PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL PROBLEMA DE BALANCEO DE MALLAS CURRICULARES MEDIANTE BCO

YEISON ISMAEL SEVERINO ROMERO

INFORME FINAL DEL PROYECTO PARA OPTAR AL TÍTULO
PROFESIONAL DE INGENIERO CIVIL EN INFORMÁTICA

Junio 2015

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL PROBLEMA DE BALANCEO DE MALLAS CURRICULARES MEDIANTE BCO

YEISON ISMAEL SEVERINO ROMERO

Profesor Guía: **Ricardo Soto De Giorgis**

Carrera: **Ingeniería Civil Informática**

Junio 2015

Dedicatoria

A mi familia, amigos y amigas por todo el apoyo recibido.

Índice

LISTA DE FIGURAS ..... **IV**

RESUMEN ..... **V**

LISTA DE ABREVIATURAS ..... **VI**

1 INTRODUCCIÓN **1**

2 DEFINICIÓN DE OBJETIVOS **2**

2.1 OBJETIVO GENERAL 2
2.2 OBJETIVOS ESPECÍFICOS 2

3 ESTADO DEL ARTE **3**

4 MARCO TEÓRICO **4**

4.1 MALLA CURRICULAR EN LA ENSEÑANZA SUPERIOR 4
4.1.1 Organización de la Malla Curricular *4*
4.1.2 Sistema de Créditos *5*
4.2 BACP 6
4.2.1 Formulación del BACP *6*
4.2.2 Modelo matemático *7*
4.3 CONCEPTOS RELACIONADOS AL ÁREA DE OPTIMIZACIÓN 8
4.3.1 Optimización Combinatoria *8*
4.3.2 Instancia de un Problema de Optimización *8*
4.3.3 Vecindario *9*
4.3.4 Óptimo Local *9*
4.3.5 Óptimo Global *9*
4.4 BÚSQUEDA LOCAL 9
4.5 METAHEURÍSTICAS 10
4.6 OPTIMIZACIÓN BASADA EN COLONIAS DE ABEJAS 11
4.6.1 Las colonias de abejas naturales *11*
4.6.2 Del comportamiento de las abejas naturales hacia metaheurística BCO *11*
4.6.3 Elección del algoritmo basado en la metaheurística BCO *12*
4.7 ALGORITMO ABC 12
4.7.1 Esquema general del algoritmo *14*
4.7.2 Fase de inicialización *14*
4.7.3 Fase de abejas empleadas *15*
4.7.4 Fase de abejas observadoras *16*
4.7.5 Fase de Abejas exploradoras *17*
4.7.6 Memorizar la mejor solución *18*
4.7.7 Función de reparación *18*

5 PROPUESTA DE SOLUCIÓN **20**

5.1 ESQUEMA DE SOLUCIÓN PROPUESTA 20
5.2 ESTRUCTURA DE RESOLUCIÓN 21
5.3 LENGUAJE DE PROGRAMACIÓN Y ENTORNO DE IMPLEMENTACIÓN 22
5.4 RESULTADOS DEL ALGORITMO 22
5.4.1 BACP8 ..... *22*
5.4.2 BACP10 ..... *23*

5.4.3 BACP12 ..... *23*
5.4.4 BACP ICI PUCV malla antigua *24*
6 CONCLUSIONES **25**
7 REFERENCIAS **26**

Lista de Figuras

FIGURA 4. 1 MALLA CURRICULAR UNIVERSITARIA 5
FIGURA 4. 2 BÚSQUEDA DE SOLUCIONES POR MEDIO DE ABEJAS 13
FIGURA 5. 1 ESQUEMA DE LA SOLUCIÓN AL BACP 20
FIGURA 5. 2 FUENTES DE ALIMENTO VECTORIALES 21

Resumen

El problema de Balanceo de Mallas Curriculares (BACP) consiste en la asignación de cursos a periodos académicos con el fin de que la carga académica se encuentre balanceada, es decir, que todos los semestres tengan un número similar de créditos y que la mayor carga académica entre todos los periodos sea la menor posible respetando ciertas restricciones impuestas por las diversas casas de estudio. Para diseñar una malla curricular balanceada existen diversas técnicas dentro del área de optimización. En esta investigación, se resolvió este problema usando Optimización basada en Colonia de Abejas (BCO).

BCO consiste en el uso de abejas artificiales que imitarán el comportamiento que tienen las abejas reales en la búsqueda de néctar. Esta metaheurística puede ser aplicada para resolver diversos problemas de optimización. En el presente documento, se resolverán tanto instancias de prueba como instancias reales con el fin de medir el rendimiento del algoritmo y entregar soluciones que permitan simplificar el proceso de diseño de una malla curricular.

Palabras clave: Problema de balanceo de mallas curriculares, Optimización basada en colonia de abejas, Algoritmo basado en colonia de abejas artificiales, Metaheurísticas.

Abstract

The Balanced Academic Curriculum Problem (BACP) consists in the assignment of courses to academic periods in order that the academic load is balanced, i.e., all semesters have a similiar number of credits and the greater academic load between all the periods is the smallest possible respecting certain restrictions imposed by the given university. To design a balanced curriculum there are several techniques within the area of optimization. In this investigation, this problem was solved using Bee Colony Optimization (BCO).

BCO consists in the use of artificial bees that will mimic the behavior of the real bees in the search for nectar. This metaheuristic can be applied to solve various optimization problems. In this document, both test instances and real instances will be solved in order to measure the performance of the algorithm and provide solutions that simplify the process of designing a curriculum

Keywords: Balanced Academic Curriculum Problem, Bee Colony Optimization, Artificial Bee Colony Algorithm, Metaheuristics.

Lista de Abreviaturas

ABC	: Artificial Bee Colony
BACP	: Balanced Academic Curriculum Problem
BCO	: Bee Colony Optimization
BO	: Bee Optimization
BSO	: Bee Swarm Optimization
CSP	: Constraint Satisfaction Problem
CSPLib	: Library of test problems for constraint solvers
EA	: Evolutionary Algorithm
EDA	: Estimation Distribution Algorithm
GBACP	: Generalized Balanced Academic Curriculum Problem
GRASP	: Greedy Randomize Adaptative Search Procedures
HBCO	: Hibrid Bee Colony Optimization
ILS	: Iterated Local Search
OCH	: Optimización basada en Colonia de Hormigas
PSO	: Particle Swarm Optimization
SA	: Simulated annealing
SI	: Swarm Intelligence
SS	: Scatter Search
TS	: Taboo Search
VNS	: Variable Neighbourhood Search

1 Introducción

Cuando se diseñan mallas curriculares para una carrera Universitaria se toman en consideración factores como la cantidad de asignaturas que posee la carrera, el número de periodos en el que se deben asignar estos cursos, la carga académica mínima y máxima aceptada, y el número de cursos permitido por semestre. Con toda esta información se conforman las restricciones o regulaciones académicas relacionadas al plan de estudios, las cuales son usadas para su construcción mediante un enfoque de prueba y error hasta lograr una malla adecuada.

Ya que el grado de esfuerzo que se requiere para aprobar una asignatura se puede medir en créditos, el éxito que puedan tener los estudiantes tiene directa relación con la carga académica que enfrenten en cada periodo. La carga académica corresponde al número de créditos por semestre. Es por esta razón que las mallas curriculares deben estar balanceadas, es decir, la cantidad de créditos para cada periodo debe ser similar para que la carga que enfrenten los alumnos sea la menor posible. Por lo tanto, es de interés minimizar este costo incurrido en diseñar un plan de estudios mediante un algoritmo que realice este esfuerzo de manera automática y sin incurrir a errores.

