la asignatura a , entonces:

$$x_a < x_b$$

$$\sum_{r=1}^{j-1} x_{ar} \leq x_{bj} \quad \forall j = 2, \dots, n \quad (4)$$

- Máxima y Mínima carga academica por semestre:

$$\beta \leq c_k \leq \gamma \quad \forall k = 1, \dots, n \quad (5)$$

- Máxima y Mínima cantidad de asignaturas por semestre:

$$\delta \leq \sum_{i=1}^m x_{ij} \leq \varepsilon \quad \forall j = 1, \dots, n \quad (6)$$

Para este caso en específico no se considerarán como restricciones fuertes (6) dado que en la UTFSM se prioriza la cantidad de créditos tomados por el estudiante durante un periodo semestral por sobre la cantidad de asignaturas, de esta forma se define si el estudiante es en efecto de dedicación completa (mínima carga académica) o posee una sobrecarga académica (máxima carga académica), además se utilizará la siguiente función objetivo para el cálculo del valor de la solución :

$$Objetivo(h) = \min \sum_{k=1}^n |C_k - AVG|^2 \quad (7)$$

$$AVG = \frac{\sum_{k=1}^m \alpha_m}{n} \quad (8)$$

La ventaja que presenta (7) por sobre (2) es que el error absoluto cuadrático expone una mejor información respecto a la diferencia entre el valor real asignado al semestre y el mejor valor posible alcanzable si se pudieran distribuir todos los créditos equitativamente entre todos los semestres académicos a cursar. Además permite una mejor interpretación de la solución generando valores mucho más explícitos y notables frente a la diferencia óptima,

de esta forma es mucho más fácil analizar e interpretar la solución en cada semestre diferenciando la calidad de solución y sus repercusiones en el resto de semestres.

El modelo rígido presentado anteriormente crea un conflicto con el modelo actual semiflexible, en donde los estudiantes tienen más control y responsabilidad frente al desarrollo de su carrera universitaria, por ejemplo, la asignación de cursos mínimos y máximos por semestre se ve en conflicto con la asignación mínima y máxima de créditos para instancias donde pocas asignaturas maximicen la cantidad de créditos en el semestre o, sea necesario tomar muchas asignaturas para superar el mínimo necesario de créditos para ser considerado alumno a tiempo completo.

La distribución de créditos también es afectada por la cantidad de semestres a cursar, dado que un estudiante puede optar por acortar o alargar su carrera, un modelo rígido (o estático) no puede acomodarse a las necesidades estudiantiles modernas, es necesario cambiar ciertas estructuras y restricciones en función de los estatutos de la casa de estudio y los estudiantes. Aún así, el modelo definido previamente es una buena base de trabajo y plantea los principales objetivos y conceptos necesarios para el desarrollo de un modelo nuevo.

3.2. Metaheurística propuesta : *Tabu Search*

La solución propuesta al problema se basa en la metaheurística de *Tabu Search* (Búsqueda Tabú). La búsqueda Tabú es un método utilizado para resolver problemas de optimización combinatoria. La idea principal detrás de este método es que, mediante el uso de una “memoria”, se fuerza al método a explorar nuevas áreas en el espacio de búsqueda. Es decir, puede “memorizar” los operadores de búsqueda (comúnmente movimientos de alguna índole) que han sido utilizados recientemente, y estas instancias de decisión se vuelven prohibidas (tabú) durante la consideración de las siguientes soluciones.

Comparación frente a otras Metaheurísticas Una comparación de algoritmos fue realizada por [Di Gaspero y Schaerf, 2008], en esta se eligieron los siguientes algoritmos (Ver figura ??)

- Randomized Hill Climbing (RHC)
- Steepest Descent (SD)
- First Descent (FD)
- Simulated Annealing (SA)
- Tabu Search (TS)
- Dynamic Tabu Search (DTS)

La comparación se basa en el porcentaje de éxito de los algoritmos (obtención de solución óptima) y su tiempo de ejecución en instancias de CSPlib ([Gent y Walsh, 1999]). En dicha comparación se puede concluir que las técnicas no triviales SA, TS y DTS obtienen la solución óptima casi el cien por ciento de las veces con tiempos de ejecución bajos. Mientras que dentro de estas técnicas TS presenta un mejor porcentaje de éxito, así como los tiempos más bajos seguido de cerca por SA. Esto funciona como un buen indicativo del uso de TS para la resolución del BACP.

3.3. Representación de la Solución

Existen múltiples formas de representar la solución del BACP en la literatura, para la realización de este documento se evaluaron las dos principales en base a su espacio de búsqueda.

3.3.1. Representación matricial de la solución

La representación matricial supone que la malla curricular académica solución será interpretada como una matriz tal que los valores de la matriz serán 1 si la asignatura índice pertenece al semestre columna, 0 en caso contrario.

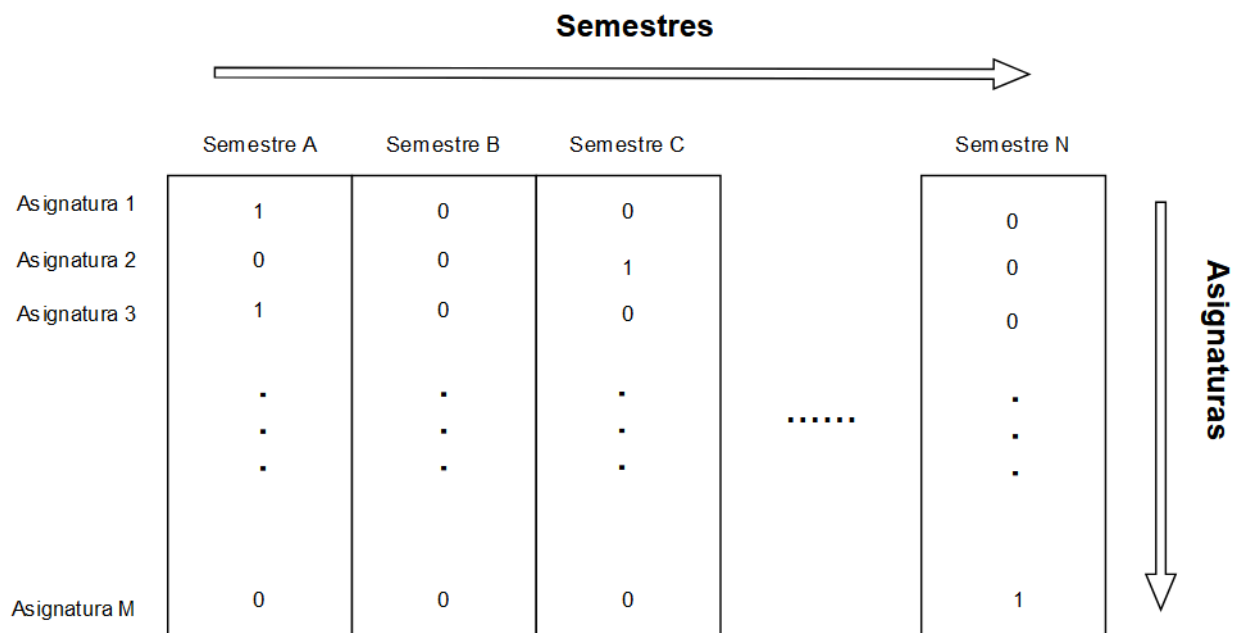


Figura 5: Representación Matricial
Fuente: Elaboración propia

Espacio de búsqueda matricial

$$\mathcal{S} = A \in \{0, 1\}^{m \cdot n}, \quad (9)$$

donde A es una matriz de dimensión $m \times n$, m representa la cantidad de semestres a cursar, n la cantidad de asignaturas y cada elemento de A puede tomar un valor de 0 o 1. Entonces el espacio vectorial de la malla académica de ing. civil informática de la UTFSM es :

$$2^{11 \cdot 59} = 2^{264} \approx 2,6 \cdot 10^{195} \quad (10)$$

3.3.2. Representación vectorial de la solución

La representación vectorial supone que la malla curricular académica solución será interpretada como un vector donde cada índice representa una asignatura y cada valor el semestre correspondiente a esa asignatura. Un ejemplo de esta representación puede ser visto en [Rubio *et al.*, 2013], donde se utiliza una representación similar para referirse al grafo solución de la ejecución de ACO BWAS.

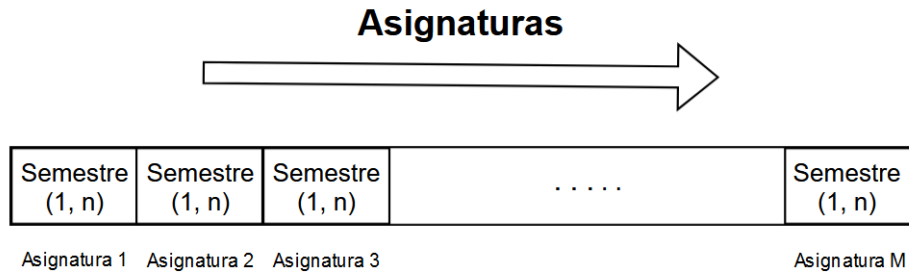


Figura 6: Representación Vectorial
Fuente: Elaboración propia

Espacio de búsqueda vectorial

$$\mathcal{S} = \mathbf{v} \in \{1, 2, \dots, m\}^n, \quad (11)$$

donde $\mathbf{v}$ es un vector de longitud m y cada elemento de $\mathbf{v}$ puede tomar un valor entero entre 1 y n . Entonces el espacio vectorial de la malla académica de ing. civil informática de la UTFSM es :

$$11^{59} \approx 2,3 \cdot 10^{61} \quad (12)$$

Cada representación tiene ventajas y desventajas, sin embargo, en este caso la magnitud de la diferencia entre ambos espacios de búsqueda es considerablemente grande, lo cual

vuelve a la representación vectorial la mejor elección, esto debido a que menos espacio de búsqueda implica menos posibles soluciones y/o movimientos innecesarios, lo cual mejora substancialmente la ejecución del algoritmo.

3.4. Descripción del algoritmo - Solución inicial

En la siguiente sección se explicará la lógica detrás del algoritmo y el cómo se obtiene una solución inicial la cual cumple las restricciones propuestas en la formulación matemática del problema, para esto, es necesario primero explicar el concepto de *rango* y el cómo se obtiene.

El rango son todas las posibles asignaciones semestrales que puede tener una asignatura considerando (inicialmente) sus límites inferiores y superiores, estos se obtienen a través de un grafo de orden topológico el cual ordena y asigna rangos en función de la cantidad de semestres y los prerrequisitos correspondientes a las asignaturas. Se tiene entonces (como ejemplo) que se debe aprobar cuatro asignaturas matemáticas en seis semestres.

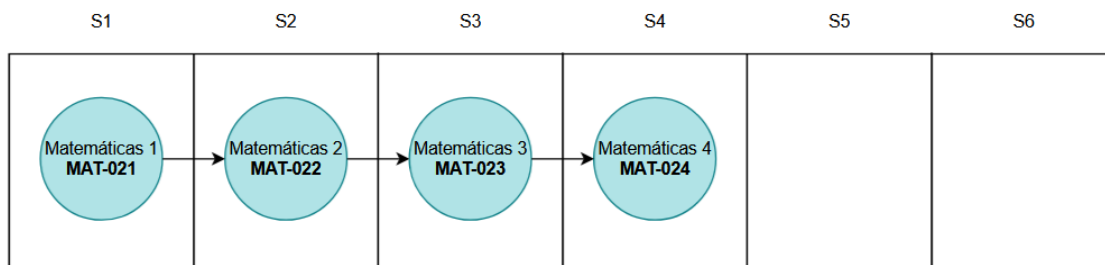


Figura 7: Rangos Matemáticos 1
Fuente: Elaboración propia

En la figura anterior se puede notar que se plantean las asignaturas, los semestres y un orden necesario de aprobación, además de ya estar realizadas las asignaciones semestrales, MAT-021 en el primer semestre y MAT-024 en el cuarto, por lo tanto, se puede deducir que la asignación de MAT-021 debe ser [1], MAT-022 [2], MAT-023 [3], y por último MAT-024, el cual está asignado en [4] sin embargo, como se puede notar, este también podría ser asignada en el semestre [5] o [6], esto debido a que no existe ninguna restricción que lo prohíba, volviendo su rango final [4,5,6].

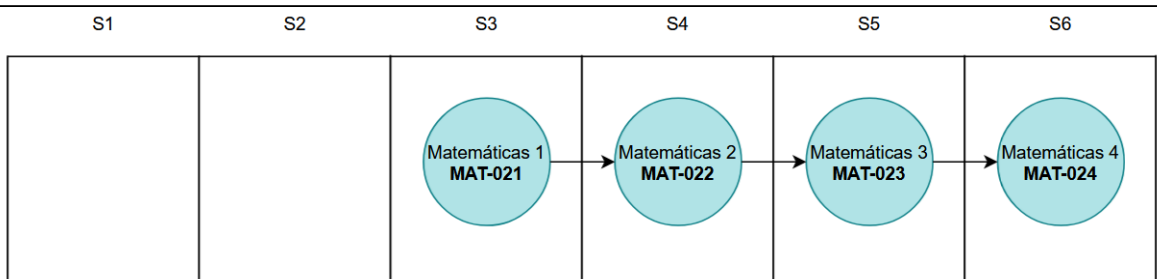


Figura 8: Rangos Matemáticos 2
Fuente: Elaboración propia

A continuación se presenta el caso opuesto, se puede notar que este es el mismo que el anterior, solo que esta vez la asignatura que queda con la flexibilidad de posición semestral es MAT-021 con un rango de [1,2,3].

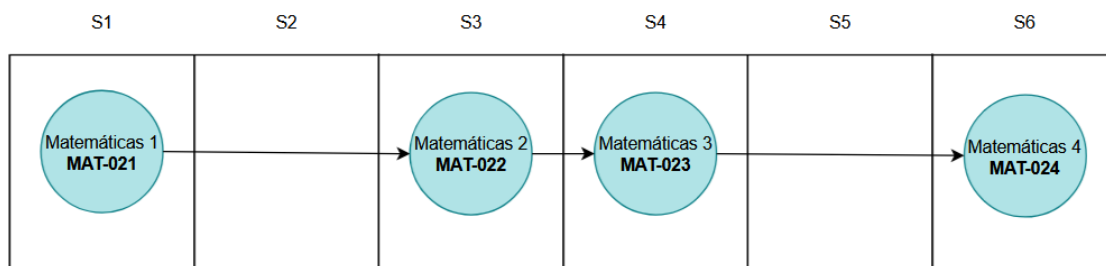


Figura 9: Rangos Matemáticos 3
Fuente: Elaboración propia

Como se puede ver en este último caso, MAT-021, al no tener prerequisites tiene un rango mínimo de [1] y un rango máximo de [2] pues su sucesor MAT-022 está asignado en el tercer semestre, por consiguiente el rango total de MAT-021 es [1,2]. A su vez MAT-022 está en el tercer semestre [3], dado que tiene un predecesor no puede ir en el primer semestre, sin embargo, su rango mínimo final está dado por la posición de su predecesor, ya que MAT-021 está en el primer semestre, el rango final de MAT-022 debe ser [2,3], ahora si se aplica la misma lógica en las últimas dos matemáticas se puede llegar a la conclusión que el rango de MAT-023 es [4,5] y el de MAT-024 [5,6].

