DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: DISEÑO E IMPLEMENTACIÓN DE CONTROLADORES DE TEMPERATURA APLICANDO LOS ALGORITMOS DE OPTIMIZACIÓN BIO-INSPIRADOS COLONIA DE ABEJAS ARTIFICIALES Y ENJAMBRE DE PARTÍCULAS.

AUTOR: BISARREA ESPINOSA, FERNANDA MISHEL

DIRECTOR: ING. IBARRA JÁCOME, OSWALDO ALEXANDER

SANGOLQUÍ

2019

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "DISEÑO E IMPLEMENTACIÓN DE CONTROLADORES DE TEMPERATURA APLICANDO LOS ALGORITMOS DE OPTIMIZACIÓN BIO-INSPIRADOS COLONIA DE ABEJAS ARTIFICIALES Y ENJAMBRE DE PARTÍCULAS", fue realizado por la señorita Bisarrea Espinosa, Fernanda Mishel, el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 04 de Julio de 2019

DIRECTOR
ING. IBARRA JÁCOME, OSWALDO ALEXANDER

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Yo, Bisarrea Espinosa, Fernanda Mishel, declaro que el contenido, ideas y criterios del trabajo de titulación: "DISEÑO E IMPLEMENTACIÓN DE CONTROLADORES DE TEMPERATURA APLICANDO LOS ALGORITMOS DE OPTIMIZACIÓN BIO-INSPIRADOS COLONIA DE ABEJAS ARTIFICIALES Y ENJAMBRE DE PARTÍCULAS" es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 04 de Julio de 2019

Bisarrea Espinosa, Fernanda Mishel

C.C: 1004236053

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Yo, Bisarrea Espinosa, Fernanda Mishel autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación:"DISEÑO E IMPLEMENTACIÓN DE CONTROLADORES DE TEMPERATURA APLICANDO LOS ALGORITMOS DE OPTIMIZACIÓN BIO-INSPIRADOS COLONIA DE ABEJAS ARTIFICIALES Y ENJAMBRE DE PARTÍCULAS", en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 04 de Julio de 2019

Bisarrea Espinosa, Fernanda Mishel

C.C: 1004236053

Dedicado a Dios, por ser mi guía,
mi fortaleza y la razón de mi existir.

Y a mis más grandes amores, Marlene, Josué e Ibeth,
por estar siempre a mi lado y llenar mis días de alegría.

Agradecimiento

Quiero agradecer a Dios, por haberme dado la oportunidad de alcanzar esta meta; porque gracias a sus bendiciones, pude conocer buenas personas, vivir experiencias enriquecedoras, y dar por concluido satisfactoriamente este trabajo.

A mis queridos padres, porque gracias a sus esfuerzos y desvelos, hoy soy quien soy. Gracias a mi padre Fernando, por haber confiado y creído en mí, y por siempre desear lo mejor para mi vida. Gracias a mi madre Marlene, por ser un pilar fundamental en mi vida, mi ejemplo de dedicación, valentía y amor incondicional; gracias mamita, por escucharme sin importar la distancia, por reiniciarme con tus abrazos y por enseñarme a luchar por mis sueños.

A mis hermanos, Josué e Ibeth, quienes se han convertido en mi fuerza para seguir avanzando; mis chiquitos, les agradezco de todo corazón por el amor que me brindan, por todas las travesuras que hacen con el fin de hacerme reír, y por todos los momentos que comparten conmigo.

A mis amigos, quienes durante esta etapa universitaria se convirtieron en mi familia. Particularmente, agradezco a mi más grande y mejor amigo, Marcos Yenchong, por estar en las buenas y en las malas, por todo cuanto has hecho y haces por mí; y aunque sé que lo haces sin esperar nada a cambio, espero poder retribuir todo tu cariño de alguna manera; también, agradezco a toda tu familia, por haberme acogido con cariño en su hogar y permitido ser parte de su día a día. Igualmente, agradezco a Yomara Esmeralda, por su confianza, apoyo y comprensión.Y sin duda, agradezco a todos mis amigos de la carrera, porque a pesar que no fue un camino fácil, lo convirtieron en una experiencia grata e inolvidable.

Finalmente quiero expresar mi más sincero agradecimiento al Ing. Alexander Ibarra, principal colaborador durante todo este proceso; quien, con su dirección, conocimiento, enseñanza, motivación y paciencia, permitió el desarrollo de este trabajo. De igual manera, agradezco a la Universidad de las Fuerzas Armas – ESPE y sus docentes, quienes con ahínco me han ayudado a desarrollarme como profesional durante toda la carrera.

Mishel Bisarrea

Índice de Contenidos

Índice de Figuras

Índice de Tablas

RESUMEN

En las últimas décadas, los algoritmos bio-inspirados, desarrollados dentro del campo de la Inteligencia Computacional, se han convertido en un paradigma de la optimización metaheurística, debido a los resultados exitosos presentados tras la aplicación de los mismos. Sin embargo, la mayoría de los estudios investigativos realizados, se han enfocado en el desarrollo de más algoritmos bio-inspirados; ya sea, en la creación de un algoritmo basado en un nuevo modelo, la generación de una versión mejorada o la hibridación de dos o más algoritmos. Es por eso que, en el presente trabajo investigativo se propone el diseño de controladores tipo proporcional, integral, derivativo, a través del uso de los algoritmos de optimización Enjambre de Partículas y Colonia de Abejas Artificiales, y su implementación en el control de un sistema de flujo de aire caliente del módulo PCT-2 disponible en los laboratorios de Electrónica de la Universidad de las Fuerzas Armadas – ESPE. Posteriormente, se complementa la investigación, con un análisis de los resultados obtenidos de la salida del sistema, en comparación del sistema controlado asociado al uso del algoritmos mencionados y en base al uso del Algoritmo Genético. En este trabajo, primero se detalla un análisis del fundamento teórico, tanto del Algoritmo Genético, como de los algoritmos de optimización Enjambre de Partículas y Colonia de Abejas Artificiales; y el estudio de los trabajos previos que han sido usados como base para el desarrollo del mismo. Luego se muestra el desarrollo de la construcción de los códigos de los algoritmos como métodos de sintonización de los controladores, y finalmente el análisis comparativo de los resultados alcanzados en simulación y en las mediciones reales del sistema controlado.

PALABRAS CLAVE:

- **BIO-INSPIRADO**
- **COLONIA DE ABEJAS ARTIFICIALES**
- **OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS**
- **SINTONIZACIÓN**

ABSTRACT

In recent decades, bio-inspired algorithms, developed within the field of Computational Intelligence, have become a paradigm of metaheuristic optimization, due to the successful results presented after their application. However, most of the research studies carried out have focused on the development of more bio-inspired algorithms; either, in the creation of an algorithm based on a new model, the generation of an improved version or the hybridization of two or more algorithms. Therefore, in the present investigative work, the design of proportional, integral, derivative type controllers is proposed, through the use of the optimization algorithms Particle Swarm and Artificial Bee Colony, and its implementation in the control of a hot air flow system of the PCT-2 module available in the electronics laboratories of the Universidad de las Fuerzas Armadas – ESPE. Subsequently, the research is complemented with an analysis of the results obtained from the system output, in comparison to the controlled system associated with the use of the mentioned algorithms and based on the use of the Genetic Algorithm. In this work, first an analysis of the theoretical foundation is detailed, both of the Genetic Algorithm, as well as of the optimization algorithms of Particulate Swarm and Colony of Artificial Bees; and the study of previous works that have been used as a basis for its development. Then the development of the construction of the codes of the algorithms is shown as methods of tuning the controllers, and finally the compa-rative analysis of the results achieved in simulation and in the real measurements of the controlled system.

KEYWORDS:

- **BIO-INSPIRED**
- **ARTIFICIAL BEE COLONY**
- **PARTICLE SWARM OPTIMIZATION**
- **TUNING**

Capítulo 1

Introducción

La aplicación de técnicas bio-inspiradas en el área de control, representa un cambio en la ideología del control clásico. El uso de los diversos algoritmos bio-inspirados ha facilitado el diseño total de los sistemas de control, incluyendo la etapa de sintonización de los diferentes tipos de controladores.

En este capítulo se presenta una introducción al presente trabajo de investigación, donde se detallan los antecedentes, la justificación e importancia, el alcance del proyecto y los objetivos planteados para el mismo.

1.1. Antecedentes

Dentro del sector industrial, los controladores tipo proporcional, integral, derivativo (PID), son los más empleados en la automatización de procesos debido a su simplicidad para el control de temperatura, humedad, nivel, flujo, etc., en diferentes sistemas. Sin embargo, Van Overschee y De Moor (O'Dwyer, 2009), indicaron que el 80 % de los controladores PID que conformaron su investigación, estaban mal sintonizados; conjuntamente informaron que, el 30 % de los controladores PID operan en forma manual, mientras que el otro 30 % de los lazos controlados aumenta la variabilidad a corto plazo del proceso a controlar (debido a una acción integral demasiado fuerte). Estos

datos, como otros que han sido expuestos en (O'Dwyer, 2009), reafirman que la determinación de los parámetros de ajuste del controlador es un problema significativo en varias aplicaciones.

La selección adecuada de los parámetros del sistema (valores de la banda proporcional, la constante de integración y la constante de derivación), determinan un correcto y mejorado funcionamiento del sistema; si algo falla provocaría una inestabilidad en el controlador y, por ende, llegaría a fallar todo el sistema (Arian, 2010). No obstante, para muchos problemas con alto grado de complejidad de la vida real, a menudo basta con encontrar soluciones aproximadas, donde, los algoritmos metaheurísticos son alternativas prometedoras (B. Mishra et al., 2013).

Los algoritmos de optimización o de búsqueda popularizados en los últimos años, son los algoritmos bio-inspirados, y se basan en diferentes procesos presentes en la naturaleza; su principal clasificación son los algoritmos evolutivos y los basados en inteligencia de enjambres (Swarm Intelligence SI). Estos últimos, se originaron a partir del estudio de colonias de insectos (como hormigas, abejas, luciérnagas, etc.) y enjambres de organismos sociales tales como bandadas de aves o bancos de peces; es decir, se basan en la idea de un sistema natural con comportamiento colectivo inteligente, pero compuesto por partes simples de capacidades limitadas. Este comportamiento colectivo inteligente surge de las características de autoorganización, adaptabilidad y robustez natural de estos agentes simples. Es por esto que, a los algoritmos basados en inteligencia de enjambres, se los ha llegado a considerar como métodos de optimización innovadores y eficientes; en consecuencia, se ha abierto un amplio campo de investigación, y a pesar de los numerosos trabajos existentes, los investigadores se mantienen en la exploración de nuevas técnicas de aplicación basados en dichos métodos, dentro del áreas de ingeniería (Agarwal y Mehta, 2014; Kulkarni y Desai, 2016; Kurdi y How, 2016).

Por una parte, los algoritmos genéticos (Genetic Algorithm GA) son técnicas computacionales basadas por la teoría de la evolución de Darwin: "la supervivencia del más apto". Donde, su intención es imitar el proceso de selección natural (selección, cruce y mutación). Las ventajas de este algoritmo son que, consideran simultáneamente muchos puntos en el espacio de búsqueda, trabajan tanto con parámetros discretos como continuos, presentan múltiples soluciones de Pareto para problemas de optimización, es de código abierto y de fácil implementación (Li et al., 2017); es por ello que las investigaciones, aplicaciones y referencias en cuanto a algoritmos genéticos es

muy extensa, en comparación, a los ya mencionados, algoritmos basados en inteligencia de enjambre, tales como, algoritmos de optimización colonia de abejas artificiales (Artificial Bee Colony - ABC) y enjambre de partículas (Particle Swarm Optimization - PSO).

En el estudio realizado por Mousavi-Avval et al., se aplicó algoritmos genéticos multiobjetivo con el fin de optimizar la producción de canola de semillas oleaginosas, donde se tomó en cuenta los parámetros de emisión de energía, económicos y ambientales, con el fin de obtener una producción más limpia; con una muestra de 150 granjas se obtuvo en comparación con la situación actual de producción, el aumento del $24,1\,\%$ en la energía de salida y del $14,2\,\%$ en la relación beneficio/costo, conjuntamente una reducción del $32,1\,\%$ de las emisiones ambientales (Mousavi-Avval et al., 2017).

En otra aplicación de los algoritmos genéticos, (Askarzadeh, 2018) publicó una propuesta de un método de algoritmos genéticos basados en memoria (Memory-based Genetic Algorithm MGA) para optimizar la potencia demandada entre los recursos de energía distribuido en una microred; como resultados presentó que, el MGA buscó de manera más precisa minimizar el costo de producción de energía en el marco de la red inteligente, en comparación con los resultados obtenidos en la implementación de dos variantes de optimización por enjambre de partículas (Particle swarm optimization with inertia weight – PSOw y Particle swarm optimization with constriction factor – PSOcf). El autor también resaltó que, con el método MGA sólo tardó 2 segundos en resolver el problema programado.

Por otra parte, el algoritmo de optimización colonia de abejas artificiales (Artificial Bee Colony - ABC), propuesto por Dervis Karaboga (2005) y basado en el comportamiento inteligente de las abejas al momento de buscar néctar, se caracteriza por presentar una gran robustez, rápida convergencia y una alta flexibilidad (G. Yan y Li, 2011). Fallah et al. (2018)., en su estudio del arte, cita el trabajo realizado por Safamehr; el cual, empleó el algoritmo de optimización ABC y una técnica cuasi estática para reducir el costo de energía y remodelar el perfil de carga con la reducción de la demanda máxima. Tras la simulación realizada, se proyectó una aproximación de la reducción de costos en un $8{,}33\,\%$ y un $11{,}11\,\%$ en la reducción máxima.

Así mismo, Zhang et al. (2017), aplicaron por primera vez una variante del algoritmo ABC a la

optimización experimental de la combustión y las emisiones de los motores diésel con seis variables de entrada. Los investigadores modificaron las fases de abejas empleadas y observadoras para compensar la exploración y la explotación del algoritmo; como resultado, obtuvieron una reducción del valor del objetivo general y las emisiones de NOx en un 60 % y 77 %, respectivamente; de esta manera se probó que el algoritmo ABC es una herramienta efectiva para dicha optimización, con relación a métodos similares ya probados (algoritmos PSO, GA y PSO-GA) .

De igual manera, el algoritmo de optimización por enjambre de partículas (Particle Swarm Optimization - PSO), identificado por primera vez por Kennedy y Eberhart, es otro algoritmo de búsqueda estocástica basado en el comportamiento social de las bandadas de aves y los patrones que gobiernan la capacidad de las aves para volar sincrónicamente, donde cada partícula (cada ave) representa una solución potencial al problema de optimización. Con este comportamiento colectivo de las partículas se obtiene una mejor eficiencia de convergencia y más aún, genera soluciones en regiones óptimas de un espacio de búsqueda de alta dimensión (César R. López M., 2014; Fallah et al., 2018). En efecto, en las investigaciones realizadas por Nisi et al. (2018), hace uso del algoritmo PSO y otros métodos evolutivos para obtener un ajuste óptimo de control PID. Para su estudio, emplearon una máquina de papel DCS, y tras simulaciones realizadas, concluyeron que el controlador PID multiobjetivo basado en PSO tiene la mejor ventaja de una constante de tiempo de bucle cerrado, lo que permite al controlador actuar rápidamente con un rebasamiento y un tiempo de ajuste equilibrados.

Los laboratorios del Departamento de Eléctrica, Electrónica y Telecomunicaciones de la Universidad de las Fuerzas Armadas - ESPE, están equipados para el desarrollo académico de los estudiantes y cuentan con el material necesario para las investigaciones que corresponden a esta área. Específicamente, en el laboratorio de robótica se encuentra varias plantas de temperatura, las cuales disponen de una salida lineal analógica y permiten el manejo de las mismas a partir de una tarjeta controladora de cualquier tipo. Este equipo es idóneo para la experimentación de diferentes técnicas de control, puesto que permite emular las variaciones presentes en una planta industrial.

1.2. Justificación e Importancia

A pesar de los avances tecnológicos, un diagnóstico realizado en el sector industrial de la ciudad de Cuenca (Sánchez y Pizarro, 2010), demostró que el control de los procesos se los realiza en un 48 % en modo manual, en un 27 % en modo semiautomático, en un 18 % en modo automático y en un 7 % de forma computarizada; en efecto, se concluyó que existe un bajo nivel de automatización de procesos. Es por esto que, el Plan Nacional de Desarrollo 2017-2021 "*Toda una vida*" (PNBV), a través de sus políticas plantadas, se orientan a la construcción de una nueva arquitectura productiva, donde garantiza el uso de tecnologías aplicadas al incremento de la productividad. Específicamente, el ítem 5,6 de las políticas del objetivo 5, señala que se promoverá a la investigación, el desarrollo y trasferencia de tecnología, la innovación y otros factores, para impulsar el cambio de la matriz productiva mediante la vinculación entre el sector público, productivo y las universidades (Secretaría Nacional de Planificación y Desarrollo - Se y Nplades, 2017).

Dentro de los procesos de automatización, el controlador PID, no deja de ser la opción más fácil y efectiva para el control de varios sistemas; por el contrario, el mal ajuste de los parámetros de los controladores ya implementados dentro de la industria, genera un rendimiento ineficiente en los sistemas controlados. Asimismo, en algunos sistemas que se emplea el método de Ziegler Nichols u otros métodos tradicionales, no se logra un óptimo ajuste del controlador Nisi et al. (2018). Por otro lado, los algoritmos bio-inspirados son relativamente un área nueva de investigación; ya que, basados en procesos naturales, busca soluciones a problemas de la vida real muy complejos que no han podido ser resueltos mediante la aplicación algoritmos determinísticos tradicionales. Y aunque se han creado para ese fin, al obtener buenos resultados con dichos algoritmos, se han aplicado a mejorar procesos industriales y explorar las soluciones más optimas con el uso de diferentes métodos bio-inspirados (A. Mishra, 2017).

No obstante, los algoritmos genéticos han sido motivo de estudio dentro del área de control desde los años 70, pues en esta época, por primera vez se implementó la adaptación de la genética artificial en los sistemas de control de computadora; de esta manera, consiguió una gran popularidad en las aplicaciones de control. Pero, no es hasta la última década que los algoritmos SI comienzan a ser un punto de interés para diversos estudios.

Y visto que, dentro del sector industrial, para el correcto funcionamiento de varios procesos de producción, se requiriere una alta precisión en el control de temperatura, mas aun, un control tipo PID tradicional no funciona en procesos donde el tiempo muerto es particularmente largo o la aplicación requiere menos tiempo de espera, tampoco para la planificación anticipada para evitar impulsar el esfuerzo de control o la variable de proceso fuera de sus rangos aceptables.

Por esta razón, el presente proyecto pretende dar a los lectores una perspectiva general del uso que, en los últimos años, se ha dado a los algoritmos bio-inspirados en las diferentes áreas de ingeniería, y más aún, contribuir en el análisis del rendimiento de controladores sintonizados mediante el uso de algoritmos de optimización basados en inteligencia de enjambre ABC y PSO, en comparación con el rendimiento de un controlador PID sintonizado mediante algoritmos genéticos aplicados a una planta de temperatura.

La comparación de rendimiento de los controladores PID sintonizados mediante los dos algoritmos basados en inteligencia de enjambre (ABC y PSO), se lo realizará versus a un controlador del mismo tipo pero sintonizado mediante algoritmos genéticos, debido a que dichos métodos tienen principios similares; dichos métodos se basan en procesos propios de la naturaleza y la sintonización con algoritmos bio-inspirados, sólo demandan la función de evaluación y los parámetros que influyan en la dirección de la búsqueda; mientras que, los métodos clásicos, como por ejemplo el método de Ziegler-Nichols, requiere el modelo matemático de la planta para emplear reglas determinísticas en la búsqueda de la solución. La planta de temperatura donde serán aplicados los controladores diseñados, es el único material más acorde y disponible en los laboratorios de Eléctrica y Electrónica para llevar a cabo el presente proyecto. Y a pesar que es un equipo didáctico, este es bastante completo y se aproxima con gran similitud a un proceso de el control de temperatura industrial.

Con el desarrollo descrito, se espera aportar al análisis y aplicación de nuevas alternativas para la sintonización de controladores PID, y al uso de algoritmos bio-inspirados dentro de procesos industriales. Esta investigación favorecerá, no sólo a la industria, sino que a la comunidad científica y a estudiantes; servirá como base para futuros estudios y aplicaciones en el área de sistemas de control inteligente y problemas de ingeniería basados en optimización. Además, se resalta que este trabajo es pionero en la utilización de algoritmos de optimización basados en inteligencia de enjambre ABC y PSO aplicados a la sintonización de un controlar para una planta de temperatura.

La temática de estudio en este trabajo de investigación contribuye al proyecto “Ajuste de los parámetros del controlador Fuzzy Logic del sistema de gestión energética de una microrred doméstica conectada a red mediante algoritmos de optimización inspirados en la naturaleza” (2018-REV -003).

1.3. Alcance del proyecto

En el presente proyecto se realizará la sintonización de los parámetros de un controlador PID (constantes proporcional, integral y derivativa), mediante las técnicas de ajuste basadas en los algoritmos de optimización *Colonia de Abejas Artificiales* y *Enjambre de Partículas*, donde se pretenderá lograr que el bucle de control corrija eficazmente y en el mínimo tiempo posible los efectos de las perturbaciones aplicadas a una planta de temperatura. Al mismo tiempo, se realizará la comparación del rendimiento alcanzado por los controladores sintonizados mediante algoritmos ABC y PSO con un controlador sintonizado mediante algoritmos genéticos GA. Además, para la evaluación del rendimiento de los controladores se empleará perturbaciones propias de una planta de temperatura.

La implementación de los controladores previamente diseñados, se lo realizará en una tarjeta controladora para el manejo de una de las plantas de temperatura disponibles en el laboratorio de robótica de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas – ESPE. A continuación, se muestra la planta física a emplearse en la Figura 1 y su principio de funcionamiento en el diagrama de la Figura 2, donde se muestra el cilindro calentado por una niquelida, la entrada de aire frío por medio de un ventilador y la salida del aire caliente. También se presenta las perturbaciones que pueden introducir al cilindro (Figura 3).

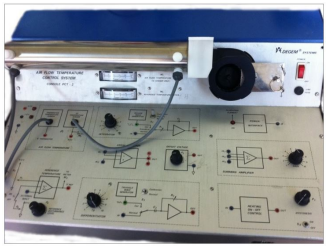


Figura 1. Planta de temperatura marca Degem System

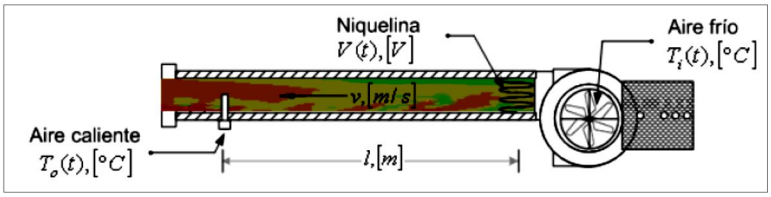


Figura 2. Principio de funcionamiento de la planta de temperatura
Fuente: Miniguano (2008)

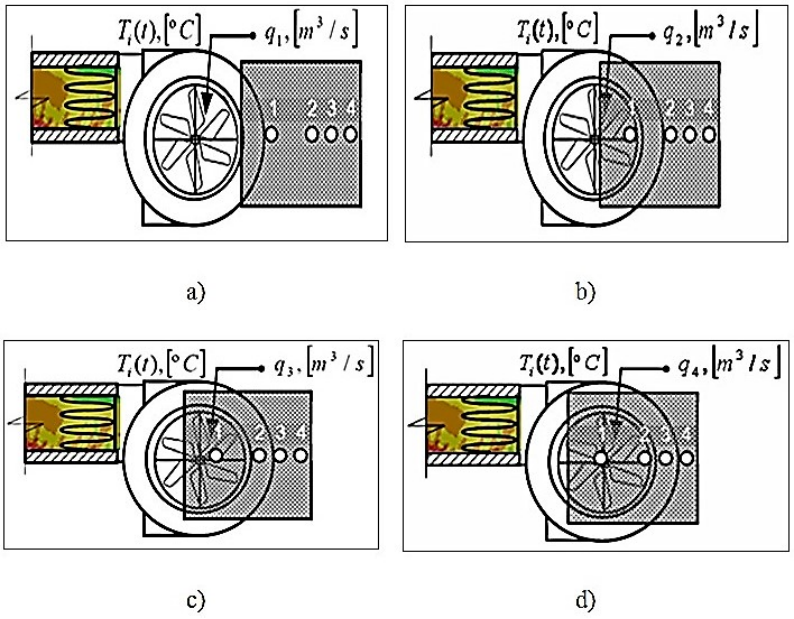


Figura 3. Perturbaciones de la planta
Ingreso de aire frío: a) por medio del orificio 1; b) por medio del orificio 1 y 2; c) por medio del orificio 1, 2 y 3; d) por medio de los 4 orificios.
Fuente: Miniguano (2008)

Por consiguiente, primero se realizará el modelamiento de la planta de temperatura a controlar, posteriormente se diseñará e implantará un control PID sintonizado mediante algoritmos genéticos, pues este controlador será una referencia para la comparación de los otros controladores a diseñarse. Seguido, se realizará la sintonización de los controladores mediante algoritmos de optimización ABC y PSO, e implementación de los mismos en la tarjeta controladora.

Una vez llevada a cabo la fase de diseño e implementación, se dará inicio a las pruebas de rendimiento y robustez de los controladores, bajo los criterios más apropiados y definidos durante el desarrollo del proyecto.

Finalmente, se presentará las particularidades que presente cada controlador, y los resultados del análisis comparativo del rendimiento de los controladores sintonizados mediante los dos algoritmos de optimización basados en inteligencia de enjambres ABC y PSO, y el controlador sintonizado mediante GA.

1.4. Objetivos

1.4.1. Objetivo Específico

Presentar un análisis comparativo del rendimiento de dos controladores tipo PID sintonizados mediante algoritmos basados en inteligencias de enjambre Colonia de Abejas Artificiales y Enjambre de Partículas, frente al rendimiento de un controlador tipo PID sintonizado mediante Algoritmos Genéticos, aplicados a una planta de temperatura.

1.4.2. Objetivos Específicos

- Desarrollar el controlador PID sintonizado mediante el algoritmo de optimización basados en inteligencia de enjambre Colonia de Abejas Artificiales (ABC).
- Desarrollar el controlador PID sintonizado mediante el algoritmo de optimización basados en Enjambre de Partículas (PSO).
- Evaluar a cada controlador diseñado bajo los criterios de desempeño que permiten evaluar el comportamiento al sistema en régimen transitorio y permanente.
- Analizar los resultados obtenidos de la evaluación de desempeño de cada controlador implementado en régimen transitorio y permanente.

Capítulo 2

Estado del Arte

En este capítulo se presenta un estudio sobre los principales aspectos que conforman las bases de la Computación Bio-inspirada, y un análisis de la trasformación de los modelos naturales a los procesos sistemáticos que rigen a los algoritmos de optimización GA, PSO y ABC.

2.1. Algoritmos Bio-inspirados

2.1.1. Generalidades

A partir el origen de la tierra, la naturaleza ha evolucionado con el fin de adaptarse a nuevas condiciones que se han presentado al largo de los tiempos; con ello, se han desarrollado organismos con diseños misteriosos, únicos, peculiares e ingeniosos que se han convertido en modelos de inspiración para el ser humano. Dichos mecanismos biológicos diseñados para hacer frente a los retos del entorno, por medio de la observación, han sido el principal eje de apoyo para la mayor parte de avances científicos, es decir ha sido una fuente de soluciones prácticas y eficientes para los desafíos que la ciencia ha enfrentado.

Desde campos como la arquitectura, medicina, economía, tecnología, entre otros, se han basado en la naturaleza no sólo para crear objetos que mejoren la calidad de vida de las personas (como

sonares inspirados en murciélagos, velcro inspirados en plantas de cardo, sistemas de ventilación inspirados en nidos de termitas, agujas hipodérmicas inspirados en colmillos de serpiente, etc.). Sino que también, la observación de la naturaleza ha sido la base para establecer las leyes que rigen nuestro día a día como las leyes de la termodinámica o las leyes de Newton.

De igual manera, la computación bio-inspirada, subcampo de la inteligencia artificial, emula el comportamiento de los sistemas naturales con el fin de diseñar métodos heurísticos no determinísticos. Dentro de esta área, encontramos a los procesos artificiales bio-inspirados metaheurísticos, más conocidos como algoritmos bio-inspirados que han sido empleados para clasificación, aproximación, búsqueda, reconocimiento, aprendizaje, caracterización y optimización.

Los primeros algoritmos bio-inspirados surgieron a partir de la teoría evolutiva publicada en el libro titulado *"Origen de las Especies"* en 1859 por Charles Darwin, donde se proponía la selección natural y una continua competencia entre organismo; pues aseguraba que los organismos que consiguieran adaptarse mejor lograrían más recursos y una mejor reproducción. A partir de esto, también surgió el trabajo de Gregor Mendel, que a la vez, dio origen a la Ley de la Herencia de Mendel; otro concepto que sirvió como base para los algoritmos bio-inspirados, ya que relacionaron ADN, genes, cromosomas, selección, cruce, mutación, estado físico y generaciones (Arana et al., 2018).

En la década de 1960, John Holland y su grupo de trabajo fueron los pioneros en el desarrollo de algoritmos genéticos en la Universidad de Michigan, alcanzaron publicar su trabajo en 1975 (X.-S. Yang, 2011). Pero no fue hasta, aproximadamente, las dos últimas décadas que los algoritmos bio-inspirados han ganado popularidad tanto en el estudio de nuevos algoritmos bio-inspirados orientados a la resolución de problemas complejos de manera más efectiva y eficiente, como también en la parte aplicativa de los mismos, especialmente enfocado a la búsqueda de una optimización global. Es por eso que, relativamente, esta área de investigación es nueva, no obstante, los algoritmos genéticos son los métodos que paulatinamente se han empleado tanto en el área industrial como en el área científica y social por sus años de estudio y renombre.

La particularidad de los algoritmos bio-inspirados, es el de encontrar la solución o soluciones a problemas complejos de la vida real a partir de condiciones y reglas simples, y con el mínimo

o ningún conocimiento del espacio de búsqueda. Esto, está vinculado al hecho que los algoritmos imitan las mejores características de la naturaleza.

Ahora bien, los algoritmos bio-inspirados al ser algoritmos metaheurísticos (procesos estocásticos con aleatorización y exploración global), realizan una técnica de búsqueda local y global, gracias a dos componen principales: la diversificación y la intensificación. La diversificación permite generar diversas soluciones para explorar el espacio de búsqueda de manera global; en cambio, la intensificación centra la búsqueda en una zona local en base a la información zonal de una buena solución actual (X.-S. Yang, 2011).

2.1.2. Clasificación de los Algoritmos Bio-inspirados

A partir de la base descrita anteriormente, se han desarrollado múltiples algoritmos metaheurísticos inspirados desde la fisiología y la biología molecular hasta la ecología, desde la zoología a la botánica, e incluyen organización social. A continuación, se presenta una clasificación de los algoritmos bio-inspirados más conocidos en la figura 4 (Binitha y S Siva, 2012; Darwish, 2018; Fister et al., 2013; A. Mishra, 2017).

Actualmente, por la diversidad de algoritmos bio-inspirados con los que se cuenta, el campo de aplicación está rápidamente expandiéndose, principalmente porque permiten resolver problemas de naturaleza muy compleja, indiferentemente del área de manejo.

Se conoce que los algoritmos bio-inspirados han sido aplicados en procesamiento de imágenes médicas para diagnóstico preventivo, optimización de rutas para robots y vehículos no tripulados, enrutamiento de redes, diseño óptimo de dispositivos de ingeniería, predicción en finanzas, análisis de datos, optimización industrial, logística, sistemas de control de tráfico, computación en la nube, sistema de información geográfica, bioinformática, meteorología, etc., (Abdel-Basset y Shawky, 2018).

El resto del capítulo, hace una referencia general de los principios básicos de los algoritmos evolutivos y algoritmos genéticos. Además, se analizará a detalle dos de los algoritmos basados en inteligencia de enjambre: colonia de abejas artificiales (ABC) y optimización por enjambre de

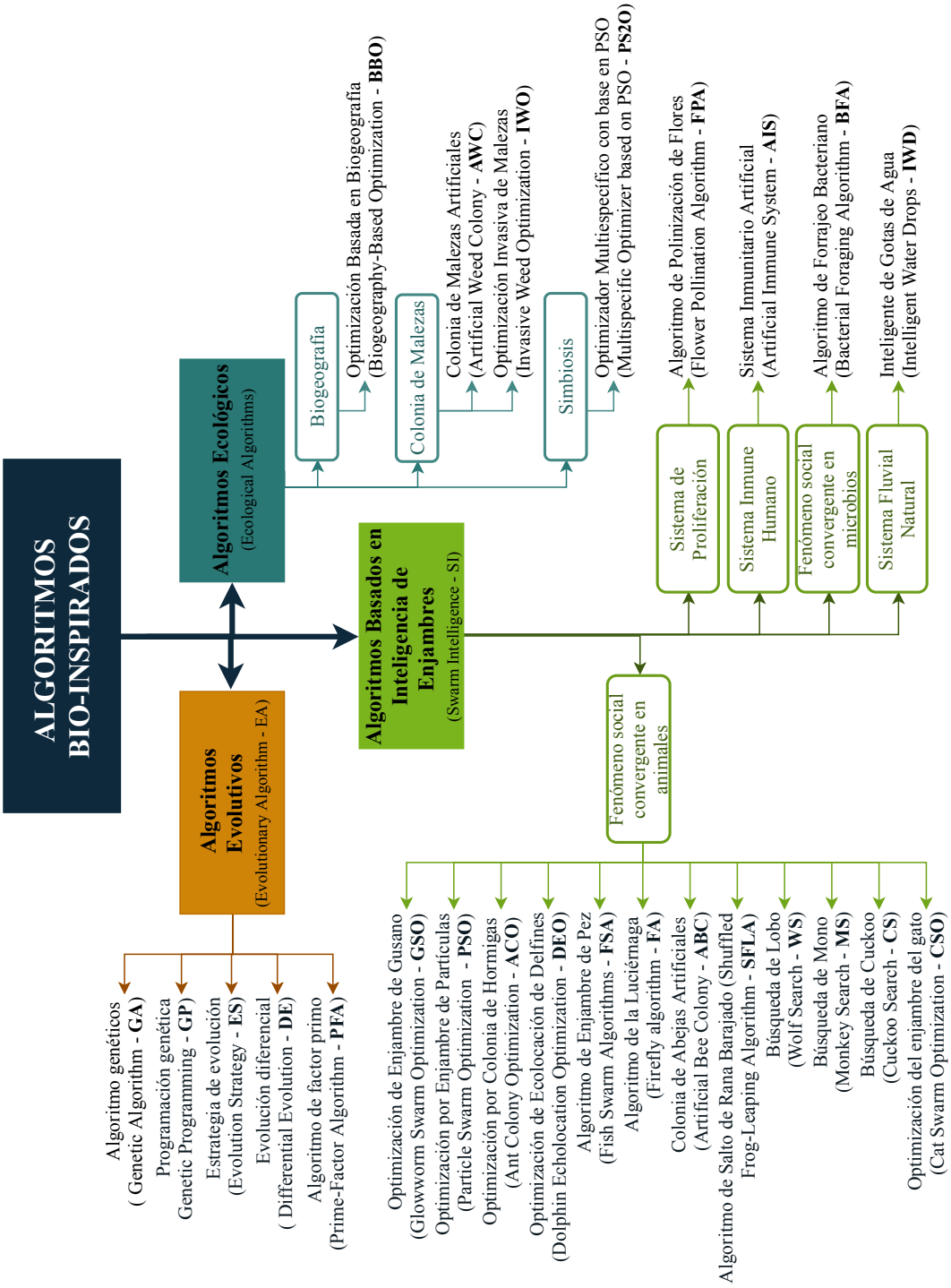


Figura 4. Clasificación de Algoritmos Bio-inspirados.

partículas (PSO); se describirá la inspiración, la metodología de cada algoritmo, y adicionalmente se dará a conocer los trabajos realizados en basados en estos métodos más recientes.

2.2. Algoritmos Evolutivos

Como fue mencionado anteriormente, los algoritmos evolutivos EA (por sus siglas del inglés: *"Evolutionary Algorithm"*), fueron los primeros en ser formulados, y se fundamentan en el proceso de transformar una especie en otra especie descendiente; es decir, se empleó el cambio adaptativo de las especies por medio de la selección natural, donde los individuos más débiles son los que mueren y así no contribuyen a la herencia genética de las futuras generaciones (Back, 1996).

Los EA se basan en cuatro pilares: población, diversidad, herencia y selección. La población está compuesta por un grupo de individuos de una misma especie que compiten según su capacidad para mejor la población. Los individuos de dicha población, son evaluados según su aptitud física, y los clasifican, serán quienes darán origen a una nueva generación con características mejoras. El progreso de la especie se obtendrá mediante un continuo proceso de cruce, mutación y selección.

Desde la década de 1950 varios investigadores han trabajado en el desarrollo de algoritmos inspirados en la evolución para mejor los procesos computacionales. Entre estos investigares se puede mencionar a Box en el año de 1957, a Friedman en el año de 1959, a Bledsoe en 1961, Bremermann en 1962 y Reed y su equipo en 1967, sin embargo, dichos trabajos no tuvieron mucha trascendencia. Por otro lado, Rechenberg en los años 1965 a 1973 desarrolló el método Estrategias de Evolución, utilizado para la optimización de los parámetros de valor en dispositivos aerodinámicos; Fogel y su equipo en 1966 desarrollaron el método de la Programación Evolutiva, donde las posibles soluciones se representaban como máquinas de estado finito; y Holland junto sus colaboradores, propusieron el método de Algoritmos Genéticos, el cual se amplía más adelante (Mitchell, 1998b).