Este problema es conocido en la literatura como el problema de balanceo de mallas curriculares (BACP), y es del tipo CSP (Problema de Satisfacción de Restricciones), en el cual se buscan satisfacer todas las restricciones asociadas y luego optimizar la calidad de la solución encontrada. Existen prototipos y modelos estudiados que resuelven el BACP. Generalmente este problema ha sido abordado utilizando el paradigma de programación con restricciones y algoritmos híbridos, para los cuales se utilizaron algoritmos genéticos, esquemas de colaboración, búsquedas locales, etc.

Esta investigación se centra en resolver tanto las instancias de prueba reconocidas por la CSPLib y una instancia real. Esta instancia corresponde a la carrera de Ingeniería Civil Informática de la Pontificia Universidad Católica de Valparaíso. Se utilizará la metaheurística de optimización basada en colonia de abejas (BCO) a través del algoritmo Artificial Bee Colony (ABC) propuesto por Karaboga [15].

A continuación, se definirán los objetivos, se expondrá el estado del arte del tema y se detallará el marco teórico y contexto del problema. Luego, se presentará una propuesta de solución con un esquema definido y detallado. Finalmente se presenta la conclusión de la investigación.

2 Definición de objetivos

A continuación, se detallan cada uno de los objetivos, tanto general, como específicos.

2.1 Objetivo general

Resolver el problema del balanceo de mallas curriculares mediante la optimización basada en colonia de abejas.

2.2 Objetivos específicos

- Desarrollar un modelo de balanceo de mallas curriculares basado en optimización con colonia de abejas (BCO).
- Diseñar un esquema de solución del problema de balanceo de mallas curriculares.
- Implementar un algoritmo de balanceo de mallas curriculares utilizando optimización basado en colonia de abejas (BCO).
- Evaluar el rendimiento del algoritmo propuesto mediante la utilización de instancias.

3 Estado del Arte

El problema de balanceo de mallas curriculares (BACP) fue presentado por Castro y Manzano [1] quienes solucionaron dicho problema minimizando la máxima carga académica por estudiante. Además, para resolver este problema, utilizaron programación con restricciones, y demostraron al mismo tiempo que es una técnica más eficiente en comparación a la programación lineal entera, para resolver problemas de optimización combinatorios.

También, el BACP fue abordado por Hnich, Kiziltan y Walsh [2], quienes utilizaron programación con restricciones y programación lineal entera, a modo de solución. A través de su trabajo, integraron los distintos modelos de tal manera que se complementaran las fortalezas de cada uno. Los resultados experimentales dejaron en claro, que al integrar los modelos se acotaba mejor el dominio de las variables del problema, lo que producía una disminución en el tiempo de ejecución para resolver varias instancias.

Posteriormente, Lambert, Castro, Monfroy y Saubion [3] utilizaron técnicas híbridas compuestas por algoritmos genéticos y propagación de restricciones. Trabajando en conjunto, desarrollaron un modelo teórico, por medio del cual lograron una resolución híbrida. Además, fueron capaces de calcular la solución óptima para dos instancias de prueba pequeñas, ambas presentadas en la CSPLib, en menos de 40 segundos. En relación a la tercera instancia de prueba, ésta no pudo ser resuelta incluso después de una hora de ejecución.

Monette, Schaus, Zampelli, Deville y Dupont [4], fueron quienes estudiaron el BACP de forma extensa. Lograron resolver las tres instancias de prueba de la CSPLib de manera óptima, incluso, introdujeron un generador de instancias aleatorias, con el fin de obtener más instancias desafiantes. Enfocaron sus experimentos en 720 instancias, con 200 cursos y variadas características. Lo más destacado y/o relevante, es que centraron su atención al criterio de balanceo de carga; extendiendo la idea original de carga máxima sobre los periodos, ya utilizada en los trabajos previos, hacia la máxima desviación de la carga promedio, y la suma cuadrática y lineal de tales desviaciones.

Carlos Castro y Broderick Crawford resolvieron modelos matemáticos, utilizando técnicas completas e incompletas para diseñar mallas curriculares balanceadas. Luego, José Miguel Rubio, et al., publicaron un trabajo en el cual utilizaron optimización basada en colonia de hormigas, para resolver el problema de balanceo de mallas curriculares [5].

En el trabajo titulado “The Balanced Academic Curriculum Problem Revisited”, publicado por Chiarandini, Di Gaspero, Gualandi y Schaerf, se deja en claro que el problema teórico es más simple, en comparación al problema real que deben resolver las universidades en la práctica. Además, propusieron superar tal brecha definiendo una nueva formulación, más generalizada, que incluye la anterior formulación, de tal manera que se modela la situación de una forma más real. Esta nueva formulación hace posible incluir más de un currículo, con cursos compartidos entre planes de estudio, e incluso, contiene las preferencias de los profesores. Esta nueva formulación del problema se conoce como Problema Generalizado de Balanceo de Mallas Curriculares [6] (GBACP). Además, se introdujeron 10 nuevas instancias obtenidas de la Universidad de Udine. Debido a su origen diferente, estas instancias son mucho más grandes y tienen estructuras distintas. Finalmente, esta formulación y las nuevas instancias resultaron ser más difíciles de resolver que las instancias previas. Por lo que no se pudo llegar a una solución óptima.

4 Marco Teórico

En el siguiente apartado se describen los conceptos más relevantes para la comprensión del trabajo realizado. Los temas a abordar corresponden a la malla curricular en sí, la optimización basada en colonia de abejas, formulación del BACP, entre otros tópicos derivados.

4.1 Malla Curricular en la Enseñanza Superior

Tradicionalmente se ha designado el plan de estudios como un conjunto de asignaturas y actividades graduadas, sistematizadas y armonizadas, de manera que concurran a la obtención de un objetivo o grupo de objetivos, correspondientes a un nivel educativo. Dentro de los esquemas de la pedagogía moderna, el plan de estudios es denominado Currículo, y se le define como "El conjunto de enseñanzas, teorías y prácticas, que han de realizar para ser promovidos los alumnos, como el orden de ellos dentro de una institución docente" [7].

4.1.1 Organización de la Malla Curricular

Las asignaturas que se dictan en las instituciones de educación superior se pueden clasificar de la siguiente manera (Figura 4.1):

- ***Cursos básicos a nivel general:*** Corresponden a las asignaturas que proporcionan a los estudiantes una cultura básica universitaria en distintas áreas, como las ciencias o las humanidades.
- ***Cursos requisitos del programa académico:*** Son las asignaturas que deben ser estudiadas y aprobadas por todos los estudiantes que siguen un programa académico. Los cursos requisitos del programa académico son seleccionados, coordinados y evaluados por la dirección del programa académico respectivo.
- ***Cursos electivos:*** Estas son las asignaturas en que el estudiante puede elegir entre varias posibilidades sin restricciones, pero siempre deben cumplir el número total de créditos que le exige el currículo de su programa académico.

