UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ÁREA DE ESTUDIOS DE POSTGRADO
MAESTRÍA EN MATEMÁTICAS Y COMPUTACIÓN

ALGORITMO BASADO EN LA APLICACIÓN DE TÉCNICAS METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE BALANCEO DE CARGA CURRICULAR

Autor:
Lic. Daniel Humberto Rosquete De Mora

Tutor:
Dr. Amadís Antonio Martínez Morales

Bárbula, Febrero de 2014

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ÁREA DE ESTUDIOS DE POSTGRADO
MAESTRÍA EN MATEMÁTICAS Y COMPUTACIÓN

ALGORITMO BASADO EN LA APLICACIÓN DE TÉCNICAS METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE BALANCEO DE CARGA CURRICULAR

Autor:
Lic. Daniel Humberto Rosquete De Mora

Trabajo presentado ante el Área de Estudios de Postgrado de la Universidad de Carabobo para optar al Título de Magíster en Matemática y Computación.

Bárbula, Febrero de 2014

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ÁREA DE ESTUDIOS DE POSTGRADO
MAESTRÍA EN MATEMÁTICAS Y COMPUTACIÓN

ALGORITMO BASADO EN LA APLICACIÓN DE TÉCNICAS METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE BALANCEO DE CARGA CURRICULAR

Autor:
Lic. Daniel Humberto Rosquete De Mora

Aprobado en el Área de Estudios de Postgrado de la Universidad de Carabobo por Miembros de la Comisión Coordinadora del Programa:

_____

_____

_____

Bárbula, Febrero de 2014

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ÁREA DE ESTUDIOS DE POSTGRADO
MAESTRÍA EN MATEMÁTICAS Y COMPUTACIÓN

VEREDICTO

Nosotros, Miembros del Jurado designado para la evaluación del Trabajo de Grado titulado: **ALGORITMO BASADO EN LA APLICACIÓN DE TÉCNICAS METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE BALANCEO DE CARGA CURRICULAR**, presentado por: **Lic. Daniel Humberto Rosquete De Mora** para optar al título de **Magíster en Matemática y Computación**, estimamos que el mismo reúne los requisitos para ser considerado como: **Aprobado**.

_____

_____

_____

Bárbula, Febrero de 2014

RECONOCIMIENTOS

A la vida por permitirme llegar hasta este momento tan importante, superando todas las dificultades que me fueron colocadas, deliberadamente o no, para alcanzar esta meta. Las circunstancias reafirmaron en mí, el coraje, los buenos valores, la constancia y por encima de todo, que ante la injusticia, debe prevalecer la razón.

A mis padres, Ysabel y Omar, por darme la vida, el cariño, la educación y los valores, que hoy, dan frutos académicos. Siempre han estado pendientes de cada paso, y sólo eso, es motivo para agradecerles, pero, como si no fuera suficiente, ustedes siempre han dado todo de sí. No basta una vida para agradecerles.

A mi hermano Omar, quien siempre nos ha demostrado a todos que con: sentido común, trabajo duro y buenas intenciones; se logran más éxitos que con cien títulos y toneladas de reconocimientos. Un título sólo abre las puertas, lo que haces después, es lo que marcará la diferencia.

A mi novia Kiara, mi gran amiga, mi amor y mi compañera de vida en todas las circunstancias. Tú has sido la inspiración del motor de ideas, la fuerza de seguir adelante, de no desanimar. Cada paso, lo hemos dado juntos y nunca nos hemos soltado las manos para avanzar sin el otro. A lo largo del proyecto, has sido apoyo, consulta, pensamientos y mucho más, que hoy, completa este trabajo. Este logro es nuestro, te amo!

A mi gran amigo y tutor académico, Amadís, quien siempre ha trazado el camino a seguir y ha apoyado cada una de las batallas sin importar sus resultados. Siempre has estado allí amigo mío y te agradezco que haya sido así y siga siéndolo.

A mi gran amigo, Jesús, sin su ayuda y apoyo este trabajo nunca hubiese llegado a presentarse. A mis etéreos, siempre están en mis recuerdos. ¡Gracias... Totales!

INDICE GENERAL

CAPÍTULO III

MARCO METODOLÓGICO

CAPÍTULO IV

RESULTADOS

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

ALGORITMO BASADO EN LA APLICACIÓN DE TÉCNICAS METAHEURÍSTICAS PARA LA RESOLUCIÓN DEL PROBLEMA DE BALANCEO DE CARGA CURRICULAR

Autor:
Lic. Daniel Humberto Rosquete De Mora

Tutor:
Dr. Amadís Antonio Martínez Morales

Bárbula, Febrero de 2014

El balanceo de carga curricular, consiste en un problema combinatorio con restricciones. El planteamiento, es ubicar un conjunto de materias de un *curriculum* en un período (trimestre, semestre u otro), cumpliendo con todas las prelaciones que puede tener una asignatura y adicionalmente, que ciertas asignaturas no deben ser cursadas en determinados períodos. Estas restricciones, constituyen una gran limitante para armar un *pensum* de una carrera, además, se sabe que si un estudiante, tiene una gran carga horaria durante un período, no podrá internalizar todo el conocimiento en su completitud y por lo tanto su rendimiento no será sobresaliente. Es por ello, que dadas las restricciones, el objetivo es balancear la carga curricular lo mejor posible para que el estudiante no se encuentre afectado con una carga mal distribuida. La solución a este problema puede plantearse de múltiples formas, desde programación lineal, hasta el uso de técnicas heurísticas, teniendo cada algoritmo sus ventajas. En este trabajo de investigación, se implementa el algoritmo de GRASP junto a una estructura de datos diseñada para el problema y se demuestra que el resultado de utilizar un método heurístico para resolver el problema, permite obtener buena calidad en las soluciones y bajos tiempos de respuesta.

Palabras clave: Heurística, GRASP, BACP, problema de balanceo, combinatoria, programación lineal.

INTRODUCCIÓN

En el área de la educación superior existen múltiples reglas generales, que se han establecido para poder llevar todas las carreras por el mismo camino de trabajo. Algunas de estas reglas han sido mejoradas con el tiempo, otras han sido olvidadas y otras se han mantenido. La mayoría de las labores relacionadas con el objetivo de las carreras universitarias las ejerce un departamento de control de estudios, en los cuales desde hace unos cinco años hasta la fecha de este trabajo han intentado automatizar muchas de sus labores.

Actualmente, muy pocos departamentos de control de estudios llevan el registro de sus estudiantes de forma manual, ni el *pensum* de todas las carreras universitarias, sólo lo llevan en formato físico. Las universidades han migrado la mayoría de sus sistemas de control a un formato digital y a sistemas complejos que son administrados por diferentes entidades.

La elaboración de una planificación curricular es una tarea ardua, donde algunas de las variables que se consideran son: número de cursos, cantidad de períodos académicos (trimestres, cuatrimestres, semestres, años), cuota de créditos por curso, el número mínimo y máximo de asignaturas y/o créditos que podrían cursar por período académico. Esta situación lleva a que, en algunos períodos académicos, el estudiante tenga la carga máxima de horas y en otros la carga mínima, implicando que en las etapas de mayor carga horaria se tenga menor tiempo para estudiar cada asignatura y esta situación se refleja en sus calificaciones.

Un factor importante a tomar en consideración cuando se evalúa el éxito obtenido por un estudiante universitario, es el número de horas que debe cursar por cada período académico. En algunas universidades ocurre que los estudiantes tienen a lo largo de sus períodos académicos un número regular de horas de clase; sin embargo, esto no es el caso común.

El objetivo de esta investigación consiste en minimizar el tiempo de ejecución utilizando una técnica metaheurística sobre casos de prueba sintetizados y exhaustivos; estos casos de prueba exhaustivos son de libre acceso y están planteados por los autores que han abordado la solución el problema con programación lineal entera.

Este trabajo está estructurado en cuatro capítulos, sin incluir esta introducción. En el capítulo 1 se realiza la formulación del problema a resolver, se plantean los objetivos de la investigación y su justificación. El capítulo 2 presenta el marco teórico que sirve como base para el desarrollo de este trabajo. En el capítulo 3 se introduce el marco metodológico utilizado. El capítulo 4 contiene los resultados de esta investigación: las estructuras de datos diseñadas, los algoritmos sobre estas estructuras, su implementación y el estudio experimental correspondiente. Posteriormente, se presentan las conclusiones, recomendaciones y trabajos futuros.

CAPÍTULO I

EL PROBLEMA

Planteamiento del Problema

En diversas universidades de todo el mundo están sucediendo cambios curriculares, que permiten adaptar el orden y el contenido de las asignaturas en las carreras, tanto tecnológicas como científicas, a las exigencias del mundo moderno. Por ejemplo, los estudios relacionados con informática, sistemas y computación, entendiéndose estas como áreas afines pero diferentes entre sí, son actualizados de manera constante, ya que las tendencias mundiales generan nuevas tecnologías a una gran velocidad. Algunos ejemplos de ellos son el auge e implementación de herramientas como *Moodle* en universidades a nivel mundial, incluyendo la Universidad de Carabobo, u otras herramientas como *Blackboard*, *Helvia*, entre otros.

Tener nuevas tecnologías para investigación y desarrollo lleva a los entes científicos a tener la necesidad de actualizar sus conocimientos, sin que esto implique renovarlos. Es por esta razón, que los encargados del manejo curricular de las carreras tecnológicas, tienen que permanecer en constante actualización del *pensum* académico, por lo que algunas materias se dejan de impartir mientras que otras las reemplazan. El problema que esto conlleva es que, a través de los períodos académicos, se empiezan a diferenciar en gran magnitud, las horas de clase que se imparten entre un período y otro, provocando que un estudiante tenga mejor rendimiento en algunos períodos y peor en otros.

El problema que se plantea es diseñar una carga curricular balanceada entre períodos académicos. Para esto, es necesario especificar que una asignatura tiene un número de horas semanales de sesiones presenciales, denominado carga académica. La labor entonces consiste en asignar cursos a períodos de forma que la carga académica entre períodos sea tan similar como sea posible, es decir, tener un número similar de horas con una diferencia aceptable de 20% entre períodos.

El estudio del problema de balanceo de carga curricular ha sido desarrollado previamente utilizando la técnica de programación lineal entera, y aunque a través de esta técnica se consigue la solución, el tiempo de respuesta es alto. Las metaheurísticas generalmente se aplican a problemas que no tienen un algoritmo específico que dé una solución satisfactoria; o bien cuando no es posible implementar ese método óptimo. La mayoría de las metaheurísticas tienen como objetivo los problemas de optimización combinatoria, pero por supuesto, se pueden aplicar a cualquier problema que se pueda reformular en términos heurísticos. Blum (2003)

En las ciencias de la computación, específicamente en la teoría de algoritmos, uno de los aspectos más importantes a considerar es la complejidad computacional de la solución a un problema determinado. En la complejidad se estudia, de manera teórica, los recursos requeridos durante el cómputo de un algoritmo para resolver un problema. En este contexto, se entiende por recursos requeridos el tiempo de respuesta y el espacio en memoria utilizado por el algoritmo para ofrecer una solución. Una categoría de estas complejidades es el subconjunto de problemas NP (*Nondeterministic Polynomial time*) que significa, por sus siglas en inglés, Polinómico no Determinista. Este tipo de problemas tienden a ser muy estudiados por la complejidad de plantear una solución capaz de resolverlos en tiempo polinomial.

En los problemas de optimización combinatoria el objetivo es encontrar el máximo (o el mínimo) de una determinada función sobre un conjunto finito de soluciones que denotaremos por *S*. No se exige ninguna condición o propiedad sobre la función objetivo o la definición del conjunto *S*. Es importante notar que dada la finitud de *S*, las variables han de ser discretas, restringiendo su dominio a una serie finita de valores. Habitualmente, el número de elementos de *S* es muy elevado, haciendo poco práctica la evaluación de todas sus soluciones para determinar el óptimo. Martí (2003)

Dentro del conjunto de problemas NP, se tienen diversas categorías, en este caso, el problema que se desea resolver está enmarcado en el concepto definido por Martí (2003) como un problema de optimización combinatoria y algunos autores lo colocan con complejidad computacional de NP-complejo, lo que significa que su complejidad puede ser tan alta como la de un problema NP-completo, pero su tiempo de ejecución puede reducirse a polinomial utilizando la técnica adecuada.

El problema de balancear las cargas curriculares no es sencillo, aunque parece simple, ya que existe un conjunto de restricciones que aumentan la complejidad del problema, conforme se aumentan el número de asignaturas que se desea impartir a lo largo de la carrera; de igual manera influye la cantidad de períodos en el que se divide toda la carga.

Otra característica que añade dificultad al momento de plantear el balanceo de una carga curricular es el número de horas que ocupa cada curso que se dicte, existen casos donde es imposible realizar un balance perfecto y se tiende a tener soluciones con un margen de horas de diferencia mínimas entre cada período.

Actualmente, la Facultad Experimental de Ciencias y Tecnología (FaCyT) de la Universidad de Carabobo (UC), atraviesa un proceso de semestralización ya que todas las carreras mantenían un régimen anual; este proceso exige una reformulación de la carga curricular.

El problema de balanceo curricular ha sido estudiado desde el 2001. Sin embargo, las soluciones que se han desarrollado, representan algoritmos cuyas complejidades son altas y en muchas oportunidades tienden a ser búsquedas exhaustivas, lo que resulta en altos tiempos de respuesta para obtener solución.

En la sección de antecedentes se mencionan a algunos de los investigadores que han trabajado sobre este problema, quienes han desarrollado algunos casos de prueba sintetizados con los cuales han evaluado sus algoritmos y han reportado tiempos de solución no muy prometedores.

El planteamiento de este proyecto de investigación, se basa en la afirmación de Martí (2003), la cual asegura, que la aplicación de técnicas metaheurísticas, es con el objetivo de conseguir una buena solución en un tiempo muy corto, aunque no necesariamente, sea la solución óptima. Las técnicas metaheurísticas han dado excelentes resultados en múltiples problemas NP para resolver esta situación, que hasta ahora ha sido abordada desde los enfoques tradicionales.

Finalmente lo que se desarrollará en esta investigación, es un algoritmo heurístico para resolver el problema de balanceo de carga curricular planteado, para posteriormente implementarlo en el lenguaje de programación C++, obteniendo así el *software* con el que se realizarán las pruebas.

Formulación del Problema

¿Cómo desarrollar una solución algorítmica eficiente, para resolver el problema de diseño de mallas curriculares balanceadas para *pensum* universitario, aplicando una técnica metaheurística?

Objetivos de la Investigación

Objetivo General

Desarrollar un algoritmo basado en una técnica metaheurística para la resolución del problema de balanceo de carga curricular, con el propósito de mejorar el tiempo de respuesta.

Objetivos Específicos

– Identificar los elementos fundamentales del problema.

– Diseñar el modelo a utilizar para resolver el problema de balanceo de carga curricular.

– Establecer las diferencias entre las técnicas utilizadas para resolver el problema y la técnica que se aplicará en este trabajo de investigación.

– Determinar la metaheurística adecuada para resolver de forma apropiada el

problema planteado.

– Determinar las variantes que se pueden aplicar al algoritmo base.

– Desarrollar el algoritmo heurístico de solución del problema de balanceo de carga curricular.

– Desarrollar el *software* y evaluarlo con casos de prueba sintetizados para determinar su correcto funcionamiento

– Comparar resultado de tiempo, entre las variantes implementadas y el planteamiento de esta solución.

Justificación del Estudio

Realizar este estudio viene justificado porque la Facultad Experimental de Ciencias y Tecnología de la Universidad de Carabobo (FaCyT-UC) requiere tener a su disposición, un *software* heurístico que resuelva el problema de balanceo de carga curricular de forma efectiva, es decir, que sea capaz de plantear los cursos de forma balanceada a lo largo de los semestres, para las cinco carreras de pregrado que allí se dictan (Licenciaturas en: Computación, Biología, Química, Física y Matemáticas).

Con fines prácticos, el *software* final además de resolver una problemática presente en la FaCyT-UC, permite dar respuesta a un problema que se presenta en múltiples Universidades a nivel mundial, por ejemplo, cuando un instituto desea iniciar

una nueva línea de estudios, o cuando desea modificarla para ajustarla a algún estándar moderno.

Representa un aporte importante en el área de las ciencias computacionales y las teorías combinatorias, ya que a través de la solución para el problema de balanceo de carga curricular, el modelo y técnicas utilizados se pueden aplicar en otros contextos que requieran de una combinatoria similar.

En el área académica se resuelve una situación que se presenta en todas las universidades a nivel nacional e internacional, con frecuencia los estudiantes responsabilizan al programa académico y su carga curricular, de su rendimiento (bueno o malo) que obtienen en las mismas, y por esta razón se debe crear las condiciones adecuadas para que el estudiante se sienta cómodo cursando sus períodos lectivos.

A nivel teórico, el resultado de esta investigación representará un apoyo a lo que define Martí (2003), cuando se refiere a que un método heurístico a diferencia de un método exacto, es mucho más rápido en dar una buena solución y además permite aplicarse en casos que de acuerdo al problema, se definen como grandes, mientras que en las soluciones exactas no son posibles de modelar, ni de obtener respuesta.

Por todas las razones antes expuestas y tomando en cuenta que aunque el objetivo principal no es desarrollar la aplicación para el caso específico de la FaCyT, se dará una aplicación rápida, directa y altamente necesaria al trabajo de investigación propuesto.

CAPÍTULO II

MARCO TEORICO REFERENCIAL

En este capítulo se establecen los antecedentes, las variables e hipótesis, la definición de términos básicos y bases teóricas, junto con la razón por la que fueron incluidas como parte de este trabajo de investigación.

Antecedentes

El problema de balanceo de carga curricular no ha sido ampliamente tratado, sin embargo existen trabajos recientes ya que es un área de estudio relativamente reciente, tiene cerca de 10 años, como se constata en las referencias y sólo en pocos países se ha evaluado este problema, además el problema no ha sido planteado a nivel nacional.

Chiarandini, M (2011) en su trabajo titulado "***The Balanced Academic Curriculum Problem Revisited*",** desarrolla una solución para el problema, lo que hace notar la vigencia del tema propuesto. Este trabajo fue realizado en conjunto con investigadores de universidades italianas y uno de Dinamarca quienes, dadas las restricciones particulares de este tema, decidieron incorporar el problema a la librería CSPLib, el cual es un repositorio de problemas con múltiples restricciones. Es importante recordar que muchos de estos problemas deben resolverse con programación lineal entera, pero dadas las características de este método, no siempre se obtiene la solución deseada en el tiempo esperado. Este estudio se relaciona con el presente trabajo de investigación, debido a que la CSPLib es una herramienta de gran

importancia para esta área de estudios, ya que permite formalizar de manera estándar los problemas con restricciones que son resueltos con cierta demora por los computadores y que son además tareas arduas de resolver para el ser humano. Algunas de las motivaciones para crear esta librería son las siguientes: medir el rendimiento, hacer competencia para mejorar la calidad de las soluciones, establecer modelos, añadir realismo a los modelos planteados, crear nuevas aplicaciones para resolver los llamados "embudos industriales" que son los que ralentizan el proceso y plantear nuevos retos, por esta razón decidieron colocar este problema en la librería.

Otro estudio publicado fue el realizado por Aguilar, J (2009) en su trabajo "**Estudio de diferentes conceptos de balance en modelos BACP**". BACP (*Balanced Academic Curricular Problem*) es el problema que se trata en esta investigación. El cual fue defendido por investigadores de la Universidad Popular Autónoma del Estado de Puebla en México, quienes aportaron ajustes al modelo inicialmente planteado por Castro (2001) y lo compararon con tres modelos adicionales del mismo problema, resultando en que el modelo propuesto daba mejores resultados que los anteriores. En su investigación utilizaron una extensión del *software* LINDO, lo cual los enmarcó en el esquema de programación lineal entera, esto aporta la primera solución computacional sólida a este problema. Aguilar desarrolló para este trabajo algunos casos de prueba de BACP que fueron utilizados en este estudio para validar los resultados del algoritmo.

Previamente, Aguilar, J (2007) en "**El problema de balancear un plan de estudios: Un modelo matemático"**. Presentó un primer modelo matemático, para formalizar el problema de balanceo de carga curricular, incluyendo por primera vez variables de entorno real como son: preferencias de ubicación de asignaturas y períodos

con distinta carga deseada, es decir, que aunque se aplica el balanceo de cargas a los períodos lectivos se puede ajustar un determinado período de forma intencional, y aplicar el balanceo en función de este. De este antecedente, se utilizó el planteamiento del modelo matemático, para fundamentar el algoritmo que se desarrolló.

Otro de los aportes importantes para este trabajo fue realizado por Lambert, T (2005), titulado "***Solving the Balanced Academic Curriculum Problem with an Hybridization of Genetic Algorithm and Constraint Propagation***" en la Universidad Católica de Louvain en Bélgica donde se realizó una implementación para la búsqueda de la solución del problema de balanceo de carga curricular, sólo que utilizando las técnicas: programación con restricciones y búsqueda local, las cuales han sido tomadas en cuenta para comparación de resultados *a posteriori*. En este trabajo se aportan bases del modelo, al igual que un conjunto de fórmulas y criterios de balanceo que ayudan a definir mejor el problema. Para efectos de la investigación actual, el aporte fue la definición de criterios más ajustada a casos de prueba reales.

