UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA QUÍMICA Y BIOTECNOLOGÍA

MODELACIÓN MATEMÁTICA DE LA ABSORCIÓN DE HIERRO EN CÉLULAS CACO-2 MEDIANTE ALGORITMOS DE PROGRAMACIÓN GENÉTICA Y FENOMENOLÓGICOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL QUÍMICO

RODRIGO ANDRÉS ROJAS KAPPES

PROFESOR GUÍA:
J. CRISTIAN SALGADO HERRERA

MIEMBROS DE LA COMISIÓN:
ZIOMARA GERDTZEN HAKIM
ÁLVARO OLIVERA NAPPA

Este trabajo ha sido parcialmente financiado por el proyecto FONDECYT Regular 1130317

SANTIAGO DE CHILE
2017

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL QUÍMICO
POR: RODRIGO ANDRÉS ROJAS KAPPES
FECHA: NOVIEMBRE 2017
PROF. GUÍA: J. CRISTIAN SALGADO HERRERA

MODELACIÓN MATEMÁTICA DE LA ABSORCIÓN DE HIERRO EN CÉLULAS CACO-2 MEDIANTE ALGORITMOS DE PROGRAMACIÓN GENÉTICA Y FENOMENOLÓGICOS

El hierro es un nutriente esencial para el ser humano, en donde su exceso o deficiencia puede provocar graves enfermedades como la anemia o hemocromatosis. Es por esto que es de suma importancia la regulación en su absorción, y si bien se han realizado numerosas investigaciones sobre el tema, no se sabe lo suficiente para describir totalmente el proceso, por lo que es de suma importancia seguir investigando para tener más información de este y así ayudar a prevenir o tratar enfermedades.

El objetivo de este trabajo es la modelación matemática de la absorción de hierro en células Caco-2 utilizando algoritmos de programación genética y algoritmos similares. Esto se basa en estudios preliminares experimentales y de modelación (utilizando algoritmos de programación genética) para la absorción de hierro, lo cual permite tener los datos experimentales iniciales para utilizarlos en nuevos modelos. Se expone un modelo empírico para modelar el proceso anterior en base a algoritmos de programación genética, y también 2 algoritmos de optimización para sistemas no lineales como son: Artificial Bee Colony Programming (ABCP) y Dynamic Ant Programming (DAP) los cuales se basan en técnicas de regresión simbólica para para desarrollar nuevos modelos empíricos y poder generar funciones a partir de datos experimentales.

Los algoritmos se debieron implementar completamente en un comienzo para luego utilizarlos en el problema de absorción de hierro, los cuales por sus características mostraron diferentes modelos para el mismo problema, pero en ambos casos, las curvas mostraron un comportamiento creciente sin oscilaciones ni cambios de pendiente. Lo anterior sirvió para comparar sus desempeños entregando como resultado que el algoritmo que mejor se adapta al sistema es ABCP con un $R^2 = 0.86$ y el que posee menor tiempo de ejecución es DAP con 1 hora.

Los algoritmos son capaces de representar los datos experimentales, los que sirven como una buena aproximación a la fenomenología. Aun así, no son capaces de captar todos los fenómenos del sistema, debido a que son modelos empíricos y no se les entrega información previa. Dado lo anterior, la metodología y los algoritmos utilizados pueden ser aplicados a otros problemas de investigación con el respaldo de que en este estudio entregaron buenos resultados.

AGRADECIMIENTOS

En primer lugar debo agradecer a mi familia, por estar siempre presente y apoyarme en todo momento, y además por siempre confiar en mí. A mi tía por ayudarme en la escritura de este trabajo.

A todos mis amigos que me acompañaron en este proceso, a los que comenzamos en plan común Nicolás, Felipe, Mario y Karim, que a pesar de encontrarnos en distintas especialidades, la hora de almuerzo era sagrada para comentar nuestros avances o planear el fin de semana.

A mis compañeros de IQBT, Eduardo y Samanta por ayudarme con las tareas, acompañarme a carretear o simplemente conversar cuando era necesario.

A los miembros del laboratorio PMDC, en especial a Nicolás que ya terminó su proceso, y que pasamos horas divagando con las “pausas activas”, y también a Ignacio por estar siempre presente, además del clásico puchito que los 3 compartimos. Al resto de los miembros por sus comentarios y críticas que ayudaron a seguir avanzando en el proyecto.

Al profesor Cristian Salgado por ser mi profe guía y tomarse las molestias de ayudarme en lo que necesitara, que gracias a sus constantes comentarios me ayudaron a elaborar de la mejor forma posible este trabajo.

Por ultimo agradecer el financiamiento otorgado por el proyecto FONDECYT Regular 1130317

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

NOMENCLATURA

a^*	Pseudo-parámetro
$c(k)$	Valor entregado por el modelo para el individuo k
C_0	Concentración inicial de hierro
$d(k)$	Valor experimental para el individuo k
f_{ib}	Mejor Fitness en DAP
fit_i	Fitness solución en ABCP
fit_{besst}	Fitness del mejor individuo en ABCP
$j_{i,k}$	Soluciones elegidas para generar una nueva en ABCP
k	Individuos de la población
L_i	Cantidad de nodos generados en el trayecto i DAP
m	Número de hormigas
M	Concentración [moles]
MSE_{jk}	Error cuadrático medio Jackknife
N_i	Cantidad de nodos en DAP
$P(i)$	Probabilidad de insertar un nodo en DAP
P_i	Probabilidad de que un nodo o terminal sea elegido en ABCP
p_{kij}	Probabilidad de que un nodo k sea elegido en el trayecto i, j en DAP
$r(k)$	Fitness del individuo k
R^2	Coeficiente de determinación
t	Tiempo [min]
v_i	Nueva solución generada en ABCP
X	Variable dependiente
Y	Variable independiente
α	Coeficiente de regresión
β	Coeficiente de regresión
ρ	Tasa de evaporación en DAP
τ_i	Cantidad de feromonas en el tramo i en DAP
τ_{maximo}	Parámetro máximo de feromonas en DAP
τ_{minimo}	Parámetro mínimo de feromonas en DAP

1. INTRODUCCIÓN

1.1. ANTECEDENTES GENERALES

El hierro es un nutriente muy importante para los seres humanos, ya que por su capacidad de aceptar o donar electrones, es capaz de participar en una gran variedad de procesos celulares, como son el transporte de oxígeno, producción de energía o participar del sitio activo enzimas [1]. Como se producen pérdidas de hierro normalmente debido a recambio de células (exfoliación) o sangramiento, éste debe ser ingerido por las personas regularmente para mantener su regulación, en donde las fuentes principales del metal traza, son alimentos con un gran contenido de éste, como las carnes, leche, huevos y también productos de origen vegetal (frutas o semillas) [2].

Como el ser humano no posee un mecanismo específico para la excreción de hierro [4], la cantidad presente en el organismo es regulada por la absorción del metal, la cual es llevada a cabo por las células epiteliales del duodeno llamadas enterocitos [4]. El proceso comienza con el ingreso del hierro desde el lumen del intestino hacia el interior del enterocito, por la acción de una proteína transportadora (DMT1) presente en el extremo apical. Dentro de la célula, el hierro puede realizar distintas acciones acciones, ser almacenado dentro de ferritina, ser llevado al torrente sanguíneo mediante la acción de otra proteína transportadora llamada ferroportina o ser utilizado para las propias funciones de la célula [2].

El proceso de absorción es extremadamente importante para la regulación del hierro, ya que la deficiencia o exceso de este elemento puede provocar enfermedades hereditarias como anemia, hemocromatosis y asterosclerosis y enfermedades neuronales como Alzheimer, Huntington, Parkinson, ataxia de Friedreich y Pica [3], por lo que para poder prevenir estas enfermedades, uno de los pasos iniciales es conocer el proceso de la absorción; que si bien, existen investigaciones al respecto [1-4], aun no se comprende a cabalidad todos los componentes que interactúan en él.

1.2. MOTIVACIÓN

A pesar de los esfuerzos del mundo científico durante los últimos años para comprender en detalle los procesos biológicos del ser humano, los cuales en su mayoría están enfocados a prevenir y curar enfermedades que afectan a las personas, aún existen varios procesos que no están del todo determinados, como es la absorción de hierro en el intestino. Si se conociera perfectamente el proceso, entonces se podrían generar curas y tratamientos para enfermedades comunes del déficit de hierro, como es la anemia, por lo que es importante utilizar herramientas como modelos matemáticos que permitan comprender de una mejor manera lo que ocurre en este fenómeno y así contribuir al bienestar de las personas.

1.3. DEFINICIÓN DEL PROYECTO

Por la relevancia que posee el fenómeno de absorción de hierro, se hace necesario el estudio de la cantidad de este que ingresa al organismo bajo diferentes escenarios. La construcción de un modelo matemático puede permitir describir la situación, y con la información obtenida, determinar actores y procesos relevantes del fenómeno.

Lo anterior puede permitir diseñar nuevos experimentos que puedan ayudar a describir de mejor forma el fenómeno, y proponer alternativas a los aspectos de la medicina que se puedan ver beneficiados con el estudio.

Al no conocerse información detallada de los componentes del sistema, el modelamiento se puede abordar de una forma empírica, es decir, utilizando solo los datos experimentales. Para esto, se puede utilizar la técnica de regresión simbólica utilizada con mucho éxito en la programación genética, la cual permite resolver problemas de alta complejidad como son los que contienen soluciones altamente no lineales.

2. OBJETIVOS

2.1. OBJETIVO GENERAL

- Estudiar la absorción de hierro en células Caco-2 mediante el análisis de datos experimentales y el modelamiento matemático utilizando regresión simbólica.

2.2. OBJETIVOS ESPECÍFICOS

- Implementación de algoritmo de regresión simbólica Artifial Bee Colony Programming en MATLAB.
- Implementación de algoritmo de regresión simbólica Dynamic Ant Programming en MATLAB.
- Validación de los algoritmos utilizando funciones de prueba.
- Construcción de modelos para datos experimentales de absorción de hierro utilizando ambos algoritmos.
- Validación y análisis de los modelos obtenidos considerando la naturaleza del problema.

3. MARCO TEÓRICO Y ANTECEDENTES

3.1. DESCRIPCIÓN DEL SISTEMA

3.1.1. EL HIERRO

El hierro es un micronutriente muy importante para el desarrollo normal de las funciones de una gran parte de los organismos vivientes. En los mamíferos, el hierro se puede encontrar unido a proteínas como centro catalítico de enzimas clave para el organismo, como lo son la hemoglobina, mioglobina, proteínas de regulación transcripcional, estabilización estructural y varios componentes de la cadena transportadora de electrones [5,6]. Lo anterior sitúa al hierro como un elemento de vital importancia para procesos de distribución de oxígeno a los tejidos, procesos celulares y control de expresión de genes [7].

La carencia de este metal en el organismo es uno de los problemas nutricionales más importantes de los seres humanos, en donde la Organización Mundial de la Salud (OMS), estima que cerca de 2 billones de personas padecen de anemia [8], la cual es la principal enfermedad producida por la carencia de hierro, y principalmente por un déficit de este metal en la alimentación, infecciones o trastornos hereditarios. Las consecuencias de esta enfermedad son: una deficiencia en el desarrollo físico y cognitivo, en el caso de presentarse en una temprana edad, y una disminución en la capacidad física para los adultos [8].

De igual forma, un exceso del hierro en el organismo produce daño en lípidos, DNA y proteínas, debido a la capacidad oxidante del hierro, por lo que una sobrecarga de este metal puede desencadenar enfermedades como: hemocromatosis, talasemia y anemia falsiforme [9].

En condiciones normales, un adulto pierde entre 0.5 y 2 mg de hierro por día [10]. Lo anterior se debe principalmente a: la exfoliación natural de las células epiteliales de la piel, cabello, tracto intestinal, entre otros. Dado lo anterior, es recomendable consumir la misma cantidad de hierro mencionada, para mantener la homeostasis del hierro. El hierro presente en el cuerpo varía entre 3 y 4 g [11], por lo que la cantidad que se debe ingerir es mínima en comparación a la cantidad presente en el organismo.

El hierro presente en la dieta se puede encontrar en 3 formas nutricionales: el hierro hemo proveniente de fuentes animales, el cual se encuentra usualmente unido a proteínas que contengan el grupo hemo, como hemoglobina y mioglobina; el hierro no-hemo, proveniente de fuentes vegetales como: leguminosas, frutas y verduras, donde se puede encontrar en forma de sales o quelatos [12], y la tercera forma es el hierro no-hemo unido a ferritina (FTN), el cual se encuentra principalmente en las legumbres. El primer tipo representa entre un 5 y 10% del porcentaje total del hierro en la dieta, pero

su biodisponibilidad es mayor al del tipo no hemo $(20-30\%)$. El grupo no-hemo alcanza un $90\text{-}95\%$ de la disponibilidad en la dieta diaria, pero su baja biodisponibilidad produce que solo entre un 1 y 10% sea absorbido por el organismo [13].

3.1.2. ABSORCIÓN INTESTINAL DE HIERRO

La absorción intestinal de hierro ocurre principalmente en el duodeno, que corresponde a la primera sección del intestino delgado, también ocurre en menor medida en la parte superior del yeyuno. El proceso comienza con el ingreso desde el lumen del intestino a través de la zona apical de los enterocitos hacia el interior de estos. El paso es mediado por la proteína co-transportadora DMT1 (Divalent Metal Transporter 1), la cual necesita un proton y un ion ferroso ($\boldsymbol{Fe^{2+}}$) para realizar su tarea [14]. Como la mayor parte del hierro en la dieta se encuentra en su forma férrica ($\boldsymbol{Fe^{3+}}$), este debe ser reducido para entrar a los enterocitos, lo cual es realizado por una reductasa férrica presente en la zona apical de los enterocitos (DyctB).

El proceso continúa cuando el hierro se encuentra dentro del enterocito, ahí la mayor parte es utilizada en la mitocondria para la síntesis de Hemo y centros $\boldsymbol{Fe-S}$, además de ser usado en el citoplasma para su incorporación en metaloproteínas, por último, el exceso de hierro es almacenado dentro de ferritina [15]. Además, dependiendo de las necesidades del organismo, parte del hierro puede pasar al torrente sanguíneo, mediado por ferroportina 1, la cual se apoya en la ferroxidasa hedaestina, la cual vuelve a transformar el hierro a su forma férrica. En la Figura 3.1 se muestra el proceso de absorción de hierro en los enterocitos.

Finalmente, el hierro transportado por el torrente sanguíneo se dirige, a los pulmones, riñones, cerebro y principalmente al hígado, donde es utilizado para funciones propias de éste. También puede ser almacenado por procesos que amortiguan el suministro de hierro a otros tejidos. El mayor flujo ocurre hacia la médula ósea, donde es utilizado principalmente para sintetizar hemoglobina. Cuando los glóbulos rojos mueren, el hierro es recuperado por macrófagos del bazo y células del hígado pudiendo reutilizarse en la síntesis de hemoglobina [15].

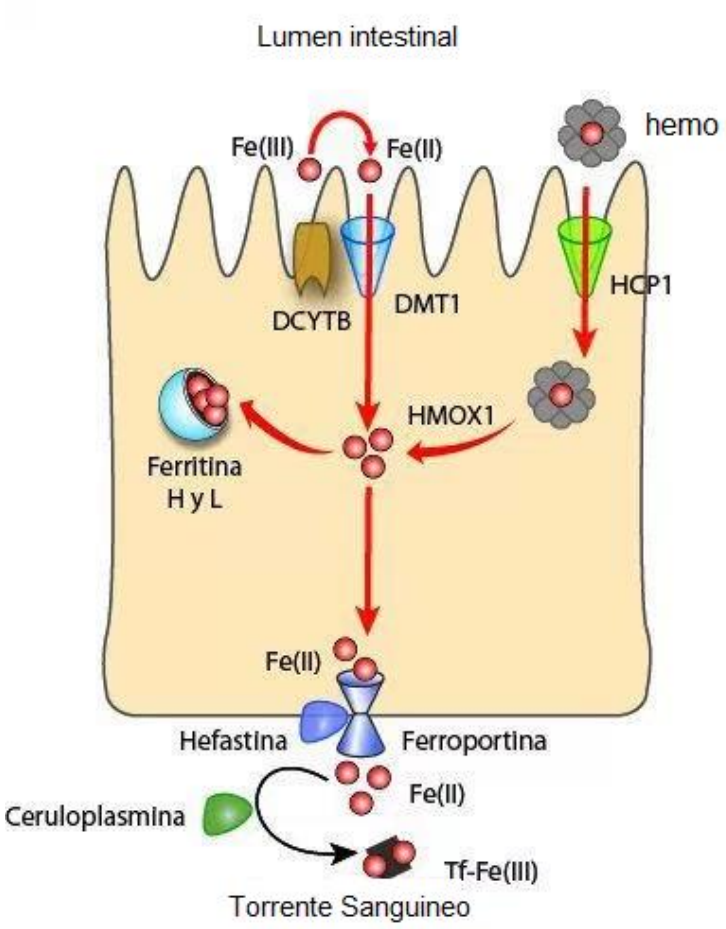

Figura 3.1: Transporte de hierro a través de los enterocitos [16].

Existen varios procesos importantes que pueden ocurrir durante la absorción de hierro como mecanismos de regulación, uno de ellos corresponde al bloqueo mucosal, el cual es un proceso putativo que describiría la habilidad de una dosis de hierro ingerida inicialmente para bloquear una segunda dosis [17]. Este fenómeno disminuiría hasta en un 30% la capacidad de absorción de hierro comparada con la primera dosis, además tendría un tiempo de respuesta de 45 a 90 minutos, haciéndolo un mecanismo de regulación bastante rápido. Además su efecto perduraría por más de 3 horas en células epiteliales [18].

Lo anterior es ratificado por ensayos *in vivo* realizado en mamíferos como ratas y perros [19], los que muestran la existencia un método de regulación, pero que no se distingue si corresponden al bloqueo mucosal u otro como regulación sistémica [20], traduccional [21] o transcripcional [22].

3.2. MODELOS MATEMÁTICOS

Como ya existen estudios experimentales para el problema de absorción de hierro, se plantea la utilización del modelamiento matemático con el fin de poder describir el sistema. Un modelo matemático es definido por la literatura de forma general como “un constructo matemático abstracto y simplificado, que relaciona a una parte de la realidad (sistema) con un objetivo definido. Dado lo anterior, el objetivo de los modelos es poder entender o resolver preguntas en base al sistema que se quiere describir, el cual es representado por planteamientos matemáticos sea fórmulas, ecuaciones, relaciones, etc. [23]. Generalmente, para dar un acercamiento correcto al sistema que se quiere describir, es necesario utilizar datos experimentales pertinentes y también se hace necesario establecer hipótesis y supuestos basados en el conocimiento a priori que se tenga del sistema que se quiere describir.

Como se mencionó anteriormente, los modelos se construyen en base a información previa que se tenga del sistema, por lo cual estos se pueden dividir en dos tipos según la cantidad de información que se posea. Primero están los modelos empíricos, que se construyen solo considerando los datos experimentales y obvian todo lo que sucede dentro del sistema, y por otro lado están los modelos fenomenológicos, los cuales utilizan información previa para determinar todas o una parte de las formas matemáticas del sistema [24].

La construcción de un modelo va a depender del objetivo particular que se tenga, ya que los tipos presentados tienen enfoques diferentes, por ejemplo, los modelos empíricos en la mayoría de los casos requiere menor tiempo y recursos para su construcción, en cambio los modelos fenomenológicos en general, realizan mejores predicciones con un rango mayor de validez, a costa del tiempo que se requiere para esta tarea.

Muchas veces se utiliza una combinación de ambos modelos, llamados semi-empíricos donde se utiliza información previa, pero componentes del sistema no se conocen. Así, la utilización de un modelo u otro va a depender del objetivo que se tenga, por ejemplo, una simple relación de datos experimentales o una descripción detallada de un proceso, la información existente, y recursos disponibles.

3.2.1. MODELOS EMPIRICOS

Los modelos empíricos comienzan con las observaciones experimentales, puesto que en base a estas se construye un modelo. La mayoría de este tipo de modelos está basado en regresiones, ya que esta técnica entrega una descripción matemática de la relación existente entre las variables de entrada y salida. Por lo anterior, estos modelos son aptos tanto para la predicción de nuevos datos, en caso de que el escenario de predicción sea similar al original donde se construyó el modelo, como para interpolaciones [25].

En una regresión, la variable dependiente es expresada en términos independientes usando diferentes tipos de ecuaciones. La más común es la regresión lineal, en la cual la variable dependiente Y e independiente X se relacionan mediante la ecuación 3.1, donde los parámetros del modelo corresponden a los coeficientes α y β los cuales son ajustados de manera que la simulación del modelo y los datos experimentales sea lo más similar posible. En la mayoría de los casos estos parámetros se determinan minimizando el residuo de la suma de los cuadrados (RQS), entre los datos experimentales $\hat{Y}(X)$ y la predicción realizada por el modelo $Y(X)$, lo que se muestra en la ecuación 3.2.

$$Y(X) = \alpha X + \beta \tag{3.1}$$

$$\min_{\alpha,\beta} RQS = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 \tag{3.2}$$

La regresión lineal es el modelo más sencillo que se puede desarrollar con esta técnica, pero dependiendo del problema, ésta se puede extender a una regresión lineal múltiple o regresiones no-lineales. En cualquiera de los casos anteriores, es necesario conocer explícitamente la función y la elección de ésta (o del modelo) va a depender de las características mostradas por los datos experimentales. A pesar de lo anterior, existen casos donde la funcionalidad de los fenómenos es compleja, por lo cual no es posible establecer una función de regresión [25]; y en estos casos se puede utilizar otras técnicas que no requieran conocer la función de antemano, como es la regresión simbólica.

La regresión simbólica [26] es una técnica utilizada cuando se obtienen datos experimentales de un proceso desconocido, la cual intenta formular una ecuación matemática, es decir, una combinación de variables, símbolos y constantes que se ajuste a los datos experimentales que se tengan.