Los EA parte de la generación de una población aleatoria, sobre la cual se aplicarán las operaciones de selección, cruce y mutación, para así finalmente, remplazar la población inicial con la nueva generación de pobladores elegidos de forma iterativa. De forma concreta, los EAs inician dada una función objetiva con atributos de evaluación, para lo cual se crea aleatoriamente

un conjunto de posibles soluciones; los elementos de este conjunto, pertenecerán al dominio de la función. Al aplicar la función, se valorará cada una de las posibles soluciones, donde, de manera abstracta se establecerá que en cuanto más alto mejor. En base a esta evaluación, los mejores candidatos (mejores soluciones) son elegidas para conformar la próxima generación (nuevo conjunto de posibles soluciones) después de la recombinación y/o mutación entre sí. El proceso de recombinación es una operación que se aplica a dos o más candidatos seleccionado y da como resultado más de un nuevo candidato (elemento/s de la nueva generación); mientras que la mutación se aplica a un solo candidato y como resultado se obtiene otro nuevo candidato. La aplicación de estos dos operadores, recombinación y mutación, generan el nuevo conjunto de candidatos (posibles soluciones), que compiten nuevamente para ser parte del próximo conjunto de candidatos. El proceso es iterativo hasta encontrar el candidato con mejor aptitud (solución más óptima) o hasta que se alcance el límite computacional anticipadamente establecido. En el diagrama de la Figura 5, se representa de forma gráfica el proceso de un algoritmo evolutivo.

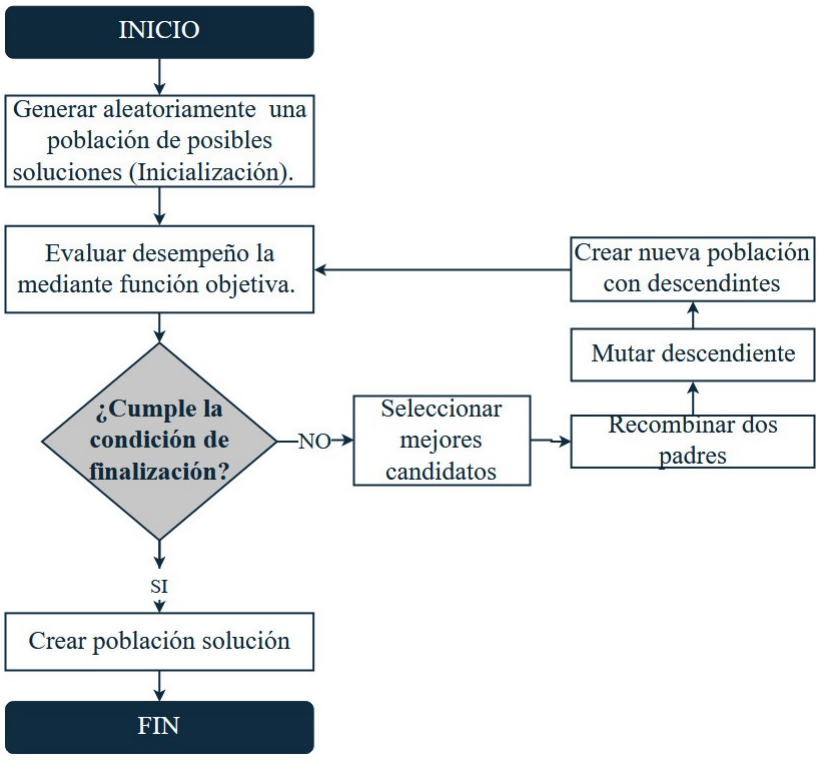


Figura 5. Diagrama de flujo de procesos de *Algoritmos Evolutivos (EA)*.

Cabe señalar que, la forma en que la población evoluciona hacia las posibles soluciones, y la forma en que elige la mejor solución o el conjunto solución, es algo inherente a cada EA.

2.2.1. Algoritmo Genético

2.2.1.1. Preámbulo

Dentro de la clasificación de los algoritmos evolutivos (Fig.4), el Algoritmo Genético (conocido como GA, por sus siglas en inglés de *"Genetic Algorithm"*) es una técnica de búsqueda y optimización basada en la teoría de la evolución, que pretende replicar el comportamiento biológico de la selección natural y la genética; mismo que ha llegado a convertirse en uno de los métodos de optimización más usados por tener la representación más simple de individuos.

El trabajo de Holland y su equipo (creadores del algoritmo), inició con el estudio meticuloso de cada aspecto que forma parte del fenómeno de adaptación en la naturaleza, para así poder desarrollar un mecanismo idéntico que pueda ser importado a los sistemas informáticos (Mitchell, 1998b). Este trabajo fue el cimiento de posteriores estudios que, tomaron como dogma el uso generalizado de cadenas binarias como cromosomas. Sin duda, con la publicación de su trabajo, no sólo se abrió un campo de nuevas investigaciones enfocadas en la computación evolutiva como modelo científico de procesos evolutivos; sino que, las aplicaciones se extendieron desde la informática hasta la ingeniería, pues se convirtió en una herramienta potencial en esta área.

El objetivo de GA es codificar una solución potencial de un determinado problema, en una estructura de datos similares a la de un cromosoma; y al igual que los EA, GA aplica operadores de selección, recombinación y mutación sobre estas estructuras, de tal manera que, se preserve la información crítica. De hecho, este método se caracteriza por centrar su proceso en la recombinación (o cruce), más que en la mutación, puesto que, con la recombinación (proceso principal) se obtiene una exploración más amplia del espacio de soluciones, siempre y cuando se asuma que es posible hallar de manera independiente partes de la solución óptima que después de ser combinadas crearán mejores soluciones, mientras que la mutación (proceso secundario) se aplica para obtener una diversidad de individuos que introduzcan información adicional a la que se obtiene del cruce (Whitley, 1994; Zomaya y Y., 2006).

2.2.1.2. Descripción del método GA

Proceso Natural de Inspiración

En cuanto a la inspiración biológica de los GAs, responden fielmente a los principios básicos de los EAs y comparten su esquema general. El proceso bio-inspirador parte del material genético contenido en los cromosomas; en síntesis, todo organismo vivo contiene células, cada célula contiene cromosomas y cada cromosoma contiene ADN, la secuencia de ADN se lo conoce como genoma (que es la colección completa de material genético), el genotipo en cambio, es el conjunto de genes contenidos en un genoma; en conclusión, de un cromosoma se obtiene las instrucciones genéticas usadas en el desarrollo y funcionamiento (características) de un individuo que lo hace único. Ahora bien, los cromosomas de un individuo vienen dados del cruce de material genético de dos padres, en este cruce se da la selección natural de ciertos atributos o la mutación de otros (definición de características propias del nuevo individuo) (Fig.6.) que, en esencia, se trasmite lo mejor del material genético de los padres para el nuevo individuo, es decir la solución más eficaz de esa población.

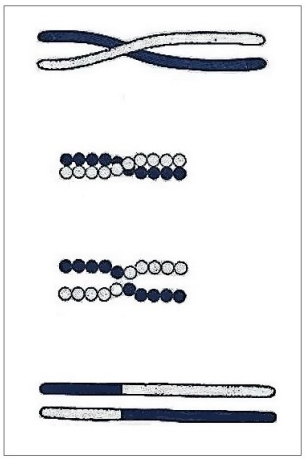


Figura 6. Recombinación genética de un cromosoma.
Fuente: (Morgan,1916)

Los GA usan una analogía al proceso de herencia y mutación genética, que lo hacen un método robusto. Este algoritmo, inicia con la creación de una población de individuos aleatorios que

representa el conjunto de posibles soluciones iniciales. A cada individuo se le es asignado un valor (puntuación) en base al nivel según los atributos de dicha solución. Dentro de la naturaleza, esto equivaldría al grado de competitividad que posee un individuo que lo hace sobresalir entre el grupo; cuanto mayor sean las cualidades de adaptación del individuo a un problema, existirá una mayor probabilidad de que este individuo sea seleccionado para su reproducción y cruce de material genético con otro individuo seleccionado, esto dará origen a un individuo descendiente que será parte de la nueva población. En cambio, para los individuos que menor grado de adaptación presenten, menor será la probabilidad que sean seleccionados para ser reproducidos, por lo que, su material genético no estará presente en las nuevas generaciones. A lo largo de las generaciones sucesivas, las mejores características se propagarán entre la población, así se consigue individuos mejor adaptados, es decir, se conseguirá las mejores soluciones que solventen el problema. Del mismo modo, si el GA es bien diseñado, la población convergerá hacia la solución más optima del problema.

Mecanismo

La implementación de un algoritmo genético comienza con la construcción de una población de cromosoma; generalmente estos son cadenas de bits generados aleatoriamente. Luego, cada cromosoma de la población actual es evaluado y asignado un valor que representa la oportunidad de reproducirse, a través de una función de evaluación de aptitud; la aptitud de un cromosoma depende de qué tan bien tal cromosoma resuelva el problema en cuestión (Mitchell, 1998b). Una vez seleccionados dos cromosomas, se aplica operadores de cruce y mutación. Por último, el nuevo cromosoma es introducido a la población para ser nuevamente evaluado. Este proceso es continuo hasta que se cumpla la condición de paro. En la figura 7, se observa un diagrama que representa de manera más simple el proceso del GA.

Los elementos de la estructura general, aplicados en los GAs, son generalmente siete. A continuación, se presenta en breve detalle de dichos elementos presentes en el pseudocódigo de los GAs.

- *Codificación*

  Las posibles soluciones pueden ser representadas como un conjunto de parámetros (genes), los cuales al ser agrupados forman cromosomas. Cada gen puede tomar el valor de

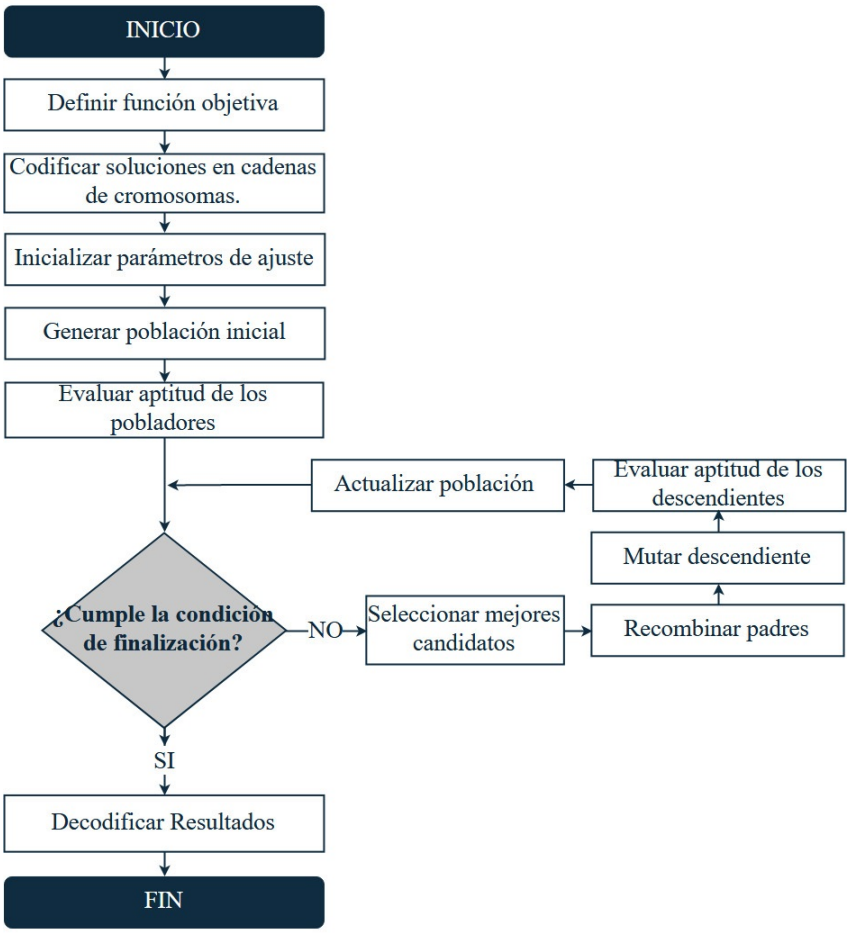


Figura 7. Diagrama de flujo del *Algoritmo Genético (GA)*.

$[0, 1]$(representación normalmente empleada).

Como se ve puede observar en la figura 8, las posibles soluciones son representadas en el cromosoma en forma de un vector binario, pero también, es posible que el cromosoma sea una matriz (Fig. 9). Esto depende de las necesidades del problema a tratarse.

- *Inicialización de un GA*

  En la inicialización se procede a definir todos los parámetros con los que desarrollará el algoritmo, tales como: tamaño de la población, longitud del cromosoma (cantidad de genes que conformará el cromosoma), población inicial, probabilidad de cruce, probabilidad

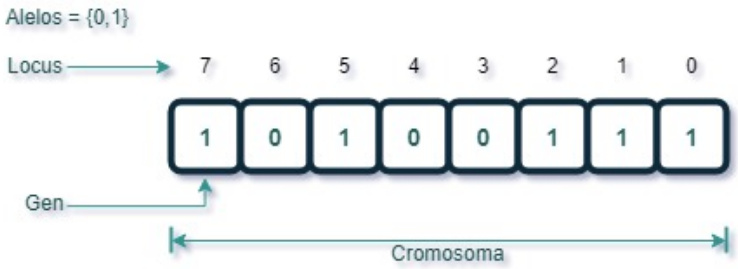


Figura 8. Codificación de un cromosoma tipo vector.

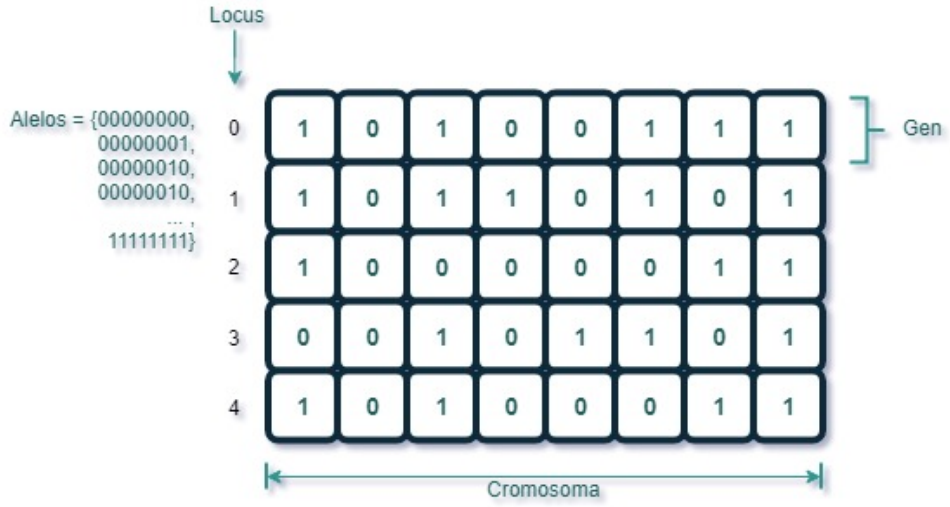


Figura 9. Codificación de un cromosoma tipo matriz.

de mutación, criterio de parada y porcentaje de cromosomas que conformaran la siguiente generación.

- *Función de aptitud*
  La función de aptitud o función de evaluación de adaptación debe ser diseñada específicamente para cada problema. Esta función, asigna un valor en números real, en referencia a la capacidad de solventar el requerimiento del problema, al cromosoma evaluado.

- *Selección*
  La selección de cromosomas candidatos para el cruce se efectúa a través de un procedimiento que favorezca a los individuos con mejores características de adaptación, y de igual manera, la elección de dos padres (quienes cruzaran su material genético) será aleatorio.

- *Cruzamiento*

  El operador de cruce, toma a los dos padres seleccionados y cortan los cromosomas en un locus (posición particular dentro de la estructura del cromosoma) escogida al azar, para luego intercambiarlos (Fig. 10). Así los descendientes heredaran genes de los dos padres. Esta operación, proporciona una exploración rápida del espacio de búsqueda.

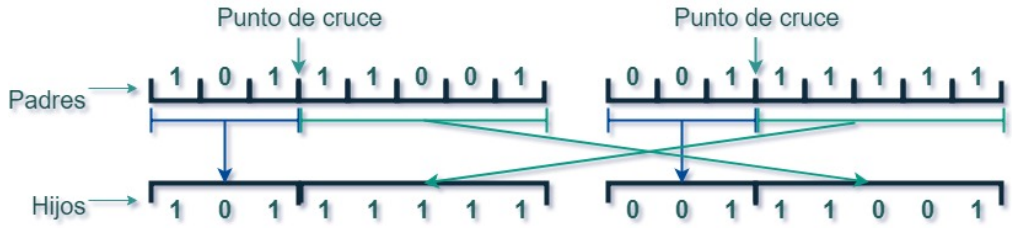


Figura 10. Proceso de cruce en base a un punto.

El cruce de un solo punto corresponde al GA original propuesto por Holland, sin embargo, las versiones posteriores incluyen algunas variantes. Como, por ejemplo: el cruce uniforme, donde cada gen en la descendencia elige al azar de uno u otro padre, según a una máscara de cruce generada aleatoriamente (si hay un 0 en la máscara, el gen es copiado de la madre, mientras que con un 1 en la máscara, el gen es copiado del padre) (Mitchell, 1998a).

- *Mutación*

  El operador de mutación es aplicado a cada cromosoma descendiente. Consiste en la modificación aleatoria de un gen componente del cromosoma (Fig. 11). Esta operación garantiza que ningún punto del espacio de búsqueda tenga una probabilidad de no ser explorado, y asegura la convergencia de los Algoritmos Genéticos a una solución.

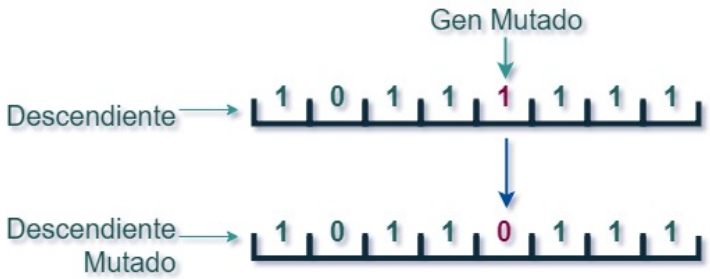


Figura 11. Proceso de mutación en base a un punto.

- *Remplazo de población*
  Diferente al proceso de selección, el remplazo elige cuales de los nuevos descendientes serán parte de la siguiente generación. Esta operación se relaciona con el proceso de supervivencia poblacional, más no en una competencia entre organismos de una especie que es lo que ocurre en la selección.

Consideraciones

Los algoritmos genéticos poseen varias características que los hacen altamente deseables para problemas de optimización. Dos características inherentes de los GAs son la capacidad de manejar problemas complejos y paralelismo. Esto se consigue gracias los múltiples descendientes que, al actuar de forma independiente, exploraran el espacio de búsqueda en diversas direcciones simultáneamente, y admiten a la vez, manipular diferentes parámetros o diferentes grupos de cadenas codificadas al mismo tiempo (X.-S. Yang, 2014).

Por el contrario, se presentan desventajas en un algoritmo GA, en la determinación de los parámetros de inicialización (función de aptitud, tamaño de la población tamaño de la población, longitud del cromosoma, probabilidad de cruce, probabilidad de mutación, etc.). Cualquier elección inapropiada causará problemas de convergencia o falsas soluciones, además no son adecuados para resolver directamente problemas de optimización de restricciones (Binitha y S Siva, 2012).

2.2.1.3. Trabajos Relacionados

A los algoritmos genéticos se los ha conceptualiza como optimizadores de funciones, no obstante, los campos de problemas a los que se han aplicado esta técnica son múltiple. A continuación, se presenta los trabajos que se han desarrollado a lo largo de los últimos años en base a algoritmos genéticos.

Un problema de gran importancia en la actualidad es la escases de agua, es por eso que los investigadores Naghibi et al. (2017), en su trabajo, presentaron la aplicación de tres técnicas empleadas para el mapeo de fuentes de agua subterráneas potenciales; estos métodos fueron: máquina de vectores de soporte, bosque aleatorio y bosque aleatorio optimizado por algoritmo genético. El

estudio se realizó en un área de $1524km^2$ dentro de la provincia de Ardebil, Irán, en la cual se debía considerar un total de 14 variables. En los resultados de la evaluación de la eficiencia de los tres métodos, se destacó la eficiencia del método híbrido *Bosque Aleatorio optimizado por Algoritmo Genético*; para el cual, se ejecutó con una población de 300 cromosomas y un criterio de parada después de 100 generaciones. El tiempo de ejecución fue de aproximadamente $2h20min$.

Por otra parte, el algoritmo genético también ha sido empleado en el diseño de estabilizadores como se presenta en artículo "*New Approach to Design SVC-Based Stabilizer using Genetic Algorithm and Rough Set Theory*". El trabajo describe el diseño sistematizado de un estabilizador basado en un compensador VAR estático y un estabilizador de sistema de energía convencional, en el cual combinan el método del algoritmo genético y la teoría de conjuntos. Este método, algoritmo GA basado en conjuntos aproximados, fue aplicado con el fin de seleccionar y optimizar los parámetros del controlador. Después de pruebas realizadas en un sistema de una sola máquina y un sistema de varias máquinas, con el diseño propuesto, se registró un tiempo de cómputo minimizado, una reducción en capacidad de almacenamiento para el problema de optimización y un aumento en el rendimiento de la estabilidad del sistema eléctrico del sistema eléctrico. Concluyendo así, que el método tiene aplicabilidad y la escalabilidad (Fetouh y Zaky, 2017).

Intrínsecamente en el área de control, el artículo publicado bajo el título traducido: "*Diseño robusto multiobjetivo jerárquico H-Infinito de control para un sistema de transferencia de energía inalámbrico mediante algoritmo genético*", explica el desarrollo de un método de optimización multiobjetivo para la búsqueda de las ponderaciones de un control H-infinito, el cual es aplicado a un prototipo experimental de transferencia de potencia inalámbrica inductiva. Particularmente, el algoritmo GA fue empleado para equilibrar el rendimiento en el dominio del tiempo y el rendimiento robusto. El ajuste de la ejecución para este caso fue de 100 cromosomas y 80 generaciones. Conjuntamente, para la función de evaluación de aptitud se empleó tres restricciones; a cada restricción se lo asoció un factor de penalización para eliminar a los cromosomas que no cumplan con el diseño de control, por esta razón, los cromosomas más aptos eran los que contaban con el mínimo costo. Una de las consideraciones tomadas fue el error en lazo cerrado y el modelo objetivo de segundo orden, así como la amplitud de las frecuencias. Los resultados tras la aplicación del controlador diseñado, mostraron que existía un rendimiento superior, en tiempo y robustez, en comparación a otros controladores probados (M. Yang et al., 2018).

Asimismo, el algoritmo GA fue empleado para el diseño de un control de velocidad de un motor BLDC. La particularidad de este sistema de control, es la elaboración del control mediante un solo circuito de trasporte de corriente de segunda generación. El tipo de controlador suficiente para el control del motor BLDC, que sólo difiere de un motor DC por la falta de un contacto físico entre el rotor y el estator, es de tipo PID; por esta razón, los parámetros de ajuste del algoritmo GA indicados para la ejecución, fue de una población de 60 individuos y 50 generaciones, con probabilidad de mutación de 0.35. Las pruebas simuladas realizadas mediante el circuito integrado AD844, demostraron que el diseño en circuito cerrado muestra una buena capacidad de seguimiento (Kumari et al., 2018).

En el caso del diseño de un controlador LQR multirate adaptativo, que es empleado por la flexibilidad que lo caracteriza, el uso de algoritmo GA se aplica para facilitar su diseño. En la investigación "*Genetic Algorithm Design of an Adaptive, Multirate LQR Controller for a Multi-Machine MVDC Shipboard Electric Distribution System with Constant Power Loads*", se enfoca en la necesidad de equipar buques de guerra con *sistemas integrados de energía*. El problema que presentan es la inestabilidad y la poca eficiencia del control lineal cuando las impedancias no lineales negativas aparecen en las cargas de potencia constante; por este motivo, aplican un control LQR-SM el cual involucra el manejo de un alto número de variables. Por otra parte, el algoritmo GA es una técnica estocástica aplicada para determinar los valores de las variables de diseño. El principal resultado es el óptimo rendimiento dinámico del sistema (Mills y Ashton, 2017).

Dentro de la robótica, una aplicación del algoritmo genético, se presentó en artículo "*Approach of Kinematic Control for a Nonholonomic Wheeled Robot using Artificial Neural Networks and Genetic Algorithms*". El objetivo de la investigación se enfocaba en mejor el nivel de autonomía de vehículos personales. El diseño del controlador se lo realizó mediante el algoritmo denominado "Neuroevolución de las topologías aumentadas", por la combinación de la topología de la red neuronal y la optimización de la misma mediante el algoritmo genético. Con esta técnica mejorada de diseñe, lograron considerar las limitantes cinemáticas del modelo físico del robot para la toma de decisiones durante la navegación en zonas desconocidas; ya que otros métodos de diseño, sólo siguen reglas de restricción y limitan el movimiento del vehículo. Tras las pruebas realizadas, lograron evitar obstáculos durante la trayectoria para llegar al punto de destino mediante soluciones fáciles generadas por la correcta definición de la condición física (Caceres et al., 2017).

2.3. Algoritmos basados en Inteligencia de Enjambre

La inteligencia de enjambre, más conocido como SI (por sus siglas del inglés *"Swarm Intelligence"*), es un conjunto de técnicas derivadas de la Inteligencia Artificial que, basan su metodología en el comportamiento colectivo natural de las especies con habilidades de autoorganización y control descentralizado, como por ejemplo: colonias de insectos, bandadas de aves, cardúmenes de peces, manadas de mamíferos, etc.

Dicho de otra forma, SI imita la forma de vida de un sistema que está compuesto por un conjunto de numerosos agentes no sofisticados que cumplen reglas simples para lograr interactuar con su entorno, estas interacciones más el intercambio de información entre agentes, miembros del mismo sistema multiagente, dan origen a una conducta colectiva inteligente, que les permite realizar tareas complejas como la búsqueda de alimento, determinación de las mejores rutas de desplazamiento, defensa ante peligros, etc. Este tipo de tareas, requeridas para su supervivencia, pero sin un control centralizado, son logradas a través de la adición de los esfuerzos individuales de todos los agentes, a pesar que ningún agente conoce el objetivo general que se debe alcanzar. Por lo cual, el enfoque SI establece un modelo práctico que facilita el diseño de soluciones distribuidas para diversos problemas (Zomaya y Y., 2006).

Históricamente, a partir de la aparición de los algoritmos evolutivos (aproximadamente en la década de 1960), varios investigadores han continuado con la búsqueda de nuevas estrategias de computación bio-inspiradas. Es así que, en el año de 1992, Marco Dorigo fue el pionero en presentar un método de optimización basado en colonias de hormigas (ACO, por sus siglas del inglés: Ant Colony Optimization), que fue enfocado para resolver problemas de optimización discretos (X. S. Yang et al., 2013). Tres años después, la optimización por enjambre de partículas, técnica más conocida como PSO (por sus siglas del inglés: Particle Swarm Optimization), apareció gracias al trabajo desarrollado Kennedy y Eberhart (Eberhart y Yuhui Shi, 2002). La aparición de este último, fue esencial para abrir paso nuevas investigaciones basadas en SI. Sin embargo, no fue hasta el año 2004 que Craig Tovey y su colaborador Sunil Nakrani propusieron el algoritmo de la abeja; pero Xin She Yang, en el año 2005, formuló el algoritmo de abejas virtuales para solventar la optimización de funciones aplicadas a problemas de ingeniería (X.-S. Yang, 2005). En ese mismo año, Karaboga formuló el algoritmo de optimización colonia de abejas artificiales (ABC, por sus siglas

del inglés: Artificial Bee Colony) (Karaboga et al., 2014). Posteriormente, a lo largo de los siguientes años, Xin She Yang continuó con el desarrollo y contribución de nuevos algoritmos basados en SI como, el algoritmo de luciérnaga (FA) en el año 2008, algoritmos de búsqueda de cuckoo (CS) [1] en el año 2009, algoritmo de murciélago (BA) en el año 2010 y algoritmo de polinización de flores (FPA) en el año 2012 (X.-S. Yang, 2014). Asimismo, como fue expuesto en la figura 4, existen aún más algoritmos basados en SI que han sido aportaciones de varios investigadores, como por ejemplo: el algoritmo de enjambre de pez (FSA) que fue propuesto por Li et al. en el año 2002 (L. Li et al., 2002) y el algoritmo de optimización de enjambre de gusano de luz (GSO) presentado por Krishnanand y Ghose el año 2005 (Y. Yang et al., 2010); no obstante, no se entrará en detalles sobre los algoritmos restantes, ya que el objetivo es sólo dar una visión cronológica general de los algoritmos basados en SI.

De manera integral, los algoritmos basados en SI presentan varias ventajas sobre los algoritmos convencionales, gracias a tres factores esenciales que los componen: simplicidad, flexibilidad y ergodicidad. Estos factores, los han llevado a popularizarse a gran escala dentro del campo de búsqueda y optimización. En cuanto al primer factor, es una característica propia de los algoritmos bio-inspirados; ya que, son fáciles de implementar y la complejidad algorítmica es bastante baja en comparación con otros métodos de optimización. El segundo factor, es una particularidad que los han convertido en una herramienta versátil y a pesar de su simplicidad, ha logrado solventar un amplio campo de problemas complejos de la vida real de optimización. Y por último, la ergodicidad (por procesos de aleatoriedad que incluyen los algoritmos) en la exploración dentro espacio de posibles soluciones que permiten descartar óptimos locales y hallar óptimos globales, aunque no hay garantía que esto ocurra en todos los casos, pero sí aplica en la gran mayoría (X. S. Yang et al., 2013).

En este sentido, lo sustancial en el diseño de los algoritmos basados en SI es la representación de los agentes simples, dado que, estos determinan el proceso de generación de soluciones y alteran directamente el potencial de exploración y explotación del proceso de búsqueda. Los agentes pueden incorporar variables en la solución o ser medios para construir dicha solución. De este modo, la representación debe diseñarse de tal modo que todas las posibles soluciones sean asequibles durante el proceso de búsqueda. También, durante el diseño, se debe tomar en cuenta

[1] Xin She Yang trabajó conjuntamente con Suash Deb para el desarrollo del algoritmo de búsqueda de cuckoo.

la definición de vecindario que, dentro de la naturaleza, corresponde a la localidad de los agentes (entorno); la definición de un vecindario delimita los posibles movimientos a realizarse durante la construcción de la solución. Esta localización en conjunto como una regla se decisión, garantizarán que el proceso de exploración se lo ejecute dentro de todo el espacio de búsqueda y el algoritmo no converja en un óptimo local; y mediante el mecanismo de explotación (basada en la experiencia del enjambre) permite indagar a profundidad las zonas más prometedoras del espacio de búsqueda. En definitiva, con un buen diseño, el algoritmo convergerá con éxito en un óptimo global (İnkaya et al., 2016).

Como es evidente en la figura 4, se han desarrollado varios tipos de algoritmos basados en SI, puesto que en la naturaleza existen diferentes tipos modelos de especies adaptados a distintos entornos ambientales. A continuación, se tratará únicamente de los algoritmos de Optimización por Enjambre de Partículas y Colonia de Abejas Artificiales.

2.3.1. Optimización por Enjambre de Partículas

2.3.1.1. Preámbulo

El algoritmo de optimización por enjambre de partículas, más conocido como PSO (por sus siglas en ingles de "Particle Swarm Optimization"), es uno de los métodos basados en SI que, imita el comportamiento social e inteligente de la bandada de aves y la escolarización de peces. Es decir, el algoritmo PSO permite una búsqueda colaborativa basada en la población e introduce funciones de paralelismo (Zomaya y Y., 2006). Además, PSO tiene un enfoque a la resolución de problemas de optimización no lineales con restricciones, gracias a su robustez, simplicidad y flexibilidad. Este método es idóneo para aplicaciones donde las posibles soluciones puedan ser representadas como un punto o una superficie en un espacio multidimensional.

La técnica metaheurística PSO, fue introducido por primera vez por Kennedy y Eberhart en año de 1995; aunque el propósito original fue la animación computarizada de la sutileza e impredecibilidad de los movimientos que ejecuta una bandada de aves, lograron que las partículas se desplacen dentro de un espacio de búsqueda, permitiéndoles así, encontrar una solución óptima

para el enjambre al actualizar su velocidad y posición (más adelante se ampliará los datos históricos). Y a pasar que, el algoritmo de colonia de hormiga apareció años antes, PSO, es considerado como el punto de partida del desarrollo de algoritmos basados en inteligencia de enjambre.

En la práctica, similar al algoritmo genético (GA), en un algoritmo PSO, también se genera inicialmente una población de posibles soluciones de forma aleatoria; sin embargo, el algoritmo PSO difiere de un GA, ya que, el conjunto de posibles soluciones (conjunto de partículas) se conserva desde el momento que son generadas hasta concluir la prueba (Kennedy, 2010). Asimismo, el conjunto de partículas es evaluado bajo un determinado criterio de aptitud, y de acuerdo a la información obtenida de la experiencia de vuelo propia y del resto de partículas (datos derivados del desplazamiento de las partículas dentro del espacio de exploración), a cada una se le es asignada una velocidad ajustada dinámicamente para que esta continúe su vuelo por medio del espacio de búsqueda. Tras varias iteraciones, las partículas son aceleradas hacia aquellas partículas que presenten mejores valores de aptitud, convergiendo así, a una solución óptima global de manera eficiente (Shi y Eberhart, 1999).

En resumen, el algoritmo PSO determina el mejor lugar en el espacio de búsqueda, por medio del vuelo de un conjunto de partículas (población que no tienen masa ni volumen) con velocidades actualizadas de acuerdo con las experiencias que obtuvieron a lo largo de la búsqueda (İnkaya et al., 2016). Por ende, computacionalmente se logra que, las actualización de las partículas se realiza de manera paralela, los nuevos valores dependan sólo del valor anterior de la propia partícula y de sus vecinos, y que cualquier actualización se ejecute bajo los mismos parámetros determinados en la inicialización (Del Valle et al., 2008).

2.3.1.2. Descripción del método PSO

Proceso Natural de Inspiración

Una de la cosas más esplendidas y enigmáticas que guarda la naturaleza es la existencia de organismos sociales inteligentes, los cuales, mediante un movimiento grupal anejado entre sí y autoorganizado, hacen frente a las adversidades presentes para la supervivencia de su especie en los diversos ecosistemas (Lukeman et al., 2010).

Figura 12. Exhibición aérea de aves estornino.
Fuente: Ballerini et al. (2008)

Figura 13. Movimiento colectivo en bancos de peces.
Fuente: (Copyright © Chucknado y Hughes, www.flickr.com, con permiso).

Estos organismos sociales, no solo han generado paisajes visualmente fascinantes, sino que su comportamiento colectivo ha introducido varias incógnitas en el mundo científico. La mayor interrogante, en la que muchos investigadores han trabajado, es ¿cómo interactúan los organismos simples para formar sociedades inteligentes? Por ejemplo, las bandadas de aves (Fig.12) migran de forma ordenada, o giran al unísono para evadir obstáculos; mientras, que los bancos de peces se dividen para evadir a un depredador y después vuelven a agruparse, o nadan de tal forma que las corrientes marinas no los afecten (Fig.13). Pero lo que han hallado los investigadores de sugestivo en los patrones que siguen estos grupos de animales es que, no son totalmente regulables ni tampoco son aleatorios, simplemente son complejos (Sumpter, 2010).

De esta complejidad, nace un campo de estudio del comportamiento colectivo inteligente, donde se ha afirmado que cada miembro de una sociedad sigue reglas extremadamente simples de interacción entre los individuos, y la suma de estas interacciones, puede dar lugar a comportamientos grupales complejos. Aunque de la naturaleza de estas interacciones se conoce muy poco, algunos estudios han demostrado que un ave interactúa con un número fijo de vecinos (aproximadamente seis o siete), más no con todos los vecinos que están dentro del rango de cercanía de dicha ave. Otro aspecto que se conoce con certeza, es que el objetivo principal de las interacciones es conservar la cohesión del grupo para la supervivencia (Ballerini et al., 2008).

Para los estudios llevados a cabo alrededor del comportamiento colectivo de especies, como aves o peces, se han usado en gran medida simuladores tanto para recrear un modelo físico como biológico.

La primera introducción de una emulación de sistemas de partículas fue realizada por Reeves (1983), su trabajo planteaba una técnica para el modelado de objetos de clase difusa; donde las partículas se creaban, cambiaban de forma, se desplazaban dentro del entorno de modelado y desaparecían. Posteriormente, en el año 1986, Craig Reynolds propuso el *"Modelo Boid"*, que trataba de una simulación por computadora de un sistema de partículas (aves). Reynolds fijó reglas simples que otorgó al sistema una autonomía al comportamiento de las partículas y también determinó reglas de bajo nivel (separación, alineación y cohesión) a las que las partículas debían acatar para dar crear un comportamiento sin aglomeraciones y con una formación sólida. Este trabajo tenía un alto grado de complejidad, ya que la no linealidad propia de las partículas producía un comportamiento caótico, que, a su vez, generaba una retroalimentación negativa para las reglas de ajo nivel; asimismo se conoce que cada partícula para conocer la referencia de otras partículas, presentaba un nivel de complejidad computacional de $O(n^2)$, por ello, Reynolds bajó este nivel con un modelo de vecindad con intercambio de información entre partículas. Este trabajo fue un pilar fundamental para la formulación del algoritmo de optimización por enjambre de partículas (PSO) en el año 1995, dado que, Eberhart y Kennedy pretendían continuar con la optimización del trabajo de Reynold; en consecuencia, Eberhart y su equipo de trabajo incorporó el intercambio de información local por medio de la correlación de la velocidad del vecino más cercano, y alcanzaron que el enjambre convergiera de la manera acorde a lo esperado; igualmente, eliminaron variables auxiliares, e incorporaron una búsqueda n-dimensional y la aceleración por distancia. Durante las pruebas,

observaron que el modelo conceptual en realidad era un optimizador. Finalmente, descartaron los elementos innecesarios para un optimizador y consiguieron presentar un método de optimización a la comunidad científica (Sengupta et al., 2018).