Aunque han ocurrido un conjunto de trabajos posteriores, la investigación pionera fue realizada por Castro, C. (2001), "***Variable and Value Ordering When Solving Balanced Academic Curriculum Problems***". Planteado en la Universidad Técnica Federico Santa María en Chile, en este trabajo se crea el problema de Balanceo de Carga Curricular con su modelo y restricciones iniciales, igualmente los autores de este artículo propusieron una solución por programación lineal entera. Con el tiempo se estudió que, aunque las bases establecidas son sólidas, al modelo le faltaban múltiples variantes que lo harían ajustarse a un caso más real. Este estudio es la base a todos los presentados posteriormente, incluyendo el actual, ya que es en este trabajo donde se plantea este problema, como una posibilidad de estudio en el área de combinatoria.

Bases Teóricas

El área de estudio denominada: Investigación de Operaciones o Fundamentos de Optimización Computacional es una rama de las matemáticas fundamentada en la construcción de modelos matemáticos utilizando las estadísticas y algoritmos. Esta área de estudios es una de las más importantes para simular múltiples situaciones de la vida diaria como son, por ejemplo, las colas.

La Investigación de Operaciones tiene como propósito fundamental mejorar u optimizar los sistemas reales a través de la toma de decisiones teniendo en cuenta múltiples restricciones como escasez de recursos, tiempo máximo de permanencia, capacidad de personal, entre otras. Generalmente el objetivo que se estudia en esta área es la maximización de beneficios junto a la minimización de costos.

El área investiga dos tipos de problemas: los determinísticos y los estocásticos, de acuerdo a Ríos, I. (2004) los determinísticos son aquellos en los que se maneja toda la información completa para obtener una solución exacta, y los estocásticos son aquellos donde no se maneja toda la información necesaria para modelar el problema de manera exacta sino que por el contrario, se debe hacer uso de la estadística como se mencionó previamente. Para el problema a tratar en este trabajo de investigación se ubica de manera precisa como un método determinista ya que se tiene el orden de prelaciones completo, la carga en unidades de crédito, las materias electivas y restricciones adicionales impuestas por el modelo, además de que investigaciones previas ya lo han definido como tal.

Para esta área de estudios existen múltiples algoritmos definidos, al igual que varias técnicas de programación, de la cual hasta ahora se ha utilizado sólo programación entera para resolver el problema de balanceo de carga curricular. La programación lineal entera es un procedimiento algorítmico mediante el cual se busca resolver un problema generalmente de combinatoria formulado a través de inecuaciones lineales que representan las restricciones definidas, y se plantea de igual manera una función objetivo o ecuación objetivo que es la que se busca maximizar o minimizar de acuerdo a lo que se busque. Parte de los estudios previos a este, han buscado la solución al problema de balanceo de carga curricular utilizando el mismo método, ya que es un poco más sencillo de plantear, da resultados exactos para las soluciones y además el problema que se plantea necesita que se obtengan valores enteros, ya que en un período lectivo un estudiante no puede cursar 2.5 materias, aunque un profesor sí puede dictar una asignatura de forma colegiada, junto con otro colega, y ésta contaría a nivel del *curriculum* como una sola materia.

El aspecto que hará diferente este trabajo al igual que sus resultados al menos en tiempo de ejecución es el uso de metaheurística, para entender este concepto primero es necesario definir el término heurística. La heurística es la capacidad que tienen todos los seres humanos para crear, innovar o mejorar una situación, sistema u objeto. Como detalle curioso, la palabra heurística comparte su etimología con la palabra *Eureka*, que se utiliza para hacer referencia a una idea brillante que se ocurre.

Los computadores son máquinas matemáticas-lógicas que no tienen la capacidad de razonar por sí solas, ni de tomar decisiones al estilo “pensamiento divergente” como hace cualquier ser humano. Sin embargo, los científicos de la computación han llevado la tendencia de programación a que un computador pueda facilitar cálculos muy

complejos, labores de trabajo pesado (a través de la robótica), y un amplio conjunto de objetivos que se realizan en la vida diaria, es por esta razón que actualmente se trabaja en algoritmos conocidos como metaheurísticas, que al igual que los algoritmos de ordenamiento, existen una amplia gama de ellos para realizar objetivos similares.

Los algoritmos heurísticos tienen como objetivo tomar una decisión de dependiendo del método heurístico, de forma estadística, búsqueda de patrones, entre otras técnicas. Un ejemplo claro de estas técnicas son los antivirus actuales para la fecha de este trabajo, ya que utilizan métodos heurísticos para la detección de posibles amenazas dentro de un computador, esto lo logran a través de patrones de comportamiento de cada proceso activo. Por ejemplo, si un proceso inicialmente trata de modificar los módulos de funcionamiento del sistema operativo, el antivirus no sólo lo detendrá sino que además lo tomará como posible amenaza reportándolo a la central de la compañía.

Las metaheurísticas sólo se deben aplicar, cuando no existe un algoritmo satisfactorio para resolver el problema en todos los casos específicos, al igual que cuando sus resultados son ineficientes o inadecuados; existen casos en los que se utiliza esta técnica porque el algoritmo óptimo para resolver un problema no puede ser implementado, sea por la complejidad de analizarlo o por exceso de trabajo en el caso de lenguajes que no tengan múltiples herramientas ya implementadas.

Las metaheurísticas como método algorítmico son ampliamente utilizadas debido a los excelentes resultados que han arrojado en la resolución de problemas de optimización combinatoria de acuerdo a Martí (2003), lo cual se relaciona enteramente con el planteamiento de este estudio, ya que las materias dictadas en una carrera y la

forma de ordenarlas para balancearlas es un problema de combinaciones de dónde colocar cuál materia en cuál período.

Algunas de las metaheurísticas más reconocidas son: Algoritmos voraces, Mejoras iterativas (*iterative improvement), simulated annealing,* búsqueda tabú *(tabu search),* colonia de hormigas, búsqueda esparcida *(scatter search),* programación multiobjetivo, y los más actuales para la fecha son: búsqueda *cuckoo,* búsqueda *mono* y el algoritmo de las luciérnagas. De igual manera, para el año en que se desarrolla esta investigación, en la universidad de Southampton se planteó el modelo creación de metaoptimizadores para los algoritmos y optimizadores utilizados en la resolución de problemas, por lo que el uso de estos métodos metaheurísticos está siendo ampliamente estudiado y de hecho en plena vigencia.

En otro ámbito, la teoría de la complejidad computacional es la rama de la teoría de la computación que estudia, de manera teórica los recursos requeridos durante el procesamiento de un algoritmo. De esta rama se deriva el estudio de complejidad asintótica, que es comprobar de manera teórica el rendimiento que tiene un algoritmo estableciendo una relación entre la cantidad de datos y el tiempo de procesamiento.

Existen múltiples métodos para realizar los análisis asintóticos de los algoritmos, sin embargo todos concluyen en la misma tabla de resultados que van desde una complejidad O(1) siendo ésta la mejor, hasta una complejidad pésima de O(n!), donde sí se grafican ambas funciones, se obtiene que la primera será una línea recta trazada a partir del 1 de forma paralela al eje *x* (Datos) mientras que la última se muestra aumentado sus valores de forma rápida cercana al eje *y* (Tiempo), lo que indica que aunque se tenga poca cantidad de datos a evaluar el tiempo de ejecución será muy alto.

Este tipo de análisis no necesita que se implementen los algoritmos en ningún lenguaje de programación ni en ningún computador, ya que consiste en el análisis puro y científico del algoritmo a través de técnicas matemáticas y el estudio de los límites.

Una parte de este estudio es realizar la comparación entre un método metaheurístico y la programación lineal entera, por lo que es necesario realizar las mediciones a nivel de tiempo y memoria y no a nivel de complejidad asintótica, es sabido que cualquier método heurístico tiene altas complejidades asintóticas.

Sistema de Variables e Hipótesis

Variables

La forma más simple de entender las variables es definirla como la capacidad de cambiar de los objetos, situaciones, personas y otros aspectos que puedan estar involucrados en la investigación, todas estas variables pueden variar como su nombre lo indica y adoptar valores diferentes a los iniciales. Según Briones, G (1995:34) se define que: "Una variable es una propiedad, característica o atributo que puede darse en… grados o modalidades diferentes… Son conceptos clasificatorios que permiten ubicar a los individuos en categorías o clases y son susceptibles de identificación y medición."

Variable Independiente

Metaheurística a implementar**:** Esta variable va a definir la solución para el tipo de problema que se está estudiando, en este trabajo de investigación se utilizará sólo una

metaheurística que resuelva problemas determinísticos, en lugar de los estocásticos.

Variable Dependiente

Resultados en tiempo: De acuerdo al tipo de metaheurística que se implementará, los resultados se deben obtener en tiempos diferentes, la variable dependiente consiste en establecer cuál de las técnicas resuelve este problema en la menor cantidad de tiempo.

Lenguaje: Dependiendo cual lenguaje de programación se utilice para implementar la solución algorítmica, se obtendrán mejores o peores tiempos.

Hipótesis

Las hipótesis son el punto en que se encuentran la teoría y la observación, la importancia de esta radica en que da el rumbo a la investigación y además permite sugerir los pasos y/o procedimientos que deben darse para obtener el conocimiento, especialmente si se trata de la generación de nuevas ideas y conceptos. Para este caso que existe un proceso investigativo es fundamental el uso de hipótesis con el objetivo de afirmarlas al finalizar el trabajo de investigación.

De acuerdo a Laurence, B. (2002:73), es "una afirmación provisional que se pretende verificar recurriendo a procedimientos de análisis. Es una suposición cuyo origen está en la intuición y que queda en suspenso en tanto que no ha sido sometida a la prueba de datos seguros". Según lo anterior, la hipótesis que guiará esta investigación es:

Hi: Un algoritmo, basado en una metaheurística para resolver el problema de balance de carga curricular, requiere un tiempo de ejecución comparable a los de las implementaciones conocidas.

Definición de Términos Básicos

Análisis Asintótico: Consiste en evaluar la calidad de un algoritmo en comparación con otros algoritmos o en comparación a la complejidad del problema o alguna cota de complejidad conocida. Sin embargo, tópicamente el conteo de pasos de computación falta de precisión en el sentido que no es claro que cosas se consideran operaciones básicas. Schaeffer, E. (2011)

Metaheurística: "Procedimientos simples a menudo basados en el sentido común, que se supone que obtendrán una buena solución (no necesariamente óptima) a problemas difíciles de un modo sencillo y rápido". Zanakis, S (1981:331). "Las metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos." define Kelly, J. (1996:174)

CAPÍTULO III

Marco Metodológico

En este capítulo se explicarán las técnicas y métodos necesarios, para llevar a cabo esta investigación por medio de un conjunto de pasos a seguir, los cuales permitirán alcanzar el objetivo de la investigación.

Tipo y Modalidad de la Investigación

"La investigación científica es la búsqueda intencionada de conocimientos o de soluciones a problemas de carácter científico" Murillo, W. (2007). El tipo de investigación que se desarrollará es científico ya que el objetivo es utilizar técnicas actuales para desarrollo de algoritmos en un problema que sólo se ha estudiado de forma científica, asociando a esta investigación, la aplicación práctica.

Este estudio, se ajusta a la modalidad exploratoria y explicativa, ya que Hernández, R (2006:58) define: "Los estudios exploratorios se efectúan… cuando la revisión de la literatura reveló que únicamente hay guías no investigadas e ideas vagamente relacionadas con el problema de estudio". Ya que existen menos de 10 antecedentes internacionales y ningún antecedente nacional. Adicionalmente, los antecedentes extranjeros que se consiguen no están enmarcados en el área de metaheurística de forma directa, por el contrario, se utiliza el método de Programación Lineal Entera para obtener solución al problema planteado.

El presente estudio propone resolver un problema poco explorado, con una técnica algorítmica nunca antes utilizada en este ámbito.

Diseño de la Investigación

En este estudio, se utilizará el método cuantitativo, ya que se tomarán mediciones con respecto a los experimentos y los resultados en tiempo, de la *data* sintetizada se encuentran expresados de manera numérica para establecer comparaciones.

Define Hernández, R (2006:27) que "Los estudios cuantitativos siguen un patrón predecible y estructurado. … La meta principal es la construcción y demostración de teorías.". Este método cuantitativo aunque puede parecer sólo una base de datos, se deslinda de esta apreciación ya que existe un análisis de los mismos, mientras que una base de datos es sólo un conjunto de valores sin interpretación. Otra de las características de este método, es que dado el problema y una vez recolectados los datos, es posible definirlo y saber los límites del mismo, desde dónde se inicia hasta cuál es el objetivo y en qué dirección se debe orientar. El método cuantitativo es muy útil en términos de validez externa, ya que las pruebas numéricas generalmente no son refutables si los mecanismos de recolección de datos está bien establecidos y quedan claro para cualquier investigador que a posteriori utilice el estudio para continuar la investigación.

El diseño de la investigación será experimental, porque se hace un análisis y manipulación de las variables mencionadas en el capítulo II. De acuerdo a Hernández, R (2006:188) "Los experimentos "auténticos o puros" manipulan variables independientes para ver sus efectos sobre variables dependientes en una situación de

control.” Es por esto que este estudio, se encuentra enmarcado en la definición de experimento puro, esto se debe a que una vez definidas las variables en el Capítulo II, se establece que al implementar diferentes técnicas metaheurísticas (variable independiente), se obtendrá tiempos de respuesta diferentes de las soluciones al problema planteado, el tiempo sería la variable dependiente.

Población y Muestra

Según Arias, J (2006:81) “la población es el conjunto finito o infinito de elementos con características comunes, para los cuales serán extensivas las conclusiones de la investigación. Esta queda limitada por el problema y por los objetivos del estudio.”

Para el caso específico de esta investigación, se requiere el uso de población, cuya muestra es la población completa, utilizando el caso específico de estudio de la Facultad Experimental de Ciencias y Tecnología de la Universidad de Carabobo (FaCyT-UC). Por otra parte para el estudio se utilizarán también casos de prueba sintetizados, es decir, casos hipotéticos, obtenidos de la librería CSPLib (*Constraint Problem Library*), la cual es ampliamente utilizada por tener las definiciones, modelos y casos de prueba de múltiples problemas de restricciones.

Tomando en cuenta el caso de la FaCyT-UC entonces se podría calificar como muestra la recolección de datos de las materias que se desean dictar en cada carrera dentro de la Facultad como un subconjunto de las materias que se dictan dentro de la Universidad de Carabobo, y el universo muestral se desarrolla dentro del ambiente de la Facultad.

Se considera una muestra cuantitativa ya que la obtención de estos datos permitirá realizar una simulación utilizando el *software* final de esta investigación, para establecer si se llegó a una solución válida o si deben colocarse nuevas restricciones dentro del modelo para ajustar mejor a la realidad. Los datos recolectados de muestras cuantitativas llevan a un análisis directo y conciso de la realidad del universo que se está analizando. De acuerdo a Arias, J (2006:83) “La muestra se define como el subconjunto representativo y finito que se extrae de la población accesible.”

Técnicas e Instrumentos de Recolección de Datos

Falcón, J. (2005:12) se refieren al respecto que “se entiende como técnica, el procedimiento o forma particular de obtener datos o información”.

La aplicación de una técnica conduce a la obtención de información, la cual debe ser resguardada mediante un instrumento de recolección de datos.

Instrumento de Recolección de Datos. Establecen además que "son dispositivos o formatos (en papel o digital), que se utiliza para obtener, registrar o almacenar información".

Para esta investigación se utilizará la búsqueda de casos de prueba sintetizados en repositorios como CSPLib, mediante el cual se obtiene una gran cantidad de información específica para este problema y otros de combinatoria. La consulta por *Internet* para este trabajo será la herramienta a utilizar para conseguir los datos sintetizados.

Al momento de tomar una muestra, se debe tener en cuenta la confiabilidad y la validez, Hernández, R (2006). Los datos a utilizar son sintetizados con el objetivo de probar escenarios exhaustivos y complejos ante el *software* que la procese. Razón por la cual, se toman estos datos como confiables y válidos.

Al momento de tener todos los datos recolectados y una vez ejecutado el *software* resultante de este trabajo de investigación, se establecerán como técnicas de procesamiento de los datos, tablas, cuadros, gráficas y mediciones comparativas del método propuesto en este estudio, con respecto a los utilizados previamente para resolver el problema de balanceo de carga curricular, con el objetivo de demostrar cuál de los métodos es el que arroja mejores resultados finales.

Análisis de los datos

A nivel de la investigación según Sabino, C. (2000:451) "el análisis cuantitativo se define como una operación que se efectúa, con toda la información numérica resultante de la investigación. Esta, luego del procesamiento… se nos presentará como un conjunto de cuadros y medidas, con porcentajes ya calculados". Esto permitirá sacar porcentajes y representar gráficamente los resultados de los datos obtenidos para tener la información ordenada con representaciones visuales que nos permitan su posterior estudio.

Procedimientos Previstos

Se realizará un trabajo comparativo entre los resultados obtenidos de la aplicación

final desarrollada y los resultados de las investigaciones previas referidas en los antecedentes, con el objetivo de demostrar que la hipótesis planteada es satisfecha. Los procedimientos previstos se deben ejecutar en orden de etapas para no perder el sentido de la investigación:

1. Identificar los elementos fundamentales del problema: En este objetivo, los pasos a realizar son: determinar la función objetivo y determinar el número de variables que pueden influir en la solución.
2. Diseñar el modelo a utilizar para resolver el problema de balanceo de carga curricular: En este punto, se debe plantear formalmente la función objetivo.
3. Establecer las diferencias entre las técnicas utilizadas para resolver el problema y la técnica que se aplicará en este trabajo de investigación: Es importante delimitar al momento de decidir la técnica para resolver el problema de balanceo de carga curricular, que no haya sido previamente utilizada y que diferencia a las técnicas anteriores de una nueva. En este caso en particular, es que no es una técnica exacta, sino heurística.
4. Determinar la metaheurística adecuada para resolver de forma apropiada el problema planteado: no todas las metaheurísticas dan resultados buenos para todos los problemas, es por esta razón que existen múltiples, por lo que hay que identificar a través de los elementos definidos en el primer objetivo, cuáles podrían ser resueltos a través de una técnica específica.
5. Determinar las variantes que se pueden aplicar al algoritmo base: En este punto se debe estudiar si el algoritmo es paralelizable, o si puede ser aplicado desde distintos puntos de la estructura general.
6. Desarrollar el algoritmo heurístico de solución del problema de balanceo

de carga curricular.: En este objetivo, la actividad es desarrollar el algoritmo completo utilizando las variantes definidas y cumpliendo con los objetivos planteados.

7. Desarrollar el *software* y evaluarlo con casos de prueba sintetizados para determinar su correcto funcionamiento: En esta meta, se propone implementar el algoritmo planteado en el paso anterior, sobre un lenguaje de programación definido. Posteriormente, utilizar los casos de prueba establecidos por la CSPLib y los otorgados por la FaCyT-UC para evaluar si los resultados están correctos.
8. Comparar resultado de tiempo, entre las variantes implementadas y el planteamiento de esta solución: Este paso consiste en compilar y ejecutar los códigos fuentes de los investigadores mencionados en los antecedentes, en el mismo computador que la solución desarrollada en este trabajo para poder establecer tablas y gráficas comparativas.

El presente estudio concluye con la presentación de una aplicación y un informe que reporta los resultados comparativos de, los estudios referenciados en el capítulo II en la sección de antecedentes, con respecto al actual, por lo que es simple dados estos datos establecer la comparación numérica de las investigaciones con respecto al tema. Estas comparaciones se llevarían a cabo sólo a nivel de tiempo de ejecución, de acuerdo con lo planteado en la sección de variables del capítulo anterior.

ASPECTOS ADMINISTRATIVOS

En este apartado se presentan el análisis costo-beneficio y el cronograma de actividades a desarrollar para su finalización.

Análisis Costo-Beneficio

Recursos Materiales

Para realizar el estudio de investigación propuesto, se requiere de pocos materiales, inicialmente de un computador con al menos 2 Ghz de Procesador, 1 GB de Memoria RAM, el acceso a *Internet* es necesario pero no frecuente, para poder descargar los casos de prueba del repositorio. Es fundamental una impresora para presentar los resultados previos, códigos y el presente documento. Finalmente se requiere aproximadamente de tres resmas de hojas blancas base 20, tamaño carta.

Recursos Humanos

No se requiere la contratación de ningún personal para poder llevar a cabo esta investigación. El único recurso humano a utilizar, son dos tutores.