De esta forma se puede definir los rangos como los posibles semestres asignables de cada asignatura, donde la cota inferior viene dada por la cantidad de asignaturas previas y la ubicación de la última asignatura previa + 1, mientras que la cota superior por cuantas asignaturas posteriores tenga el curso y la ubicación de la asignatura directamente posterior - 1. En consecuencia, la asignación de cada asignatura cambiará el posible rango de todas las

asignaturas dependientes de ésta, de esta forma, es posible trabajar indirectamente con la restricción de precedencia.

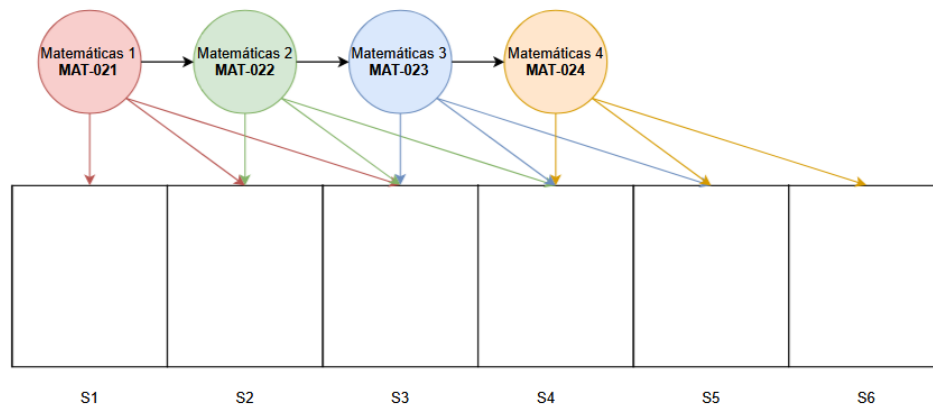


Figura 10: Posibles asignaciones de todas las matemáticas

Fuente: Elaboración propia

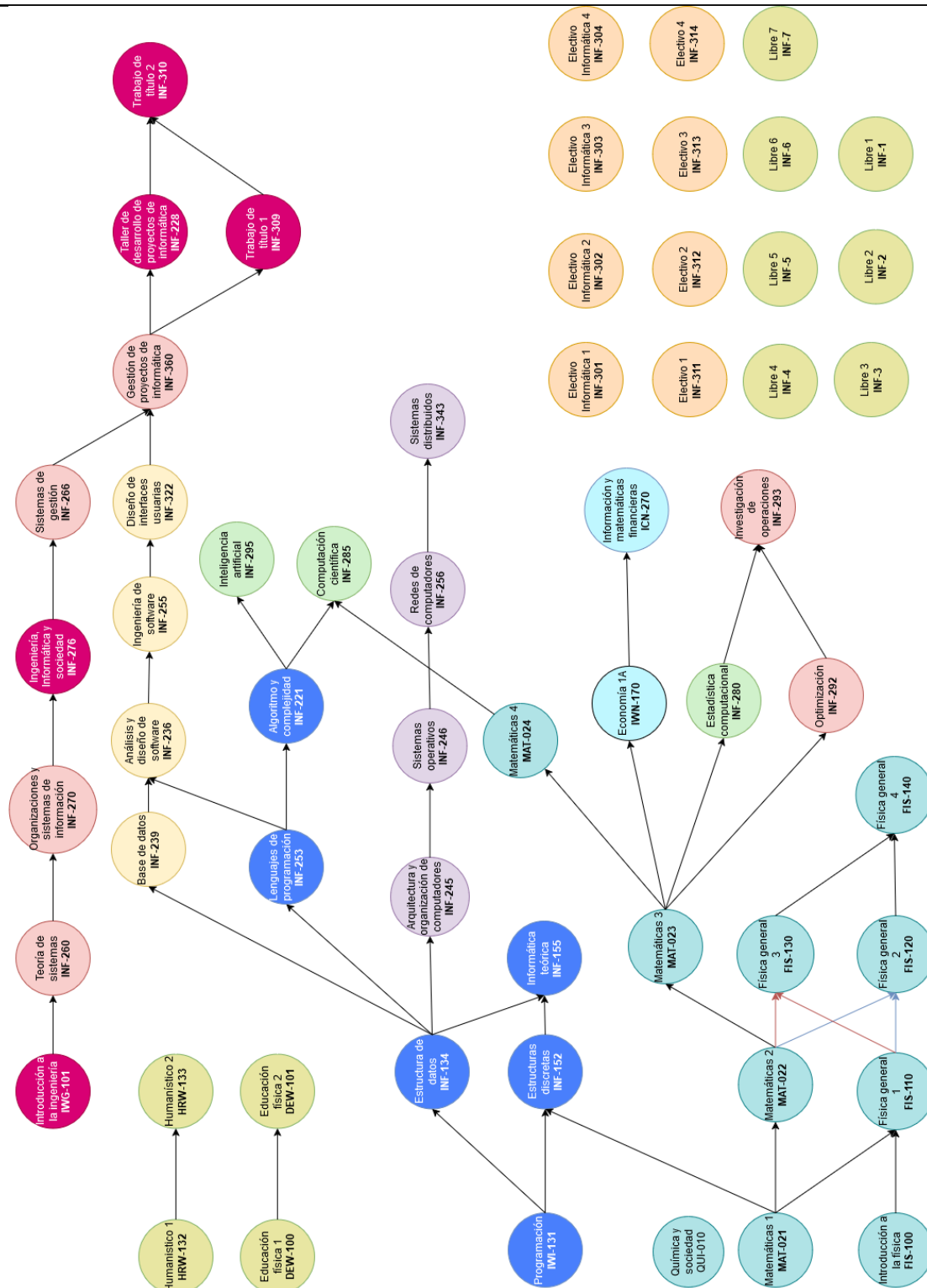


Figura 11: Diagrama de precedencia (2023).
Fuente: Elaboración propia

En el diagrama anterior se presentan todas las relaciones de precedencia que existen para las asignaturas de la malla del departamento de ing. civil informática UTFSM(año 2023). Las asignaturas independientes son representadas de forma similar a las dependientes, pero sin una conexión en forma de flecha con otras.

A continuación se presenta el pseudocódigo del algoritmo encargado de la asignación de rangos iniciales.

Algorithm 1 Calcular Rangos de Asignaturas

```
1: function CalcularRangos(asignaturas, prerequisites, total de semestres)
2:   Inicializar un grafo como lista de adyacencia.
3:   Inicializar grafo de prereq como lista de adyacencia.
4:   Inicializar grado_entrada[asignatura] en 0 para todas las asignaturas
5:   for all (asig, pre) en prerequisites do
6:     Añadir asig a la lista de grafo[pre]
7:     Añadir pre a la lista de prereq_grafo[asig]
8:     Incrementar grado_entrada[asig] en 1
9:   end for
10:  Crear cola con las asignaturas cuyo grado_entrada es 0
11:  Inicializar rangos[asig] como [1, total_semestres] para todas las asignaturas
                                     ▷ Límites Inferiores
12:  while cola no esté vacía do
13:    actual ← extraer el primer elemento de cola
14:    semestre_inicio, semestre_fin ← rangos[actual]
15:    for all dependiente en grafo[actual] do
16:      nuevo_semestre_inicio ← semestre_inicio + 1
17:      rangos[dependiente][0] ← máx(rangos[dependiente][0], nuevo_semestre_inicio)
18:      Decrementar grado_entrada[dependiente] en 1
19:      if grado_entrada[dependiente] == 0 then
20:        Añadir dependiente a cola
21:      end if
22:    end for
23:  end while
                                     ▷ Límites Superiores
24:  Crear cola con asignaturas sin dependencias
25:  while cola no esté vacía do
26:    actual ← extraer el primer elemento de cola
27:    semestre_inicio, semestre_fin ← rangos[actual]
28:    for all prereq en prereq_grafo[actual] do
29:      nuevo_semestre_fin ← semestre_fin - 1
30:      rangos[prereq][1] ← mín(rangos[prereq][1], nuevo_semestre_fin)
31:      Eliminar actual de grafo[prereq]
32:      if grafo[prereq] esté vacío then
33:        Añadir prereq a cola
34:      end if
35:    end for
36:  end while
37:  return rangos
38: end function
```

A partir de la obtención de los rangos, se puede comprender la lógica asociada al trabajo de

la solución inicial, esta se puede subdividir en tres partes:

- Ciclo principal - Elección de asignaturas y semestres
- Verificación de restricciones
- Actualización de rangos

3.4.1. Ciclo principal - Elección de asignaturas y semestres

El ciclo principal se encarga de generar la solución inicial a través del uso de parámetros iniciales y verificaciones constantes. Para esto se utilizan los rangos definidos previamente, además de considerar la existencia de dos preguntas que se deben responder:

- ¿Qué asignaturas se deberían priorizar en la asignación de la malla académica?
- ¿Qué semestre debería elegirse para asignar el curso a la malla curricular?

Para lo siguiente debemos recordar que se consideran las restricciones iniciales para la creación de la solución inicial, esto implica:

- Todas las asignaturas deben estar asignadas a la solución inicial (**Dura**).
- La cantidad de créditos semestrales no debe superar el máximo de créditos semestrales propuestos (Siempre y cuando sea viable) (**Blanda**) .
- La cantidad de créditos semestrales debe ser a lo menos equivalente al mínimo de créditos semestrales propuestos (Siempre y cuando sea viable) (**Blanda**) .
- Todas las asignaturas deben respetar su orden de precedencia y posterioridad (Prerrequisitos) (**Dura**) .

La respuesta a estas preguntas la entregarán los métodos o las heurísticas que mejor resultados otorguen (tanto en valor de solución como distribución de asignaturas), por lo cual para ambas preguntas se trabajó una serie de posibilidades como: Elección completamente aleatoria, Parcialmente aleatoria, Heurística basada en el método de *NorthWest*, Maximización de rangos de asignaturas posteriores, Priorización de asignaturas con mayor cantidad de créditos, Asignación de asignaturas directamente posterior a la seleccionada.

Los mejores resultados se obtuvieron a través de un método híbrido para resolver ambas preguntas, en particular:

Priorizar la asignaturas con el menor rango: Se considerará primero las asignaturas cuyos rangos sean más estrictos, esto significa, menos “flexibilidad” frente a las posibles ubicaciones que puede tomar la asignatura en la malla curricular, de esta forma se está disminuyendo considerablemente la probabilidad de generar rangos vacíos (soluciones no viables) en asignaturas dependientes a lo largo de la ejecución del algoritmo, esto significa por ejemplo, llenar un semestre con asignaturas “flexibilidad” para luego no tener espacio dentro de este para asignar un curso que solo puede ir en ese semestre.

Priorizar el semestre con menos carga académica: Al momento de la asignación se considerará como primera alternativa siempre el semestre cuya carga académica sea menor de entre todos los rangos viables de la asignatura, de esta forma se distribuye mucho mejor la carga académica en los semestres y se considera de manera implícita la restricción de mínima carga académica semestral.

A continuación se presenta el pseudocódigo del algoritmo encargado de la creación de la solución inicial.

Algorithm 2 Ciclo Principal

```
function Ciclo Principal(Máx créditos, Semestres, Asignaturas)
2:   Calcular el total de créditos.
      Calcular el promedio de carga semestral.
4:   Inicializar un vector de créditos semestrales VCS [].
      Inicializar un vector semestre [].
6:   Inicializar asignaturas aprobadas [].
      Rangos ← ordenar por tamaño CalcularRangos(asignaturas, prerequisites, total de
      semestres)
8:   while asignaturas != asignaturas aprobadas do
      for all asignatura en rangos do
10:      if asignatura existe en asignaturas aprobadas then
          Sigüente asignatura.
12:      end if
          Semestre propuesto ← min(rango(asignatura) en VCS)
14:      Semestre elegido ← VerificarRestricciones(asignatura, VCS, créditos[asignatura],
          rango[asignatura], semestre propuesto)
          Actualizo el vector semestre [ind. asignatura] con el semestre elegido.
16:      Actualizo el VCS [ind. semestre] con créditos[asignatura].
          Añado al vector asignaturas aprobadas[asignatura]
18:      end for
      end while
20:   return Vector semestre, VCS
end function
```

3.4.2. Verificación de Restricciones

Verificación de restricciones se encarga de comprobar que la selección del semestre propuesto no infrinja el límite máximo de créditos semestrales, además de asegurar la reelección de un semestre (en lo posible) en el caso que el propuesto vuelva vacío alguno de los rangos de sus asignaturas dependientes (previa o posterior), en el caso que ningún semestre pueda ser elegido directamente, se elegirá el semestre con menor carga siempre que no provoque vacíos, trabajando con la restricción de máxima cantidad de créditos semestral violada. De no poder elegir ningún semestre (dado que todas las opciones violan la restricción de precedencia), se acaba la ejecución y es necesario un *re-start* con un punto de inicio(asignatura A asignada a semestre B) distinto.

Algorithm 3 VerificarRestricciones

```
function VerificarRestricciones(asignatura, VCS, creditos[asignatura], rango[asignatura],  
semestre propuesto)  
    rango temporal  $\leftarrow$  rango[asignatura].  
3:   Inicializar un vector min carga [].  
    while rango temporal no esté vacío do  
        if creditos[asignatura] + VCS[asignatura]  $\leq$  30 then  
6:         if Actualizar rangos(asignatura, semestre propuesto) then  
            return semestre propuesto.  
        else  
9:         Eliminar el semestre propuesto de rango temporal  
            Semestre propuesto  $\leftarrow$  random(rango temporal)  
            Añadir al vector min carga  $\leftarrow$  carga, semestre  
12:        end if  
    end if  
    if creditos[asignatura] + VCS[asignatura]  $>$  30 then  
15:        Eliminar el semestre propuesto de rango temporal  
            Semestre propuesto  $\leftarrow$  random(rango temporal)  
            Añadir al vector min carga  $\leftarrow$  carga, semestre  
18:        end if  
    end while ▷ En caso todos los semestres estén llenos  
    for all carga, semestre in sorted(min carga) do  
21:        if Actualizar rangos(asignatura, semestre) then  
            return semestre.  
        end if  
24:    end for  
    return No hay solución.  
end function
```

3.4.3. Actualización de rangos

Actualización de rangos se encarga de verificar la integridad de los rangos, comprobando que la selección de semestre no provoque un vacío en las asignaturas dependientes, de ser positiva la respuesta, se actualizan los rangos de todas las asignaturas dependientes.

Algorithm 4 Actualización de rangos

```
function ActualizarRangos(asignatura, semestre propuesto)
  Inicializar vector rangos temporal [].
  for all Prerrequisitos do
4:    if asignatura tiene prerrequisito then
      Agregar a rangos temporal  $\leftarrow$  rangos a actualizar.
      if No se agregaron elementos a rangos temporales then
        return 0
8:    end if
      end if
      if asignatura es prerrequisito then
        Agregar a rangos temporal  $\leftarrow$  rangos a actualizar.
12:    if No se agregaron elementos a rangos temporales then
      return 0
      end if
    end if
16:  end for
    for all rangos in rangos temporales do
      Actualizar el vector de rangos originales
    end for
20:  return 1
  end function
```

3.4.4. Diagrama de flujo - Solución Inicial

A continuación se presenta un diagrama de flujo el cual ejemplifica la interacción entre las partes desarrolladas anteriormente.