Mecanismo

La implementación de un algoritmo PSO se compone de procesos iterativos y aleatorios que operan sobre un conjunto de partículas. Cada partícula representa una solución y se mantiene en vuelo dentro del espacio de búsqueda con velocidades actualizadas de acuerdo a la valoración de aptitud que obtienen durante la búsqueda.

Una partícula mantiene el registro de las coordenadas de la mejor posición que haya alcanzado hasta el momento (más conocido como registro de experiencia de vuelo o registro de condición de aptitud). El valor de aptitud se lo denomina *pbest* y es comparado con los valores alcanzados por el resto de la población; dependiendo de esta comparación, se ajusta la velocidad de la partícula hacia las localidades más prometedora detectadas hasta el momento en el espacio de búsqueda.

Una de las variantes presentes en posteriores versiones del algoritmo PSO original es la comparación local dentro de una topología de vecindad determinada inicialmente y la ampliación de aplicación a problemas binarios o discretos; ya que en la versión original estaba diseñado sólo para la optimización de problemas con valores reales (Ahmed y Glasgow, 2012).

Otro valor registrado en la versión original del algoritmo PSO es el mejor valor general alcanzado por una partícula y su posición correspondiente; esta ubicación de denomina **gbest**. Por lo tanto, a cada iteración, las partículas son evaluadas y después de una comparación, sus velocidades son actualizadas para así converger finalmente a la mejor posición.

En definitiva, los parámetros de la *i-ésima* partícula, tratada como un punto dentro de un espacio *N-dimensional* y de acuerdo a la versión original, son:

i. El vector $x_i = \{x_{i1}, \quad x_{i2}, \quad x_{i3}, \quad .... \quad , x_{in}\}$ registra la posición actual de la partícula.

ii. El vector velocidad $v_i = \{v_{i1}, \quad v_{i2}, \quad .... \quad , v_{in}\}$ registra la la dirección que toma la partícula durante su movimiento dentro del espacio de búsqueda.

iii. El vector $pBest_i = \{p_{i1}, \quad p_{i2}, \quad .... \quad , p_{in}\}$ registra la mejor posición alcanzada hasta el momento por la partícula.

iv. El vector $gBest_i = \{g_{i1}, \quad g_{i2}, \quad .... \quad , g_{in}\}$ registra la mejor posición alcanzada hasta el momento por todo el conjunto de partículas.

v. El valor $aptitud\_pBest$ registra el valor obtenido de la evaluación actual de la partícula.

vi. El valor $aptitud\_gBest$ registra el desempeño de la mejor solución global encontrada hasta el momento por todo el conjunto de partículas.

Matemáticamente, el algoritmo de PSO se describe mediante dos ecuaciones, las cuales permite la actualización del vector velocidad (ecu. 2.2) y vector posición (ecu. 2.1).

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand_1 \times (pBest_i(t) - x_i(t)) + c_2 \times rand_2 \times (gBest(t) - x_i(t)) \quad (2.1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2.2)$$

Donde:

- $v_i(t)$: Velocidad de la partícula i en la *t-ésima* iteración.
- w: Factor de inercia o peso inercial.
- c_1, c_2: Factores de aprendizaje.
- $rand_1, rand_2$: Número aleatorios entre el rango de $[0 \sim 1]$.
- $x_i(t)$: Posición actual de la partícula i hasta el momento.
- $pBest_1, gBest$: Mejor solución alcanzada por la partícula i y la mejor solución alcanzada por el enjambre.

Como se observa en la ecuación 2.1, el algoritmo PSO introduce un factor o peso de inercia para estabilizar el impacto de la nueva velocidad fijada sobre la velocidad actual, a la vez presenta un efecto sobre la destreza de exploración global; a mayor peso de inercia se facilita una mejor exploración global, mientras que a un menor peso de inercia se facilita una mejor búsqueda local. Generalmente, se inicia con un valor de peso inercial alto y se lo disminuye linealmente hasta para conseguir que al inicio de la ejecución se busque nuevas áreas prometedoras con una óptima exploración global, y cerca de finalizar se realice una exploración local que sintonice el área de búsqueda actual (Shi y Eberhart, 1999). Una manera de poder ajustar el valor de este factor con disminución gradual es de acuerdo a la ecuación 2.3.

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \cdot iter \tag{2.3}$$

Cuando se fija de forma correcta el factor inercial, no suele ser necesariamente estricta la declaración de una velocidad máxima V_{max} en la inicialización. No obstante, para controlar que el algoritmo no diverja cuando este tiende a tener una rápida aceleración, es importante hacerlo.

El límite de velocidad suele ser fijado empíricamente en base al conocimiento que tiene el programador sobre el problema. Pero se debe tomar en cuenta que, este límite no debe ser demasiado alto; ya que, causa movimientos erráticos en las partículas. Por otro lado, con un límite muy corto, también se generaría una divergencia; puesto que se limitaría demasiado el movimiento de la partícula.

Conjuntamente, C_1, C_2 son dos coeficientes de aceleración, necesarios para garantizar la convergencia del algoritmo, ya que también, el movimiento de la partícula depende de estos coeficientes. Tanto C_1 como C_2, han sido estudiados como una sola constante ($C_{1,2}$) para determinar su efecto sobre el movimiento de la partícula; tras varias pruebas, se obtuvo los patrones de movimientos que siguen las partículas con coeficiente de valores alto y pequeño, con lo que se concluyó que el valor máximo del coeficiente debe ser $C_{1,2} = 4$, esto significa que:

$$C_1 + C_2 = 4 \tag{2.4}$$

Donde C_1 y C_2 no necesariamente deben ser iguales (Del Valle et al., 2008).

En cambio, la ecuación 2.2, configura el movimiento de la *i-ési*ma partícula en cada iteración mediante la posición actual y la dirección dada por el vector velocidad. En la figura 14, se puede observar una representación esquemática del movimiento de una partícula durante la ejecución de PSO.

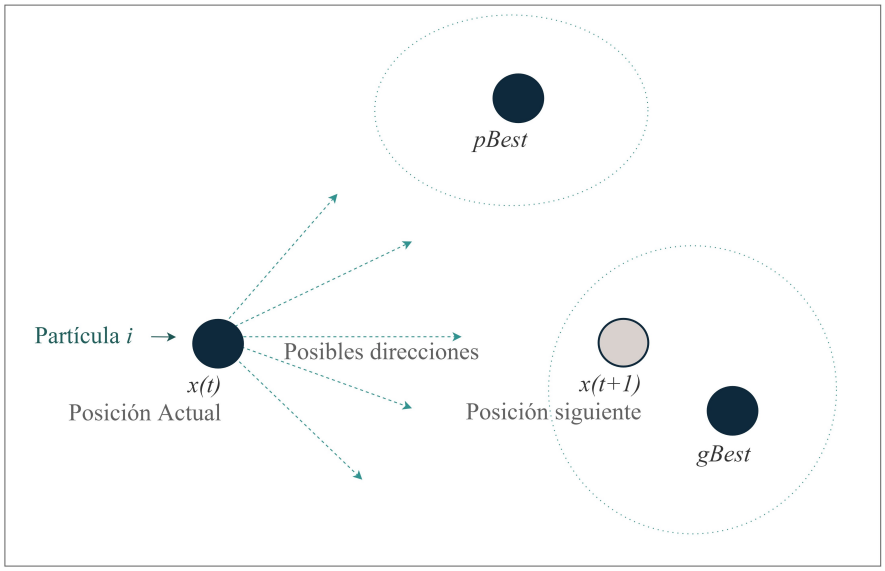


Figura 14. Movimiento de una partícula en PSO.
Representación gráfica del movimiento de una partícula en PSO dentro del espacio de búsqueda; se toma en cuenta la posición *pBest* y *gBest*.

Otro factor que aparece gracias a Clerc (1999), es el *factor de constricción* o *constante de inercia*, el cual aumenta el porcentaje de convergencia del algoritmo PSO. Datos recogidos empíricamente, demostraron que, aunque exista coeficientes de aceleración y velocidad máxima definidos de forma correcta, en problemas de naturaleza muy compleja, el algoritmo PSO puede llegar a no converger. El factor de constricción equivale a fijar el peso de inercia w como k, y que C_1 y C_2 cumplen la condición $C_{1,2}\,4$. Matemáticamente, el factor de constricción se estable de la siguiente manera (ecu. 2.7):

$$v_i(t+1) = k\left[v_i(t) + c_1 \times rand_1 \times (pBest_i(t) - x_i(t)) + c_2 \times rand_2 \times (gBest(t) - x_i(t))\right] \quad (2.5)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2.6)$$

Donde:

$$K = \frac{2}{\left|2 - C_{1,2} - \sqrt{(C_{1,2})^2 - 4C_{1,2}}\right|} \tag{2.7}$$

Cabe aclara que, cuando el algoritmo PSO fue propuesto como método de optimización, no existía una base matemática sólida; ya que, el algoritmo se originó a partir del desarrollo de un trabajo de modelamiento de organismos sociales colectivos. No obstante, de acuerdo a las variaciones realizadas a partir del PSO original, se han incorporado algunas ecuaciones para determinar ciertos parámetros. Por ejemplo, con respecto al factor de inercia, existen algunas maneras de determinar su valor (ya se un valor fijo o un valor dinámico)(Sengupta et al., 2018), dichas maneras, pueden ser:

- Selección Aleatoria
- Variación del tiempo lineal
- Variación del tiempo no lineal
- Adaptador Fuzzy

Para describir de mejor manera el mecanismo de un algoritmo PSO clásico, se presenta en la figura 15 un diagrama de flujo de procesos (Shi y Eberhart, 1999).

Otro proceso que también presenta gran robustez, es la versión PSO con vecindario local o más conocida como simplemente PSO local (*LBest*), donde la velocidad de la *i-ésima* partícula es ajustada de acuerdo con su mejor propio desempeño y el mejor rendimiento alcanzado por un vecindario con topología definida (dicha topología es determina en la inicialización) (Hu y Eberhart, 2002).

El método PSO local (*LBest*), es una versión mejora del algoritmo PSO original; ya que, a pesar que teóricamente PSO global tiene el mayor número de comunicaciones entre las partículas del enjambre; en la práctica, esto equivale a realizar el seguimiento a solo una de las mejores posiciones encontradas en el espacio de soluciones. Mientras que, en PSO local, los subgrupos de partículas del enjambre (miembros de un vecindario) tienen la habilidad de converger de manera

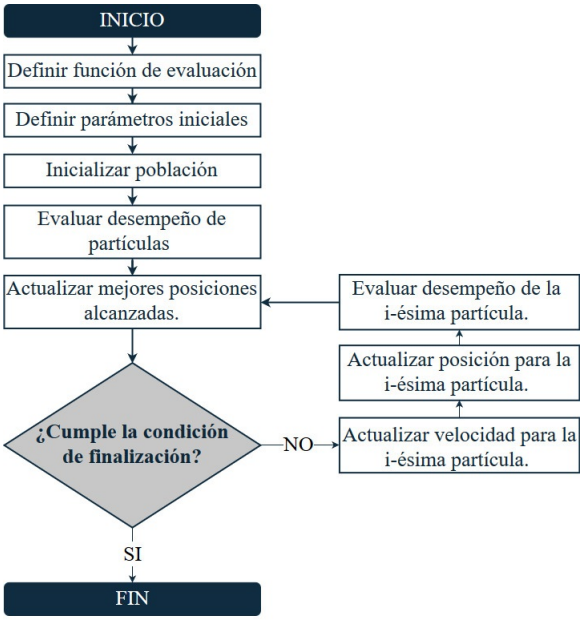


Figura 15. Diagrama de flujo del *Algoritmo PSO (Versión Original)*.

independientemente y paralela en múltiples óptimos dentro del espacio de soluciones (aunque también conlleva más tiempo de ejecución) (Kennedy, 2006).

Cuando Eberhart y Kennedy estudiaron el método PSO local, en su publicación, mostraron una comparación de tres tipos de versiones: la versión original (*GBest*) y las versiones *LBest* con dos y seis vecinos, donde concluyeron que la versión original presenta un mejor desempeño en cuanto al número de iteraciones para alcanzar la convergencia, mientras que la versión *LBest* con dos vecinos es más eficiente en no atascarse en óptimos locales (R. Eberhart y Kennedy, 2002).

En la figura 16 se representa las topologías de vecindarios más conocidas e investigadas por diferentes autores.

Si bien, varios autores destacan y recomiendan el uso de una topología de vecindario tipo estrella (Fig.16:b)), Del Valle y su equipo, en la publicación realizada sobre conceptos de optimización por enjambre de partículas, mencionan que la topología de vecindario más eficiente dependerá del problema al que se aplica el algoritmo; puesto que una estructura de vecindario puede desempeñarse de mejor manera para cierto tipo de problemas, mientras que con la misma topología puede no

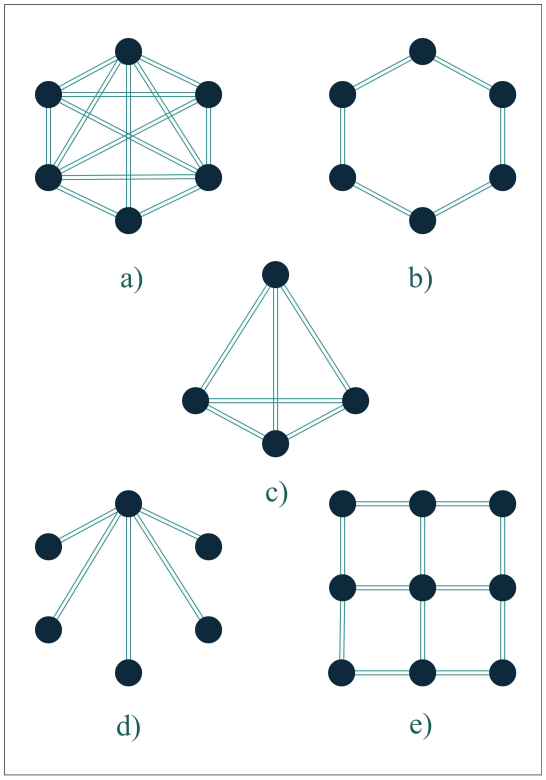


Figura 16. Topologías de vecindarios para PSO.
Topología: a) Global, b) Anillo, c) Pirámide, d) Rueda, y e) Von Neumann.

llegar a converger en una solución óptima cuando este es aplicado a problemas de diferente clase (Del Valle et al., 2008).

Consideraciones

La óptima convergencia de un algoritmo PSO depende de la perspicacia del programador; ya que, durante la programación, se debe definir de forma correcta, tanto el tamaño de enjambre de partículas, como el resto de parámetros que actúan durante la ejecución (factor de aprendizaje y de inercia).

Con un número adecuado de partículas, se garantiza entre la eficacia de la solución alcanzada y el coste computacional. En las publicaciones de Kennedy se menciona que, la mayoría de investigadores emplean entre veinte a cincuenta partículas para la formación de un enjambre (Kennedy, 2006).

Por otro lado, el factor inercial también actuará directamente en el rendimiento de búsqueda; puesto que, el algoritmo PSO suele llegar a generar velocidades con magnitudes demasiado grandes que, sin un peso inercial adecuado, el enjambre se podría desviar y nunca llegaría a converger. Con un factor inercial $w \geq 1$, implica que la velocidad aumentará de tal medida que alcanzará rápidamente la V_{max}, impidiendo que las partículas puedan cambiar su rumbo para volver a las posiciones más prometedores. En cambio, con un factor inercial $w < 1$, disminuirá la aceleración de las partículas hasta que se transforme en una función compuesta sólo por los factores de aceleración (Sengupta et al., 2018). Usualmente se emplea un factor inercial fijo entre el rango de $w \approx [0{,}5 \sim 0{,}9]$, si la naturaleza del problema al que se ha aplico el algoritmo es simple (X.-S. Yang, 2014).

En cuanto a versiones, pruebas realizadas con PSO local, se afirma que se logra alcanzar un mejor desempeño con un tamaño promedio de una vecindad de 5, en comparación al método PSO que fija aleatoriamente número de colindantes de una vecindad (Shi y Eberhart, 2004).

2.3.1.3. Trabajos relacionados

Con las diferentes modificaciones y mejoras que se han ido dando al algoritmo PSO original, varios investigadores han desarrollado trabajos aplicativos con el fin de conseguir resultados favorables. A continuación, se presenta dichos trabajos.

Todas las áreas de ingeniería, han hecho uso del algoritmo PSO para mejor y facilitar el manejo de problemas complejos. En el artículo " *Applications of Particle Swarm Optimization in Geotechnical Engineering: A Comprehensive Review*", presentan de forma resumida, la descripción de los principales estudios que han empleado el algoritmo para el manejo de probleas complejos propios de la Ingeniería Geotécnica, como túneles, espacios subterráneos, pilotes, cimientos, etc. Algunos de los problemas mencionados son: la dependencia multivariable de las repuestas del suelo y las rocas, el manejo de funciones no convexas y discontinuas, la multidimensionalidad, etc. Tras el

análisis de los resultados que cada aplicación ha conseguido, los autores recomiendan usar con precaución el algoritmo PSO para evitar divergencias en la solución (Hajihassani et al., 2018).

Por otra parte, en el manejo de datos también está presente el uso del algoritmo PSO, no sólo porque es una herramienta eficiente de búsqueda, sino que permite el manejo de gran cantidad de información. Como se presenta en el artículo " *Bayesian calibration of building energy models with large datasets*", el algoritmo PSO fue probado con un grupo de información que contenía datos importante disperso entre datos irrelevantes (ruido). La técnica desarrollada incorporaba la formación de subgrupos de información del mismo tipo para acelerar la clasificación de los mismo; a la vez lograron una reducción del costo computacional (Chong et al., 2017).

Pero a pesar que el algoritmo PSO es una herramienta potencial, la mala determinación de los parámetros de ajuste del algoritmo provocan el atasco en soluciones óptimas locales o una divergencia total del algoritmo. Por este motivo, en el artículo "Density-based particle swarm optimization algorithm for data clustering" se propone un método de estimación para fijar el valor de los coeficientes de aprendizaje. El método propuesto, implementa el cálculo de los coeficientes de aprendizaje mediante el uso de los máximos locales obtenidos durante la ejecución del ciclo de búsqueda realizado por las partículas, específicamente, el cálculo se lo realiza después de las dos primeras etapas que los investigares propusieron (inicialización y determinación de las mejores posiciones locales). Las pruebas realizadas en base a la clasificación de datos, presentó un gran desempeño superior en comparación con otros métodos. La descripción completa del método se encuentra detalla en (Alswaitti et al., 2018).

Para problemáticas actuales, varios investigadores han hecho uso del algoritmo PSO. En el artículo "*Water Distribution System Design Using Multi-Objective Particle Swarm Optimisation*" (Patil et al., 2019), los investigadores describen el diseño de un sistema de distribución de agua mediante tres modificaciones al algoritmo PSO multiobjetivo: búsqueda local, estrategia para seleccionar líder y operación de mutación modificada. Para las pruebas fueron considerada cuatro redes de referencia y después de una comparación con la aplicación de Pareto, se registra en los resultados que, aunque se obtiene nuevas soluciones durante la búsqueda, el costo computación también va en aumento (Patil et al., 2019).

En el artículo, "*The potential application of particle swarm Optimization algorithm for forecasting the air-overpressure induced by mine blasting*", se explica la importancia de la estimación de la sobrepresión del aire producida en los entornos de una mina. El cálculo de dicha estimación se lo realizó mediante la aplicación de tres variantes del algoritmo PSO, donde se hizo uso del criterio ce determinación y el error cuadrático medio como criterios de evaluación para la construcción de la función objetiva. En este trabajo, se recalca que la aplicación del algoritmo PSO es más rápida que la aplicación del algoritmo GA, y más fácil por contar con pocos parámetros de ajuste. Las pruebas se realizaron con un enjambre de 350 partículas y 450 iteraciones, un factor inercial de 0.75, y en cuanto a los coeficientes de aprendizaje, se realizaron varias pruebas con distintos valores para PSO lineal, y se fijó un valor de 2 (en C_1 y C_2) para el PSO cuadrático y de 1.5 (en C_1 y C_2) para PSO power. Finalmente, los resultados mostraron que la estimación mediante PSO lineal fue más precisa que la estimación por medio PSO power y la estimación por PSO cuadrático (AminShokravi et al., 2018).

Una de las aplicaciones, específicamente en el área de control, se publicó bajo el título "Particle Swarm Optimization Based Fuzzy Logic MPPT Inverter Controller for Grid Connected Wind Turbine". Este trabajo propone un controlador inversor de seguimiento del punto de máxima potencia para un sistema de conversión de energía eólica; ya que las fuentes presentan problemas por su naturaleza intermitente. El fin del control es lograr maximizar la eficiencia de captura de energía del sistema, para lo cual, el algoritmo PSO (en el artículo no se detallan los valores de los parámetros de ajuste) sirve para encontrar los valores óptimos de los parámetros de diseño del controlador de lógica difusa. Los resultados mejorados del sistema controlado, se muestran en base a la tensión de salida generada, análisis de onda de corriente y potencia, onda de voltaje de enlace de DC y la velocidad del generador (Parvin et al., 2019).

En combinación con otras técnicas de búsqueda y clasificación, la publicación titulada "*Quantum neural network-based intelligent controller design for CSTR using modified particle Swarm optimization algorithm*", describe el control de un sistema de reactor continuo de tanque agitado. Este sistema al ser no lineal y con una dinámica altamente sensible, presenta dificultades de rendimiento con un control de coeficientes fijos. Por este motivo, los autores Salahshour et al. (2019), proponen el uso de una red neuronal cuántica multicapas para el desarrollo de una estructura adaptativa para un controlador tipo PID. El algoritmo PSO se empleó para el entrenamiento de la red

neuronal, sin embargo, para garantizar la convergencia del algoritmo, se ha introducido una fase de eliminación inicial; en este proceso se eliminan las partículas que no han conseguido un costo aceptable en la evaluación de aptitud. Los resultados presentan un aumento en el rendimiento del sistema en comparación con el control del sistema mediante un controlador PID tradicional (de un costo de 0.034912 a un costo de 0.032665).

2.3.2. Colonia de Abejas Artificiales

2.3.2.1. Preámbulo

Los intentos por hallar nuevos métodos para la resolución de problemas de naturaleza compleja inspirados en inteligencia colectiva, ha ido más allá del estudio del comportamiento colectivo de bandadas de aves o escolarización de peces. Varios investigadores, han tomado como modelo a la autoorganización social de diferentes insectos como hormigas, termitas, avispas, y sin duda a las enigmáticas abejas. Perteneciente a la familia Apidae, las abejas melíferas (Fig. 17), han sido uno de los insectos que ha presentado mayores habilidades a nivel de colonia gracias a que cada individuo, dentro de la colmena, cumple una determinada actividad de acuerdo al su casta y edad. Por ende, el comportamiento colectivo altamente organizado de las abejas ha sido la base principal para el desarrollo un algoritmo orientado a la resolución de problemas de optimización N-dimensional y multimodal.

El *Algoritmo Colonia de Abejas Artificiales*, conocido como algoritmo ABC (por sus siglas en ingles de "Artificial Bee Colony"), ha sido considerado unas de las herramientas de optimización más simples, flexibles y robustas. Por su capacidad de exploración y explotación, esta herramienta ha sido ampliamente usada para diversos problemas de la vida real en diferentes áreas de aplicación; esto en comparación con otros algoritmos basados en el enjambre de abejas (Bitam et al., 2010; Mernik et al., 2015).

La autoorganización y la división de trabajo, fueron las dos propiedades inherentes del comportamiento inteligente de las abejas que, el investigador Karaboga consideró necesarias y suficientes para obtener un sistema de resolución de problemas distribuidos. En su modelo de colonia artificial, fueron implementadas tres clases de abejas: empleadas, observadoras y exploradoras; donde

Figura 17. Abejas melíferas de la familia Apidae.
Fuente:(Copyright © Bob Peterson in Swamp, www.flickr.com, con permiso).

por cada fuente de alimentación existe una abeja empleada. La abeja empleada cambia su papel a abeja exploradora sólo cuando la fuente de alimentación se haya agotado, y se convierte en exploradora, mientras que las abejas observadoras son quienes determinan si la fuente aun cuenta con un nivel suficiente de néctar para seguir con la explotación; análogo a este proceso, fue desarrollado el algoritmo de optimización ABC que resultó ser muy simple y flexible (Karaboga, 2005).

A partir de la propuesta de Karaboga, muchos investigadores han puesto en prueba la efectividad de resolución del algoritmo de optimización ABC, y a pesar que casi siempre se ha obtenido bueno resultados, se han realizado algunas modificaciones al algoritmo original con el fin de garantizar la convergencia hacia una solución óptima. Actualmente, Xiang y su grupo de trabajo han propuesto un algoritmo de optimización ABC mejorado basado en el modelo de gravedad (Xiang et al., 2018); Kong y su equipo, propusieron el método ECABC que se basa en la orientación de un grupo de élite y la estrategia de búsqueda combinada de profundidad (Kong et al., 2018); Dunwei Gong y colaboradores, plantearon el método híbrido ABC discreto multi-objetivo HDABC, para el problema de programación de una tienda de flujos BLSFS (siglas del inglés: Blocking Lot-Streaming Flow Shop) (Gong et al., 2018). También, (Taetragool et al., 2018) propusieron una versión mejorada de ABC, donde ejecutan un responsabilidad dinámica de las abejas, el registro de satisfacción de la propia abeja en cada nido y un nuevo mecanismo que aumenta la convergencia

del mismo; (Kumar et al., 2019) proponen el algoritmo Arrhenius ABC (aABC) con el objetivo de mejorar el estabilidad entre la diversificación y la intensificación del algoritmo de optimización ABC original; e indudablemente Karaboga y su equipo también presentan una mejora en el algoritmo de optimización ABC en la parte explotación de la fuente de alimentos, que a la vez aumenta el rendimiento de convergencia temprana sin afectar la calidad de las soluciones finales (Aslan et al., 2019).

Después de esta breve introducción, se presenta de manera más detallada el funcionamiento y datos generales del algoritmo de optimización colonia de abejas artificiales.

2.3.2.2. Descripción del método ABC

Proceso Natural de Inspiración

La vida de una colmena de abejas se podría decir que es casi perenne, gracias a su autoorganización social. La observación científica aplicada a este sistema, ha conducido a la comunidad científica a realizar cientos de estudios para determinar como un ente con capacidades tan reducidas puede conformar una colectividad inteligente.

Existe varias especies de abejas clasificadas de acuerdo a diversos factores; pero, para los estudios mencionados sobre comportamiento colectivo, las colmenas de las abejas de la especia Apis melífera, han sido el principal modelo de observación. En efecto, se conoce que una colmena de abejas está compuesta por una sola abeja reproductora (abeja reina), unos cientos de abejas machos (zánganos), cientos de larvas y miles de abejas obreras, las cuales cumplen varias labores de acuerdo a su periodo de vida.

La abeja reina, al ser la única reproductora en la colmena, es quien pone miles de huevos, de los cuales, la mayor parte nacerán abejas no fértiles que se convertirán en las abejas obreras, de la misma manera, nacerán algunos zánganos, y en una sólo ocasión nacerá una abeja fértil (una nueva reina).

En la colonia de abejas melíferas, presenta un conjunto de actividades cooperativas, que les permite ser una organización inteligente. Estas tareas varían ente producción de su propio alimento, construcción de nidos, matrimonios, exploración, etc. Particularmente, una de las actividades que

realiza una abeja obrera adulta es el forrajeo de néctar; las abejas exploran las zonas cercanas a la colmena, con el fin de encontrar una fuente de alimento prometedora y luego poder atraer más abejas hacia dicha fuente para recolectar el néctar de las flores. Asimismo, las abejas hacen la búsqueda de nuevos nidos. Cuando la abeja reina ha generado una nueva abeja reina, antes de nacimiento, la abeja reina madre abandona la colmena con la mitad de la colonia pera iniciar con ellas una nueva colmena en otro sitio. Es así que, unas cuantas abejas salen a explorar distintos lugares con el objetivo de encontrar el lugar más apropiado para el nuevo nido.

De estas exploraciones, nace otra peculiaridad con la que cuentan las abejas, que es la precisión en la comunicación. Las abejas han desarrollado un sistema de toma de decisiones mediante la comunicación. Este intercambio de información lo realizan mediante el baile, donde, de acuerdo al tipo de giro, la velocidad, intensidad o repetitividad de ejecución de algún movimiento, las abejas, indican distancia y posición de la zona en la cual se encuentra la nueva fuente de alimento o el lugar donde se asentará la nueva colmena (nido) (Kirchner et al., 1988). Es así que, a pesar que las abejas tienen diferentes juicios respecto a una fuente de alimentación o el nuevo lugar para el nido, siempre seleccionan el mismo sitio.

A pesar que, existen otras actividades con la misma importancia que el forrajeo, que también, han servido de modelo para múltiples algoritmos bio-inspirados; existe un mayor porcentaje de investigaciones inspiradas en el comportamiento de forrajeo de las abejas. Dentro de este grupo, existe dos subgrupos que abarcan los estudios en base a búsqueda de fuente de alimentos, y los estudios en base a búsqueda de nuevo nido. Los algoritmos basados en el forrajeo de fuentes de alimento, aunque tienen el mismo fundamento, los investigadores han propuesto algoritmos que se diferencia en la proyección, el objetivo o la aplicabilidad (Bitam et al., 2010).

En el modelo de Karaboga, usa a toda la colonia como abejas obreras y las divide en tres grupos diferentes, los cuales trabajan cooperativamente. El primer grupo lo conforman las abejas empleadas, las cuales cumplen la función de trasladar el néctar de la fuente a la colmena; a la vez, comunican detalles acerca de la calidad del néctar, la distancia a la que se encuentra la fuente explotada, con el segundo grupo de abejas observadoras. Este segundo grupo, quienes permanecen en la colmena, son las que deciden cual será la fuente a ser explotada, pero no sin antes verificar la información recibida por parte del primer grupo. El proceso de selección de la nueva fuente de alimentación es probabilístico, puesto que tiene mayor posibilidad de ser selecciona una fuente,

en función con su calidad de néctar. Si la calidad del néctar es alta, contará igualmente con una alta preferencia por parte de las abejas observadoras. Posterior a selección de la fuente, las abejas observadoras se pasan a conformar el grupo de exploración y explotación de la fuente seleccionada. Finalmente, el tercer grupo, conformado por las abejas explotadoras, son quienes buscan fuentes no visitadas de manera aleatoria; este grupo también permanece dentro de la colmena hasta que son motivadas a salir en busca de nuevas fuentes de alimentación en zonas no exploradas. A este grupo, pueden incorporarse las abejas empleadas, sólo cuando su fuente de alimentación se ha agotado (Aksoy et al., 2018).

El ciclo se mantiene constantemente, hasta que se cumpla la condición de parada; en este caso, hasta que se cumplan el número de iteraciones establecidas por el programador.

En síntesis, Karaboga hizo uso de las cuatro propiedades inherentes de la autoorganización de una colonia de abejas, las cuales son:

1. Retroalimentación positiva
2. Retroalimentación negativa
3. Fluctuaciones
4. Interacciones múltiples

La retroalimentación positiva hacer referencia al comportamiento de una abeja que promueve a otras a juntar esfuerzos (reclutamiento), mediante el baile. La retroalimentación negativa, en cambio, controla el patrón colectivo; evita las acumulaciones de las abejas en una misma actividad. Las fluctuaciones se pueden observar en los comportamientos aleatorios de las abejas, que son importantes para generar innovación en la colonia. Y las interacciones múltiples corresponde al grupo mínimo de abejas que cumplen reglas básicas para realizar sus propias actividades (Akay y Karaboga, 2012). De donde resulta que, las abejas del primer y segundo grupo (abejas empleadas y observadoras) ejemplifican la retroalimentación positiva y las interacciones, mientras que, el tercer grupo de las abejas exploradoras introducen la retroalimentación negativa y las fluctuaciones.

Mecanismo

El objetivo del algoritmo de optimización ABC es encontrar una fuente de alimentación con la mayor cantidad de néctar. equivalente a este fin, en la primera versión, Karaboga define a cada fuente de alimentación como una solución en un vector *N*-dimensional, la cantidad de néctar como el nivel de aptitud y cada abeja equivale a los operadores de variación; ya que, al desplazarse desde la colmena hacia la una nueva fuente de alimentación, calculan la posición de una nueva solución. Al igual que el algoritmo PSO, ABC registra la ubicación de la última mejor fuente visitada; cuando hace una visita a una nueva fuente, que es mejor que la última registrada, esta ubicación remplazará a la anterior.

Detalladamente y en términos matemáticos, el algoritmo de optimización ABC ha sido implementado de la siguiente forma:

- *Producción de fuentes de alimento*
  Inicialmente, el algoritmo ABC, genera aleatoriamente las fuentes de alimentación dentro del espacio de búsqueda. Se considera los límites de los parámetros establecidos para esta etapa de inicialización (ecu.2.8).

$$x_{ij} = x_j^{min} + rand\,(0,1)\left(x_j^{max} - x_j^{min}\right) \quad para \quad i = 1,...,SN, \quad j = 1,...,D \tag{2.8}$$

  Donde:

  - *SN*: es el número de fuentes.
  - *D*: es el número de parámetros de optimización.

  En esta etapa de inicialización, luego de haberse generado las fuentes de néctar, los contadores que registran el número de pruebas se establecen en cero; e inicia a un proceso repetitivo de búsqueda por los tres grupos de abejas sobre las fuentes creadas, hasta que se cumpla el número de iteraciones máximas establecidas o cumpla la tolerancia de error.

- *Asignación de abejas empleadas a las fuentes de alimentación*
  Para cada fuente, se asigna una abeja empleada, por ende, el número de abejas empleadas

es igual al número de fuentes de alimento. Cuando la abeja empleada se encuentra en la fuente de alimento, esta modifica la posición registrada de la fuente (modifica la solución), con el fin de hallar una fuente de alimento vecina. Esto se lo realiza a través de ecuación 2.9. Posteriormente, las abejas evalúan la calidad del néctar de la nueva fuente y comparan con la fuente que les fue asignada inicialmente para elegir entre una de las dos fuentes; dicha comparación se la realiza en base al costo fijado por la función de evaluación.

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{2.9}$$

Donde

- x_i es la ubicación actual de la fuente de alimento registrada en memoria.
- j es un número aleatorio en el rango de $[1 \sim D]$.
- k es el índice aleatorio en el rango de $\{1, 2, ..., SN\}$, pero diferente a i.
- ϕ es un número aleatorio real en el rango $[-1 \sim 1]$.

Con la ecuación 2.9, al mismo tiempo se logra que mientras la búsqueda se acerque a la solución óptima, la longitud del paso disminuya de manera adaptativa, pues a media que la diferencia entre x_{ij} y x_{kj} disminuyen, la perturbación en x_{ij} se reduce (Akay y Karaboga, 2012).

Varios estudios afirman que las abejas empleadas y observadoras realizan una búsqueda de manera efectiva por medio del esfuerzo bien coordinado, puesto que estos dos grupos de abejas realizan un proceso de búsqueda similar.

- *Cálculo de la probabilidad para toma de decisiones*

  Cuando las abejas empleadas regresan a la colmena, comparten información con las abejas observadoras, relacionada con el nivel de néctar con el que cuenta la fuente visitada y la posición de la misma (interacción múltiple). Las abejas observadoras, seleccionan probabilísticamente una fuente de acuerdo a la evaluación de la información recibida. Esta probabilidad de selección depende específicamente de la cantidad de néctar de la fuente, y se calcula mediante la ecuación 2.10.

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{2.10}$$

- Selección de la posición de la fuente de alimentación
  Durante la ejecución del algoritmo ABC, se genera un número aleatorio en el rango $[0 \sim 1]$ para cada fuente; si el valor de probabilidad p_i (ecu.2.10) es mayor que el valor aleatorio, la abeja observadora realiza una modificación en la posición de la dicha fuente de alimento a través de la ecuación 2.9, similar al proceso que realizan las abejas empleadas. Consecutivamente, se evalúa la fuente de néctar, y dependiendo del resultado, la abeja observadora registra la nueva posición o no.

- *Criterios de abandono*
  Después de cada ciclo, se realiza la verificación del criterio de abandono de una fuente. Para ello se hace uso de los contadores utilizados durante la búsqueda. Cuando el valor del contador supera el valor del parámetro de control (conocido como "limite"), la abeja abandona la fuente y se dirige a una nueva fuente hallada por una abeja exploradora (retroalimentación negativa y fluctuaciones). El algoritmo ABC lo simula mediante la producción de una nueva posición aleatoria y se la remplaza con la posición abandonada (se actualiza x_i).

Para mejor compresión del proceso de ejecución en un algoritmo de optimización ABC, se presenta un diagrama esquemático de los procesos que se ejecutan en el algoritmo ABC (fig.18, fig.19, fig.20, fig.21).