Uno de los primeros algoritmos desarrollado con esta técnica fue la programación genética (GP) [33], el cual es el más popular usado en este tipo de regresión [27] ya que entrega buenos resultados para procesos desconocidos y ha sido usado con éxito en una gran cantidad de problemas reales. Un ejemplo de esto es el estudio anterior para la absorción de hierro, el cual utilizó este algoritmo para encontrar un modelo que se ajuste a los datos experimentales [48]. A pesar de la popularidad de este algoritmo, en los últimos años se han desarrollado varias alternativas que provienen de algoritmos planteados para resolver problemas de optimización, con la intención de obtener mejores desempeños que los entregados por GP utilizando la regresión simbólica [27], en esta categoría se encuentran algoritmos como: Particle Swarm optimization (PSO) [28], Differential Evolution (DE) [29], Artificial Bee Colony (ABC) [27], Ant Colony Optimization (ACO) [30], Probabilistic Incremental Program Evolution(PIPE) y también modificaciones realizadas a GP, ya sea sobre el mismo algoritmo o con la combinación de algún otro con el mismo fin anterior[31,32].

Como se mencionó anteriormente, la regresión simbólica es un método de optimización y se diferencia principalmente de los métodos tradicionales por 3 puntos: Primero es la independencia del gradiente, que al no utilizar esta información, pueden abordar problemas más complejos como son los de naturaleza discreta. Segundo se encuentra el paralelismo, lo cual es, dada la generación de nuevos individuos en cada generación, estos pueden ser tratados como secciones independientes, si se plantean ciertas condiciones, lo que permite paralelizar el problema. Por último está la capacidad exploratoria, gracias a los operadores de los algoritmos (los cuales presentan una naturaleza estocástica), permiten ampliar el dominio de búsqueda y que no se entrampen en mínimos locales [34].

Otra de las características de este tipo de algoritmos es la representación, la cual codifica los individuos como estructuras de árboles, donde en sus nodos hojas se van a encontrar las variables y constantes, que son llamadas “terminales”, y en los nodos internos se encuentran los operadores y funciones permitidas, las cuales son llamadas “funciones”. Es importante destacar que estos nodos van a presentar una cantidad de hijos igual al número de argumentos que requiera el operador o función que representan. Un ejemplo de este tipo de estructura es mostrado en la Figura 3.2.

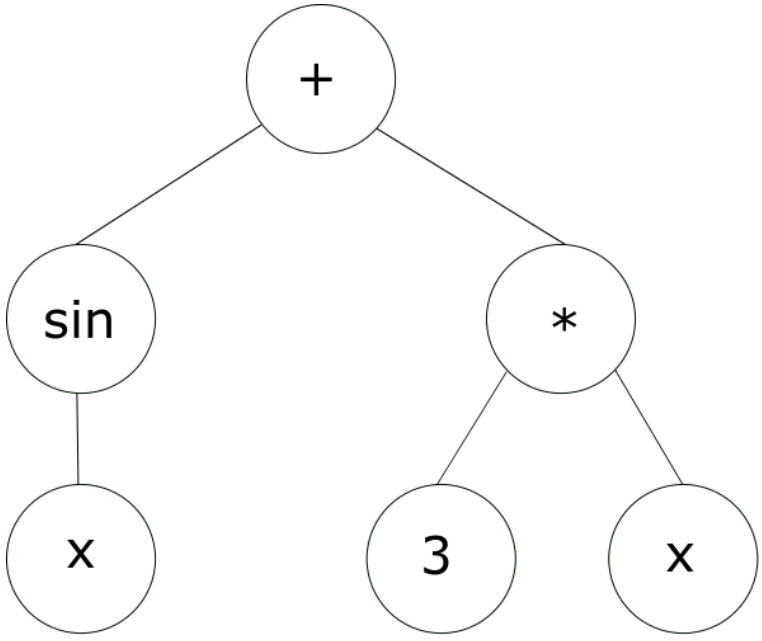

Figura 3.2: Representación de la función sin(x)+3*x en estructura de árbol [34].

Otro de los aspectos comunes en estos algoritmos es el conjunto de terminales y funciones. Los primeros son representados por los nodos hojas del árbol y los tipos son 3: Las variables, que corresponden a las entradas externas a los programas, las constantes, que pueden ser fijadas previamente o creadas de forma aleatoria según las operaciones de los algoritmos y las funciones sin argumentos (o de aridad cero) las cuales no pueden tener nodos hijos, como es la función $rand()$, que entrega un número aleatorio.

También existe el conjunto de funciones, las cuales van a depender de la naturaleza que tenga el problema, así para el tipo de problemas numéricos pueden existir las funciones aritméticas básicas $(+, -, *, /)$, o para el tipo de problemas lógicos se pueden especificar las funciones booleanas (AND, OR, XOR, NOT). Independiente del tipo de problema que se quiera describir, estas funciones deben satisfacer ciertas propiedades [34]:

1. Clausura: Indica que todas las funciones deben recibir argumentos de un tipo y retornar variables del mismo, es decir, las operaciones aritméticas pueden ser definidas para números reales y deben entregar estos números, en cambio sí se define esto para funciones lógicas esto no se cumple. Además, esto permite que ante las evaluaciones de las funciones generadas, estas no generen problemas que puedan hacer fallar los algoritmos, donde es típico que existan funciones que tengan cierta indeterminación en alguno de sus puntos, como el caso más común de la división por cero. Ante esto, en los algoritmos se definen "funciones protegidas", donde está la función original y algún criterio de retorno ante estos puntos en conflicto, por ejemplo, en el caso anterior la función protegida podría entregar una constante cuando ocurra el caso de indeterminación.

2. Suficiencia: Se cumple cuando el conjunto de funciones es suficiente en sí, para que ante todas las posibles combinaciones recursivas del conjunto original contenga al menos una solución del problema. Esta propiedad es difícil de comprobar ya que no es posible determinar a priori cuál conjunto entregará una solución, por lo que la teoría, experiencia u otro método indica el cumplimiento de esta propiedad.

Otros criterios comunes a los algoritmos de este tipo es que deben tener una función de fitness, la cual evalúa numéricamente lo buena que es la solución propuesta para el problema planteado, esta va a depender del tipo de problema planteado y puede ser diferente para el tipo de algoritmo que se quiera utilizar. También se encuentran los criterios de término, que al igual que el caso anterior va a depender del algoritmo que se use, pero en general, tienen que ver con la convergencia de la solución, máximo número de generaciones y una solución que sea aceptable para cierto criterio. Por último se encuentran los parámetros propios de los algoritmos, característica que le da robustez a los algoritmos ya que estos pueden entregar buenos resultados utilizando diferentes conjuntos de parámetros, por lo que se deben tener en consideración pero no se deben asignar muchos recursos a la hora de buscar los parámetros óptimos, ya que al realizar se estaría entregando información previa al modelo, lo cual es una característica de los modelos fenomenológicos, no empíricos [34].

3.2.2. PROGRAMACIÓN GENÉTICA

Ya que el presente estudio se basa en la comparación de resultados de distintos algoritmos, se toma como base lo realizado con programación genética en una primera aproximación [48]. Este algoritmo es parte de los denominados algoritmos evolutivos,

los cuales se basan en la teoría de evolución de Darwin y tienen por objetivo resolver problemas de optimización donde los métodos clásicos no presentan soluciones adecuadas [35].

La teoría de evolución de Darwin plantea que ya sea por mutación, selección o recombinación genética, si nace un individuo que se adapta mejor al ambiente que la media de la población a la que pertenece, éste tendrá mayores posibilidades de sobrevivir, y si nace con características que lo sitúen bajo la media, tendrá menores posibilidades de sobrevivir. En el caso de la programación genética, los individuos de la población serán las soluciones y el ambiente que se sitúa será la función objetivo, por lo tanto, el algoritmo genera primero una población inicial que luego mediante criterios parecidos a los planteados por Darwin, se seleccionan ciertos individuos para crear una nueva población que se ajuste mejor a la función objetivo.

La programación genética nace como una mejora a los algoritmos genéticos (GA), donde su principal característica es resolver los problemas mediante una inducción de programas y algoritmos que puedan solucionarlos. Estos siguen las mismas bases biológicas y funcionamiento que los GA, pero difieren en la forma de la decodificación del problema, donde la representación de los individuos es mediante árboles, lo cual permite resolver problemas que los GA no pueden [36].

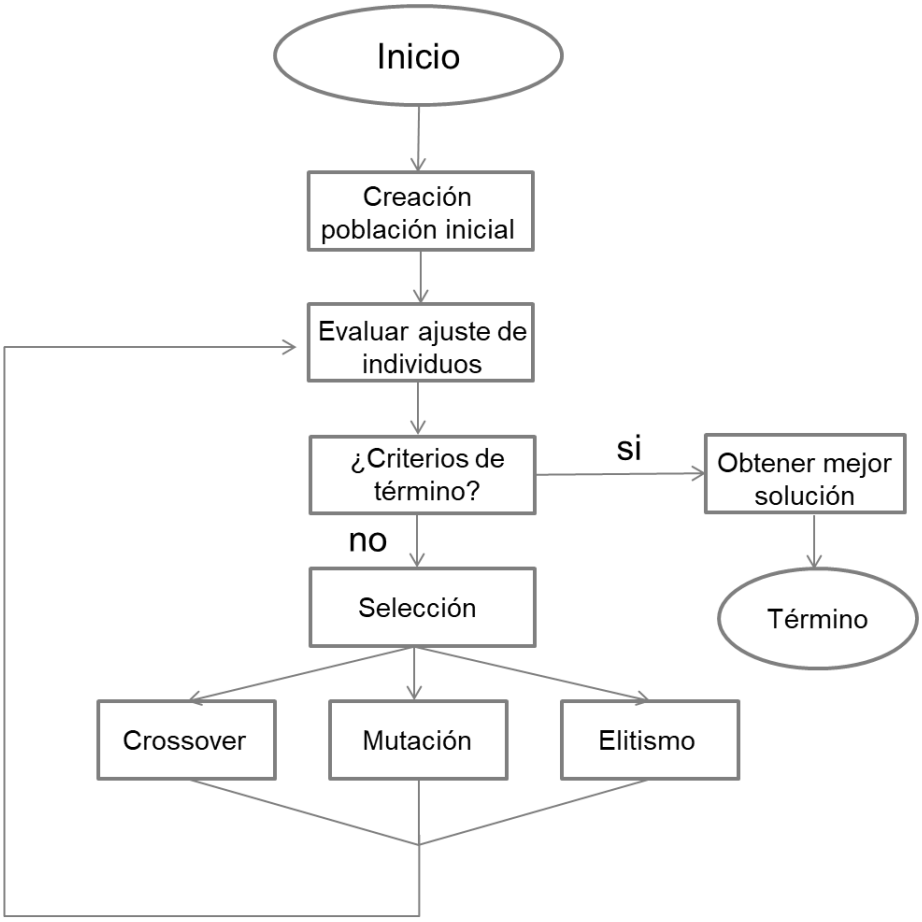

Figura 3.3: Esquema de trabajo de Programación Genética. Adaptado de [37].

En la Figura 3.3 se muestran los principales pasos de los algoritmos genéticos, donde se observa que se comienza creando una población inicial de posibles soluciones y luego se evalúan los individuos de la población, si se alcanza alguno de los criterios de término, se finaliza la iteración y se entrega la mejor solución evaluada. De no satisfacer la condición anterior, se crea una nueva población que se evalúa y el proceso continúa hasta alcanzar alguna condición de término.

Para indicar que una solución es aceptable [38], el fitness del individuo se debe encontrar en el intervalo $[0,1]$, ya que valores menores o mayores a esto indican soluciones complejas o divergentes. La solución va a converger si el 95% de la población presenta la misma solución y el número de generaciones [34] va entre 50 y 100, ya que si bien en las primeras generaciones se presentan los mayores cambios en la población, el aumentar el número ayuda a refinar la solución.

Las principales características de la programación genética se describen a continuación, y es lo que lo distingue de otros algoritmos que utilicen la regresión simbólica como tipo de regresión:

a) Creación de la población inicial [34]: Para la generación de la población inicial existen 3 métodos clásicos, grow, full y ramped half-half. En todos los casos se generan árboles aleatoriamente utilizando elementos de los conjuntos de funciones (F) y el conjunto de terminales (T). Es importante ver que los árboles generados por estos métodos presentan componentes estocásticos que deben cumplir parámetros establecidos anteriormente, como es el número máximo de individuos por generación o máxima profundidad, esta última definida como el número de aristas que se necesitan para conectar un nodo con la raíz (que tiene profundidad 0) y por lo tanto la máxima profundidad es la profundidad del nodo que se encuentre más alejado a la raíz.

   El método grow consiste en elegir un elemento del conjunto F en la raíz y luego elegir aleatoriamente un elemento del conjunto T o F, en que si se elige un elemento de F, se repite el proceso, y si es de T, este termina. En el método full, se eligen elementos del conjunto F hasta que el nodo se encuentre en la máxima profundidad, caso en el que se elige un elemento del conjunto T. Para este último, se dice que el árbol se encuentra balanceado ya que todas las hojas tienen la máxima profundidad establecida. Ramped Half-Half es un método en que el 50% de la población se construye con el método grow y el otro 50% con el método full. Un ejemplo de ambos métodos se encuentra en la Figura 3.4.

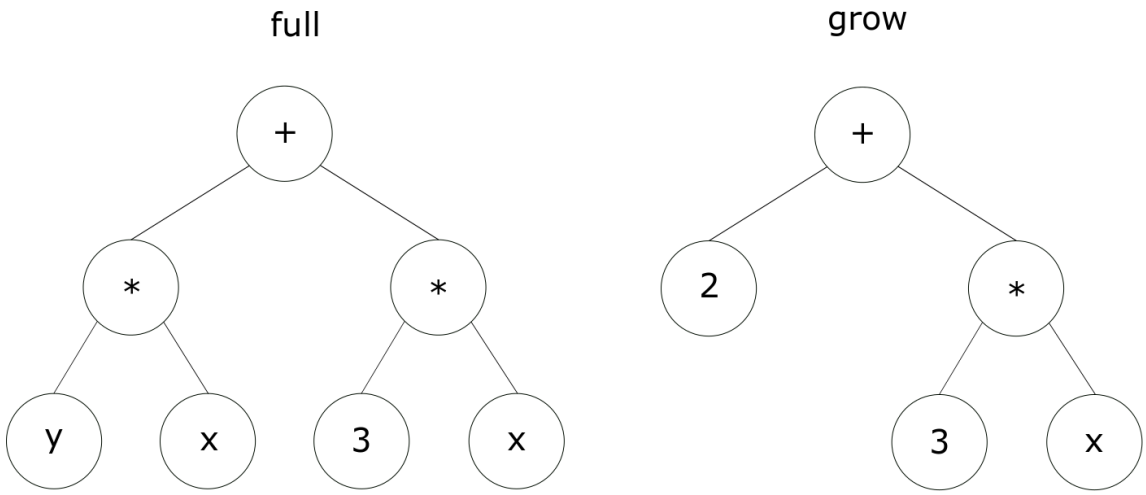

Figura 3.4: Métodos de construcción de árbol, a la izquierda el método full y a la derecha grow.

b) Selección [38]: Para crear la siguiente generación de individuos, en GP se deben seleccionar individuos de la generación actual probabilísticamente en base a sus características deseadas, donde los seleccionados son llamados padres. Esto es igual a la teoría de la evolución donde los individuos que tienen una mayor capacidad de adaptación son los que tienen mayor probabilidad de generar descendencia. Para seleccionar a los padres, este algoritmo usualmente tiene 2 métodos: Ruleta es el más común, y consiste en asignar a cada uno de los individuos una sección en la ruleta (la cual es proporcional a su desempeño) la que entre todos debe sumar 1. Para seleccionar el individuo se escoge un número

aleatorio entre $[0,1]$ y retorna el individuo que se encuentra en esa posición. El otro es Torneo, en el cual de toda la población, se elige a k cantidad de individuos aleatoriamente que participan en un torneo, donde se evalúan a los participantes y se elige al que tenga el mayor desempeño.

c) Operadores [34]: Luego de seleccionar a los padres, se generan nuevos individuos mediante la aplicación de los operadores genéticos sobre los padres según la probabilidad asignada a cada operador. Los más comunes son:

- Cruce o crossing over: Consiste en generar 2 individuos a partir de la combinación de 2 padres. Para la creación de estos individuos, se toma primero un nodo de cada padre, los cuales llevan el nombre de *punto de cross over.* Luego reemplaza el sub-árbol que tiene como raíz el punto de crossover del primer padre, por el sub-arbol que tiene como raíz el punto de crossover del segundo padre. Este operador es el más utilizado en GP y su probabilidad se elige entre 0.9 y 0.95. Un ejemplo de este operador se muestra en la Figura 3.5.

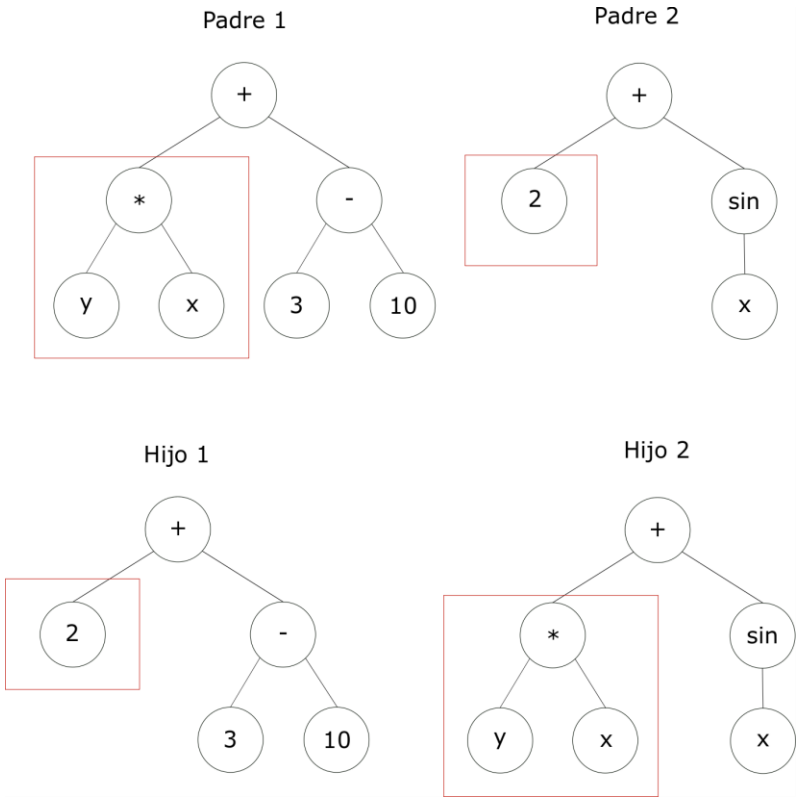

Figura 3.5: Ejemplo de Crossover

- Mutación: Consiste en escoger un nodo como punto de mutación, y reemplazar el sub-árbol que tiene como raíz este punto por un árbol aleatorio completamente nuevo. Este tipo de operación crea saltos en el espacio de búsqueda lo que produce un aumento de la variabilidad de la población, por lo anterior su

probabilidad es mucho menor que la del cruce, fijándose típicamente entre [0.1, 0.01]. Un ejemplo de esto se muestra en la Figura 3.6.

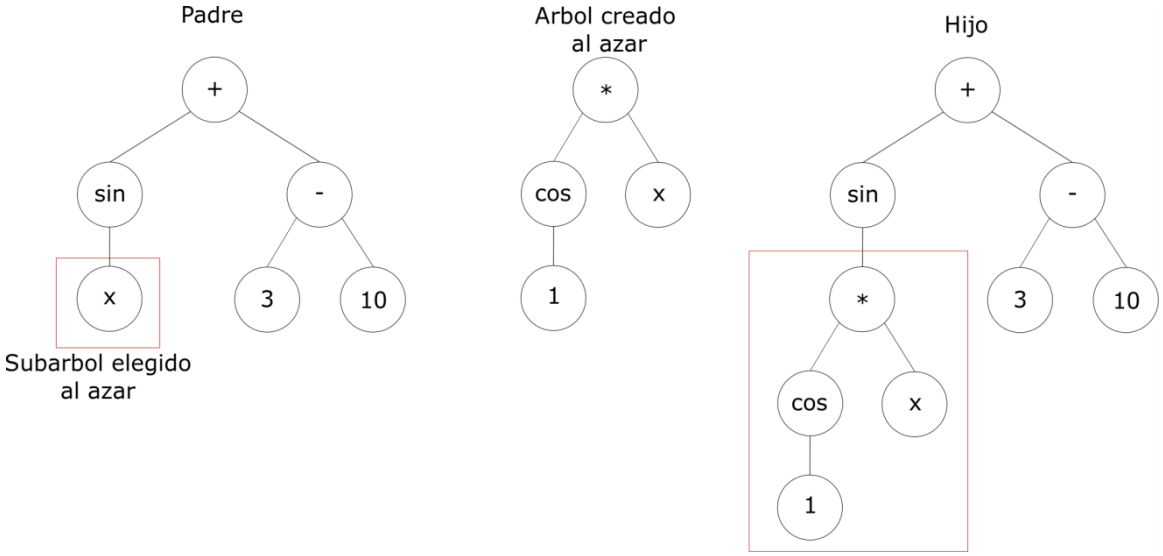

Figura 3.6: Ejemplo de Mutación.

- Elitismo: Es un operador que no siempre se incluye, por lo que es complementario a los anteriores. Consiste en elegir al individuo con el mejor desempeño de la generación, y se crea una copia de este el cual es insertado en la siguiente generación, lo que permite asegurar que la calidad aumenta a medida que transcurren las generaciones, ya que esta será mayor o igual a la generación anterior. Es importante destacar que este operador debe ser utilizado con una probabilidad muy baja, debido a que si se generan muchas copias, la población perderá variabilidad.

d) Componentes específicos para GP [33]: De los componentes comunes mostrados para algoritmos que utilizan regresión simbólica, a continuación se detallan estos componentes para el caso de la programación genética:

- Función de fitness: Como se mencionó anteriormente, esta función de adaptación evalúa qué tan buena es la solución para el problema planteado. GP utiliza la función de fitness puro $r(k)$ el cual indica la diferencia entre el valor deseado $d(k)$ del individuo k y su valor obtenido $c(k)$, como se muestra en la ecuación 3.3.

$$r(k) = \sum_{k=1}^{n} |d(k) - c(k)| \tag{3.3}$$

- Tamaño de la población: Este es el parámetro más importante del conjunto e indica la cantidad de individuos (o soluciones) que tendrá la población, por lo tanto es preferible que sea una cantidad razonable para que la variabilidad genética de la generación sea apta, entonces el número se encuentra entre 100 a 500 individuos pudiendo aumentarse según la complejidad del problema.