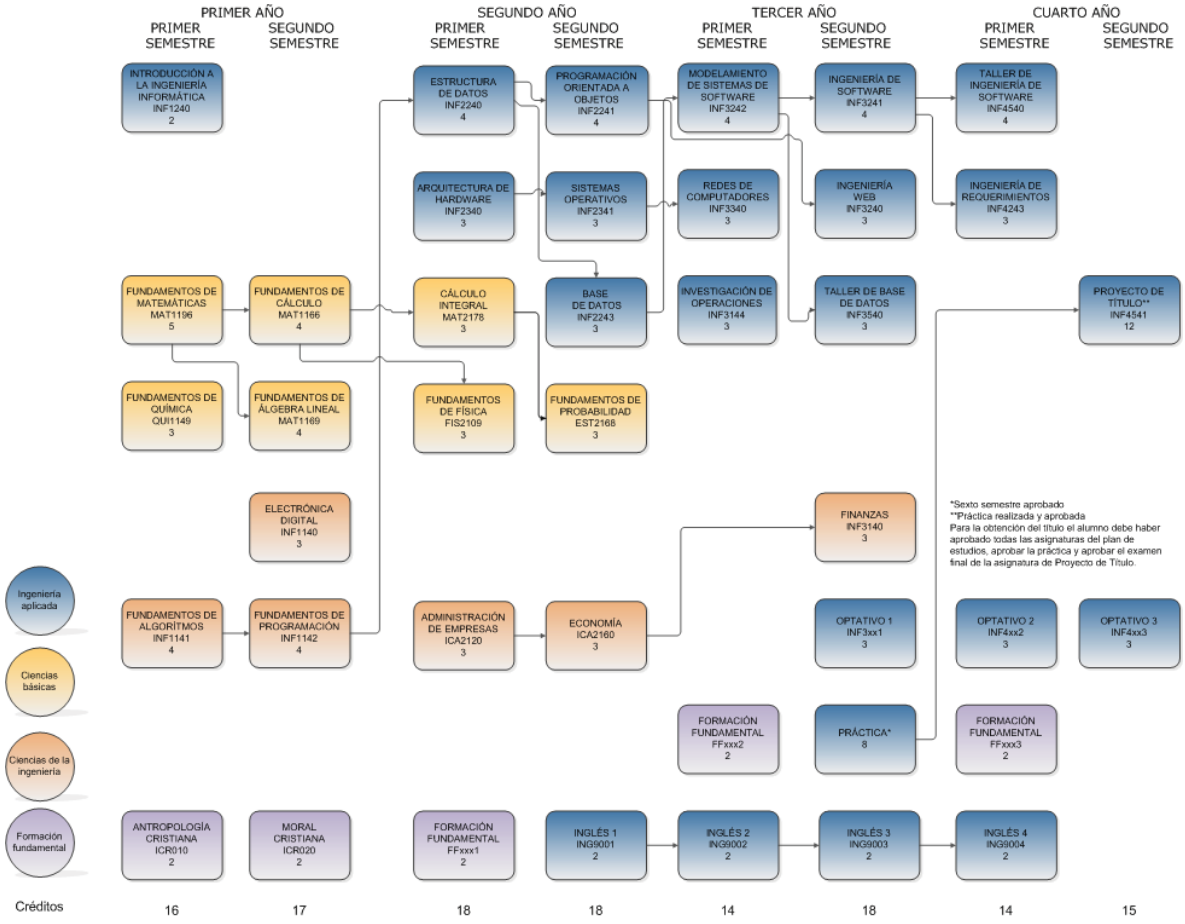


Figura 4. 1 Malla curricular Universitaria

4.1.2 Sistema de Créditos

La flexibilidad de la malla curricular requiere un instrumento operativo que permita estimar el trabajo académico de los estudiantes, y traducirlo en cifras que revelen en cualquier instante su situación y progreso dentro de la universidad. Por lo tanto, el crédito es una unidad de evaluación del trabajo efectuado para aprobar una asignatura.

Expresando en créditos el volumen de trabajo que representa cada curso, se facilita el establecimiento de pautas que regulen la distribución equilibrada de asignaturas en la malla curricular y el monto de la carga académica ponderada del rendimiento del estudiante en las diversas materias.

Los créditos representan un procedimiento exacto y racional para dar significado a las calificaciones e interpretar en cifras la situación académica real de los estudiantes. El sistema permite establecer niveles mínimos de rendimiento y determinar cuándo un alumno está encontrando dificultades en sus estudios. Un índice ponderado demasiado bajo es una señal de alarma sobre la situación académica del alumno y sus probabilidades de fracasar como estudiante.

4.2 BACP

El Problema de Balanceo de Mallas Curriculares (BACP) busca asignar cursos a los periodos académicos, de manera que la carga académica de cada semestre esté balanceada, esto quiere decir, que cada periodo debe tener una cantidad similar de créditos. Además, la malla debe respetar las restricciones de número mínimo y máximo de créditos por periodos y que también se cumplan todos los prerrequisitos. Por lo tanto, una malla curricular balanceada óptima, tiene por función objetivo minimizar la máxima carga académica para cada periodo. Generalmente, este problema ha sido abordado utilizando el paradigma de programación con restricciones, algoritmos híbridos, algoritmos genéticos, esquemas de colaboración, búsquedas locales, entre otras [3].

A continuación, se presenta la formulación básica del BACP como se propone en [1].

4.2.1 Formulación del BACP

La formulación básica consta de las siguientes entidades y restricciones:

- ***Cursos:*** Cada curso tiene un número entero de créditos, y tiene que ser dictado durante el plan de estudios de la carrera universitaria.
- ***Periodos:*** El plan de estudios, se divide en años académicos, y cada año académico se divide por un número fijo de intervalos de tiempo. Cada intervalo, es un periodo lectivo en el que los cursos pueden ser dictados. Por ejemplo, una carrera universitaria de 4 años académicos, dividida en 2 intervalos al año, tiene 8 períodos en total.
- ***Límites de carga:*** Para cada periodo, existe un número máximo de cursos que se pueden asignar al mismo. Además, también existen límites mínimos y máximos en el número de créditos totales por periodo.
- ***Prerrequisitos:*** En base a su contenido, algunos cursos tienen que ser dictados antes que otros cursos. Esto significa que existe un conjunto de pares de cursos, de manera que el periodo asignado para el primer curso tiene que ser estrictamente menor que el periodo asignado al segundo. Obviamente, la relación prerrequisito es transitiva y no puede contener ciclos.
- ***Distribución balanceada de la carga:*** La distribución de los créditos entre los períodos de enseñanza debe ser equilibrada. Lo ideal, sería que cada periodo lectivo tenga el mismo número de créditos.

Es así como el problema consiste en poder encontrar una asignación de cursos para los períodos, de tal manera que satisfaga todos los límites de carga, de prerrequisitos y minimice el número máximo de créditos totales por período.

Para resolver el BACP, se utilizan instancias de pruebas definidas en la CSPLib. Por ejemplo, la instancia BACP8 CSPLib cuenta con 46 cursos para un total de 133 créditos y 8 períodos. El número promedio de créditos por período es de 133/8 = 16.625, así, el límite inferior del número máximo de créditos por período es 17. Por lo tanto, soluciones con valor 17 son de esta forma óptimas.

4.2.2 Modelo matemático

El modelo de programación lineal entera está basado en la propuesta para resolver el problema de balanceo de mallas curriculares usado por Castro C. y Manzano S. que en [1]introdujeron este problema. El modelo plantea una variable de decisión de una dimensión para la resolución del problema.

- Parámetros:
  - m : Número de cursos
  - n : Número de periodos académicos
  - α_i : Número de créditos de curso i , donde i=1 ,…, m
  - β : Carga académica mínima por periodo
  - γ : Carga académica máxima por periodo
  - δ : Cantidad mínima de cursos por periodo
  - ε : Cantidad máxima de cursos por periodo

- Variables de decisión:
  - Vector con los periodos que cada curso tiene asignado
    $$x_i = j \ , \forall i = 1, \dots, m \qquad (4.2.2.1)$$
  - c : es la máxima carga académica para todos los periodos:
    $$c = \text{Max}\{c_1, \dots, c_n\} \qquad (4.2.2.2)$$
  - La carga académica del periodo j está definida por:
    $$c_j = \sum_{i=1}^{m} \alpha_i \zeta_i, \ \forall i = 1, \dots, m, \forall j = 1, \dots, n \qquad (4.2.2.3)$$

    $$\zeta_i = \begin{cases} 1 \ si \ x_i = j \\ 0 \ si \ x_i \neq j \end{cases} \qquad (4.2.2.4)$$