–El tutor académico del proyecto de investigación: Dr. Amadís Martínez

–El tutor metodológico del proyecto de investigación: Prof. Marlene Figueredo

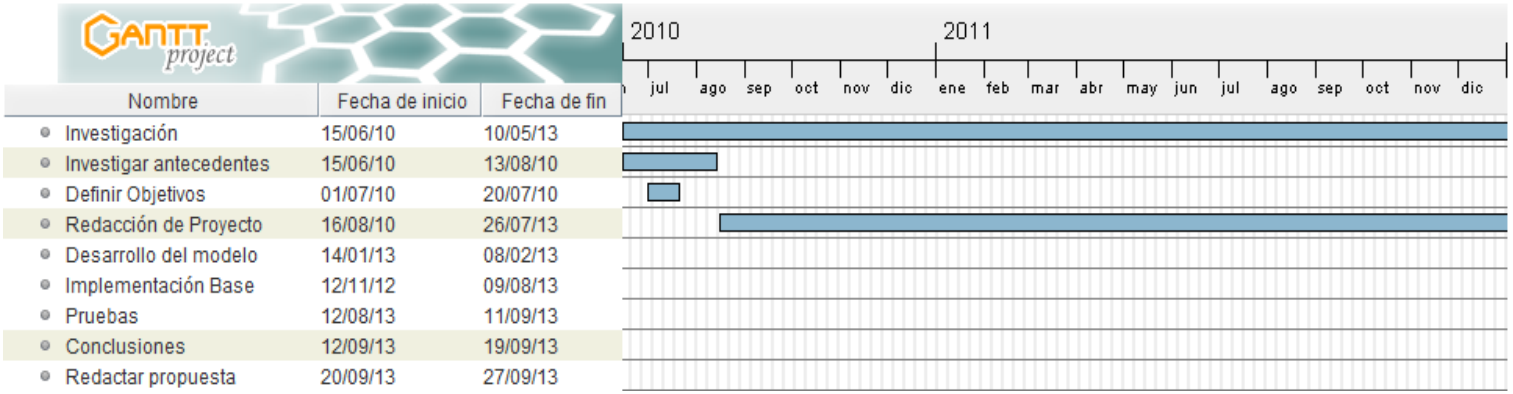
Recursos Institucionales

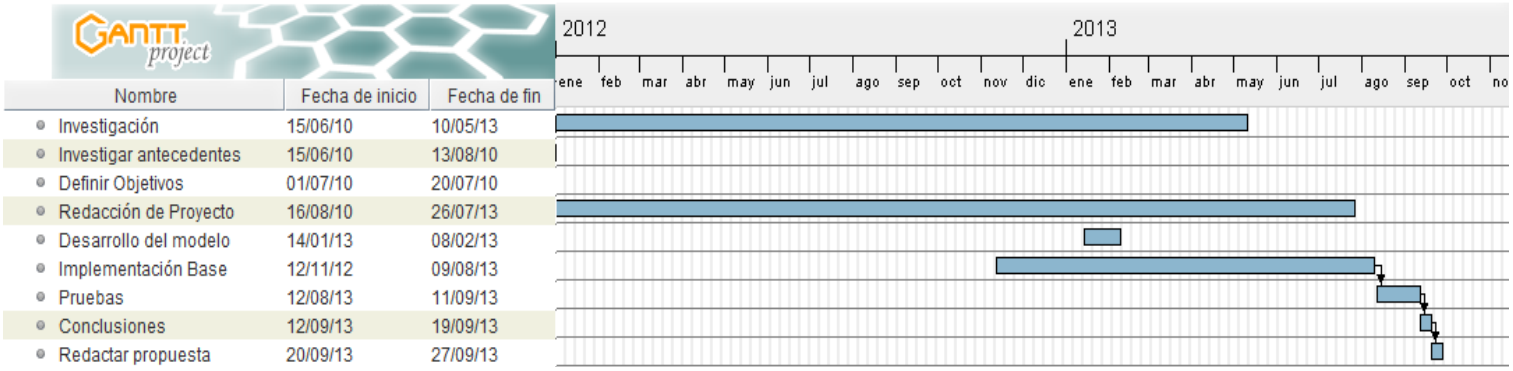
Se solicita de la Facultad Experimental de Ciencias y Tecnología de la Universidad de Carabobo (FaCyT-UC), acceso al *pensum* de todas las carreras ofertadas junto con sus notas informativas, prelaciones y unidades de crédito, la respuesta de la FaCyT-UC ha sido positiva.

Cronograma de Actividades

En la siguiente página se presenta el cronograma de actividades, a través de un diagrama de Gantt, esta ilustración fue realizada con el *software* de licencia GNU llamado Gantt Project

Tabla 1: Diagrama de Gantt 2010-2013





Fuente: Elaboración propia

CAPÍTULO IV

RESULTADOS

Con la finalidad de alcanzar los objetivos específicos de esta investigación, se realizaron un conjunto de pruebas, orientadas a determinar la información necesaria para la elaboración de la comparación que permita determinar la calidad de la solución, junto al tiempo de respuesta para el problema de balanceo de una carga curricular. En este sentido, se muestra a continuación el análisis e interpretación de los resultados, derivados de cada uno de los objetivos de estudio.

Configuración de los Experimentos

Plataforma Computacional

Los experimentos fueron realizados en un computador de escritorio, cuyas especificaciones técnicas están detalladas en la Tabla 2.

Especificaciones Técnicas	
Sistema Operativo	Ubuntu 12.04 Kernel 3.2.0-48
Procesador	Intel i7-2600K 3.4 GHz x 8 núcleos
Memoria RAM	8 GB DDR3 1600 MHz
Disco Duro	750 GB en Disco Duro de 7200 RPM
Sistema de Archivos	Ext4

Tabla 2: Características del equipo

El *software* utilizado para realizar las pruebas de la aplicación fue: Lenguaje C++ con compilador g++ versión 4.6.3 con un solo *flag* de optimización -O2.

Formato de Archivo de Entrada

En la tabla 3, se muestra el formato de entrada correspondiente a cada uno de los casos de prueba, como se puede detallar, se especifica la información necesaria para procesar cada una de las instancias a estudiar. En la primera línea se encuentra el número de años, luego la cantidad de períodos por año, lo que indica la cantidad de períodos a formar no es más que el total de años multiplicado por los períodos, seguidamente el número de asignaturas, cantidad de *curricula*, la carga mínima y máxima por período respectivamente, cantidad de prelaciones, cantidad de períodos no deseados o lo que se denomina como preferencias; este último, consiste en que si una materia tiene como requisito, no estar antes de determinado período, puede colocarse a través de este parámetro, escribiendo el código de la asignatura y el período donde no debe estar. Concluida esa parte de la información, se procede a la lectura de los códigos de las asignaturas con la cantidad de unidades de crédito asociadas a la misma, para luego seguir con la distribución de las materias en cada uno de los *curricula* a construir, donde primero se coloca el nombre del *curriculum*, luego un N para indicar la cantidad de asignaturas y luego la lista de códigos. De inmediato, sigue la lista de prelaciones y el detalle de los períodos indeseados para cada una de las asignaturas.

El formato del archivo de entrada, establecido en la CSPLib, no es una propuesta de rápido acceso a los datos, ya que en cada archivo se plantea balancear *N curricula*, especificando cuales códigos de asignaturas pertenecen a cada carrera, sin embargo, todas las prelaciones aparecen sin discriminar a cuál carrera pertenecen. Es por ello,

que en cada oportunidad que se va a balancear una carrera, el primer paso, es recorrer todas las prelaciones establecidas en el archivo de entrada para verificar si pertenecen o no. Lo cual, concluye en tantos accesos a memoria secundaria, como *curricula* se planteen en el archivo. Para evitar realizar múltiples accesos a memoria secundaria, ya que puede alterar los resultados, se diseñó una estructura, que almacena todas las prelaciones y se recorre por cada *curriculum* que se desea armar, la ventaja es que se encuentra en memoria principal.

```
YEARS: 2
PERIODS_PER_YEAR: 2
NUM_COURSES: 6
NUM_CURRICULA: 3
MIN_MAX_COURSE_LOAD_PER_PERIOD: 0 2
NUM_PRECEDENCES: 3
NUM_UNDESIRED_PERIODS: 2

COURSES:
A 5
B 5
C 5
D 5
E 18
F 18

CURRICULA:
C1 4 A B C D
C2 4 B C D F
C3 4 A B E F

PRECEDENCES:
D C
D E
C F

UNDESIRED_PERIODS:
B 0
A 1
```

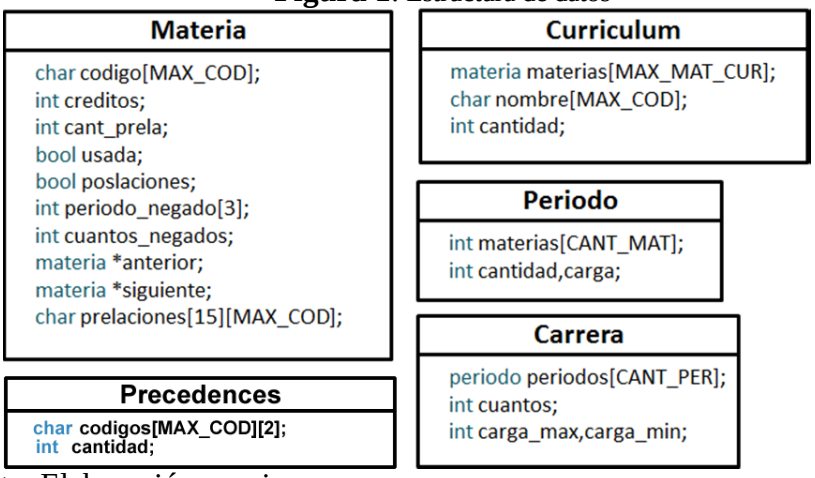
Tabla 3: Formato del archivo de entrada

Se puede observar en el tabla 3, para el caso de la prelación C F, que esta no se encuentra en la carrera C1 ni C3, sin que haya ninguna discriminación para ello en el archivo. Por lo que hubo que realizar un procedimiento aparte para discriminar cuales eran las prelaciones correctas en cada uno de los *curriculum* planteados. Esto será explicado en la sección de estructura de datos.

Estructura de Datos

La estructura de datos usada para esta investigación, fue diseñada con el fin de reducir la cantidad de accesos a memoria secundaria, motivo por el cual fue necesario alojar la información de las prelaciones en memoria principal, en una estructura aparte denominada **precedences**, ofreciendo un gran beneficio en este caso, ya que la información es de frecuente acceso. De igual forma, se presenta una estructura sencilla como se muestra en la Figura 1.

Figura 1: Estructura de datos



Fuente: Elaboración propia.

Como puede observarse en la figura 1, se maneja un registro llamado **materia** en donde estará contenida toda la información referente a cada una de las asignaturas del

pensum de estudio, como son: código de identificación, cantidad de unidades de crédito, cantidad de prelaciones, cantidad de períodos donde no deben estar, entre otros. Existen dos apuntadores, llamados anterior y siguiente, estos permiten crear una estructura dinámica de **niveles** en las materias, a través de un procedimiento que realiza los cálculos para ordenar las materias por unidades académicas. Por ejemplo, existen en el *pensum* las siguientes materias: Cálculo I, Cálculo II, Algoritmos I, Algoritmos II, Cálculo III, listados en ese orden, si la primera prelación listada, es por unidad académica, el procedimiento es capaz de armar en la estructura lo que se conoce como **niveles**, obteniendo entonces, un nivel con las materias: Cálculo I, Cálculo II y Cálculo III y otro nivel que contiene las materias: Algoritmos I y Algoritmos II. Esto, con el objetivo de reducir la cantidad de elementos a combinar, mientras más extenso el nivel, menor es la combinatoria que se debe aplicar para conseguir la carga balanceada. En el procedimiento principal se crea un arreglo para almacenar todas las materias del archivo de entrada.

La estructura de **curriculum**, se encuentra a manera de registro, cuya información corresponde a la cantidad de materias, un arreglo de las materias y el identificador o nombre del *curriculum*, es importante resaltar que esta estructura, contiene la información completa de todas las materias del *curriculum* que se desea armar, ordenadas por aparición en el archivo de entrada.

Para el registro de **periodo**, puede observarse que, es mucho más sencillo de comprender que el anterior, como su nombre lo indica, contiene los datos de cada periodo asociado al *curriculum*, indicando las materias, cantidad de materias y la unidad de créditos total para el mismo.

La estructura llamada **carrera**, está basada en **curriculum**, con la diferencia, de que en este caso, se almacenan las materias organizadas en cada período y se tiene la información de la carga máxima, mínima y la que tiene en realidad cada período. Todas las cargas se calculan en función de las unidades de crédito de las materias que se incluyen en el periodo.

Idea General del Algoritmo metaheurístico

Lo primero que realiza el *software* desarrollado, es cargar en memoria principal todas las carreras indicadas en el archivo de entrada, en un arreglo de la estructura **curriculum**, luego, carga todas las prelaciones en la estructura **precedences**, posteriormente, realiza un análisis de todas las prelaciones para identificar cuáles pertenecen a cada carrera, para optimizar la búsqueda, se realiza en un ciclo con cuatro índices, que revisan en simultáneo toda la estructura **precedences**; estos cuatro índices, lo que hacen es dividir el arreglo en cuatro partes. Por ejemplo, si se señalaran 400 materias, entonces, el índice 1 buscaría del 1 al 100, el índice 2 del 101 al 200, el índice 3 del 201 al 300 y el índice 4 del 301 al 400. Esto lleva a que se realicen máximo 100 iteraciones en lugar de 400 para hacer la búsqueda de prelaciones.

Al momento de que alguno de los cuatro índices identifique que la prelación pertenece a la carrera, se procede a procesar la información para armar los **niveles** en la estructura **nivel**, que es un arreglo de listas dinámicas, en las cuales se busca ir ordenando las materias, como se mencionó en la sección Estructura de Datos. Algunas de las materias se consideran independientes, ya que no tienen prelaciones ni son prelaciones de otras materias, y estas representan un arreglo aparte. En el caso de que

una materia tenga más de una prelación, como la estructura es una lista simple, se crea un nuevo nivel con esta materia, dejando la marca de que esa asignatura, tiene al menos una prelación que se encuentra en otro nivel.

Luego de que quedan establecidas las prelaciones, se aplica un procedimiento *greedy*, el cual se basa en colocar materias de acuerdo a los niveles más extensos en los períodos y sin excederse de las unidades de crédito permitidas en cada período. Luego, coloca las materias independientes, buscando colocar en los períodos menos cargados, las materias de mayor carga. Esto genera una solución inicial que cumple con al menos la restricción de las prelaciones y los límites de las unidades de crédito.

Posteriormente, se aplica un paso de selección aleatoria, en la cual, todas aquellas materias que no tienen prelaciones, se entienden como base de nivel, se ubican de forma aleatoria en el primer semestre, a partir de ese punto, se utiliza de nuevo otra selección aleatoria para decidir si se coloca o no, la materia siguiente con respecto al nivel en el período siguiente. La clave de la decisión es la cantidad de unidades de crédito restantes, este procedimiento se aplica por 400 iteraciones mientras no se consigan mejores resultados entre ellas. En caso de haber mejor resultado, se vuelve a iniciar el contador en 400 iteraciones, lo que asegura la búsqueda de mejor solución en tiempos considerablemente bajos con respecto a otras investigaciones.

Esto lleva a que el algoritmo utilizado, haya partido de una técnica metaheurística conocida como GRASP, sin embargo, es importante recordar que esta técnica se aplica después de que los datos han sido procesados y optimizados para reducir el número de combinatorias.

Algoritmos utilizados

En esta sección, en primera instancia se listan las operaciones utilizadas, junto a una breve descripción y luego se describen los principales algoritmos utilizados para el desarrollo de la solución, como resultados algorítmicos de esta investigación. El lenguaje utilizado para expresar el algoritmo es NASPI. Martínez (2010)

Listado de operaciones

Las operaciones de lista son:

– Proc inicializacion(Lista nivel): Inicializa la lista

– Lógico es_vacia(Lista nivel): Indica si una lista está vacía

– Proc ins_en_lista_vacia(Lista nivel, materia x): Inserta en lista vacía

– Proc ins_inicio_lista(Lista nivel, materia x): Inserta al inicio de la lista

– Proc ins_fin_lista(Lista nivel, materia x): Inserta al final de la lista:

– Proc ins_despues(Lista nivel, materia x, entero pos): Inserta después de determinada posición

– Proc muestra(Lista nivel): Imprime la lista

– materia primero(Lista nivel): Retorna la primera materia de una lista

– Proc materia_valida(Lista nivel): Marca a todas las materias del nivel como usadas en una lista.

Las operaciones de materias son:

– Proc. crear_prelacion(materia x, materia p): La materia P, prela a la materia X

– Func. consulta_posicion_materia(Lista nivel, cadena codigo): Entero. Consulta la posición de una materia.

– Proc buscar(curriculum vector, cadena codigo_prela, cadena codigo_materia, entero prela, entero mate): Este procedimiento devuelve las 2 posiciones en las que se encuentra, 1 materia y 1 prelación de esa materia sobre el arreglo de materias.

Las operaciones de curriculum son:

– Proc inicializar_materias(curriculum activo): Permite trabajar sobre un nuevo curriculum

– Func. buscar_curriculum(cadena codigo_prela, cadena codigo_materia, curriculum activo): Lógico. Este procedimiento ubica si una materia y su prelación se encuentran dentro del curriculum

– Func. buscar_mover(entero temp,curriculum cur, carrera car, entero periodo): Entero. Sugiere a cuál período debería moverse una materia.

– Func. buscar_posicion_materia(materia x,curriculum cur): Entero. Dada una materia, busca su posición dentro del arreglo del curriculum.

– Func. buscar_posicion_nombre(cadena x,curriculum cur): Entero. Dado el nombre de una materia x, busca su posición dentro del arreglo de curriculum.

– Proc. asignar_UC(curriculum activo, materia todas[1..N], entero cuantas_traigo). Este procedimiento realiza una búsqueda con 4 índices, sobre el arreglo de todas las

asignaturas, de manera que si consigue el nombre de una asignatura que pertenezca al curriculum, entonces asigna inmediatamente todas las características de las asignaturas correspondientes. El objetivo de los 4 índices, es hacer la cuarta parte de las iteraciones como máximo.

– Proc. crear_curriculum(curriculum activo, materia todas[1..N]). Este procedimiento, permite recorrer todo el arreglo de prelaciones (el cual tiene el orden de aparición del archivo de entrada), y permite ubicar si la prelación corresponde a ese curriculum o no, dependiendo de si ambas materias se encuentran señaladas y en caso afirmativo, incluir esta prelación dentro del curriculum activo. Adicionalmente, permite crear los niveles, identificando si hay que hacer un nuevo nivel o no, y en cuál posición debería ir cada materia dentro de ese nivel.

Las operaciones de carrera son:

– Func. verificar_todas(Lógico incluida[1..N],carrera car,curriculum cur, entero mayor, entero j): Lógico. Esta es una función de validación de que todas las materias del curriculum, fueron incluidas en la carrera.

– Func. sugieremenor(carrera car, entero mayor): Entero. Esta función ubica el período con menor carga y lo retorna para indicar el menor.

– Func. verif_negado(entero peri, carrera car, curriculum cur, entero pos): Lógico. Esta función indica si el período donde se desea incluir una asignatura, pertenece a las restricciones de ubicación de dicha asignatura.

– Proc. ins_independientes(curriculum cur, carrera car, entero minimo, entero maximo, real promedio). Este procedimiento, permite, que una vez insertadas las asignaturas que tienen o son prelaciones en la carrera, cumpliendo con el principio de

carga máxima y mínima, se insertan las materias que no tienen restricciones de prelaciones, consideradas independientes.

– Proc. ins_rezagados(curriculum cur, curriculum orig, carrera car, entero minimo, entero maximo, real promedio). Debido a las características del problema, en algunos escenarios, al momento de crear los niveles, algunas materias pueden tener múltiples prelaciones, por lo que pudiese no insertarse en el curriculum en una primera pasada ya que falta insertar el resto de los niveles para determinar dónde podría ubicarse. Este procedimiento, inserta esas materias que tienen o son prelaciones, pero que dadas sus características no se pudo insertar en un primer momento, partiendo del principio de balance.

– Proc. heur(curriculum cur, carrera car, entero carga_min, entero carga_max, entero promedio). Este procedimiento, permite reasignar con el objetivo de balancear, las materias independientes, es decir, este procedimiento no modifica la posición de aquellas asignaturas que tienen prelaciones o son prelaciones.

– Func. evaluador(carrera car, curriculum cur, entero promedio): Real. Esta función permite evaluar una carrera ya construida. Se fundamenta en la siguiente fórmula. exp(carga_periodo-carga_promedio+1)/2 en caso de que la carga sea mayor al promedio. En caso de ser menor, la fórmula es exp(carga_promedio-carga_periodo)/2. En caso de ser igual la carga_promedio a la carga_periodo, se suma 1. De esta manera, cuándo un período esté mucho más cargado o descargado, de lo calculado para el promedio, se penaliza con función exponencial, asegurando así, que el valor resultante del evaluador, sea fácil de pasar por una solución mejor. A menor valor, mejor resultado.

– Func. validador(carrera actual, curriculum cur): Lógico. Esta función verifica que todas las asignaturas del curriculum, se encuentran en la carrera balanceada.

– Func. ver_prelaciones(carrera car, entero pos, entero semestre): Lógico. Verifica que todas las prelaciones de una asignatura, ya fueron colocadas en la carrera.