- Número de Generaciones [34]: Si bien esto puede encontrarse en la categoría de criterio de término, es un componente específico para el algoritmo y como se mencionó anteriormente, en las primeras generaciones es cuando se generan los mayores cambios en los individuos, por lo tanto se recomienda que sea entre 10 y 50, pudiendo aumentarse según el nivel de refinamiento que requiera la solución, en todo caso, no se recomienda que sean mayores a 100, para evitar el problema de bloating, o crecimiento desmedido del árbol, en que este crece tanto en cantidad de nodos como en profundidad a costa de una leve mejora en la calidad de la solución.

- Profundidad del árbol: Este valor va a depender de la complejidad del problema que se esté enfrentando, como también de los terminales y funciones que se utilicen, ya que como se mencionó anteriormente, una profundidad mayor puede entregar (para ciertos problemas) una mejor solución. También es proporcional al tamaño de la población, ya que si existen una gran cantidad de individuos y la profundidad es baja, es probable que existan muchos individuos similares o iguales lo que genera una baja variabilidad de la población. Ante todos estos factores, este valor se limita entre 8 y 50 [34].

El Algoritmo 1 muestra un pseudocódigo del algoritmo general de GP, en él se define primero una población inicial, la cual debe estar restringida por el número de individuos de la población y la profundidad máxima que deben tener los árboles generados, y debe ser generada por alguno de los métodos mencionados anteriormente. Luego se debe evaluar a todos los individuos con la función de fitness deseada para comenzar el proceso iterativo sobre las generaciones que se crearán hasta que se cumpla alguno de los criterios de término.

Algorithm 1. Algoritmo Programación Genética [34]

```
1: Generate initial populations with random candidate solutions
2: Evaluate each candidate
3: REPEAT; DO
4:     Select parents
5:     Recombine/Mutate pair of parents
```

6: Select individuals for next generations
7: Evaluate new candidates
8: Choose the best solution so far
9: **UNTIL** (termination condition is satisfied)

En el ciclo se van seleccionando los padres, para luego utilizar los operadores de cruce o mutación para ir creando los individuos de la generación siguiente, recordando que su tamaño debe ser igual al de la población inicial. Alcanzado el tamaño máximo de la población, se evalúan a los nuevos individuos y se escoge el mejor, si este individuo cumple alguna de las condiciones de término, termina el programa y retorna la mejor solución obtenida, de lo contrario, se continúa con el ciclo principal para crear una nueva generación.

3.2.3. ARTIFICIAL BEE COLONY

El segundo algoritmo utilizado para la comparación en este estudio es Artificial Bee Colony Programming [27], el cual es una adaptación de Artificial Bee Colony que utiliza la regresión simbólica. El algoritmo original pertenece al grupo de los algoritmos de optimización basados en inteligencia de enjambre, siendo este el más popular de la categoría ya que ha sido aplicado en numerosos campos para resolver diferentes tipos de problemas [51].

Este algoritmo es basado en el comportamiento social que existe alrededor de una colmena de abejas a la hora de la selección de las fuentes de alimento cercanas. En ella, las abejas trabajadoras son las encargadas de recolectar el alimento y cuando encuentran una fuente realizan unos particulares movimientos en la colonia denominado *danza o baile,* el cual dependiendo del movimiento son capaces de indicar la distancia, dirección y la calidad del alimento que encontraron [27]. Lo anterior es representado en la Figura 3.7.

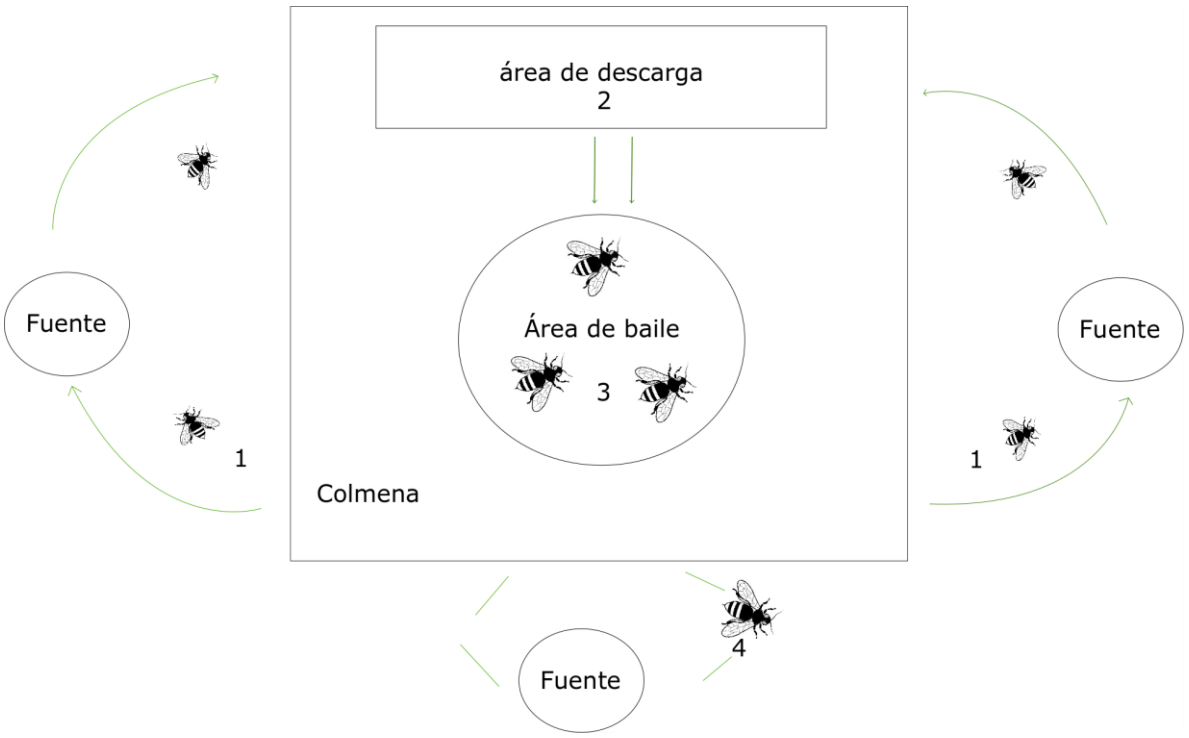

Figura 3.7. Comportamiento colonia de abejas. Desde que se dirigen a la fuente (1) hasta que abejas exploradoras descubren nuevas (4).

El algoritmo original toma el principio anterior y define una colonia artificial de abejas que contiene tres grupos: las abejas trabajadoras, las exploradoras y las observadoras. La abeja que se encuentra esperando en el área de baile, para tomar la decisión de qué fuente de alimento elegir es la observadora; la que se dirige a la fuente de alimento a verla por sí misma es la abeja trabajadora y la que busca aleatoriamente nuevas fuentes de alimento es la exploradora. Es importante destacar que en esta colonia la mitad consiste en abejas trabajadoras y la otra mitad en abejas observadoras. Además por cada fuente de alimento va a existir solo una abeja trabajadora, es decir, el número de abejas trabajadoras es igual al número de fuentes de alimentos alrededor de la colmena [27].

En este algoritmo, y similar a GP, la población consiste en un conjunto de posibles soluciones, representadas por la posición de la fuente de alimento, en que la cantidad de néctar representa la calidad de ésta. El algoritmo representa la interacción entre las abejas obreras y observadoras donde las primeras deben bailar para que las segundas elijan según las que realicen el mejor baile, cuál de todas las fuentes es la que contiene la mayor cantidad de néctar y por lo tanto la mejor solución.

Artificial Bee Colony Programming es una extensión del algoritmo Artificial Bee Colony para la regresión simbólica [27], la cual es una relación similar a la que tienen los algoritmos genéticos con GP, ya que esta adaptación utiliza estructuras más complejas para representar el problema. En ABCP, la posición de la fuente de alimento corresponde a programas computacionales generados aleatoriamente representados

por árboles (igual que GP). De igual forma, los terminales y funciones van a depender del tipo de problema que se esté enfrentando. Además, la calidad de cada fuente de alimento es determinado evaluando el desempeño de cada programa generado, que nuevamente como GP, es determinado por la función de fitness.

En la Figura 3.9 se encuentran los principales pasos de ABCP, los que luego de la generación inicial de la colonia son: creación y evaluación de nuevas funciones por las abejas trabajadoras, lo que realizan usando el mecanismo de intercambio de información, después una selección de funciones dependiendo de su calidad para seguir con la creación y evaluación de nuevas funciones por parte de las observadoras utilizando el mismo mecanismo anterior. Como existen 2 etapas de creación y evaluación de funciones, la primera se denomina fase de las trabajadoras, y la segunda es la fase de las observadoras. Luego de estas etapas, se encuentra la fase de las exploradoras, las cuales comprueban si existe alguna solución que no haya sido reemplazada en muchas iteraciones, según el valor límite de estas (si existe un individuo así, las exploradoras reemplazan esta por una nueva). El proceso es iterativo hasta que se cumple alguna de las condiciones de término.

En este algoritmo se definen 2 condiciones de término: Primero, cuando el valor de la función de fitness (mostrada en la ecuación 3.3) para algún individuo es menor a 0.01 y la segunda es el número de generaciones, la que puede ser entre $[10,500]$, ya que la convergencia está fuertemente determinada por la función objetivo que se tenga.

Las principales características que tienen en común ABCP con GP, y las propias que lo diferencian con otros algoritmos que utilicen la regresión son:

a) Comunes con GP: ABCP y GP comparten muchas características [27], ya que ambos se basan en soluciones iniciales que pueden ir cambiando para mejorarlas. La razón radica en que ABCP nace como alternativa a GP, siendo comparado sus desempeños en [27]. Por lo anterior, el método de generación de población es *ramped half-half*, la función de fitness que utiliza es la presentada en la ecuación 3.3, y el resto de los parámetros como tamaño de la población, número de generaciones y profundidad del árbol presentan las mismas características y limitaciones presentadas en GP, por lo que se utilizan los valores estándar para esto. Cabe destacar que lo anterior es porque ABCP es un método alternativo de regresión presentado recientemente, por lo que no existen estudios que muestren resultados para problemas reales que indiquen una relevancia significativa para estos parámetros.

b) Mecanismo de intercambio de información [27]: Para la creación de nuevas funciones, se utiliza un mecanismo de intercambio de información, el cual es el principal cambio que tiene respecto al algoritmo original y es debido a la representación de la estructura de las soluciones como árboles. Entonces, para generar una nueva solución v_i en ABCP, una solución es aleatoriamente escogida

llamada j_i, luego se selecciona una solución vecina j_k, que corresponde a otra solución de la misma generación, de forma aleatoria, y de ese árbol es seleccionado un nodo interno con un probabilidad P_{ip} (su valor es generalmente 0.9) o un nodo hoja con una probabilidad $1 - P_{ip}$. Este nodo elegido determina qué y cuánta información será compartida con la nueva solución. Después, un nodo de la solución actual j_i va a determinar cómo será utilizada esta información, el cual es elegido bajo las mismas condiciones anteriores. Por lo tanto la nueva solución v_i es generada reemplazando el nodo elegido de j_i por el seleccionado en j_k. Un ejemplo de este mecanismo es el mostrado en la Figura 3.8.

c) Proceso de selección codiciosa [27]: Luego de que v_i es evaluado, debe ser comparado con j_i, y si v_i es mejor que j_i, reemplaza a j_i y se convierte en un nuevo miembro de la población. Es importante destacar que en este proceso una solución puede no sea reemplazada durante toda la iteración del algoritmo, ante esto se introduce el parámetro límite.

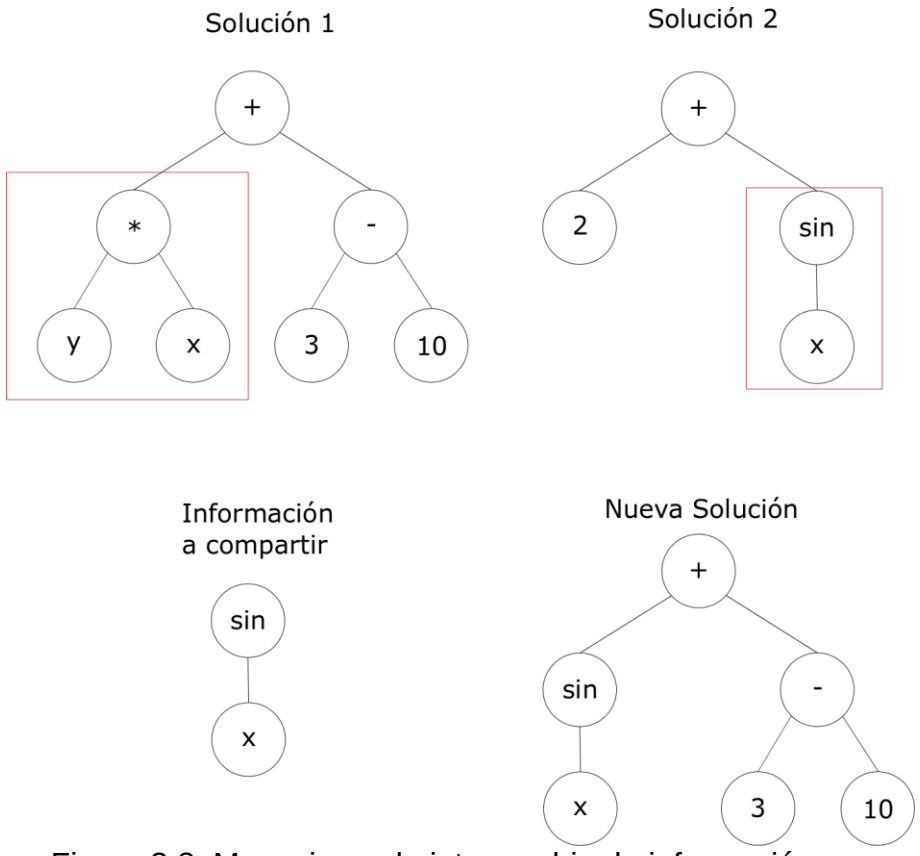

Figura 3.8. Mecanismo de intercambio de información.

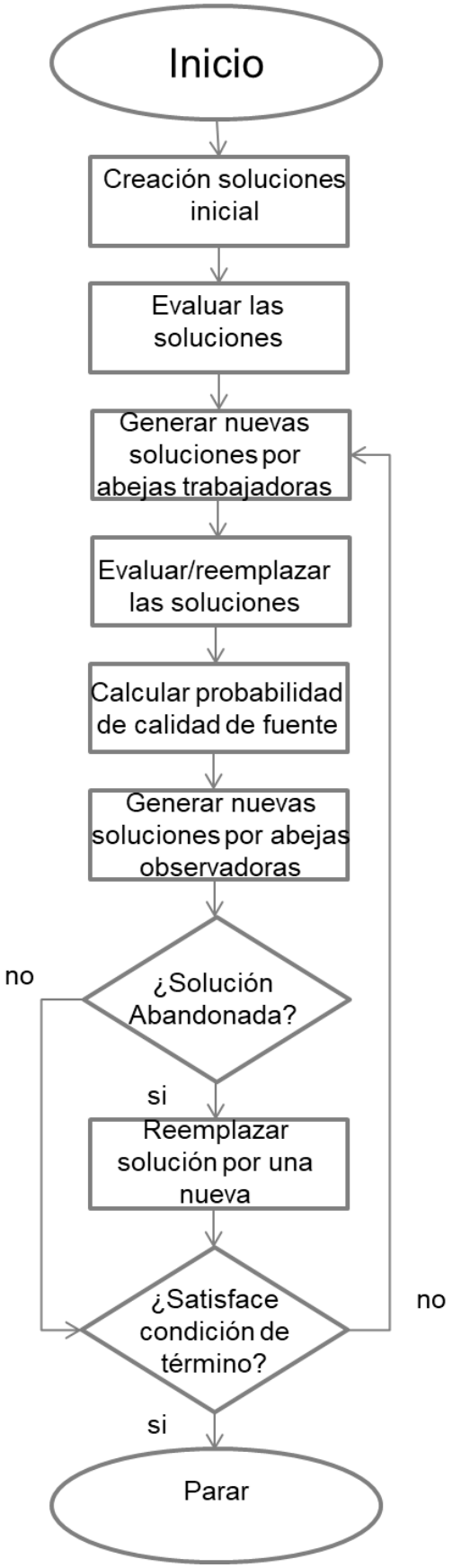

Figura 3.9: Esquema de trabajo algoritmo ABC, adaptado de [27].

d) Límite [27]: Como el mecanismo de intercambio de información es aleatorio, no todas las soluciones serán reemplazadas al aplicar el proceso de selección codiciosa, ante esto, cada vez que una solución no es reemplazada se contabiliza, y si esta llega al valor de límite, es reemplazada por una nueva generada mediante el método $grow$. El valor de límite es fijado entre $[10,1000]$, siendo 500 el valor que muestra los mejores resultados.

El Algoritmo 2 muestra un pseudocódigo general de ABCP. En él se muestra que se genera una población inicial mediante el método *ramped half-half*, donde esta debe cumplir las mismas restricciones que GP en tamaño de población y máxima profundidad de los árboles creados. Luego la población debe ser evaluada utilizando la función de fitness, para continuar en el proceso iterativo donde la población va cambiando al generarse nuevas soluciones.

Algorithm 2. Algoritmo de Artificial Bee Colony Programming [27]

1: Generate initial functions (x_i) with ramped half-half method
2: Evaluate Functions
3: **REPEAT**
4: **for** each employed bee
Produce new function (v_i) by using information sharing mechanism
Evaluate the functions
Apply greedy selection process between x_i and v_i
5: Calculate the probability p_i for the functions
6: **for** each onlooker bee
Select a function x_i depending on p_i probabilisticaly
Produce new function (v_i) by using information sharing mechanism
Evaluate the functions
Apply greedy selection process between x_i and v_i
7: **if** there is an abandoned solution
Then replace it with new function generated by grow method
8: memorize the best solution so far
9: **until** (termination condition is satisfied)

El ciclo comienza con la fase de las trabajadoras, donde todas las soluciones generadas en la etapa inicial pasan por la etapa de intercambio de información (la cual se explicó en el punto b anterior), pudiendo ser reemplazadas por soluciones mejores en el caso que la solución generada por la etapa tenga un mejor fitness que la solución original (que sirve como base para crear la nueva solución). Luego se calcula una probabilidad a todas las soluciones p_i, la cual representa la preferencia que tendrán las abejas observadoras (en la fase de las observadoras) para elegir una solución sobre otra. Esta probabilidad depende de la calidad de la solución, (la cual representa la cantidad de néctar que tenga la fuente) es decir su fitness y es calculada según la ecuación 3.4.

$$p_i = \frac{0.9 \cdot fit_i}{fit_{best}} + 1 \tag{3.4}$$

Donde fit_i es la calidad de la solución i, o el valor entregado por la función de fitness para esa solución y fit_{best} es la calidad de la mejor solución encontrada entre todas las soluciones (fitness del mejor individuo).

Luego sigue la fase de las observadoras donde las soluciones candidatas para generar nuevas funciones son determinadas en base a la probabilidad calculada anteriormente, y para elegirlas se utiliza el método de ruleta. En esta fase también utiliza el mecanismo de intercambio de información para crear nuevas soluciones tomando como base la solución elegida por el método de ruleta. Estas nuevas soluciones son evaluadas mediante la función de fitness, e igual que en caso de la fase de las trabajadoras, se aplica el proceso de selección codiciosa en el caso que su fitness sea mejor al que tiene la solución original, reemplazando la solución original en la población.

Sigue la fase de las exploradoras que verifica si existe una solución abandonada, la cual corresponde a soluciones que no han sido reemplazadas, y son determinadas por el parámetro límite. Si existe alguna que haya llegado a este punto, se reemplaza por una nueva generada por el método *grow*. Después de esto, se guarda la mejor solución que exista hasta ese momento. El ciclo finaliza si se alcanza alguna de las condiciones de término. Cuando ocurre esto se retorna la mejor solución, y en el caso contrario, se continúa en el ciclo principal hasta que se alcance alguno de los criterios.

3.2.4. DYNAMIC ANT PROGRAMMING

El tercer algoritmo es Dynamic Ant Programming [39], el cual es una adaptación de Ant Colony Optimization , usado para la regresión simbólica. Al igual que ABC, ACO pertenece al grupo de los algoritmos de optimización basados en la inteligencia de enjambre, siendo uno de los primeros algoritmos de este tipo utilizados para resolver problemas de optimización, específicamente destinado a resolver el problema del vendedor viajero (TSP) en un tiempo de cómputo razonable.

La inspiración del algoritmo es el comportamiento social que tienen las hormigas reales, que cuando buscan su alimento, las hormigas exploran inicialmente el área que rodea su hormiguero de manera aleatoria. Tan pronto como una hormiga encuentra una fuente de alimento, esta evalúa la calidad y cantidad llevando consigo un poco de vuelta al hormiguero. En el viaje de regreso, la hormiga va dejando en el camino una feromona química, donde la cantidad que deposita va a depender de la cantidad y calidad de alimento encontrado, y sirve para guiar a otras hormigas a la fuente, lo cual se puede observar en la Figura 3.10. Esta interacción entre todas las hormigas permite encontrar

el camino más corto entre el hormiguero y la mejor fuente de alimento y es el principio utilizado para resolver los problemas de optimización [40].