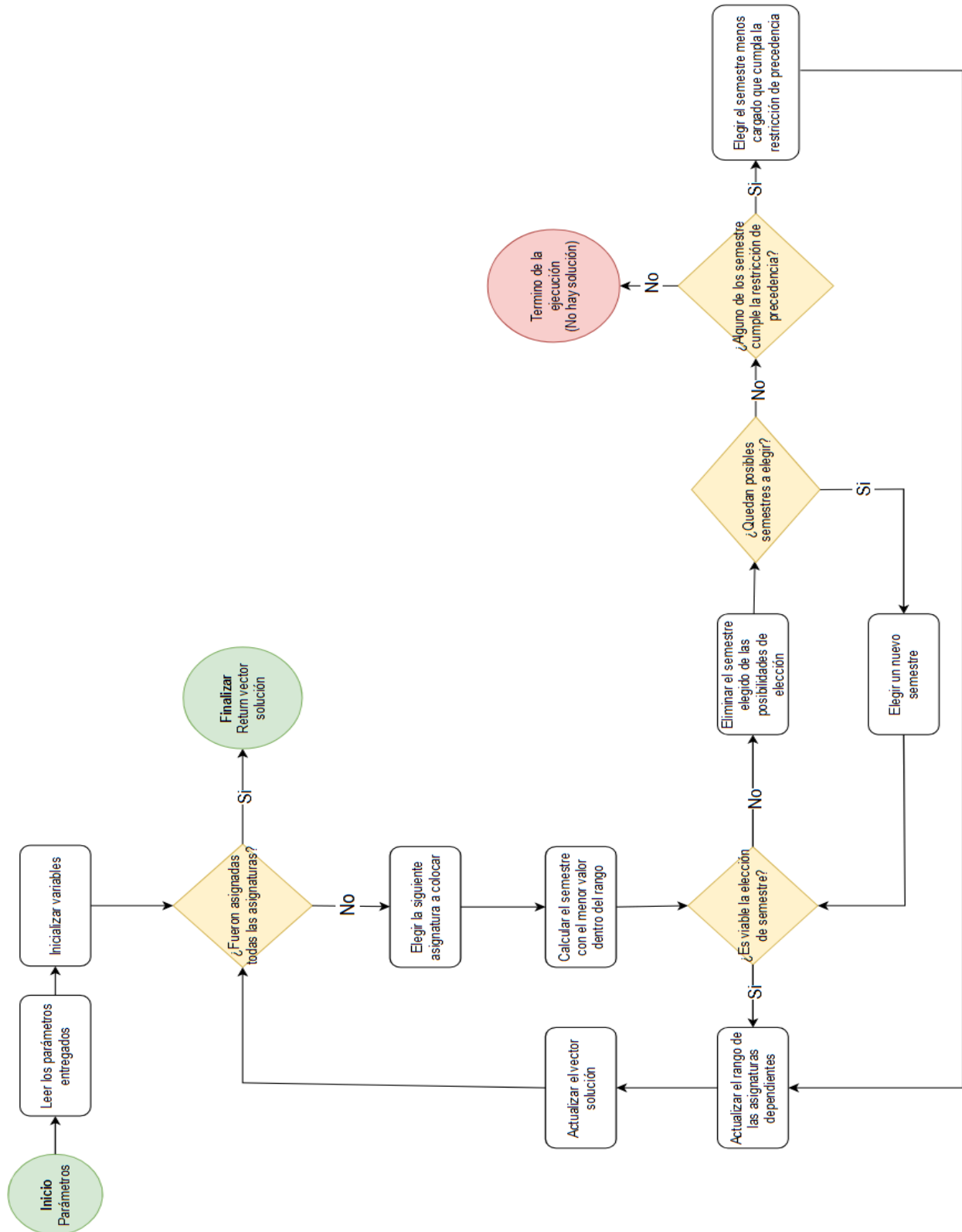


Figura 12: Diagrama de flujo - Solución inicial
Fuente: Elaboración propia

La principal ventaja que presenta Tabu Search en la problemática del BACP es el manejo de la lista tabú para considerar soluciones simétricas y ciclos.

3.5. Soluciones simétricas y óptimos locales: Ejemplos

Para explicar lo que se entiende en este trabajo por solución simétricas se ilustra con el siguiente ejemplo.

Se tiene la siguiente propuesta de solución:

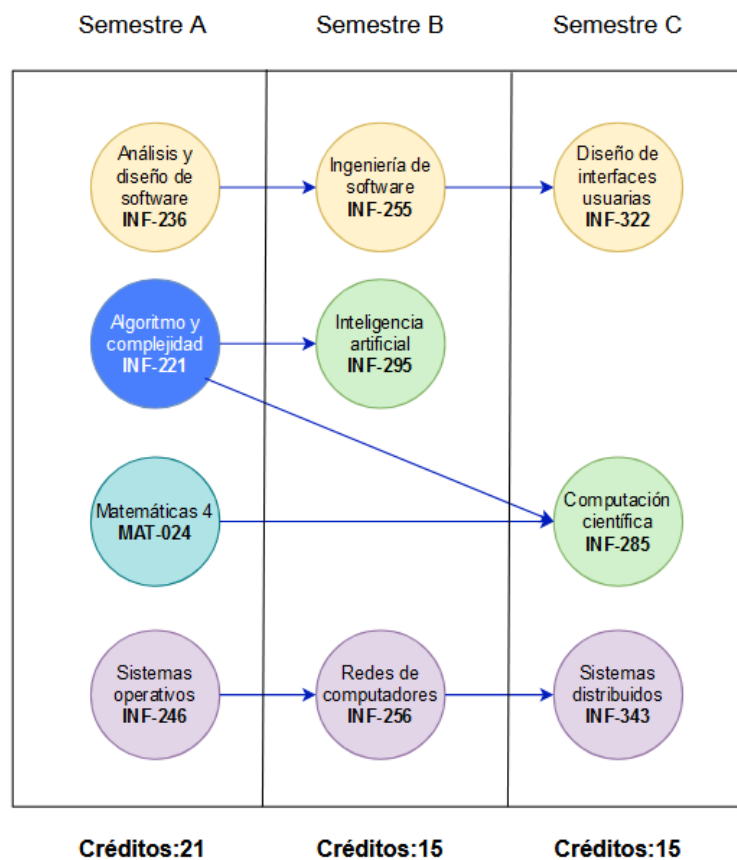


Figura 13: Solución en tres semestres con créditos

Fuente: Elaboración propia

En la figura anterior se presentan tres semestres : A, B, C. Cada uno de estos semestres tiene sus propias asignaturas, dependencias y una cantidad de créditos fija, cada semestre tiene entonces un total de créditos. Si se supone que se tiene en una instancia α considerada la mejor solución local en su vecindario, esto se ve representado en el orden de las asignaturas,

lo cual indica la cantidad de créditos semestrales, lo que se esperaría entonces es buscar una mejora a partir de esta instancia α , sin embargo, si se asume que se está en un óptimo local la mejor solución a partir de esta tendría a lo más el mismo valor que la presentada actualmente.

Se propone entonces el siguiente movimiento que representa el mismo valor de solución:

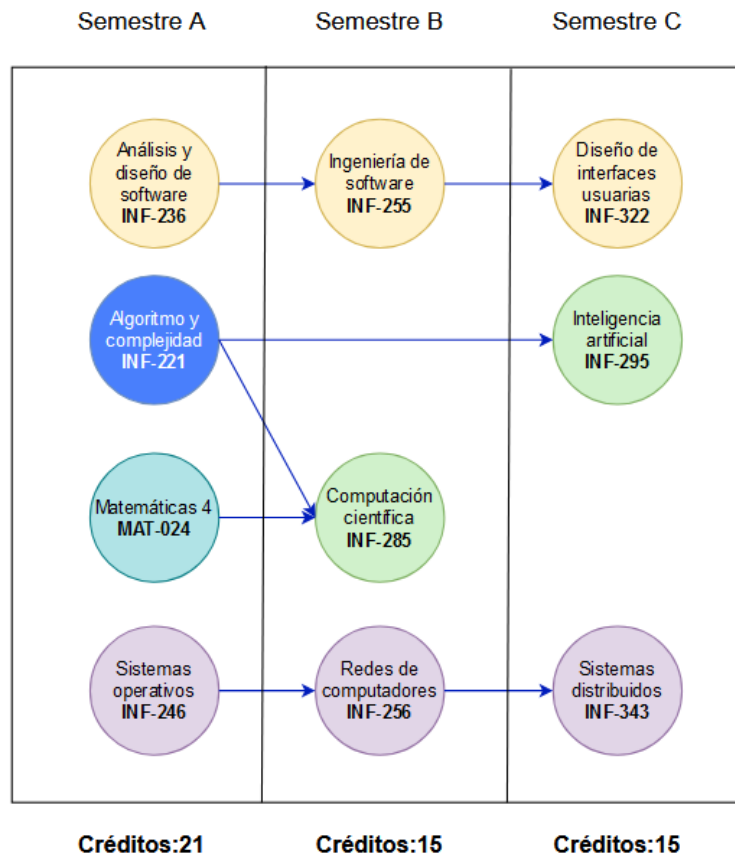


Figura 14: Solución en tres semestres con créditos

Fuente: Elaboración propia

Como se puede apreciar en la figura anterior, las asignaturas **INF-285** e **INF-295** han sido intercambiadas de semestres, sin afectar las restricciones propuestas y manteniendo el mismo valor semestral, y por ende, el mismo valor de solución general.

Lo que se acaba de construir entonces es lo que se refiere en este documento a *soluciones simétricas* las cuales dada la naturaleza del problema se generan en gran medida una vez encontrado un óptimo local (esto no significa que no puedan encontrarse previo al óptimo local) dada la cantidad de posibles asignaturas intercambiables entre semestres que mantienen un valor óptimo (asignaturas libres, electivos, asignaturas sin prerequisites).

3.6. Ventajas de la Búsqueda Tabú

Para resolver el problema anteriormente planteado es necesario forzar la exploración de la solución local al punto que puede generar nuevos valores alterando el orden las asignaturas dentro de las restricciones planteadas, para esto existen múltiples planteamientos viables (ver sección 2.2), a continuación se describirán los principales motivos para elegir Búsqueda Tabú:

Lista tabú y ciclos El principal método por el cual la búsqueda tabú opera es a través de la lista tabú. La lista tabú permite el almacenamiento de movimientos previos por parte del algoritmo otorgándole "memoria", esta memoria permite discernir que movimientos han sido realizados dentro de un rango temporal definido (conocido como el tamaño de la lista tabú) y cuales no, obligando al algoritmo a elegir rutas no exploradas (en caso de ser posible).

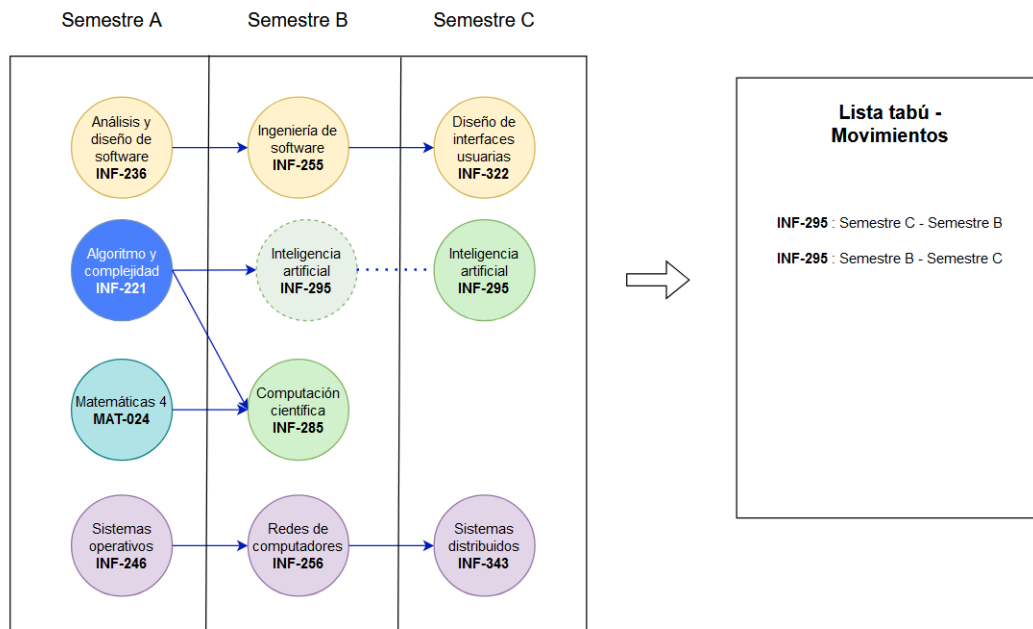


Figura 15: Ejemplo de lista tabu 1
Fuente: Elaboración propia

En la figura anterior se tiene el caso hipotético del siguiente movimiento del algoritmo, este se representa como **INF-295** del semestre B al C, esto significa que ese movimiento en particular será agregado a la lista tabú, y por consiguiente, prohibido durante una cierta cantidad de iteraciones

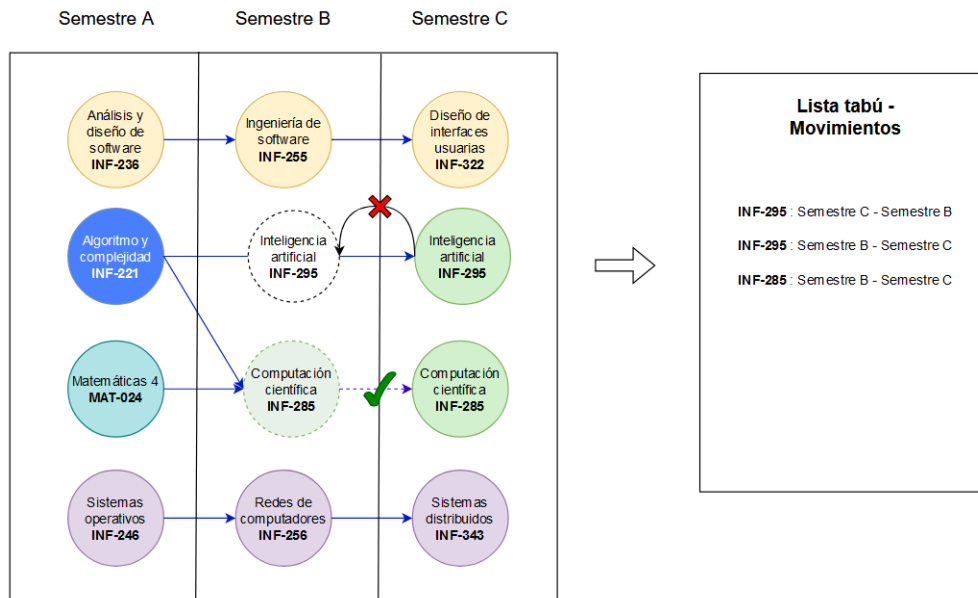


Figura 16: Ejemplo de lista tabú 2
Fuente: Elaboración propia

Si se avanza a la siguiente iteración se observa que el único movimiento viable dada las restricciones, es **INF-285** del semestre B al C, dado que en la iteración anterior fue prohibido **INF-295** del semestre B al C, este movimiento no puede volver a elegirse, así se evitan ciclos y soluciones simétricas dentro de la explotación y forzamos al algoritmo a explorar nuevas posibles soluciones, volviendo así el tamaño de la lista tabú el principal parámetro de flexibilidad.

3.7. Elección de movimiento

Una de las piezas fundamentales de las metaheurísticas son los operadores de búsqueda, estos definen el como se generará una nueva solución viable a partir de la solución actual, en el caso de la búsqueda tabú, este espacio de soluciones se conoce como vecindario y el operador elegido en este caso, un movimiento local.