- Función Objetivo:
  Min c

- Restricciones:
  - Todos los cursos i deben ser asignados a un periodo j:
    $$0 < x_i \leq n \ , \forall i = 1, \dots, m \qquad (4.2.2.5)$$
  - El curso a tiene un curso b como prerrequisito:
    $$x_b < x_a \ , \quad 1 \leq x_a, x_b \leq n \qquad (4.2.2.6)$$
  - La carga académica máxima está definida por: c = Max $\{c_1, \dots, c_n\}$. Lo que puede ser representado por el siguiente conjunto de restricciones lineales:
    $$c_j \leq c \ , \forall j = 1, \dots, n \qquad (4.2.2.7)$$

  - La carga académica del periodo j debe ser mayor o igual que el mínimo requerido:
    $$c_j \geq \beta \ , \forall j = 1, \dots, n \qquad (4.2.2.8)$$

- La carga académica del periodo j debe ser menor o igual que el máximo requerido:

$$c_j \leq \gamma\ , \forall j = 1, \dots, n \tag{4.2.2.9}$$

- El número de cursos del periodo j debe ser mayor o igual que el mínimo requerido:

$$\sum_{i=1}^{m} \zeta_i\ \geq \delta, \forall j = 1, \dots, n \tag{4.2.2.10}$$

$$\zeta_i = \begin{cases} 1\ si\ x_i\ =\ j \\ 0\ si\ x_i\ \neq\ j \end{cases} \tag{4.2.2.11}$$

- El número de cursos del periodo j debe ser menor o igual que el máximo requerido:

$$\sum_{i=1}^{m} \zeta_j \leq \varepsilon, \forall j = 1, \dots, n \tag{4.2.2.12}$$

$$\zeta_i = \begin{cases} 1\ si\ x_i\ =\ j \\ 0\ si\ x_i\ \neq\ j \end{cases} \tag{4.2.2.13}$$

4.3 Conceptos Relacionados al Área de Optimización

En esta sección se describieron algunos conceptos relacionados al área de optimización combinatoria con el fin de lograr una mayor comprensión de la naturaleza del problema y como se resolverá.

4.3.1 Optimización Combinatoria

Es una rama de la optimización en matemáticas aplicadas y en ciencias de la computación, relacionada a la investigación de operaciones, teoría de algoritmos y teoría de la complejidad computacional. Los algoritmos de optimización combinatoria resuelven instancias de problemas que se creen ser difíciles en general, explorando el espacio de soluciones (usualmente grande) para estas instancias. Los algoritmos de optimización combinatoria logran esto reduciendo el tamaño efectivo del espacio, y explorando el espacio de búsqueda eficientemente.

4.3.2 Instancia de un Problema de Optimización

Corresponde a un par (*F*, *c*), donde "*F*" es cualquier conjunto dentro del dominio de puntos factibles y "c" es la función de costos, tal que [7]:

$$c: F\ \rightarrow\ \mathbb{R} \tag{4.3.2.1}$$

El problema es encontrar un $f\ \in F$, tal que:

$$(c(f) \leq\ c(y), \forall\ y \in F \tag{4.3.2.2}$$

4.3.3 Vecindario

Dado un punto factible $f \in F$ en una instancia, se define vecindario al conjunto de puntos $N(f)$ que están "cerca" del punto f.

4.3.4 Óptimo Local

Dada una instancia (*F, c*) de un problema de optimización y un vecindario *N*, una solución *f* es llamada óptimo local respecto de *N* si:

$$(c(f) \leq c(g), \forall\, g \in N(f) \tag{4.3.2.3}$$

4.3.5 Óptimo Global

El óptimo global se puede definir como el mejor óptimo local, de todos los vecindarios en una instancia de un problema de optimización. Dada una instancia (*F, c*) de un problema de optimización y $N(f)$*, el conjunto de todos los vecindarios posibles, *f* será óptimo global si:

$$(c(f) \leq c(y), \forall\, y \in N(f) * \tag{4.3.2.4}$$

4.4 Búsqueda Local

La búsqueda local se basa en lo que es quizás el método más antiguo del área de optimización, la prueba y error. La idea es simple y natural, de tal modo que ha tenido éxito en una gran serie de problemas de optimización combinatorios [10].

Dada una instancia (F, c) donde "F" es cualquier conjunto dentro del dominio de puntos factibles y "c" es la función de costos, se elige un vecindario:

$$N: F \rightarrow 2^{\mathrm{F}} \tag{4.4.1}$$

Que se busca en el punto t ∈ F para mejoras usando la subrutina:

$$mejora(t) = \begin{cases} \text{s} \in N(t) \text{donde } c(s) < c(t), \text{si existe} \\ No, \text{en caso contrario} \end{cases} \tag{4.4.2}$$

La idea es iniciar con una solución generada aleatoriamente hallada con algún otro algoritmo, para luego aplicar una transformación de algún conjunto dado de transformaciones para mejorar la solución actual. Luego la solución mejorada se convierte en la solución actual y este proceso se repite hasta que ninguna transformación del conjunto mejore la solución actual.

La solución encontrada no necesariamente será la óptima, ya que depende de las transformaciones aplicadas. Cuando no es posible obtener un mejoramiento en la solución el algoritmo se detiene en un óptimo local.

4.5 Metaheurísticas

Clase de algoritmos aproximados que no garantizan que la solución encontrada sea el óptimo global, cuya idea básica es combinar diferentes métodos heurísticos a un nivel más alto para conseguir una exploración del espacio de búsqueda de forma eficiente y efectiva. Las metaheurísticas presentan las siguientes propiedades [11]:

- Las metaheurísticas son estrategias o plantillas generales que guían el proceso de búsqueda.
- El objetivo es una exploración eficiente del espacio de búsqueda para encontrar soluciones (casi) óptimas.
- Las metaheurísticas son algoritmos no exactos y generalmente son no deterministas (tipo de algoritmo que dada una entrada, no siempre produce la misma salida).
- Pueden incorporar mecanismos para evitar regiones no prometedoras del espacio de búsqueda.
- El esquema básico de cualquier metaheurística tiene una estructura predefinida.
- Las metaheurísticas pueden hacer uso de conocimiento del problema que se trata de resolver en forma de heurísticos específicos que son controlados por la estrategia de más alto nivel.

Resumiendo estos puntos, se puede acordar que una metaheurística es una estrategia de alto nivel que usa diferentes métodos para explorar el espacio de búsqueda. Las metaheurísticas se clasifican en dos grupos: las metaheurísticas basadas en trayectoria y las metaheurísticas basadas en población. Las primeras manipulan un elemento en el espacio de búsqueda, mientras que las segundas trabajan sobre un conjunto de elementos (población).

- Basadas en trayectoria encontramos:
    - Enfriamiento simulado (SA)
    - Búsqueda Tabú (TS)
    - GRASP
    - Búsqueda con vecindario variable (VNS)
    - Búsqueda local iterada (ILS)
- Entre las basadas en población encontramos:
    - Algoritmos evolutivos (EA)
    - Algoritmos de estimación de la distribución (EDA)
    - Búsqueda dispersa (SS)
    - Optimización por enjambre de partículas (PSO)
    - Optimización basada en Colonia de Hormigas (ACO)
    - Optimización basada en Colonia de Abejas (BCO)

En la siguiente sección se puso énfasis únicamente en esta última metaheurística, la optimización basada en colonia de abejas.

4.6 Optimización basada en colonias de abejas

La Optimización basada en colonias de abejas (BCO) es una metaheurística basada en el comportamiento inteligente de los enjambres de abejas para la búsqueda y evaluación de su alimento [12]. BCO corresponde a una especialización de Inteligencia de Enjambre (Swarm Intelligence, SI) donde los miembros del enjambre son abejas. La comunicación entre los miembros del enjambre se realiza a través de la "Danza de las abejas" ("Waggle Dance") para intercambiar información sobre la ubicación de fuentes de comida [13].