Consideraciones

Un estudio realizado por Hussain y su equipo, concluyeron que el algoritmo de optimización ABC tiene un rendimiento proporcionalmente inverso al comportamiento de las abejas exploradoras; ya que, continuamente generan soluciones independientemente del avance de la búsqueda. Es decir, el comportamiento de las exploradoras puede causar una divergencia durante la ejecución de del algoritmo. Esto sucede principalmente cundo el algoritmo ABC es aplicado a problemas restringidos, funciones compuestas o funciones no divisibles (Akay y Karaboga, 2012; Hussain et al., 2018).

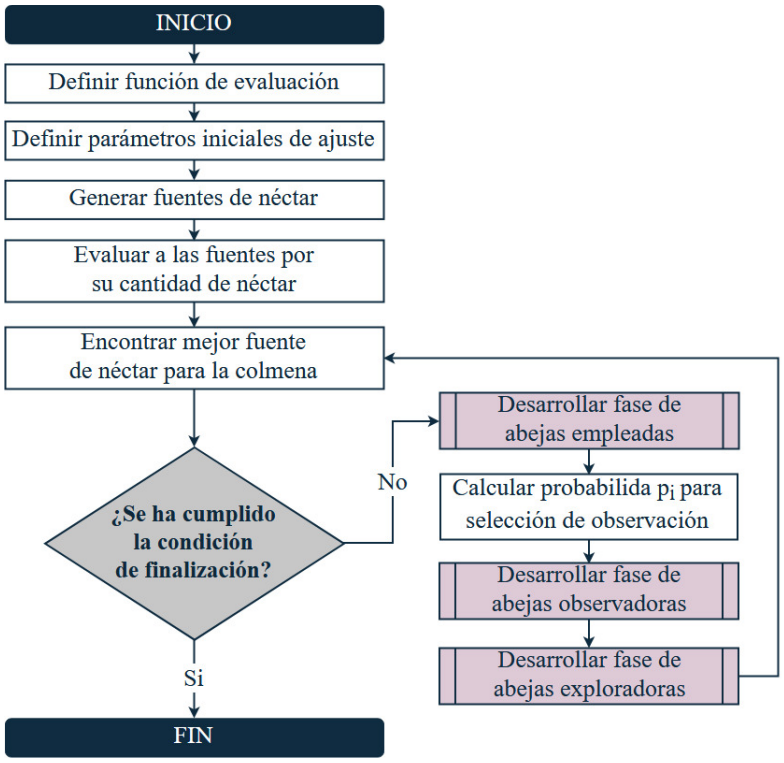


Figura 18. Diagrama de flujo del *Algoritmo ABC (Versión Original)*.

Figura 19. Diagrama de flujo del proceso de las *Abejas Empleadas*.

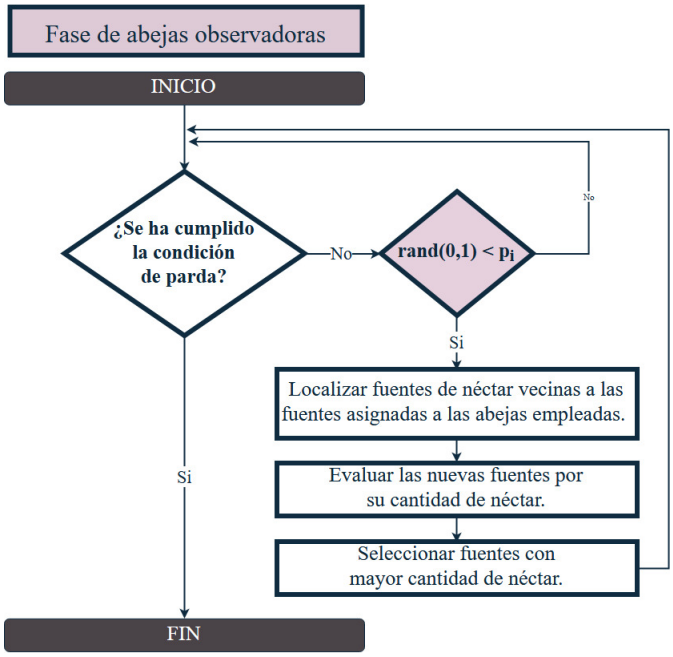


Figura 20. Diagrama de flujo del proceso de las *Abejas Observadoras*.

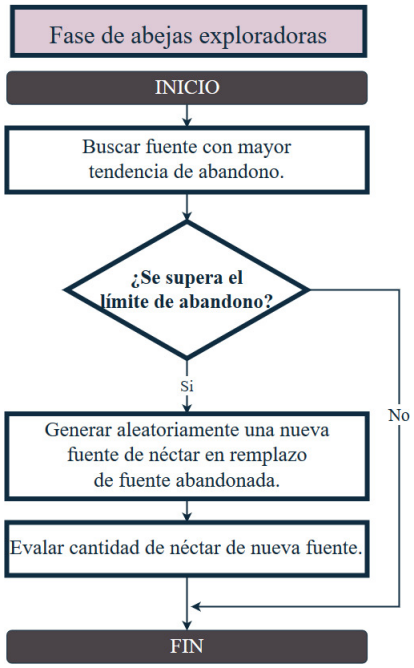


Figura 21. Diagrama de flujo del proceso de las *Abejas Exploradoras*.

En la versión original, se debe tener en cuenta que sólo se puede agotar una fuente de alimento en cada ciclo, pero si más de un contador supera el valor límite, se deberá preferir el abandono de la fuente con el valor más alto en su contador mediante programación.

2.3.2.3. Trabajos relacionados

El algoritmo de optimización ABC ha sido catalogado como un método eficiente y fácil de implementar, por lo que, varios investigadores lo han prefiero, por encima de otros algoritmos basados en abejas. Su uso en diferentes problemas de optimización ha dado buenos resultados, por lo que ha ganado popularidad en los últimos años. A continuación, se da a conocer algunos de los trabajos aplicativos más actuales en referencia a este algoritmo de optimización bio-inspirado. Y especialmente, se presentará los estudios aplicativos enfocados a la sintonización de controladores.

Con el objetivo de mejorar el contraste de una imagen, la cual pierde calidad durante la captura de la misma, un grupo de investigadores (Chen et al., 2018), consideraron el uso del algoritmo de optimización ABC para solventar este problema. Ellos, propusieron una nueva función de aptitud para la evaluación de las posibles soluciones y lograron una guía del movimiento de búsqueda de las abejas artificiales. En su artículo publicado, describen que, el uso del algoritmo ABC fue considerado ya que una de las ventajas de este algoritmo es que tanto la búsqueda global como local se ejecutan a la vez durante el ciclo. Como resultado, obtuvieron imágenes mejoras en términos de calidad visual y medición objetiva de desempeño.

En el artículo “Dynamic data clustering by combining improved discrete artificial bee colony algorithm with fuzzy logic”, se presenta la aplicación del algoritmo ABC discreto para la agrupación de datos según su similitud. El problema a enfrentar, era el manejo de datos multidimensionales, por lo cual, los autores de este trabajo propusieron un método difuso innovador con una colonia de abejas artificiales discretas. La implementación se lo realizó mediante la herramienta Matlab y las pruebas se realizaron con cinco conjuntos de datos referenciales disponibles en la base de datos de aprendizaje automático; se consideró costo mínimo máximo, promedio y porcentaje de la taza de error. Consecuentemente, con el algoritmo propuesto, consiguieron superar los resultados obtenidos de otros métodos empleados en agrupación de datos (Dehkordi y Amiri, 2018).

Otra propuesta se presenta en el artículo "A novel artificial bee colony algorithm for the workforce scheduling and balancing problem in sub-assembly lines with limited buffers", donde los autores (Yurtkuran et al., 2018), con el fin de determinar la mínima fuerza de trabajo necesaria para procesar lotes divididos en las estaciones de ensamble, proponen un algoritmo mejorado de ABC discreto con reglas de aceptación de solución y búsqueda múltiple. El problema al que se enfrentaron, fue considerado NP-duro, por lo que se diseñó las mejoras al algoritmo ABC. Los resultados alcanzados en las más de 30 pruebas realizadas con problemas diferentes, fueron mejores que los obtenidos mediante la aplicación de los algoritmos PSO, DE y una variante de ABC discreto. Una de las ventajas obtenidas de algoritmo mejorado ABC, es el aumento de la capacidad de convergencia, y la eficiencia y eficacia del coste computacional.

Dentro de área social, los autores (Martín-Moreno y Vega-Rodríguez, 2018), publicaron su trabajo sobre el algoritmo de optimización ABC, aplicado al diseño de rutas turísticas, bajo el título "Multi-Objective Artificial Bee Colony algorithm applied to the bi-objective orienteering problem". En esta publicación, se describe las modificaciones realizadas al algoritmo ABC original para poder aplicarlo a problemas multiobjetivo (MOABC). La implementación fue realizada en C++ y comparada con los algoritmos P-ACO (Pareto Ant Colony Optimization) y P-VNS (Pareto Variable Neighborhood Search). Para dicha comparación, se ejecutó cada algoritmo con el mismo número de iteraciones y se fijó indicadores de rendimiento. Finalmente, obtenidos los resultados de los indicadores y después de un estudio estadístico, se determinó que MOABC presentaba un mejor rendimiento ante problemas multiobjetivo.

Como se puede apreciar, la aplicación del algoritmo ABC tanto en su versión original como sus variantes, son aplicada en muchas áreas investigativas. Ahora, se presenta los trabajos más recientes aplicados específicamente a sintonización de controladores.

El artículo publicado con el título "Fractional PID Controller Design for Fractional Order Systems using ABC Algorithm", expone sobre el rendimiento de los controladores de orden fraccional PID (FOPID), sintonizados mediante el algoritmo de optimización ABC. Para la evaluación de rendimiento, se utilizaron cuatro funciones de evaluación con diferentes características de respuesta ante pasos críticos. Los autores (Senberber y Bagis, 2017), presentaron tablas comparativas entre los controladores FOPID y PID sintonizados mediante algoritmo ABC, y evidencian que el uso

del algoritmo ABC favoreció en las especificaciones de rendimiento. En su publicación se puede encontrar la información detallada de los valores que alcanzaron cada uno de los dos controladores.

Otra aplicación se presenta en el artículo "A General Technical Route for Parameter Optimization of Ship Motion Controller Based on Artificial Bee Colony Algorithm". Los investigadores (Tian et al., 2017), se enfocan en la optimización global de los parámetros del controlador PID mediante la aplicación del algoritmo ABC; estos parámetros son equivalente a la fuente de néctar, y el índice integral de tiempo multiplicada por el valor absoluto de error (ITAE por sus siglas en inglés) conformó la función de evaluación. En la implementación, generaron una población de 30 abejas artificiales, 15 fuentes de alimento y definieron un proceso de iterativo de 100 ciclos. Los resultados fueron favorables; ya que, consiguieron una alta velocidad de seguimiento, un sobre impulso menor al 10 % y una mejora en el rendimiento $(ITAEmin = 11{,}01J)$.

Un ajuste en un control de frecuencia, se lo pude encontrar en la publicación "Load frequency control by de-loaded wind farm using the optimal fuzzy-based PID droop controller". El controlador fue diseñado para el manejo de carga para sistemas de energía en parques eólicos y a la vez el manejo de la cooperación sincrónica de unidades térmicas para evitar poner en riesgo la inercia del sistema. El algoritmo ABC es aplicado para determinar el valor de los parámetros de las funciones de pertenencia, pues de estas funciones dependerá el rendimiento del controlador fuzzy. En la etapa de inicialización, se describe la composición de una matriz como el espacio de búsqueda, donde el número de filas es equivalente a la suma de las abejas observadoras y empleadas, y las columnas es igual al número de parámetros desconocidos, es decir una matriz $x_{espacio \quad de \quad bsqueda} = \left[x_{ij}\right]_{m\times n}$. Durante la ejecución, para cada fila de la matriz (espacio de búsqueda), se calcula la respuesta de frecuencia y se verifica el valor de la función de costo. De manera concluyente, se consiguió el diseño de un controlador difuso capaz de controlar un sistema de energía en comparación con la incapacidad de los controladores clásicos; puesto que con el controlador fuzzy se logra la modificación repentinamente la inyección de potencia activa, que, a la vez, permite que la curva de frecuencia caiga gradualmente (Abazari et al., 2018).

Para el control de manipuladores manejados por cables, se presentó una publicación titulada "Time delay control of cable-driven manipulators with artificial bee colony algorithm" (F. Yan et al., 2018). En base a las ventajas que presenta esta práctica (manejo por cables de manipuladores),

los autores realizaron un estudio del control de retardo de tiempo, en el cual se ha implementado una sintonización del controlador PID mediante la aplicación del algoritmo ABC, para hacer frente a los valores muy reducidos de dichos parámetros originados como medida de compensación a cambios dinámicos desconocidos. Con la implementación, se espera que los manipuladores puedan buscar trayectorias específicas con el mínimo consumo energético, es decir que, el error de seguimiento y la salida de control de los motores sean bajos. Para cumplir con su objetivo, se hizo uso de la función del error absoluto integral como función objetiva o función de evaluación. Las pruebas se realizaron en tres casos diferentes: para el caso uno, se ejecutó el controlador con estimación de retardo de tiempo en un rango de $[0 \sim 20]$ para los parámetros de PID; para el caso dos, el controlador sin estimación de retardo de tiempo en un rango $[0 \sim 20]$ para los parámetros de PID; y para el caso tres, el controlador sin estimación de retardo de tiempo en un rango de $[0 \sim 200]$ para los parámetros de PID. En consecuencia, el caso uno obtuvo el valor más apto de aptitud, seguido del cas tres con una diferencia de aproximadamente 0,01, pero el valor de adecuación final en el primer caso tres, es más bajo que en caso uno. En definitiva, los autores afirmaron que el control de retardo de tiempo que incluye el método de estimación de retardo de tiempo es altamente efectivo al considerar el error de seguimiento y la salida del controlador.

Capítulo 3

Diseño de Controladores Bio-Inspirados

En este capítulo se presenta la descripción del diseño de controladores tipo PID, mediante la aplicación de algoritmos bio-inspirados para el ajuste de ganancias de los mismos. En concreto, este apartado tiene como objetivo presentar el uso de los algoritmos de optimización *Enjambre de Partículas-PSO* y *Colonia de Abejas Artificiales-ABC* como dos métodos de sintonización de controladores tipo PID programados en entorno de desarrollo MATLAB.

Además, previo a la descripción de los métodos propuestos, se detalla el ajuste de ganancias de un controlador mediante Algoritmos Genéticos, con el fin de generar una base de comparación para los resultados que se obtendrán tras la implementación de dichos controladores.

3.1. Modelo de la Planta

Con el propósito de verificar el desempeño real de los controladores sintonizados mediante algoritmos bio-inspirados, se ha planteado el uso del módulo *PCT-2*, disponible en el laboratorio de instrumentación de la Universidad de las Fuerzas Armadas – ESPE, para el control de temperatura de flujo de aire de dicho módulo, a través de la implementación de los controladores propuestos. A continuación, se describe los aspectos fundamentales de la planta.

3.1.1. Descripción General

El módulo PCT-2 es un dispositivo experimental que dispone de un sistema de generación de flujo de aire caliente. La estructura del dispositivo permite el control de la temperatura del aire que circula a través de un conducto. En la figura 22, se presenta el módulo PCT-2 que se dispone para el presente trabajo; y en el figura 23 se presenta el modelo físico de sistema de temperatura del módulo.

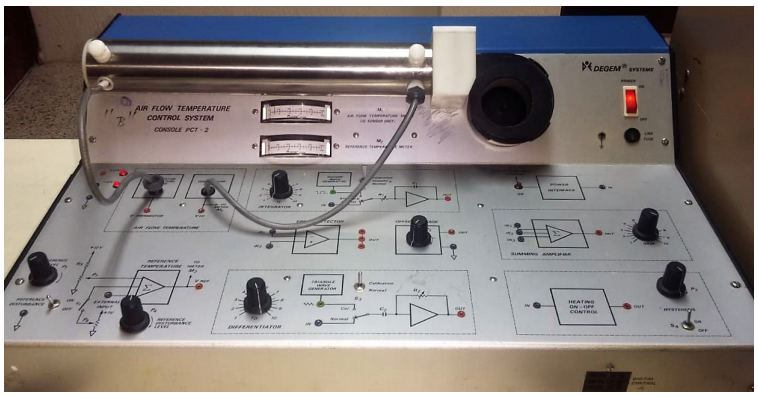


Figura 22. Módulo PCT-2 marca DEGEM SYSTEMS.

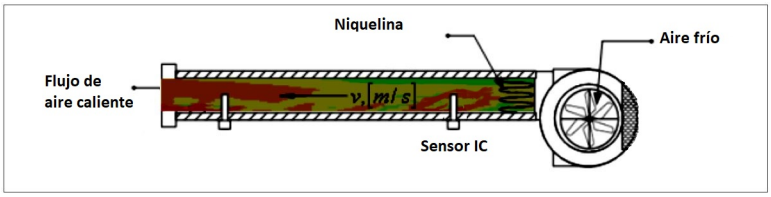


Figura 23. Proceso térmico del módulo PCT-2.
Fuente: Miniguano (2008)

Como se puede observar en las figuras anteriores, el sistema consta de: un sensor de temperatura tipo IC de estado sólido, un conducto de flujo de aire, un ventilador que permite la circulación del aire dentro del conducto y una niquelina (resistencia) encargada de calentar el aire que ingresa al conducto en un rango de $[20-70]^{\circ}C$ aproximadamente; la variación de la temperatura dentro del conducto dependerá principalmente de la tensión $V(t)$ que alimente a la niquelina.

3.1.2. Componentes del Sistema

Los componentes principales del sistema de flujo de aire con temperatura controlada, son:

- ***Elemento primario:*** Sensor-transmisor de temperatura tipo IC, con salida analógica de tensión en el rango de $[0-5]V$.
- ***Controlador:*** Microcontrolador *Teensy 3.6*. Sus pines de entrada y salida análogos están encargados de manejar tanto la señal del sensor, como la señal de control.
- ***Actuador:*** Módulo experimental PCT-2 marca *Degem Systems*. Su calentador eléctrico de temperatura (la cual varía dependiendo de la señal de control en un rango de $[0-5]V$, simula un proceso de control de temperatura.

3.1.3. Función de Transferencias de la Planta

Para la obtención de modelo matemático del flujo de aire caliente, se ha realizado la toma de datos de la variación de la señal de salida en función del tiempo ante una entrada escalón. Dichos datos, se han procesado mediante la aplicación *System Identification*, propia del software *MatLab*, para la obtención de la función de transferencia.

Previo a esto, se obtuvo la ecuación 3.1, que representa del funcionamiento del sensor IC de estado sólido ante los cambios de temperatura:

$$y = 10x + 20 \tag{3.1}$$

Donde "y" es la temperatura medida y "x" el voltaje de salida del sensor. Esta ecuación se obtuvo mediante la interpolación de los datos registrados en la Tabla 1.

Tabla 1
Respuesta del sensor tipo IC del módulo PCT-2.

Temperatura $[°C]$	**20**	**30**	**40**	**50**	**60**	**70**
Voltaje de salida $[V]$	0	1	2	3	4	5

Una vez determino el funcionamiento lineal del sensor, se ha procedido al muestreo de la señal de salida de la planta por un periodo de 10 minutos y un registro de datos a intervalo de 1 segundo. Los datos fueron: salida de tensión del sensor IC y temperatura marcada.

Siendo sometida a una entra escalón de 3.3 V, se obtuvo la siguiente gráfica:

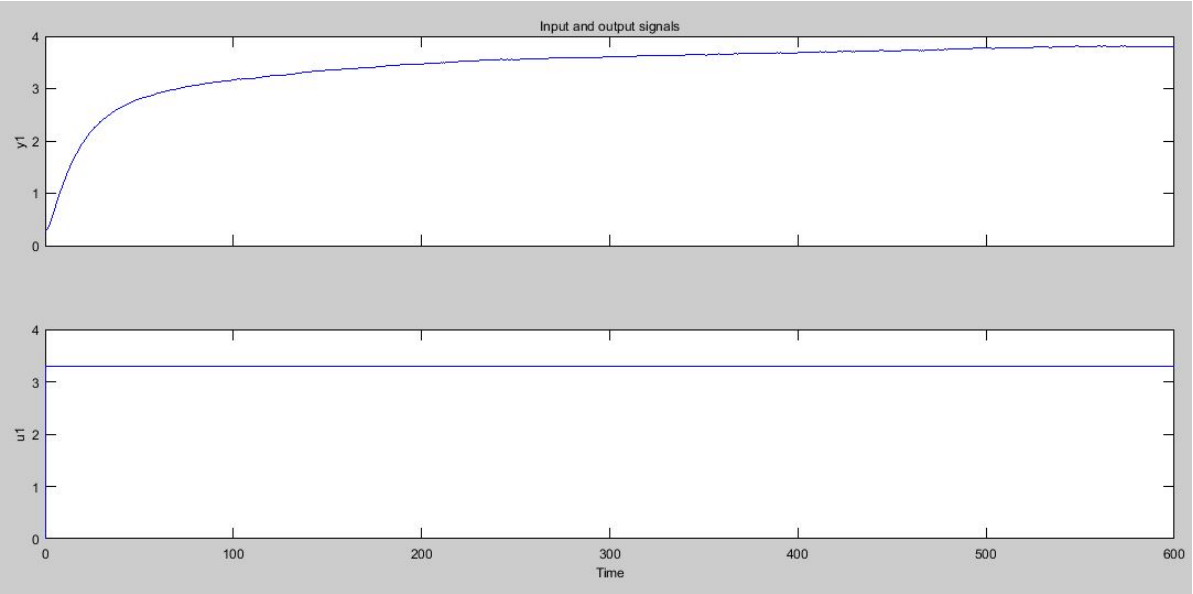


Figura 24. Toma de datos de la salida del sistema ante una entrada escalón de 3,3V.

Mediante la interfaz *System Identification* se realizó la estimación del modelo matemático con la toma de datos anterior. Para considerar todas las posibles opciones, tomando en cuenta el menor grado posible de la función y el mayor acercamiento a la planta real, se hicieron cuatro estimaciones (variando el número de polos y considerando la presencia de un cero); en consecuencia, se logró obtener los datos expuestos en la tabla 2.

Tabla 2
Exactitud obtenida por las funciones generadas para diferentes casos.

Caso	Modelo	Exactitud
Caso 1	1 polo	1 cero 76,65 %
Caso 2	2 polos	1 cero 96,67 %
Caso 3	3 polos	1 cero 97,14 %
Caso 4	1 polo	0 cero 76,43 %

Las estimaciones desarrolladas (correspondientes a la tabla 2), se presentan gráficamente en la figura 25.

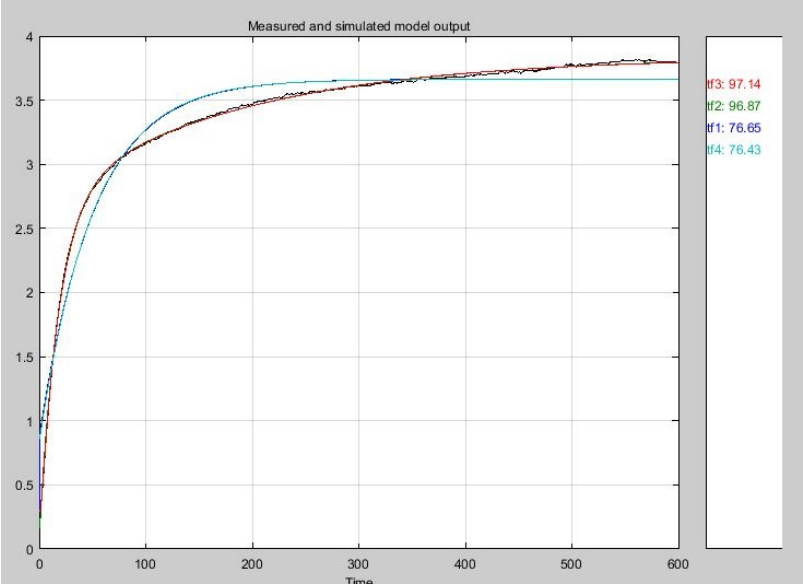


Figura 25. Comparación entre el sistema real y los modelos matemáticos generados.

En función de la comparación de los resultados observados en la tabla 2, y optando por la función con menor grado y mayor porcentaje de aproximación, se ha selecciona la función de transferencia asociada al segundo caso. La función de trasferencia es:

$$TF = \frac{0{,}04644s + 0{,}0003349}{s^2 + 0{,}0593s + 0{,}0002877} \tag{3.2}$$

3.2. Desarrollo de Controladores

3.2.1. Objetivos de Control

Para el diseño adecuado de los controladores y la validación de los mismo, se han planteado un número mínimo de parámetros a alcanzar por el sistema (objetivos de control), los cuales se detallan en la tabla 3:

Tabla 3
Objetivos de control

Parámetros	**Especificaciones**
Temperatura de salida	$35°C$
Sobreimpulso	$< 20\,\%$
Tiempo de establecimiento	$< 20seg$
Error de estado estable	$2\,\%$

Entonces, al encontrarse la salida del sensor entre un rango de $[0-5]V$, y al requerir una temperatura de $35°C$, se realiza la respectiva conversión para obtener el valor de referencia en voltaje (setpoint). Por ende, partiendo de la ecuación 3.1, determinamos el valor de x:

$$x = 0{,}1y - 2 \qquad donde \quad y = 35;$$
$$\therefore \quad x = 1{,}5$$

Siendo así, el valor de la referencia igual a $1{,}5V$ (valor que la variable controlada debe alcanzar).

3.2.2. Sintonización Clásica

Entre los método más difundido y utilizado para la sintonización de controladores P, PI y PID, se encuentran los métodos propuesto por Ziegler-Nichols (ZN). Estos métodos, son un conjunto de reglas que permiten el ajuste empírico en basan a mediciones realizadas sobre la planta. Existen dos tipos: el método oscilatorio y el método de la curva de reacción.

Para los dos métodos, los autores proponen valores que llevan al sistema a lograr una respuesta ante entrada escalón con características de rapidez y estabilidad aceptable. Pero por lo general, cuando se aplica estos métodos, se deber realizar un proceso posterior de refinamiento (o ajuste fino) de las ganancias del controlador para conseguir las características especificadas en la salida del sistema controlado.

Por lo cual, para el presente trabajo, se empleará el método de la curva de reacción, con el fin de encontrar los valores aproximados para las ganancias del controlador de la planta de flujo de aire caliente descrita anteriormente. A la vez, esto nos permitirá fijar los límites del espacio de búsqueda de los algoritmos GA, PSO y ABC, que se detallan más adelante.

3.2.2.1. Método de la Curva de Reacción de Ziegler-Nichols

Para hallar los valores de Kp, Ki y Kd, debemos obtener las características de la respuesta del sistema ante entrada escalón (entre el 10 % y el 20 % del valor nominal de entrada) en lazo abierto (fig.26), específicamente los valores de: tiempo muerto y tiempo de subida.

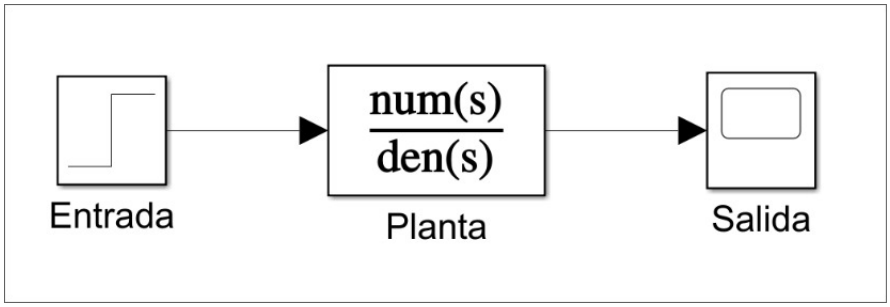


Figura 26. Estructura del sistema en lazo abierto

.

En la figura 27, se presenta la respuesta del sistema ante una entrada escalón en lazo abierto junto con la gráfica de la aproximación lineal de las misma. Esta aproximación nos permitirá determinar los valores requeridos para la sintonización del controlador.

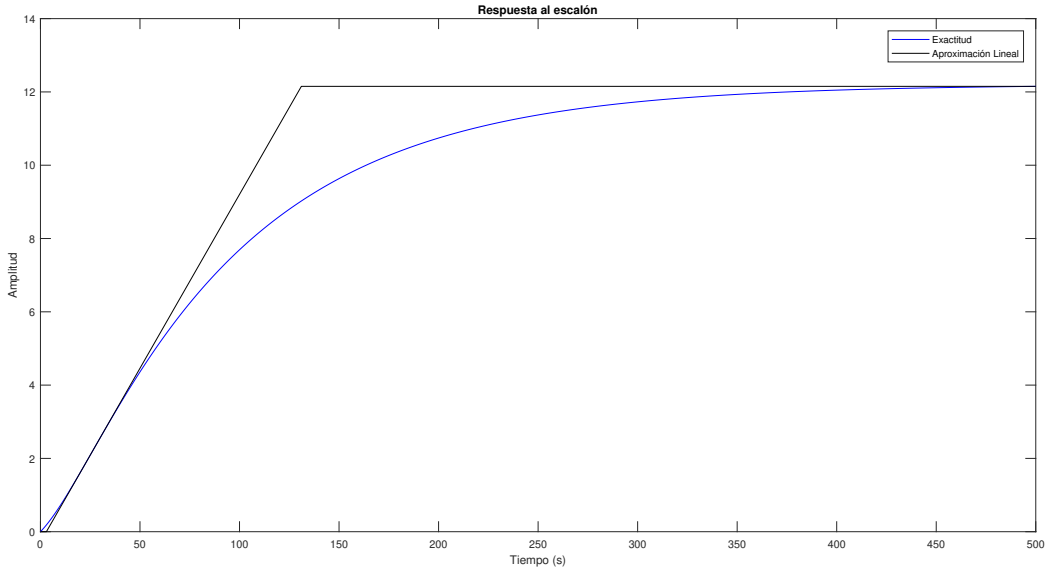


Figura 27. Respuesta del sistema ante entrada escalón en lazo abierto.
Señal de salida como respuesta del modelo descrito mediante la ecuación 3.2 ante una entrada escalón, y una aproximación lineal de la misma.

Luego, se calcula los parámetros de la señal de respuesta mediante las ecuaciones 3.3 $\sim$ 3.5, e inmediatamente se aplica lar reglas propuestas por Ziegler-Nichols mostradas en la tabla 4:

$$k_0 = \frac{y_\infty - y_0}{u_\infty - u_0} \tag{3.3}$$

$$\tau_0 = t_1 - t_0 \tag{3.4}$$

$$\gamma_0 = t_2 - t_0 \tag{3.5}$$

Tabla 4
Valores de Ziegler-Nichols.

	Kp	**Tr**	**Td**
P	$\frac{\gamma_0}{K_0\tau_0}$		
PI	$\frac{0,9\gamma_0}{K_0\tau_0}$	$3\tau_0$	
PI	$\frac{1,2\gamma_0}{K_0\tau_0}$	$2\tau_0$	$0,5\tau_0$

Con los cálculos respectivos, finalmente se obtiene los valores para las ganancias del controlador de sistema de temperatura (Tabla 5):

Tabla 5
Ganancias del controlador PID calculadas por Ziegler-Nichols.

Parámetros	**Ganancias**
Kp	15.879
Ki	11.8191
Kd	5.7890

En resumen, el controlador PID generado mediante el método de la curva de reacción se representa mediante la ecuación 3.6, y los valores característicos de la señal de salida se representan en la figura 28 (detallados en la tabla 6).

$$PID_{ZN} = \frac{5,79s^2 + 15,88s + 11,82}{s} \tag{3.6}$$

Tabla 6
Característicos de la señal de salida del sistema controlado por Ziegler-Nichols.

Parámetros	**Especificaciones**
Tiempo de levantamiento	0.705 segundos
Tiempo de estabilización	21.20 segundos
Sobreimpulso	30,6 %
Valor absoluto máximo	0,31

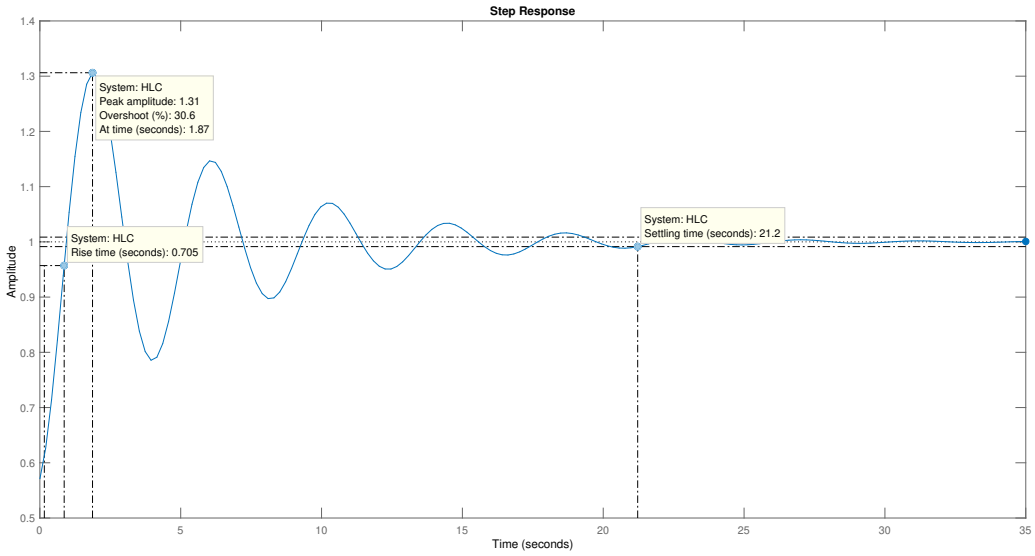


Figura 28. Respuesta del sistema ante una entrada escalón con la implementación de un controlador PID sintonizado a través del método de la curva de reacción de Ziegler-Nichols.

Con estos resultados, se justifica el proceso de refinamiento que se debe realizar tras la sintonización con el método de Ziegler-Nichols, puesto que no se cumple los objetivos de control establecidos para el sistema de flujo de temperatura, pero si permite conocer a qué valores próximos se encuentran las ganancias óptimas (rango de valores que contienen las soluciones óptimas).

3.2.3. Sintonización mediante Algoritmo Genético

Para facilitar al lector la comprensión de los métodos de sintonización mediante algoritmos bio-inspirados, se describe todo el proceso de construcción de los códigos de dichos métodos, dividido en cinco etapas, las cuales son:

1. Definición del problema
2. Determinación de parámetros iniciales
3. Inicialización del algoritmo

4. Proceso iterativo

5. Resultados

En cada una de estas etapas, se detallará los diferentes procesos, dependiendo del algoritmo bio-inspirado empleado, que conlleva a la búsqueda de las ganancias del controlador.

Por consiguiente, para el desarrollo del método de sintonización mediante GA, previo a la descripción de las etapas determinadas, se incluye el pseudocódigo del algoritmo GA empleado (Algoritmo 1, en referencia al diagrama de flujo de procesos de la figura 7, incluida en la sección 2.2.1.2).

Algoritmo 1: ALGORITMO GENÉTICO - GA

Datos: Parámetros de ajuste del algoritmo GA.

```
1  Inicio
2      Definir: Función Aptitud.
3      Codificar soluciones en cromosomas.
4      Generar población inicial.
5      Evaluar a cada individuo de la población actual mediante Función Aptitud.
6      Mientras la condición de parada no se cumpla hacer
7          Seleccionar mejores candidatos (padres).
8          Generar nuevos individuos mediante la probabilidad de recombinación de padres.
9          Generar nuevos individuos mediante la probabilidad de mutación de descendientes.
10         Evaluar aptitud de descendientes. Actualizar población.
11     fin
12     Devolver mejor individuo descendiente.
13     Decodificar solución.
14 Fin
```

3.2.3.1. Definición del problema

El problema planteado en el presente trabajo, es la sintonización de un control tipo PID. Ahora bien, el diseño del controlador sintonizado mediante GA, se realizó en función del conocimiento del proceso y de las restricciones que se le imponen al mismo (objetivos de control). Por ello, para esta etapa es necesario el ingreso de tres parámetros en forma de vector, para facilitar el manejo de datos durante la programación del algoritmo:

a) Modelo de la planta: Función de transferencia 3.2.

b) Valores deseados para los parámetros de diseño del controlador.

$$parm = [Mp \quad Ts] \tag{3.7}$$

c) Ponderaciones (pesos) para cada uno de los parámetros de diseño.

$$w = [w_1 \quad w_2] \tag{3.8}$$

Con estos datos ingresados se prosigue a la definición en código del problema, que equivale a la definición de la función de evaluación o función objetiva.

Función de evaluación

Para conocer y cuantificar el comportamiento del controlador con las ganancias encontradas por el algoritmo, implementado en la planta, se realiza la simulación del funcionamiento del sistema ante una entrada escalón y así obtener la señal de salida de dicho sistema.

Se extraer los datos necesarios de la señal de salida, y se procede a evaluar su desempeño. La configuración empleada para dicha simulación se presenta en la figura 29.

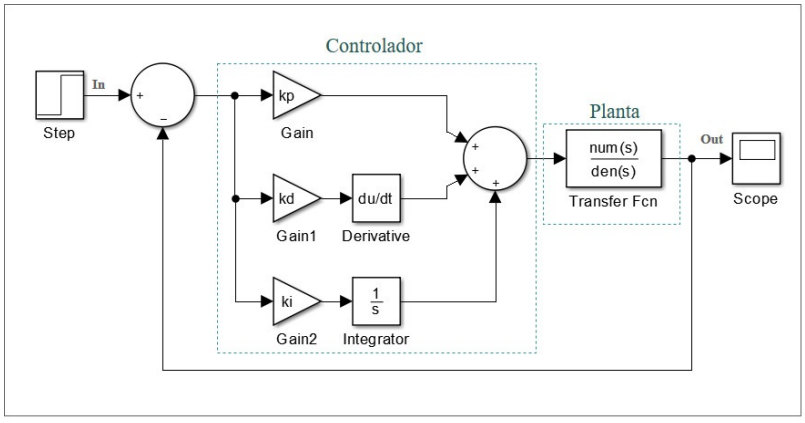


Figura 29. Esquema general del sistema controlado.