El movimiento consiste en buscar el semestre más cargado académicamente, obtener todos los rangos de las asignaturas pertenecientes y desplazar la asignatura α al semestre β que menos impacte la solución final (función de evaluación), en caso de no existir movimientos posibles dentro del semestre(todos los rangos son únicamente el semestre en cuestión), se elegirá el siguiente semestre con mayor carga y así sucesivamente.

A continuación se presenta un ejemplo basado en una instancia presentada en la sección 3.5

(ver figura 14).

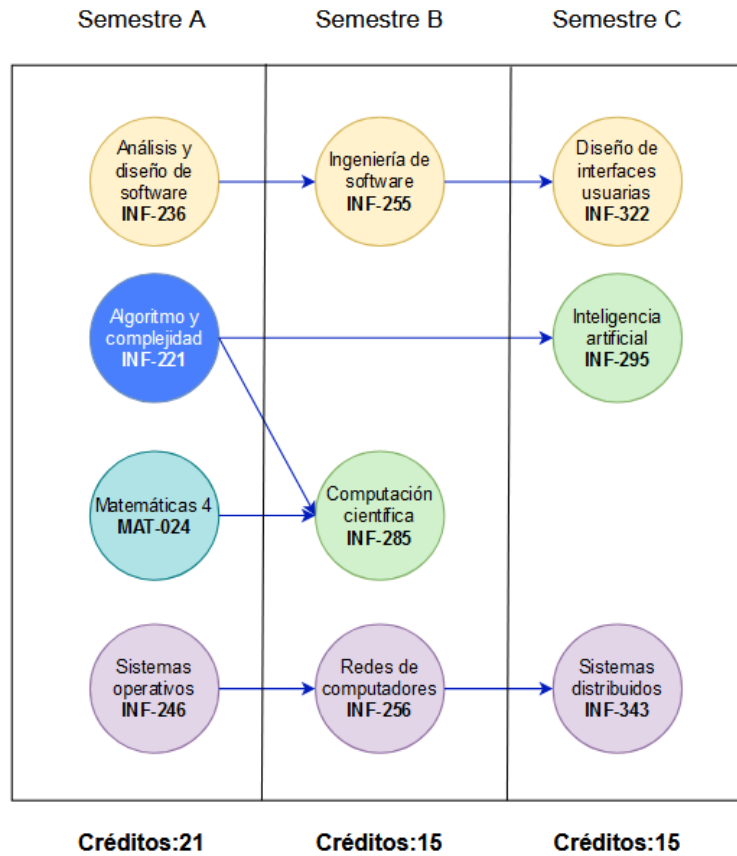


Figura 17: Ejemplo de movimiento 1

Fuente: Elaboración propia

En este pequeño ejemplo, el semestre A es aquel con la mayor cantidad de créditos, por lo que es el semestre candidato a mover una asignatura, dado que la única asignatura con un rango flexible es INF-221 [SEMESTRE A, SEMESTRE B], esta es considerada automática como el movimiento por defecto (asumiendo claro, el movimiento no está en la lista tabú).

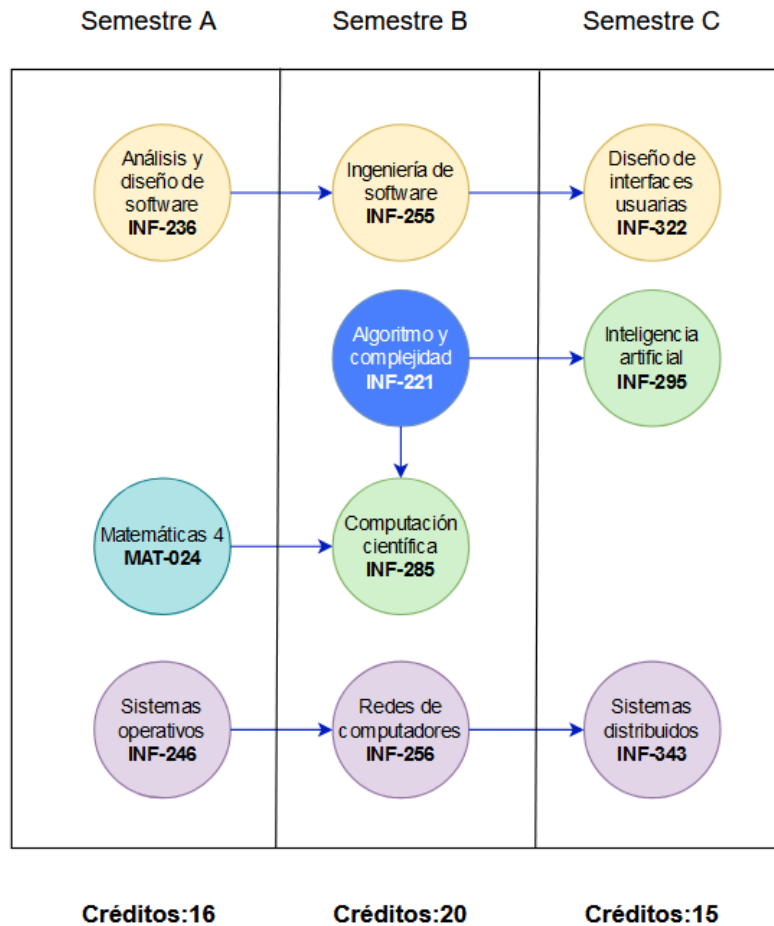


Figura 18: Ejemplo de movimiento 2

Fuente: Elaboración propia

Como se puede apreciar en la figura anterior, INF-221 ha sido desplazada entre semestres y actualizado la cantidad de créditos semestrales reflejado en la suma total de créditos semestral. Este movimiento será definido entonces como :

$$INF - 221 : \{A \rightarrow B\}$$

La lógica detrás del movimiento es simple pero efectiva, mover una asignatura del semestre más pesado al menos pesado (accesible) busca balancear la distribución de créditos globales. El mayor inconveniente que puede encontrar este algoritmo es caer en un bucle o una solución simétrica la cual evite la exploración de nuevas posibles soluciones (para esto es necesario mover las suficientes asignaturas hasta llegar a los prerrequisitos base), y es exactamente este el motivo por el cual la búsqueda tabú tiene un excelente desempeño en esta problemática. La lista tabú hace un buen trabajo lidiando con este problema; sin embargo,

es necesario destacar que la exploración que aporta la lista tabú está directamente ligada al largo de la lista y este es un parámetro (en el caso de búsqueda tabú normal) estático y cuyo valor óptimo debe ser obtenido a través de distintos tipos de testeos, en otras palabras, un hiperparámetro.

Algorithm 5 Movimiento

```
function Movimiento(SemestreCargado)
  Inicializar Posibles movimientos []
  for all Asignaturas en SemestreCargado do
    for all rangos in asignaturas do
5:       if semestre(asignatura) es igual a SemestreCargado then
          Sigiente rango.
        end if
        Añadir Posibles movimientos  $\leftarrow$  [asignatura, [semetre actual, semestre a mo-
ver]]
    end for
10:  end for
    return min(ValidarValorSolucion(posibles movimientos))
end function
```

3.8. Programa Propuesto

La elaboración de un programa propone la aplicación de la resolución del BACP estudiantil de tal manera que los estudiantes de ing. civil informática puedan utilizarlo de manera personal, por lo tanto es necesario que cumpla con ciertos requisitos propuestos en este documento:

- Facil de utilizar, intuitivo.
- Accessible.
- Respuesta dinámica en base al input del usuario.
- Entregar la solución correcta en caso de existir.
- Resultado entendible por el usuario.
- Tener un tiempo de ejecución razonable.

Estos son los puntos principales en consideración. Afortunadamente existe una herramienta web diseñada y desarrollada por alumnos y exalumnos de ing. civil informática de la UTFSM, la cual cumple con varios de estos requisitos y servirá de base para la implementación del BACP, esta herramienta siendo; La malla interactiva ing. civil Informática.

3.8.1. Malla interactiva

La Malla Interactiva [Aedo, 2018] es un proyecto originalmente desarrollado por Sebastián Aedo (actualmente mantenido en un repositorio de forma pública [Paulangelo, 2024]), el cual busca presentar la malla académica de ing. civil informática (y actualmente de otras carreras) de forma interactiva, presentando información como créditos, asignaturas, códigos, prerequisites y por supuesto, la opción de permitir que el usuario clickee asignaturas para simular manualmente su avance por la carrera de informática, para de esta forma, visualizar qué asignaturas le quedan por cursar, cuáles tiene disponible directamente para inscribir en un semestre futuro y cuántas lleva aprobadas del total de asignaturas (además de otras funcionalidades más específicas).

DESARROLLO DE SOFTWARE BASADO EN METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE
BALANCEO DE MALLAS CURRICULARES DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL ACADÉMICO
PARA ALUMNOS DE INFORMÁTICA DE LA UTFSM CC

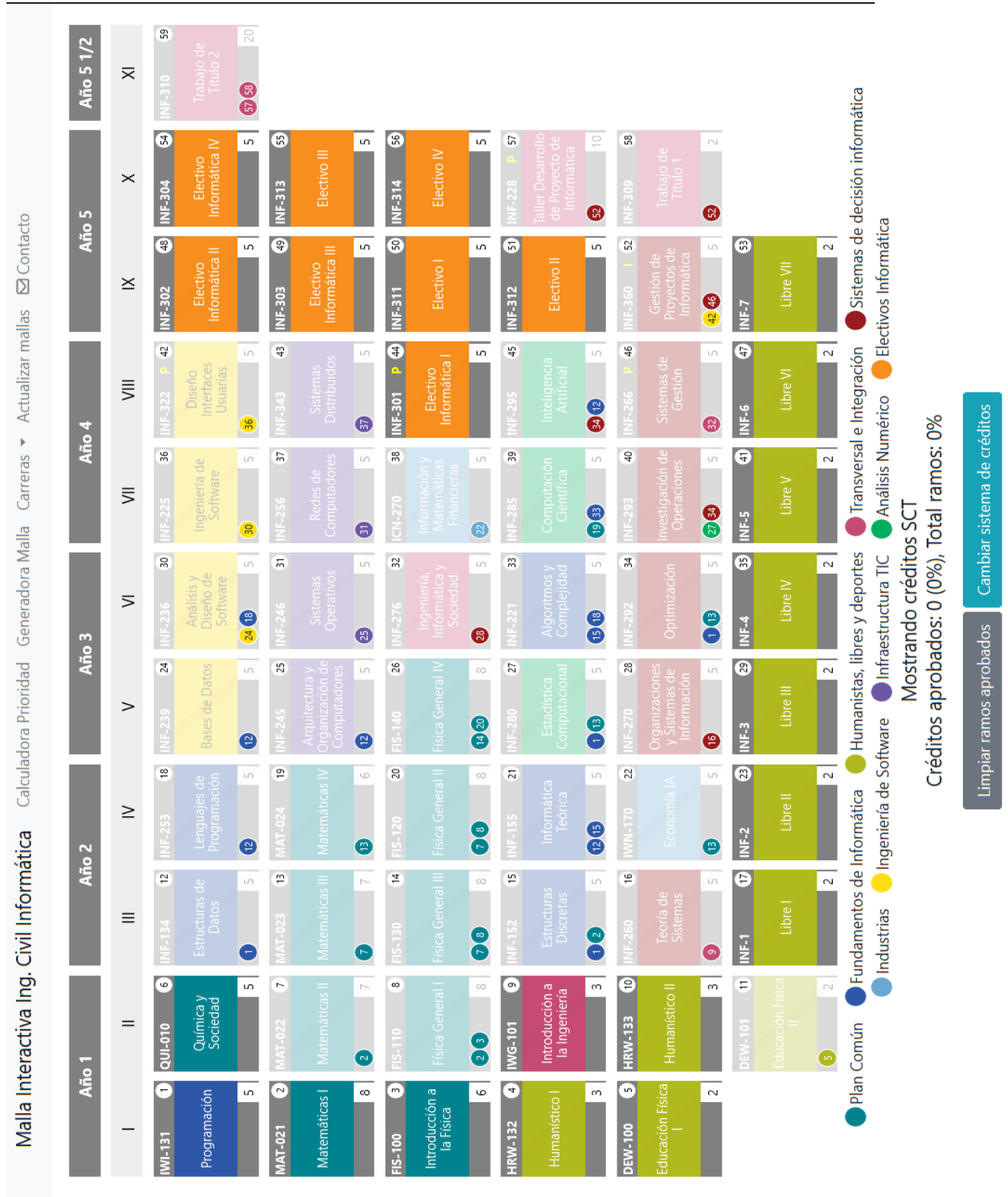


Figura 19: Malla Interactiva Original
Fuente: [Aedo, 2018]

DESARROLLO DE SOFTWARE BASADO EN METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE
BALANCEO DE MALLAS CURRICULARES DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL ACADÉMICO
PARA ALUMNOS DE INFORMÁTICA DE LA UTFSM CC

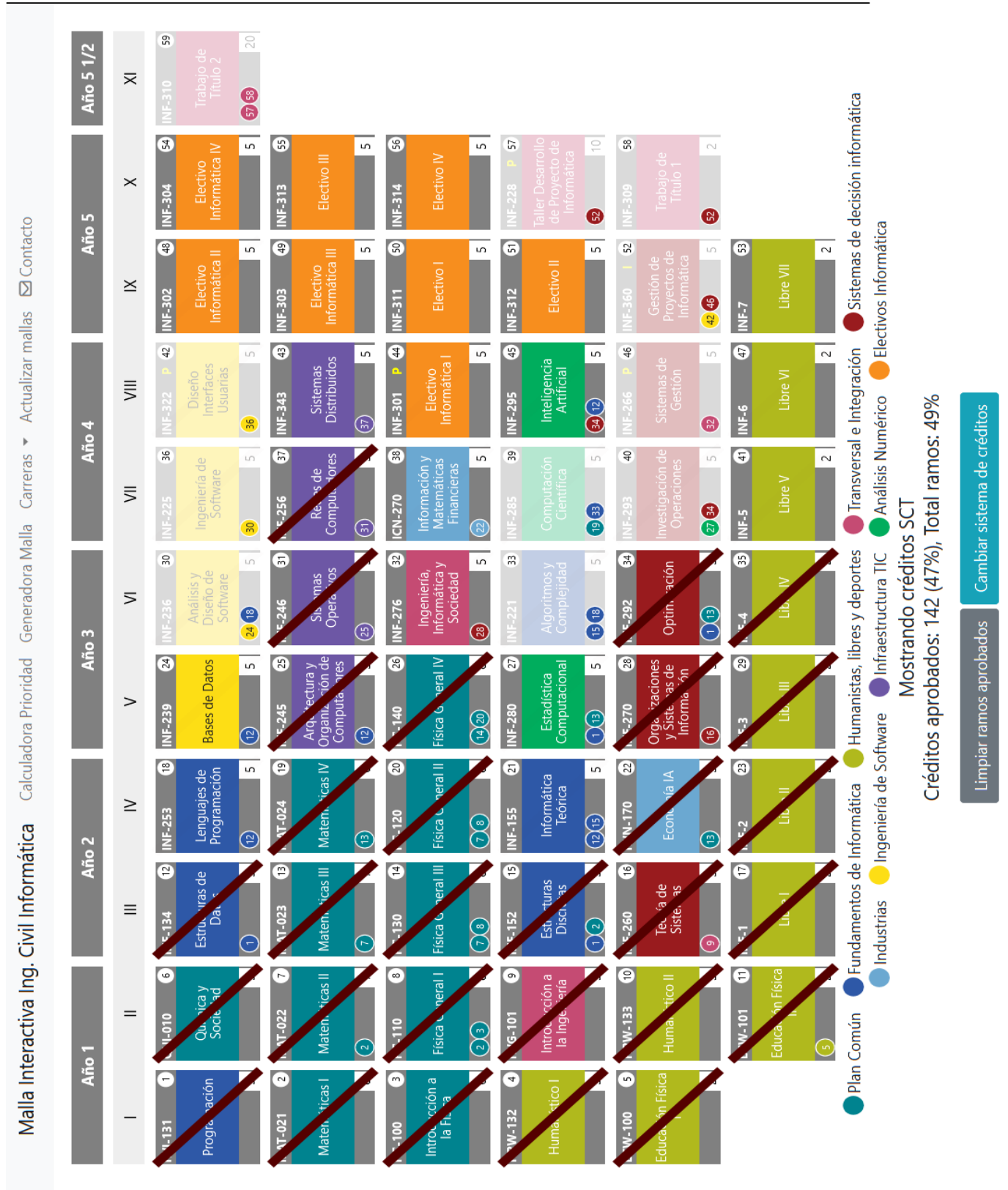


Figura 20: Malla Interactiva: Selección de asignaturas aprobadas
Fuente: [Aedo, 2018]

3.8.2. Desarrollo e Integración

La idea entonces consiste en crear un método el cual interactué con la malla interactiva de tal forma que pueda aplicar la resolución del BACP acomodado al input del usuario (ver figura 20). Para esto es necesario transformar el algoritmo de resolución en un *endpoint* de tal forma que este pueda recibir la malla curricular que se desea optimizar (además de los parámetros necesarios) y este modifique la visualización de la interfaz de la malla interactiva, de tal forma que el usuario mantenga el entendimiento de cómo interactuar con esta y pueda reconocer e interpretar fácilmente la solución propuesta.