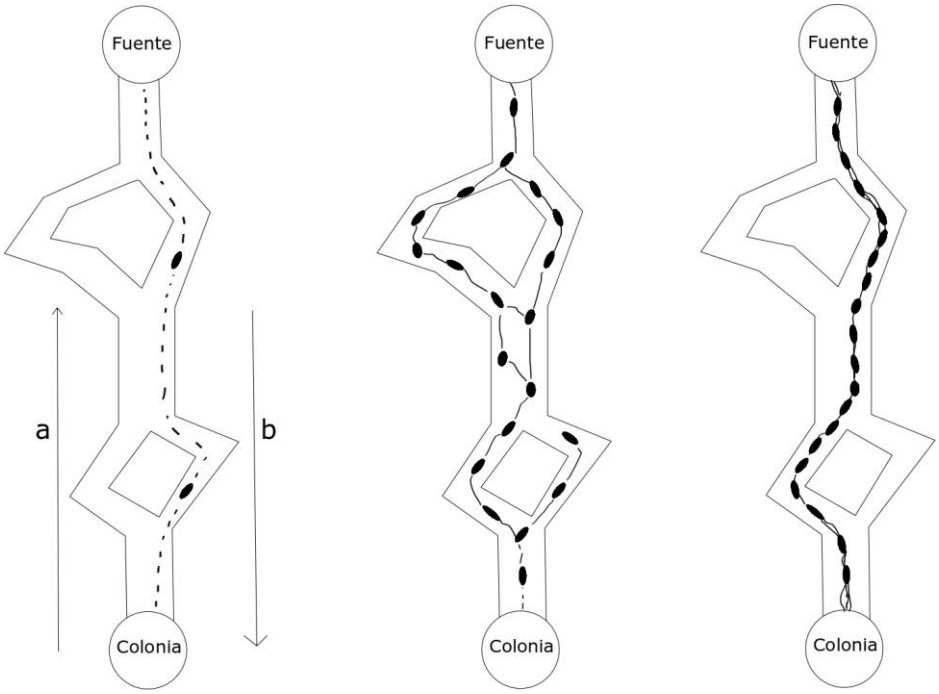

Figura 3.10: Búsqueda de alimento por parte de hormigas reales [40].

DAP no es el único algoritmo que se ha propuesto para realizar regresión simbólica utilizando como base ACO, también se encuentra [41]: Ant Programming (AP), Ant Colony Programming (ACP) y Generalized Ant Programming (GAP), pero DAP presenta ventajas respecto a los algoritmos anteriores, como la etapa de eliminación de nodos que serán mostradas en detalle a continuación, que permite a este algoritmo tener un rendimiento superior [39]. DAP, también representa a los individuos como estructuras de árboles, los cuales representan las posibles soluciones al problema planteado, pero la principal diferencia con GP o ABC, es que la población no se crea inicialmente y va cambiando a través de las iteraciones, sino que en cada iteración se crea un nuevo conjunto de soluciones (o nuevos caminos) según los pasos que hayan tenido más éxito al encontrar el alimento en el pasado (iteración anterior). Es importante destacar que las hormigas son las que crean las soluciones, y la población en este caso será el conjunto de caminos que crean las hormigas, es decir, cada hormiga va a crear solo un camino. La población inicial se define entonces como la cantidad inicial de hormigas.

Lo anterior se realiza mediante una tabla de feromonas, la cual representa qué tan probable es que la hormiga elija un nodo sobre otro. Su razón es porque la forma de construcción del árbol es similar a cómo la hormiga busca el camino a su alimento, es decir, la hormiga primero elije un nodo, y luego debe decidir, según el valor de la feromona, a qué nodo moverse hasta completar el paso. Aprovechando esta forma de construcción de los árboles, los pasos que sean menos probables que se utilicen se van

eliminando, por lo tanto, el tamaño de las estructuras es menor y evita el problema que posee GP con el crecimiento desmedido que pueden producirse en sus árboles. Como es posible que se eliminen nodos, la tabla de feromonas cambia dinámicamente a través de la ejecución de este algoritmo [40].

En la Figura 3.11 se encuentran los principales pasos de DAP, primero comienza con la definición del número de hormigas que participarán en el proceso, luego se entra en el ciclo donde la primera etapa es que cada hormiga debe construir el camino (solución), luego se debe evaluar pasos creados y guardar el que entrega el mejor camino. La siguiente etapa es actualizar la tabla de feromonas, utilizando la información del mejor camino creado, para seguir con la eliminación e inserción de los nodos. El proceso es iterativo hasta que se cumple alguna de las condiciones de término las cuales son las mismas que las mostradas para ABCP, a excepción que se utilice como función de fitness el presentado en la ecuación 3.5, donde la condición de término es que sea mayor a 0.95.

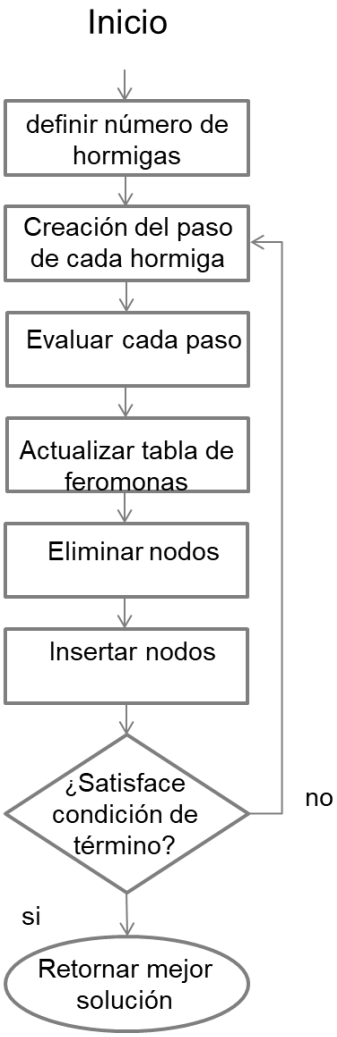

Figura 3.11: Esquema de trabajo algoritmo DAP, adaptado de [40].

$$fit_i = \frac{1}{1 + \sum_{i=1}^{n} |d(i) - c(i)|} \tag{3.5}$$

Donde $d(i)$ es el valor estimado para el paso i y $c(i)$ es el valor obtenido.

Características comunes con otros algoritmos y propias de DAP se presentan a continuación:

a) Tamaño de la población [40]: Los estudios sobre este algoritmo, indican que el tamaño de la población está fuertemente determinado por el problema objetivo y las funciones que se utilicen. En general se prefieren pocos individuos, puesto que las estructuras generadas tienen pocos miembros, entonces si se aumenta el número de hormigas, van a existir muchos caminos que serán iguales, a pesar de lo anterior el rango de número de individuos es alto. $[10{,}1000]$.

b) Número de generaciones [40]: El número de generaciones para el algoritmo es recomendable que sea alto, ya que en las primeras generaciones las etapas de eliminación e inserción de nuevos nodos no actúan frecuentemente dado que estas dependen de la tabla de feromonas, la cual por ejemplo, normalmente demora más de 20 generaciones en que un nodo alcance el valor τ_{minimo} . Dado lo anterior, el rango preferible para esto es entre $[100{,}2000]$.

c) Construcción de la estructura de árbol [40]: Para la construcción del camino, primero la hormiga elige un nodo inicial, luego elige el siguiente nodo utilizando el valor de la tabla de feromonas. Una característica importante es que la hormiga nunca elige un nodo interno que ya haya sido utilizado, por lo tanto en esta etapa se debe recordar los que fueron utilizados, pero puede elegir un nodo hoja las veces que quiera. Si se elige un nodo hoja, la búsqueda de ese camino finaliza. La característica de no repetir los nodos, es la que permite que las estructuras sean de tamaño pequeño o poca profundidad, un ejemplo de la creación de la estructura se encuentra en la Figura 3.12, la que muestra que todas las funciones son utilizadas solo una vez, y los terminales se repiten.

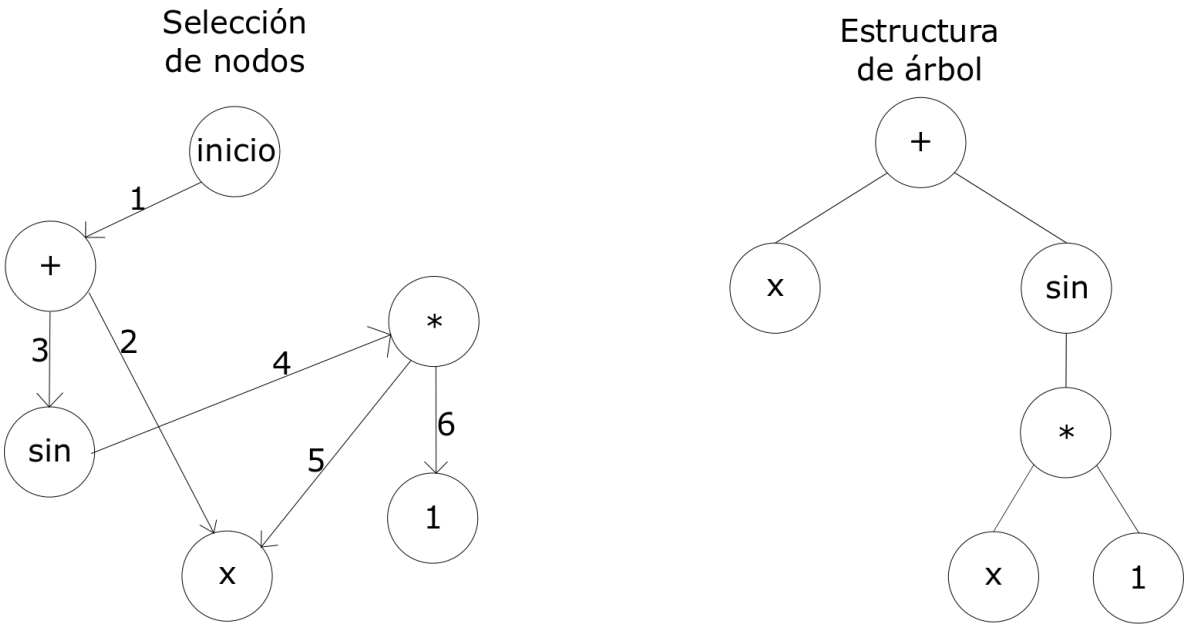

Figura 3.12: Ejemplo de creación del paso en DAP, la hormiga visita los nodos {inicio→+→x→sin→*→x→1} en la izquierda formando la estructura x+sin(x) a la derecha.

En esta etapa, mientras mayor sea el valor de la feromona, mayor es la probabilidad de que ese nodo sea elegido, donde la probabilidad de que una hormiga k posicionada en el nodo i se mueva al nodo j es la presentada en la ecuación 3.6.

$$p_{kij} = \frac{\tau_{ij}}{\sum N_i \tau_i} \tag{3.6}$$

Donde τ_{ij} es la cantidad de feromonas en el trayecto (i,j) y N_i es la cantidad de nodos que no han sido visitados. Un ejemplo de una tabla de feromonas se encuentra en la Tabla 3.1 donde las funciones son $[sin, +, *]$ y los terminales $[1, x]$. Donde por ejemplo, la celda $(1,2)$ representa la cantidad de feromonas dejadas en el viaje del nodo sin al terminal x.

Tabla 3.1: Ejemplo de Tabla de Feromonas [40]

Nodo o terminal	**1**	**x**	**sin**	**+**	*****
Sin	0.3	0.5	1	1	1
+	0.5	0.6	0.6	0.3	0.2
*****	1	0.3	0.2	0.1	0

d) Actualización de feromonas [40]: Para evitar el estancamiento de las soluciones, el rango de los valores de las feromonas es limitado en el intervalo $[\tau_{minimo}, \tau_{maximo}]$.

Estos valores son parámetros del algoritmo, donde sus valores óptimos discutidos en [40] son 0.1 y 1 respectivamente. Esta etapa ocurre luego de que todas las soluciones son evaluadas, y la tabla de feromonas es actualizada según la ecuación 3.7.

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + f_{ib}(t) \tag{3.7}$$

Donde ρ es un parámetro que va entre $[0,1]$ y es llamado *tasa de evaporación*, la cual representa la evaporación de las feromonas al pasar el tiempo, y $f_{ib}(t)$ es el valor de la función de fitness del mejor camino encontrado en esa iteración. Es importante destacar que la cantidad de feromonas en la iteración inicial es igual a τ_{maximo}.

e) Eliminación e inserción de nodos: Estas etapas ocurren después de la actualización de la tabla de feromonas. Un nodo es eliminado si todos los valores de feromonas a ese nodo son τ_{minimo}.

Además, en cada iteración se determina si se inserta o no un nuevo nodo, la cual es definida por la probabilidad mostrada en la ecuación 3.8.

$$p(i) = \frac{\frac{1}{m}\sum_{k=1}^{m} L_i^k}{N_i} \tag{3.8}$$

Donde L_i^k es el número de nodos generados por la hormiga k; N_i es el número total de nodos y m es el número de hormigas (tamaño de la población). El valor de las feromonas para el nodo insertado es igual a τ_{maximo}. Ambas etapas suponen una modificación de la tabla de feromonas, por lo cual cambia dinámicamente en cada iteración, y a pesar de esto, las hormigas encuentran buenas soluciones utilizando soluciones parciales (ante el cambio de las soluciones en todas las iteraciones).

El Algoritmo 3 muestra el pseudocódigo de DAP, el que comienza con la creación de la tabla de feromonas, en base a la cantidad de funciones, constantes y variables que se definan. Luego de esto se entra al ciclo, en el cual cada hormiga va creando su propio paso, la forma que realiza esto es elegir un nodo inicial, y de ahí seleccionar el siguiente en base a una probabilidad que depende de la cantidad de feromonas que se haya depositado en ese trayecto. El proceso continúa hasta que la hormiga construye todo su camino, y se repite para todas las hormigas (su tamaño debe ser definido al inicio del algoritmo), obteniéndose una cantidad de soluciones igual al número de hormigas.

Algorithm 3 Algoritmo de Dynamic Ant Programming [40]

1: Create pheromone table
2: **Repeat**
3: **for** each ant; **repeat**
From current node i, select next node j, with probability p_i
until: full path has been constructed
4: Evaluate functions
5: Update pheromone
6: Deletion of nodes
7: Insertion of nodes
8: **until** (termination condition is satisfied)

Luego que el conjunto de soluciones se encuentra creado, este es evaluado por la función de fitness presentada en la ecuación [algo] y se debe guardar la solución que presente el mejor fitness. Con la información anterior, se debe actualizar la tabla de feromonas según lo presentado anteriormente, para dar paso a la etapa de inserción y eliminación de los nodos. Después de esto se verifica si alguna de las condiciones de término se ha cumplido, si es así, se retorna el mejor individuo encontrado, y en caso contrario se continúa con el ciclo hasta alcanzar los criterios.

3.2.5. VALIDACIÓN

Después que se haya creado algún modelo que sea capaz de representar los datos experimentales, es necesario evaluar su calidad. Para la tarea anterior, comúnmente se utilizan los métodos de remuestreo, los cuales son métodos originalmente utilizados para la estimación de parámetros que no poseen una distribución teórica previa [42]. A partir del remuestreo se elabora una distribución empírica de los parámetros a estimar, lo cual permite una evaluación de la estabilidad de los valores encontrados.

Los métodos de remuestreo más usados son 3: Bootstrap, validación cruzada y Jackknife[43]. El método que es utilizado para validar los modelos de los algoritmos anteriores es Jackknife, ya que aproxima mejor los valores estadísticos que el método de validación cruzada y entrega buenos resultados al trabajar con pocos datos experimentales como es este caso. Este método consiste en repetir un ajuste de parámetros N veces, donde N es el número de muestras que se tengan. Luego en cada iteración se retira uno de los datos experimentales del conjunto de entrenamiento y con los nuevos parámetros encontrados del ajuste se predice el valor del elemento que fue retirado. De esta forma en cada iteración se calcula el error de predicción del modelo y la variación entre los parámetros originales y los obtenidos con cada subconjunto de datos lo que permite que al final del proceso se tenga una estimación del error de generalización, lo cual es entregado por el valor Jackknife Mean Square Error (MSE_{jk}) que se encuentra definido en la ecuación 3.9[44].

$$MSE_{jk} = \sum_{i=1}^{N} \left| \widehat{Y^{-\iota}}_i - Y_i \right| \tag{3.9}$$

$\widehat{Y^{-\iota}}_i$ es el valor predicho y Y_i es el valor experimental que es retirado de la muestra. Es importante destacar que este valor no es posible analizarlo por sí mismo, sino es un indicador utilizado para comparar la capacidad de generalización de los modelos. También, con este método es posible calcular los pseudo-parámetros, los que pueden ser utilizados para determinar los parámetros de interés ya que corresponden a un estimador no sesgado de los parámetros obtenidos en la etapa de ajuste. Estos son calculados utilizando la ecuación 3.10.

$$a^* = \frac{1}{N} \sum_{i=1}^{N} (N \cdot a - (N-1) \cdot a_i) \tag{3.10}$$

a^* corresponde al valor del pseudo-parámetro, a es el valor original del parámetro, N es la cantidad de datos observados y a_i es el parámetro obtenido al retirar el valor Y_i.

Este método es conocido también como *leave-one-out* y aproxima de mejor manera los valores estadísticos deseados en comparación con otros métodos como validación cruzada [45] ya que utiliza la mayoría de los datos de entrenamiento. Sin embargo, por esa razón es más costoso computacionalmente por realizar muchas veces un ajuste. Dado el costo anterior, este método es utilizado cuando no se tienen una gran cantidad de datos experimentales.

3.3. ESTUDIO EXPERIMENTAL DE ABSORCIÓN DE HIERRO

Existe un estudio anterior realizado por A. Colins en el cual realiza un estudio de la absorción apical de hierro. Para lo anterior se utilizó un sistema *in vitro* (el cual corresponde a ensayos que son realizados en un ambiente controlado fuera de un organismo vivo), ya que presenta ventajas sobre sistemas *in vivo*, como por ejemplo, eliminar la variación entre animales que permite determinar con precisión las variables y condiciones en que se realiza el experimento (temperatura, pH) [28].

A. Colins para el estudio de células epiteliales del intestino utilizó la línea celular Caco-2, la cual proviene del adenocarcinoma del colon humano y tiene la capacidad de diferenciarse espontáneamente y formar monocapas de enterocitos intestinales maduros. Esta línea ha sido ampliamente usada en ensayos de toxicología y farmacología, y aunque su cultivo y diferenciación tienen algunos inconvenientes en la

reproducibilidad de los experimentos, es el mejor modelo de barrera intestinal existente [46].

La mayor diferencia entre las células Caco-2 y los enterocitos es que los últimos presentan una capa gruesa de glicolix en la superficie apical, por lo que debe ser considerada a la hora de extrapolar los resultados de modelos *in vitro* a *in vivo* [47]. En la sección A de anexos se detalla el método utilizado para determinar los datos experimentales.

El experimento realizado por A. Colins consideró en un comienzo el cultivo de células Caco-2, el cual como primera parte se debe asegurar que éste presenta las condiciones adecuadas. La característica más importante que debe presentar el cultivo es que la monocapa esté íntegra, con lo cual se evita el paso de hierro por transporte paracelular.

Los ensayos realizados por A. Colins se realizaron con cloruro férrico ($FeCl_3$) utilizando 3 concentraciones ($5\mu M, 10\mu M, 20\mu M$) en que se determinó la cantidad de hierro absorbida por el lado apical en un lapso de tiempo de 15 minutos. En los resultados del experimento anterior, presentados en la Figura 3.13 se observa que a medida que aumenta la concentración en la cara apical, mayor es la absorción [48].

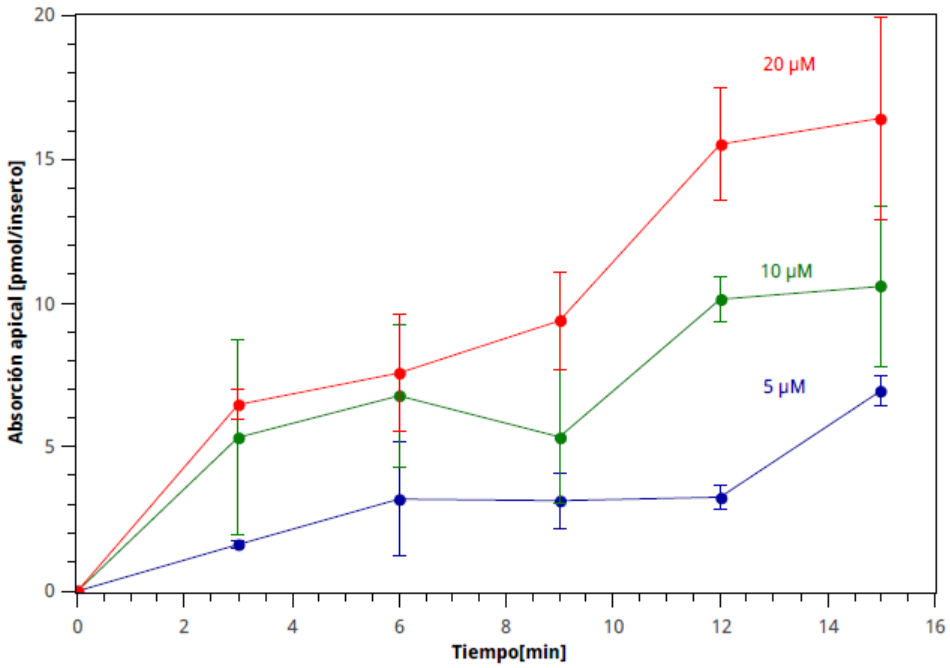

Figura 3.13: Resultado determinación experimental absorción de hierro en células Caco-2. Las barras representa la desviación estándar [48].

A. Colins analiza la Figura 3.13 en detalle, observando que en la concentración $10\mu M$ existe una velocidad negativa, lo que significaría que existen pérdidas por la membrana apical, entre 6 y 9 minutos, lo que no es posible ya que las proteínas transportadoras DMT1 están en cantidad muy superior a las FPN1.

Otro fenómeno importante considerado por A. Colins, es el hecho que en la Figura 3.13 no se observa una tendencia clara en el tiempo para cada concentración, es decir, no se nota una diferencia entre las curvas de absorción total entre las diferentes concentraciones. Además, los resultados están claramente afectados por la desviación estándar de los datos observados, pero aun así, el orden de magnitud de los resultados concuerda con otros ensayos realizados [49], para condiciones experimentales similares.

Como se mencionó anteriormente, las células Caco-2 poseen la desventaja de presentar una variabilidad natural entre los cultivos, lo que trae como consecuencia una desviación estándar mayor a la habitual, lo que puede ocurrir por muchos factores como la misma manipulación de los cultivos.