Las operaciones de nivel son:

– Proc. ins_nivel_sin_prel_carrera(Lista nivel, curriculum cur, carrera car, entero minimo, entero maximo, real promedio, curriculum rezagados). Al momento de armar los niveles, la primera materia que lo conforma podría o no tener una prelación; en caso de no tener, se llama a este procedimiento, para insertar todo el nivel en la carrera. Dentro del procedimiento se determina a partir de cuál semestre se puede insertar este nivel. En caso de que alguna materia no se pueda insertar, pasa a quedar como rezagada y se procesa con el procedimiento ins_rezagados en otro paso del algoritmo general.

– Proc. ins_nivel_con_prel_carrera(Lista nivel, curriculum cur, carrera car, entero minimo, entero maximo, real promedio, curriculum rezagados). Al momento de armar, los niveles, la primera materia que lo conforma, podría tener una prelación, en este caso se llama a este procedimiento para insertar todo el nivel en la carrera.

Desarrollo de Algoritmos

Los algoritmos que se colocan en esta sección, son aquellos que forman parte de la heurística y el balanceo, además del algoritmo principal.

Algoritmo General.

```
Proc. Principal()
    //declaración de variables locales
    inicializar_niveles()
    carga_archivos()
    para i ← 1 hasta cant_curriculum en 1 hacer
        inicializar_materias(cur_activo)
        asignar_UC(cur_activo,todas_materias,cant_materias)
        //creación de niveles
        para j ←1 hasta cant_materias en 1 hacer
            si (buscar_curriculum(materia,prelacion,cur_activo) entonces
                buscar(cur_activo,prelaciones,pos_prelacion,pos_materia)
                crear_curriculum((cur_activo, materia todas[])
            fsi
        fpara
        inicializar_carrera()
        si (cant_niveles=0) entonces
            ins_independientes(curriculum_activo,carrera_actual)
        fsi
        mejor_puntuacion←∞
        // Construcción de solución inicial
        para ind_nivel ← 1 hasta cant_niveles en 1 hacer
            repetir
                tj←ti
                si(nivel[tj].cant_prela≤0) entonces
                    ins_nivel_sin_prelacion(nivel[tj],carrera_actual)
                sino
                    si(nivel[tj]≠nulo) entonces
                        ins_nivel_con_prelacion(nivel[tj],carrera_actual)
                    fsi
                fsi
                si (tj=ind_nivel) entonces
                    tj←0
                fsi
                tj←tj+1
            mientras(tj≠ti)
        fpara
        ins_rezagados(rezagados,cur_activo,carrera_actual)
        ins_independientes(cur_activo,carrera_actual)
        heur(cur_activo,carrera_actual)
        mejor_puntuacion<--evaluador(carrera_actual)
        //Fin de solución inicial
        si (cant_niveles > 0) entonces
            k←0
            // Se pueden hacer N iteraciones en búsqueda de la mejor solución
            mientras (k < N) hacer
                cur_temp←cur_activo
                car_temp←carrera_actual

                //Cuando se crea la solución inicial, los niveles se colocan
                //orden que fueron creados. Entonces se les coloca de manera
                //aleatoria ahora el orden, para buscar nuevas soluciones.

                mezcla_aleatoria(orden_niveles)
                k←k+1
                para tj← 0 hasta ind_nivel en 1 hacer
                    si(nivel[tj].cant_prela≤0) entonces
```

```
                    ins_nivel_sin_prelacion(nivel[tj],car_temp)
                sino
                    si(nivel[tj]≠nulo) entonces
                        ins_nivel_con_prelacion(nivel[tj],car_temp)
                    fsi
                fsi
            fpara
            ins_rezagados(rezagados,cur_temp,car_temp)
            ins_independientes(cur_temp,car_temp)
            heur(cur_temp,car_temp)
            si (evaluador(car_temp)<mejor_puntuacion) entonces
                mejor_car←car_temp
                mejor_puntuacion<--evaluador(car_temp)
                k←0
            fsi
            k←k+1
        fmientras
    fsi
    mostrar_resultados(mejor_car)
    liberar_memoria()
 fpara
fproc
```

Algoritmo para insertar materias rezagadas

```
Proc. Ins_rezagados(rezagados[],cur_actual,car_actual,carga_min,carga_max,promedio)
   //declaración de variables locales
   mientras(i<cant_materias^cont≠cant_materias)
      si(¬cur_actual.materia.usada) entonces
         //Se utiliza al recomendador para que dependiendo de la carga de la materia
         //se pueda ubicar en el menor periodo posible. Luego se itera buscando.
         apartirde←recomendador(cur_actual.materia,cur_actual)
         si(apartirde=-2) entonces
            si(cur_actual.materia.credito>promedio) entonces
               menor←car_actual.periodo[apartirde].carga
               ind_per←0
               para j ← apartirde+1 hasta cant_peri en 1 hacer
                  si(menor>car_actual.periodo[j].carga) entonces
                     menor← car_actual.periodo[j].carga
                     ind_per←j
                  fsi
                  car_actual.periodo[ind_peri].materia←rezagados[i]
                  cur_actual.materia.usada←verdadero
                  i←0
                  cont←cont+1
               fpara
            sino
               ind_per←apartirde-1
               repetir
                  ind_per←ind_per+1
               mientras(restricciones_permitan(cur_actual,car_actual))
               car_actual.periodo[ind_peri].materia←rezagados[i]
               cur_actual.materia.usada←verdadero
               i←0
               cont←cont+1
            fsi
         fsi
      fsi
      i←i+1
      si(i=cur.cantidad) entonces
         i←0
      fsi
   fmientras
fproc
```

Algoritmo para crear un curriculum

```
Proc. crear_curriculum((cur_activo, materia todas[])
   //Si materia fue usada y prelacion tambien entonces no
   //Se debe hacer nada en niveles.
   si (materia.usada ^ prelacion.usada) entonces
      crear_prelacion(cur_activo,materia,prelacion)
   fsi

   //Si materia fue usada y prelacion no, entonces hay que
   //verificar si materia tiene alguien antes en su nivel, en
   //caso de no, entonces se coloca en el mismo nivel, sino en otro
   si (materia.usada ^ ⌐prelacion.usada) entonces
      ind_nivel← -1
      repetir
         ind_nivel← ind_nivel+1
         si(nivel[ind_nivel]≠nulo) entonces
              pos_materia←consultar_posicion_materia(nivel[ind_nivel],prelacion)
              si(pos_materia=0) entonces
                ins_inicio_lista(nivel[ind_nivel],cur_actual,prelacion)
                prelacion.usada←verdadero
                crear_prelacion(cur_actual,materia,prelacion)
              fsi
         fsi
      mientras(pos_materia=nivel[ind_nivel].tamaño+1)
      si(pos_materia≠0) entonces
         cant_nivel←cant_nivel+1
         ins_en_lista_vacia(nivel[cant_nivel],prelacion)
         prelacion.poslacion←verdadero
         crear_prelacion(cur_actual,materia,prelacion)
         prelacion.usada←verdadero
      fsi
   fsi

   //Si prelacion fue usada pero materia no, hay que insertar al final
   // del nivel.
   si (⌐materia.usada ^ prelacion.usada) entonces
      ind_nivel←-1
      repetir
         ind_nivel← ind_nivel+1
         si(nivel[ind_nivel]≠nulo) entonces
              pos_materia←consultar_posicion_materia(nivel[ind_nivel],materia)
              si(pos_materia=nivel[ind_nivel].tamaño-1) entonces
                ins_fin_lista(nivel[ind_nivel],cur_actual,materia)
                materia.usada←verdadero
                crear_prelacion(cur_actual,materia,prelacion)
              fsi
         fsi
      mientras(pos_materia=nivel[ind_nivel].tamaño+1)
   fsi

   //Si ninguno fue utilizado entonces se colocan ambos en un nuevo nivel
   si (⌐materia.usada ^ ⌐prelacion.usada) entonces
      cant_nivel← cant_nivel+1
      ins_en_lista_vacia(nivel[cant_nivel],prelacion)
      prelacion.usada←verdadero
      ins_despues(nivel[cant_nivel],prelacion,materia)
      materia.usada←verdadero
```

```
        crear_prelacion(cur_actual,materia,prelacion)
    fsi
fproc
```

Algoritmo para insertar materias independientes

```
proc. ins_independientes(cur,car,minimo,maximo,promedio)
   //declaración de variables locales

   temp_car←car
   cont←contar_materias_independientes(car,cur)
   list←listar_materias_independientes(car,cur)

   mezcla_aleatoria(list,cont)
   para i←0 hasta cont en 1 hacer
      l←buscar_posicion_materia(list[i],cur)
      si(list[i].creditos>promedio) entonces
         si(list[i].cuantos_negados≤0) entonces
            menor←temp_car.periodos[0].carga
            peri ← 0
            para j←0 hasta temp_car.cuantos en 1 hacer
               si(menor>temp_car.periodos[j].carga) entonces
                  menor←temp_car.periodos[j].carga
                  peri←j
               fsi
            fpara
         sino
            menor←temp_car.periodos[0].carga
            peri ← 0
            para j←0 hasta temp_car.cuantos en 1 hacer
               si((menor>temp_car.periodos[j].carga)^
                  (verif_negado(j,temp_car,cur,l))) entonces
                     menor←temp_car.periodos[j].carga
                     peri←j
               fsi
            fpara
         fsi
         temp_car.periodos[peri].materias[temp_car.periodos[peri].cantidad] ←l
         temp_car.periodos[peri].cantidad← temp_car.periodos[peri].cantidad+1
         temp_car.periodos[peri].carga←temp_car.periodos[peri].carga+
                                       list[i].creditos
      sino
         si(list[i].cuantos_negados≤0) entonces
            menor←promedio
            peri ← 0
            para j←0 hasta temp_car.cuantos en 1 hacer
               si(menor>temp_car.periodos[j].carga) entonces
                  menor←temp_car.periodos[j].carga
                  peri←j+1
               fsi
            fpara
            si(temp_car.periodos[peri].carga>promedio) entonces
               peri←temp_car.cuantos
               repetir
                  peri←peri-1
               mientras(((temp_car.periodos[peri].cantidad>maximo) v
                        (temp_car.periodos[peri].carga≥promedio)) ^
                        (verif_negado(peri,temp_car,cur,l)))
         sino
            menor←promedio
            peri ← 0
            para j←0 hasta temp_car.cuantos en 1 hacer
               si((menor>temp_car.periodos[j].carga)^
```

```
                    (verif_negado(j,temp_car,cur,l))) entonces
                        menor←temp_car.periodos[j].carga
                        peri←j+1
                    fsi
                fpara
                si(temp_car.periodos[peri].carga>promedio) entonces
                    peri←temp_car.cuantos
                    repetir
                        peri←peri-1
                    mientras(((temp_car1.periodos[peri].cantidad>maximo)v
                            (temp_car1.periodos[peri].carga≥promedio))^
                            (verif_negado(peri,temp_car,cur,l)))
                fsi
            fsi
            temp_car.periodos[peri].materias[temp_car.periodos[peri].cantidad] ←l
            temp_car.periodos[peri].cantidad←temp_car.periodos[peri].cantidad+1
            temp_car.periodos[peri].carga←temp_car.periodos[peri].carga +
                                        lrcl[i].creditos
        fsi
    fpara
    car ← temp_car
fproc
```

El algoritmo de insertar materias independientes, consiste en colocar asignaturas que no tienen prelaciones ni son prelaciones de otras, en los mejores períodos posibles. Busca inicialmente revisar la carga del periodo y compararla con el promedio, el cual se calcula sumando todas las unidades de crédito de un semestre y dividiéndolo entre el número de periodos. El procedimiento ubica el período con menor carga y allí coloca de manera *greedy* la materia que está evaluando.

Algoritmo heurístico

```
proc heur(cur, car, carga_min, carga_max, promedio)

    //declaración de variables locales

    carrera copia,mejor
    //Ciclo de periodos mayores
    para i←0 hasta car.cuantos en 1 hacer
        si(car.periodos[i].carga>promedio) entonces
            band_per←falso
            j←0
            //Ciclo de periodos menores
            mientras(((j≠i)^(j<car.cuantos))^(¬band_per)) hacer
                band_mat←falso
                k←0
                //Ciclo de materias independientes para mover
                mientras((k<car.periodos[i].cantidad)^(¬band_mat)) hacer
                    si((((cur.materias[car.periodos[i].materias[k]].cant_prela=0) ^
                        (¬cur.materias[car.periodos[i].materias[k]].poslaciones)) ^
                        (car.periodos[j].carga+cur.materias[car.periodos[i].materias[k]]
                            .creditos≤promedio)) ^
                        (¬cur.materias[car.periodos[i].materias[k]].usada)) entonces
                        si(((car.periodos[j].cantidad+1≤carga_max)^
                            (car.periodos[i].cantidad-1>carga_min))^
                            (verif_negado(j,car,cur,car.periodos[i].materias[k])))
                                entonces
                                    car.periodos[j].materias[car.periodos[j].cantidad] ←
                                        car.periodos[i].materias[k]
                                    car.periodos[j].cantidad← car.periodos[j].cantidad+1
                                    car.periodos[j].carga←
                                        car.periodos[j].carga +
                                        cur.materias[car.periodos[i].materias[k]].creditos
                                    car.periodos[i].materias[k] ←
                                      car.periodos[i].materias[car.periodos[i].cantidad-1]
                                    car.periodos[i].cantidad←car.periodos[i].cantidad - 1
                                    car.periodos[i].carga ← car.periodos[i].carga -
                                    cur.materias[car.periodos[j].materias[car.periodos[j]
                                        .cantidad-1]].creditos
                            fsi
                        fsi
                        si(car.periodos[i].carga<promedio) entonces
                                band_mat←verdadero
                        fsi
                        k←k+1
                fmientras
                si(car.periodos[i].carga<promedio) entonces
                    band_per←verdadero
                fsi
                j←j+1
            fmientras
        fsi
    fpara
    copia←car
    para i←car.cuantos hasta 0 en -1 hacer
        si(car.periodos[i].carga>promedio) entonces
            band_per←falso
            j←car.cuantos-1
```

```
            //Ciclo de periodos menores
            mientras(((j≠i)^(j>0))^(~band_per)) hacer
                band_mat←falso
                k←0
                //Ciclo de materias independientes para mover
                mientras((k<car.periodos[i].cantidad)^(~band_mat)) hacer
                    si((((cur.materias[car.periodos[i].materias[k]].cant_prela=0) ^
                        (~cur.materias[car.periodos[i].materias[k]].poslaciones)) ^
                        (car.periodos[j].carga+cur.materias[car.periodos[i].materias[k]]
                            .creditos≤promedio)) ^
                        (~cur.materias[car.periodos[i].materias[k]].usada)) entonces
                        si(((car.periodos[j].cantidad+1≤carga_max)^
                            (car.periodos[i].cantidad-1>carga_min))^
                            (verif_negado(j,car,cur,car.periodos[i].materias[k])))
                                entonces
                                   car.periodos[j].materias[car.periodos[j].cantidad] ←
                                       car.periodos[i].materias[k]
                                   car.periodos[j].cantidad← car.periodos[j].cantidad+1
                                   car.periodos[j].carga←car.periodos[j].carga +
                                       cur.materias[car.periodos[i].materias[k]].creditos
                                   car.periodos[i].materias[k] ←
                                     car.periodos[i].materias[car.periodos[i].cantidad-1]
                                   car.periodos[i].cantidad←car.periodos[i].cantidad - 1
                                   car.periodos[i].carga← car.periodos[i].carga -
                                    cur.materias[car.periodos[j].materias[car.periodos[j]
                                       .cantidad-1]].creditos
                        fsi
                    fsi
                    si(car.periodos[i].carga<promedio) entonces
                           band_mat←verdadero
                    fsi
                    k←k+1
                fmientras
                si(car.periodos[i].carga<promedio) entonces
                    band_per←verdadero
                fsi
                j←j-1
            fmientras
        fsi
    fpara
    si(evaluador(copia,cur,promedio)<evaluador(car,cur,promedio)) entonces
        car←copia
    fsi
fproc
```

La heurística, básicamente, persigue el objetivo de colocar las materias independientes en los mejores periodos de acuerdo al balance, para hacer esto, compara un periodo base, con los periodos siguientes y evalúa mover las materias independientes entre períodos. Este proceso lo realiza 2 veces, en una primera instancia evaluando los periodos del 1 al *N*, y luego en sentido contrario. Finalmente los compara a través del evaluador y toma al mejor.

Casos de Prueba

Los experimentos se han realizado con la finalidad de determinar el tiempo de respuesta obtenido de la técnica metaheurística propuesta, en contraste con los mejores resultados obtenidos por los antecedentes.

De igual forma, se estudiaron trece casos de prueba sintetizados tomados tanto de la librería CSPLib como de las instancias propuestas en Chiarandini et al. (2011), los cuales se identifican como bcap8, bcap10, bcap12, UD1, UD2, UD3, UD4, UD5, UD6, UD7, UD8, UD9 y UD10, adicionalmente, será estudiada una nueva instancia correspondiente a la Facultad Experimental de Ciencias y Tecnología (FACYT).

La lista de los casos de prueba a estudiar se encuentra en la tabla 4, donde se presenta la información sobre la cantidad de restricciones o prelaciones, asignaturas, preferencias, períodos por año y *curricula* a formar en cada instancia.

Tabla 4: Casos de prueba

Instancia	Materias	Años x Periodos	Curriculums	Prelaciones	Preferencias
bacp8	46	8 (4 x 2)	1	33	0
bacp10	42	10 (5 x 2)	1	33	0
bacp12	66	12 (6 x 2)	1	65	0
UD1	307	9 (3 x 3)	37	1383	90
UD2	268	6 (2 x 3)	20	174	79
UD3	236	9 (3 x 3)	31	1092	66
UD4	139	6 (2 x 3)	16	188	40
UD5	282	6 (3 x 2)	31	397	54
UD6	264	4 (2 x 2)	20	70	55
UD7	302	9 (3 x 3)	37	1550	83
UD8	208	6 (2 x 3)	19	149	60
UD9	303	9 (3 x 3)	37	1541	85
UD10	188	6 (2 x 3)	15	214	55
FACYT	136	8 (4 x 2)	5	143	15

Fuente: Elaboración propia.

Tiempo de procesamiento

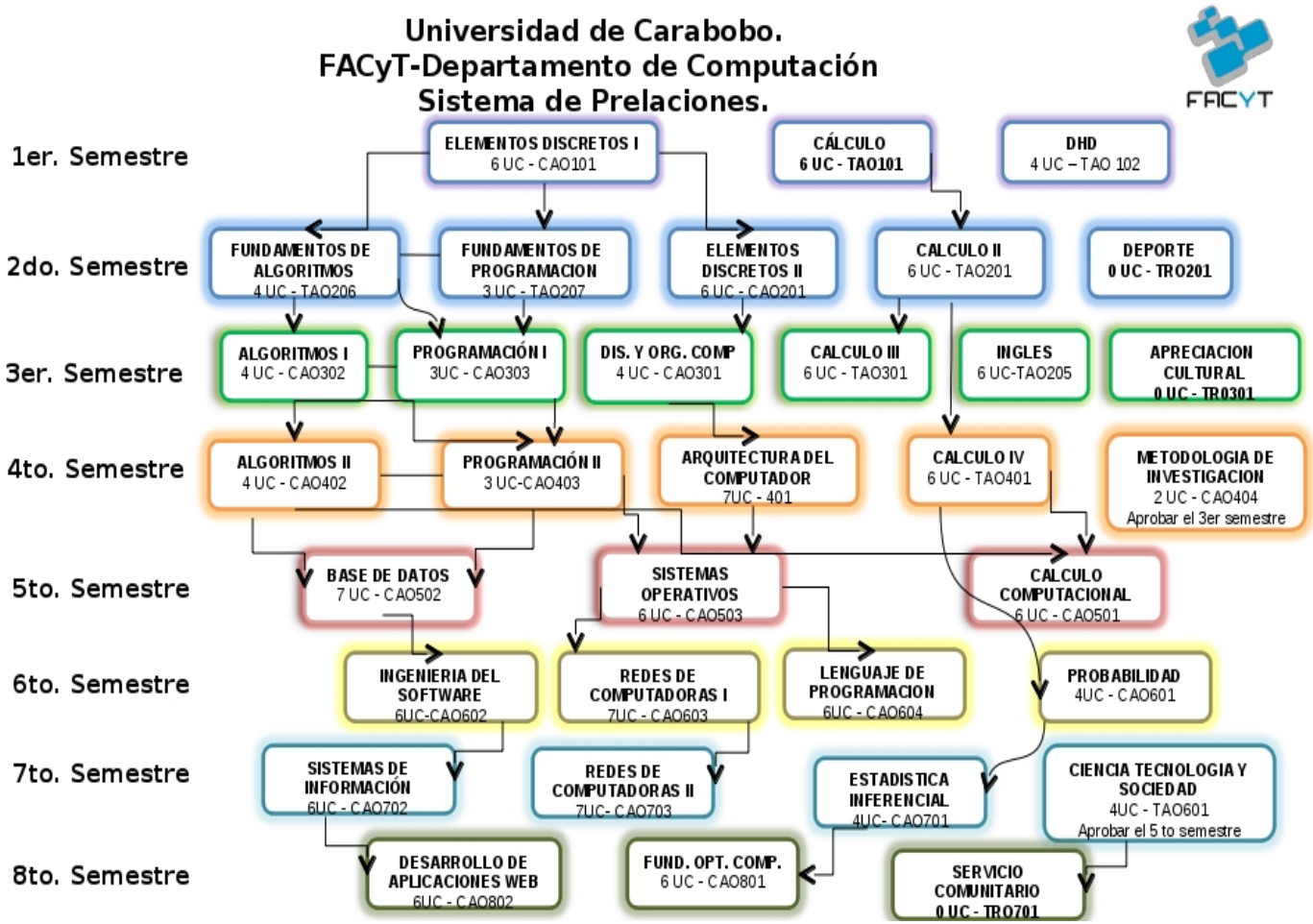
Cuando se hace investigación acerca de procedimientos metaheurísticos, es importante destacar, tanto los tiempos de procesamiento, como la calidad de los resultados obtenidos. Es por ello que esta sección es una de las más importantes del presente trabajo de investigación.