Para este desarrollo se utilizó Flask 3.10⁴ para la integración y levantamiento de endpoints en Python 3.12⁵

El *endpoint* se encarga de recibir los datos y generar el semestre solución correspondiente utilizando el mismo formato de JSON⁶ necesario para la visualización en la malla interactiva. En esta estructura, cada semestre tiene un serie de asignaturas asociada las cuales se definen de la siguiente forma:

1. **Ramo:** El nombre completo del ramo.
2. **Sigla:** Sigla del ramo. **Única** para cada ramo.
3. **Créditos USM:** Entero, la cantidad de créditos USM.
4. **Créditos SCT:** Entero, la cantidad de créditos SCT.
5. **Categoría:** Referente al color asignado en la malla interactiva.
6. **Prerrequisitos:** Una lista de strings que contiene las siglas de los prerrequisitos del ramo.
7. **Indicador Par o Impar:** Indica si el ramo se dicta en un semestre Par, Impar o en Ambos.

A continuación una figura que representa una asignación para un segundo semestre ("s2"), con las asignaturas: "QUI-010", "MAT-022", "FIS-110", "IWG-101", "HRW-133", "DEW-101".

⁴Flask es un framework ligero para el desarrollo de aplicaciones web WSGI. <https://pypi.org/project/Flask/>

⁵Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. <https://www.python.org/doc/>

⁶JavaScript Object Notation (JSON) es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript. https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/JSON

```
{
  "s2": [
    ["Química y Sociedad", "QUI-010", 3, 5, "PC", [], "A"],
    ["Matemáticas II", "MAT-022", 5, 7, "PC", ["MAT-021"], "A"],
    ["Física General I", "FIS-110", 5, 8, "PC", ["MAT-021", "FIS-100"], "A"],
    ["Introducción a la Ingeniería", "IWG-101", 2, 3, "TIN", [], "A"],
    ["Humanístico II", "HRW-133", 2, 3, "HUM", [], "A"],
    ["Educación Física II", "DEW-101", 1, 0, "HUM", ["DEW-100"], "A"]
  ],
  // ...
}
```

Figura 21: Ejemplo de formato JSON para un semestre particular

Fuente: [Paulangelo, 2024], Extracto de Documentación: 4.¿Cómo funciona la malla?

Parámetros Es necesario que el usuario pueda entregar al software dos parámetros fundamentales en la ejecución del algoritmo, estos parámetros son utilizados para la verificación de restricciones inmediatas.

- Número de semestres a cursar.
- Número máximo de créditos semestrales.

En la sección 3.1 se definieron las restricciones necesarias del algoritmo además de un factor *AVG* el cual representa el promedio crudo de distribución académica entre el número de semestres y el total de créditos, de esta forma:

- Si $AVG < \text{Mínima carga académica semestral}$: La distribución óptima no permite que el estudiante sea alumno regular durante uno o más semestres, esto no puede considerarse una solución viable (inicialmente), y es necesario disminuir la cantidad de semestres o aumentar la cantidad de asignaturas.
- Si $AVG > \text{Máxima carga académica semestral}$: La distribución óptima esta por encima del máximo permitido por la institución, lo cual significa que existe demasiada carga académica y es necesario distribuirla entre más semestres o disminuir la cantidad de asignaturas.

Esto produce un diagrama de flujo con la siguiente estructura:

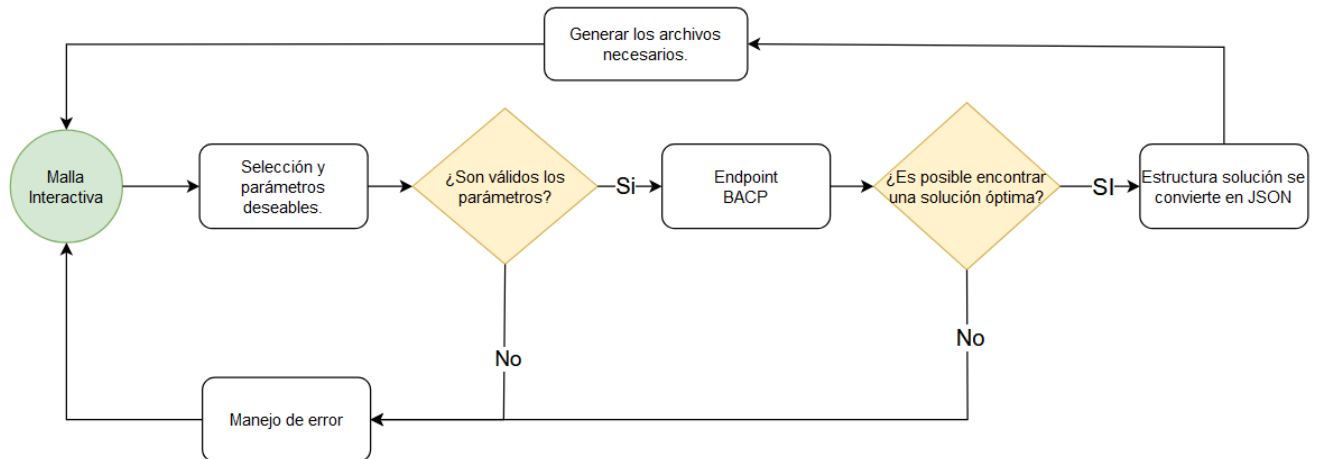


Figura 22: Diagrama de flujo, Malla Interactiva - BACP
Fuente: Elaboración propia

3.8.3. Interfaz usuaria - Ejemplos

En la siguiente sección se interactuará con la malla interactiva base (ver figura19) con distintos inputs y parámetros con la finalidad de visualizar el cambio en la malla curricular y su reestructuración dinámica.

DESARROLLO DE SOFTWARE BASADO EN METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE
BALANCEO DE MALLAS CURRICULARES DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL ACADÉMICO
PARA ALUMNOS DE INFORMÁTICA DE LA UTFSM CC

Malla Interactiva BACP

I	II	III	IV	V	VI	VII	VIII	IX	X	XI
INF-131 Programación 5	QUI-010 Química y Sociedad 5	HRW-133 Humanístico 2 3	FIS-110 Física General 1 2 3 8	FIS-130 Física General 3 9 19 8	FIS-120 Física General 2 9 19 8	FIS-140 Física General 4 25 30 8	HRW-132 Humanístico 1 3	ICH-270 Información y Matemáticas Francésas 26 5	INF-256 Redes de Computadores 36 5	INF-5 Libre 5 57 2
MAF-021 Matemáticas 1 8	MAF-022 Matemáticas 2 2 7	MAF-023 Matemáticas 3 9 7	MAF-024 Matemáticas 4 12 6	IRW-170 Economía 1A 14 5	INF-245 Arquitectura y Organización de Computadores 10 5	INF-246 Sistemas Operativos 31 5	DEW-101 Educación física 2 2	INF-293 Investigación de Operaciones 27 42 6	INF-285 Computación científica 14 37 5	INF-343 Sistemas Distribuidos 38 5
FIS-100 Introducción a la física 6	INF-134 Estructura de Datos 1 5	INF-253 Lenguajes de Programación 10 5	INF-2 Libre 2 2 14 5	INF-280 Estadística Computacional 1 14 5	INF-225 Ingeniería de Software 28 5	INF-221 Algoritmos y Complejidad 11 15 5	INF-292 Optimización 14 5	INF-304 Electivo Informática 4 37 42 5	INF-295 Inteligencia Artificial 37 42 5	INF-310 Trabajo de Título 2 50 56 20
DEW-100 Educación física 1 2	INF-152 Estructuras Discretas 1 2 5	INF-155 Informática Teórica 10 11 5	INF-239 Bases de datos 10 5	INF-236 Análisis y Diseño de Software 15 22 5	INF-303 Electivo Informática 3 5	INF-322 Diseño de Interfaces Usuarios 32 5	INF-4 Libre 4 40 2	INF-228 Taller de Desarrollo de Proyectos de Informática 46 11	INF-302 Electivo Informática 2 5	
INF-101 Introducción a la Ingeniería 3	INF-260 Teoría de Sistemas 5 5	INF-270 Organizaciones y Sistemas de Información 17 5	INF-276 Ingeniería Informática y Sociedad 17 5	INF-266 Sistemas de Gestión 23 5	INF-314 Electivo 4 5	INF-311 Electivo 1 5	INF-301 Electivo Informática 1 5		INF-313 Electivo 3 5	
INF-1 Libre 1 2		INF-3 Libre 3 2	INF-7 Libre 7 2				INF-312 Electivo 2 5		INF-309 Trabajo de Título 1 46 2	
INF-6 Libre 6 2							INF-360 Gestión de Proyectos Informáticos 33 29 5			

● Plan Común ● Fundamentos de Informática ● Humanistas, libres y deportes ● Transversal e Integración ● Electivos Informática ● Sistemas de decisión informática ● Industrias ● Ingeniería de Software ● Infraestructura TIC ● Análisis Numérico

Créditos aprobados: 0 (0%), Total ramos: 0%

Figura 23: BACP - 59 asignaturas, 11 semestre, 30 créditos máx.

Fuente: Elaboración propia

En la figura anterior se consideró la malla curricular completa del departamento de informática(2023) con exactamente los mismos parámetros que presenta la UTFSM, generando una malla interactiva BACP con la solución obtenida. A partir de esta solución es posible interactuar si se desea para cambiar los parámetros o, como en las siguientes figuras, generar una instancia nueva con asignaturas aprobadas y nuevos parámetros.

Malla Interactiva BACP



Plan Común ● Fundamentos de Informática ● Humanistas, libres y deportes ● Transversal e Integración ● Sistemas de decisión informática ● Industrias ● Ingeniería de Software ● Infraestructura TIC ● Análisis Numérico ● Electivos Informática

Créditos aprobados: 107 (35%), Total ramos: 38%

Figura 24: BACP - Selección dinámica de asignaturas aprobadas en base a la malla BACP
Fuente: Elaboración propia

Malla Interactiva BACP

I	II	III	IV	V	VI	VII
INF-253 Lenguajes de Programación 5	FIS-120 Física General 2 8	FIS-140 Física General 4 8	DEW-101 Educación física 2 2	HRW-132 Humanístico 1 3	INF-293 Investigación de Operaciones 6	INF-4 Libre 4 2
MAT-024 Matemáticas 4 6	INF-155 Informática Teórica 5	INF-225 Ingeniería de Software 5	INF-322 Diseño de Interfaces Usuarios 5	INF-280 Estadística Computacional 5	INF-311 Electivo 1 5	ICN-270 Información y Matemáticas Financieras 5
IWN-170 Economía 1A 5	INF-236 Análisis y Diseño de Software 5	INF-256 Redes de Computadores 5	INF-343 Sistemas Distribuidos 5	INF-301 Electivo Informática 1 5	INF-313 Electivo 3 5	INF-310 Trabajo de Título 2 20
INF-245 Arquitectura y Organización de Computadores 5	INF-246 Sistemas Operativos 5	INF-285 Computación científica 5	INF-302 Electivo Informática 2 5	INF-303 Electivo Informática 3 5	INF-228 Taller de Desarrollo de Proyecto de Informática 10	
INF-292 Optimización 5	INF-221 Algoritmos y Complejidad 5	INF-295 Inteligencia Artificial 5	INF-312 Electivo 2 5	INF-360 Gestión de Proyectos Informáticos 5	INF-309 Trabajo de Título 1 2	
INF-5 Libre 5 2			INF-314 Electivo 4 5	INF-304 Electivo Informática 4 5		

Figura 25: Malla solución resultante selección dinámica - 7 semestres, 35 créditos máx.

Fuente: Elaboración propia

Para este caso en particular la malla académica solución BACP (ver figura 24) presenta las asignaturas aprobadas por el estudiante, las cuales pueden ser representadas por los cursos tachados en la malla interactiva, esto hace un símil muy cercano al historial académico del estudiante luego de una cantidad fija de semestres. Posteriormente se ejecuta el BACP estudiantil en la malla interactiva y este reorganiza las asignaturas en base la cantidad de semestres y la máxima cantidad de créditos semestrales, generando una nueva malla BACP con la solución encontrada (ver figura 25), de esta forma, el estudiante puede iterar sobre su propia malla actualizándola en función de sus necesidades. Es importante destacar que las asignaturas independientes como los cursos libres, electivos y electivos de informática, son intercambiables entre sí mismos, dado que en primera instancia no presentan prerequisites y poseen una misma cantidad de créditos.

3.9. Propuesta de Solución: Conclusiones

En este capítulo se presentó la definición matemática del problema basada en la definición del modelo original propuesto por [Castro y Manzano, 2001], con cambios a la función objetivo y una visión más actual de las restricciones acorde a las reglas y estatutos de la UTFSM Casa Central 2024, centradas en la cantidad de créditos semestrales y la precedencia de las asignaturas.

La solución propuesta se basa en utilizar un método híbrido de heurísticas para obtener una solución inicial, la cual se intentará mejorar con la aplicación de *tabu search*, buscado la mejor distribución de créditos semestrales posibles. Esto se logra gracias al concepto de rangos (referirse a la sección 3.4), los cuales permiten determinar y delimitar todas las posibles asignaciones de cursos en los semestres académicos.

Las ventajas de utilizar *tabu search* radican en la lista tabú, la cual permite evitar ciclos y explorar considerando la obtención de soluciones simétricas (referirse a 3.5), logrando así explotar el espacio de búsqueda reducido por la asignación de rangos.