4.6.1 Las colonias de abejas naturales

En la naturaleza existen un largo número de diferentes especies de insectos las cuales varían su comportamiento, pero aun así poseen patrones que les permiten ejecutar una variedad de tareas complejas. En este caso, las abejas naturales son capaces de recolectar y procesar las fuentes de néctar de una forma organizada. Cada abeja decide si seguir o no a una compañera cercana que ha descubierto una fuente de flores. En las colmenas se encuentra el área de danza, en la cual la abeja que ha descubierto fuentes de alimento trata de convencer a sus compañeras para que la sigan. Si deciden seguirla, llegarán a la fuente de alimento para cargar el néctar y volver a la colmena [14].

4.6.2 Del comportamiento de las abejas naturales hacia metaheurística BCO

La metaheurística BCO es una aproximación basada en la "cooperación" que brinda un marco de trabajo algorítmico que puede ser aplicado en varios problemas de optimización [14]. En esta investigación se obtienen los siguientes conceptos desde el comportamiento de las abejas reales: Fuentes de comida, abejas empleadas y abejas desempleadas [12].

- ***Fuentes de comida***: El valor de las fuentes de comida depende de muchos factores, tanto la proximidad a la colmena como la calidad del néctar.
- ***Abejas empleadas***: Éstas están asociadas con una fuente de alimento particular, la cual están "explotando". Estas abejas transportan información sobre una fuente de alimento particular (ubicación y calidad) para luego compartirla con una cierta probabilidad.
- ***Abejas desempleadas***: Estas abejas están observando fuentes de alimento para explotar. Existen dos tipos; Abejas exploradoras, las cuales buscan alrededor del ambiente de la colmena por nuevas fuentes de comida; y las Abejas observadoras esperan en la colmena y encuentran fuentes de comida a través de la información traspasada por las abejas empleadas.

El intercambio de información es un punto importante para la formación de conocimiento colectivo de las abejas. Cuando se examinan las colmenas es posible distinguir partes que son comunes para todas. La comunicación ocurre en el área de Danza para intercambiar la información sobre la calidad de una fuente de alimento. Las abejas observadoras pueden llegar a observar numerosas danzas, es aquí cuando la probabilidad de elegir una fuente de alimento para explotar es de suma importancia para asegurar la explotación de una fuente con calidad alta [12].

4.6.3 Elección del algoritmo basado en la metaheurística BCO

La Computación Evolutiva (Evolucionary Computing, EC) es una clase aproximación algorítmica basada en la evolución biológica o comportamientos biológicos de los miembros de una población. Dentro de este espectro se encuentra BCO como una clase de SI. Dentro de las SI se encuentran los algoritmos relacionados con el comportamiento de hormigas, aves, peces, entre otros. BCO como marco de trabajo algorítmico tiene muchas variantes, algunas conocidas como: BSO (Bee Swarm Optimizacion), ABC (Artificial Bee Colony), BO (Bee Optimization), HBCO (Hybrid Bee Colony Optimization), entre muchas otras [13].

La variante a elegir es ABC basada en el trabajo realizado por Karaboga D. [15] el año 2005. La elección se basa en los resultados obtenidos por Karaboga D. expuestos el año 2008 en el cual demuestra que el rendimiento del algoritmo ABC se comporta de mejor manera comparado con la evolución diferencial, la optimización de enjambre de partículas y los algoritmos evolutivos para problemas multidimensionales. El rendimiento de ABC es muy bueno en términos de optimización local y global en la selección de abejas empleadas y en el mecanismo para la producción de vecindarios [12].

4.7 Algoritmo ABC

El algoritmo de Colonias de Abejas Artificiales es un algoritmo de optimización propuesto por Dervis Karaboga en el año 2005 [15], el cual se basa en el comportamiento inteligente de búsqueda de fuentes de alimento de las colonias de abejas. En el algoritmo ABC, el espacio de búsqueda se simula como la búsqueda de alimento en el medio ambiente, donde cada punto en el espacio de búsqueda corresponde a una fuente de alimento que las abejas artificiales podrían explotar. La posición de una fuente de alimento representa una posible solución al problema de optimización, y la cantidad de néctar que ésta posea se asocia con la calidad (fitness) de dicha solución. Es así, como las abejas salen de su colmena (Hive) para buscar fuentes de alimento de distinta calidad es un espacio de soluciones (Figura 4.1):

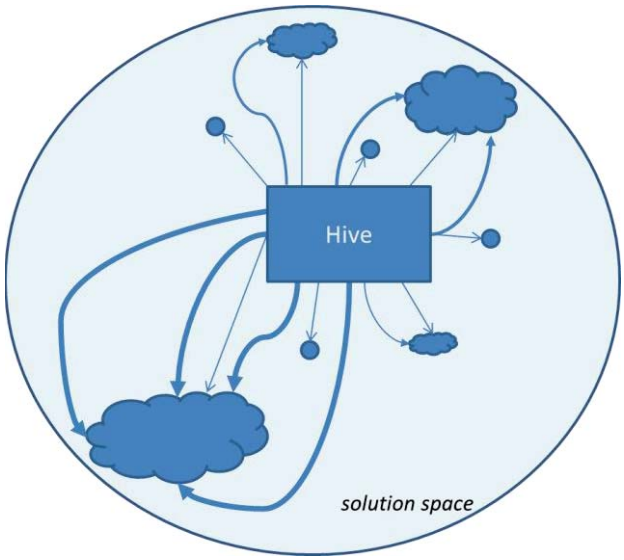


Figura 4. 2 Búsqueda de soluciones por medio de abejas

Dentro de la colonia de abejas, existen dos tipos de abejas: abejas empleadas y desempleadas. Las abejas empleadas se encargarán de explotar las fuentes de alimento. Las abejas desempleadas explotarán las fuentes de alimento según la información entregada por las abejas empleadas. Una vez agotada una fuente de alimento las abejas toman el rol de exploradoras para buscar nuevas fuentes de alimento.También se presentan las siguientes consideraciones:

- La mitad de la colonia se compone de abejas empleadas, mientras que la otra mitad incluye a las abejas observadoras.
- Cada fuente de alimento es explotada por solo una abeja empleada.
- La cantidad de fuentes de alimento es igual a la mitad del tamaño de la colonia, que a su vez es exactamente igual a la cantidad de abejas empleadas y observadoras.

Las fuentes de alimento son generadas al azar y asignadas a cada abeja empleada. Éstas van a la fuente asignada en su memoria y determinan la calidad de la solución para luego volver a la colmena y compartir esta información con las abejas observadoras en el área de danza. Las abejas observadoras evalúan la información acerca de las fuentes de alimento, y eligen una dependiente de la cantidad de néctar ésta posea. Mientras mayor sea la cantidad de néctar que la fuente posea mayor será su probabilidad de ser elegida por una abeja observadora. Una abeja empleada cuya fuente de alimento ha sido abandonada se convierte en una abeja exploradora. Estas abejas buscan aleatoriamente nuevas fuentes de alimento en el entorno. Una vez que han cumplido su labor vuelven a ser abejas empleadas.

4.7.1 Esquema general del algoritmo

El algoritmo general ABC se representa de la siguiente manera:

Algoritmo 4.1 – Pseudocódigo del Algoritmo ABC

```
Inicio
      Fase de Inicialización
      Repetir
            Fase de Abejas Empleadas
            Fase de Abejas Observadoras
            Fase de Abejas Exploradoras
            Memorizar la mejor solución lograda hasta ahora
      Hasta (Iteración = Máximo número de iteraciones)
Fin
```

Al inicio de algoritmo se presenta la fase de inicialización para generar las fuentes de alimento aleatoriamente y luego evaluar su calidad. Luego, según el criterio de término del algoritmo, se presenta un ciclo iterativo con las tres fases principales del algoritmo y la memorización de la mejor solución lograda. En las siguientes secciones se detallará cada fase del algoritmo.