Con la señal de salida (ante una entrada escalón como ya mencionó) y los valores de los parámetros de diseño planteados inicialmente, se efectúa la valoración del desempeño de acuerdo a los siguientes criterios:

A. Costo en base al porcentaje de sobreimpulso alcanzado.

B. Costo en base al tiempo de establecimiento.

C. Costo en base al cumplimiento global.

A. *Costo en base al porcentaje de sobreimpulso alcanzado.*
El costo de este primer criterio de evaluación se lo calcula en base a al sobreimpulso medido en la simulación y el valor deseado para dicho parámetro, mediante la siguiente ecuación (3.9):

$$Cost_{MP} = \frac{Mp_{medido} \quad (\%)}{Mp_{deseado} \quad (\%)} \tag{3.9}$$

Con esta ecuación (ecu.3.9), se garantiza una proporcionalidad entre el valor del sobreimpulso medido y el costo en base a este parámetro; ya que, mientras el sobreimpulso de la señal disminuya, el costo también disminuirá.

Para hallar el valor actual del sobreimpulso de la señal (valor medido), existen varias formas; una de ellas es extrayendo el valor máximo de la señal y aplicando la siguiente ecuación (3.18):

$$Mp_{medido} \quad (\%) = \left(\frac{Mp_{max} \quad - Ref}{Ref} \right) \times 100\,\% \tag{3.10}$$

Sin embargo, existe una manera más fácil de obtener dichos parámetros. Esta manera es empleando una función de análisis de señales propia de Matlab; la cual, calcula las características de la respuesta ante una entrada escalón de un modelo dinámico. Entre las características que entrega dicha función está (Fig.30):

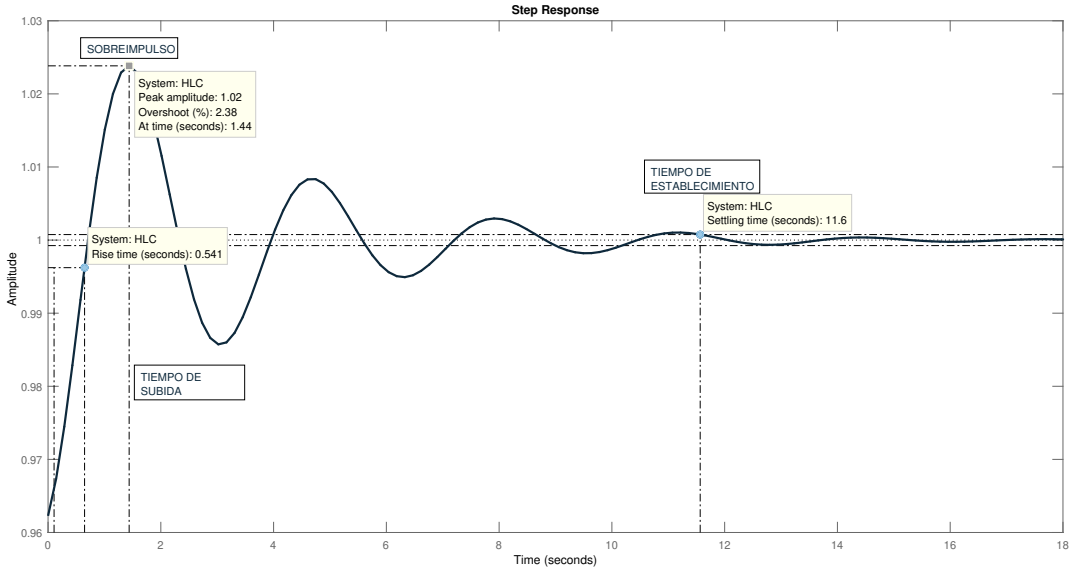


Figura 30. Análisis de la respuesta ante entrada escalón mediante Matlab.

a) Tiempo de subida.

b) Tiempo de asentamiento al 2 % de error del valor final.

c) Overshoot.

d) Undershoot.

e) Valor absoluto máximo de la señal.

f) Tiempo en que se produce el valor máximo.

Por lo tanto, en el presente trabajo se ha considerado el uso de las herramientas de Matlab para simplificar las líneas de código.

B. *Costo en base al tiempo de establecimiento.*

Al igual que el anterior caso, el costo en base al tiempo de establecimiento se lo calcula en base al tiempo que se necesitó para alcanzar el régimen permanente con el 2 % de error del valor final (según lo recomendado en Ogata (2003)) y el valor deseado para el mismo, mediante la siguiente relación (ecuación 3.11):

$$Cost_{Ts} = \frac{Ts_{medido}}{Ts_{deseado}} \tag{3.11}$$

Con esta relación (ecu.3.11), también se puede aseverar que a menor tiempo de establecimiento, menos costo en base a este parámetro.

Asimismo, en este caso también se aplicará una función propia de Matlab para el cálculo del tiempo de establecimiento de la señal analizada (misma función que en el anterior caso).

C. *Costo en base al cumplimiento global*
Por último, para encontrar una solución que cumpla con los dos anteriores criterios, se efectúa el cálculo de un costo global por medio de una sumatoria pondera de los dos anteriores costos (ecu.3.12):

$$Cost_{Total} = (w_1 \times Cost_{MP}) + (w_2 \times Cost_{Ts}) \tag{3.12}$$

Donde, w_1 y w_2 son los pesos de importancia para cada uno de los parámetros de diseño. Además, w_2 es complemento de w_1.

En base a este resultado, las operaciones que se desarrollaran dentro del algoritmo, realizaran la toma de decisiones para obtener una solución con el mínimo costo global.

La ***función de evaluación*** es uno de los subprocesos más importantes dentro de los algoritmos de optimización; ya que, engloba todo el objetivo de búsqueda. Esencialmente, la construcción de la función de evaluación equivale a acoplamiento del algoritmo GA como herramienta para la sintonización de controlador PID (esto aplica no solo para el algoritmo GA, sino que también para el algoritmo de optimización PSO y ABC).

Como ya se explicó que parámetros intervienen en la función de evaluación y como calcularlos, ahora, se presenta, en la figura 31, el proceso sistemático en el cual consiste la función de evaluación.

En el resto del capítulo, se recalcará las veces que interviene la función de evaluación para lograr la convergencia del algoritmo en una solución óptima global.

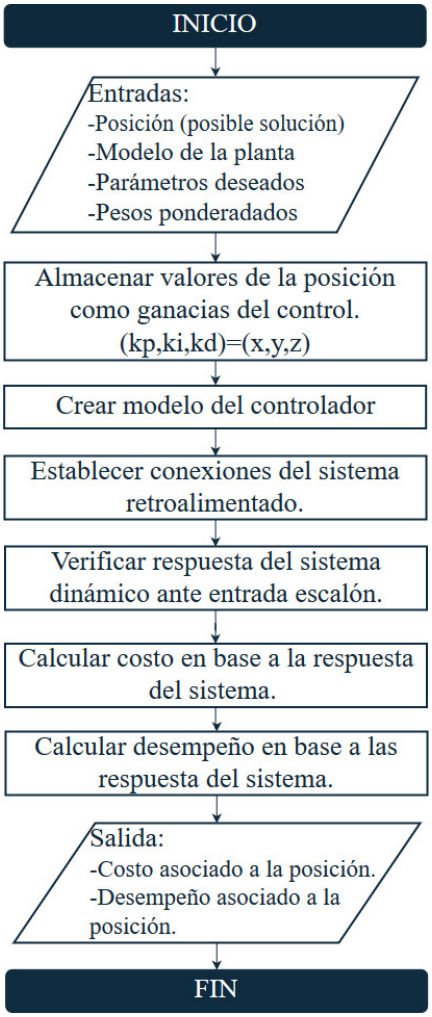


Figura 31. Diagrama de flujo de la *Función de Evaluación*.

3.2.3.2. Determinación de valores de ajuste del algoritmo GA

Existen ciertos parámetros que deben ser definidos inicialmente para el ajuste del algoritmo. Como se mencionó en capítulo II, la convergencia hacia una solución óptima, dependerá de la definición correcta de los parámetros iniciales por parte del programador. Estos parámetros no sólo delimitan la ejecución del algoritmo, sino que también, lo estructuran para trabajar el problema definido en la sección anterior. Para el caso del algoritmo GA, se han definido los siguientes parámetros.

a) Tamaño de la población.

b) Número de generaciones.

c) Longitud del cromosoma.

d) Probabilidad de cruce y mutación.

e) Límite del espacio de búsqueda.

Según lo recomendado en varias fuentes bibliográficas citados en sección 2.2.1, el tamaño de la población debe estar dentro del rango de $[20-50]$ individuos. Por ende, para este problema se ha elegido una población de 25 individuos (suficiente para lograr una exploración completa dentro del espacio de búsqueda).

El número de generaciones será tomado como criterio de parada; por lo cual, en la programación equivale al número de iteraciones del proceso de búsqueda de la solución óptima global. En este caso, se realizará un proceso repetitivo de 50 generaciones (valor que garantiza evitar convergencia prematura).

Mientras que, la longitud del cromosoma L, que representa la estructura de la solución escrita en código binario, se lo definido de la manera que se representa en la figura 32:

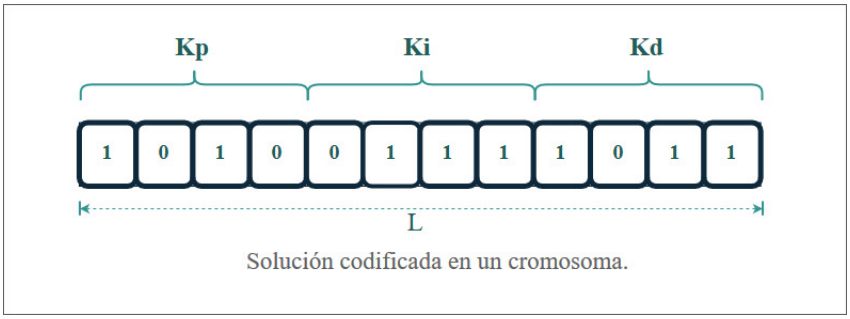


Figura 32. Estructura del cromosoma para la búsqueda de ganancias del controlador PID.
Ejemplo de la estructura de un cromosoma de 4 bits para cada parámetro de búsqueda.

En cuanto a los valores de las probabilidades de cruce y mutación se han asignado de la siguiente forma:

$$P_{cruce} = 85\,\% \qquad P_{mutación} = 15\,\%$$

Estas probabilidades han sido fijadas según los rangos recomendados por X.-S. Yang (2014), donde se sugiere que la probabilidad de cruce sea muy alta en comparación de la probabilidad de mutación, para asegurar la eficiencia de la evolución de las generaciones, y a la vez no se salten soluciones óptimas por la introducción de genes mutados. El rango de la probabilidad de cruce se encuentra generalmente entre $[0{,}7 \sim 1]$; mientras que el rango de la probabilidad de mutación es de $[0{,}001 \sim 0{,}005]$. No obstante, para la búsqueda de las ganancias del controlador se ha fijado la probabilidad de mutación como complemento de la probabilidad de cruce, como se recomienda en otras fuentes bibliográficas (Melanie, 1998).

Además, los límites del espacio de búsqueda se han establecidos en base a las ganancias halladas a través de la sintonización clásica (método de Ziegler-Nichols), y son ingresados mediante la siguiente matriz.

$$Lim = \begin{bmatrix} Kp_{min} & Kp_{max} \\ Ki_{min} & Ki_{max} \\ Kd_{min} & Kd_{max} \end{bmatrix}$$

3.2.3.3. Inicialización del algoritmo GA

En la etapa de inicialización del algoritmo GA, se genera la población inicial de forma aleatoria. Es decir, se crea una matriz del tamaño de la longitud del cromosoma l, por el número de pobladores definidos n; adicionalmente, se incluyen vectores para la información auxiliar que facilitará la toma de decisiones y la conversión de las soluciones de código binario a números naturales.

$$Población = \begin{bmatrix} Kp_{1bin} & Ki_{1bin} & Kd_{1bin} & Aux_1 \\ Kp_{2bin} & Ki_{2bin} & Kd_{2bin} & Aux_2 \\ Kp_{3bin} & Ki_{3bin} & Kd_{3bin} & Aux_3 \\ \vdots & \vdots & \vdots & \vdots \\ Kp_{nbin} & Ki_{nbin} & Kd_{nbin} & Aux_n \end{bmatrix}$$

Cada cromosoma, como se representa en la figura 32, cuenta con cierto número de genes (bits del cromosoma escrito en código binario) que conforman la solución en binario de las ganancias del controlador (Kp, Ki, Kd). Este grupo de genes (para la búsqueda de cada solución), representa una resolución de $m - bits$ que tendrá la solución.

$$Kp_{nbin} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 & ... & b_m \end{bmatrix}$$
$$Ki_{nbin} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 & ... & b_m \end{bmatrix}$$
$$Kd_{nbin} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 & ... & b_m \end{bmatrix}$$

Mediante esta configuración de la población inicial, la solución final está dada por el vector:

$$Solución = \begin{bmatrix} Kp_{bin} & Ki_{bin} & Kd_{bin} & Aux \end{bmatrix}$$

3.2.3.4. Proceso iterativo del algoritmo GA

Una vez generada la población inicial, se da inicio al proceso de exploración de la mejor solución global; que, mediante cada repetición de las operaciones de búsqueda, se va eliminando las soluciones menos prometedoras, y a la vez potenciando las soluciones más viables. Las operaciones de búsqueda son:

i. Evaluación

ii. Selección

iii. Recombinación o cruce

iv. Mutación

v. Actualización de población

Evaluación de aptitud

La evaluación de aptitud se aplica a cada individuo de la actual población. En esta operación interviene la *función de evaluación* (definida en la primera etapa, pág. 65), por medio de la cual se valora el desempeño y costo de cada individuo.

Selección

La operación de selección de padres no es más que la elección de dos de los individuos más aptos para que a partir de ellos se cree una nueva población, mejor a la actual.

La creación de una generación de nuevos individuos se realiza en base a un proceso probabilístico, donde, se toma como preferencia a los individuos con mayor aptitud. En otras palabras, se selecciona a los individuos con máximo desempeño alcanzado durante la evaluación.

Este proceso, inicia a partir del cálculo de un criterio probabilístico de selección para cada individuo (ecu.3.13). La probabilidad de selección del i-ésimo individuo dentro de un conjunto de n-pobladores se da mediante la siguiente ecuación:

$$p_{(selección)i} = \frac{P_i}{\sum_{i=1}^{n} P_i} \qquad (i = 1,2,3,..,n) \tag{3.13}$$

Donde, P_i representa el desempeño (performance) alcanzado por la solución i-ésima.

Con el anterior cálculo, y a través del método de la ruleta se hace la selección de los individuos padres. El método de la ruleta consiste en imitar el giro de la ruleta sobre una superficie formada por las posibles soluciones; los individuos seleccionados serán donde la ruleta haya parado (Fig.33).

Viendo desde otra perspectiva, para selección mediante el método de la ruleta, se crea un segmento de línea recta de longitud $R = \sum_{i=1}^{n} P_i$ dividida en tramos que representan el desempeño de cada individuo $P_i \quad con \quad (i = 1,2,3,..,n)$ (como se muestra en la figura 34); luego se genera un número aleatorio dentro del rango $[0 \sim \sum_{i=1}^{n} P_i]$, este valor se localizará dentro de algún tramo el cual será pertenecerá al individuo seleccionado (Fig.34).

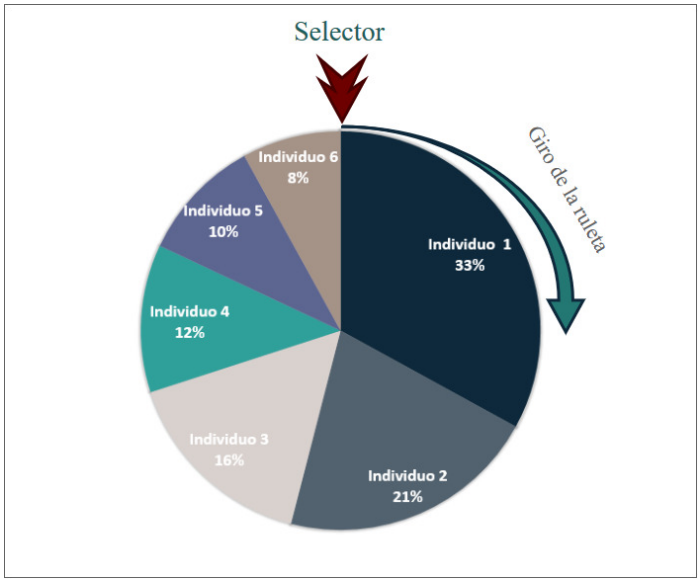


Figura 33. Método de la ruleta.

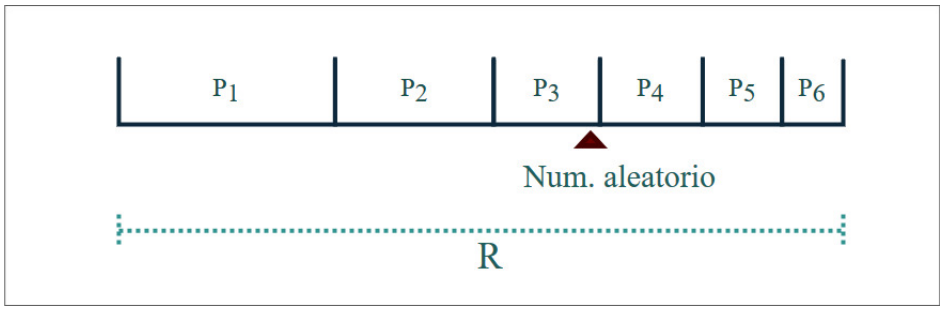


Figura 34. Representación del método de la ruleta de forma lineal.

El proceso de selección a través del método de la ruleta, garantiza que la probabilidad de selección sea proporcional a la aptitud del individuo, siguiendo así el principio del algoritmo GA: a mayor adaptación, mayor probabilidad de reproducción.

Recombinación

La operación de recombinación o cruce es el proceso más característico del algoritmo GA; ya que, mediante las repeticiones de esta operación se amplía el espacio de soluciones. De hecho, el fin de la recombinación es principalmente generar una mezcla de las soluciones más potenciales, y la convergencia en un área específica del espacio de búsqueda.

La recombinación de dos individuos padres genera como resultado dos individuos hijos, los cuales, conservan las características esenciales de los padres. Este es un proceso bastante sencillo, sin embargo, existen muchas variantes para ejecutarlo; pero para el presente problema de búsqueda, se ha decidido realizar el cruce en base a un solo punto para evitar demasiadas rupturas sobre las mejores soluciones.

El punto de cruce se genera de forma aleatoria (entre el número de genes escritos en forma binaria), para cada operación de recombinación que deba realizarse.

Esta operación se realiza en base a los individuos seleccionados como padres en la operación anterior y dependiendo de la probabilidad de cruce definida en la segunda etapa del desarrollo del algoritmo GA.

Mutación

A diferencia de la operación de recombinación, la operación de mutación aumenta la diversidad de individuos y evita los óptimos locales como solución final.

Esta operación se ejecuta sobre los individuos descendientes generados en el proceso de recombinación. Su mecanismo es introducir una variación en un punto aleatorio del individuo (en un gen determinado al azar). Gráficamente, el proceso está representado dentro de la sección 2.2.1.2. (Fig.11).

Cabe mencionar, que este proceso de mutación, también depende de la probabilidad de mutación definida en la segunda etapa del desarrollo del algoritmo GA.

Actualización de población

No todos los individuos descendientes serán parte de la nueva población o nueva generación. Sino que, a través de la valoración de la aptitud, tanto de los individuos padres como de los hijos, será elegidos los que mayor aptitud presenten para conformar la nueva generación.

Este proceso es bastante parecido a la operación de selección de padres, sin embargo, en la actualización de la población se valora la aptitud tanto de los antiguos individuos como de los nuevos.

3.2.3.5. Resultados

Una vez terminada la etapa iterativa por el número máximo de repeticiones. Se analiza los valores de costo (mínimo) y desempeño (máximo) de los individuos de la última generación, y se decodifica las soluciones encontradas. Estas soluciones se presentan en forma de vector de acuerdo a los parámetros evaluados, como se muestra a continuación:

$$Sol_{MP} = (Kp, \quad Ki, \quad Kd) \tag{3.14}$$

$$Sol_{Ts} = (Kp, \quad Ki, \quad Kd) \tag{3.15}$$

$$Sol_{Global} = (Kp, \quad Ki, \quad Kd) \tag{3.16}$$

Para el análisis de desempeño, se presenta la evolución del costo y desempeño alcanzado durante cada generación por el mejor individuo de la población actual (fig.35 y fig.37).

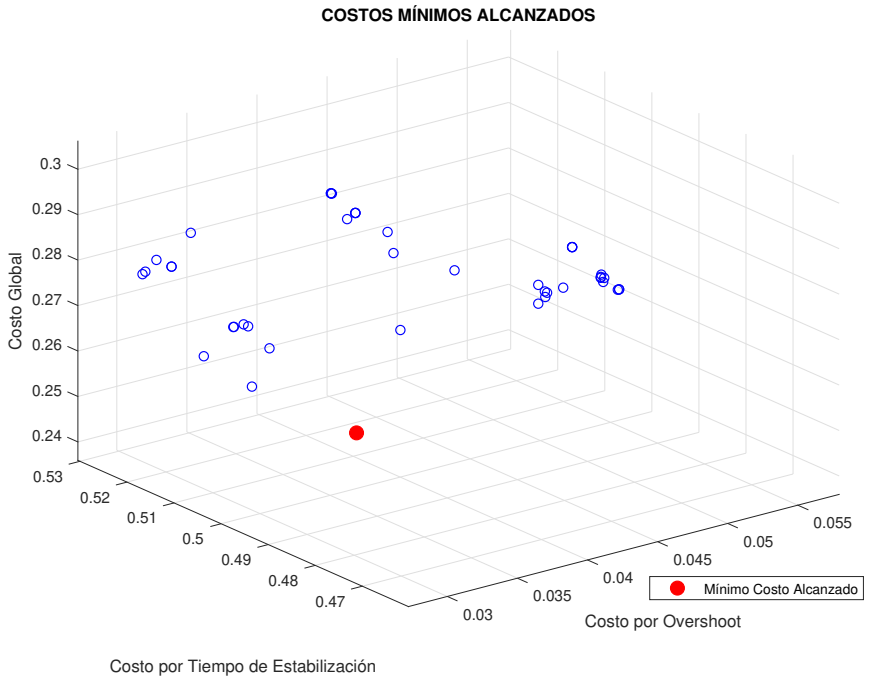


Figura 35. Ejemplo: Costos globales resultantes del algoritmo ABC.

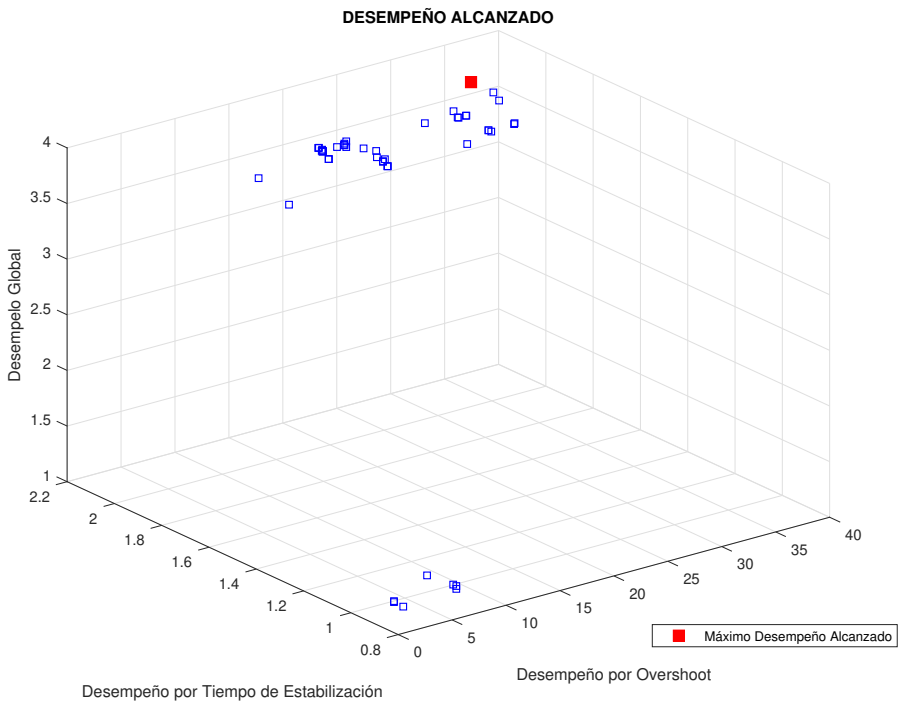


Figura 36. Ejemplo: Desempeños globales resultantes del algoritmo GA.
Ejemplificación de la presentación de resultados: Desempeño global alcanzado por las posibles soluciones durante la ejecución del algoritmo GA.

Y como resultado final, se presenta la respuesta del sistema controlador con mejor desempeño (fig.38).

3.2.4. Sintonización mediante Algoritmo PSO

La sintonización de un controlador tipo PID, mediante el algoritmo de optimización PSO (descrito gráficamente en la fig.15), se origina a partir del requerimiento de encontrar los valores más óptimos para las ganancias de dicho controlador de una manera sencilla y efectiva. Por eso, haciendo uso del algoritmo PSO (herramienta potencial de búsqueda), conocida por emplear simples proceso inspirados en el comportamiento de las bandadas de aves y bancos de peces, se ha desarrollado la sintonización de un controlador para la planta de flujo de aire caliente descrita en la sección 3.1.

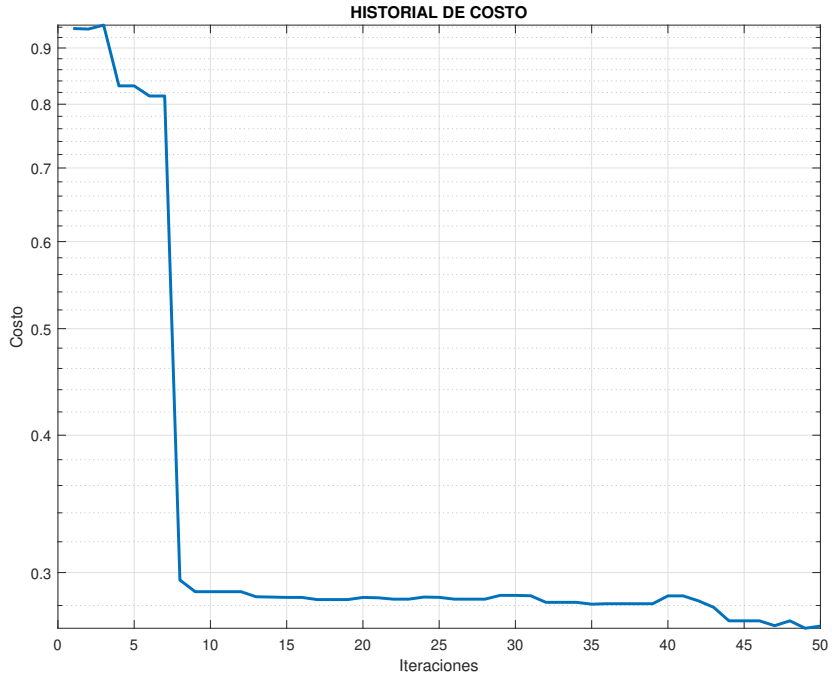


Figura 37. Ejemplo: Historial de costos resultantes del algoritmo GA.
Ejemplificación de la presentación de resultados: Historial de los costos generados durante cada iteración.

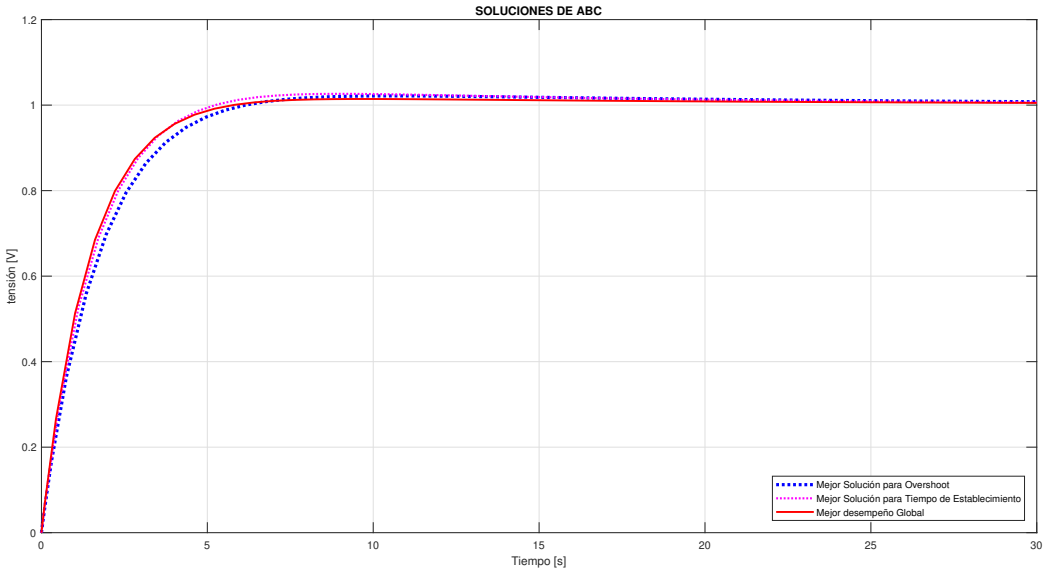


Figura 38. Ejemplo: Respuesta del controlador obtenido mediante GA.

Para la descripción del proceso de sintonización, se presenta el pseudocódigo de método PSO (Algoritmo 2), en base al cual, se ha elaborado el código de búsqueda de las ganancias del controlador.

Algoritmo 2: OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS - PSO

Datos: Parámetros de ajuste del algoritmo PSO.

```
Inicio
    Definir: Función de Evaluación.
    Inicializar posición y velocidad para n-partículas.
    Evaluar todo el conjunto de partículas generadas.
    Memorizar mejor pBest de cada partícula del enjambre.
    Encontrar y memorizar gBest del enjambre.
    Mientras la condición de parada no se cumpla hacer
        Para i = 0 hasta i = n en las N − dimensiones hacer
            Actualizar la velocidad de la partícula i mediante la ecuación 2.1.
            Actualiza la posición de la partícula i mediante la ecuación 2.2.
            Evaluar la nueva posición de la partícula i.
            Encontrar la mejor posición de la partícula i hasta el momento.
            Memorizar pBest de la partícula i.
        fin
        Encontrar y memorizar gBest del enjambre.
    fin
    Devolver la mejor posición alcanzada (gBest).
Fin
```

Cabe mencionar que, no se han utilizado modificaciones en la estructura del algoritmo PSO para la implementación de la sintonización por medio de este algoritmo. Por cual, se mantienen los diagramas de flujo-de proceso originales.

A continuación, se detalla las cinco etapas que comprende el método de sintonización a través del algoritmo PSO.

3.2.4.1. Definición del problema

Como se expone en el inicio del presente capítulo, el problema a tratar, será la búsqueda de las ganancias de un controlador de temperatura. Por lo cual, al igual que en la definición del problema para la ejecución del algoritmo GA, se deberá ingresa:

a) Modelo de la planta (Función de trasferencia 3.2).

b) Valores de los parámetros de diseño del controlador (Matriz 3.7).

c) Pesos de cada uno de los parámetros de diseño (Matriz 3.8).

Además, se hace uso de la función de evaluación generada para la sintonización mediante el algoritmo GA, descrita en la sección *Función de Evaluación* (pág.65); ya que, el objetivo final esperado es el mismo. No obstante, a diferencia del algoritmo GA (que evalúa los genes de los cromosomas generados), en el algoritmo PSO se evalúa la posición alcanzada por las partículas; por esa razón no es necesario ningún tipo de codificación de las soluciones.

Recapitulando, por medio de la función de evaluación se analizará el desempeño alcanzado de las posibles soluciones (posiciones de las partículas) dentro del sistema de control, mediante tres parámetros:

A. Costo en base al sobreimpulso alcanzado.

B. Costo en base al tiempo de establecimiento.

C. Costo en base al cumplimiento global.

3.2.4.2. Determinación de valores de ajuste del algoritmo PSO

La definición correcta de estos parámetros es fundamental para garantizar la convergencia del algoritmo PSO en una solución óptima global; puesto que, a través de los valores fijados en los parámetros que se detallan a continuación, el algoritmo quedará ajustado para encontrar una solución al problema planteado. Estos parámetros son:

a) Número máximo de iteraciones.

b) Tamaño del enjambre (número de partículas).

c) Número de dimensiones del espacio de vuelo.

d) Límite del espacio de vuelo.

e) Límite máximo y mínimo de velocidad de vuelo.

f) Límites de los coeficientes de aceleración.

g) Límites del factor inercial.

Cada uno de estos parámetros ha sido fijado de acuerdo a rangos sugeridos en el apartado de *Consideraciones* de la sección 2.3.1.2. Por lo consiguiente, los valores finales han sido determinados de la siguiente manera:

- Como condición de parada, se ha determinado según el número máximo de repeticiones del proceso de búsqueda, por lo que, el número máximo de iteraciones se ha fijado en un valor igual a 50.

- El tamaño del enjambre será igual a:

$$Swarm_{Size} = 20$$

  Número suficiente para que el algoritmo pueda convergir en una solución y no se atasque en óptimos locales.

- El número de dimensiones del espacio de vuelo está determinado por el número de parámetros a optimizar, como en este caso, los parámetros a optimizar son Kp, Ki, Kd, entonces:

$$N_{dimensiones} = 3$$

  Donde:

  · El plano x representa las soluciones para Kp.

  · El plano y representa las soluciones para Ki.

  · El plano z representa las soluciones para Kd.

- Los límites del espacio de vuelo (Lim_{max} y Lim_{min}), se determina de acuerdo a los rangos dentro de los cuales se espera obtener una solución, por lo que es necesario e indispensable tener un conocimiento previo del problema. Por esta razón, los límites del espacio de vuelo quedan fijados de acuerdo a las ganancias obtenidas mediante la sintonización clásica (similar a la definición del espacio de búsqueda del algoritmo GA).

- El límite de velocidad, al depender de del espacio de vuelo o espacio de búsqueda que se haya fijo, queda determinado de la siguiente manera:

$$Vel_{max} = 0{,}01\,(Lim_{max} - Lim_{min}) \tag{3.17}$$

$$Vel_{min} = -Vel_{max} \tag{3.18}$$

- Los coeficientes de aceleración, como establece la teoría, tanto C_1 como C_2 pueden llegar a obtener un valor máximo de 4, siempre y cuando se cumpla la condición de la 2.4. En esta etapa, solo se definirá los valores mínimos y máximos que pueda llegar a alcanzar cada uno de los coeficientes.

$$C = \begin{bmatrix} C1_{min} & C1_{max} \\ C2_{min} & C2_{max} \end{bmatrix} \tag{3.19}$$

- La variación de factor inercial W se ha determinado dentro del rango de $[0{,}9 \sim 0{,}5]$. De mayor a menor para lograr una exploración amplia al inicio y más fina al final. Para este factor, dentro de esta etapa, de igual manera que en el caso de los coeficientes de aceleración, solo se definirá su rango.

3.2.4.3. Inicialización del algoritmo PSO

En la etapa de inicialización del algoritmo PSO, se genera, de manera aleatoria y dentro del espacio de vuelo, las posiciones iniciales de cada partícula (puntos de partida). Asimismo, en esta etapa se debe definir la velocidad inicial de cada partícula; sin embargo, para simular el proceso natural de una bandada de aves, se inicia desde un estado de reposo ($Vo_i = 0$).

Tanto las posiciones de la población como sus velocidades quedan contenidas en matrices como se muestra a continuación:

$$Posiciones = \begin{matrix} Kp & Ki & Kd \end{matrix} \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \tag{3.20}$$

$$\overrightarrow{Vel} = \begin{bmatrix} (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ (x_3, y_3, z_3) \\ \vdots \\ (x_n, y_n, z_n) \end{bmatrix} \tag{3.21}$$

Donde n es el número de partículas que conforma el enjambre.

En la figura 39, se muestra las posiciones aleatorias generadas para un enjambre de 20 partículas.

Todas las partículas son evaluadas por la poción actual que hayan alcanzado, y lo valores de costo y desempeño generados por la función de evaluación son registrados según a la partícula que corresponda.

Por ser la primera posición que obtiene cada partícula del enjambre, esta es almacenada como el *pBest* de cada partícula, pues es la mejor posición que tienen hasta ahora.

$$pBest_i = \begin{bmatrix} (x, y, z) & Cost_{global} & Desemp._{global} \end{bmatrix} \tag{3.22}$$

Y para finalizar esta etapa, se explora dentro de todo el enjambre, la partícula con menor costo global. Una vez hallada a dicha partícula, se guarda, tanto su posición como su correspondiente costo global, en un espacio de memoria denominado *gBest*.