Finalmente se utilizó un herramienta de software basada en la funcionalidad de la malla interactiva de informática, la cual desde su creación se ha vuelto bastante utilizada por estudiantes de múltiples carreras. La herramienta permite la selección de asignaturas por parte del estudiante, imitando la toma de asignaturas y su historial académico, permitiéndole generar mallas solución en base a la resolución del BACP estudiantil de manera dinámica y acorde a sus necesidades. De esta forma, se espera que la facilidad en el ordenamiento equitativo de créditos semestrales ayude a los estudiantes en la toma de decisiones respecto a la organización de su malla académica y tenga un efecto positivo en su desarrollo universitario.

En el siguiente capítulo se presenta la validación de la solución.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

En el siguiente capítulo se pondrá a prueba la solución propuesta en base a una serie de instancias consideradas relevantes para la demostración de validez, además de instancias adicionales, las cuales reflejan mejor la malla curricular informática de la casa de estudios. Cada resultado será comparado y presentado de forma resumida para luego evaluar y comparar todos los resultados juntos.

4.1. Especificaciones del equipo de trabajo

Todas las instancias y pruebas realizadas a continuación se ejecutaron en un sistema con un procesador AMD Ryzen 7 5800H (16 CPUs, 3.2GHz), 16 GB de RAM DDR4 y una unidad SSD de 512 GB. El sistema operativo utilizado fue Windows 11 Home Single Language(64 bits). Para las implementaciones se utilizó Python 3.12 y Flask 3.10 para levantar los endpoints requeridos.

4.2. Instancias de prueba - Benchmarks

En la sección 2.2 se analizaron múltiples soluciones al presente problema, las cuales siempre demuestran su validez en base a la resolución de las *benchmarks* referidas en [Gent y Walsh, 1999]. Estas instancias presentan una serie de asignaturas definidas, sus correspondientes créditos académicos y su relación de precedencia, a continuación se listarán los parámetros en específico entregados por instancia (a partir de ahora, las instancias serán referidas por su cantidad de semestres y cantidad de asignaturas):

- 8 semestres, 46 asignaturas, 24 créditos máximo por semestres, 10 créditos mínimos por semestre, 2 asignaturas mínimo por semestre, 10 asignaturas máximo por semestres.
- 10 semestres, 42 asignaturas, 24 créditos máximo por semestres, 10 créditos mínimos por semestre, 2 asignaturas mínimo por semestre, 10 asignaturas máximo por semestres.
- 12 semestres, 66 asignaturas, 24 créditos máximo por semestres, 10 créditos mínimos por semestre, 2 asignaturas mínimo por semestre, 10 asignaturas máximo por semestres.

Se evaluarán las soluciones encontradas por el algoritmo inicial y luego la mejora proporcionada por la búsqueda tabú; el tamaño de la lista tabú (inicialmente) será de la mitad de la cantidad de asignaturas entregadas.

4.2.1. 8 semestres - 46 asignaturas

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	18	1.890625	
s2	17	0.140625	
s3	17	0.140625	
s4	16	0.390625	
s5	15	2.640625	
s6	18	1.890625	
s7	16	0.390625	
s8	16	0.390625	
TOTAL	133	7.875	3

Tabla 6: Resultados Solución inicial (CP): 8 semestres, 46 asignaturas

Fuente: Elaboración Propia.

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	16	0.390625	
s2	17	0.140625	
s3	17	0.140625	
s4	17	0.140625	
s5	17	0.140625	
s6	17	0.140625	
s7	16	0.390625	
s8	16	0.390625	
TOTAL	133	1.875	1

Tabla 7: Resultados Solución Final (TS): 8 semestres, 46 asignaturas

Fuente: Elaboración Propia.

Las tablas se componen de los semestres a cursar, la cantidad de créditos finales por semestre, el error absoluto cuadrático en cada semestre y la función de evaluación clásica (diferencia entre el semestre con más carga y el con menor carga). De esta forma, se logra una comparación directa entre el impacto que tiene búsqueda tabú en cada una de las instancias (además de observar si existe alguna notoriedad en algún semestre en particular).

N° de iteraciones vs error absoluto cuadrático

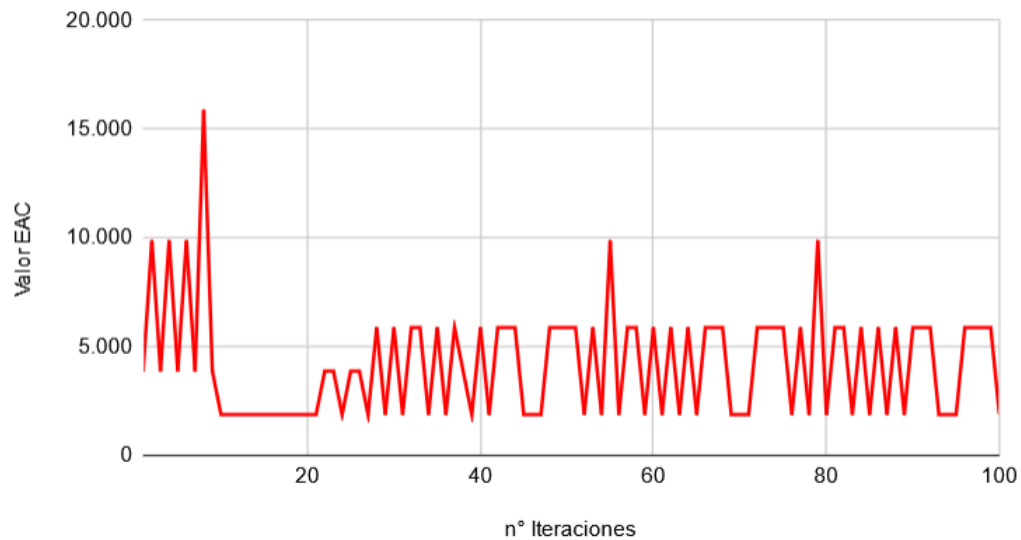


Figura 26: TS - Iteraciones vs EAC
Fuente: Elaboración propia

Como se puede apreciar en la figura anterior, TS encuentra la mejor solución cercana a la décima iteración con una lista tabú de 23 elementos.

4.2.2. 10 Semestres - 42 asignaturas

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	15	2.55999	
s2	15	2.55999	
s3	14	0.35999	
s4	14	0.35999	
s5	12	1.96000	
s6	12	1.96000	
s7	12	1.96000	
s8	16	6.75999	
s9	16	1.96000	
s10	12	1.96000	
TOTAL	134	22.4	4

Tabla 8: Resultados Solución inicial (CP): 10 semestres, 42 asignaturas
Fuente: Elaboración Propia.

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	13	0.16000	
s2	13	0.16000	
s3	13	0.16000	
s4	14	0.35999	
s5	13	0.16000	
s6	14	0.35999	
s7	14	0.35999	
s8	14	0.35999	
s9	13	0.16000	
s10	13	0.16000	
TOTAL	134	2.4	1

Tabla 9: Resultados Solución final (TS): 10 semestres, 42 asignaturas
Fuente: Elaboración Propia.

N° de iteraciones vs error absoluto cuadrático

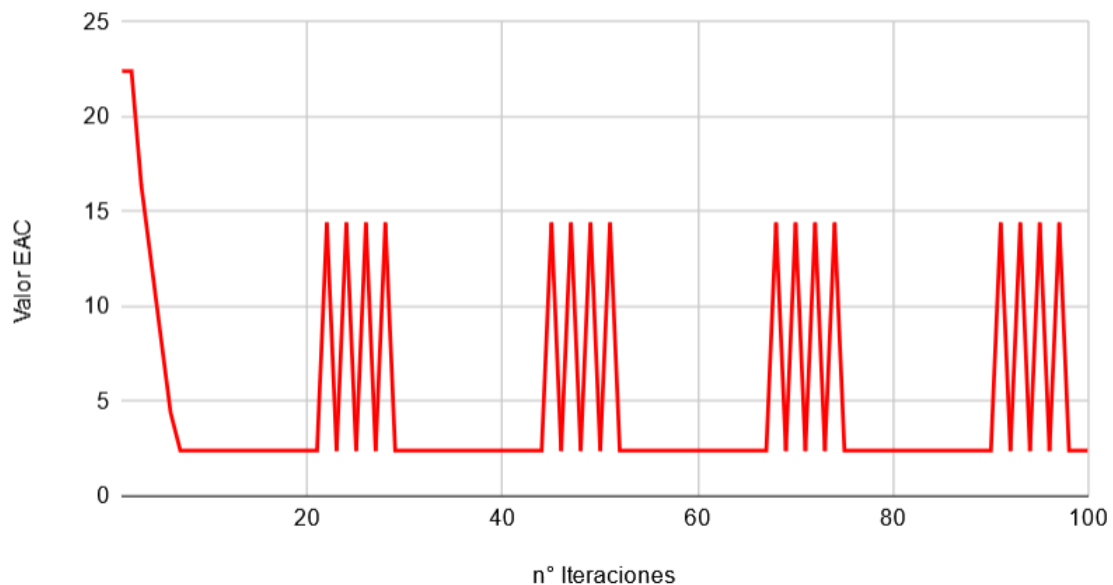


Figura 27: TS - Iteraciones vs EAC
Fuente: Elaboración propia

Como se puede apreciar en la figura anterior, TS encuentra la mejor solución casi al instante (7ª iteración) con una lista tabú de 21 elementos.

4.2.3. 12 Semestres - 66 asignaturas

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	16	1	
s2	17	0	
s3	18	1	
s4	19	4	
s5	17	0	
s6	17	0	
s7	19	4	
s8	15	4	
s9	15	4	
s10	17	0	
s11	17	0	
s12	17	0	
TOTAL	204	18	4

Tabla 10: Resultados Solución inicial (CP): 12 semestres, 66 asignaturas
Fuente: Elaboración Propia.

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	17	0	
s2	17	0	
s3	17	0	
s4	17	0	
s5	17	0	
s6	17	0	
s7	17	0	
s8	17	0	
s9	17	0	
s10	17	0	
s11	17	0	
s12	17	0	
TOTAL	204	0	0

Tabla 11: Resultados Solución final (TS): 12 semestres, 66 asignaturas
Fuente: Elaboración Propia.

N° de iteraciones vs error absoluto cuadrático

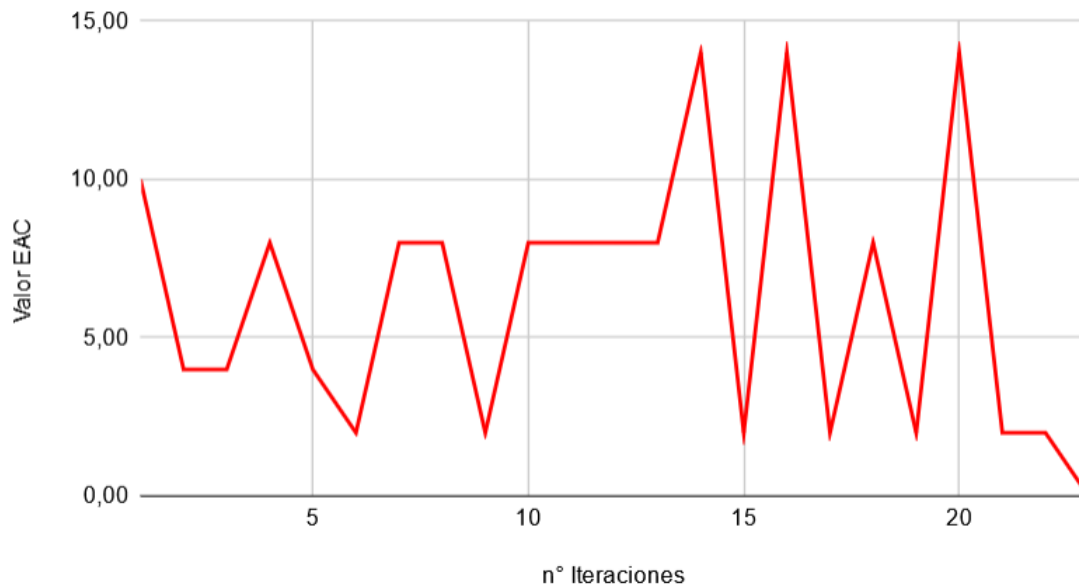


Figura 28: TS - Iteraciones vs EAC

Fuente: Elaboración propia

A diferencia de las ejecuciones anteriores, el algoritmo encontró el mejor valor en la iteración número 23, lo que significa que el promedio AVG (17⁷) es un número entero el cual además es posible conseguir organizando todas las asignaturas en los semestres correspondientes de tal forma que la suma de sus créditos sea AVG . La lista tabú es de 33 elementos.

4.2.4. Resumen Comparativo - Instancias de Benchmark

La siguiente tabla resume los resultados obtenidos en 100 iteraciones de la búsqueda tabú; esta tabla contiene:

- **AVG:** Promedio de carga semestral (créditos totales sobre número de semestres).
- **Suma EAC Semestral:** La suma de todas los errores absolutos cuadráticos, entre más cerca de 0 mejor.
- **Tiempo de ejecución:** El tiempo que toma el algoritmo en ejecutarse completamente en segundos.

⁷*En la sección 2.2 existen registros del resultado de esta instancia siendo 18, esto es debido a que en algún punto la instancia de CSPLib fue edita y se eliminó una o más asignatura(s), por lo que ahora la solución óptima es 17.

- **Mejor Vector de Créditos Semestrales:** El vector solución entregado por el algoritmo al termino de la ejecución, cada posición representa el semestre y el valor la cantidad de créditos asignados.

Instancia	AVG	Suma EAC Semestral	Tiempo de Ejecución Promedio [s]	Mejor Vector de Créditos Semestrales
8-Semestres	16,625	1.875	0.0215	[16, 17, 17, 17, 17, 17, 16, 16]
10-Semestres	13,4	2.4	0.0205	[13, 13, 13, 14, 13, 14, 14, 14, 13, 13]
12-Semestres	17	0	0.0260	[17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17]

Tabla 12: Resumen Comparativo - Instancias Benchmark

4.3. Instancias de prueba reales - Malla Informática USM (2023)

En la siguiente sección se evaluarán una serie de instancias en base al currículum académico de ing. civil informática UTFSM (2023), estas instancias se basarán en la solución del problema base (la malla completa) con 35 créditos máximos semestrales, para luego generar nuevas soluciones en base a ésta.

4.3.1. 11 semestres - 59 asignaturas

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	28	0.40495	
s2	27	0.13223	
s3	27	0.13223	
s4	28	0.40495	
s5	28	0.40495	
s6	28	0.40495	
s7	28	0.40495	
s8	27	0.13223	
s9	26	1.85950	
s10	27	0.13223	
s11	27	0.13223	
TOTAL	301	4.54545	2

Tabla 13: Resultados Solución inicial (CP): 11 semestres, 59 asignaturas
Fuente: Elaboración Propia.

DESARROLLO DE SOFTWARE BASADO EN METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE
BALANCEO DE MALLAS CURRICULARES DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL ACADÉMICO
PARA ALUMNOS DE INFORMÁTICA DE LA UTFSM CC

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	28	0.40495	
s2	27	0.13223	
s3	27	0.13223	
s4	28	0.40495	
s5	28	0.40495	
s6	28	0.40495	
s7	28	0.40495	
s8	27	0.13223	
s9	26	1.85950	
s10	27	0.13223	
s11	27	0.13223	
TOTAL	301	4.54545	2

Tabla 14: Resultados Solución final (TS): 11 semestres, 59 asignaturas
Fuente: Elaboración Propia.