4.7.2 Fase de inicialización

En primer lugar se establece el número de la colonia de abejas, el cual nos brindará inmediatamente la cantidad de abejas empleadas, abejas desempleadas y las fuentes de comida (fc). Luego, se asigna el parámetro "Límite" para la cantidad de intentos en las fuentes de comida al ser evaluada por una abeja, y los límites superiores e inferiores para cada parámetro a optimizar [12].

Las fuentes de alimento correspondientes a las soluciones en el espacio de búsqueda. Cada fuente de alimento es representa a un vector correspondiente a los periodos asignados para cada curso. Por esto, la cantidad de parámetros de una fuente de alimentos corresponde a los cursos de un plan de estudio. Dichas fuentes se producen al azar dentro del rango establecido para cada parámetro (periodos académicos):

$$fc_{qi} = x_i^{min} + rand(0,1) * \left(x_i^{max} - x_i^{min}\right), \forall q = 1, \dots, NF \ , \ \forall i = 1, \dots, m \quad (4.7.2.1)$$

NF es el número de fuentes de alimento, y m es la dimensión del problema que representa el número de parámetros a optimizar (cantidad de cursos del currículo). x_i^{min} y x_i^{max} son los límites inferiores y superiores del i-ésimo parámetro (periodo académico asignado).

Una vez generadas las fuentes de alimento se determina el fitness (calidad) de cada una de ellas utilizando:

$$fitness_q = c_q \quad (4.7.2.2)$$

Donde c es la función objetivo por minimizar correspondiente a la máxima carga de los periodos académicos.

Luego, se inicializan en cero los contadores de intentos para cada fuente de alimento, las que irán aumentando a medida que fracase un intento de nueva solución o vuelvan a cero si una nueva solución presenta mejor calidad que el mejor fitness almacenado. A continuación presentamos el pseudocódigo de esta fase:

Algoritmo 4.2 – Pseudocódigo de la Fase de inicialización

```
Inicio
      Repetir
            Repetir
                  Calcular número aleatorio
                  Calcular fuente de alimento
            Hasta (Iteración = Numero de parámetros del problema)
            Calcular función objetivo
            Calcular Fitness
            Inicializar intentos en cero
      Hasta (Iteración = Número de Fuentes de Alimento)
      Inicializar Mínimo Global con 1ra solución
      Inicializar Parámetros Globales con la 1ra fuente de alimento
Fin
```

4.7.3 Fase de abejas empleadas

En esta fase, cada abeja empleada es enviada a la fuente de alimento asignada en su memoria y encuentra una nueva fuente en el vecindario. Esta nueva fuente se produce con:

$$v_{qi} = fc_{qi} + \emptyset(fc_{qi} - fc_{ki}) \qquad (4.7.3.1)$$

Donde k es una fuente de alimento seleccionada al azar distinto de q; i es una dimensión seleccionada aleatoriamente y $\emptyset$ es un número aleatorio distribuido uniformemente dentro del rango [-1,1]. Se debe asegurar que la generación del nuevo valor entre dentro de los parámetros permitidos.

Posteriormente, se calcula el fitness de la nueva fuente de alimento v , y se aplica una selección codiciosa (greedy selection) entre la nueva fuente y la que ya existía en su memoria.

Si la nueva fuente posee mejor fitness que la existente en su memoria, la abeja empleada memorizará la posición de la nueva fuente y su contador de intentos volverá a cero. Es de suma importancia destacar que el considerar de mejor calidad el fitness nuevo ante el almacenado, se deberá a si este último es mayor que el fitness nuevo. Este último criterio se deduce ya que el problema a resolver es de minimización.

En caso contrario, la abeja empleada desechará la nueva fuente, manteniendo en su memoria la posición de la fuente anterior, y su contador de intentos se incrementará en uno.

Algoritmo 4.3 – Pseudocódigo de la Fase de abejas empleadas

```
Inicio
      Repetir
            Calcular número aleatorio para cambio de parámetro
            Calcular número aleatorio para cambio de fuente de alimento
            Repetir
                  Almacenar solución de apoyo
            Hasta (Iteración = Número de parámetros)
            Calcular nueva fuente de alimento
            Validar límites de los nuevos valores
            Calcular función objetivo
            Calcular fitness
            Si (Mejor Fitness > Actual Fitness) //Selección codiciosa
                  Intentos de la fuente cambiada = cero
                  Actualizar valores de función objetivo
                  Actualizar valores de fitness
            Sino
                  Incrementar los intentos en uno
      Hasta (Iteración = Número de Fuentes de Alimento)
Fin
```

4.7.4 Fase de abejas observadoras

En esta fase, las abejas observadoras reciben información de las fuentes de alimento por parte de las abejas empleadas y eligen una fuente para explotar dependiendo de una probabilidad asociada al fitness de dicha fuente. Es decir, una o más abejas espectadoras pueden escoger la misma fuente de alimento si esta posee un buen fitness. La probabilidad de selección de la fuente de alimento se calcula por medio de:

$$P_q = \frac{fitness_q}{\sum_{i=1}^{NF} fitness_i} \qquad (4.7.4.1)$$

Luego de asignadas las probabilidades para cada fuente de alimento, se calcula un número aleatorio [0,1].

Si este número es menor, la fuente de alimento en cuestión se somete al mismo proceso que en la fase de las abejas empleadas, cada abeja observadora encuentra una nueva fuente en el vecindario por medio de la ecuación (4.7.3.1) y aplica una selección codiciosa entre la nueva fuente y la original con el fin de memorizar la fuente con mejor fitness. El contador de intentos se incrementará en uno si selecciona la fuente original, o volverá a cero si se escoge la nueva fuente.

Si el número aleatorio es mayor a la probabilidad, la fuente de alimento no se selecciona y la abeja observadora pasa a verificar si probablemente elige o no la siguiente fuente de alimento. Al no ser elegida una fuente de alimento, el contador del ciclo iterativo de esta fase no incrementa. Con esto se asegura que esta fase termine cuando cada abeja observadora elija una fuente de alimento para evaluar su fitness.

Algoritmo 4.4 – Pseudocódigo de la Fase de abejas observadoras

```
Inicio
      Calcular Probabilidades
      Inicializar índice de probabilidades
      Repetir
            Calcular número aleatorio
            Si (Número aleatorio < Probabilidad de la fuente actual)
                  Incrementar Iteración
                  Calcular número aleatorio para cambio de parámetro
                  Calcular número aleatorio para cambio de fuente de a.
                  Repetir
                        Almacenar solución de apoyo
                  Hasta (Iteración = Número de parámetros)
                  Calcular nueva fuente de alimento
                  Validar límites de los nuevos valores
                  Calcular función objetivo
                  Calcular fitness
                  //A continuación selección codiciosa
                  Si (Nuevo Fitness > Fitness anterior)
                        Intentos de la fuente cambiada = cero
                        Actualizar valores de función objetivo
                        Actualizar valores de fitness
                  Sino
                        Incrementar los intentos en uno
            Sino
                  No aumentar iteración
            Aumentar índice de probabilidades
            Si (índice de probabilidades = número de fuentes de alimento)
                  Índice de probabilidades = cero
      Hasta (Iteración = Número de Fuentes de Alimento)
Fin
```

4.7.5 Fase de Abejas exploradoras

Esta última fase se encarga de verificar si el valor del contador de intentos para cada fuente de alimentos es mayor que el parámetro conocido como "Límite". En caso de ser así, la fuente de alimento es abandonada y la abeja empleada asociada a dicha fuente se convierte en una abeja exploradora, la cual produce aleatoriamente una nueva fuente de alimento en el espacio de búsqueda, utilizando la ecuación (4.8.2.1), y el contador de intentos volverá a cero. Luego se calcula el fitness asociado a la nueva fuente de alimento generada.