A continuación se presentan un conjunto de tablas que corresponden a los tiempos de procesamiento y calidad de la solución planteada. Para medir la calidad de la solución, se calculó el promedio de unidades de crédito que debía haber por períodos,

sumando todas las unidades de crédito y dividiéndolas entre el número de períodos, luego, se aproximó de manera superior a un número entero.

Posteriormente, se utiliza una fórmula evalúa utilizando como parámetro en la función exponencial, el valor absoluto de la diferencia entre las unidades de crédito que tiene el período evaluado y el promedio calculado, de esta manera, las soluciones que tengan mayor diferencia con respecto al promedio en un período, obtienen peores resultados.

Figura 2: Sistema de prelaciones de FaCyT - Computación



Fuente: Facultad de Ciencias y Tecnología - Universidad de Carabobo

En una primera instancia, se evaluó el caso de FaCyT - Computación. Para desarrollar mejor la idea del caso, se adjunta la figura 2. Para este caso específico, los semestres actualmente disponen de la siguiente distribución de carga:

- Período 1: 16 Unidades de crédito
- Período 2: 19 Unidades de crédito
- Período 3: 23 Unidades de crédito
- Período 4: 22 Unidades de crédito
- Período 5: 19 Unidades de crédito
- Período 6: 23 Unidades de crédito
- Período 7: 21 Unidades de crédito
- Período 8: 12 Unidades de crédito

Lo cual resulta en 155 unidades de crédito a distribuir entre ocho períodos, lo que en promedio serían 19,375 unidades de crédito, aproximándolo al número entero superior, serían 20 unidades de crédito por período. Una vez ejecutado el algoritmo, para este caso el tiempo de respuesta fue inferior a 1 segundo y el resultado fue el siguiente:

- Período 1: 18 Unidades de crédito
- Período 2: 19 Unidades de crédito
- Período 3: 17 Unidades de crédito
- Período 4: 20 Unidades de crédito
- Período 5: 23 Unidades de crédito
- Período 6: 21 Unidades de crédito
- Período 7: 19 Unidades de crédito
- Período 8: 18 Unidades de crédito

Cumpliendo con todas las restricciones de prelación y períodos no deseados para las materias que señalan en la figura 2, quedando distribuida la carrera de la siguiente manera:

- Período 1: [TAO101 CAO101 TAO205]
- Período 2: [TAO201 TAO206 TAO207 CAO201]
- Período 3: [TAO301 CAO302 CAO303 CAO301]
- Período 4: [TAO401 CAO402 CAO403 CAO401]
- Período 5: [CAO503 CAO601 CAO501 CAO502]
- Período 6: [CAO603 CAO701 CAO602 TAO601]
- Período 7: [CAO703 CAO801 CAO702]
- Período 8:[CAO802 TRO301 TAO102 CAO604 CAO404 TRO201 TRO701]

De las siete materias que muestra el período 8, tres de ellas, son requisitos obligatorios, sin carga. En lo que respecta al procesamiento de niveles, el algoritmo arroja la respuesta de los niveles que logró armar de prelaciones directas, teniendo los siguientes:

- [TAO101 TAO201 TAO301 TAO401 CAO501]
- [CAO101 TAO206 CAO302 CAO402 CAO502 CAO602 CAO702 CAO802]
- [TAO207 CAO303 CAO403 CAO503 CAO603 CAO703]
- [CAO201 CAO301 CAO401]
- [CAO601 CAO701 CAO801]

Estos datos pueden certificarse con la figura 2, donde se evidencian las prelaciones directas entre estas materias, es por ello que el algoritmo genera los niveles

para reducir la combinatoria. Estas prelaciones directas, no vienen explícitas en el archivo de entrada, son construidas en el procesamiento de los datos.

A continuación se mostrarán dos tablas (5 y 6) de resultados obtenidos para los casos de prueba, se mostrarán estudios anteriores y los resultados del presente estudio, la unidad de medida de tiempo es en segundos. Cada cuadro corresponde al número de iteraciones, en un caso son 40 iteraciones y en el otro 400 iteraciones, luego de múltiples pruebas, se evidenció que no existía mejoría significativa en la calidad de la solución, por encima de 400 iteraciones, lo cual se muestra entre las figuras. Para cada columna se tomó el mejor tiempo peor tiempo y promedio.

Tabla 5: Algoritmo con 40 iteraciones

	Metaheurística 40 iteraciones		
Instancia	**Min (Seg)**	**Max. (Seg)**	**Promedio (Seg)**
bacp8	0,3982	0,7512	0,5747
bacp10	0,3252	0,7854	0,5553
bacp12	0,7058	1,4568	1,0813
UD1	6,5968	8,3654	7,4811
UD2	1,6541	2,5847	2,1194
UD3	4,4587	6,5881	5,5234
UD4	1,3218	2,9875	2,15465
UD5	2,6548	2,8745	2,76465
UD6	1,4294	2,5478	1,9886
UD7	7,5731	10,8957	9,2344
UD8	0,6096	1,2165	0,91305
UD9	6,2298	8,2146	7,2222
UD10	1,4294	3,5781	2,50375
FACYT	0,5856	1,2879	0,93675

Fuente: Elaboración propia.

Tabla 6: Algoritmo con 400 iteraciones

	Metaheurística 400 iteraciones		
Instancia	**Min (Seg)**	**Max. (Seg)**	**Promedio (Seg)**
bacp8	1,6999	2,8578	2,27885
bacp10	1,0204	2,0215	1,52095
bacp12	1,1251	1,9835	1,5543
UD1	19,2962	22,5745	20,93535
UD2	5,9266	8,215	7,0708
UD3	12,1906	14,8661	13,52835
UD4	3,6238	5,0157	4,31975
UD5	8,6958	10,1253	9,41055
UD6	3,5123	4,2544	3,88335
UD7	18,8596	21,8573	20,35845
UD8	3,5915	4,1058	3,84865
UD9	18,1506	20,9126	19,5316
UD10	4,9463	6,3547	5,6505
FACYT	0,9435	1,9054	1,42445

Fuente: Elaboración propia.

El tiempo de respuesta del algoritmo, es medido a través de estructuras de tiempo basadas en librerías estándar de lenguaje C, se comienza a contar el tiempo antes de abrir el archivo de entrada, es decir, sin que haya habido procesamiento previo. La medición de tiempo concluye cuando se cierra el archivo de salida, luego de haber procesado todos los *curricula* previstos en el archivo de entrada y se haya liberado la memoria.

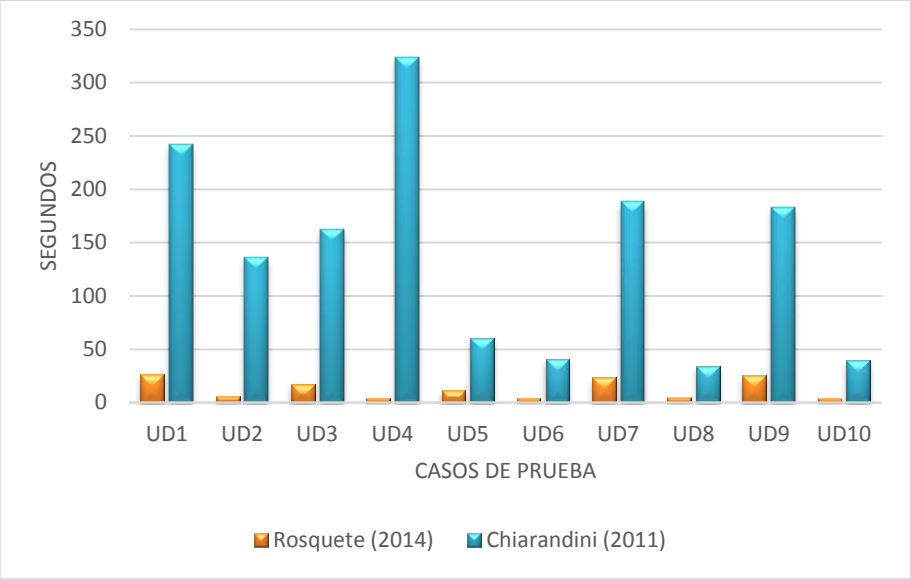
En la tabla 7 y figura 3 se presentan los resultados comparativos entre el método existente y el implementado en este trabajo de investigación.

Tabla 7: Comparación entre métodos con 400 iteraciones el propuesto, utilizando como medida, el tiempo.

	Rosquete (2013)	Chiarandini (2011)	Mejora (Seg)
bacp8	2,2789	0	-
bacp10	1,5210	0	-
bacp12	1,5543	0	-
UD1	20,9354	242	221
UD2	7,0708	136,2	129
UD3	13,5284	162,2	149
UD4	4,3198	324,2	320
UD5	9,4106	60,6	51
UD6	3,8834	40,8	37
UD7	20,3585	188,6	168
UD8	3,8487	34	30
UD9	19,5316	182,8	163
UD10	5,6505	39,4	34
FACYT	1,4245	N/A	N/A

Fuente: Elaboración propia.

Figura 3: Gráfica de resultados comparativos UD1 a UD10

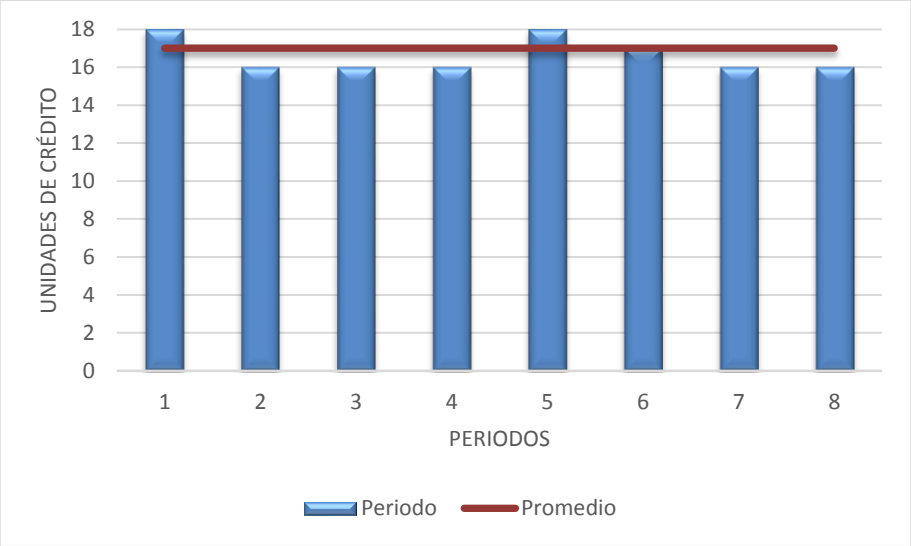


Fuente: Elaboración propia.

A través de la figura 3, donde se coloca cada caso en función de la cantidad de segundos que tomó el procesamiento. Se evidencia gráficamente por los resultados de las tablas que la solución propuesta tiene un mejor rendimiento en la medida de tiempo de procesamiento. Las siguientes figuras, son demostrativas de la calidad de las soluciones, utilizando una línea horizontal para mostrar el promedio y barras verticales para identificar la carga del período para cada *curriculum*.

En cada una de las figuras de la 4 a la 15, se colocarán subcasos representativos, que forman parte de los archivos completos de casos de prueba, estos subcasos fueron elegidos de manera aleatoria como una muestra de la calidad de la solución algorítmica planteada, excepto el caso FaCyT por ser la población real, donde se pondrán todos los casos evaluados. El eje X representa cada período, mientras que el eje Y representa las unidades de crédito asignadas en ese semestre.

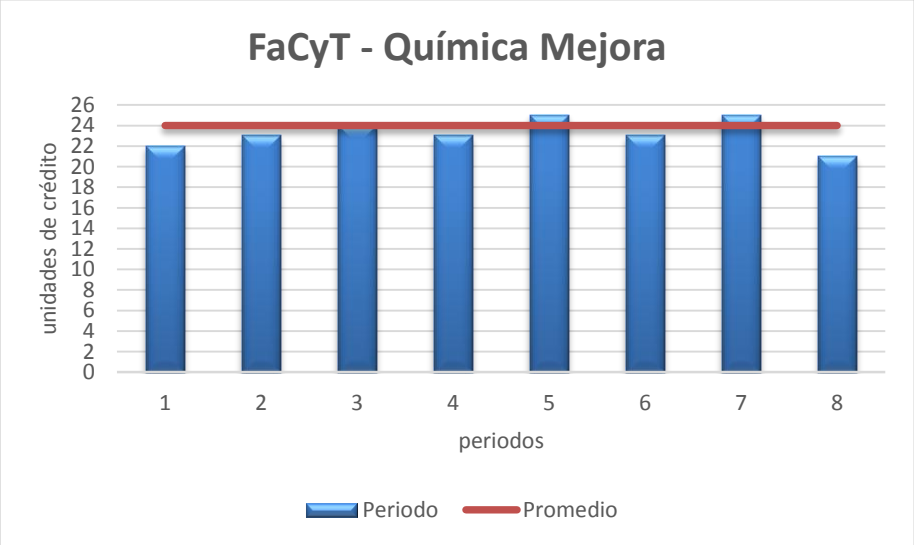
Figura 4: Caso Bacp8



Fuente: Elaboración propia.

Para la figura 4, se evidencia buen balance, dadas las características del caso.

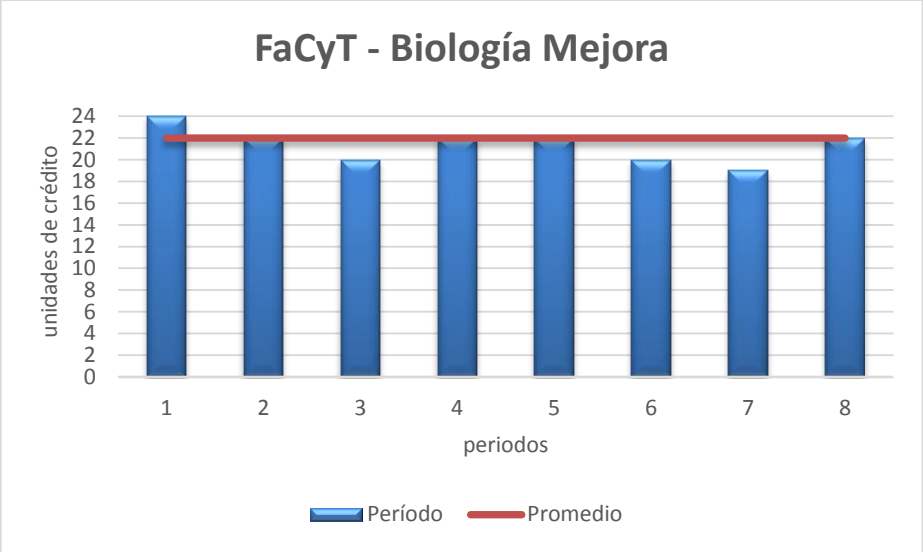
Figura 5: Caso FaCyT - Química



Fuente: Elaboración propia.

Para la figura 5, se evidencia un buen balance.

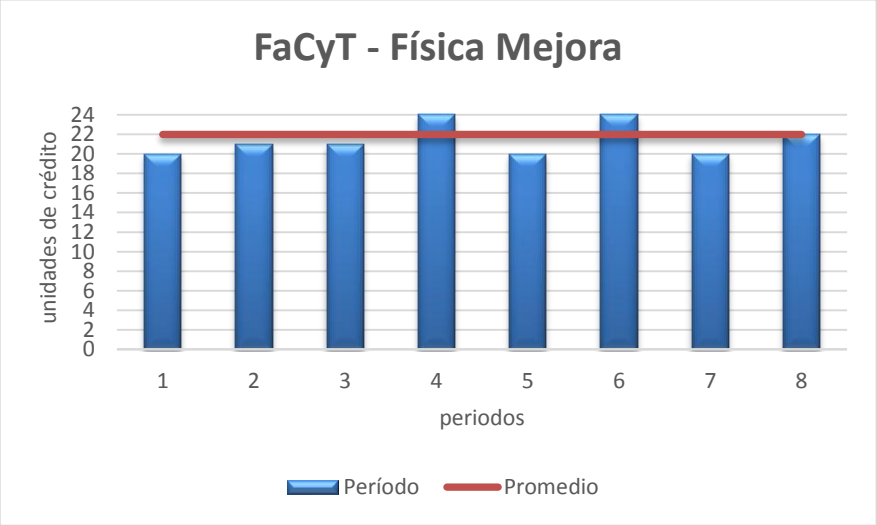
Figura 6: Caso FaCyT - Biología



Fuente: Elaboración propia.

Para la figura 6, se muestra un mejor balance de cargas al utilizado actualmente en la carrera. El promedio fue 22.

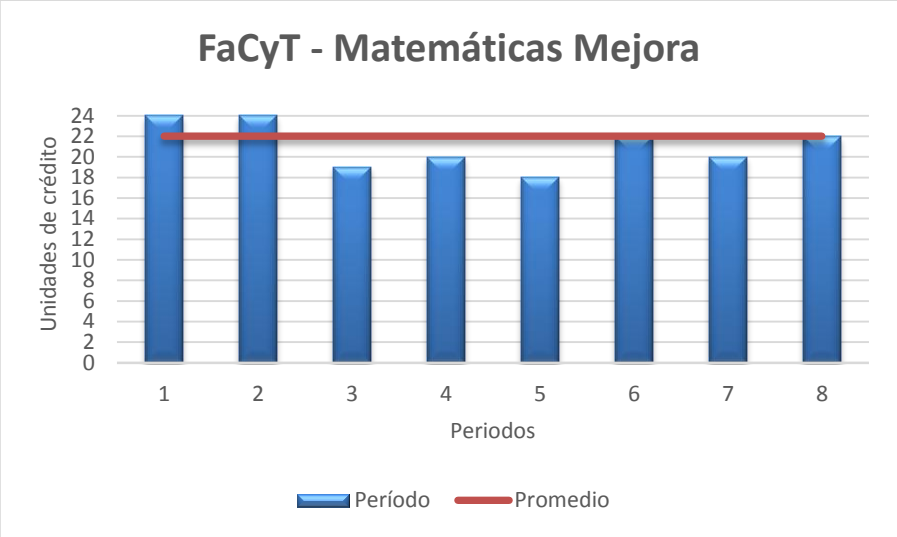
Figura 7: Caso FaCyT - Física



Fuente: Elaboración propia.

Para la figura 7 se muestra que el algoritmo obtuvo resultados significativos. El promedio fue 22

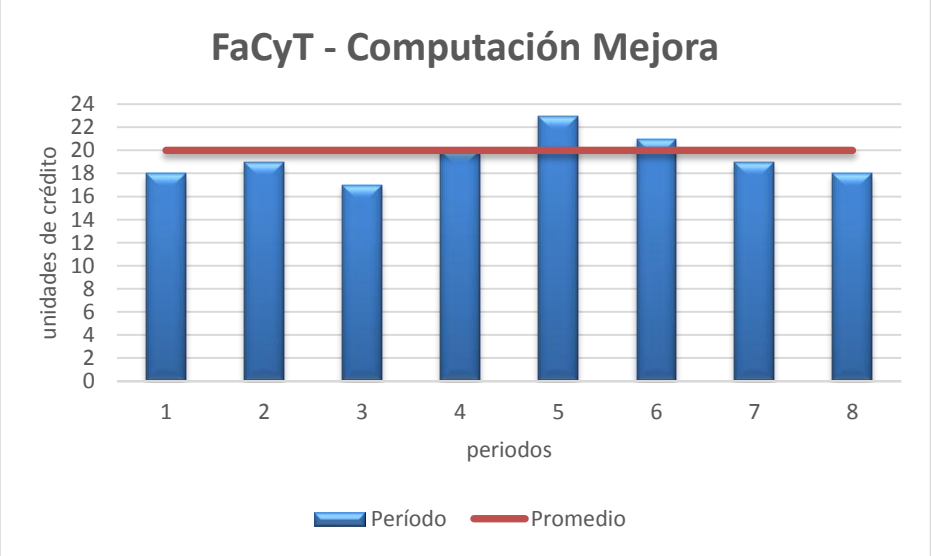
Figura 8: Caso FaCyT - Matemáticas



Fuente: Elaboración propia.