Como se puede ver en la tabla anterior, los mejores valores entre la solución inicial y la solución con tabu search son los mismos, para el segundo se utilizó múltiples cantidades de iteraciones: 100, 200, 500, 1000 y una variación del tamaño de la lista tabu: 0.5, 1, 1.5 y 2 veces la cantidad total de asignaturas, el valor no muestra ningún tipo de mejora, por lo que se considerará como el mejor valor.

4.3.2. 10 semestres - 59 asignaturas

Una compresión semestral para las mismas cantidad de asignaturas y parámetros de decisión.

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	31	0.80999	
s2	30	0.01000	
s3	30	0.01000	
s4	29	1.21000	
s5	33	8.40999	
s6	30	0.01000	
s7	30	0.01000	
s8	29	1.21000	
s9	29	1.21000	
s10	30	0.01000	
TOTAL	301	12.8999	4

Tabla 15: Resultados Solución inicial (CP): 10 semestres, 59 asignaturas
Fuente: Elaboración Propia.

DESARROLLO DE SOFTWARE BASADO EN METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE
BALANCEO DE MALLAS CURRICULARES DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL ACADÉMICO
PARA ALUMNOS DE INFORMÁTICA DE LA UTFSM CC

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	31	0.80999	
s2	30	0.01000	
s3	30	0.01000	
s4	29	1.21000	
s5	30	0.01000	
s6	30	0.01000	
s7	30	0.01000	
s8	30	0.01000	
s9	31	0.80999	
s10	30	0.01000	
TOTAL	301	2.89999	2

Tabla 16: Resultados Solución final (TS): 10 semestres, 59 asignaturas
Fuente: Elaboración Propia.

N° de iteraciones vs error absoluto cuadrático

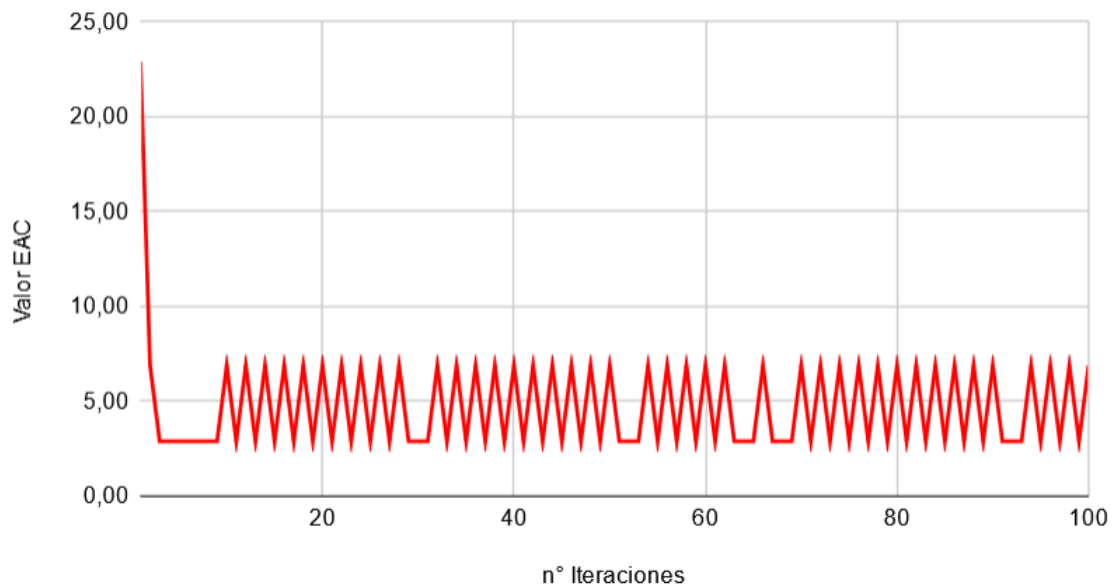


Figura 29: TS - Iteraciones vs EAC
Fuente: Elaboración propia

En esta problemática se logra encontrar la mejor solución en la 3ª iteración, equilibrando mejor la carga académica. La lista tabú es de 30 elementos.

4.3.3. 7 semestres, 35 asignaturas

Para la siguiente resolución se propondrá un currículum académico mucho más cercano al de un estudiante en el transcurso de su carrera. Se utilizará la malla interactiva para simular el historial académico de un alumno a través del tachado de asignaturas, lo cual será interpretado como asignaturas aprobadas a caso de una cantidad fija de semestres . A continuación, una representación gráfica de la situación académica del estudiante.

Malla Interactiva Informática

I	II	III	IV	V	VI	VII	VIII	IX	X	XI
INF-131 Programación	INF-010 Química y Sociedad	INF-134 Estructuras de Datos	INF-253 Lenguajes de Programación	INF-239 Bases de Datos	INF-236 Análisis y Diseño de Software	INF-225 Ingeniería de Software	INF-322 Diseño Interfazes Usuarías	INF-302 Electivo Informática II	INF-304 Electivo Informática IV	INF-310 Trabajo de Título 2
INF-021 Matemáticas I	INF-022 Matemáticas II	INF-023 Matemáticas III	MAT-024 Matemáticas IV	INF-245 Arquitectura y Organización de Computadores	INF-246 Sistemas Operativos	INF-256 Redes de Computadores	INF-343 Sistemas Distribuidos	INF-303 Electivo Informática III	INF-313 Electivo III	
INF-100 Introducción a la Física	INF-110 Física General I	INF-130 Física General III	FIS-120 Física General II	FIS-140 Física General IV	INF-276 Ingeniería, Informática y Sociedad	INF-285 Computación Científica	INF-301 Electivo Informática I	INF-311 Electivo I	INF-314 Electivo IV	
INF-132 Humanístico I	INF-101 Introducción a la Ingeniería	INF-152 Estructuras de Datos	INF-155 Informática Teórica	INF-280 Estadística Computacional	INF-221 Algoritmos y Complejidad	INF-285 Computación Científica	INF-295 Inteligencia Artificial	INF-312 Electivo II	INF-228 Taller Desarrollo de Proyecto de Informática	
INF-100 Educación Física I	INF-133 Humanístico II	INF-260 Teoría de Sistemas	INF-170 Economía IA	INF-270 Organizaciones y Sistemas de Información	INF-292 Optimización	INF-293 Investigación de Operaciones	INF-266 Sistemas de Gestión	INF-360 Gestión de Proyectos de Informática	INF-309 Trabajo de Título 1	
	INF-101 Educación Física II	INF-151 Libre I	INF-152 Libre II	INF-153 Libre III	INF-4 Libre IV	INF-5 Libre V	INF-6 Libre VI	INF-7 Libre VII		

Figura 30: Malla Curricular - Instancia de prueba

Fuente: [Aedo, 2018]

DESARROLLO DE SOFTWARE BASADO EN METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE
BALANCEO DE MALLAS CURRICULARES DE MANERA DINÁMICA CONSIDERANDO EL HISTORIAL ACADÉMICO
PARA ALUMNOS DE INFORMÁTICA DE LA UTFSM CC

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	28	1.30612	
s2	27	0.02040	
s3	25	3.44897	
s4	30	9.87755	
s5	25	3.44897	
s6	28	1.30612	
s7	25	3.44897	
TOTAL	188	22.857	5

Tabla 17: Resultados Solución inicial (CP): 7 semestres, 35 asignaturas

Fuente: Elaboración Propia.

Semestres	Créditos	Error absoluto cuadrático	$\max(c_i) - \min(c_j)$
s1	26	0.73469	
s2	27	0.02040	
s3	28	1.30612	
s4	27	0.02040	
s5	25	3.44897	
s6	28	1.30612	
s7	27	0.02040	
TOTAL	188	6.85714	3

Tabla 18: Resultados Solución final (TS): 7 semestres, 35 asignaturas

Fuente: Elaboración Propia.

Como se puede notar en la tabla la solución inicial, se tiene un error absoluto cuadrático total de 22.857, donde el mayor valor representado en la tabla es de 9.8 para el cuarto semestre, lo que significa, que existe un leve desequilibrio de carga académica entre los semestres.

Se tiene 188 créditos total sobre 7 semestres, lo que implica una distribución promedio de 26.85, o bien, 27 en términos prácticos, y como se puede ver en la tabla, existen múltiples semestre con 25 créditos (1.85 créditos por debajo) y uno de 30 (3.15 créditos por sobre).

En la tabla de resultados para la aplicación de *tabu search* se puede ver una disminución considerable de error absoluto cuadrático siendo el total 6.85. Como se puede ver en la columna de créditos, están mucho mejor distribuidos más cercanos al valor óptimo teórico (26.85).

Un caso interesante es el del semestre 5, este conserva el error incluso después de la búsqueda tabú, lo cual puede significar que no es posible asignar otras asignaturas al semestre 5 de tal manera que su valor se acerque más al óptimo sin empeorar demasiado el error de otro semestre, esto se puede deber a distintos factores como por ejemplo: Cuello de botella en el semestre 5, una lista tabú demasiado pequeña para explorar lo suficiente, los prerre-

quisitos deben estar ordenados de esa forma en particular, entre otros. La información que entrega la función de valor es importante para discernir problemas, pero para comprender la causa de estos, es necesario un análisis más profundo.

N° de iteraciones vs error absoluto cuadrático

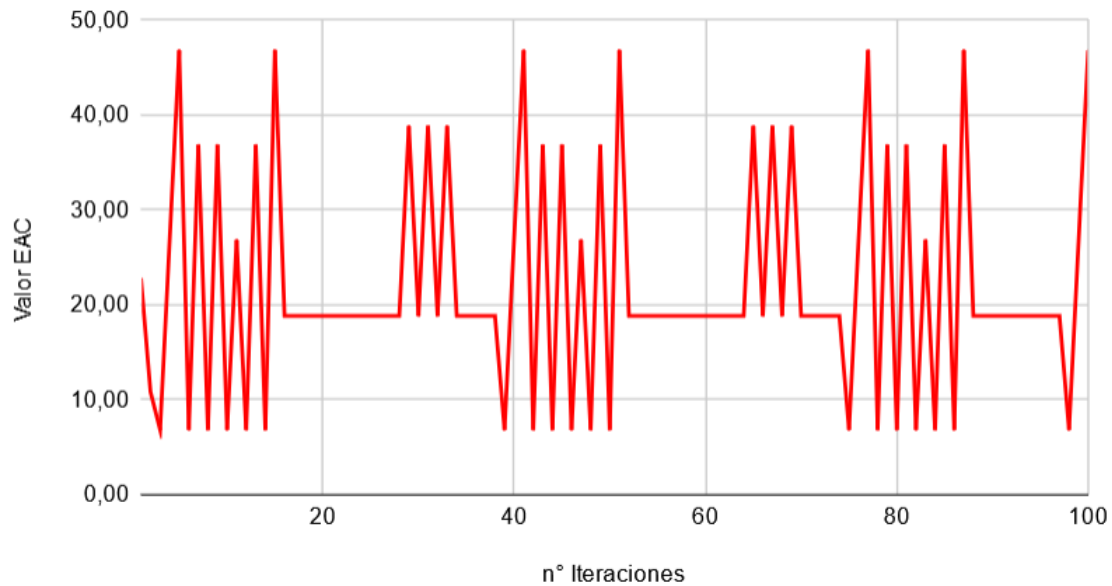


Figura 31: TS - Iteraciones vs EAC
Fuente: Elaboración propia

En esta problemática se logra encontrar una solución óptima en la 3ª iteración. La lista tabú es de 35 elementos.

A continuación se presenta una representación gráfica de la solución obtenida para la problemática planteada (ver figura 30) para 7 semestres con 35 créditos máximos semestrales.

Malla Interactiva BACP

I	II	III	IV	V	VI	VII
MAT-024 Matemáticas 4 1 6	FIS-120 Física General 2 6 8	INF-245 Arquitectura y Organización de Computadores 12 5	INF-246 Sistemas Operativos 17 5	INF-295 Inteligencia Artificial 23 5	INF-256 Redes de Computadores 27 5	INF-343 Sistemas Distribuidos 28 5
INF-280 Estadística Computacional 2 5	INF-221 Algoritmos y Complejidad 7 5	FIS-140 Física General 4 13 8	INF-4 Libre 4 18 2	INF-303 Electivo Informática 3 24 5	INF-293 Investigación de Operaciones 28 6	INF-6 Libre 6 34 2
INF-236 Análisis y Diseño de Software 3 5	INF-225 Ingeniería de Software 8 5	INF-292 Optimización 14 5	INF-302 Electivo Informática 2 19 5	INF-304 Electivo Informática 4 25 5	INF-301 Electivo Informática 1 29 5	INF-310 Trabajo de Título 2 35 20
INF-276 Ingeniería, Informática y Sociedad 4 5	INF-5 Libre 5 9 2	INF-285 Computación científica 15 5	INF-312 Electivo 2 20 5	INF-228 Taller de Desarrollo de Proyecto de Informática 20 10	INF-311 Electivo 1 30 5	
ICN-270 Información y Matemáticas Financieras 5 5	INF-266 Sistemas de Gestión 10 5	INF-322 Diseño de Interfaces Usuarios 10 5	INF-360 Gestión de Proyectos Informáticos 21 5		INF-313 Electivo 3 31 5	
	INF-7 Libre 7 11 2		INF-314 Electivo 4 22 5		INF-309 Trabajo de Título 1 32 2	

Figura 32: Semestre resultado - 7 semestres, 35 asignaturas
Fuente: Elaboración propia

4.3.4. Resumen comparativo - Instancias reales

La siguiente tabla resume los resultados obtenidos en 100 iteraciones de la búsqueda tabú, con exactamente los mismos atributos presentados anteriormente (Ver Tabla 12).

Instancia	AVG	Suma EAC Semestral	Tiempo de Ejecución Promedio [s]	Mejor Vector de Créditos Semestrales
7-Semestres	26,85	6.8571	0.0374	[26, 27, 28, 27, 25, 28, 27]
10-Semestres	30,1	2.8999	0.0541	[31, 30, 30, 29, 30, 30, 30, 30, 31, 30]
11-Semestres	27,363	4.5454	0.0559	[28, 27, 27, 28, 28, 28, 28, 27, 26, 27, 27]

Tabla 19: Resumen Comparativo - Instancias Reales

4.4. Validación de la Interfaz

En la siguiente sección se validará el desarrollo de la interfaz en base a pruebas realizadas a distintos usuarios, con la finalidad de analizar la primera instancia de uso que los usuarios proponen y el cómo interactúan con la solución para instancias específicas.

Las siguientes pruebas se realizaron en mezcla de 10 alumnos y ex-alumnos de ing. civil informática de la universidad técnica Federico Santa María, en base a las siguientes pruebas:

1. Resolver la malla completa en BACP para 11 semestres y 35 créditos máximos semestrales.
2. Interactuar con la malla generada y aprobar los 2 primeros semestres.
3. Re estructurar la malla de forma óptima (BACP) en base a la solución anterior para 8 semestres y 35 créditos máximos semestrales.