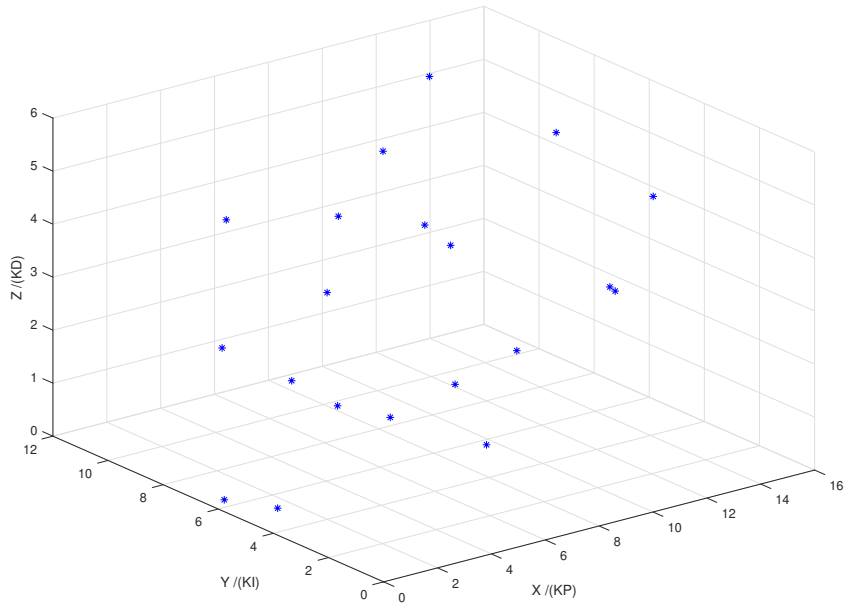


Figura 39. Inicialización del algoritmo PSO.
Posición inicial de un enjambre de 20 partículas dentro del espacio de vuelo.

$$gBest = \begin{bmatrix} (x,y,z) & Cost_{global} & Desemp._{global} \end{bmatrix} \tag{3.23}$$

3.2.4.4. Proceso iterativo del algoritmo PSO

En la etapa iterativa del algoritmo PSO se realiza la búsqueda de la mejor posición para el enjambre mediante el vuelo repetitivo de las partículas. Para cada nuevo movimiento que realice una partícula, se considerará el historial de la mejor posición particular lograda "*pBest*", y la mejor posición alcanzada por el enjambre "*gBest*". En otras palabras, cada partícula es atraída hacia su mejor posición registrada y hacia la mejor posición de todo el grupo de partículas (Fig.40). Por esta razón, la actualización de la velocidad y la posición de las partículas se realiza mediante las ecuaciones 2.1 y 2.2 respectivamente.

Sin embargo, para la actualización de la velocidad de la partícula interviene el factor de inercia y los coeficientes de aceleración. Por ende, en cada iteración que se realice, se calculará un nuevo valor tanto para el factor inercial como para los coeficientes de aceleración. En la sección *Mecanismo del Algoritmo PSO* (pág.33), se presenta la ecuación 2.3 con la cual se actualizará en valor del factor inercial; y en cuanto a la actualización de los coeficientes del aceleración, en este trabajo, se los actualizará mediante la ecuación 3.24.

$$C_{1,2} = \left(\frac{C_{min} - C_{max}}{Nmero \quad de \quad Iteraciones} \times Iter \right) + C_{max} \tag{3.24}$$

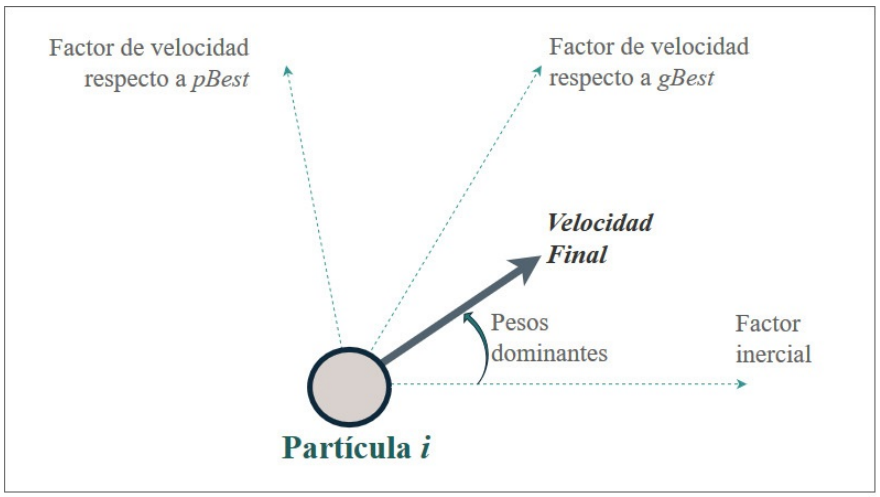


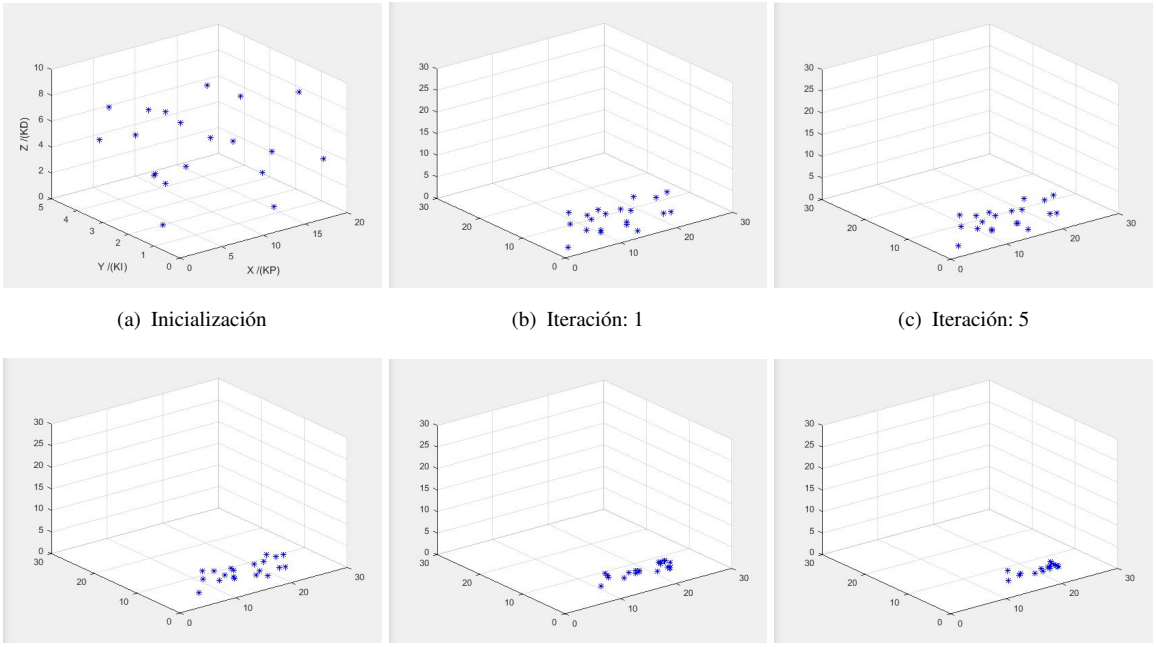
Figura 40. Factores que intervienen en el movimiento de una partícula.

Por cada posición alcanzada por la partícula, se evalúa el desempeño del sistema considerando la posible solución (posición) como parte del controlador. Inmediatamente, se realiza un cotejo en base al desempeño actual y el desempeño registrado como *pBest*; específicamente, en este trabajo se realizó la comparación del costo en base al cumplimiento global alcanzado por la partícula.

Y en base dicha comparación, se procede a la actualización del *pBest* y *gBest*. Donde, si el costo en base al cumplimiento global registrado anteriormente como *pBest* es mayor que el costo actual de la partícula, todos los datos de *pBest* son sustituido por la nueva partícula que alcanzó una mejor posición; caso contrario, el *pBest* queda invariante. Del mismo modo se procede

con la actualización de *gBest*, si el costo en base al cumplimiento global registrado hasta ahora como *gBest* es mayor que el costo actual de la partícula *i*, se remplazan los valores de *gBest*, caso contrario *gBest* se mantiene.

A continuación, se muestra la evolución de los movimientos de un enjambre de 20 partículas (Fig.41).



(a) Inicialización (b) Iteración: 1 (c) Iteración: 5

(d) Iteración: 15 (e) Iteración: 30 (f) Iteración: 50

Figura 41. Movimiento de un enjambre.

3.2.4.5. Resultados

Una vez concluidas el proceso de búsqueda mediante el cumplimiento de la condición de parada (número máximo de iteraciones), se presenta la posición (solución) que ha alcanzado el mínimo costo (o mejor desempeño para el sistema de control), es decir se presenta el *gBest* como la solución más óptima para el problema.

Adicional a los valores entrados por el vector *gBest*, se presenta gráficamente la localización en el espacio tridimensional de los valores de costo global generados durante la búsqueda de solución más optima (Fig.42) y la evolución de los mismos (Fig.43).

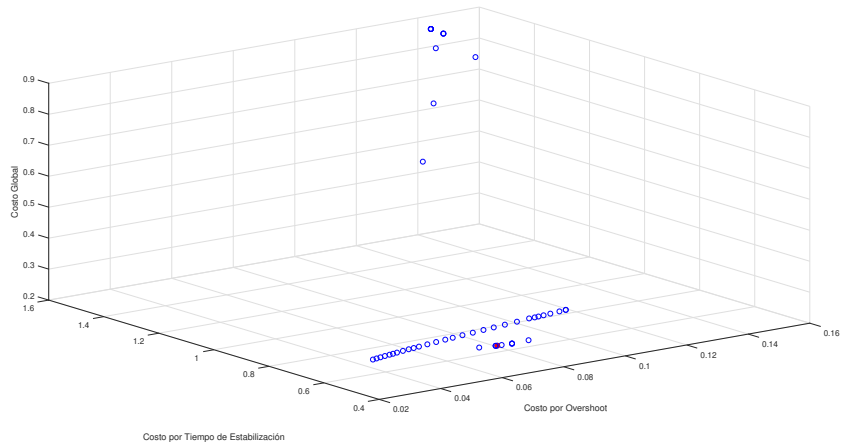


Figura 42. Ejemplo: Costos globales resultantes del algoritmo PSO.
Ejemplificación de la presentación de resultados: Costo global alcanzado por las partículas durante la ejecución del algoritmo PSO.

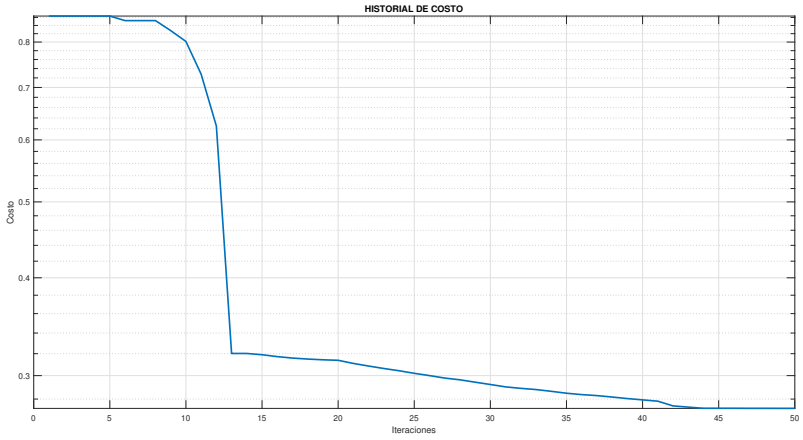


Figura 43. Ejemplo: Historial de los valores de costo registrados como *gBest*.
Ejemplificación de la presentación de resultados: Evolución del costo global alcanzado por las mejores soluciones durante cada iteración del algoritmo PSO.

En la figura 44, se muestra la localización tridimensional del desempeño global alcanzad por las posibles soluciones generadas durante la búsqueda de la solución más óptima.

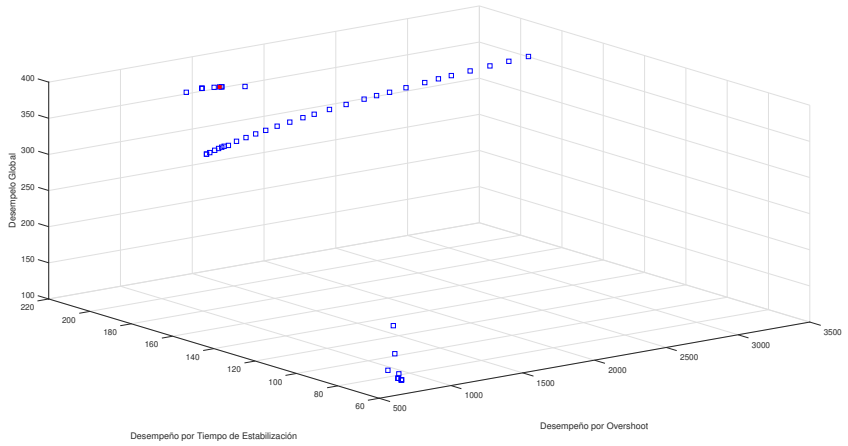


Figura 44. Ejemplo: Desempeños globales resultantes del algoritmo PSO.
Ejemplificación de la presentación de resultados: Desempeño global alcanzado por las partículas durante la ejecución del algoritmo PSO.

Y con el objetivo de comparar las soluciones generadas mediante la evaluación de los tres diferentes parámetros (costo en base a: sobreimpulso, tiempo de establecimiento y cumplimentero global, se presenta la siguiente gráfica (Fig.45).

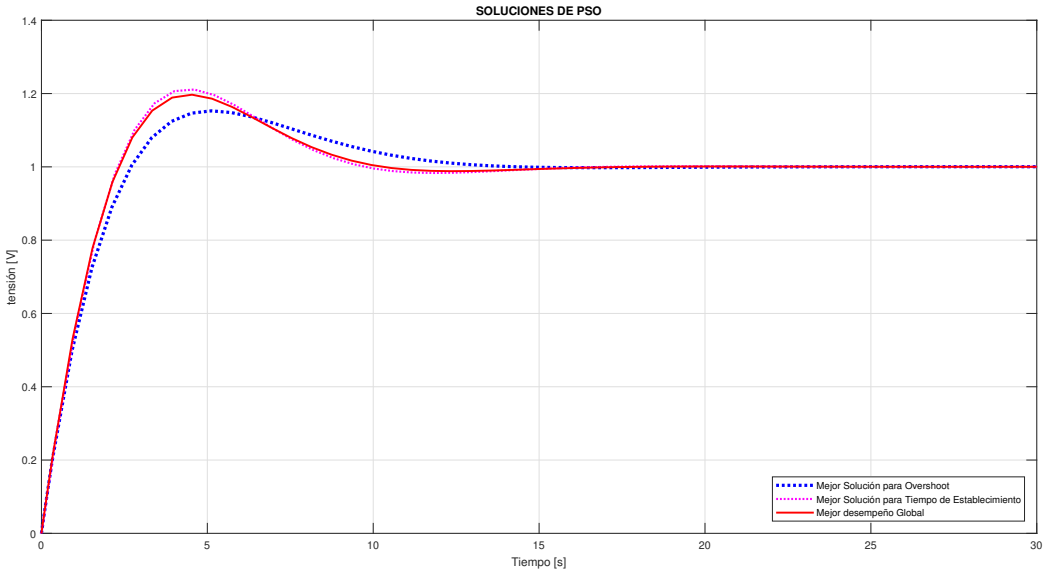


Figura 45. Ejemplo: Respuesta de los controladores obtenidos mediante PSO.

3.2.5. Sintonización mediante Algoritmo ABC

El diseño de un método de sintonización de un controlador tipo PID para una planta de flujo de aire caliente, mediante el uso de algoritmo de optimización ABC, parte del conjunto ordenado de operaciones propuesto originalmente por Karaboga (2005), explicado y expuestos gráficamente en la capítulo anterior (pág.48).

Para dar iniciar con la descripción del presente método, se presenta el pseudocódigo de las operaciones que se realizan durante la ejecución del algoritmo ABC (Algoritmo 3).

Como se indicó teóricamente, los procesos que comprenden el algoritmo ABC, simulan el comportamiento de una colonia de abejas compuesta por sólo tres tipos de abejas: empleadas, observadoras y exploradoras. El trabajo en conjunto de estos tres grupos de abejas, tienen como fin, encontrar la posición de mejor fuente de néctar; análogo a esta acción, en el método de sintonización, equivale a encontrar los valores de las ganancias del controlador que generen la señal de salida deseada de la planta (que cumpla con los parámetros de diseño establecidos). Para llevar a cabo dicha búsqueda, se detalla a continuación los parámetros y las operaciones que conforman cada una de las etapas que se han definido anteriormente para la construcción de los códigos.

3.2.5.1. Definición del problema

En esta etapa, al igual que en los anteriores casos, se ingresan tres datos fundamentales sobre el problema a optimizar:

a) Modelo de la planta (Función de trasferencia 3.2).

b) Valores de los parámetros de diseño del controlador (Matriz 3.7).

c) Pesos de cada uno de los parámetros de diseño (Matriz 3.8).

Asimismo, en este caso se reutilizará la función de evaluación creada en el desarrollo del algoritmo GA para la valorización del desempeño y costo de las posibles soluciones. En este caso, y similar al algoritmo PSO, el algoritmo ABC evalúa las posiciones en D-dimensiones de las

fuentes de néctar disponibles para la exploración de las abejas; por lo tanto, no se necesita realizar el proceso de codificación de las soluciones.

En resumen, para la evaluación de las fuentes de néctar se tomará en cuenta los tres parámetros con los que se han venido trabajando (ver *Función de Evaluación*, pág.65):

A. Costo en base al sobreimpulso alcanzado.

B. Costo en base al tiempo de establecimiento

C. Costo en base al cumplimiento global.

3.2.5.2. Determinación de valores de ajuste del algoritmo ABC

En esta etapa se definen los parámetros de ajuste del algoritmo ABC. Estos parámetros se los puede agrupar en dos grupos: constantes para el ajuste de acuerdo al problema y constantes para el ajuste de la ejecución del algoritmo.

Grupo 1 Constante para el ajuste de acuerdo al problema:

a) Número de dimensiones del espacio de búsqueda (número de variables a encontrar).

b) Límites del espacio de búsqueda.

Grupo 2 Constantes para el ajuste de la de la ejecución del algoritmo:

a) Número de iteraciones.

b) Número de fuentes de néctar.

c) Número de abejas que comprenderán la colmena.

d) Límite de abandono.

El ajuste del algoritmo para trabajar sobre el problema planteado inicia con la definición del número de dimensiones que tendrá en espacio de búsqueda. Al igual que en el caso del algoritmo PSO, el número de dimensiones representa el número de variables a optimizar; por lo cual, para la búsqueda de los valores de Kp, Ki, y Kd, se fija un espacio de 3 dimensiones. Complementario a esto, se fijan los límites de cada uno de los planos de las dimensiones, que equivale a los límites donde que se considera que los valores de las ganancias se hallarán.

$$Lim_{búsqueda} = \begin{bmatrix} Kp_{min} & Ki_{min} & Kd_{min} \\ Kp_{max} & Ki_{max} & Kd_{max} \end{bmatrix} \tag{3.25}$$

Por otra parte, los parámetros de ajuste para el control de la ejecución del algoritmo ABC, se los ha definido en base a las sugerencias bibliográficas presentadas en la sección 2.3.2.2, y en base a las consideraciones tomadas para la posterior comparación entre métodos. Por ende, dichos parámetros se han definido de la siguiente manera:

- El número de iteraciones, correspondiente a la condición de parada del algoritmo ABC, se lo ha definido en un valor igual a:

$$iteraciones = 50$$

- El número de fuentes de néctar o número de fuentes de alimento distribuidas por todo el espacio de búsqueda será igual a:

$$Num.Fuentes = 25$$

- El número de abejas que conformará la colmena, está relacionado con el número de fuentes, puesto que este el número de abejas debe ser mayor que el número de fuentes; por esta razón se ha definido que:

$$Num.Abejas = 2 \times Num.Fuentes$$

- El límite de abandono, que está directamente relacionado con la fase de ejecución de las abejas exploradoras se ha fijado de la siguiente manera.

$$Lim_{abandono} = \quad 1/2 \cdot Num.Fuentes$$

3.2.5.3. Inicialización del algoritmo ABC

Dentro de la inicialización, se genera las posiciones de las primeras fuentes de néctar de manera aleatoria (posibles soluciones). Dichas posiciones deberán estar dentro del entorno de la colmena, equivalente al espacio de búsqueda especificado especificado a traves de los rangos de los límites de los parámetros a optimizar.

Para la elaboración del código, se obtiene una matriz de posiciones generadas al azar. Las dimensiones de dicha matriz están especificadas por el número de fuentes iniciales N, y el número de dimensiones D.

$$Fuentes \quad \begin{matrix} F_1 \rightarrow \\ F_2 \rightarrow \\ = F_3 \rightarrow \\ \vdots \\ F_N \rightarrow \end{matrix} \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix} \tag{3.26}$$

Una vez generada la matriz de fuentes de alimento, se procede a la evaluación del desempeño de cada una de las posibles soluciones como parte del controlador (análogo a evaluar de la cantidad de néctar que posee cada fuente). Los valores de desempeño y costo son memorizados por cada fuente, y en base a estos valores se determina la mejor posible solución (la mejor fuente actual para la colmena) registrada en el vector solución (3.27).

$$Sol_{MejorFuente} \quad = \begin{bmatrix} (x,y,z) & Cost & Desemp. \end{bmatrix} \tag{3.27}$$

En la figura 46, se muestra la distribución aleatoria de varias fuentes de néctar dentro de los límites de búsqueda.

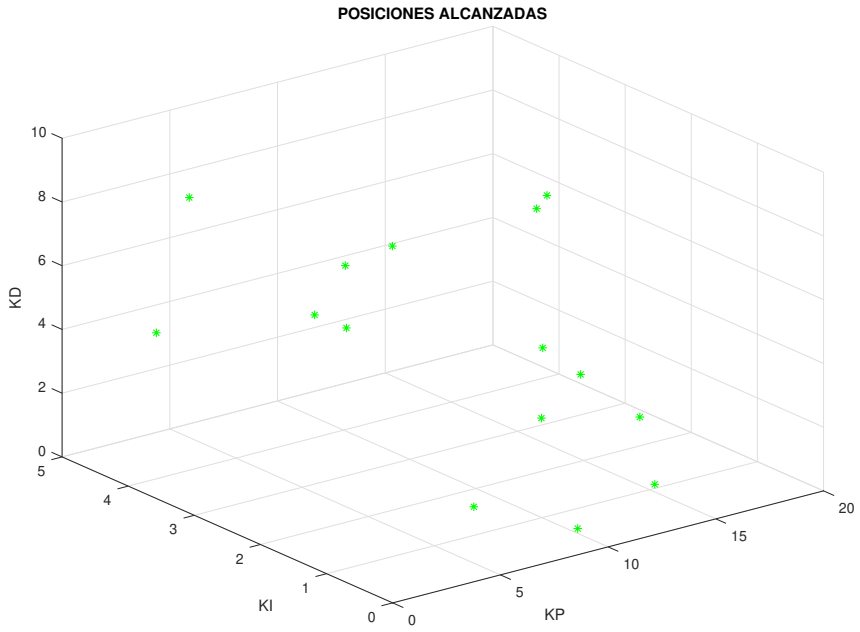


Figura 46. Posiciones de las fuentes de alimento iniciales en ABC.
Ubicación aleatoria de 15 fuentes de alimento generadas inicialmente para el desarrollo del algoritmo ABC.

3.2.5.4. Proceso iterativo del algoritmo ABC

Dentro de esta etapa encontramos tres fases correspondientes a los procesos que cumplen los tres grupos de abejas (empleadas, observadoras y exploradoras). El trabajo coordinado de estas tres fases da lugar al proceso de búsqueda de los valores más óptimos para las ganancias del controlador, es decir la mejor fuente de alimento dentro del espacio delimitado de exploración.

Como fue fijado en la segunda etapa de desarrollo del algoritmo ABC, mientras no se cumpla el número máximo de repeticiones de proceso de búsqueda se ejecutarán las siguientes fases:

Fase de abejas empleada

El grupo de abejas empleadas, que su número es igual al número de fuentes de néctar (puesto

que a cada abeja empleada sólo se le es asignado una fuente), cumplen la tarea de determinar una fuente vecina a la fuente asignada a cada una de las ellas. Y tras una evaluación a la nueva fuente de alimento por la abeja empleada, esta realiza una comparación entre la cantidad de néctar que poseen las dos fuentes de alimento (fuente asigna y fuente vecina), para luego seleccionar la fuente más prometedora.

En otros términos, las abejas empleadas generan nuevas posibles soluciones en relación a las posibles soluciones actuales, con el fin de explorar todo el espacio de búsqueda y encontrar la solución con mejor desempeño.

El proceso de determinación de una fuente vecina a la fuente asociada a una abeja empleada, se ejecuta mediante una pequeña modificación en la posición actual registrada dependiendo de la información local; en el código del algoritmo ABC, se lo realiza mediante la ecuación 2.9. Y el cálculo de la cantidad de néctar de la nueva fuente, se lo realiza a través de la función de evaluación.

Dado que la abeja empleada sólo debe mantener en memoria la posición de una sola fuente, se realiza la comparación de la cantidad de néctar; donde, si la fuente vecina contiene mayor cantidad de néctar que la fuente asociada a la abeja, se guardará en memoria la nueva posición, caso contrario se mantendrá registrada la posición que inicialmente la abeja empleada guardaba en memoria, además que se aumentará el contador de tendencia de abandono.

Una vez que todas las abejas empleadas hayan explorada las fuentes de néctar y se haya registrado toda la información respecto a las fuentes actuales, se da paso a la siguiente fase: grupo de abejas observadoras.

En la figura 47, se muestra un ejemplo de las fuentes generadas inicialmente y las fuentes registradas después de la ejecución de la fase de abejas empleadas. Aquí se puede observar claramente como se conservan algunas fuentes actuales (sobreposición de puntos azules sobre verdes) y como otras son remplazadas por las fuentes vecinas.

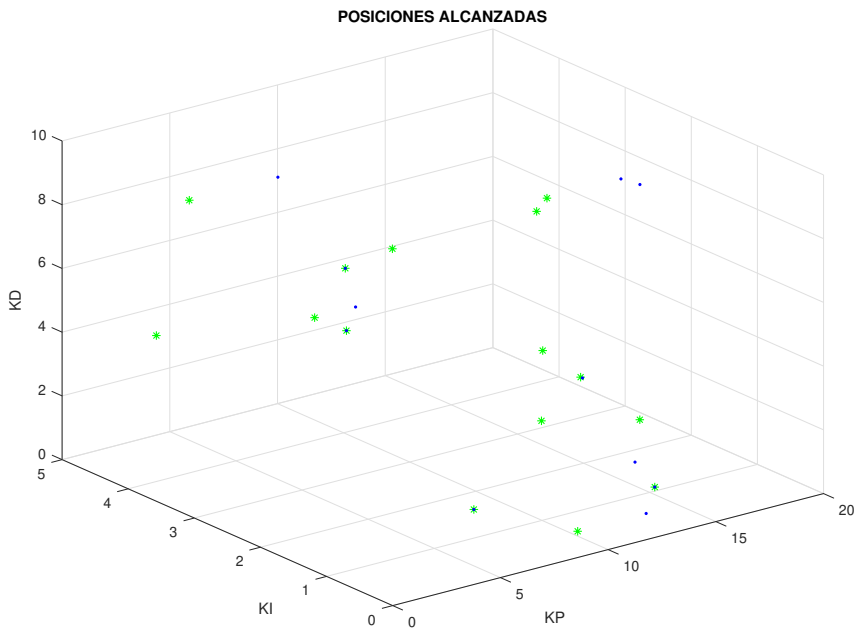


Figura 47. Fuentes de alimento registras por grupo de abejas empleadas.
Fuentes de alimento iniciales (verde) y fuentes de alimento registradas por el grupo de abejas empleadas (azul) en la primera iteración del proceso de búsqueda.

Fase de abejas observadoras

El proceso natural que sirve de inspiración para el algoritmo ABC, indica que, una vez concluido el trabajo de las abejas empleadas, estas vuelan hacia la colmena a compartir la información concerniente a la posición y la cantidad de néctar de sus fuentes con las abejas observadoras mediante danzas. En términos de programación, equivale al cálculo probabilístico para selección de las exploraciones que realizarán las abejas observadoras.

La probabilidad de selección p_i se lo realiza mediante el cálculo de la ecuación 2.10, en la cual intervienen directamente los valores desempeño que ha tenido la posible solución dentro del sistema de control. Este proceso de selección es igual al proceso de selección empleado en el algoritmo GA, pues en los dos casos se emplea en método de la ruleta.

Continuando con el proceso natural, una vez que las abejas observadoras han decido cual es la fuente de néctar más prometedora, vuelan hacia las coordenadas de la fuente elegida y exploran el área.

El proceso que se ejecuta dentro del algoritmo ABC, para logran una analogía a lo mencionado, es generar un número aleatorio en el rango $[0 \sim 1]$ y comparar con la probabilidad p_i asociado a la fuente i; una de las abejas observadoras realizará la exploración si solo se cumple la condición:

$$rand(0 \sim 1) < p_i \tag{3.28}$$

Luego de verificar que la condición se cumpla, la función que realizan las abejas observadoras es idéntica a la función que realizan las abejas empleadas.

Una abeja observadora introduce una modificación en la posición de la fuente de alimento actual, utilizando la ecuación 2.9. Es decir, se genera una fuente vecina a la fuente visitada. Posteriormente se evalúa la nueva fuente generada para la compasión con la fuente antigua. Si la nueva fuente supera en desempeño a la antigua fuente de alimento, se olvida la posición de esta y se memorizar posición de la nueva fuente de alimento; al mismo tiempo, se reinicia el contador de abandono. Caso contrario se mantiene en memoria la posición de la fuente antigua y se aumenta el contador de abandono.

El proceso de las abejas observadoras se repite hasta que todas las abejas que conforman este grupo se hayan distribuido en las fuentes de alimentos.

En la figura 49, se muestra el aumento del espacio de exploración a partir de las fuentes generadas inicialmente, después de la primera ejecución de la fase de las abejas empleadas y la fase de las abejas observadoras.

Fase de abejas exploradoras

La posición de la fuente vecina se halla Cuando se ha completado el proceso de búsqueda mediante la ejecución de las fases de las abejas empleadas y las abejas observadoras, se verifica si alguna de

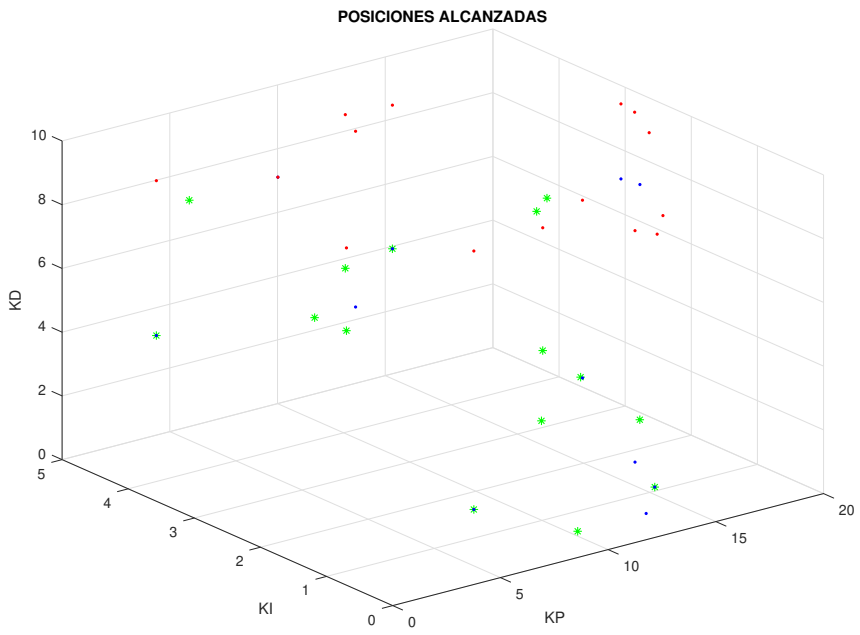


Figura 48. Fuentes de alimento registras por grupo de abejas observadoras.
Fuentes de alimento: iniciales (verdes), registradas por el grupo de abejas empleadas (azules) y registradas por el grupo de abejas observadoras (rojas), en la primera iteración del proceso de búsqueda.

las fuentes de alimentación ha sido agotada para poder abandonarla. De esta manera se completa un ciclo completo de búsqueda.

Para esta verificación se utiliza los contadores de tendencia de abandono que se han sido actualizados en cada proceso de exploración por parte de las abejas empleadas y observadoras. Cuando alguno de los contadores de abandono supera el límite de abandono definido en la segunda etapa, la fuente asociada a dicho contador es remplazada por una nueva fuente de néctar descubierta por las abejas exploradoras.

Matemáticamente, la posición de una nueva fuente de alimento es generada de forma aleatoria dentro de los límites del espacio de búsqueda por medio de la ecuación 2.8, si se cumple la condición:

$$Cont.Abandono_i > limit.Abandono \tag{3.29}$$

Y al igual que en los anteriores casos, la nueva fuente es evaluada para determinar su cantidad de néctar (determinación del desempeño de la posible solución).

Seguidamente, se registra la toda la información de la nueva fuente en el espacio de memoria de la fuente abandonada.

En la figura 49, se muestra la fuente con mayor tendencia de abandono; sin embargo, la fuente no es remplazada porque en un primer ciclo de búsqueda, ningún contador puede superar el límite de abandono.

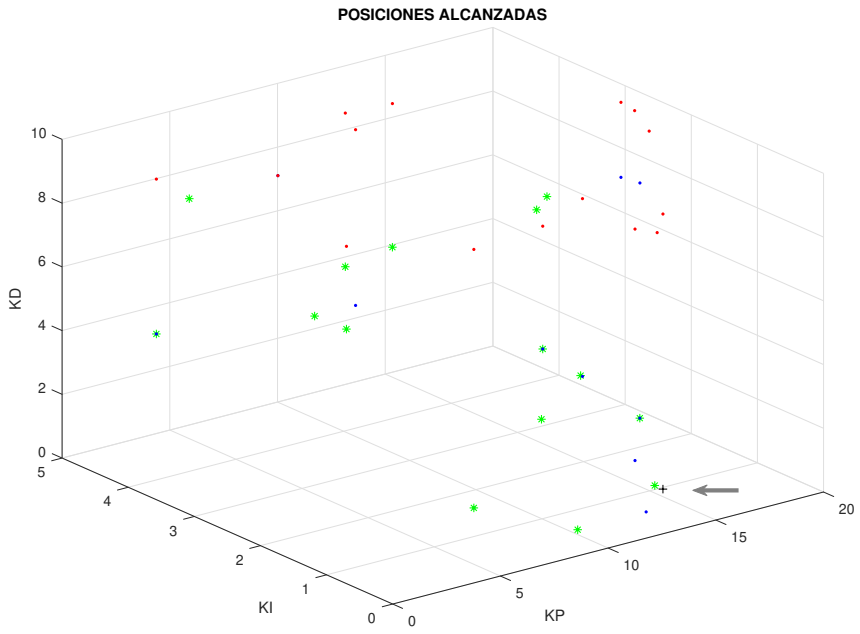


Figura 49. Fuente de alimento con mayor tendencia de abandono.
La fuente señalada (+) indica que es la fuente de alimento mayor explotada.

Cambios realizados

El autor del código original del algoritmo ABC sugiere que, en la fase de las abejas exploradoras, sólo se debe realizar el abandono de una fuente por cada ciclo de ejecución (como se muestra en el diagrama de flujo de la figura 21 en la pág.49). Sin embargo, en esta ocasión se ha realizado una ligera modificación en el código; en cada ciclo de ejecución se remplazará el cambio de todas

las fuentes de néctar que superen el límite de abandono. En la figura 50 se representa el nuevo diagrama de flujo de la fase de las abejas exploradoras que se ha implementado.

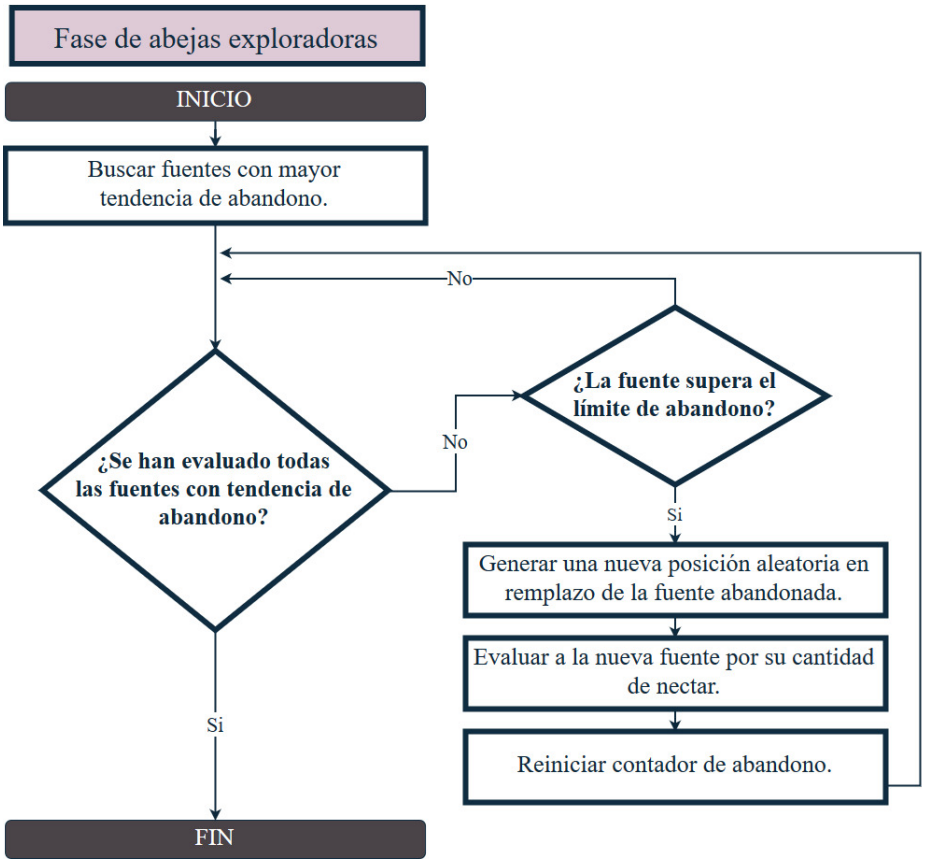


Figura 50. Diagrama de flujo del proceso de las *Abejas Exploradoras Modificado*.