Algoritmo 4.5 – Pseudocódigo de la Fase de abejas exploradoras

```
Inicio
      Repetir
            Si (Intentos de la fuente de alimento > Parámetro de límite)
                  Repetir
                        Calcular número aleatorio
                        Calcular fuente de alimento
                  Hasta (Iteración = Numero de parámetros del problema)
                  Calcular función objetivo
                  Calcular fitness
                  Inicializar intentos en cero
      Hasta (Iteración = Número de Fuentes de Alimento)
Fin
```

4.7.6 Memorizar la mejor solución

Por cada iteración después de haber ejecutado las fases del algoritmo, se debe almacenar la mejor solución obtenida hasta el momento, por lo que al cumplir la condición de término, el algoritmo da como resultado la mejor solución encontrada.

Cabe mencionar que las fases de las abejas empleadas, espectadoras y exploradoras se repetirán mientras no se alcance la condición de término.

Algoritmo 4.6 – Pseudocódigo de Memorizar la mejor solución

```
Inicio
      Repetir
            Si (Valor de la función objetivo < Mínimo Global)
                  Mínimo Global = Valor de la función objetivo
                  Repetir
                        Parámetros Globales = Fuente de comida
                  Hasta (Iteración = número de parámetros del problema)
      Hasta (Iteración = Número de Fuentes de Alimento)
Fin
```

4.7.7 Función de reparación

El término función de reparación es muy importante para reparar una posible solución que posea una mejor calidad. La estrategia es realizar un movimiento dependiendo del semestre que este la violación. Aun así si al ejecutar el movimiento la solución empeora, inmediatamente no se considerará la solución con el nuevo movimiento.

Algoritmo 4.7 – Pseudocódigo de la Función de reparación

```
Inicio
      Repetir
            Si (Prerrequisitos o Cursos o Créditos inválidos)
                  Calcular número de violaciones de restricciones
                  Ejecutar movimiento de reparación
                  Si (violaciones aumentan)
                        Volver a la solución anterior
                  Sino
                        Actualizar solución con el nuevo movimiento
      Hasta (Existencia de Prerrequisitos o Cursos o Créditos inválidos)
Fin
```

5 Propuesta de Solución

Para resolver el problema BACP, se propone trabajar mediante un algoritmo ABC el modelo de programación lineal entera propuesto.

5.1 Esquema de Solución Propuesta

La solución propuesta en términos generales se encargará de recibir una instancia de prueba BACP propuesta por la CSPLib. Luego, el algoritmo ABC se encargará de procesar la instancia para generar soluciones. Finalmente, el Algoritmo ABC retornará la mejor solución encontrada.



Figura 5. 1 Esquema de la solución al BACP

El BACP contemplará todos los parámetros, variables de decisión y restricciones presentadas dentro del capítulo 4. Los periodos académicos y los cursos de la malla curricular formarán una vector que podrá tomar los valores descritos por la variable de decisión x_j.

Aparte de los parámetros del marco teórico, se deberán ingresar:

- Cantidad de iteraciones.
- Tamaño de la colmena (cantidad total de abejas).
- Valor "Límite" para los intentos en cada fuente de comida.

Así es como las fuentes de alimento (fs) se crearán aleatoriamente en la fase de inicialización.La cantidad de cursos será la dimensión que tendrá cada fuente de comida. El mínimo Global es la función objetivo a minimizar (c).

5.2 Estructura de Resolución

A continuación la Figura 5.2 presenta como el vector FC_i representa una fuente de alimento en el algoritmo ABC:

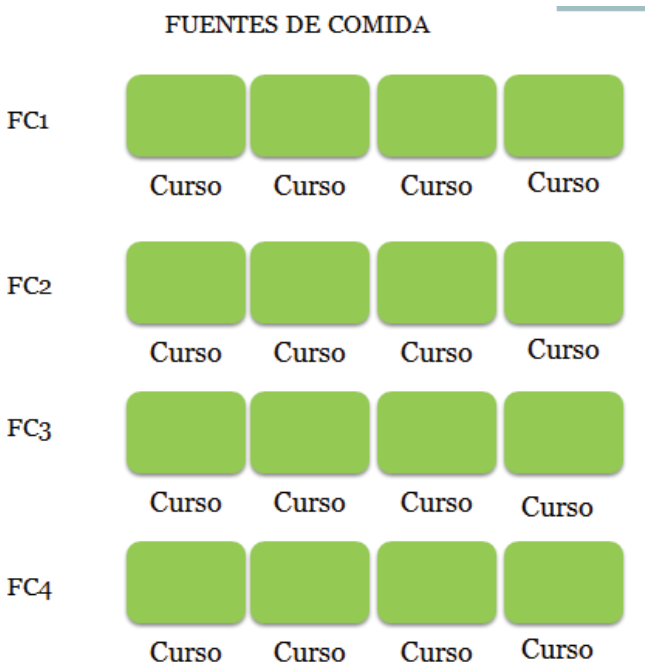


Figura 5. 2 Fuentes de alimento vectoriales

Cada fuente de alimento tendrá un vector FC_i cuya longitud es la cantidad de cursos del plan de estudios. El valor de cada casilla del vector representará el periodo académico que se le asignará a un curso i-ésimo.

También debe considerarse el vector para los intentos y las probabilidades, ambos con longitud igual a la cantidad de fuentes de alimento generadas.

5.3 Lenguaje de programación y entorno de implementación

El entorno para desarrollar la solución propuesta es en Netbeans IDE 7.3, en el lenguaje de programación orientado a objetos Java. Se escoge este lenguaje por sus características de independencia de plataforma, su alto rendimiento, fácil de aprender y basado en estándares.

El computador de pruebas es un Samsung con sistema operativo Windows 8 pro 64 bits, cuyo procesador es 2,5 GHZ Intel i3-3120M CPU y 6GB de RAM.

5.4 Resultados del algoritmo

En esta sección de presentan los resultados obtenidos por el prototipo funcional (Figura 1). El algoritmo se encarga de ejecutar las 3 instancias propuestas por la CSPLib y una instancia real. Se realizaron 10 ejecuciones por instancia:

5.4.1 BACP8

Esta instancia es la más pequeña de las tres propuestas por la CSPLib, por ende es la más rápida de resolver. La instancia BACP8 comprende de:

- 8 periodos académicos.
- 46 cursos.
- 10 créditos mínimos por periodo académico.
- 24 créditos máximos por periodo académico.
- 2 cursos mínimos por periodo académico.
- 10 cursos máximos por periodo académico.
- 37 relaciones de prerrequisitos.

Los parámetros considerados para esta instancia fueron los siguientes:

- Cantidad de iteraciones: 100.
- Valor límite: 10.
- Cantidad de abejas: 4.

La mejor ejecución corresponde a alcanza los 20 créditos como la máxima cantidad de créditos por periodo académico para dicha solución en un tiempo de 3 minutos. Los 20 créditos obtenidos no corresponden a los 17 de la solución óptima conocida de esta instancia. Si bien la ejecución fue de 100 iteraciones, el mejor fitness apareció a la iteración 30. Dicho comportamiento se repitió en varias ejecuciones (95%).

5.4.2 BACP10

La instancia BACP10 comprende de:

- 10 periodos académicos.
- 42 cursos.
- 10 créditos mínimos por periodo académico.
- 24 créditos máximos por periodo académico.
- 2 cursos mínimos por periodo académico.
- 10 cursos máximos por periodo académico.
- 34 relaciones de prerrequisitos.

Los parámetros considerados para esta instancia fueron los siguientes:

- Cantidad de iteraciones: 600.
- Valor límite: 10.
- Cantidad de abejas: 10.