Por lo anterior, existen muchas incógnitas aún, sobre la fenomenología del proceso, por lo que para analizar los datos se utilizó un modelo empírico, el cual como resultado entrega ecuaciones que modelen el transporte de hierro, a través de la membrana apical. Para esto se utilizó el algoritmo genético dada su efectividad en aplicaciones para problemas fuertemente no lineales.

3.3.1. MODELO EMPÍRICO

Para la creación del modelo empírico, A. Colins optó por utilizar el algoritmo de programación genética, ya que su observación de los datos experimentales obtenidos anteriormente, notaron comportamientos altamente no lineales, lo que le agrega complejidad al problema, al no poder establecer una función de regresión a priori.

El algoritmo utilizado por A. Colins se desarrolló según los parámetros mostrados en la Tabla 3.2. Recordando que el algoritmo posee una gran cantidad de pasos y elementos determinados probabilísticamente, cada vez que se ejecuta encontrará un nuevo modelo que se ajuste a los datos experimentales (funciones con estructuras y parámetros diferentes), debido a esto, el estudio consideró 50 repeticiones para obtener 50 modelos con la finalidad de encontrar al más adecuado.

Tabla 3.2: Parámetros utilizados para la Programación Genética [34].

Parámetro	Valor o criterio
Profundidad inicial del árbol	6
Máxima profundidad del árbol	15
Tamaño de Población	500
Número de Generaciones	100
Población inicial	Ramped-Half-Half
Funciones	$cos, sin, +, -, *, /, a^b, \ln, exp$
Terminales	Variables: C_0, t Constantes: $1, 10, 100, 1000$
Límite	500

Por lo anterior, se definen criterios que permitan establecer el modelo más apropiado para representar los datos experimentales; y el que mejor siguiera las características fenomenológicas del problema de forma general. Los criterios presentados fueron: Descartar modelos que presenten coeficiente de determinación (R^2) menor a 0.8, también los que no tengan sentido biológico al presentar concentraciones negativas y elegir el que presente menor error de generalización MSE_{jk}.

En la ecuación 3.11 se muestra el mejor modelo obtenido en esta etapa del estudio, donde su coeficiente de determinación es $R^2 = 0.84$ lo que indica que es capaz de representar adecuadamente el conjunto de datos. Además, su error de generalización es igual a $MSE_{jk} = 1.45$. Si se analiza la ecuación, ésta indica comportamientos no lineales, además de presentar comportamientos oscilatorios, lo cual no es posible ya que representaría una salida de hierro a través de la cara apical [48].

$$[Fe_{in}(C_0, t)] = \frac{\sin\left(sin\left(\left(\frac{t}{exp(exp(1))}\right)cos\left(sin\left(\frac{t}{exp\left(\frac{t}{exp\left(\frac{t}{exp(1)}\right)}\right)}\right)\right)^{-1}\right)\right)}{\cos(sin(C_0 + 5))} \tag{3.11}$$

Por comportamientos como el anterior, A. Colins decide seguir otra metodología que considera el error de generalización además de un ajuste de parámetros. El primero para eliminar la ambigüedad de la definición de los parámetros del algoritmo (determinados arbitrariamente) y lo segundo para aumentar la capacidad de generalización de los modelos. Por lo anterior se agregó una etapa de validación

Jackknife después del ajuste de parámetros que además considera que la función de fitness del algoritmo es la medición del error de generalización MSE_{jk}.

El estudio enfatiza que para esta etapa, al ser un problema de optimización por sí mismo, se debió limitar los recursos, por ejemplo, el número de modelos generados por este método se disminuyó a 20, y también se cambiaron parámetros como el número de generaciones que disminuyó a 50. Lo anterior es debido a que el tiempo de cómputo aumenta de manera significativa (minutos vs días), por lo que se debió compensar este hecho para encontrar un modelo que sea suficientemente bueno en un tiempo razonable [48]. En la sección B de anexos se encuentra explicado en detalle la realización de estas etapas.

Dado lo anterior, y utilizando los mismos criterios presentados anteriormente, el mejor modelo del estudio en esta etapa es la función representada por la ecuación 3.12 y la representación de los datos de entrenamiento se encuentran en la Figura 3.14.

$$[Fe_{in}(C_0, t)] = \frac{\sin\left(\frac{1.898}{-1353.453}^{C_0 - t}\right) + \left(\frac{C_0}{3.364} \cdot t^{0.138}\right)}{0.939^t \cdot 1.108^{C_0}} \tag{3.12}$$

El estudio realiza un análisis de la ecuación anterior, y al igual que la ecuación 3.11, se ven componentes no lineales que afectan los flujos de absorción. Además su coeficiente de determinación es igual a $R^2 = 0.85$, y el análisis detallado de su coeficiente de generalización, el cual es $MSE_{jk} = 1.329$, indica que es un valor menor al encontrado anteriormente, lo que se traduce en que al emplear el cambio en la función de fitness permite aumentar la capacidad de generalización del modelo [48].

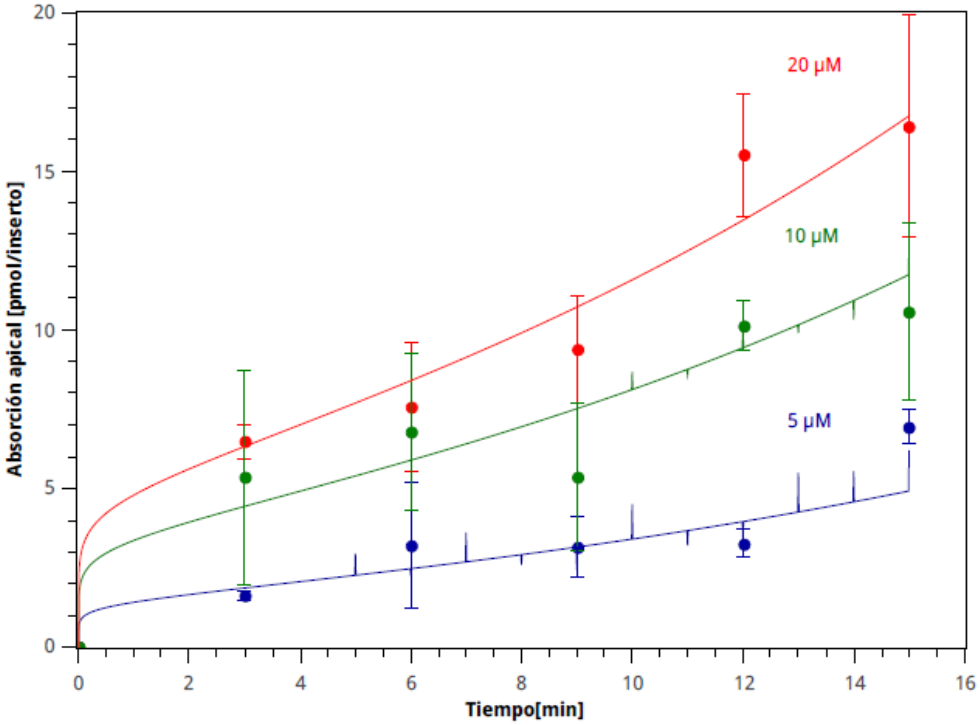

Figura 3.14: Simulación del modelo empírico de la ecuación **3. 12** [48].

Con lo realizado anteriormente, el estudio deduce que agregar una etapa de ajuste de parámetros, junto con imponer que la función objetivo sea el error de generalización, aumenta la capacidad de estimar nuevos datos observacionales, sin perder las propiedades del ajuste de parámetros. En otras palabras, se destacan las cualidades esperadas más importantes de ambas etapas.

Lo planteado en este estudio es fácilmente reproducible en otro tipo de problemas similares, ya que la baja capacidad de generalización es un rasgo típico para los modelos empíricos, sin embargo, es necesario considerar los costos de esta metodología, ya que es sabido que el método de programación genética es apto para problemas que tienen poco y nada de información, y al agregar este tipo de etapas aumenta el tiempo de ejecución por lo que se debe considerar un equilibrio dependiendo de los resultados que se quieran lograr [48].

4. MATERIALES Y MÉTODOS

4.1. MATERIALES

Matlab: Software utilizado para realizar tanto la programación de los algoritmos como para la creación de los modelos empíricos. Lo anterior se complementó con la toolbox GPlab implementada en Matlab.

Qtiplot: Software empleado para la construcción de los gráficos presentados.

4.2. MÉTODOS

4.2.1. IMPLEMENTACIÓN DE ALGORITMOS

Después de buscar algoritmos alternativos a la programación genética que utilicen la regresión simbólica se decide utilizar los algoritmos Artifial Bee Colony Programming y Dynamic Ant Programming. Al no encontrarse implementaciones en ningún lenguaje computacional se determina realizar la programación en Matlab apoyado por la toolbox GPlab.

Para ABCP, se siguieron los pasos mostrados en la Figura 4.1 para su implementación. Primero se realizó una búsqueda en GPlab dada la similitud del algoritmo con GP, para luego seleccionar los métodos comunes y utilizarlos en la implementación del algoritmo. Luego se programaron los métodos característicos del algoritmo, el cual el detalle se encuentra en la sección C de anexos, para finalizar con la validación de la implementación.

Esta validación se utiliza con funciones de prueba las cuales son funciones estándar utilizadas cuando se quiere validar un algoritmo utilizado con la técnica de regresión simbólica, donde el objetivo de la validación es que el fitness obtenido con estas funciones sea menor a 1. [35]

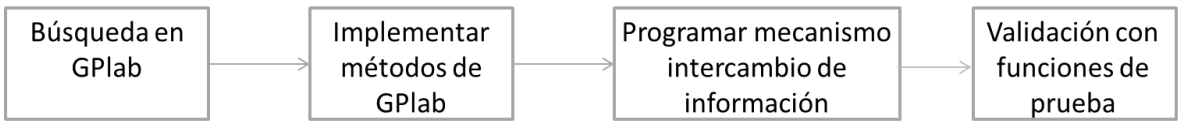

Figura 4.1: Etapas para implementar ABCP.

Para DAP, se siguió la misma metodología anterior mostrada en la Figura 4.2 para su implementación. Dada la gran diferencia de DAP con los algoritmos anteriores, la búsqueda en GPlab fue menos exitosa, debiéndose programar la mayoría de sus métodos solamente en base a la descripción bibliográfica del algoritmo. El detalle de

esto se puede encontrar en la sección C de anexos, y se finaliza igual que en el caso anterior con la validación utilizando las funciones mostradas en la Tabla 4.1.

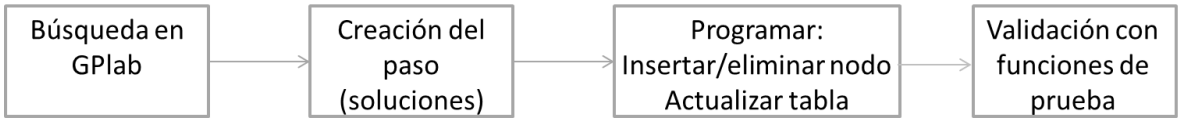

Figura 4.2: Etapas para implementar DAP.

En la validación, la función de fitness utilizada fue el Mean Square Error (MSE) presentado en la ecuación 3.4 siendo el empleado en todos los casos de validación consultado recordando que este debía ser menor a 1 en todas las funciones aplicadas para que se considerara una buena implementación.

Tabla 4.1: Funciones de prueba para validación de algoritmos.

Funciones	**Datos de entrenamiento**
$F_1 = x^3 + x^2 + x$	20 puntos aleatorios entre [-1,1]
$F_2 = \sin(x^2) \cdot \cos(x) - 1$	20 puntos aleatorios entre [-1,1]
$F_3 = \log(x+1) + \log(x^2+1)$	20 puntos aleatorios entre [0,2]

Es importante destacar que los parámetros de los algoritmos utilizados en las pruebas, son los mostrados en la Tabla 4.2 para ABCP y en la Tabla 4.3 para DAP, los cuales son los parámetros propios que aseguran el buen desempeño de cada uno de los algoritmos.

4.2.2. CONSTRUCCIÓN DE LOS MODELOS

Luego de realizar la validación de los algoritmos, se debe construir los modelos para ambos algoritmos en el problema de absorción de hierro. Para lo anterior se utilizaron 2 versiones siguiendo el principio de la regresión simbólica. En todos los casos el modelo tiene 2 variables, la concentración inicial de hierro en el medio apical y el tiempo; mientras que la variable de salida es la cantidad de hierro que traspasa la cara apical de células. Los datos experimentales o datos de entrenamiento son la absorción a distintas concentraciones en los 15 primeros minutos mostrados en la sección 3.3.

Las 2 versiones varían la forma en que se determinan los parámetros del modelo y la función de fitness utilizada. Para ABCP en su forma clásica, la función de fitness utilizada consiste en minimizar el Error Cuadrático Medio o Mean Square Error (MSE).Como DAP utiliza una función diferente de fitness, mostrada en la ecuación 3.5, se elige utilizar de forma paralela la función de fitness de ABCP para elegir el mejor modelo, ya que el fitness de DAP es importante para la ejecución del algoritmo al ser utilizado en la etapa de modificación de la tabla de feromonas. Esta es una modificación

leve a la forma clásica que no supone una versión diferente y fue la utilizada para todos los análisis.

En la versión modificada (versión error de generalización “EG”), se agrega una etapa de ajuste de parámetros y un paso de estimación del error de generalización mediante el método de Jackknife, en el cual su valor fue el utilizado como función de fitness.

Para ABCP, el ajuste de parámetros se realiza cuando se deban evaluar los individuos, sea al comienzo o cuando se generan nuevos individuos y el error Jackknife es calculado justo después del ajuste y es utilizado como nueva función de fitness. Es importante destacar que para esta etapa de refinamiento cambia el número de generaciones de 100 a 50, pero el resto de los parámetros son los utilizados en la versión clásica los cuales se muestran en la Tabla 4.2.

Tabla 4.2: Parámetros de ABCP.

Parámetro	**Valor o criterio**
Profundidad inicial del árbol	6
Máxima profundidad del árbol	15
Tamaño de Población	500
Número de Generaciones	100
Población inicial	Ramped-Half-Half
Funciones	$cos, sin, +, -, *, /, a^b, \ln, exp$
Terminales	Variables: C_0, t Constantes: $1, 10, 100, 1000$
Límite	500

Para ABCP en su versión clásica se realizaron 50 repeticiones, y para la versión EG 20. Cada repetición contiene un modelo capaz de representar los datos experimentales y para determinar el mejor se utilizaron diferentes criterios, por ejemplo, analizar la función de fitness, ver que se cumpla la mayor cantidad de restricciones fenomenológicas y validar los parámetros utilizando el método de Jackknife.

En DAP, y siguiendo con la metodología empleada en ABCP, el ajuste de parámetros se realiza en la evaluación de los individuos y el error de Jackknife se calcula justo después de esta etapa. Nuevamente este error es utilizado como función de fitness en este algoritmo, donde el mejor individuo se elige utilizando el error de Jackknife, pero para los métodos del algoritmo se sigue utilizando la antigua función de fitness del individuo seleccionado. Al igual que en ABCP, se cambia el número de generaciones a 50 manteniendo invariados el resto de los parámetros de la versión clásica, los que se muestran en la Tabla 4.3.

Nuevamente para la versión clásica de este algoritmo se realizaron 50 repeticiones y 20 para EG donde cada repetición representa un modelo y se utilizaron los mismos criterios de ABCP para seleccionar el mejor.

Finalmente se realizó una comparación del desempeño de los 3 algoritmos para determinar el que entrega mejores resultados. Los criterios que se utilizaron para realizar esta comparación fueron el valor de las funciones de fitness encontradas por los mejores modelos en sus 2 versiones, el tiempo de ejecución y las estructuras generadas.

Tabla 4.3: Parámetros de DAP.

Parámetro	**Valor o Criterio**
Tamaño de Población	500
Número de Generaciones	100
Funciones	$cos, sin, +, -, *, /, a^b, \ln, exp$
Terminales	Variables: C_0, t Constantes: $1, 10, 100, 1000$
Tasa de evaporación	0.5
Cantidad mínima de feromonas	0.01
Cantidad máxima/inicial de feromonas	1

5. RESULTADOS Y DISCUSIONES

A partir de la metodología planteada anteriormente, a continuación se muestran los resultados obtenidos para la validación de los algoritmos ABCP y DAP además de los modelos empíricos generados. Lo que se busca estimar con estos modelos es la cantidad de hierro que ha traspasado la cara apical en el tiempo, en que la variable principal es la concentración inicial en el medio apical y el tiempo. Para lo anterior, se utiliza como entrenamiento del modelo los datos experimentales obtenidos por A. Colins en su estudio [48] los cuales son mostrados en la Figura 3.10**.** Para cada modelo, los resultados son analizados individualmente para ver si estos pueden representar de alguna manera el fenómeno. También son comparados entre ellos y con los de A. Colins (el cual utiliza la programación genética para construir el modelo empírico) para determinar cuál algoritmo posee el mejor desempeño en este problema en particular.

Cabe destacar, que ante la naturaleza del problema o calidad de los datos experimentales, los algoritmos en su versión clásica no generaron modelos que pudieran abordar completamente el fenómeno, situación similar a la encontrada en estudios anteriores utilizando la programación genética. Por lo anterior, se siguió la metodología realizada con el algoritmo anterior, y tanto a ABC como a DAP se le realizaron las mismas modificaciones a sus versiones clásicas que se realizaron en GP para poder representar de mejor manera el fenómeno.

5.1. VALIDACIÓN DE ALGORITMOS

Antes de desarrollar los modelos para la absorción de hierro, primero es necesario saber si estos se encuentran bien implementados, ya que DAP y ABCP fueron programados casi desde cero. Para verificar esto, se utilizaron funciones de prueba con las cuales el objetivo fue que de 100 modelos generados en las versiones clásicas de los algoritmos, el mejor debía tener un fitness menor a 1 para las funciones de prueba. Los resultados de la implementación se encuentran en la Tabla 5.1 utilizando las funciones de prueba mostradas en el Capítulo 3.

Tabla 5.1: Resultados del valor de fitness para funciones de prueba en algoritmos.

Algoritmo	**F1**	**F2**	**F3**	**F4**
ABCP	0.03	0.09	0.13	2.7
DAP	0.02	0.18	0.7	3.2

De los resultados mostrados, los dos algoritmos implementados muestran un fitness menor a 1 para todas las funciones, por lo que se concluye que sí son capaces de resolver el problema de regresión simbólica. Analizando los resultados, por ejemplo para **F1** en ABCP se encontraron funciones con muchos términos, teniendo un árbol de profundidad 9 y teniendo funciones como log, sin, cos, siendo una función muy compleja de analizar. Lo mismo ocurrió para el resto de las funciones manteniéndose el

sobre-crecimiento de las estructuras para mejorar el fitness, con lo cual se espera que para los resultados para un problema particular presente este tipo de comportamiento siendo en general desviada con respecto a la función real pero que se ajusta correctamente.

Otro punto es si agrega ruido a los datos, es decir variar el valor real a uno muy cercano lo cual se asemeja a un problema real. Lo anterior se realizó para **F1**, con lo cual el fitness aumentó a 0.12 en ABCP y 0.24 para DAP, lo que se traduce en que los modelos intentan ajustarse a los nuevos datos con ruido, pero con un desempeño menor al agregarle complejidad al problema.

Por otro lado DAP es mejor en una función solo dependiente de la variable x, empeorando el desempeño cuando existen otro tipo de funciones. En ABCP, se muestra el mismo comportamiento, pero menos pronunciado, por lo que se puede estimar que ante procesos muy difíciles de describir, entregará mejores resultados además respaldado por el punto anterior que al agregar ruido, el desempeño de ABCP no se vio tan afectado como en DAP.

El último punto a analizar es lo que ocurre con problemas multivariados, como es el caso de la absorción de hierro. Esto es analizado en [27][39] los cuales muestran que al utilizar 2 variables, el fitness es superior a 1. Para mostrar lo anterior se utiliza la función de prueba sugerida la cual es $\boldsymbol{F4} = \sin(x) + \sin(y)$ para ambos algoritmos encontrando como resultado lo mostrado en la Tabla 5.1 lo que indica un rendimiento más bajo al plantear problemas multivariados, pero de todas formas se entrega un buen ajuste a los datos.

Dado que la implementación de los algoritmos es correcta, es posible comparar sus desempeños en el problema de absorción de hierro, para esto se suman los resultados obtenidos en GP, donde la comparación se basó en el tiempo de ejecución, funciones entregadas por sus mejores modelos, y los resultados estadísticos de estos. La comparación se hizo primero para el caso clásico de todos los algoritmos, y luego para su versión con error de generalización, ya que las versiones clásicas comparten los mismos parámetros básicos y en la versión modificada se realizaron cambios a estos parámetros, por lo que para que sea justa la comparación, se eligió de esta forma.

5.2. MODELO DESARROLLADO CON ABC CLÁSICO

Este algoritmo se ejecutó según los parámetros mostrados en la Tabla 4.2. Dada las características del algoritmo, cada vez que es ejecutado se genera un nuevo modelo que se intenta ajustar a los datos experimentales, que corresponden a funciones con estructuras y parámetros diferentes. Por lo anterior, y para encontrar el modelo que mejor se adapte a los datos experimentales se realizaron 50 repeticiones, con lo cual respectivamente se generaron 50 modelos.

Las 50 repeticiones es para tener una muestra considerable de modelos para evaluar, además, es el mismo número utilizado para encontrar los modelos utilizando programación genética, por lo cual es importante mantener los parámetros lo más similar posible si se quiere realizar una comparación justa.

Con los 50 modelos seleccionados se debió escoger el más adecuado que permitiera representar lo datos experimentales y capturar la fenomenología del problema de forma general. Para lo anterior, se definieron criterios para discriminar entre los modelos y así poder encontrar el que cumpliera con lo deseado. Los 3 criterios son:

1. Descartar los modelos que tuvieran un coeficiente de determinación (R^2) menor a 0.8, por representar un bajo desempeño.
2. Descartar los modelos que presentasen concentraciones negativas, ya que carecen de sentido biológico.
3. Seleccionar el modelo que presentase menor error de generalización MSE_{jk}.