La ejecución ordenada de las tres fases, consigue determinar las mejores soluciones para el problema de búsqueda; con las repeticiones del ciclo se logra expandir el espacio de búsqueda y descartar las soluciones menos prometedoras.

Cabe recalcar que el algoritmo ABC es muy robusto en cuando a no caer en óptimos locales, puesto que las operaciones de coordinas de las abejas evita este problema.

3.2.5.5. Resultados

Cuando el número de ciclos de búsqueda se cumple, se determina dentro del registro de las fuentes de alimento, la posición de la fuente de néctar que presenta el mínimo costo global o en efecto, el mejor desempeño como parte del controlador. Esta posición, almacenada en el vector 3.27, equivale a la solución más óptima para el problema de búsqueda.

Derivado al desarrollo del algoritmo, también se incluye las gráficas que representan la evolución durante cada ciclo de búsqueda. Con estas gráficas se pretende exponer la exploración del espacio de búsqueda, el costo generado y el desempeño alcanzado. Por ejemplo, en la figura 51 se muestra los puntos explorados por los tres grupos de abejas dentro de los límites de espacio de búsqueda.

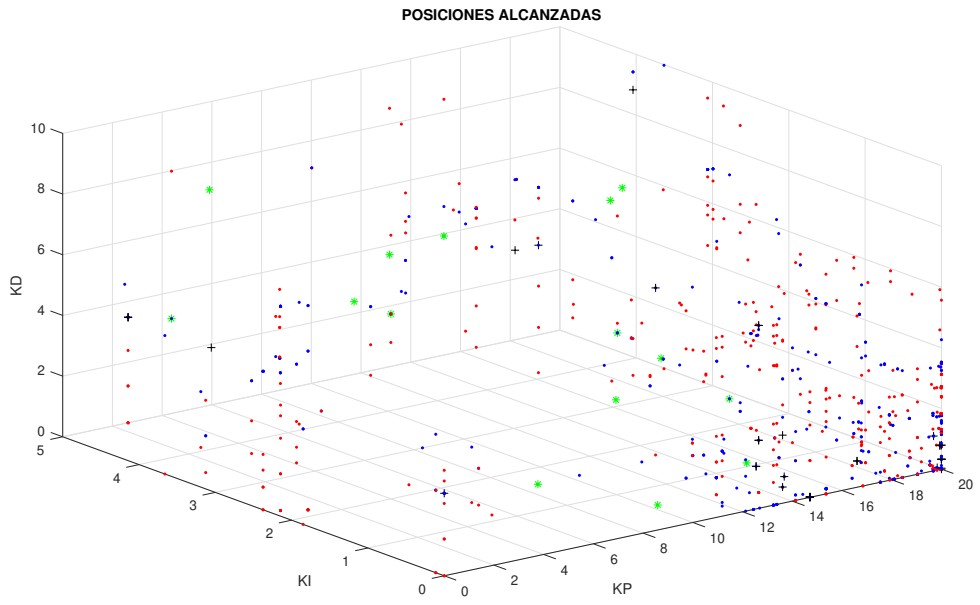


Figura 51. Ejemplo: Exploración del espacio de búsqueda en ABC.

En las gráficas de las figuras 52 y 53, generadas en base al costo global, se muestra la localización en un espacio tridimensional de los valores de costo global generados por las abejas empleadas, observadoras y exploradoras (diferenciadas por colores), y el historial de costo global generado durante cada ciclo, respectivamente.

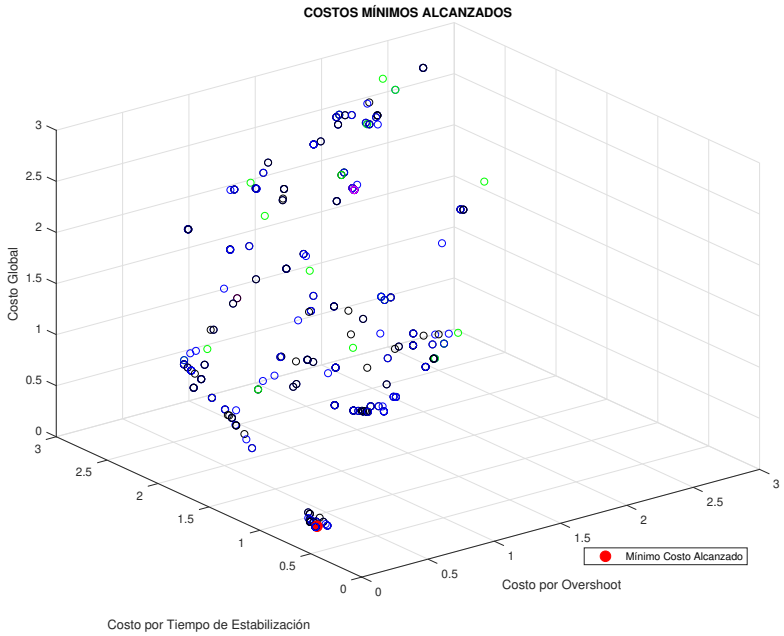


Figura 52. Ejemplo: Costos globales resultantes del algoritmo ABC.

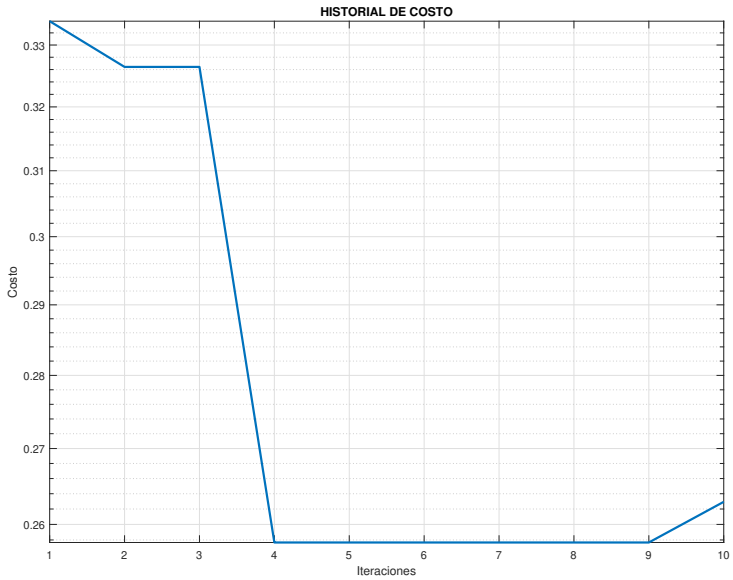


Figura 53. Ejemplo: Historial de los valores de costo globales alcanzados en ABC.
Ejemplificación del historial del costo global alcanzado por las posibles soluciones durante la ejecución del algoritmo ABC.

Mientras que en la figura 54, se presenta la localización tridimensional del desempeño global alcanzado durante toda la buscada por los tres grupos de abejas.

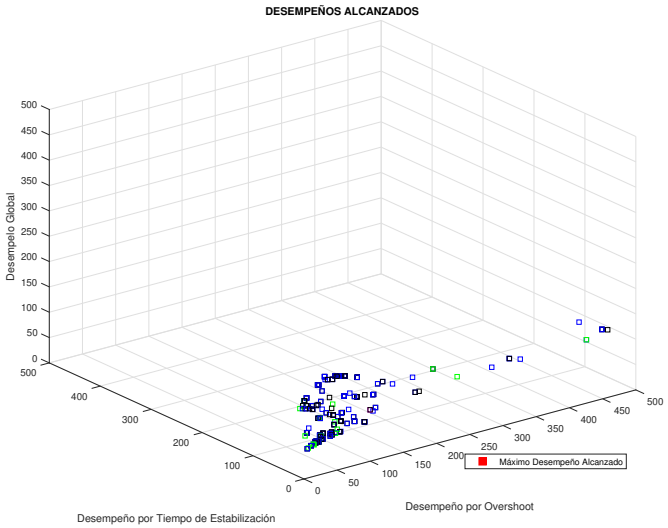


Figura 54. Ejemplo: Desempeños globales resultantes del algoritmo ABC.
Ejemplificación de la presentación de resultados: Desempeño global alcanzado por las posibles soluciones durante la ejecución del algoritmo ABC.

Y para concluir, se muestra una gráfica comparativa de las respuestas del sistema mediante el uso del controlador sintonizado por medio de las soluciones generadas en base a los tres parámetros de evaluación: sobreimpulso, tiempo de establecimiento y cumplimentero global (fig.55).

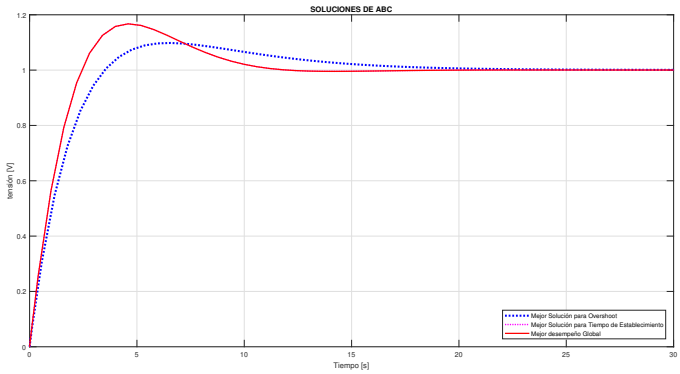


Figura 55. Ejemplo: Respuesta de los controladores obtenidos mediante ABC.

Algoritmo 3: COLONIA DE ABEJAS ARTIFICIALES - ABC

Datos: Parámetros de control del algoritmo ABC.

Inicio
  Definir: *Función de evaluación*.
  Generar la posición aleatoria en *D-dimensiones* para *N*-fuentes de néctar.
  Evaluar la cantidad de néctar de las *N*-fuentes.
  Memorizar la mejor *Fuente de Néctar* actual.
  Mientras *la condición de parada no se cumpla* **hacer**
    // *Fase de Abejas Empleadas*
    Para $i = 1$ *hasta* $i = N$ **hacer**
      Localizar una nueva fuente vecina a la actual y *evaluar* su cantidad de néctar.
      Si *la nueva fuente tiene mayor cantidad de néctar que la fuente actual* **entonces**
        Memorizar la nueva fuente localizada y reiniciar contador de abandono.
      fin
      En otro caso aumentar tendencia de abandono de la fuente actual. **fin**
    fin
    Calcular probabilidad p_i según la ecuación 2.10.
    // *Fase de Abejas Observadoras*
    Repetir
      Actualizar contador de repeticiones: *S*.
      Si *random* $< p_i$ **entonces**
        Localizar una nueva fuente vecina actual y *evaluar* su cantidad de néctar.
        Si *la nueva fuente tiene mayor cantidad de néctar que la fuente actual* **entonces**
          Memorizar la nueva fuente localizada y reiniciar contador de abandono.
        fin
        En otro caso aumentar tendencia de abandono de la fuente actual. **fin**
      fin
    hasta que $s<N$;
    // *Fase de Abejas Exploradoras*
    Buscar fuente con mayor tendencia de abandono.
    Si *el valor de la tendencia de abandono* $>$ *Límite de abandono* **entonces**
      Generar aleatoriamente una nueva fuente de néctar y *evaluar* su cantidad de néctar.
      Remplazar la fuente actual y reiniciar contador de abandono.
    fin
    Explorar y memorizar la mejor *Fuente de Néctar* actual para la colmena.
  fin
  Devolver: Mejor *Fuente de Néctar* encontrada.
Fin

Capítulo 4

Análisis del Desempeño

En este capítulo se describen los resultados alcanzados tras la implementación de los controladores sintonizados mediante los algoritmos GA, PSO y ABC, en la planta de flujo de aire caliente del módulo PCT-2.

Posteriormente, se presenta el análisis comparativo entre los diferentes métodos de sintonización y en base a las respuestas obtenidas del sistema controlado. Mernik et al. (2015).

4.1. Especificaciones Generales

Para lograr comparar el funcionamiento de los controladores diseñados a través de algoritmos bio-inspirados en una planta de flujo de aire caliente (módulo PCT-2), se fijó varios parámetros de control que ya se ha detallado en la tabla 3.

En resumen, mediante los métodos de diseño de controladores tipo PID desarrollados en el anterior capítulo, se esperaba conseguir que los controladores permitan, durante el régimen transitorio, una señal de salida con un sobreimpulso menor al 20 %, y en régimen permanente, que el tiempo de establecimiento de la señal sea menor a 20 *segundos* con un criterio del 2 % de error de estado estable.

Con referencia al cumplimiento de los objetivos de control establecidos para el presente trabajo, se procedió a la creación de la función de evaluación, conforme se ha descrito en la página 65. Además, para lograr un balance entre el costo en base al sobreimpulso y el costo en base a tiempo de estabilización, se fijó un peso del 50 % para w_1 y para w_2; así, la suma pondera que se opera en el costo en base al desempeño global sería equitativa. El código de dicha función se adjunta en el Anexo A.4.

Otros aspectos generales, son las dimensiones y los límites del espacio de búsqueda. Para la ejecución de los tres algoritmos de búsqueda (GA, PSO y ABC), se aplicó los mismos rangos que se encuentran detallados en la tabla 7.

Tabla 7
Límites del espacio de búsqueda

Parámetro	Plano	Límite mínimo	Límite máximo
Kp	x	0	15.879
Ki	y	0	11.8191
Kd	z	0	5.7896

Como ya se mencionó en el tercer capítulo, las dimensiones del espacio de búsqueda se basan en el número de los parámetros de búsqueda, y los límites de cada plano se fijaron mediante los valores de las ganancias halladas en la sintonización del controlador por medio del método de la curva de reacción de ZN. (pág.60).

Por otro lado, el número del conjunto de buscadores (población, enjambre y colmena) y el número de iteraciones del proceso de búsqueda, en cada uno de los algoritmos bio-inspirados, fue el mismo para cada caso como se presenta en la tabla 8.

Tabla 8
Parámetros de ajuste general

Parámetros de ajuste general	Valor
Tamaño del conjunto de búsqueda	25
Número de iteraciones	50

4.2. Implementación

4.2.1. Sintonización de Controladores

4.2.1.1. Uso del Algoritmo Genético

Con el establecimiento de los objetivos de control definidos en la tabla 3 se procedió a la revisión final de la función de evaluación creada bajos los objetivos de control (ver Axeno A.4).

Y con los valores especificados en la tabla 9, se complementa todos los parámetros de ajuste requeridos para la ejecución del algoritmo GA (complementario a las tablas 7 y 8).

Tabla 9
Definición del espacio de búsqueda

Parámetro de ajuste	**Valor**
Longitud del parámetro de búsqueda	8 bits
Longitud total del cromosoma	24 bits
Porcentaje de cruce	85 %
Porcentaje de mutación	15 %

Una vez determinados todos los parámetros de ajuste del algoritmo GA y definida correctamente la función de evaluación, se procedió a la ejecución del código del algoritmo GA, indicado en el Anexo A.1, del cual se obtuvo los siguientes resultados.

La evaluación del costo alcanzó las posiciones presentadas en la figura 56. Donde cada punto representa el costo que se generó en cada generación, y finalmente se encuentra una solución con el mínimo costo alcanzado (ver valores finales en tabla 10). La figura 57, muestra la iteración en la que se alcanzó el mínimo costo; al ir mejorando la solución tras cada nueva generación, se observa en dicha figura que, el mínimo costo se alcanzó en aproximadamente la iteración N° 48.

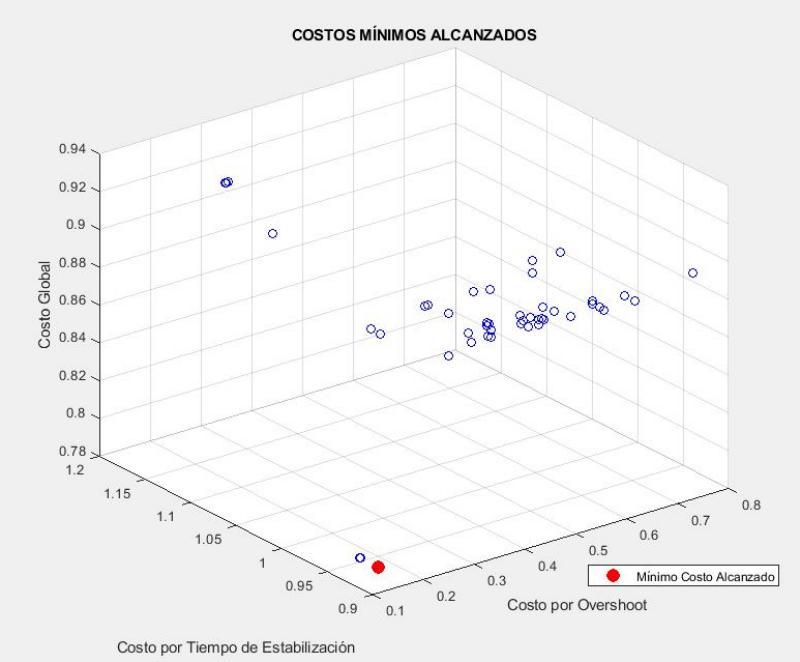


Figura 56. Costos alcanzados durante la búsqueda de ganancias a través de GA.

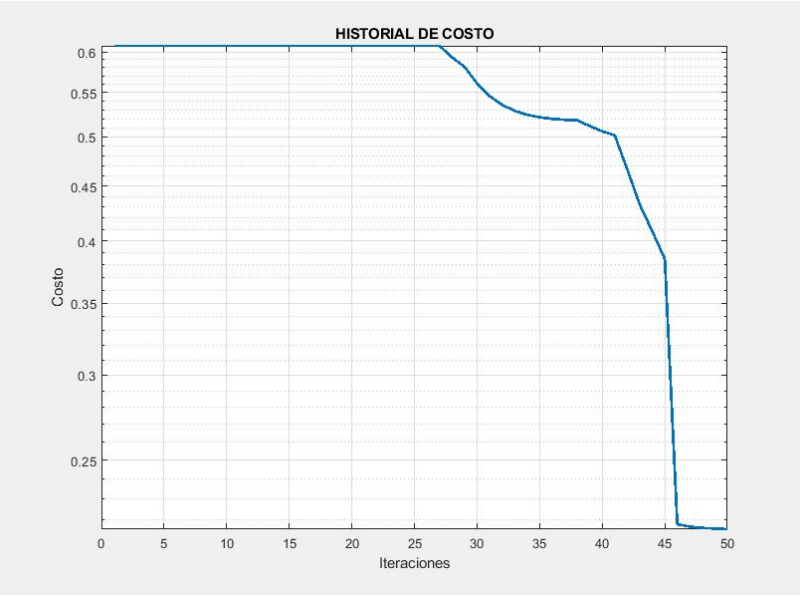


Figura 57. Historial de los costos alcanzados durante la búsqueda de ganancias a través de GA.

En cuanto a la evaluación del desempeño, las posiciones alcanzadas son proporcionalmente inversas a las posiciones generadas por el costo. En la gráfica de la figura 58 se marca el mayor desempeño alcanzado por alguna de las posibles soluciones.

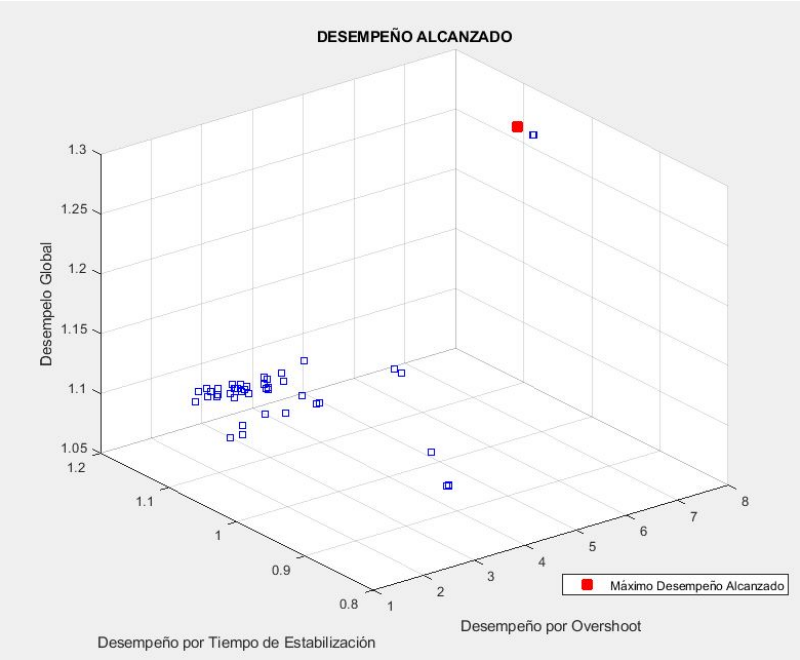


Figura 58. Desempeños alcanzados durante la búsqueda de ganancias a través de GA.

Finalmente, se presenta los valores de las soluciones generadas en base a los parámetros fijados en la tabla 10.

Tabla 10

Ganancias para un control PID mediante el uso de algoritmo GA

	Kp	**Ki**	**Kd**	**Costo**	**Desempeño**
En base a mejor Mp	15.5054	1.42427	4.66898	0.7906	1.2649
En base a mejor Ts	13.6995	6.38829	1.08075	0.9085	1.1007
Global	15.5054	1.42427	4.66898	1.4243	1.8864

4.2.1.2. Uso del Algoritmo PSO

Haciendo uso de la función de evaluación creada bajo los objetivos de control y complementando los parámetros de ajuste de las tablas 7 y 8 con la tabla 11, se procedió a la ejecución del algoritmo PSO para la búsqueda de las ganancias Kp, Ki y Kd.

Tabla 11
Parámetros de ajuste del algoritmo PSO.

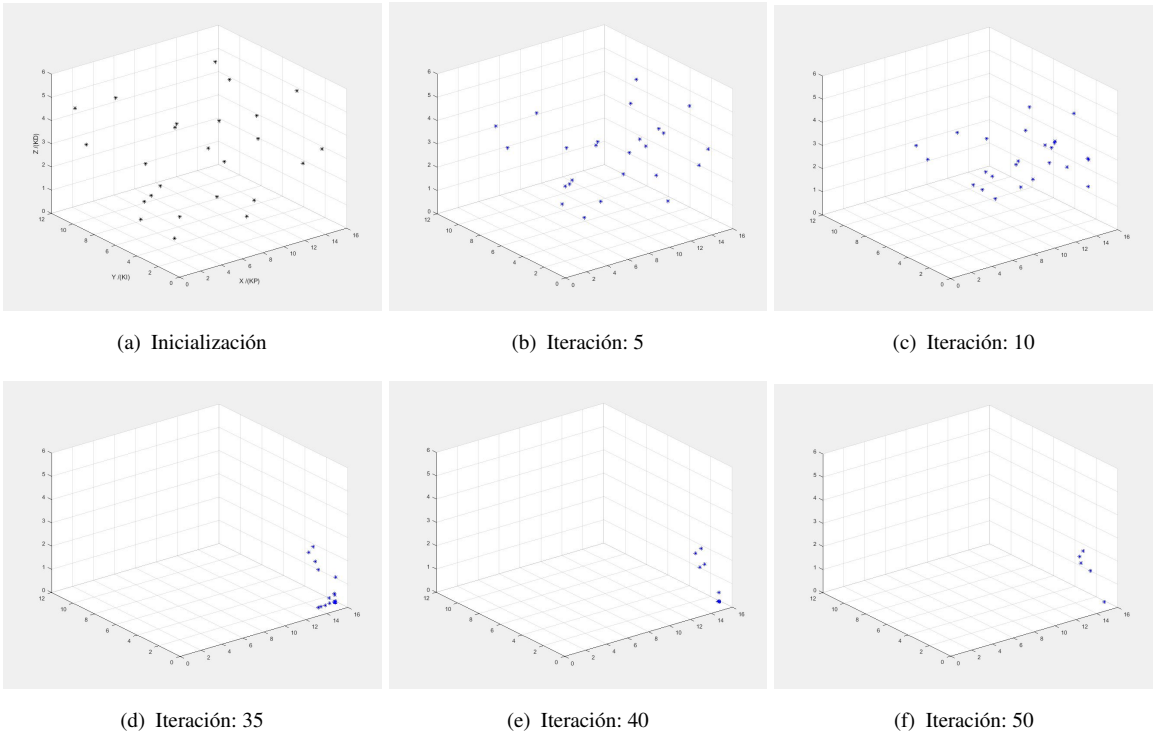
Parámetro de ajuste	**Valor**
Velocidad mínima de vuelo	$0{,}01 \times (Lim_{Max} - Lim_{Min})$
Velocidad máxima de vuelo	$-0{,}01 \times (Lim_{Max} - Lim_{Min})$
Coeficientes de aceleración	$0{,}1 \sim 4{,}0$
Factor inercial	$0{,}5 \sim 0{,}9$

Durante el desarrollo del algoritmo PSO, mediante la ejecución del código adjunto como Anexo A.2, se generó el movimiento del enjambre de partículas, dentro del espacio de búsqueda como se muestra en la figura 59.

La evolución de la exploración formuló diversas soluciones que alcanzaron diferentes valores de desempeño y costo. En la figura 60 se presenta la localización tridimensional de los costos generados y en la figura 61 el historial del costo *gBest* generado en cada iteración. Complementariamente, en la figura 62, se presenta las posiciones del desempeño alcanzado; en cada figura se resaltó la mejor respuesta alcanzada.

Como referencia a la figura 61, el mínimo costo se alcanza en aproximadamente la iteración N° 44; en la figura 59(d), también se constata que el agrupamiento de partículas se aproxima a la zona final a partir de la iteración N° 35.

En resumen, los valores del mínimo costo y el máximo desempeño asociados a una posición, se presentan en la tabla 12. Y en la tabla 13 se muestra específicamente las respuestas obtenidas como *gBest* de los criterios planteados en la función de evaluación.



(a) Inicialización (b) Iteración: 5 (c) Iteración: 10

(d) Iteración: 35 (e) Iteración: 40 (f) Iteración: 50

Figura 59. Movimiento de un enjambre de 25 partículas.

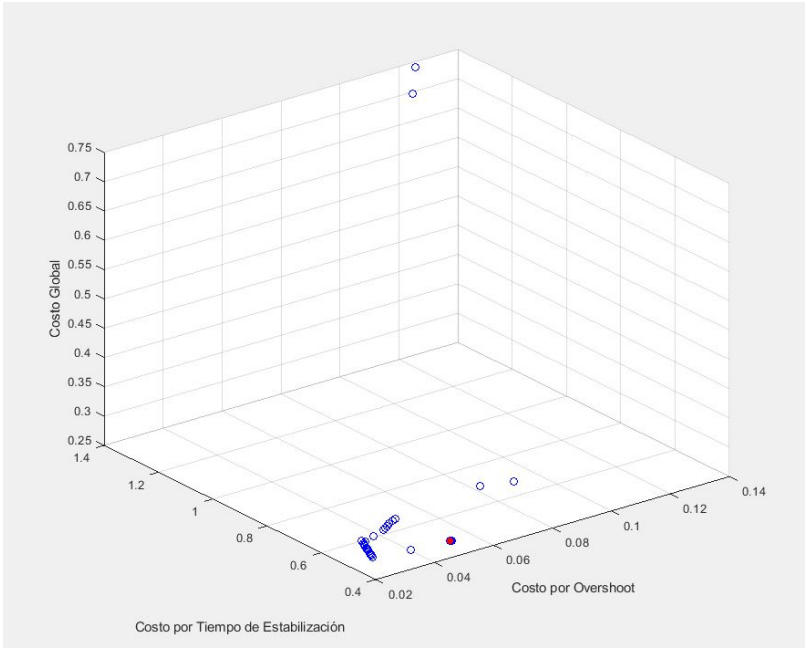


Figura 60. Costos alcanzados durante la búsqueda de ganancias a través de PSO.

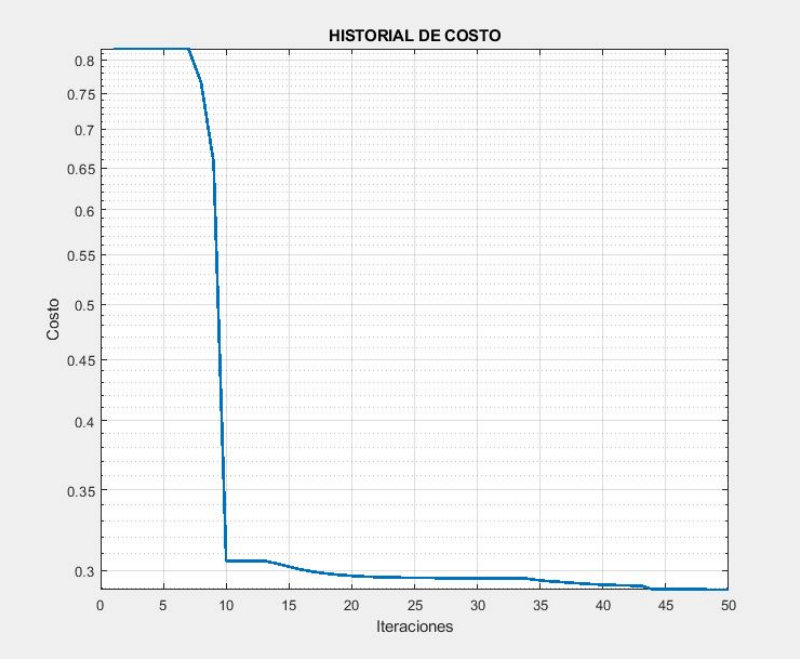


Figura 61. Historial de costos alcanzados durante la búsqueda de ganancias a través de PSO.

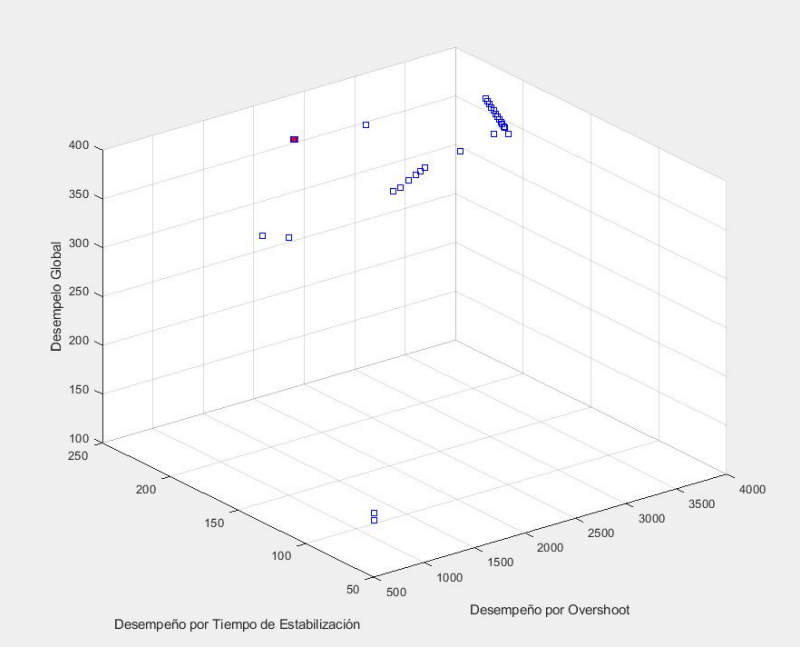


Figura 62. Desempeños alcanzados durante la búsqueda de ganancias a través de PSO.

Tabla 12
Mínimos costos y máximos desempeños asociados a mejores posiciones alcanzadas en PSO.

Parámetro de evaluación	**Posición** (x,y,z)			**Costo Alcanzado**	**Desempeño** $[\%]$
En base a Mp	(14,528	1,0464	1,2470)	0,2621	381,4964
En base a Ts	(14,527	1,1121	1,2522)	0,2633	379,7715
Glabal	(14,074	4,1140	2,6111)	0,2613	382,7629

Tabla 13
Ganancias para un control PID mediante el uso del algoritmo PSO.

Ganancia	**En base a mejor Mp**	**En base a mejor Ts**	**Global**
Kp	14.528	14.527	14.074
Ki	1.0464	1.1121	4.1140
Kd	1.2470	1.2522	2.6111

4.2.1.3. Uso del Algoritmo ABC

De igual manera a los casos anteriores, se hizo uso de la función de evaluación, generada en base a los objetivos de control, para la ejecución del algoritmo ABC como método de sintonización de un controlador tipo PID.

Los parámetros de ajuste propios del algoritmo ABC se presentan en la tabla 14, como complemento de los parámetros de ajuste globales que rigen los tres algoritmos bio-inspirados implementados (tabla 7 y 8).

Tabla 14
Parámetros de ajuste del algoritmo ABC.

Parámetro de ajuste	**Valor**
Número de fuentes de néctar	25
Límite de abandono	12

Tras la ejecución del código del algoritmo ABC (Anexo A.3) para la búsqueda de las ganancias óptimas de un controlador, se consiguió la inspección de casi todo el espacio de búsqueda como se muestra en la figura 63, y se obtuvo los valores especificados en la tabla 15. Estos valores hacen referencia a las posiciones de costos (fig.64) y desempeños (fig.66), alcanzados luego de completar cada fase de búsqueda (fase de abejas empleadas, observadoras y exploradoras). A cada fase se le fue asignado un color para facilitar la distinción de los costos y desempeños alcanzados por cada uno de los grupos de abejas (verde: fuentes iniciales, azul: fase de abejas empleadas, negro: fase de abejas observadoras y magenta: fase de abejas exploradoras).

Cabe aclara que, para efectuar el análisis comparativo frente a los resultados obtenidos mediante el uso de los algoritmos GA y PSO, se modificó el número de repeticiones del proceso de búsqueda en el algoritmo ABC; ya que, se tomó en cuenta el número de veces que se efectúa la evaluación de todo el conjunto de posibles soluciones (fuentes de néctar) durante todo el proceso ABC. Por ende, en la figura 65 se indica que el mínimo costo se generó en aproximadamente la iteración N° 8.

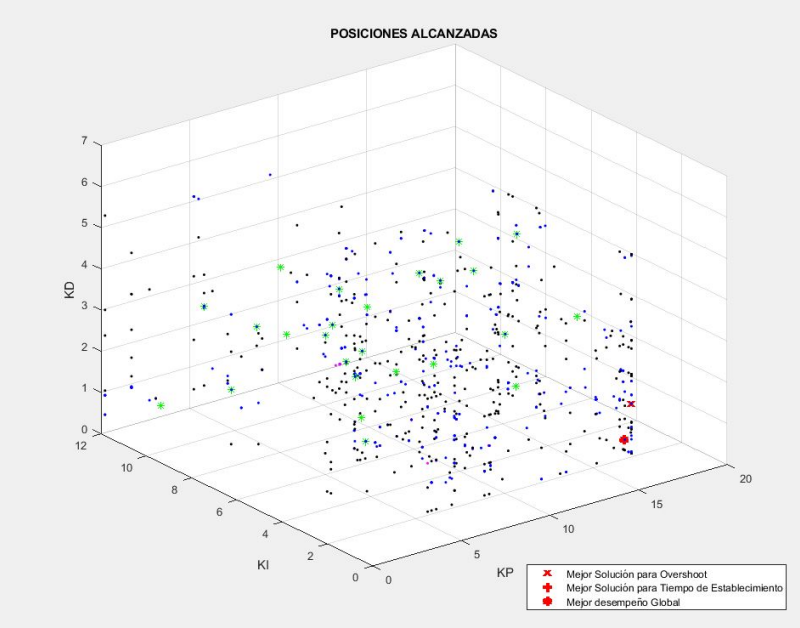


Figura 63. Exploración del espacio de búsqueda a través de ABC.

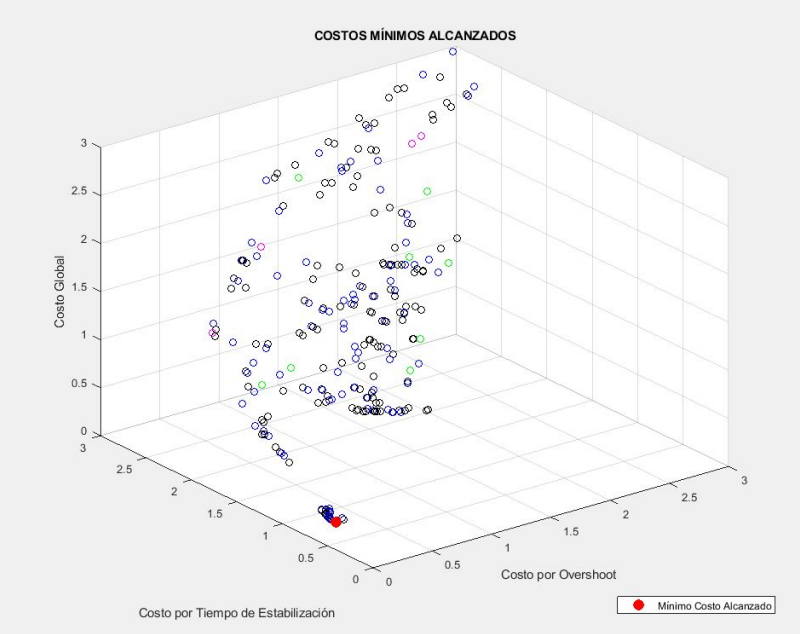


Figura 64. Costos alcanzados durante la búsqueda de ganancias a través de ABC.