4.4.1. System Usability Scale

Para las evaluaciones se utilizó un cuestionario de *System Usability Scale* (SUS)[Brooke, 1996] el cual consiste de las siguientes 10 afirmaciones:

1. Creo que me gustaría utilizar este sistema con frecuencia
2. Encontré el sistema innecesariamente complejo
3. Pensé que el sistema era fácil de usar
4. Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema
5. Encontré que las diversas funciones de este sistema estaban bien integradas

6. Pensé que había demasiada inconsistencia en este sistema
7. Me imagino que la mayoría de la gente aprendería a utilizar este sistema muy rápidamente
8. Encontré el sistema muy complicado de usar
9. Me sentí muy seguro usando el sistema
10. Necesitaba aprender muchas cosas antes de empezar con este sistema

Cada afirmación es evaluada de 1 a 5, siendo el valor más bajo “Estoy en Desacuerdo” y el valor más alto “Estoy Deacuerdo”, luego es posible cuantificar el resultado total del cuestionario a través de los siguientes 3 pasos:

1. $X = (X_i - 1)$ Restar 1 a todos los valores de preguntas impares.
2. $Y = (5 - Y_i)$ Restar a 5 todos los valores de preguntas pares.
3. $\text{puntaje SUS} = (X + Y) \cdot 2,5$

Tras un estudio realizado en más de 500 webs y aplicaciones, se concluye que el puntaje promedio es 68. Un resultado por debajo de esta cifra indica que hay varios aspectos a corregir. [Sauro, 2011]

4.4.2. Resultados

Todas las pruebas fueron realizadas con éxito por los usuarios sin complicación alguna entregando los siguientes resultados.

Tester	Pregunta 1	Pregunta 2	Pregunta 3	Pregunta 4	Pregunta 5	Pregunta 6	Pregunta 7	Pregunta 8	Pregunta 9	Pregunta 10	SUS Puntaje
1	4	1	5	1	4	1	5	1	5	1	95
2	5	1	5	1	5	1	5	1	5	1	100
3	5	1	5	1	5	1	5	1	5	1	100
4	5	1	5	1	5	1	5	1	5	1	100
5	5	1	5	1	5	1	5	1	5	1	100
6	5	1	5	1	5	1	5	1	5	1	100
7	5	1	5	1	5	1	5	1	5	1	100
8	4	1	5	1	5	1	5	1	5	1	97,5
9	4	1	5	1	5	1	5	1	5	1	97,5
10	3	1	5	1	5	1	5	1	5	1	95
Promedio	4,5	1	5	1	4,9	1	5	1	5	1	98,5

Tabla 20: Tabla Resultados SUS

El resultado final de puntaje SUS es 98,5 lo cual implica que su uso es favorable frente a usuarios, es importante destacar que los usuarios (como fue comentado con anterioridad) son alumnos y ex alumnos de ing. civil informática por lo que el uso de este tipo de herramientas no es particularmente complicado para ellos, además de estar basado en un aplicación web bastante conocida dentro de la casa de estudios.

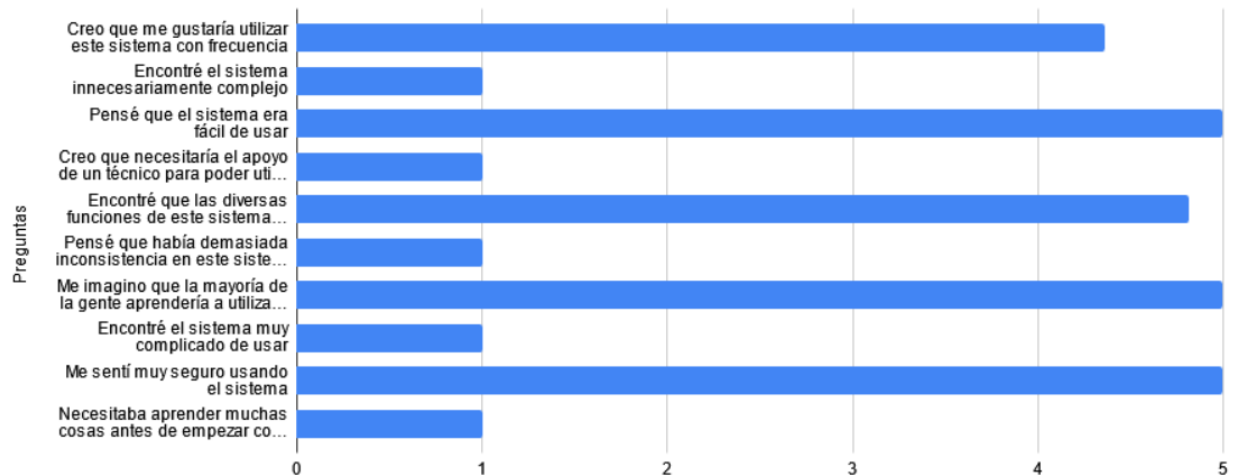


Figura 33: Grafico Resultados SUS
Fuente: Elaboración propia

4.5. Validación de Solución: Conclusiones

En este capítulo se presentaron los resultados obtenidos a través de la aplicación de *tabu search* tanto en instancias de referencia como en instancias más cercanas a la realidad, obteniendo resultados satisfactorios en cada una de las problemáticas planteadas.

El comportamiento de los gráficos presentados en las secciones 4.2 y 4.3 indican que existen múltiples soluciones simétricas encontradas a partir de la mejor solución (dada la lista tabú, se sabe que no pueden ser ciclos), lo que evidencia que no solo es una la mejor respuesta encontrada sino que a partir de una ejecución del algoritmo, es posible ofrecer múltiples soluciones (en caso de existir) al estudiante, mejorando las alternativas propuestas y ayudándole a tomar una mejor decisión respecto al orden de su malla académica.

La malla interactiva se muestra como una excelente base a utilizar dado el conocimiento de uso por parte de la comunidad informática USM, la baja curva de aprendizaje, accesibilidad y la utilidad que aporta como herramienta diseñada específicamente para estudiantes. Es lógico entonces, la incorporación de sistemas que ayuden cada vez más a los alumnos en la toma de decisiones académicas, como lo sería la resolución y ordenamiento que propone la solución del BACP estudiantil. Así puede verse en los resultados obtenidos del *System Usability Scale* sobre la malla interactiva con la capacidad de resolver el BACP estudiantil, obteniendo un muy alto puntaje SUS.

En el siguiente capítulo se presentan las conclusiones generales del documento.

CAPÍTULO 5

CONCLUSIONES

La organización eficiente de mallas curriculares es un desafío clave para las instituciones de educación superior, ya que impacta directamente en la experiencia académica de los estudiantes y en la eficacia operativa de la universidad. Este trabajo presentó el diseño y desarrollo de una herramienta basada en *Tabu Search* para abordar el Balanced Academic Curriculum Problem (BACP) estudiantil, adaptada específicamente a las necesidades de la carrera de ing. civil informática en la Universidad Técnica Federico Santa María.

A través del uso de técnicas avanzadas de optimización con búsqueda tabú, se logró modelar y resolver el problema de distribución de asignaturas, respetando restricciones fundamentales como la carga académica por semestre, relaciones de precedencia y la cantidad mínima y máxima de créditos. La herramienta propuesta no solo automatiza un proceso que tradicionalmente es manual y propenso a errores, sino que también genera soluciones balanceadas y personalizadas, contribuyendo a mejorar la planificación curricular.

Como se comentó en la sección 4.5, la resolución del BACP estudiantil tiende a generar distintas soluciones satisfactorias simétricas, lo que implica que puede generar múltiples posibles mallas alternativas (en caso de existir), lo cual, si se aplica correctamente, puede otorgar múltiples posibilidades de elección al estudiante, facilitando mucho más su proceso de elección y acomodándose más a las necesidades temporales del estudiante, puesto que, al tener más elección, es posible considerar elementos tales como: tope de horarios, preferencias de asignaturas, horario académico, entre otras. Para el alcance de este trabajo y documento no se consideró la elección de múltiples mallas académicas simétricas (más allá de las que pueda generar el estudiante mismo con la herramienta), sin embargo, es una idea muy interesante y propone una extensión del uso de la resolución del BACP estudiantil, por lo que es importante a considerar como una posible mejora y/o trabajo futuro.

Resultados Los resultados obtenidos destacan la eficacia de las metaheurísticas implementadas para encontrar soluciones que cumplen con las restricciones definidas, optimizando la carga académica y el avance lógico en la trayectoria de los estudiantes. Además, la flexibilidad del modelo permite su adaptación a otras carreras o instituciones, lo que amplía el alcance y las posibles aplicaciones de esta solución. Cada instancia presentada se pudo solucionar en pocas iteraciones con un muy buen tiempo de ejecución incluyendo las instancias reales asociadas a la malla curricular informática (2023) donde se puede apreciar un reordenamiento cercano a los 27 créditos semestrales, demostrando la efectividad de generar una solución inicial en base al uso de un método híbrido para luego explotar la solución y explorar posibles mejores respuestas con búsqueda tabú. La herramienta de software diseñada para facilitar el uso del BACP a estudiantes fue altamente calificada en función de su utilidad y comprensión de uso, por lo que en ese ámbito es considerado un éxito.

Análisis de técnica utilizada La búsqueda tabú presenta una estructura simple pero efectiva, siendo capaz de “recordar” y prohibir movimientos recientes forzando la exploración de la malla curricular y así evitar ciclos o respuestas simétricas (en lo posible). El tamaño de la lista tabú es el parámetro más importante para la exploración, en este caso, determina cuanto puede extenderse la búsqueda tabú prohibiendo estados anteriores, sin embargo, una lista tabú muy larga podría evitar llegar a las mejores soluciones dado que se podrían prohibir todos los movimientos viables, por otro lado una lista tabú muy corta podría ser propensa a generar ciclos dada la naturaleza del problema. Lo que vuelve interesante al problema en cuanto a su resolución es que no sólo depende de la cantidad de semestres para asignación de cursos, sino que además de cuántos cursos son y las restricciones de carga académica, esto vuelve más importante aún utilizar el valor correcto para el tamaño de la lista tabú.

En el caso de la solución inicial obtenida, los resultados fueron bastante buenos, muy cercanos a los óptimos sin evitar que la búsqueda tabú pudiera encontrarlos. Uno de los problemas más notables es que depende mucho de las heurísticas de ordenamiento implementadas para la asignación de cursos, puesto que con la implementación de rangos utilizada, una mala distribución inicial podría implicar una solución no viable la cual asignada con el método correcto pudo haber sido encontrada fácilmente, por esto es fundamental la aplicación de la(s) heurísticas(s) de orden correctas.

Posibles Mejoras - Recomendaciones Aplicar más restricciones sobre la construcción de la solución inicial podría ayudar a delimitar mejor los rangos definidos, evitando situaciones con soluciones no viables, también utilizar distintos tipos de métodos y ordenamientos podrían proponer un algoritmo más sólido frente a curriculums académicos poco ortodoxos. Trabajar un sistema con parámetros adicionales como disponibilidad de profesores, horarios preferidos por los estudiantes (paralelos) y dificultades de asignaturas podría extender el problema base del BACP y considerar directamente los horarios de los estudiantes en vez de solo un semestre con una carga académica distribuida, en el mundo real existen muchos más factores que impactan las decisiones semestrales de los alumnos y el cómo estos deciden qué asignaturas tomar durante un periodo académico, por lo que es importante poder avanzar más el problema.

Trabajo Futuro Finalmente, este trabajo abre nuevas posibilidades para futuras investigaciones y mejoras, como la incorporación de nuevas restricciones y objetivos o la integración de la herramienta con sistemas de gestión universitaria ya existentes. La solución presentada demuestra el potencial de la computación y la optimización para resolver problemas complejos en contextos educativos, ofreciendo beneficios tanto a nivel institucional como estudiantil. Es importante considerar que el BACP es solo una problemática específica que de base considera solo la asignación equitativa de carga académica, aun hay mucho espacio para trabajar y mejorar esta asignación considerando al estudiante como prioridad.

REFERENCIAS BIBLIOGRÁFICAS

- [Aedo, 2018] Aedo, S. (2018). Malla interactiva. <https://github.com/etra0/ramos>. Accessed: Dec 24, 2024.
- [Brooke, 1996] Brooke, J. (1996). Sus: A quick and dirty usability scale. *Usability Evaluation in Industry*.
- [Castro y Manzano, 2001] Castro, C. y Manzano, S. (2001). Variable and value ordering when solving balanced academic curriculum problems. *arXiv preprint cs/0110007*.
- [Chiarandini et al., 2012] Chiarandini, M., Di Gaspero, L., Gualandi, S., y Schaerf, A. (2012). The balanced academic curriculum problem revisited. *Journal of Heuristics*, 18:119–148.
- [Cordon et al., 2000] Cordon, Oscar and de Viana, I Fernández and Herrera, Francisco and Moreno, Llanos and others (2000). A new aco model integrating evolutionary computation concepts: The best-worst ant system. *Proc. ANTS 2000*, pp. 22–29.
- [Di Gaspero y Schaerf, 2008] Di Gaspero, L. y Schaerf, A. (2008). Hybrid local search techniques for the generalized balanced academic curriculum problem. En *Hybrid Metaheuristics: 5th International Workshop, HM 2008, Málaga, Spain, October 8-9, 2008. Proceedings 5*, pp. 146–157. Springer.
- [Gent y Walsh, 1999] Gent, I. P. y Walsh, T. (1999). Csplib: a benchmark library for constraints. En *Principles and Practice of Constraint Programming-CP'99: 5th International Conference, CP'99, Alexandria, VA, USA, October 11-14, 1999. Proceedings 5*, pp. 480–481. Springer.
- [Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1st edición.
- [Lambert et al., 2006] Lambert, T., Castro, C., Monfroy, E., y Saubion, F. (2006). Solving the balanced academic curriculum problem with an hybridization of genetic algorithm and constraint propagation. En *Artificial Intelligence and Soft Computing-ICAISC 2006: 8th International Conference, Zakopane, Poland, June 25-29, 2006. Proceedings 8*, pp. 410–419. Springer.
- [Monette et al., 2007] Monette, Jean-Noël and Schaus, Pierre and Zampelli, Stéphane and Deville, Yves and Dupont, Pierre and others (2007). A cp approach to the balanced academic curriculum problem. En *Seventh International Workshop on Symmetry and Constraint Satisfaction Problems*, volumen 7.
- [Paulangelo, 2024] Paulangelo, C. (2024). Malla interactiva. <https://github.com/BooterMan98/malla-interactiva>. Accessed: Dec 24, 2024.

- [Rubio *et al.*, 2013] Rubio, J.-M., Palma, W., Rodriguez, N., Soto, R., Crawford, B., Paredes, F., y Cabrera, G. (2013). Solving the balanced academic curriculum problem using the aco metaheuristic. *Mathematical Problems in Engineering*, 2013(1):793671.
- [Rubio *et al.*, 2018] Rubio, J. M., Soto, R., Jorquera, H., Aguilera, J., y Vidal, C. (2018). Resolución del problema de balanceo de mallas curriculares mediante algoritmo de luciérnagas. *Ingeniare. Revista chilena de ingeniería*, 26:102–112.
- [Sauro, 2011] Sauro, J. (2011). Sus valor. <https://measuringu.com/sus/>. Accessed: Dec 27, 2024.
- [Villalobos-Cid *et al.*, 2019] Villalobos-Cid, M., Orellana, M., Vásquez, O. C., Pinto-Sothers, E., e Inostroza-Ponta, M. (2019). Dealing with the balanced academic curriculum problem considering the chilean academic credit transfer system. En *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–7. IEEE.
- [Wolsey y Nemhauser, 1999] Wolsey, L. A. y Nemhauser, G. L. (1999). *Integer and combinatorial optimization*. John Wiley & Sons.
- [Yang y He, 2013] Yang, X.-S. y He, X. (2013). Firefly algorithm: recent advances and applications. *International journal of swarm intelligence*, 1(1):36–50.