Al realizar la selección según los criterios antes mostrados, el mejor modelo es el presentado en la ecuación 5.1 y la simulación de los datos se puede ver en la figura 5.1. Este modelo presenta un coeficiente de determinación $R^2 = 0.85$ lo que demuestra que es capaz de representar de buena forma el conjunto de datos. Cabe destacar que como los datos experimentales presentan desviación estándar, el coeficiente de determinación no podrá ser igual a 1, esperando la no linealidad de los modelos. Otro aspecto relevante encontrado, es que los 50 modelos tuvieron un buen desempeño con un coeficiente de determinación mayor a 0.8 lo que indica que este algoritmo permite ajustar bien datos de procesos desconocidos, lo que supone una ventaja sobre GP ante este tipo de problemas.

$$[Fe_{in}(C_0, t)] = 1 \cdot \log\left(\frac{\log\left(\log\left(100 C_o \, log\left(log\left({C_0}^{t}\right)\right) C_o \, log\left({C_0}^{t}\right)\right)\right)}{t sin\left(\frac{t}{C_0}\right)}\right) \tag{5.1}$$

De la ecuación anterior es posible notar su comportamiento no lineal, lo cual era uno de los supuestos planteados. Además, los logaritmos indican una función creciente y convexa, tal como es el caso del modelo seleccionado. Es importante resaltar que la presencia de la función sin, se podría traducir en un comportamiento oscilatorio, el cual se exhibe en el cambio de pendiente abrupto de la curva en el comienzo, esto debido a que a pesar de ser siempre creciente el logaritmo, al encontrarse dividido por el seno, provoca que en los puntos iniciales sea decreciente ya que la variable del tiempo se encuentra entre 0 y 1 pudiendo afectar también al logaritmo. Este comportamiento es poco probable en el sistema ya que una pendiente negativa significaría que existe una salida neta de hierro a través de la cara apical, situación que no es posible en este

sistema, por lo que este modelo escaparía de las características fenomenológicas esperadas.

De los datos estadísticos para este modelo, respecto al valor del error de generalización $MSE_{jk} = 1.4$, este no se puede analizar en sí mismo ya que es un término utilizado para comparar la capacidad de generalización de los modelos. Sobre los parámetros, estos difieren de los pseudo-parámetros como se muestra en la Tabla 5.1, lo cual es un indicativo que podría existir otro conjunto de parámetros que permita que el modelo tenga un mejor desempeño estadístico, lo cual se pretende lograr al integrar una etapa de ajuste de parámetros y utilizar como función de fitness el error de generalización para así obtener un mejor desempeño sin comprometer la capacidad de generalización del modelo obtenido.

Tabla 5.2: Datos estadísticos del modelo ABCP Clásico.

Parámetro	**Pseudo-parámetro**
1	2
100	-58.167

Como los modelos empíricos no utilizan información previa del sistema que se quiere conocer, encontrar un modelo que pueda satisfacer las características propias del sistema es muy difícil ya que solo intentan ajustarse a los datos experimentales. Así, de los 50 modelos seleccionados, observando los que no tienen concentraciones negativas, solo un 42% de los modelos son seleccionables. Lo anterior indica la dificultad para elegir los modelos si se comienzan a imponer condiciones que el algoritmo no sabe que debe cumplir, por ejemplo, qué sería lo esperado si se impone que el modelo no debe presentar oscilaciones, probablemente la cantidad de modelos seleccionables disminuiría, y estos podrían presentar un menor desempeño que los que no cumplan los criterios o en un caso extremo, que el algoritmo no presente soluciones que puedan ser analizadas, por lo que los criterios deben ser elegidos de forma razonable para que el algoritmo pueda desempeñarse de forma que pueda entregar resultados.

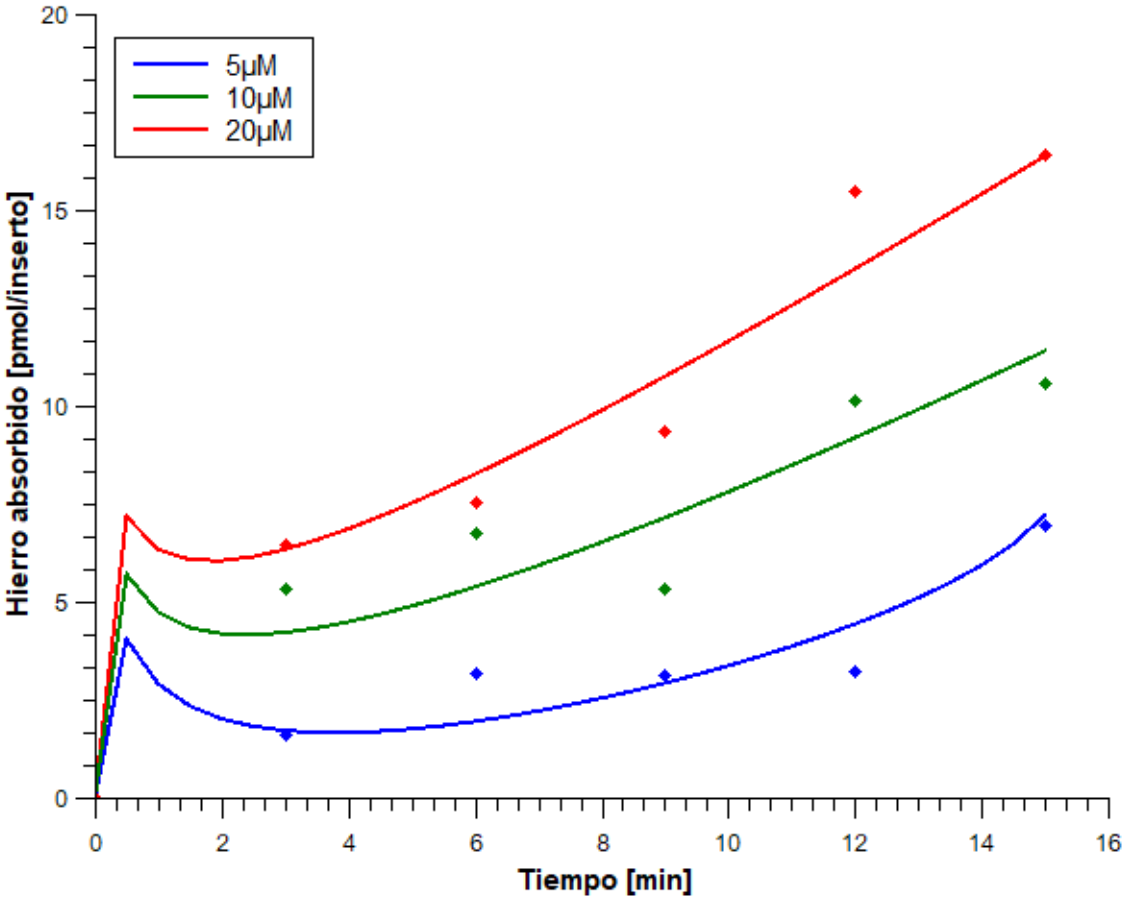

Figura 5.1: Representación mejor modelo ABCP clásico.

Otro aspecto a considerar en la ecuación 5.1 es la presencia de indeterminaciones, específicamente cuando $t = 0$ ya que se tendría una división por 0. La razón de la presencia de lo anterior en el modelo obtenido, es que el algoritmo considera la división como una función protegida, es decir, al encontrarse un caso así, esta función toma solamente el valor del numerador, que en este caso al evaluar la función en $t = 0$, su valor es 0, lo que no genera un problema. Como la construcción del modelo implica interacciones de los nodos de los árboles, obviar esta función (o el logaritmo), reduce la generalización que pueda tener el modelo, ya que, al limitar las funciones, produce que los modelos sean más particulares, situación no deseada ya que se espera la realización de un modelo empírico.

Analizando el desempeño del algoritmo, uno de los aspectos a considerar es la diversidad de la población, la que considera, qué tan únicos son los modelos generados o dicho de otro modo, si es que existen estructuras, variables o constantes que se repiten sucesivamente en los árboles generados. De los 50 modelos obtenidos, al compararlos entre ellos, se observó que no existe una convergencia clara hacia funciones determinadas, lo que significa que el algoritmo no se entrampa en una sola solución y por lo tanto genera una gran diversidad de soluciones que permite explorar diferentes alternativas al fenómeno que se quiere conocer. Esto de igual forma puede

significar que el sistema no converge, lo cual en este caso no es cierto ya que las restricciones impuestas a los modelos aseguran que los sobrevivientes cumplan con el sentido del problema y bajo este escenario se analiza la diversidad anterior.

5.3. MODELO GENERADO CON DAP CLASICO

Este algoritmo se ejecutó según los parámetros mostrados en la Tabla 4.3. El algoritmo cada vez que es ejecutado genera un nuevo modelo que se intenta ajustar a los datos experimentales por lo que igual que en el caso de ABCP, se ejecutó 50 veces para obtener 50 modelos.

Para mantener una relación en los algoritmos y el sistema, se utilizaron los mismos criterios de selección para encontrar el mejor modelo, exceptuando el del coeficiente de determinación, el cual se debió bajar a $R^2 = 0.75$. El motivo anterior fue por las características del algoritmo, el cual como se esperaba, genera funciones mucho más pequeñas, principalmente por su método de eliminación de nodos lo que evita el crecimiento desmedido del árbol como sucede en GP o ABC, que si bien existe un método de control, este solamente se basa en limitar o el número máximo de nodos de un árbol o la profundidad máxima de este. Lo anterior se traduce en un sobreajuste agregando funciones o constantes que no aportan al análisis, pero sí al fitness, por lo que su resultado es mejor en términos estadísticos. Como DAP no realiza esto por lo mencionado anteriormente, es esperable que su desempeño sea menor, lo que justifica bajar el valor del coeficiente de determinación.

Dado lo anterior, el mejor modelo seleccionado es el presentado en la ecuación 5.2 y la simulación de datos se puede ver en la Figura 5.2. Este modelo presenta un coeficiente de determinación $R^2 = 0.75$ lo que indica que no es capaz de representar de una buena forma los datos experimentales. Lo anterior se puede deber a limitaciones del algoritmo, sumado a la desviación estándar de los datos que pueden afectar su desempeño. El valor del error de generalización $MSE_{jk} = 3.43$, al tener el modelo anterior de ABCP como comparación, muestra que este modelo es tiene una menor capacidad de generalización e indica una menor capacidad predictiva ante nuevos datos.

$$[Fe_{in}(C_0, t)] = \log\left((C_0 - t + 1000)^{\frac{1 \cdot C_0 \cdot t}{100}}\right) \tag{5.2}$$

De la ecuación anterior se puede observar un comportamiento logarítmico, pero cercano a uno lineal. Dado que el modelo genera menos términos, es posible realizar un análisis “término a término” para analizar el comportamiento de la función, lo que indica que las constantes son las que entregan la forma final a la curva. Lo anterior hace notar la importancia de las constantes utilizadas, ya que pueden existir otras constantes que se adapten mejor al sistema, debido a que estas fueron elegidas de

manera arbitraria, y por otro lado, el algoritmo clásico no realiza un ajuste sobre estas, por lo que se vuelve muy relevante la manipulación de estos parámetros.

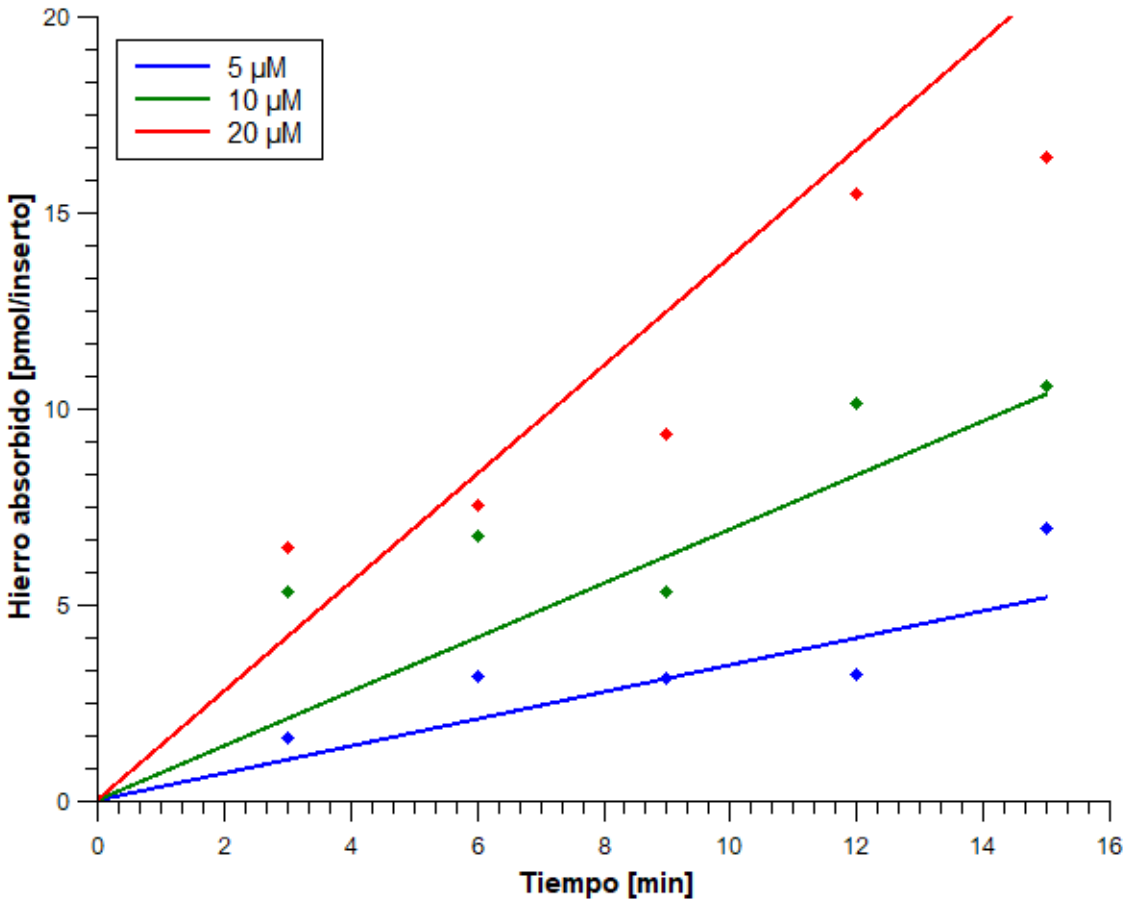

Figura 5.2: Representación mejor modelo DAP clásico.

Lo expuesto anteriormente es mostrado en la Tabla 5.2 la cual contiene los datos estadísticos, donde los parámetros mostrados difieren mucho de los pseudo-parámetros al igual que en el caso de ABCP, lo que confirma que estos algoritmos podrían tener una etapa de ajuste que permita encontrar constantes que se adapten mejor al sistema.

Tabla 5.3: Estadísticos para el modelo DAP Clásico.

Parámetros	**Pseudo-parámetros**
1	4
100	102
1000	−27.390

Otro aspecto relevante sobre las constantes, ya sea en DAP o en ABC, es que estas no se pueden simplificar, por ejemplo, si se tiene un nodo que contenga una multiplicación de dos constantes, este nodo no se va a poder alterar durante toda la ejecución del algoritmo (simplificarlo, ya que aún puede verse afectado por la selección en ABC o DAP). Sería interesante que exista una etapa de simplificación de los árboles como se muestran en la Figura 5.3 antes de que continúen el proceso de selección, ya que con esto podrían generarse nuevos parámetros además de contribuir a que el árbol no aumente su profundidad ni nodos de forma excesiva. Lo anterior se puede considerar un tipo de ajuste de parámetros que contribuiría a mejorar la calidad de las soluciones a costa de una disminución de la diversidad de la solución ya que este ajuste es una sub-solución del árbol original pudiendo perderse información valiosa gracias a la simplificación.

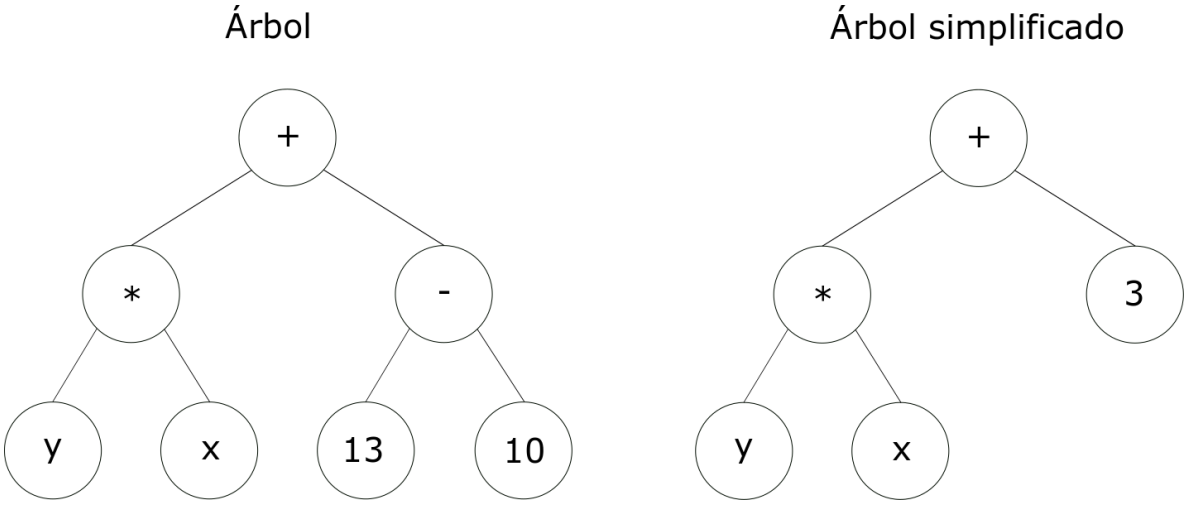

Figura 5.3: Simplificación de árboles, a la derecha el original, a la izquierda el simplificado.

Como se dijo anteriormente, DAP al entregar funciones con pocos términos, produce que tenga un bajo desempeño en este problema particular, en términos cuantitativos, solo 3 de los 50 modelos generados cumplían con el criterio de $R^2 > 0.75$, lo que dificultó el análisis. Además un 40% de los modelos presentó solo una dependencia de la concentración, los cuales fueron descartados por no poseer sentido ya que debe existir una dependencia de la concentración y el tiempo, ya que el experimento mide la cantidad de hierro que traspasa en el tiempo. El resto de los modelos no fueron seleccionables ya que en la mayoría de los casos su R^2 no fue superior a 0.5 confirmando el bajo desempeño de este algoritmo en este problema. Un aspecto destacable es que de los 50 modelos, ninguno presentó concentraciones negativas, por lo que desde el punto de vista del fenómeno, se adapta mejor al problema, por lo tanto se pueden introducir mejoras para aumentar el desempeño sin perder su capacidad de representar el proceso, por lo que la etapa de ajuste de parámetros es una buena opción para realizar esto.

Los parámetros utilizados en ambos algoritmos son los proporcionados por la literatura, los que en su mayoría fueron obtenidos en base a los resultados que entregaban estos con funciones de prueba como las presentadas en la Tabla 4.1, donde el ensayo y error ajustaron estos parámetros para encontrar el set que trabajara mejor en la gran mayoría de las funciones de prueba entregadas [27][39]. Al realizar esta metodología en la determinación de los parámetros es muy probable que para este problema en particular pueda existir un mejor set de parámetros que permita encontrar mejores resultados, pero al igual que lo planteado con las constantes numéricas, estos modelos empíricos trabajan bien bajo estos escenarios, por lo que en relación al tiempo y al objetivo del trabajo, no se hace necesario profundizar en la búsqueda de mejores parámetros para un posible mejor desempeño.

5.4. MODELO CON METODOLOGÍA DE ERROR DE GENERALIZACIÓN

Por la ambigüedad que generan los parámetros, tanto en su definición como en su selección, se hizo necesario modificar los algoritmos clásicos de ABCP y DAP para incluir una etapa de ajuste de parámetros que permita una mayor variabilidad a los parámetros y por lo tanto, obtener mejores modelos.

Ya que el fin es seguir una comparación a lo realizado por A. Colins en [48], el ajuste de parámetros se realizó de una forma similar utilizando el algoritmo Levenberg-Marquardt [50], el cual no posee restricciones para funcionar correctamente. Además, el criterio de término utilizado es el sugerido en [48], que establece como óptimo 10 iteraciones para no comprometer el desempeño del algoritmo principal, debido que a el ajuste de parámetros es un problema de optimización en sí, por lo que los recursos destinados a este deben ser limitados para dar prioridad a los algoritmos ABC y DAP.

Como sugiere A. Colins en su estudio, la implementación solo del ajuste de parámetros no es suficiente para obtener mejores resultados debido a que en sus modelos generados se encontró que existe un sobre-ajuste de los datos por lo que se pierde capacidad de generalización o capacidad para predecir nuevos datos [48], lo que no es deseable para este tipo de problemas. Siguiendo la metodología planteada por A. Colins, se realizó el mismo procedimiento de implementar una etapa de validación Jackknife a los algoritmos, con el objetivo de minimizar el error de generalización en lugar del error entre los datos y su predicción, para que los modelos tengan un buen desempeño representando los datos de entrenamiento, pero que no pierdan su capacidad de generalización.

El punto de partida para esta optimización son los valores entregados por los modelos de los algoritmos donde las constantes son seleccionadas al comienzo de este pudiendo ser estas adecuadas o no según lo discutido anteriormente. Ahora bien estos valores se pueden encontrar muy alejados de los mínimos locales del problema, lo que supone una desventaja a la hora de realizar esta optimización ya que al tener una

buena adivinanza inicial, mejores resultados entregará. Ya que depende de las constantes declaradas en un comienzo, puede que esta combinación no entregue mejores resultados al realizar la optimización en algunos casos y en otros mejore sustancialmente la solución.