Para la figura 8 se evidencia un buen balance, y el promedio fue 22

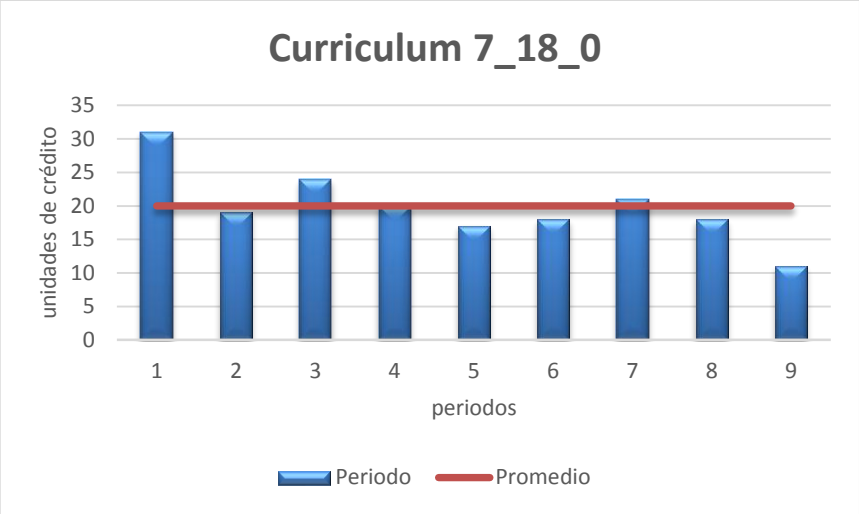
Figura 9: Caso FaCyT - Computación



Fuente: Elaboración propia.

En la figura 9 se evidencia un buen balance, con promedio de 20

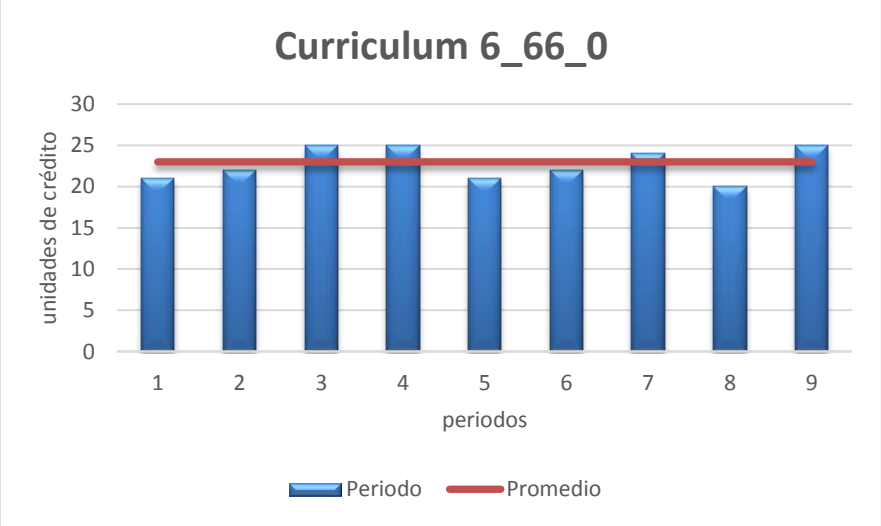
Figura 10: Caso UD1



Fuente: Elaboración propia.

Para la figura 10, se muestra un balance con mala tendencia, ya que en los períodos 2 y 7 se alcanzan 50% más de unidades de crédito de la esperada.

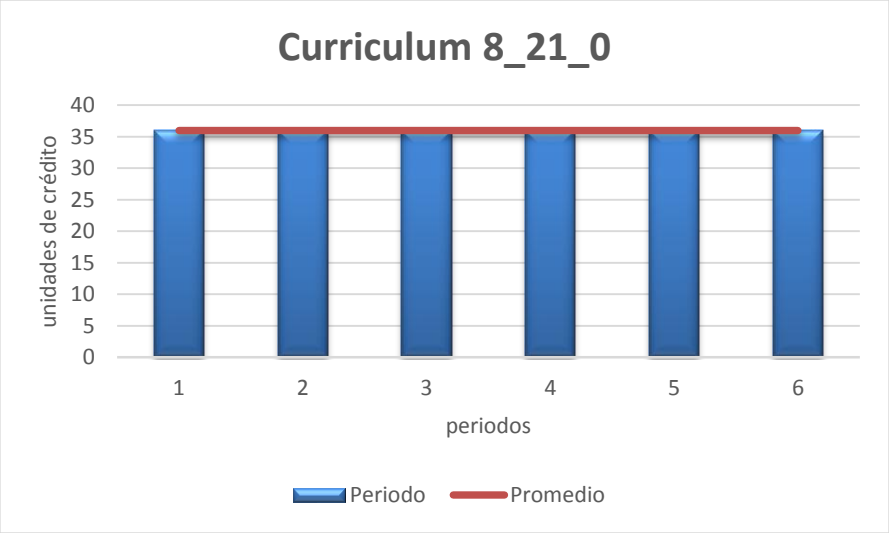
Figura 11: Caso UD1



Fuente: Elaboración propia.

Para la figura 11, el balance es bastante aceptable con respecto al promedio.

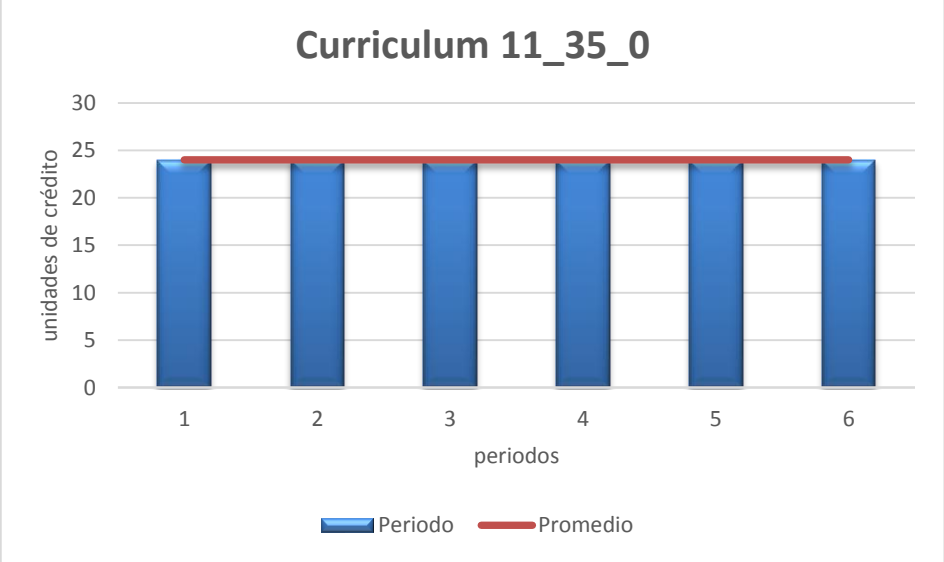
Figura 12: Caso UD4



Fuente: Elaboración propia.

Para la figura 12, se evidencia un balance perfecto.

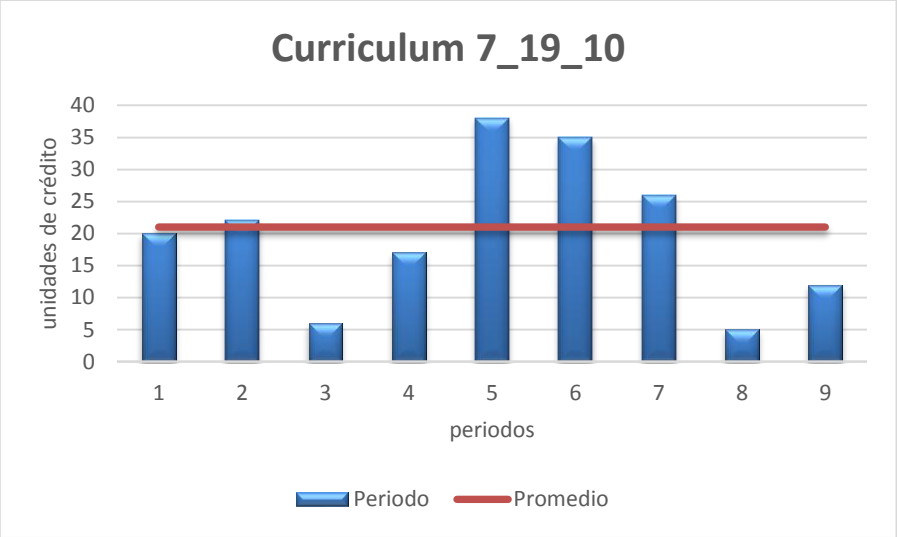
Figura 13: Caso UD4



Fuente: Elaboración propia.

Para la figura 13, se evidencia un balance perfecto.

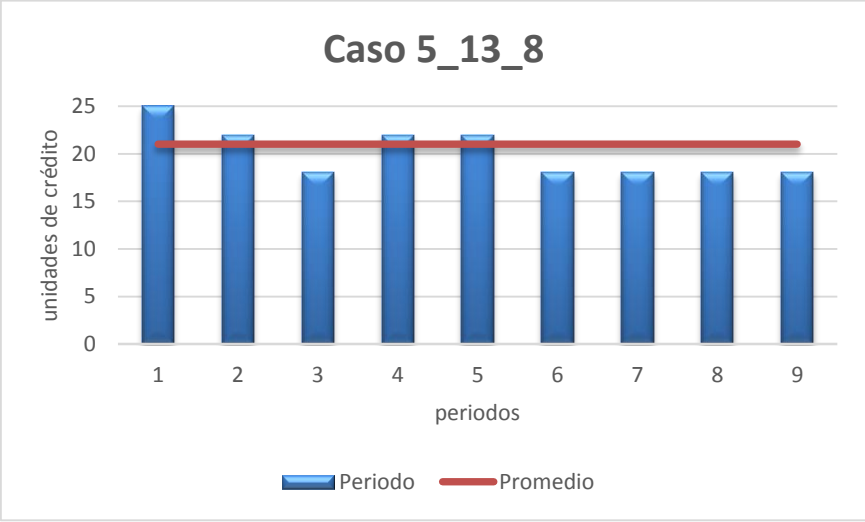
Figura 14: Caso UD7



Fuente: Elaboración propia.

Para la figura 14, se evidencia un desbalance con respecto al promedio.

Figura 15: Caso UD7

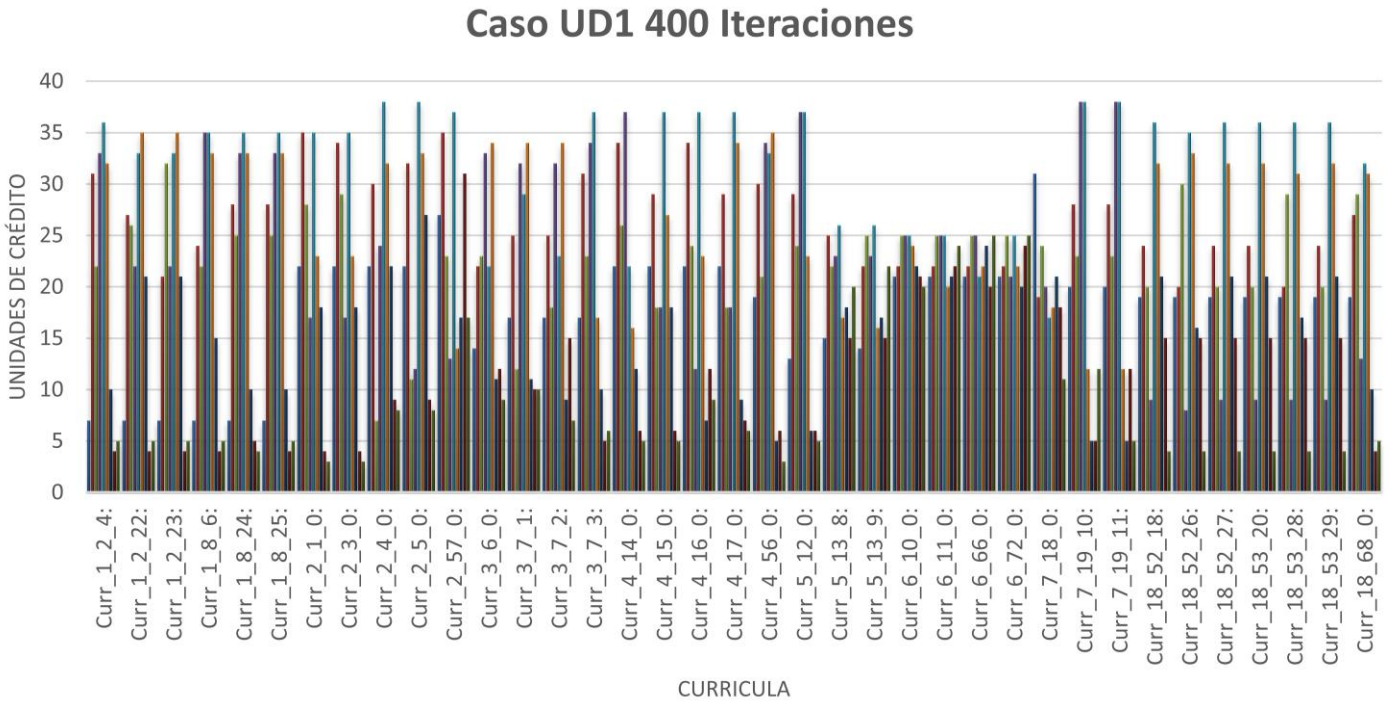


Fuente: Elaboración propia.

Para la figura 15 se evidencia un balance aceptable con respecto al promedio.

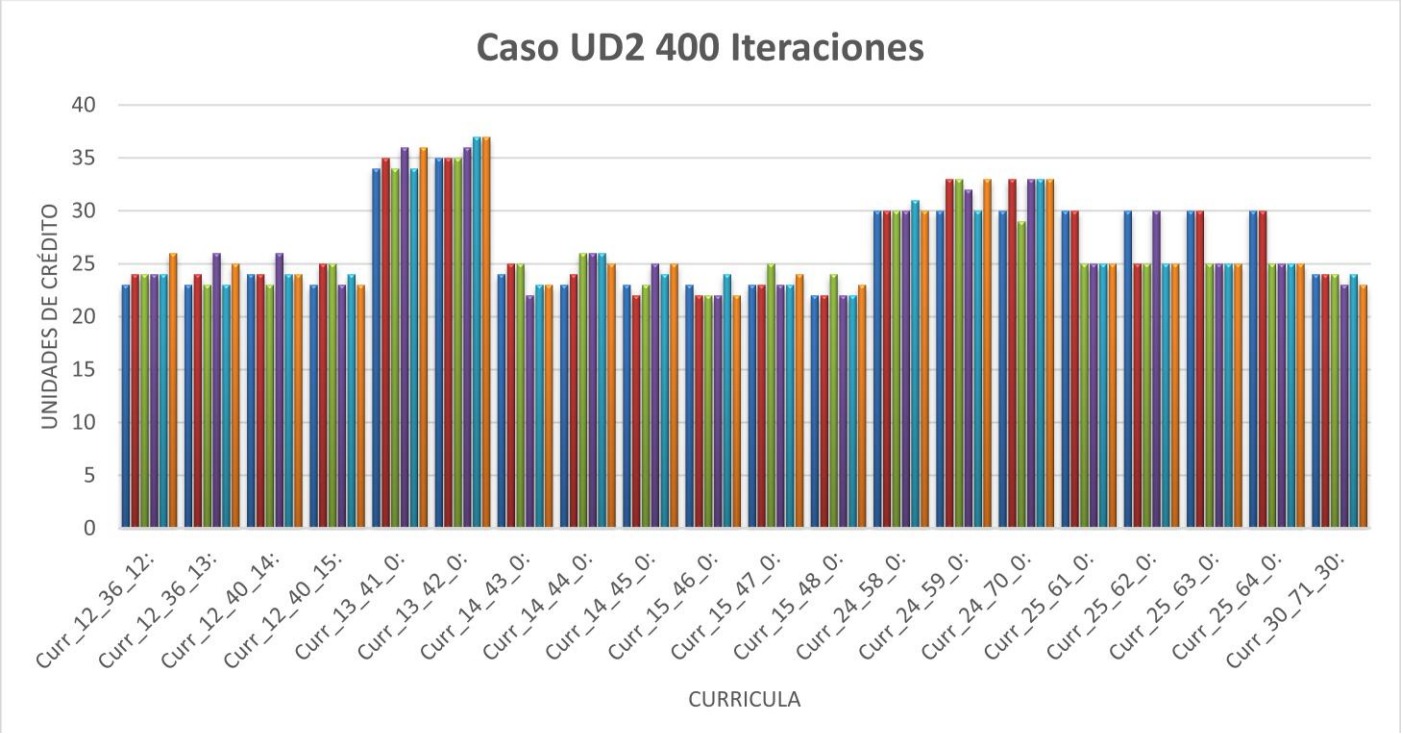
En las figuras de la 16 a la 28, el eje X corresponde a diversos bloques, donde cada bloque es un *curriculum* dentro del caso, y cada barra que se levanta dentro del bloque, representa un período, por lo que el eje Y representa las unidades de crédito. En dichas figuras, se representan todos los casos de entrada UD1 hasta UD10 por completo. En el caso UD7 se muestra unas figuras comparativas, de la solución con 40 iteraciones y con 400 iteraciones tardando esta última 21,85 segundos. Se puede notar claramente, que la mejora de la calidad de la solución es mínima. Sin embargo, en la gran mayoría de los *curricula*, la solución permanece igual, ya que son casos, que sólo tienen una solución factible.