La mejor ejecución corresponde a alcanza los 18 créditos como la máxima cantidad de créditos por periodo académico para dicha solución en un tiempo de 18 minutos. Dicha solución no es la óptima para esta instancia, 14 créditos. Si bien la ejecución fue de 600 iteraciones, el mejor fitness apareció en la iteración 150. Dicho comportamiento se repitió en la mayoría de ejecuciones (90%). Un 90% de las ejecuciones alcanzaron un valor de 20 créditos demorando en promedio 18 minutos.

5.4.3 BACP12

La instancia BACP12 comprende de:

- 12 periodos académicos.
- 66 cursos.
- 10 créditos mínimos por periodo académico.
- 24 créditos máximos por periodo académico.
- 2 cursos mínimos por periodo académico.
- 10 cursos máximos por periodo académico.
- 65 relaciones de prerrequisitos.

Los parámetros considerados para esta instancia fueron los siguientes:

- Cantidad de iteraciones: 600.
- Valor límite: 10.
- Cantidad de abejas: 10.

La mejor ejecución corresponde a alcanza los 20 créditos como la máxima cantidad de créditos por periodo académico para dicha solución en un tiempo de 45 minutos. El óptimo para esta instancia es de 18 créditos, por lo tanto el algoritmo no llego al óptimo conocido. Si bien la ejecución fue de 600 iteraciones, el mejor fitness apareció a la iteración 300. Dicho comportamiento se repitió en la mayoría de las ejecuciones (97%). Un 85% de las ejecuciones alcanzaron un valor de 24 créditos demorando en promedio 45 minutos.

5.4.4 BACP ICI PUCV malla antigua

El único caso real probado hasta el momento corresponde a la malla de antigua de la carrera de Ingeniería Civil Informática de la Escuela de Ingeniería Informática de la PUCV. Dicha configuración comprende de:

- 12 periodos académicos.
- 57 cursos.
- 10 créditos mínimos por periodo académico.
- 24 créditos máximos por periodo académico.
- 2 cursos mínimos por periodo académico.
- 10 cursos máximos por periodo académico.
- 90 relaciones de prerrequisitos.

Los parámetros considerados para esta instancia fueron los siguientes:

- Cantidad de iteraciones: 600.
- Valor límite: 10.
- Cantidad de abejas: 10.

La mejor ejecución corresponde a alcanza los 22 créditos como la máxima cantidad de créditos por periodo académico para dicha solución en un tiempo de 50 minutos. No se logró el óptimo conocido correspondiente a 18 créditos. Si bien la ejecución fue de 600 iteraciones, el mejor fitness apareció en la iteración 400. Dicho comportamiento se repitió en varias ejecuciones (95%). Un 95% de las ejecuciones alcanzaron un valor de 24 créditos demorando en promedio 50 minutos, lo cual es bastante tiempo.

Recapitulando los resultados expuestos, ninguna instancia logró llegar a un óptimo conocido. El comportamiento de la técnica aplicada no presenta buen rendimiento en encontrar la solución óptima. La cantidad de tiempo por ejecución es bastante.

6 Conclusiones

En conclusión, se espera que a través de esta investigación se haya logrado comprender la problemática expuesta, el problema de balanceo de mallas curriculares (BACP). A través del estado del arte, se intentó plasmar la situación actual en la que se encuentra el BACP, y a través del marco teórico, se trató de dar a conocer el contexto de las mallas curriculares en la enseñanza superior, la formulación del BACP, los conceptos sobre Optimización Combinacional y la Metaheurística BCO con el algoritmo ABC elegido para la solución del problema.

Para solucionar la problemática, se utilizó un modelo de programación lineal entera como base, que se consideró adecuado para captar la esencia del BACP. Una vez que se tuvo claro el problema a resolver se eligió la técnica que ayudó a resolverlo, esta fue la optimización basada en colonia de abejas.

Se presentó una propuesta de solución basada en el Algoritmo ABC detallando mediante pseudocódigo la estructura general del algoritmo con sus funciones y procedimientos.

Se implementó un algoritmo con el esquema de solución propuesto. El algoritmo no logró llegar a los óptimos conocidos de las instancias de prueba y la instancia real. Se generaban mallas curriculares válidas pero sin su carga académica balanceada por periodo.

Se recomienda a futuro ejecutar más pruebas para calibrar el modelo y obtener soluciones válidas en poco tiempo de ejecución para las instancias de prueba propuestas por la CSPLib e instancias reales. También, se insta a probar otro tipo de Metaheurística para resolver la problemática y obtener mejores resultados.

Finalmente, el esquema de solución propuesta y las directrices en las que se basa, están sujetas a cambios que considere el equipo de trabajo, con el objetivo de seguir mejorando el rendimiento del algoritmo para resolver la problemática expuesta en este documento.

7 Referencias

[1] Castro C. y Manzano S., "Variable and Value Ordering When Solving Balanced Academic Curriculum Problems", Proceedings ERCIM WG on constraints, pp. 3-4, Chile, Octubre 2, 2001.

[2] Hnich B., Kiziltan Z. y Walsh T., "Modelling a Balanced Academic Curriculum Problem", Proceedings of CP-AI-OR-2002, Vol. 2, pp. 121-131, 2002.

[3] Lambert T., Castro C., Monfroy E., Riff M. y Saubion F., "Hybridization of Genetic Algorithms and Constraint Propagation for the BACP", Logic Programming, Vol. 3668, pp. 421-423, Berlin, 2005.

[4] Monette J.N., Schaus P., Zampelli S., Deville Y. y Dupont P., "A cp approach to the balanced academic curriculum problem", Proceedings of the 7th International Workshop on Symmetry and Constraint Satisfaction Problems, 2007.

[5] Rubio J., Palma W., Rodriguez N., Soto R., Crawford B., Paredes F. y Cabrera G., "Solving the Balanced Academic Curriculum Problem using the ACO Metaheuristic", Hindawi Publishing Corporation, Mathematical Problems in Engineering ,Volume 2013, Article ID 793671, 8 pages.

[6] Chiarandini M., Di Gaspero L., Gualandi S. y Schaerf A., "The balanced academic curriculum problem revisited," Journal of Heuristics, vol. 18, no. 1, pp. 119–148, 2012.

[7] Canudas L., "El curriculum de estudios en la enseñanza superior", Revista de educación superior n°2, pp. 13-26, México, 1972.

[8] Larrosa J, Meseguer P., "Restricciones Blandas: Modelos y Algoritmos". Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 2003.

[9] Karaboga D., Akay B.,"A Survey: Algorithms Simulating Bee Swarm Intelligence; Artificial Intelligence Review"; 31 (1), pp. 68-85, 2009.

[10] Papadimitriou C. y Steiglitz K., "Chapter 19: Local Search", Combinatorial Optimization: algorithms and complexity, pp. 454-455, 1998.

[11] Luna F., "Metaheurísticas avanzadas para problemas reales en las redes de telecomunicaciones", Tesis doctoral Universidad de Málaga, pp. 19-24, España, 2008.

[12] Karaboga D., B. Basturk., "On the performance of artificial bee colony (ABC) algorithm", Applied Soft Computing,Volume 8, Issue 1, January 2008, Pages 687-697.

[13] Kaur A., Goyal S., "A survey on the Aplications of Bee Colony Optimization Techniques", International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 8 August 2011.

[14] Teodorovic, D.," Bee colony optimization (BCO)". In C. P. Lim, L. C. Jain, & S. Dehuri (Eds.), Innovations in swarm intelligence (pp. p 39–60). Berlin, Heidelberg: Springer-Verlag, 2009.

[15] Karaboga, D., "An idea based on honey bee swarm for numerical optimization" (Technical Report-Tr06), Erciyes University, Engineering Faculty Computer Engineering, Department Kayseri/Türkiye, 2005.