Al realizar esta nueva etapa de optimización, es necesario acotar alguno de los parámetros de los algoritmos, y también el número de modelos generados. En las versiones clásicas uno de los parámetros es el número de generaciones definido como 100, el cual se decide cambiar a 50 para la versión con error de generalización. Lo anterior es dado por las etapas de refinamiento agregadas, las cuales aportan más operaciones al algoritmo y por lo tanto el tiempo de ejecución aumenta de manera significativa como se indica en el estudio [48]. Además, el disminuir las generaciones no afecta demasiado el desempeño del algoritmo, ya que los mayores cambios se producen en las primeras 10 generaciones y de forma particular en ABCP se pueden obtener mejores resultados ya que está comprobado que en las últimas generaciones solo se agregan estructuras que aportan al fitness, pero no tienen sentido a la hora de estudiar la función generada (por ejemplo se produce repeticiones de términos como lo mostrado en la ecuación 5.1) por lo que esta medida puede mejorar la calidad de la solución.

Dado el aumento del tiempo de ejecución, pasando de minutos a días en generar un modelo, se decide también disminuir el número de modelos generados de 50 en las versiones clásicas a 25 en las versiones con error de generalización. Con esto, se tienen menos modelos elegibles, pero refinamiento realizado puede compensar este hecho entregando, según lo que se espera, mejores modelos.

5.4.1. CASO ABCP ERROR DE GENERALIZACIÓN

Con las modificaciones al algoritmo clásico ABC para poder utilizar el error de generalización y ajuste de parámetros se seleccionó el mejor modelo, considerando los mismos parámetros y restricciones que en la versión clásica, el cual se puede apreciar en la ecuación 5.3 y la simulación de los datos se muestran en la figura 5.4.

$$[Fe_{in}(C_0, t)] = \frac{\log(-10.04 \cdot 9.004 \cdot t \cdot C_0 \exp(C_0)) \cdot {C_0}^{\cos(C_0 - 1183.71)} - ({C_0}^2 \cdot t \cdot -0.046)}{C_0} \quad (5.3)$$

El modelo anterior sigue mostrando el comportamiento no lineal de su versión clásica, además, su coeficiente de determinación $R^2 = 0.86$ es ligeramente mayor al de su versión clásica, por lo que este método supone una leve mejora para la representación de datos. Cabe destacar la disminución del número de términos, donde no se entrampa en repeticiones sucesivas de una misma función como el caso de la ecuación 5.1, lo que se puede considerar una mejora a la hora de analizar la función generada por el modelo. Como se ha dicho, este algoritmo no tiene una simplificación

de términos, lo que al ver la ecuación 5.3 se nota que podría existir al juntar los primeros términos dentro del logaritmo. De igual forma igual se realizó una simplificación menor de términos, ya que existía el término $\log(\exp(9.004))$ el cual es 9.004 por las propiedades del logaritmo. Lo anterior permite analizar de una forma más amigable la función, pero al realizar esta etapa manual, no representa el desempeño real del algoritmo ya que esta etapa no existe.

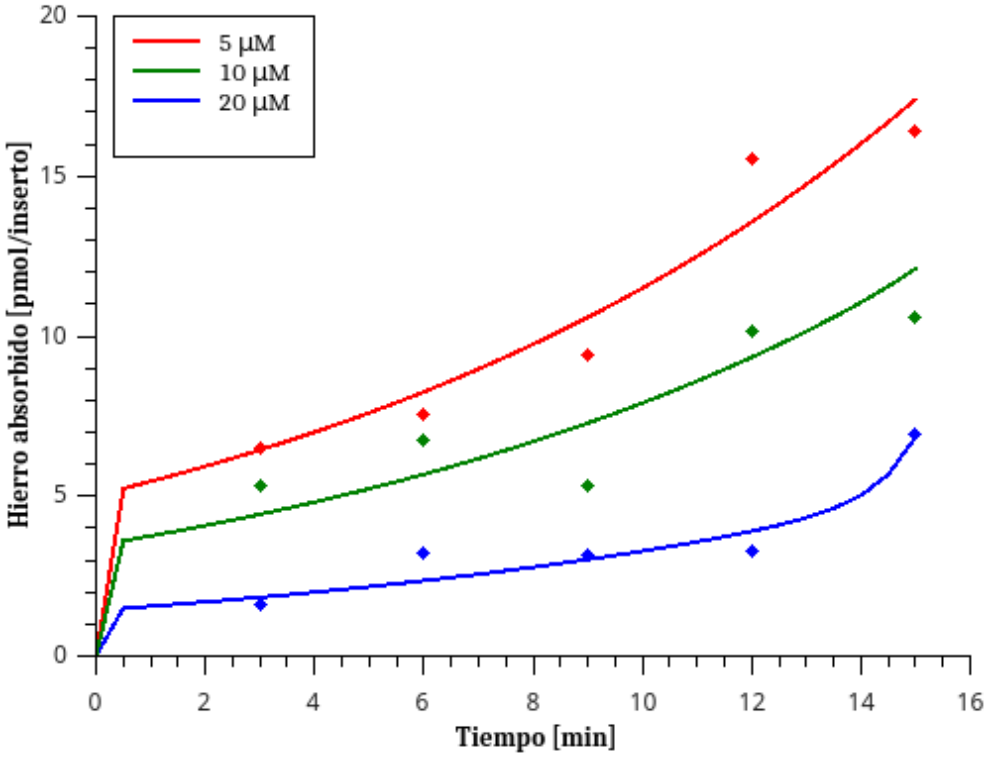

Figura 5.4: Simulación de datos para mejor modelo ABCP con error de generalización.

Sobre el error de generalización del modelo, este es $MSE_{jk} = 0.86$ el cual es menor que su caso clásico, lo que demuestra que tiene la mayor capacidad de generalización, por lo que esta etapa y tal como se creía, ayuda a disminuir el sobreajuste que se producía por el ajuste de parámetros. Sobre estos últimos, mostrados en la Tabla 5.3, y como los pseudo-parámetros son similares, significa que el modelo generado es el mejor que se puede encontrar con estos parámetros.

Analizando la ecuación 5.3 en detalle, la función encontrada es creciente lo cual es esperable al observar los datos experimentales y confirma que la absorción aumenta a medida que transcurre el tiempo, además la forma de la curva sin oscilaciones sugiere que este es un proceso creciente en el tiempo, por lo tanto los datos experimentales encontrados pueden no describir el proceso de buena forma al existir disminuciones del hierro absorbido como se muestra en $10\ y\ 20\ \mu M$.

Al igual que en el modelo clásico, está dominada por un logaritmo como término principal, lo que da indicios que la fenomenología puede seguir este tipo de curva. De la curva obtenida, se puede observar un crecimiento abrupto hasta $0.5\ [min]$, para luego suavizar la curva en su comportamiento creciente. Este fenómeno igual se observa en la versión clásica con un cambio de pendiente, preguntándose si corresponde a un aspecto fenomenológico como puede ser el bloqueo mucosal, lo cual es difícil ya que se ha reportado que ocurre después de los $15\ [min]$ en el proceso, o un aspecto matemático de la función lo que es más probable ya que este algoritmo trabaja con funciones protegidas ($\log div\ exp$) para evitar divergencias, entonces es probable que a raíz de esto se presenten estos comportamientos entre 0 y 1 sobre todo con la aparición de la función logaritmo, lo cual corrobora lo anterior ya que si el tiempo es 0, esto debería no corresponde a un número real, lo cual no es observado en la curva.

Del análisis del desempeño del algoritmo, este es superior al del algoritmo clásico en términos estadísticos, lo que justifica la incorporación de las etapas de refinamiento. Respecto a lo anterior, de los 25 modelos, el de peor desempeño obtuvo un $R^2 = 0.83$ y $MSE_{jk} = 1.4$, el cual es similar al mejor modelo obtenido para el caso clásico e indica la superioridad de este algoritmo, de igual forma solo un modelo no cumplió con la restricción biológica impuesta, aunque este tenía el mejor desempeño estadístico que el resto, tuvo que ser descartado. Es importante destacar que de los 25 modelos, la mayoría se encontraba duplicado no encontrándose modelos únicos, e incluso el mejor modelo se repitió 3 veces lo que equivale a un 12% del total, por lo que se analizó si existía una convergencia o si se repetían estructuras entre los modelos.

Del primer análisis se buscó entre las subpoblaciones (los 500 individuos en la última generación del mejor modelo) si estos eran similares al mejor individuo, lo que no fue claro, principalmente por la diferencia que existen entre los parámetros. Del segundo se comparó la estructura del mejor individuo de los 25 modelos (sin contar los repetidos), y se observó que no existe una estructura predominante para todos pero sí existen modelos repetidos que se puede considerar una estructura dominante en sí. Lo anterior demuestra que hay una convergencia hacia soluciones particulares, y sería interesante realizar en el futuro más repeticiones para determinar si los modelos repetidos actuales son los únicos que entrega el algoritmo o pueden existir más. El problema de realizar esto es el tiempo de cómputo del algoritmo por lo que no se pudo hacer.

Otro aspecto relevante del algoritmo, es el aumento del tiempo de ejecución, donde la ejecución es de alrededor de 20 horas por modelo en comparación a los 10 minutos en el caso clásico, de igual forma, es menor que para el caso de GP, el cual es de más de 1 día. Este aumento justifica la obtención de solamente 25 modelos, pero dependiendo de los recursos que se quieran destinar a resolver un problema, el tiempo puede ser un factor importante, por lo que se debe elegir bien que algoritmo utilizar (clásico o EG) ya que el primero también entrega buenos resultados.

En ABC, al agregar las etapas de refinamiento, se obtienen mejores datos estadísticos, donde el ajuste de parámetros permite eliminar la arbitrariedad de la selección de los parámetros y el utilizar el error de generalización como función de fitness elimina el sobreajuste que puede tener la etapa anterior encontrando así modelos con una mayor capacidad de generalización. Al realizar estas etapas, se encontraron mejores modelos sin perder características propias del algoritmo, por lo que el objetivo de esta incorporación se cumple.

Tabla 5.4: Estadísticos modelo ABCP con ajuste.

Parámetros	**Pseudo-parámetros**
-10.04	-10.04
1183.71	1183.7
9.004	9.004
-0.046	-0.05

5.4.2. CASO DAP ERROR DE GENERALIZACIÓN:

Al igual que en el caso de ABCP, se realizaron modificaciones al algoritmo clásico para implementar la etapa de ajuste de parámetros con error de generalización. Utilizando los mismos parámetros, sumado al cambio del número de modelos generados, se seleccionó el mejor modelo, el cual es la función representada en la ecuación 5.4 y la simulación de los datos se muestran en la Figura 5.5.

$$[Fe_{in}(C_0, t)] = (C_0 t - t)^{\cos(1000.1)} \tag{5.4}$$

El modelo presenta un $R^2 = 0.80$ el cual es superior al mostrado por el mejor modelo utilizando el algoritmo clásico e iguala el criterio utilizado para ABC, por lo cual este modelo si es capaz de representar de buena forma los datos. Al igual que en los casos anteriores, la ecuación muestra componentes no lineales, lo cual es lo esperable dado los datos experimentales que se tienen.

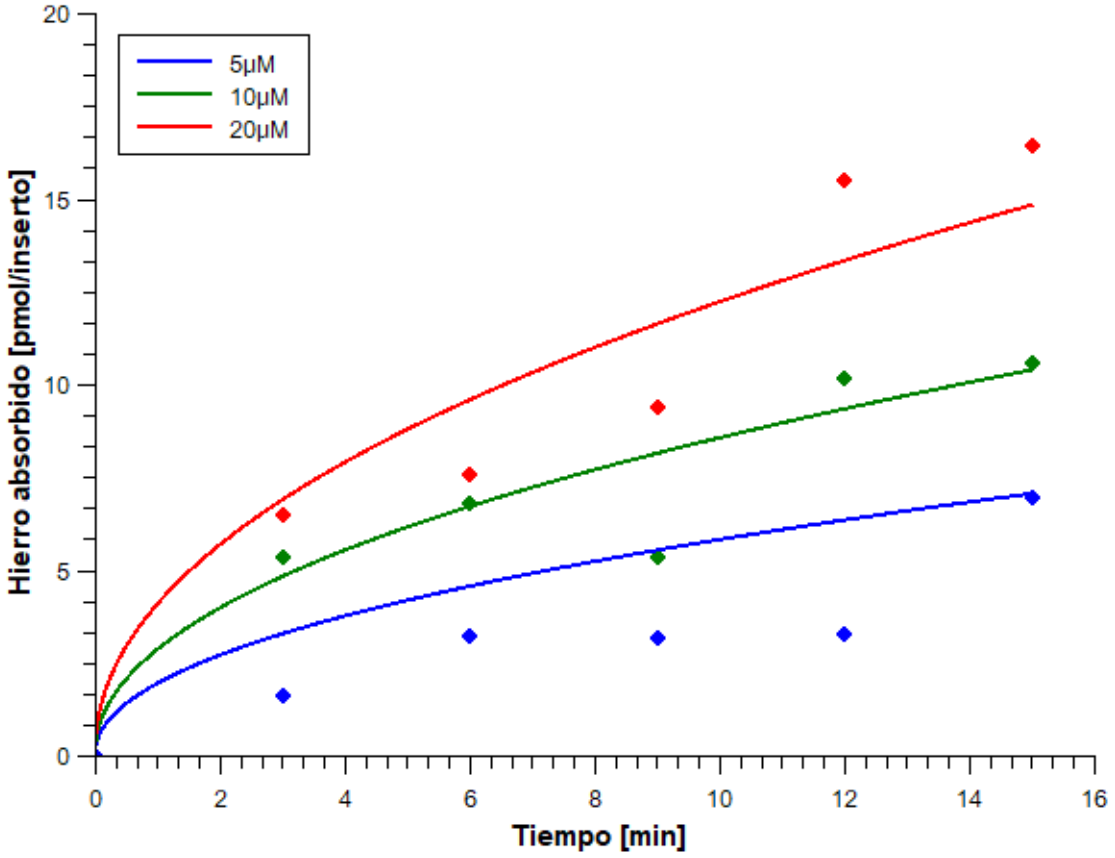

Figura 5.5: Simulación de datos de mejor modelo DAP con error de generalización.

Como se encuentra mencionado anteriormente, las constantes, o funciones que contengan constantes no se pueden simplificar, por lo cual, si bien el parámetro optimizado en la ecuación es 1000.1, perfectamente podría haber sido el valor de $cos(1000.1) = 0.4714$, el cual al utilizarlo, no cambia en nada el modelo obtenido. Nuevamente este análisis sugiere que dentro del algoritmo exista una etapa de ajuste, principalmente para obtener funciones más pequeñas y diferentes parámetros dentro de este, para generar una mayor diversidad en las soluciones.

De los resultados de la capacidad de generalización de este modelo, el error de generalización es de $MSE_{jk} = 1.78$, valor mucho menor al encontrado en el algoritmo clásico, lo cual indica que el modelo posee una mayor capacidad de generalización sobre nuevos datos experimentales. Lo anterior indica que utilizar este método como función de fitness ayuda a minimizar el sobreajuste del ajuste de parámetros, lo cual era el objetivo de esta etapa, además el valor del pseudo-parámetro de este modelo es igual al parámetro ajustado, lo que indica que este valor es el óptimo para la función encontrada.

Al analizar la ecuación 5.4 y la Figura 5.5 se tiene una función con pocos términos y una curva creciente siendo positiva en todo tiempo (todos los modelos mostraron este comportamiento). Lo anterior elimina la posibilidad de enfrentar un fenómeno biológico en el primer minuto, entonces lo que ocurre en los modelos con ABCP se debe netamente a aspectos matemáticos de las funciones protegidas. En este escenario, la ecuación 5.4 presenta solamente las variables y una constante sin encontrarse este tipo de funciones. Además no posee problemas en $t = 0$ y el comportamiento en tiempos superiores es siempre creciente sin oscilaciones ni cambios de pendiente.

La forma de la curva afirma el hecho de que la absorción es creciente con el tiempo, siendo de una forma casi lineal en este caso, con lo cual queda bastante claro que no existen disminuciones en la concentración como se muestra con los datos experimentales, sugiriendo de esta forma realizar nuevamente el experimento y probar con nuevos datos.

Del análisis del algoritmo, este cumple con su objetivo de generar soluciones con pocos términos, las cuales se pueden adaptar mejor a fenomenologías conocidas, además, con la implementación de las dos etapas de refinamiento, se lograron mejores modelos más capaces de representar los datos y también con un menor error de generalización, por lo cual, esta etapa cumple con el objetivo planteado en un comienzo. Otro aspecto importante al analizar los modelos generados, es que un 64% de los modelos mostraron un $R^2 > 0.75$ lo que indica que la refinación mejora sustancialmente el desempeño de este algoritmo en comparación con su versión clásica..

Por otra parte, el tiempo de ejecución de un modelo es mucho menor que sus contrapartes GP y ABC, el cual es de aproximadamente 1 hora, en comparación a los días que demora GP y ABC con estos ajustes. Lo anterior es debido al tamaño pequeño de las estructuras generadas y a las pocas operaciones realizadas sobre éstas. Como la comparación que se busca es manteniendo la mayoría de los parámetros iguales entre los algoritmos, no se realizaron modificaciones para aprovechar el tiempo “extra” que entrega este algoritmo al tener menor tiempo de ejecución.

Lo que se realizó fue aumentar el número de modelos de forma paralela, es decir obtener otros 25 que no participen en la comparación para poder tener una muestra mayor, la cual mostró la misma tendencia que los primeros 25, pero obteniéndose 2 modelos repetidos, por lo que se propone la misma metodología que para ABCP EG en un futuro para determinar la convergencia del algoritmo.

Otra forma de aprovechar el tiempo de es entregar más recursos a la etapa de ajuste de parámetros para encontrar parámetros aún más adecuados para este sistema.

En DAP, al agregar la etapa de ajuste de parámetros, se obtiene una mayor capacidad de representar los datos experimentales, junto con eliminar la incertidumbre generada con la elección arbitraria de parámetros. Además al utilizar la metodología de error de generalización, se mejora la capacidad de los modelos de estimar nuevos datos experimentales, sin perder las características propias del algoritmo o de la etapa anterior de ajuste de parámetros. Lo anterior indica que para el caso de DAP, ambas etapas de refinamiento cumplen con sus objetivos generando mejores modelos.

5.5. COMPARACIÓN DE ALGORITMOS

En la Tabla 5.5 se pueden apreciar los resultados relevantes de los algoritmos para el caso de los algoritmos clásicos. De esto se puede apreciar que GP y ABC tienen desempeños muy similares, esto es porque los algoritmos tienen una base parecida, es decir, se parte con una población inicial, y ésta va cambiando en base a los mejores individuos (comparten también la misma función de fitness), además tienen el mismo problema del crecimiento desmedido de la estructura de árboles, y por lo tanto de las funciones generadas, lo que si bien produce una mejora en el fitness, ésta es pequeña en comparación con el crecimiento de las funciones. Aun así, ABC es ligeramente superior en los estadísticos R^2 y MSE_{jk} a GP, lo que lo posiciona como el de mejor desempeño de los 3, ya que DAP presenta el peor desempeño estadístico. Es importante destacar que los resultados mostrados corresponden solo al problema particular de la absorción de hierro luego de la selección de los mejores modelos a partir de las restricciones impuestas a estos comparando el mejor modelo para los 3. Por otra parte, el tiempo de ejecución viene a ser un promedio de los tiempos de ejecución de cada algoritmo.

Tabla 5.5: Comparación desempeño entre algoritmos en versión clásica.

Algoritmo	R^2	MSE_{jk}	**Tiempo Ejecución**
GP*	0.84	1.45	10 minutos
ABCP	0.85	1.2	8 minutos
DAP	0.75	2	10 minutos

Es importante notar el bajo desempeño de DAP, lo anterior puede ser explicado por las características de los pasos del algoritmo, donde existe una etapa de eliminación de nodos, la cual elimina los nodos (funciones o constantes) que no son utilizados. Lo anterior evita el problema de sobre-crecimiento del árbol que ocurre en GP o ABCP, pero al recordar que esto aporta al fitness, el eliminar este problema repercute también en el fitness y por lo tanto en el desempeño del algoritmo. Es importante destacar que si se limita etapa de eliminación de nodos para que sea menos frecuente, no proporciona ninguna mejora ya que el nodo a eliminar no se está utilizando para encontrar nuevas rutas por lo tanto el limitar esta etapa solo retrasa su eliminación.

Otro aspecto que puede afectar el desempeño, es el mostrado al ejecutar las funciones de prueba, donde en funciones que requieran logaritmos o funciones trigonométricas, su desempeño comienza a disminuir ya que el paso que construye explora una solución con estos término los cuales tienen aridad mayor que 0, lo que se traduce en que se deba realizar un paso siguiente para llegar al terminal (que a su vez puede utilizar otra función con aridad mayor que 0) teniendo mayores caminos que visitar.

En términos de tiempos de ejecución, DAP posee el menor, dado principalmente por el tamaño de sus estructuras como se discutió en la sección anterior. Ahora, esta disminución del tiempo no es tan relevante en la versión clásica, ya que la diferencia es de menos de 5 minutos para los dos algoritmos, pero sí es relevante en la versión con error de generalización, donde la diferencia es de muchas horas con respecto a los otros 2 algoritmos. En relación a esto, se podrían agregar etapas extras de refinamiento para aumentar el desempeño de este algoritmo.

Con las modificaciones de ajuste de parámetros y Jackknife, los desempeños de todos los algoritmos aumentaron como es mostrado en la Tabla 5.6, siendo el de mejor resultados ABCP. Si se analiza las funciones encontradas en esta etapa, la de forma más simple es la de DAP, la cual es por las características de este algoritmo, pero como el desempeño es menor, se le puede disminuir su relevancia en comparación a las encontradas por GP o ABCP, que pueden señalar fenómenos más complejos que puedan estar ocurriendo.

Tabla 5.6: Desempeño de los algoritmos luego de las etapas de refinamiento.

Algoritmo	R^2	MSE_{jk}	**Tiempo Ejecución**
GP*	0.85	1.39	2 días
ABCP	0.86	0.86	20 horas
DAP	0.8	1.78	1 hora

Por lo anterior, se podría buscar una forma de acoplar estos algoritmos, con el fin de encontrar funciones con menos términos, pero sin sacrificar el desempeño encontrado por ABC o GP. Dada la similitud entre los algoritmos anteriores, la combinación sugerida es DAP-ABCP o DAP-GP, y dependiendo de sus resultados, sumarle las etapas de refinamiento realizadas para analizarlos y comparar con lo ya obtenido.