Figura 16: Caso UD1 Completo, 400 iteraciones



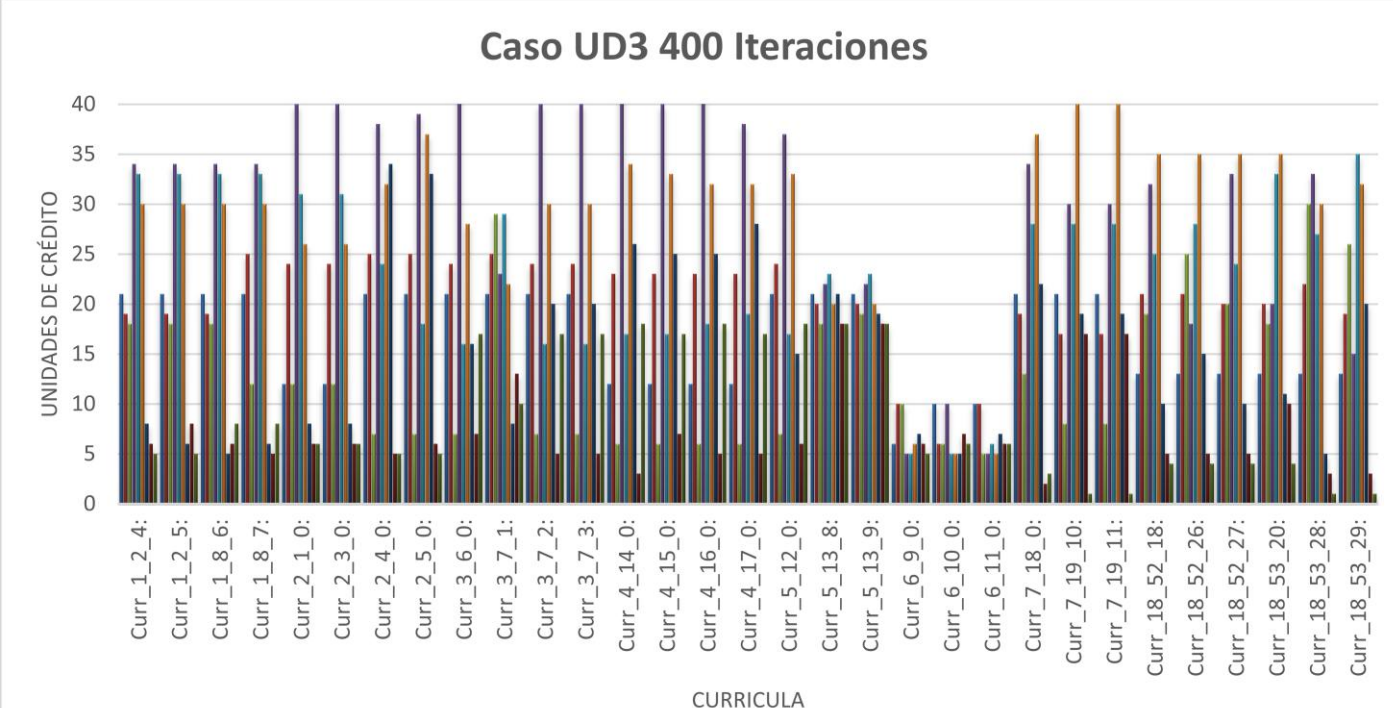
Fuente: Elaboración propia

Figura 17: Caso UD2 Completo, 400 iteraciones



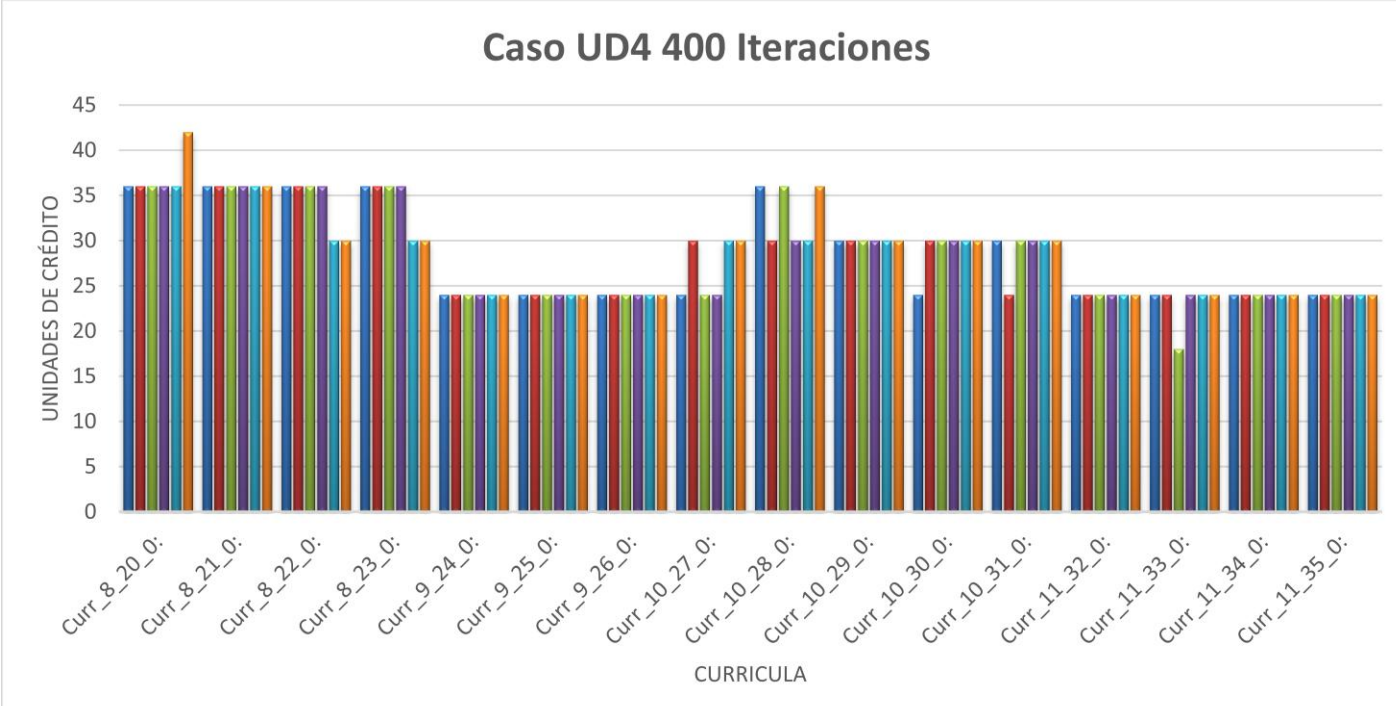
Fuente: Elaboración propia

Figura 18: Caso UD3 Completo, 400 iteraciones



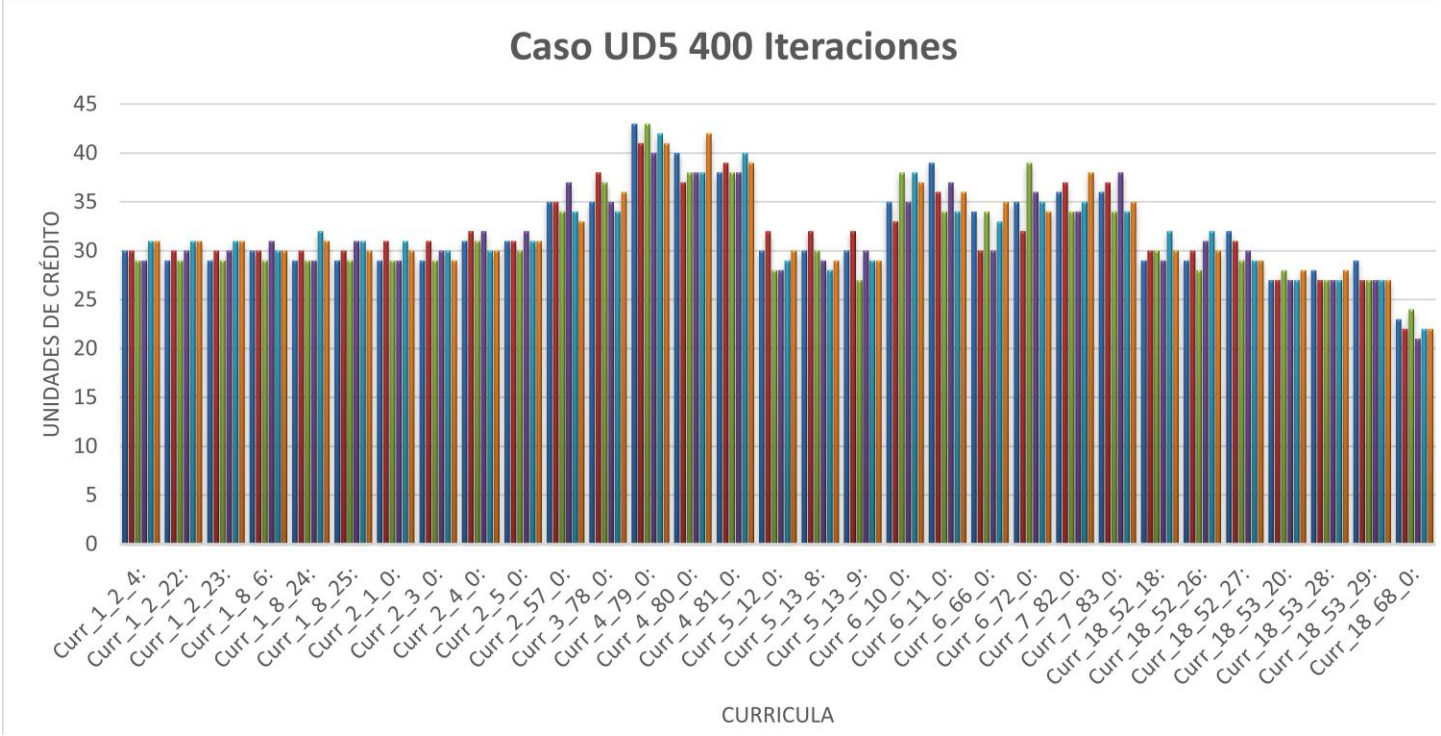
Fuente: Elaboración propia

Figura 19: Caso UD4 Completo, 400 iteraciones



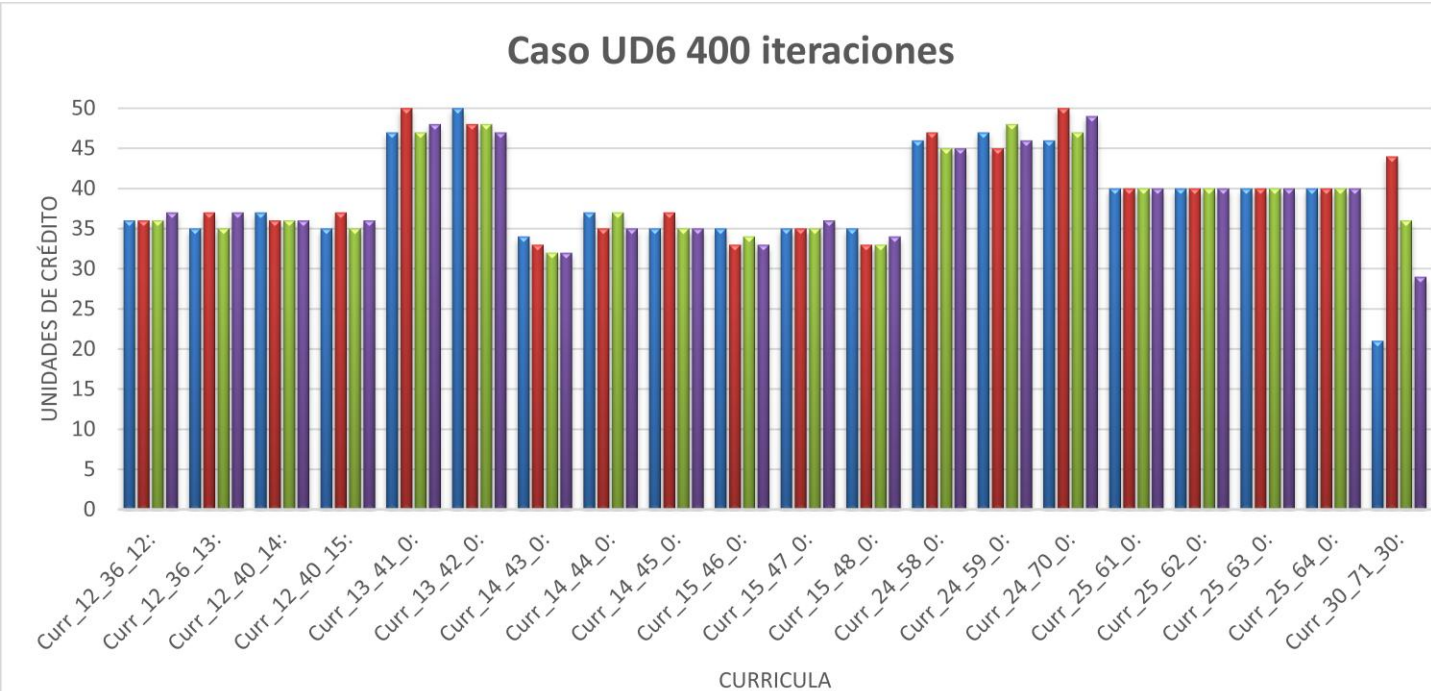
Fuente: Elaboración propia

Figura 20: Caso UD5 Completo, 400 iteraciones



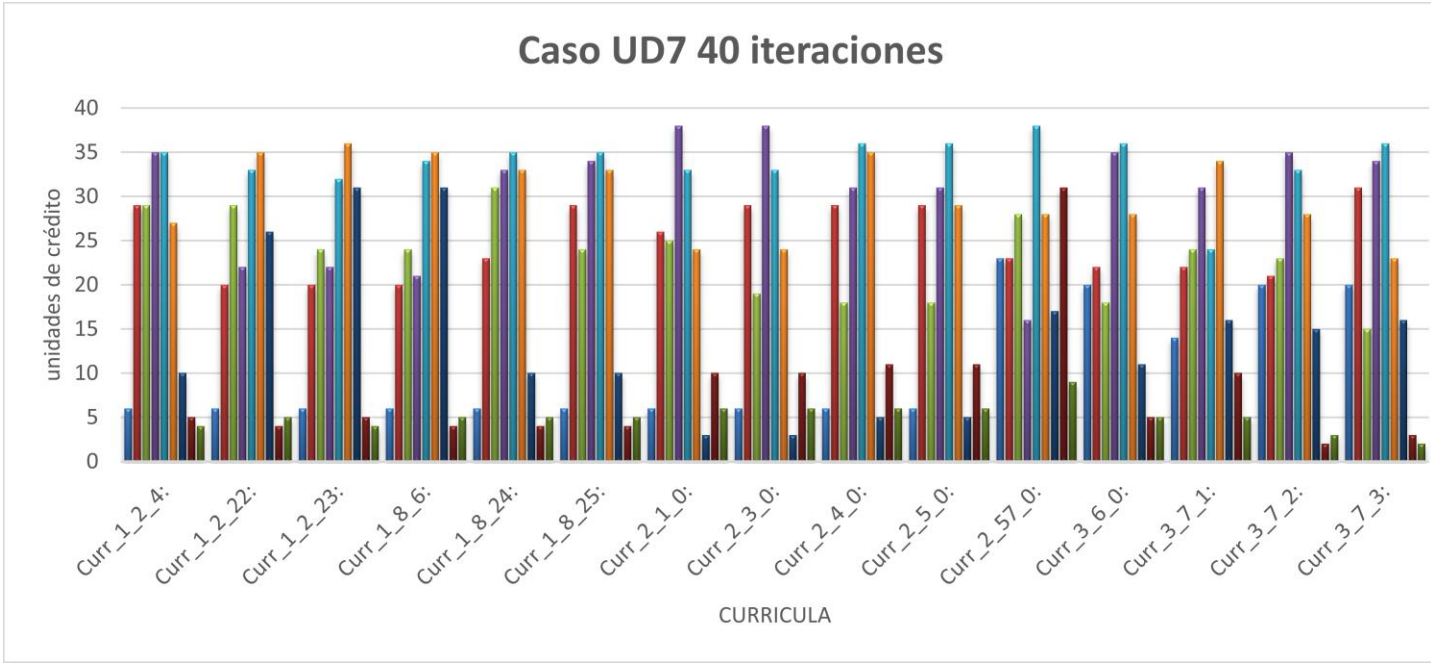
Fuente: Elaboración propia

Figura 21: Caso UD6 Completo, 400 iteraciones



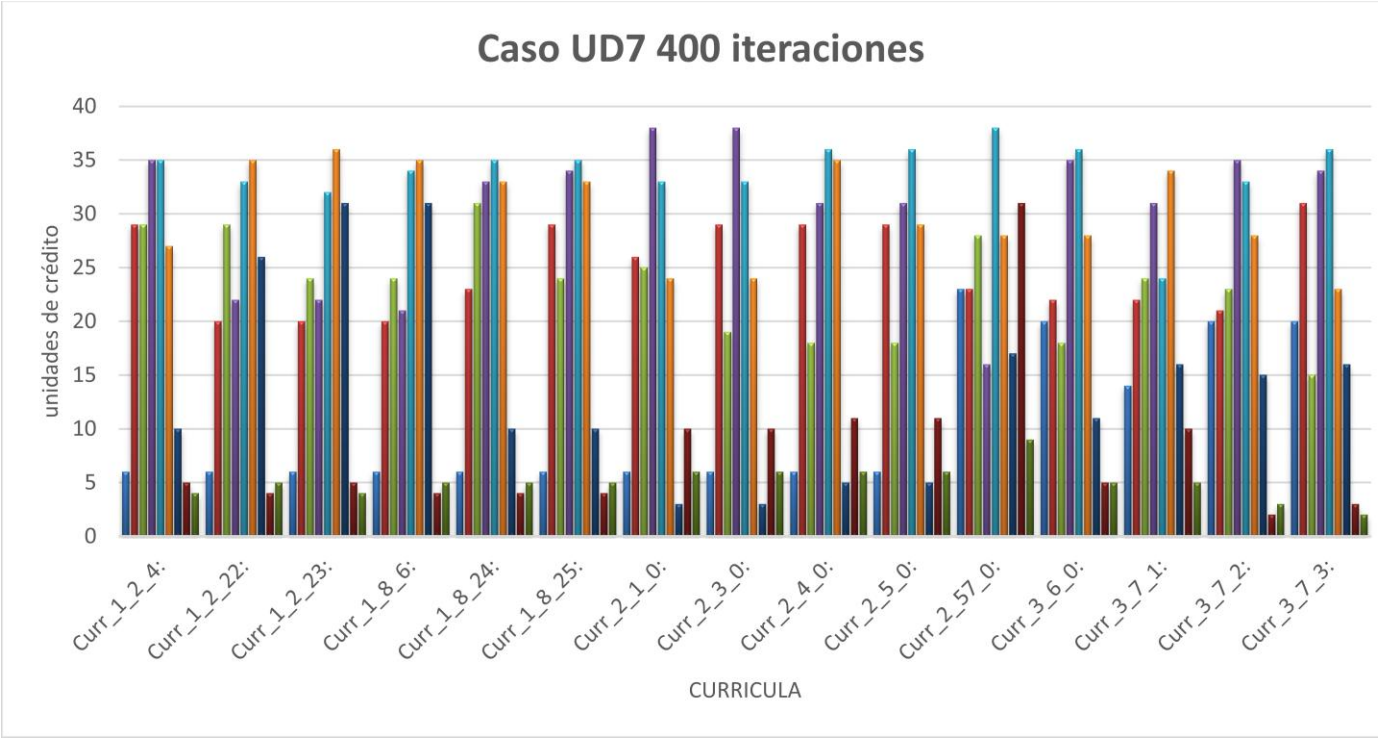
Fuente: Elaboración propia

Figura 22: Caso UD7 Parte I, 40 iteraciones



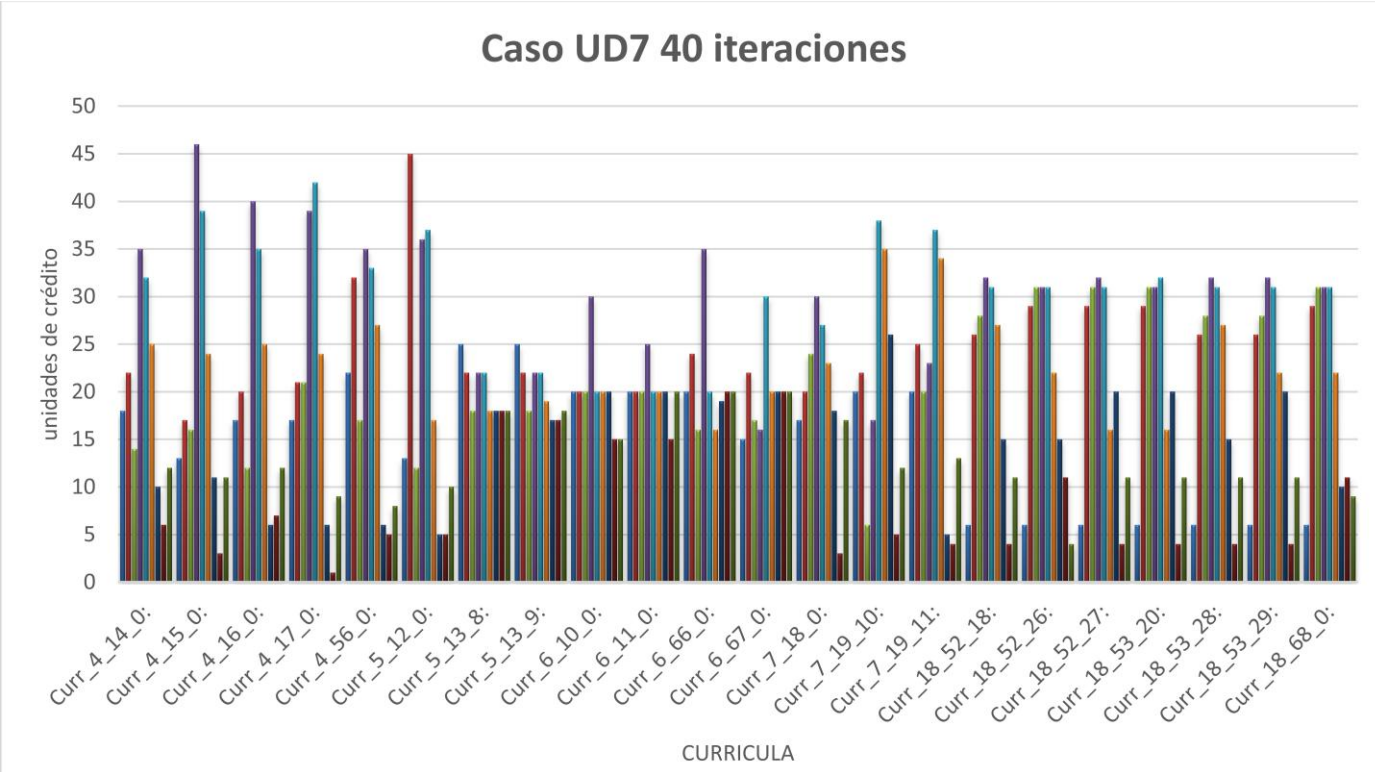
Fuente: Elaboración propia

Figura 23: Caso UD7 Parte I, 400 iteraciones



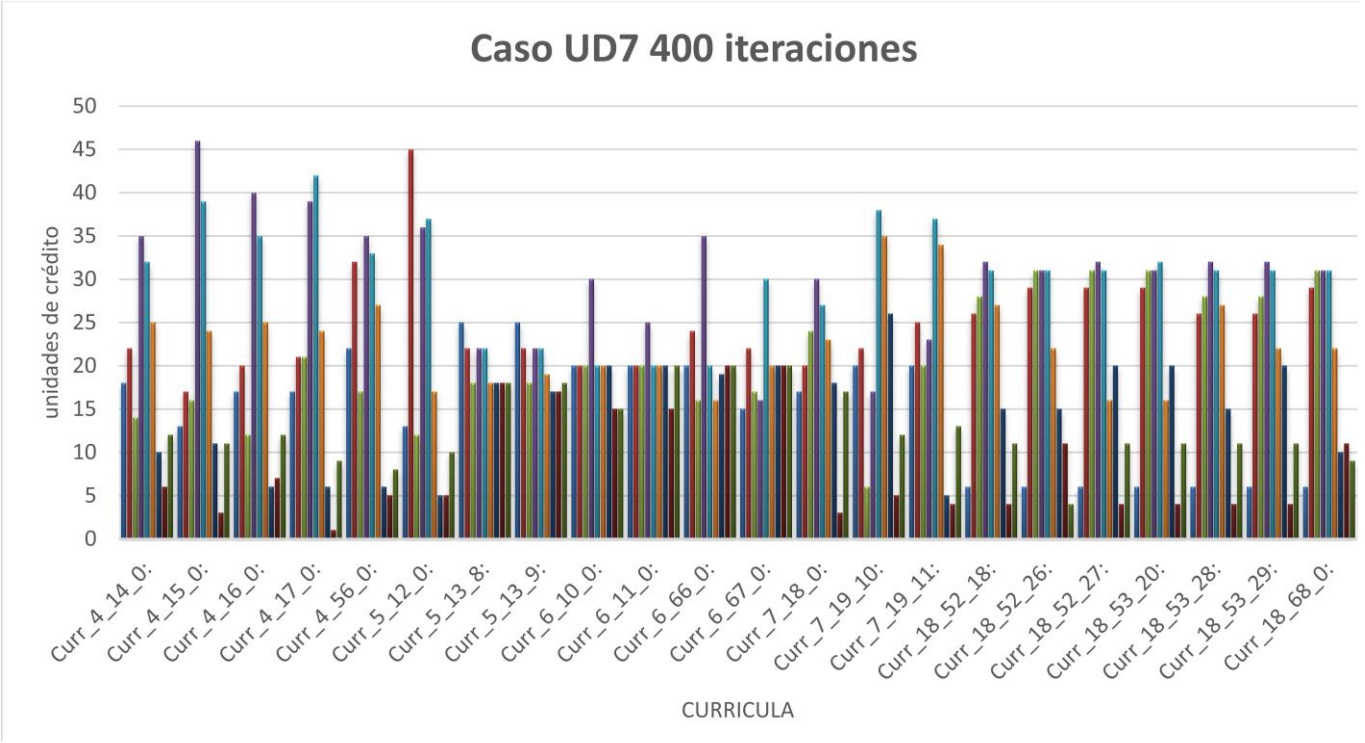
Fuente: Elaboración propia

Figura 24: Caso UD7 Parte II, 40 iteraciones



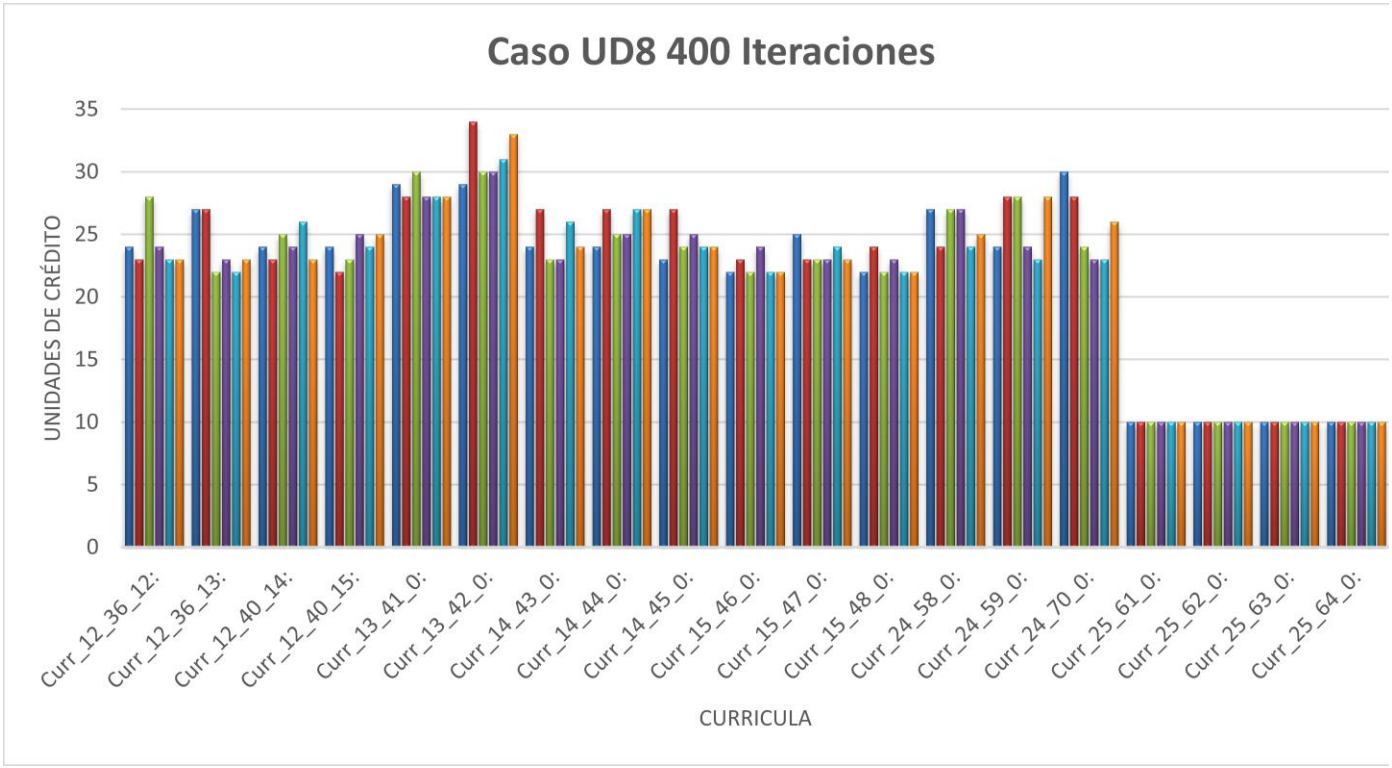
Fuente: Elaboración propia

Figura 25*: Caso UD7 Parte II, 400 iteraciones*



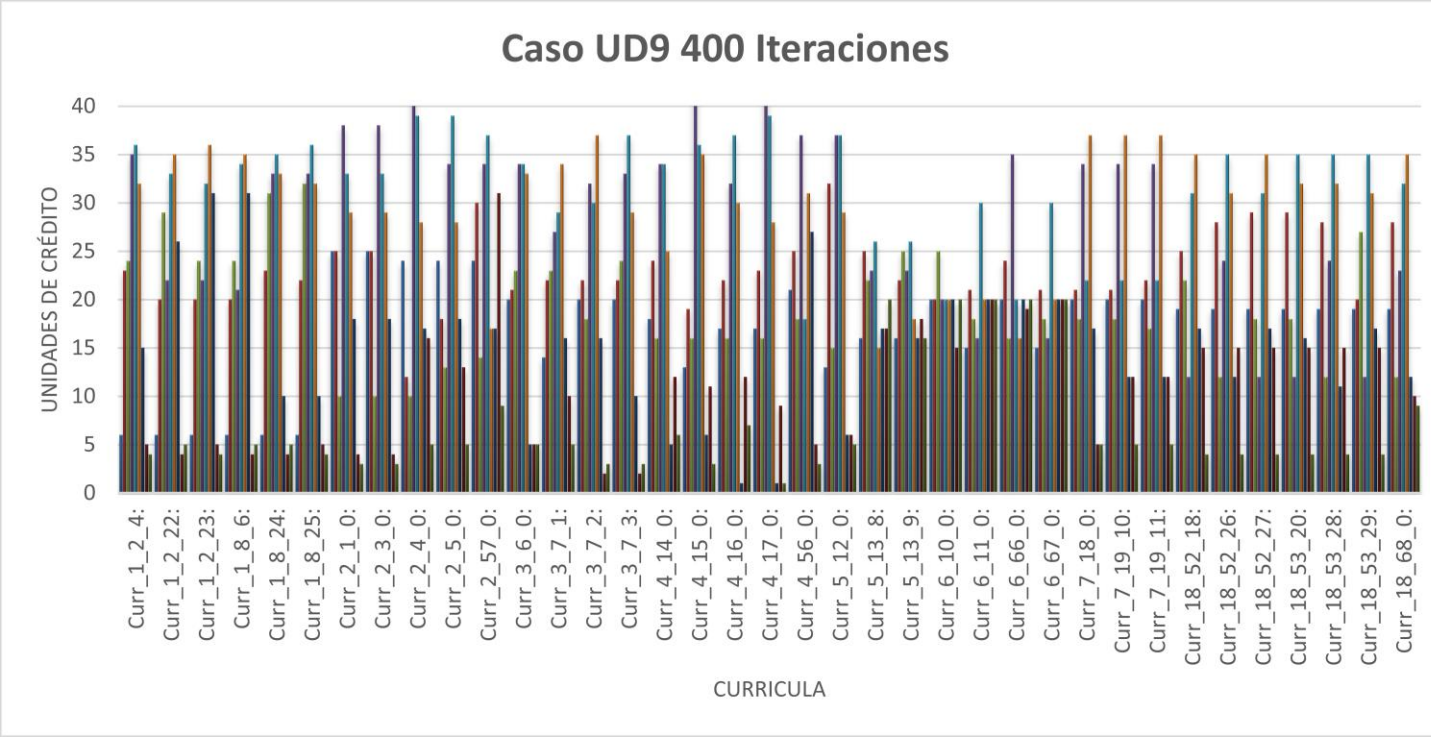
Fuente: Elaboración propia

Figura 26*: Caso UD8 Completo, 400 iteraciones*



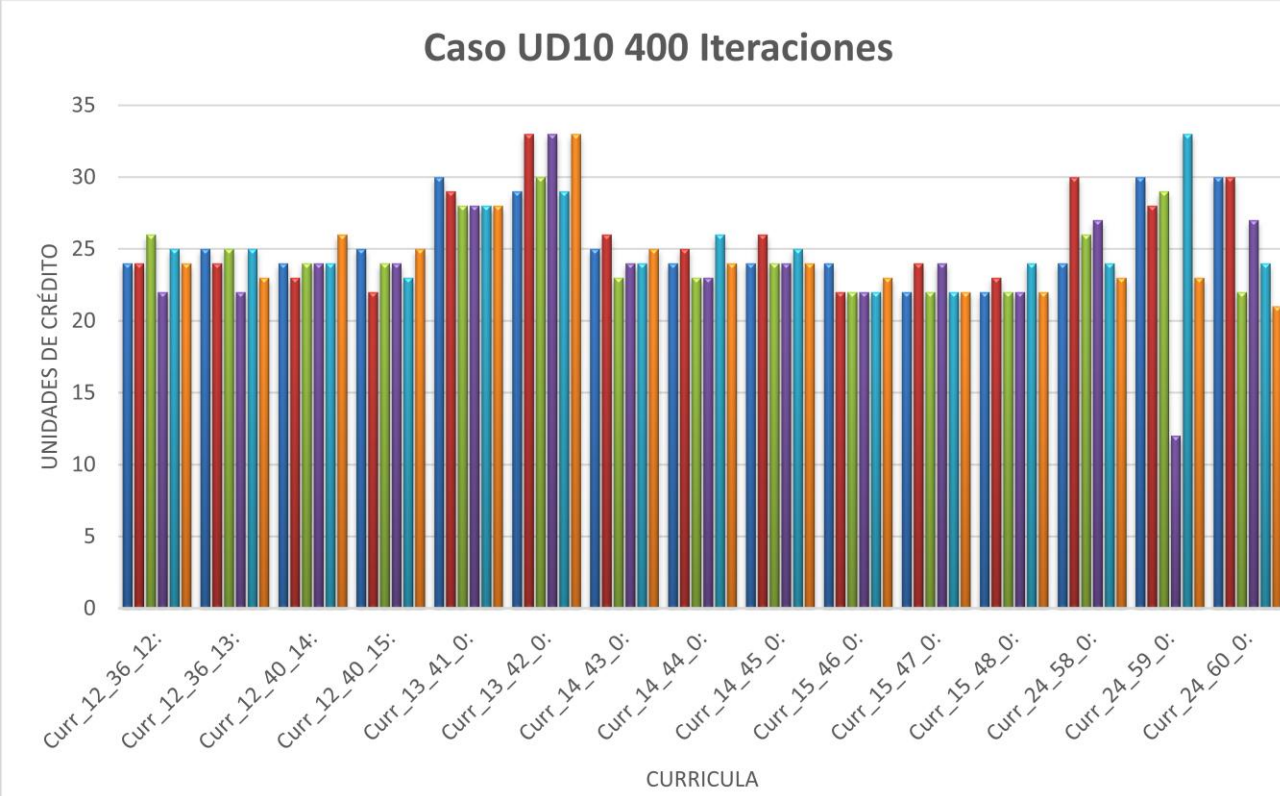
Fuente: Elaboración propia

Figura 27*: Caso UD9 Completo, 400 iteraciones*



Fuente: Elaboración propia

Figura 28: Caso UD10 Completo, 400 iteraciones



Fuente: Elaboración propia

Específicamente, en las figuras 16, 22, 23, 24, 25 y 27 se muestra un desbalance muy pronunciado. Esto se debe principalmente, a que existen múltiples asignaturas desde 1 unidad de crédito hasta 12 unidades de crédito. De igual manera, hay asignaturas como la c837 y c838 que tienen 18 y 31 unidades de crédito respectivamente, lo que las ubica por encima del promedio por sí solas, en la mayoría de los subcasos de UD1,UD7 y UD9.

De igual manera, en las figuras 17, 18, 19, 20, 21, 26 y 28, se muestra un balance muy aceptable en todos los subcasos, es de notar, que todas las barras de cada bloque, alcanzan aproximadamente la misma cantidad de unidades de créditos. En estos casos UD2, UD3, UD4, UD5, UD6, UD8 y UD10, no se presentan materias que tengan unidades de crédito superiores o muy cercanas al promedio.

Queda demostrada entonces la hipótesis de este trabajo de investigación, la técnica metaheurística junto al algoritmo de procesamiento previo, han sido capaces de solucionar el problema de balanceo de una carga curricular, en general se obtuvo un buen balance. Sin embargo, en algunos casos, como UD1 y UD3, los cuáles se evidencian en las figuras 16 y 18, a primera vista se pudiese concluir que no se obtuvo una buena calidad de solución, no obstante, es de notar que esto se debe a dos características, la primera es, que las unidades de crédito de las asignaturas que conforman algunos de los *curricula* asociados no son uniformes, es decir, existen materias con unidades de crédito muy altas y otras muy bajas, por lo que realizar una distribución balanceada se torna imposible, ya que las unidades de crédito de una asignatura no son divisibles. La segunda característica, se nota en las figuras de UD7, donde en muchos casos, sin importar la cantidad de iteraciones, el resultado no varía. Esto se debe a que en una gran parte de estos casos, hay solución única.

Se puede afirmar finalmente, a través de todas las tablas y figuras de este capítulo, que siempre que existan múltiples soluciones para un caso de prueba, el algoritmo es capaz de calcular una buena solución en muy corto tiempo en comparación con las implementaciones ya realizadas por los antecedentes. Adicionalmente, no es necesario realizar miles de iteraciones del algoritmo para conseguir mejoras en la mayoría de los casos, sin embargo, es posible aumentar este número para casos que de antemano se sepan más complejos.

CONCLUSIONES Y RECOMENDACIONES

El principal objetivo de este trabajo, ha sido evidenciar la capacidad que poseen la metaheurísticas para resolver problemas de categoría combinatoria y/o de múltiples restricciones, específicamente para el caso de diseñar o rediseñar una malla curricular en un periodo de tiempo menor al que tomaría realizarlo manualmente o con programación lineal.

Para esta solución se utilizó el esquema planteado para GRASP, con la excepción de que existieron mejoras en el procesamiento previo de los datos, para que el algoritmo redujera el número de combinaciones.

Debido a los resultados obtenidos, se puede concluir en primera instancia, que GRASP es capaz de resolver problemas cuyas características sean similares al problema de balanceo de una carga curricular, tal es el caso de, la asignación de puestos de trabajo entre empleados para una línea de producción, entre otras aplicaciones en las que el volumen de datos sea amplio, por lo que no pueda encontrarse una solución factible en un tiempo prudente. Es importante resaltar, que si bien GRASP no necesariamente consigue la solución óptima, es capaz de conseguir buenas alternativas en corto tiempo, como se demostró en las tablas y figuras de este capítulo. El *curriculum* de la figura 12, demuestra que el algoritmo es capaz de conseguir soluciones óptimas según el caso.