Una forma de realizar este acople puede ser construir la población inicial de GP o ABCP con DAP para luego comenzar a modificarla según el algoritmo que se use. Otra forma dependiendo del tiempo que se tenga, es obtener la población luego de la ejecución de DAP y utilizarla en los algoritmos anteriores, con la ventaja de comenzar con estructuras con pocos términos con lo cual la solución no va a converger tan rápidamente dando más espacio a las operaciones sobre las soluciones.

6. CONCLUSIONES

El objetivo del trabajo fue analizar el transporte de hierro en células Caco-2 mediante la modelación matemática, donde se consideró el escenario de absorción apical de hierro en el tiempo a diferentes concentraciones. La modelación con los distintos algoritmos sirvió para entregar una aproximación a la fenomenología del proceso la cual describe un aumento de la absorción en el tiempo de forma creciente sin cambios de pendiente (no sale hierro del sistema) ni oscilaciones, pero aún faltan elementos para describirlo completamente.

La elección de los algoritmos fue en base a información previa, y a comparaciones de desempeño teórico en relación con la programación genética, así los seleccionados fueron Artificial Bee Colony Programming y Dynamic Ant Programming.

Los algoritmos anteriores debieron ser implementados completamente al no encontrarse implementaciones en ningún tipo de lenguaje computacional. Al implementarlos se comprobó si fueron correctamente programados, esto se realizó con funciones de prueba utilizadas en la literatura, ante lo cual la implementación de los 2 algoritmos fue exitosa.

Se generaron 2 modelos del problema para cada algoritmo, el primero corresponde a la versión clásica de los algoritmos sin ninguna modificación y la otra versión con error de generalización, lo cual corresponde a una modificación realizada para aumentar la capacidad de generalización de los modelos.

El modelo obtenido para la versión clásica de ABCP y DAP cumple con la mayoría de las restricciones biológicas del sistema, y también logra representar adecuadamente los resultados experimentales.

Los modelos generados con la metodología de error de generalización, ayudaron a mejorar el desempeño estadístico del modelo además de aumentar su capacidad de generalización. Por otro lado ayudó a aumentar el desempeño general del algoritmo en este problema en particular para DAP aunque fue el de peor rendimiento en ambas versiones.

A la hora de comparar el desempeño de ABCP, GP y DAP en este problema en particular se tiene que tanto en la versión clásica como en la versión EG, ABCP es el que entrega mejores resultados estadísticos. Es importante destacar el tiempo como elemento de comparación donde DAP en la versión de EG presentó el menor tiempo de ejecución, casi 20 veces menos que el resto de los algoritmos.

Por último es importante destacar que la metodología y algoritmos pueden ser utilizados para distintos problemas de investigación, cuando se requieran de modelos empíricos para procesos desconocidos, teniendo la opción de utilizar los distintos algoritmos ya probados en un problema real.

BIBLIOGRAFÍA

[1]Frazer, DM. y Anderson, GJ. Iron Imports. I. Intestinal iron absorption and its regulation. Am J Physiol Gastrointest Liver Physiol 289(4):G631-5, 2005.

[2] Theil, Elizabeth C. Iron homeostasis and nutritional iron deficiency. The Journal of Nutrition, pp.724S-727S, 2011.

[3] Moss, T. y Morgan, EH. The metabolism of neuronal iron and its pathogenic role in neurological disease: review. Ann N Y Acad Sci 1012:14-26, 2004.

[4] Oates, Morgan, Phillips S. Mechanisms and regulation of intestianal iron absorption.3, Crawley : Blood Cells, Molecules, and Diseases, 2002, Vol. 29.

[5] Beinert, H., Kiley, P.J. Fe-S proteins in sensing and regulatory functions.

Current opinion in chemical biology, 3(2): 152-7, abr. 1999.

[6] Johnson, D.C., Dean, D.R., Smith, A.D., Johnson, M.K. Structure, function, and formation of biological iron-sulfur clusters. Annual review of biochemistry , 74: 247-81, ene. 2005.

[7] Collins, J.F., Anderson, G.J. Chapter 71. Molecular Mechanisms of Intestinal Iron Transport. First edit edón. Elsevier Inc., 2012, 1921-1948 págs.

[8] World Health Organization. Assessing the Iron Status of population. Inf. téc, 2004.

[9] Mackenzie, B., Garrick, M.D. Iron Imports. II. Iron uptake at the apical membrane in the intestine. American journal of physiology. Gastrointestinal and liver physiology,289, 2005.

[10] Sharp, P.A. Intestinal iron absorption: regulation by dietary & systemic factors. International journal for vitamin and nutrition research., 80(4-5): 231-42, oct. 2010.

[11] Anderson, G., McLaren, G. Iron Physiology and Pathophysiology in Humans.

[12] Theil, E.C. Iron Homeostasis and Nutritional Iron Deficiency. Online, 2011.

[13] Breuer, W., Shvartsman, M., Cabantchik, Z.I. Intracellular labile iron. The international journal of biochemistry & cell biology, 40(3): 350-4, ene. 2008.

[14] Mackenzie, B., Ujwal, M.L., Chang, M.H.H., Romero, M.F., Hediger, M.a. Divalent metal-ion transporter DMT1 mediates both H+ -coupled Fe2+ transport and uncoupled fluxes. Pflugers Archiv European Journal of Physiology, 451(4): 544-558, ene. 2006.

[15]Garrick MD. y Garrick LM. Cellular iron transport. Biochimica et biophysica acta 1790(5):309-25, 2009.

[16] Fuqua, B.K., Vulpe, C.D., Anderson, G.J. Intestinal iron absorption. J Trace Elem Med Biol, 26(2-3): 115_9, jun. 2012.

[17] CONRAD, M.E., CROSBY, W.H. INTESTINAL MUCOSAL MECHANISMS CONTROLLING IRON ABSORPTION. Blood, 22(4): 406_15, oct. 1963.

[18] Frazer, D.M., Wilkins, S.J., Becker, E.M., Murphy, T.L., Vulpe, C.D., McKie, a.T., Anderson, G.J. A rapid decrease in the expression of DMT1 and Dcytb but not Ireg1 or hephaestin explains the mucosal block phenomenon of iron absorption. Gut, 52: 340-346, 2003.

[19] STEWART, W.B., YUILE, C.L., CLAIBORNE, H.A., SNOWMAN, R.T., WHIPPLE, G.H. Radioiron absorption in anemic dogs; _uctuations in the mucosal block and evidence for a gradient of absorption in the gastrointestinal tract. The Journal of experimental medicine, 92(4): 375_82, oct. 1950.

[20] Nemeth, E., Ganz, T. Regulation of iron metabolism by hepcidin. Annual review of nutrition, 26: 323-42, ene. 2006.

[21] Nicolas, G., Chauvet, C., Viatte, L., Danan, J.L., Bigard, X., Devaux, I., Beaumont, C., Kahn, A., Vaulont, S. The gene encoding the iron regulatory peptide hepcidin is regulated by anemia, hypoxia, and in_ammation. The Journal of clinical investigation, 110(7): 1037-44, oct. 2002.

[22] Zoller, H., Theurl, I., Koch, R., Kaser, A., Weiss, G. Mechanisms of iron mediated regulation of the duodenal iron transporters divalent metal transporter 1 and ferroportin 1. Blood cells, molecules & diseases, 29: 488-497, 2002.

[23] Bender, E.A. An Introduction to Mathematical Modeling. 2012.

[24] Man, N.V.M., Ph, D. Mathematical Modeling and Simulation. 2010.

[25] Draper, N.R., Smith, H., Pownell, E. Applied regression analysis, tomo 3. Wiley New York, 1966.

[26] Schmidt1 M, Lipson H, Distilling Free-Form Natural Laws from Experimental Data. Science, Vol. 234 pag 81-85, 2009.

[27] Karaboga D, Ozturk C, Artificial bee colony programming for symbolic regression, Information Sciences 209 (2012) 1–15.

[28] R. Poli, W.B. Langdon, O. Holland, Extending particle swarm optimization via genetic programming, in: Proceedings of the Eighth European Conference (EuroGP), Lausanne, Switzerland, 2005

[29] B.M. Cerny, P.C. Nelson, C. Zhou, Using differential evolution for symbolic regression and numerical constant creation, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO08, 2008, pp. 1195-1202.

[30] B. Mariusz, Ant colony programming: application of ant colony system to function approximation, in: Raymond Chiong (Ed.), Intelligent Systems for Automated Learning and Adaptation: Emerging Trends and Applications, 2010, pp. 248–272.

[31] Z. Lim PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems, Expert Systems With Applications, 42 (2015) 1883-1897.

[32] S.Das,Recent advances in differential evolution –An updated survey. Swarm and Evolutionary Computation. 27 (2016) 1-30.

[33] Koza, J.R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. 1992.

[34] Mcphee, N.F., Poli, R., Langdon, W.B. A Field Guide to Genetic Programming. Lulu.com, 2008, 233 págs.

[35] Yang, X.S., Algorithms, G. Nature-Inspired Optimization Algorithms. pág. 300, 2014.

[36] Holland, J.H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, 1975, 183 págs.

[37] Mansour, R.F. Using Genetic Algorithm for Identification of Diabetic Retinal Exudates in Digital Color Images. Journal of Intelligent Learning Systems and Applications, 04(03): 188-198, ago. 2012.

[38] Gestal, M., Rivero, D., Rabuñal, J.R.J., Dorado, J., Pazos, A. Introducción a los algoritmos genéticos y la programación genética. Universidade da Coruña, 2010, 76 págs.

[39] Shirakawa, S., Ogino, S., Nagao, T., Dynamic ant programming for automatic construction of programs. Transaction on electrical and electronic engineering, Volume 3, Issue 5, September 2008, Pages 540–548.

[40] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B: Cybern. 26 (1) (1996) 29–41.

[41] M. Dorigo , G. Di Caro et T. Stützle, Special issue on "Ant Algorithms", Future Generation Computer Systems, volume 16, numéro 8, 2000

[42] BRAGA, L.P.V., Braga, L.P.V., Carvajal, L.I.O.V.e.S.S.R., VALENCIA, L.I.O., CARVAJAL, S.S.R. Introducción a la Minería de Datos. Editora E-papers, 218 págs.

[43] Efron, B. The Jackknife, the Bootstrap and Other Resampling Plans. 1982, 103 págs.

[44] Salkind, N. Encyclopedia of Research Design, Volume 1. SAGE Publications, 2010, 1719 págs.

[45] Arahal, M.R., Soria, M.B., Díaz, F.R. Técnicas de predicción con aplicaciones en ingeniería. Universidad de Sevilla, 2006, 340 págs.

[46] Natoli, M., Leoni, B.D., D'Agnano, I., Zucco, F., Felsani, A. Good Caco-2 cell culture practices. Toxicology in Vitro, 26(8): 1243-1246, 2012.

[47] Frey, A., Giannasca, K.T., Weltzin, R., Giannasca, P.J., Reggio, H., Lencer, W.I., Neutra, M.R. Role of the glycocalyx in regulating access of microparticles to apical plasma membranes of intestinal epithelial cells: implications for microbial attachment and oral vaccine targeting. The Journal of experimental medicine, 184(3): 1045-59, sep. 1996.

[48] Colins Rodríguez, Andrea Justina. Determinación experimental y modelación matemática de los flujos de transporte de hierro en células Caco-2(Ingeniero Civil en Biotecnología). Santiago, Chile.

[49] Alvarez-Hernandez, X., Nichols, G.M., Glass, J. Caco-2 cell line: a system for studying intestinal iron transport across epithelial cell monolayers. Biochimica et biophysica acta, 1070: 205_8, 1991.

[50] Moré, J. The Levenberg-Marquardt algorithm: Implementation and theory. En: G.A. Watson (Ed.) Numerical Analysis SE - 10, Lecture Notes in Mathematics, tomo 630, págs. 105-116. Springer Berlin Heidelberg, 1978.

[51] Karaboga D, Gorkemli B. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review, 21-57, 2014.

ANEXOS

Anexo A. Determinación experimental de la absorción de hierro por A. Colins

Cultivo celular. Las células Caco-2 fueron cultivadas en insertos bicamerales de $12\,mm$ de diámetro, sobre placas de 12 pocillos, que permanecieron en incubadora húmeda de CO_2 al 5%. Se utilizó medio Eagle modificado por Dulbecco (DMEM), suplementado con $3{,}7\ gr/lt$ de $NaOHCO3$, 10% suero FBS y 1% solución antibiótico y antimicótico, y fue esterilizado con un filtro de $0.22\ \mu m$. Las células fueron sembradas a una densidad de $30.000\ cel/cm^2$, para monitorear su crecimiento se midió la resistencia eléctrica transepitelial (TEER) mediante un multímetro cada 2 días, luego de esto se realizaba un lavado con buffer fosfato salino (PBS) y se cambiaba el medio de cultivo. Se consideró que la monocapa estaba formada cuando los registros de la TEER alcanzaron los $240\ \Omega/cm2$.

Determinación de la absorción apical de hierro. Una vez formada la monocapa, se cambió el medio de cultivo por DMEM $2\,\%$ FBS esto con la finalidad de disminuir la replicación celular y la cantidad de transferrina disponible en el medio de cultivo [137a]. Se incubó nuevamente el cultivo por 24 horas. Se preparó una solución de ácido ascórbico $100mM$ y un stock de cloruro férrico a $100\,\mu M$ donde la proporción $^{55}FeCl_3$: $^{56}FeCl_3$ es igual a $1:4$ y se mantiene una proporción $1:20$ entre $^{56}FeCl_3$:Ascorbato. Con esta solución se prepararon los medios que se colocaron en el medio apical en los ensayos a 5, 10 y $20\,\mu M$. De estos se retiraron $10\,\mu L$ para obtener el factor de conversión entre la radiación y concentración molar.

Para realizar los ensayos se utilizó DMEM sin suero en el medio basolateral en cada inserto, se cambió el medio apical del cultivo por los medios que contenían $^{56}FeCl_3$ y se incubó por el tiempo necesario ($3{,}6{,}9{,}12\,y\,15$ min) a $37\,°C$. Una vez terminado el tiempo de incubación se colocó inmediatamente la placa en hielo con la finalidad de detener la reacción y se lavaron las secciones apical y basolateral con PBS frío 3 veces. Luego se cortaron los filtros y se colocaron en $1\,mL$ de líquido de centelleo. Por otro lado, se retiraron $800\,\mu L$ de los medios basolaterales y se les agregó $3\,mL$ de líquido de centelleo. Finalmente se midió la cantidad de radiación emitida por cada muestra durante 5 minutos. Se consideró que la cantidad de hierro que traspasa la membrana apical es igual a la cantidad de hierro en el interior de las células más la cantidad presente en el medio basolateral.

Anexo B. Determinación de condiciones para ajuste de parámetros y jackknife por A. Colins

Como el ajuste de parámetros es un problema de la misma complejidad que el de programación genética, fue necesario acotar los recursos destinados a resolver esta etapa, para priorizar el algoritmo genético. Los criterios para esta etapa fueron:

- Escoger el algoritmo de optimización utilizado en el ajuste de parámetros.
- Definir el criterio de término del algoritmo escogido.

Existen diferentes tipos de algoritmos para realizar el ajuste de parámetros, los cuales poseen diferentes características y son mejores o peores para determinados problemas. Como en este caso no se puede saber con anterioridad el tipo de función que el algoritmo de programación genética propondrá, era necesario escoger un algoritmo lo más general posible. De la misma manera, al no haber restricciones que acoten el espacio de soluciones, el algoritmo no debía requerir restricciones para funcionar adecuadamente.

Tomando en cuenta lo anterior, el algoritmo escogido fue Levenberg-Marquardt [42], donde los parámetros utilizados como punto inicial son los escogidos por el algoritmo genético al crear el modelo que se desea evaluar.

Luego se debía escoger el criterio de detención del algoritmo seleccionado. Este punto es de especial importancia, pues permite controlar el tiempo de convergencia y los recursos adicionales que esta nueva etapa requeriría. Para realizar lo anterior existen varias alternativas, como criterios de detención por tolerancia en el cambio de la función objetivo o los parámetros, criterios de detención por cantidad de llamadas a la función y criterios de detención por cantidad de iteraciones. Dado que en esta etapa se deben ahorrar recursos, se prefirió optar por el criterio más simple, el cual es la cantidad de iteraciones, donde según la bibliografía un bajo número de iteraciones asegura la convergencia, por lo tanto el número será de 10 [50].

La etapa de error de generalización viene con el fin de que los modelos obtenidos tuvieran una mayor capacidad de generalización, sin perder la forma de representar adecuadamente los flujos de hierro estudiados dentro del algoritmo genético. Para esta última etapa, se reemplazó la determinación de función de fitness del algoritmo original de la ecuación 3.3, por la medición del error de generalización MSE_{jk}.

Al igual que en el caso del ajuste de parámetros, este problema de optimización es equivalente al de programación genética, por lo tanto se debe limitar los recursos

utilizados en esta etapa ya que la cantidad de optimizaciones por modelos es igual a la cantidad de datos experimentales utilizados, es decir, un total de 55 optimizaciones. Dado lo anterior, se establecieron criterios para reforzar las medidas de ahorro de recursos y tiempo ante el gran aumento de estos en comparación con los utilizados en programación genética, los cuales se muestran a continuación:

- **Disminuir el número de generaciones del algoritmo:** El número de generaciones utilizado para el algoritmo genético clásico es igual a 100. Este es un valor alto, ya que comúnmente se utiliza entre 10 y 50 debido a que los mayores cambios se producen en las primeras generaciones. Sin embargo, el número fue elegido para refinar la búsqueda del mejor modelo. En el caso de la versión modificada, se utilizó un máximo de 50 generaciones, ya que el algoritmo posee 2 etapas que refinan los modelos en la etapa de evaluación.
- **Disminuir el número de modelos generados por el algoritmo:** Un punto muy importante a considerar, es el aumento en el tiempo que se utilizaba para ejecutar cada vez el algoritmo, es decir, el tiempo que era necesario para crear un modelo del conjunto a evaluar. En el caso del algoritmo clásico, este parámetro no era un problema, ya que el tiempo necesario para ejecutar las 100 generaciones era del orden de $10\ [min]$**.** Luego, con las etapas agregadas, el algoritmo tardaba cerca de 100 veces más que la versión clásica utilizando solo 50 generaciones, valor cercano a 2 días. A partir de esto, se consideró disminuir a 20 los modelos seleccionables. Si bien con esta medida se tiene una menor cantidad de modelos elegibles, se esperaba que los modelos obtenidos fueran más adecuados que los obtenidos con las otras metodologías, lo que compensaría este hecho.

Anexo C. Detalle de la implementación de los algoritmos en Matlab

- **ABCP**

ABCP fue el primero que se programó, para esto y dada la similitud de algunos métodos con GP, se buscó en la toolbox de Matlab GPlab si existían funciones o métodos que se podían utilizar para programar el algoritmo y así no partir desde 0. Al analizar en detalle la herramienta, se encontró que muchos de los métodos podían ser utilizados directa o indirectamente (modificando parte de lo ya implementado). Se comenzó la programación siguiendo la secuencia mostrada en el Algoritmo 2, donde los métodos de generar una población inicial y evaluarla fueron implementados directamente de la toolbox. Luego de esto, se debió implementar la etapa del mecanismo de intercambio de información, la cual es la parte más importante del algoritmo ya que se utiliza en dos partes de él.

Para lo anterior, se utilizaron funciones ya implementadas en GPlab sobre la manipulación de los nodos (recordando que esta etapa es un intercambio de nodos entre 2 individuos de la población) y se programó el método considerando sus

características, como por ejemplo agregar el valor del límite cuando el individuo no cambia. Para la evaluación que sigue se utilizó la misma función implementada en la población inicial. Luego de eso, como a los individuos se le calcula una probabilidad, se utilizó el método de ruleta (el cual también se encuentra en GPlab) para seleccionar el individuo que se modificará cuando se encuentre en la siguiente etapa del algoritmo (etapa de “evaluadoras”).

Finalmente, se implementa la etapa donde se utiliza el parámetro límite, utilizado para determinar si el individuo debe ser reemplazado por uno nuevo en el caso que no haya sido modificado en sucesivas generaciones. Si se entra en esta etapa, se utiliza el método grow de GPlab para generar el individuo. Es importante destacar que al algoritmo se le agrega un “bolsillo” el cual es una estructura que guarda la mejor solución encontrada hasta ese momento

- **DAP**

En el caso de DAP, su implementación fue casi desde 0 debido a que sus características difieren mucho de GP o ABCP. A pesar de lo anterior, igual se comenzó buscando en GPlab funciones que pudieran ser implementadas dada la comprensión obtenida de la herramienta por la implementación de ABCP, pero no se encontró ninguna que pueda ser empleada directamente (a excepción del rawfitness, pero esta es utilizada de forma paralela y no es parte del algoritmo), de todas formas, se aprovecharon muchas funciones que tienen relación con la manipulación de la estructura de árbol. La forma que se programó fue siguiendo el Algoritmo 2, donde para comenzar se crea la tabla de feromonas como una matriz para luego continuar con la parte más importe del algoritmo, la creación del paso de cada hormiga, o dicho de otro modo, la creación del árbol para cada individuo en el ciclo.

Para este primer método, se creó una función recursiva, que fuera creando el árbol según las funciones que se encuentren disponibles y eliminando de este grupo las que se fueran utilizando según lo explicado en la sección 3.2.4. Para la evaluación de los individuos y dada la similitud del fitness de GP, se empleó parte de la función rawfitness de GPlab como base para programar la función de fitness de este algoritmo. Luego de la evaluación de todos los individuos, se guarda aquel con el mejor fitness, de forma homóloga al “bolsillo” implementado en ABCP.

Siguiendo con el algoritmo, se programó la etapa de modificación de la tabla de feromonas, junto con la eliminación e inserción de nodos, cada etapa en una función separada. Es importante destacar que para la implementación correcta, se programaron funciones auxiliares para mantener una consistencia entre la estructura de los individuos y la tabla de feromonas, ya que es modificada continuamente (cambia dinámicamente en cada iteración).