En el caso de la FaCyT en la UC, se pudo evidenciar, que existe la posibilidad numérica de mejorar la distribución de las unidades de crédito, correspondiente a los períodos o semestres de cada una de las carreras ofrecidas.

Previo a la realización de cualquier trabajo orientado al uso de técnicas metaheurísticas para resolver problemas combinatorios, es importante diseñar correctamente la estructura de datos que se utilizará en el algoritmo. Ya que, puede haber problemas combinatorios cuya cantidad de datos, supere la memoria principal de un computador regular, por lo que sea necesario tener acceso a memoria secundaria, influyendo así negativamente sobre los resultados en función del tiempo.

Es fundamental, además, elegir correctamente la técnica metaheurística a utilizar, si bien, cada una busca resolver problemas de formas distintas, hay que profundizar en los elementos propios del problema, para ubicarlo en una categoría. Tal es el caso de problemas combinatorios con restricciones, de manera, que la metaheurística haya sido probada en algún otro problema con características similares al que se desea resolver, claramente, sólo si ha obtenido buenos resultados.

A partir de este estudio, se puede plantear de trabajo futuro, la creación de un conjunto de nuevas soluciones, que aprovechen la estructura de datos establecida utilizando otras metaheurísticas, con el objetivo de optimizar bien el tiempo de procesamiento o la calidad de la solución. En su defecto, se puede plantear una nueva estructura de datos, utilizando el algoritmo metaheurístico propuesto, con el objetivo de probar el comportamiento en tiempo de procesamiento, de una estructura de datos fundamentada, por ejemplo, en el uso de memoria secundaria.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar, José Antonio. Martínez, José, Ríos, Mauricio y Nuño, José. (2007). **El problema de balancear un plan de estudios: Un modelo matemático.** 2° Taller Latino Iberoamericano de Investigación de Operaciones, 2007.

Aguilar, José Antonio, Martínez, José. (2009). **Estudio de diferentes conceptos de balance en modelos BACP.** 3° Taller Latino Iberoamericano de Investigación de Operaciones, 2009.

Arias, Fidias (2006). **El proyecto de investigación: Introducción a la metodología científica.** (5º. Ed.)

Blum C, Roli A (2003). **Metaheuristics in combinatorial optimization: Overview and conceptual comparison**. ACM Computing Surveys CSUR. p. 268-308

Briones, Guillermo (1995). **Métodos y Técnicas de Investigación**. Editorial Trillas.

Castro, Carlos y Manzano, Sebastián (2001). ***Variable and Value Ordering When Solving Balanced Academic Curriculum Problems.*** Proceedings of the 6th Annual Workshop of the ERCIM Working Group on Constraints.

Chiarandini, Marco, Di Gaspero, Luca. (2011) **The Balanced Academic Curriculum Problem Revisited.** Submitted to Journal of Heuristics

Falcón, Julio y Herrera, Roberto (2005). **Análisis del dato Estadístico (Guía didáctica)**. (1º Ed)

Gómez Mendoza, Miguel Angel (1999). **Análisis de contenido cualitativo y cuantitativo: Definición, clasificación y metodología. Revista de ciencias humanas número 20.**

Hernandez, Roberto, Fernández, Carlos y Baptista, Pilar. (2006). **Metodología de la Investigación**. México: McGraw-Hill.

Kelly, James y Osman, Ibrahim. (1996) **Meta-Heuristic: Theory and Applications**. Kluwer Academic Publisher.

Laurence, Bardin. ***El análisis de contenido***. (2002) ISBN: 84-7600-093-6. Akal Ediciones.

Lambert, Tony, Castro, Carlos, Monfroy, Eric y Saubion, Frédéric. (2005) ***Solving the Balanced Academic Curriculum Problem with an Hybridization of Genetic Algorithm and Constraint Propagation.*** Lecture Notes in Computer Science, 3668. 421-423.

Martí, Rafael (2003). **Procedimientos Heurísticos en Optimización Combinatoria.** Matematiques, vol. 1, No 1, p. 3-62.

Martínez, A y Rosquete, D (2010). **NASPI: Una notación algorítmica estándar para programación imperativa.** Telematique. Vol 8, No. 3.

Murillo, William (2007). **La investigación científica**. Publicación en línea

Ríos, Insua, Caballero, Alfonso, Bielza, María y Jimenez, Antonio. (2004) **Investigación operativa: modelos determinísticos y estocásticos.** Editorial Centro de Estudios Ramón Areces, S.A. ISBN 848004666X

Sabino, Carlos. (2000). **El Proceso de la Investigación.** Caracas: Panapo.

Schaeffer, Elisa. (2011). **Complejidad computacional de problemas y el análisis y diseño de algoritmos.** Publicación en línea.

Zanakis, Stelios H. y Evans, James R (1981). **Heuristic "Optimization": Why, When, and How to Use It.** Interfaces Journal. DOI: 10.1287/inte.11.5.84 Vol. 11, No. 5