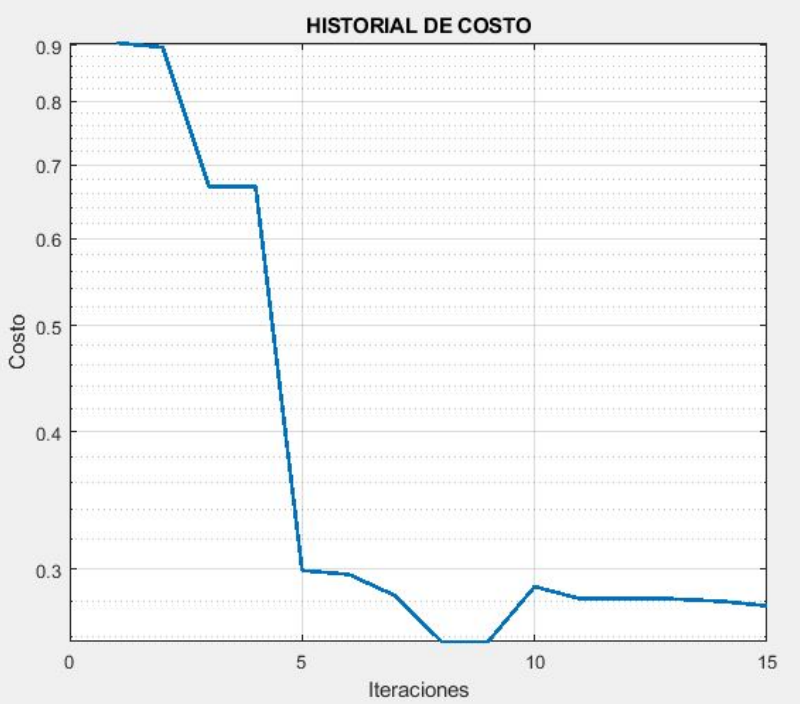


Figura 65. Historial de costos alcanzados durante la búsqueda de ganancias a través de ABC.

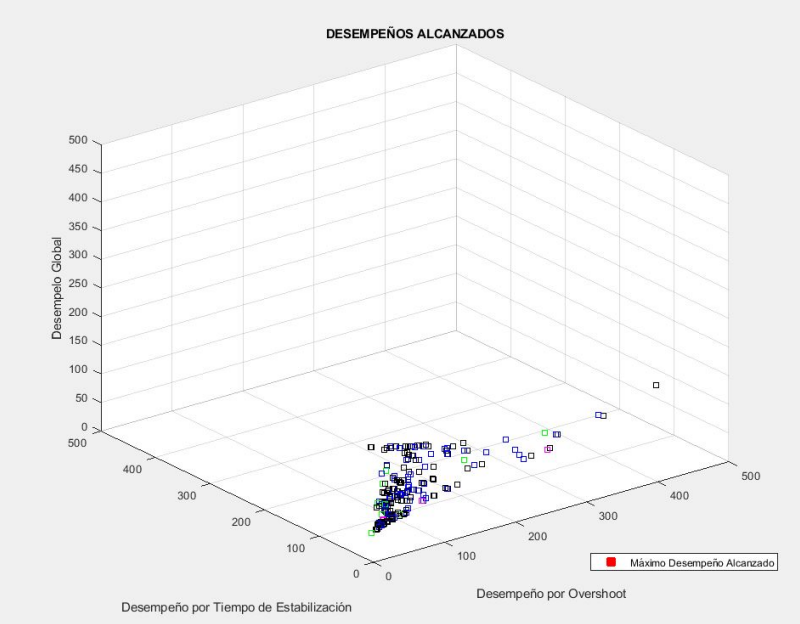


Figura 66. Desempeños alcanzados durante la búsqueda de ganancias a través de ABC.

Tabla 15
Mínimos costos y máximos desempeños asociados a mejores soluciones

Parámetro de evaluación	**Posición** (x,y,z)			**Costo Alcanzado**	**Desempeño** $[\%]$
En base a Mp	(15,8790	1,0000	1,7152)	0,0269	371,3
En base a Ts	(15,8790	1,2941	0,7664)	0,4401	227,2
Glabal	(15,8790	1,2941	0,7664)	0,2660	375,9

Por lo que, las ganancias finales que conforman el controlador, se resumen en la tabla 16.

Tabla 16
Ganancias para un control PID mediante el uso del algoritmo ABC

Ganancia	**En base a mejor Mp**	**En base a mejor Ts**	**Global**
Kp	15.8790	15.8790	15.8790
Ki	1.0000	1.2941	1.2941
Kd	1.7152	0.7664	0.7664

4.2.2. Instauración de controladores

A continuación, se presentan las señales de salida del sistema tras la aplicación de los valores de las ganancias halladas para el controlador tipo PID. Se adjunta, además, la cuantificación de las características de cada una de las señales.

a) ***Salida del sistema asociado al controlador diseñado por ZN***

En la figura 28 se presenta la señal de salida del sistema ante entrada escalón, con un periodo de muestreo de 30 segundos, y en la tabla 17 los valores característicos de la señal.

Tabla 17
Características de las señales de salida - Método ZN

Parámetro	**Valor Alcanzado**
Tiempo de subida	0,756s
Sobreimpulso	32,1 %
Tiempo de establecimiento	20s
Error de estado estable	2 %

b) ***Salida del sistema asociado al controlador diseñado por GA***

La figura 74 muestra la comparación de las señales de salida del sistema ante una entrada escalón, empleando los controladores diseñados mediante GA, en base a los tres criterios de evaluación: sobreimpulso, tiempo de estabilización y relación entre sobreimpulso y estabilización (criterio global). Las señales se encuentran diferenciadas una de otra para mejor visualización.

Como se puede apreciar (fig.74, la mejor respuesta en base al sobreimpulso ha sido tomada como la mejor respuesta global; ya que, esta salida cumple con los requerimientos de menor sobreimpulso y un tiempo de estabilización < 20 segundos. Sin embargo, con

un sobreimpulso ligeramente $> 20\,\%$ se consigue un tiempo de estabilización < 10 segundos. En la tabla 18, se detalla los valores exactos de las características de las señales de salida mostrada en la figura anterior.

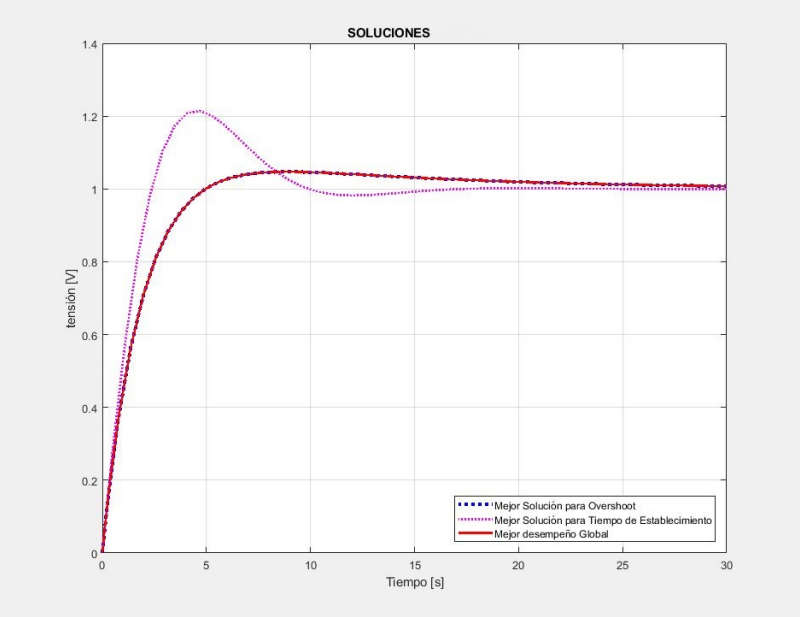


Figura 67. Respuesta de sistema ante entrada escalón mediante el uso del controlador diseñado por GA

Tabla 18
Características de las señales de salida - Método GA

Parámetro	**Valores Alcanzados**		
	En base a Mp	*En base a Ts*	*En base a criterio Global*
Tiempo de subida	3,2403s	1,8303s	3,2403s
Tiempo de establecimiento	16,2373s	9,1206s	16,2373s
Sobreimpulso	3,8941 %	21,4070 %	3,8941 %
Error de estado estable	2 %	2 %	2 %

c) ***Salida del sistema asociado al controlador diseñado por PSO***

La comparación de las señales de salida ante una entrada escalón al sistema, en conjunto con los controladores diseñados mediante PSO, se presentan en la figura 68. Adicionalmente, en la tabla 19 se especifica las características de las señales.

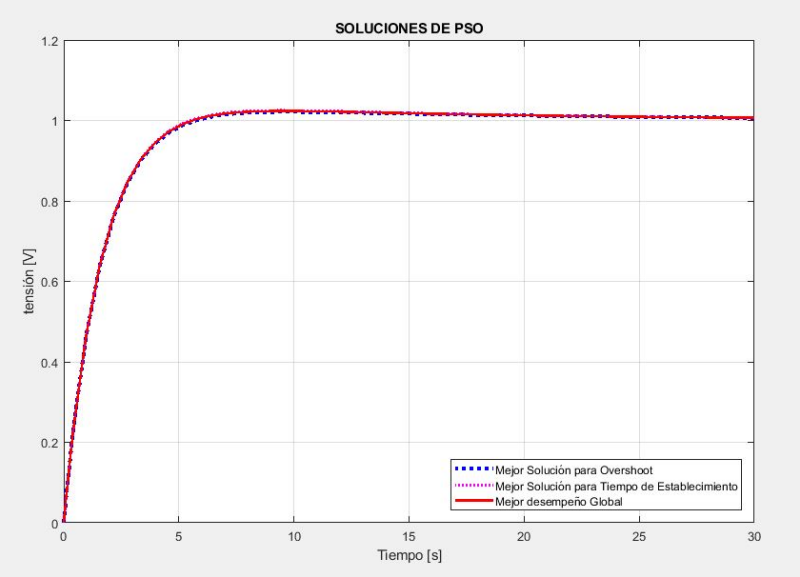


Figura 68. Respuesta de sistema ante entrada escalón mediante el uso del controlador diseñado por PSO

Mediante este método se consiguió resultados similares, puesto que, el proceso del algoritmo PSO intenta llevar la dirección de las posibles soluciones hacia un mismo punto; por ende, las respuestas son similares.

Tabla 19

Características de las señales de salida - Método PSO

Parámetro	**Valores Alcanzados**		
	En base a Mp	*En base a Ts*	*En base a criterio Global*
Tiempo de subida	3,2431s	3,1902s	3,2104s
Tiempo de establecimiento	5,1122s	4,9649s	5,0295s
Sobreimpulso	1,4789 %	1,8441 %	1,6718 %
Error de estado estable	2 %	2 %	2 %

d) ***Salida del sistema asociado al controlador diseñado por ABC***

Asimismo se presenta, en la figura 69, las tres señales de salida del sistema correspndientes a los tres controladores obtenidos por la sintonización a través del algoritmo ABC, y en la tabla 20, los valores característicos de las señales.

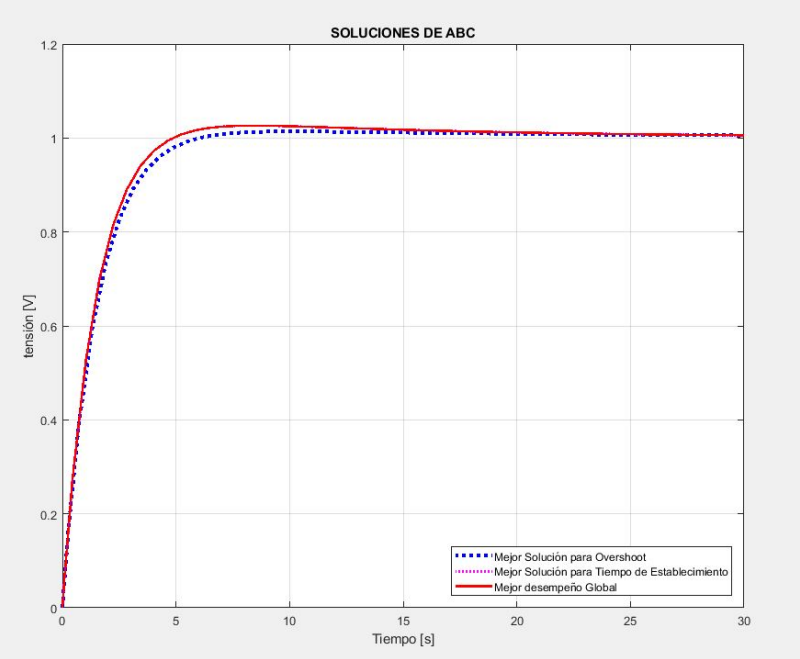


Figura 69. Respuesta de sistema ante entrada escalón mediante el uso del controlador diseñado por ABC

El algoritmo ABC, disminuyó el sobreimpulso de las señales resultantes a un valor $<$ 2,1 % de la señal de referencia. En cuanto al tiempo de estabilización, en todas las señales se consiguió un tiempo $<$ 10 %.

Tabla 20
Características de las señales de salida - Método ABC

Parámetro	**Valores Alcanzados**		
	En base a Mp	*En base a Ts*	*En base a criterio Global*
Tiempo de subida	3,1450s	2,8587s	2,8587s
Tiempo de establecimiento	9,3451s	9,2690s	9,2690s
Sobreimpulso	0,8448 %	2,0411 %	2,0411 %
Error de estado estable	2 %	2 %	2 %

En definitiva, todos los algoritmos generaron respuestas asociadas a un tiempo de establecimiento $< 20\,\%$. Pero el algoritmo que generó una solución con menor tiempo de establecimiento fue el algoritmo PSO con un tiempo $Ts = 4{,}9649s$, seguido del algoritmo ABC y finalmente el algoritmo GA que obtuvo una solución con un $Ts = 16,237s$. En cuanto al criterio de sobreimpulso, el algoritmo que alcanzó un menor porcentaje, fue el algoritmo ABC con un valore de $Mp = 0{,}8448\,\%$, seguido por el algoritmo PSO y finalmente el algoritmo GA.

4.3. Análisis Comparativo de Resultados

Se presenta una recopilación de las funciones de transferencias de los controladores, diseñados en base al tercer criterio de evaluación (relación entre sobreimpulso y tiempo de estabilización), utilizados en las pruebas realizadas (tabla 21).

Tabla 21
Controladores diseñados.

N°	**Método**	**Controlador**
1	Curva de reacción ZN	$G_{ZN} = \frac{5{,}789s^2 + 15{,}88s + 11{,}819}{s}$
2	Mediante Algoritmo Genético	$G_{ZN} = \frac{4{,}669s^2 + 15{,}51s + 1{,}424}{s}$
3	Mediante Algoritmo PSO	$G_{ZN} = \frac{2{,}611s^2 + 14{,}07s + 4{,}114}{s}$
4	Mediante Algoritmo ABC	$G_{ZN} = \frac{0{,}7664s^2 + 15{,}88s + 1{,}294}{s}$

4.3.1. Resultados Simulados

Para las pruebas respectivas de los controladores, se utilizó el entorno *Simulink* propio de Matalab, la referencia fijada fue de $1{,}5V$ (voltaje requerido para alcanzar una temperatura de $35°C$), por un tiempo total de muestreo de aproximadamente 30 segundos con intervalo de 0.5 segundos. Además, para comprobar la robustez de los controladores, se introdujo al sistema controlado una perturbación del $\pm 10\,\%$ de la señal de referencia, $0{,}15[V]$, 50 segundos después de haber iniciado la acción de control, y otra de $-0{,}15[V]$, 90 segundos después de haber iniciado la acción de control; la respuesta de cada uno de los controladores implementados se presenta subsiguientemente.

Se inició con el controlador sintonizado mediante el método de la curva de reacción de ZN. La señal de salida del sistema, alcanzó un tiempo de estabilización < 20 segundos; sin embargo,

el sobreimpulso de la señal fue mayor al 20 %, seguido de un subimpulso de aproximadamente el 5 %. La respuesta simulada del controlador se visualiza en la figura 70, y sus valores característicos exactos (requeridos para la comparación) en la tabla 22.

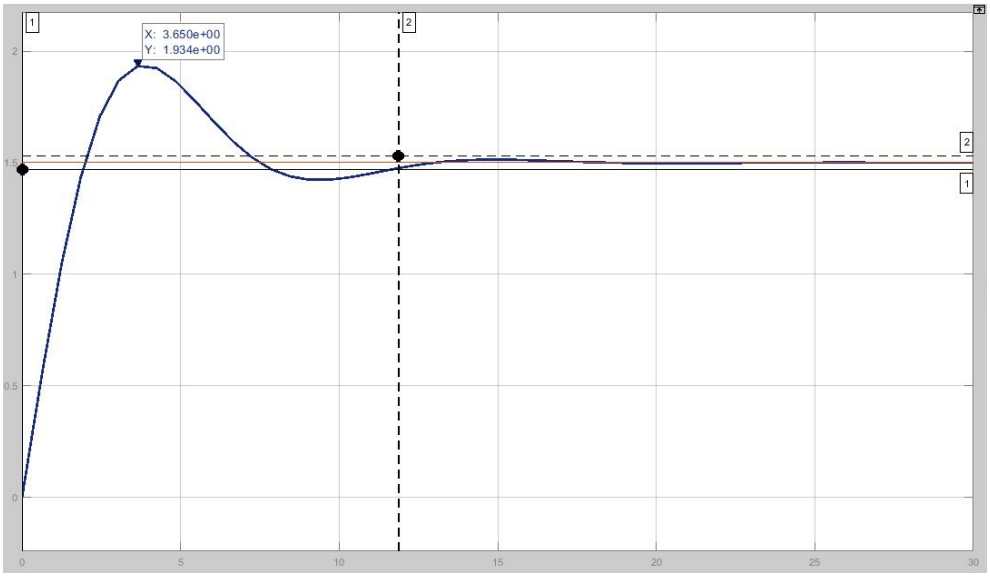


Figura 70. Simulación de la salida del sistema ante entrada escalón mediante el uso del controlador diseñado por ZN

Tabla 22
Respuesta Simulada - Método ZN

Parámetro	**Valor Alcanzado**
Valor máximo	1,934V
Tiempo de establecimiento	11,875s
Sobreimpulso	28,933 %
Error de estado estable	2 %

En la figura 71, se observa la respuesta ante dos perturbaciones. La primera perturbación eleva la señal a un valor de aproximadamente 1,53$[V]$, y la segunda perturbación disminuye la señal a un valor de 1,47$[V]$. A pesar que la señal no supera los límites del 2 % de error de estado estable con la introducción de las perturbaciones, esta se recupera en 5,43$[s]$ en las dos perturbaciones (ver figuras 72 y 73).

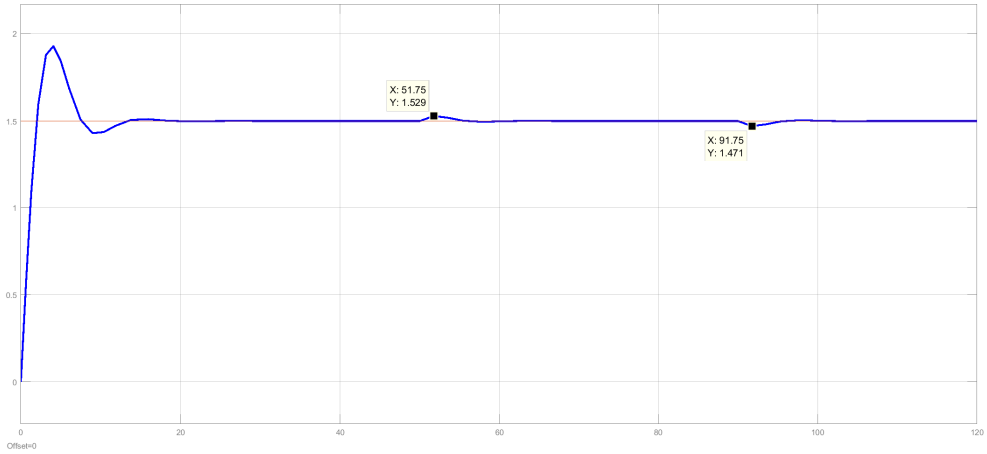


Figura 71. Simulación de la salida del sistema ante perturbaciones - Método ZN

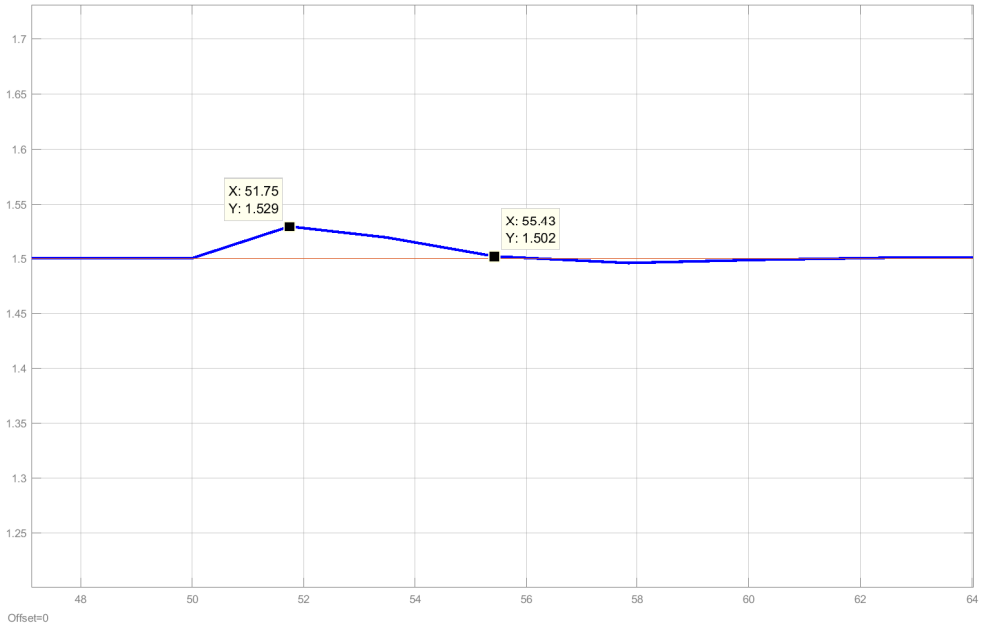


Figura 72. Simulación de la salida del sistema ante primera perturbación - Método ZN

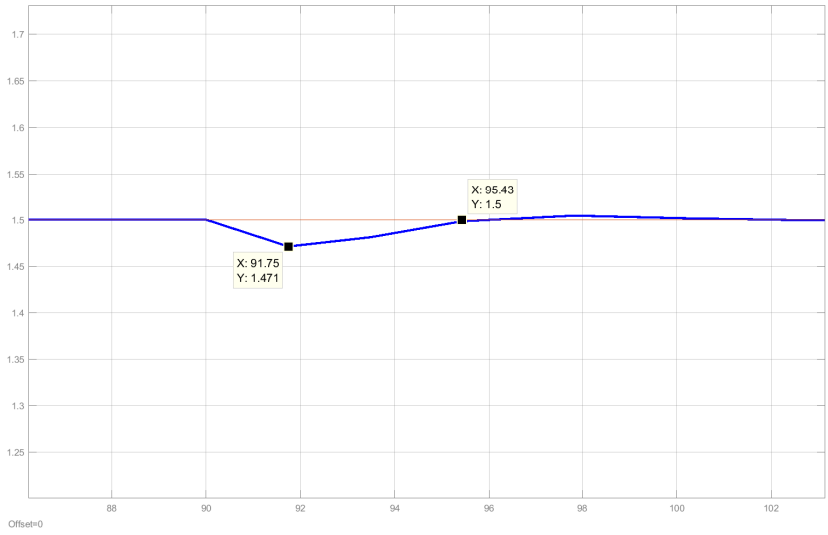


Figura 73. Simulación de la salida del sistema ante segunda perturbación - Método ZN

Por otro lado, la salida del sistema controlado mediante el uso de la función de trasferencia N° 2 de la tabla 21 como controlador (asociado al algoritmo GA), presentó la señal mostrada en al figura 74, (sus valores característicos se presentan en la tabla 23).

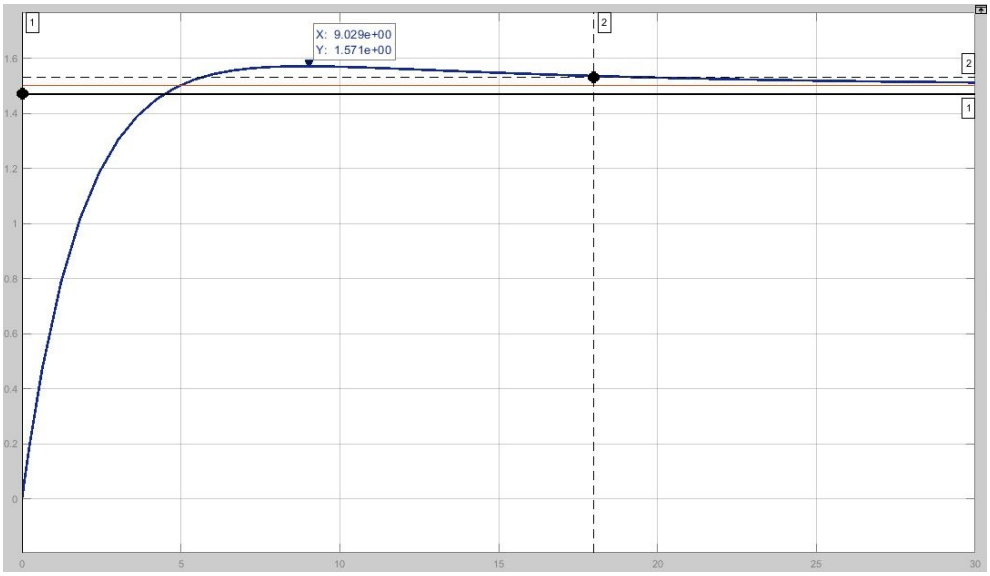


Figura 74. Simulación de la salida del sistema ante entrada escalón mediante el uso del controlador diseñado por GA

Tabla 23
Respuesta Simulada - Método GA

Parámetro	**Valor Alcanzado**
Valor máximo	1,571V
Tiempo de establecimiento	18s
Sobreimpulso	4,7338 %
Error de estado estable	2 %

En cuanto al sistema expuesto a perturbaciones, la señal de salida (fig.75), se elevó a un valor de 1,548$[V]$ en la primera perturbación (fig.76), y en la segunda perturbación disminuyó a 1,456$[V]$ (fig.77). El tiempo de recuperación fue de 12,34$[s]$ (cumpliendo el criterio del 2 % de error de estado estable) y de aproximadamente 26,74$[s]$ para eliminar el error de estado estacionario, con la primera perturbación; con la segunda perturbación, el tiempo de recuperación fue de aproximadamente 14,7 segundos.

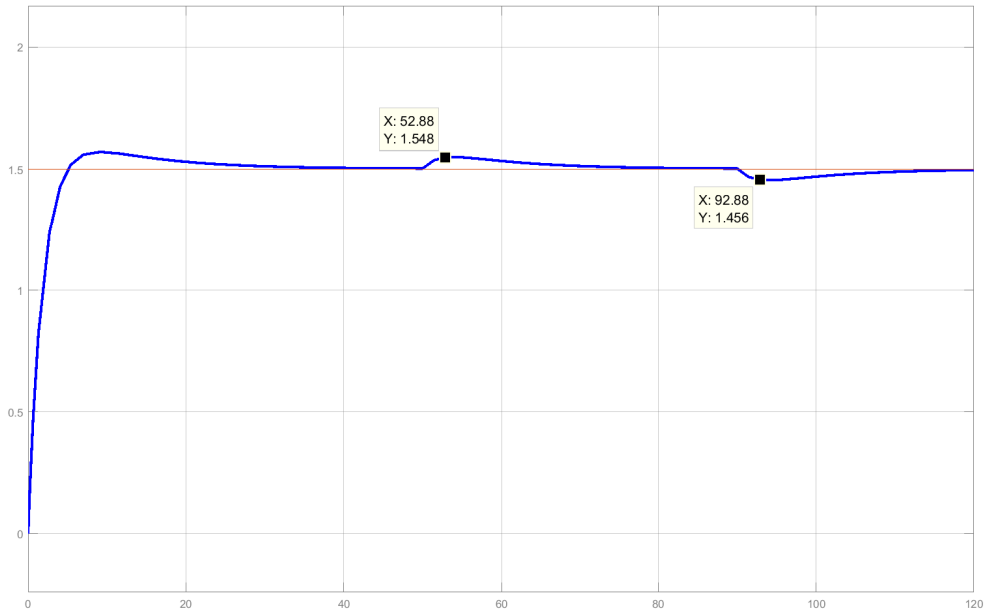


Figura 75. Simulación de la salida del sistema ante perturbaciones - Método GA

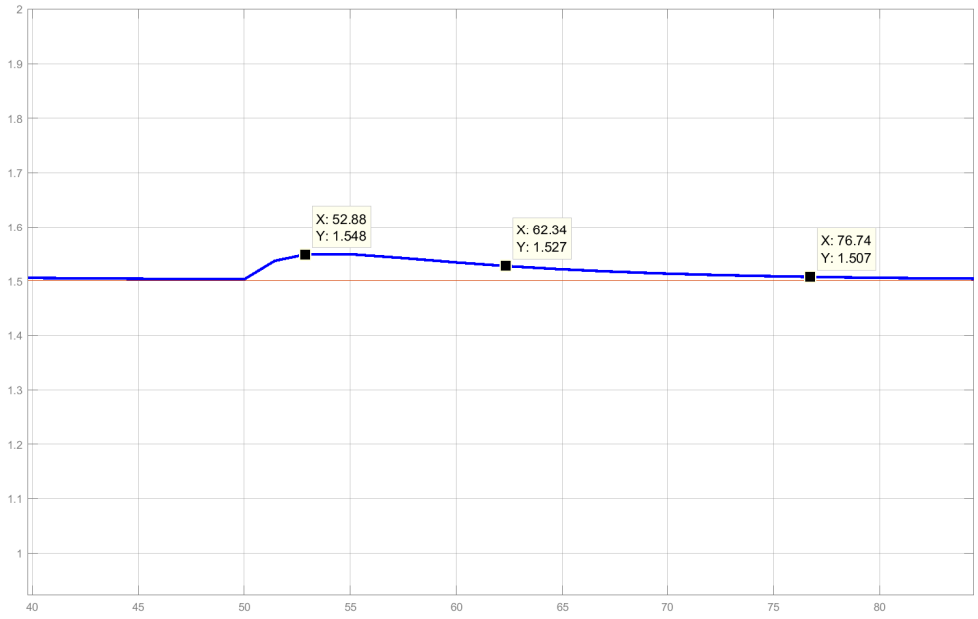


Figura 76. Simulación de la salida del sistema ante primera perturbación - Método GA

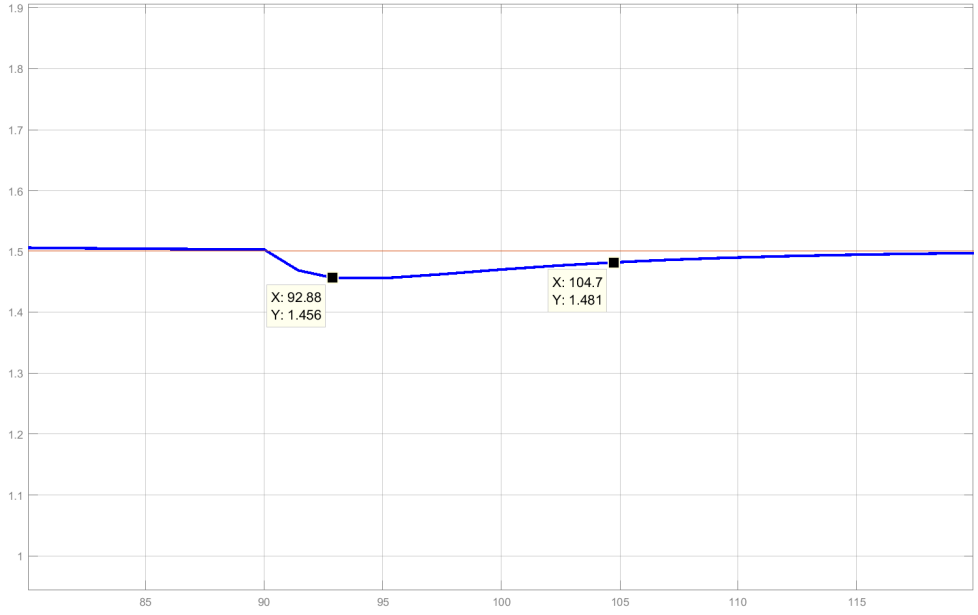


Figura 77. Simulación de la salida del sistema ante segunda perturbación - Método GA

En cambio, en la figura 78 se expone la señal de salida del controlador diseñado mediante el algoritmo PSO; como se puede evidencia, la respuesta presentó un sobreimpulso $< 5\,\%$ y un tiempo de estabilización menor al $17\,\%$. Los valores exactos de las características están detallados en la tabla 24.

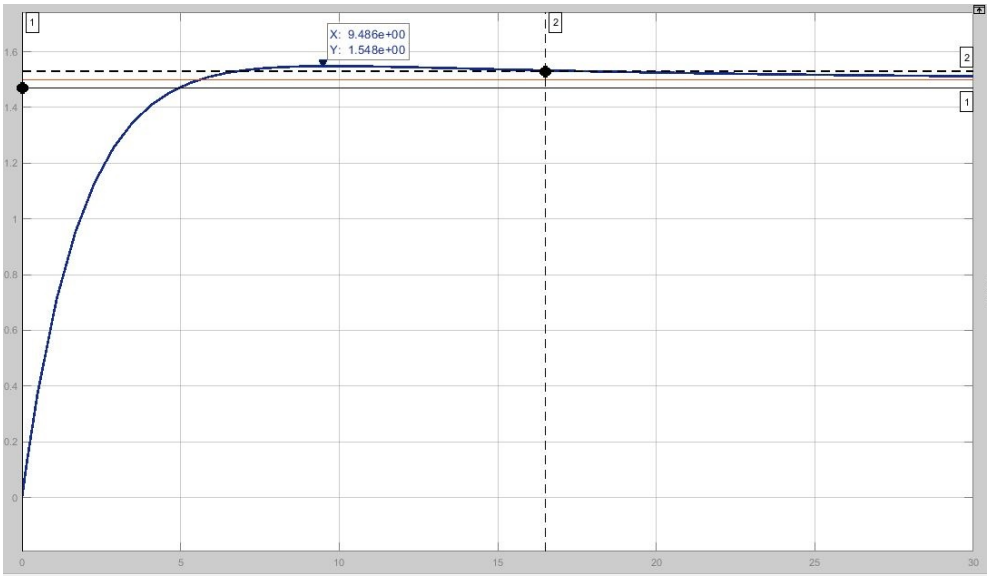


Figura 78. Simulación de la salida del sistema ante entrada escalón mediante el uso del controlador diseñado por PSO

Tabla 24
Respuesta Simulada - Método PSO

Parámetro	**Valor Alcanzado**
Valor máximo	$1{,}548V$
Tiempo de establecimiento	$16{,}50s$
Sobreimpulso	$3{,}2\,\%$
Error de estado estable	$2\,\%$

La introducción de perturbaciones en este sistema, hizo que la señal se elevara a $1{,}54[V]$ y disminuyera a $1{,}46[V]$ (fig.79). El tiempo de recuperación, en el primer caso, fue de 6,08 segundos y 8,48 segundos para eliminar el error (80); en el segundo caso, el tiempo de recuperación fue de 6,08 segundos y para eliminar el error tardó 10,9 segundos (81).

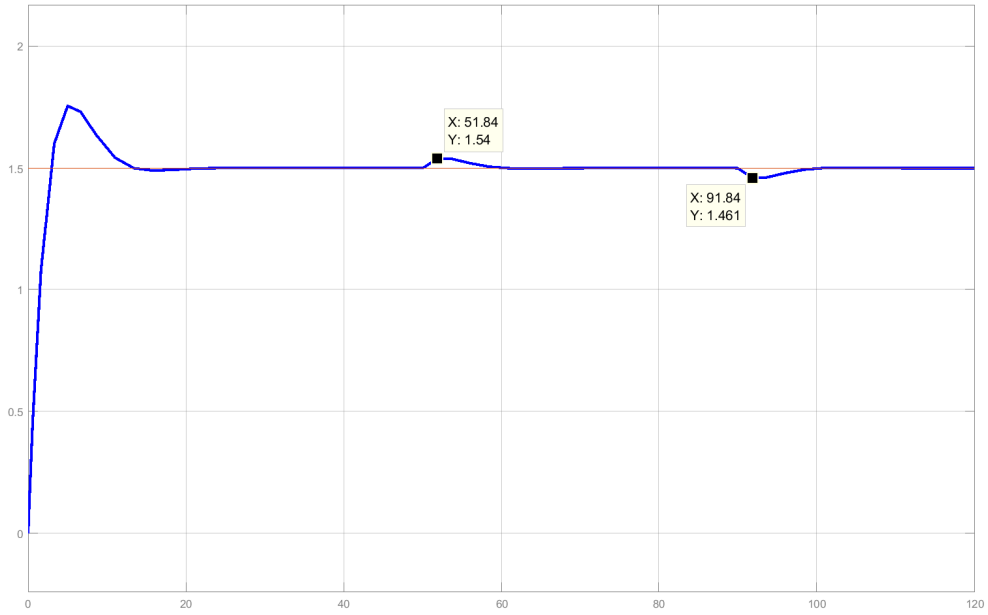


Figura 79. Simulación de la salida del sistema ante perturbaciones - Método PSO

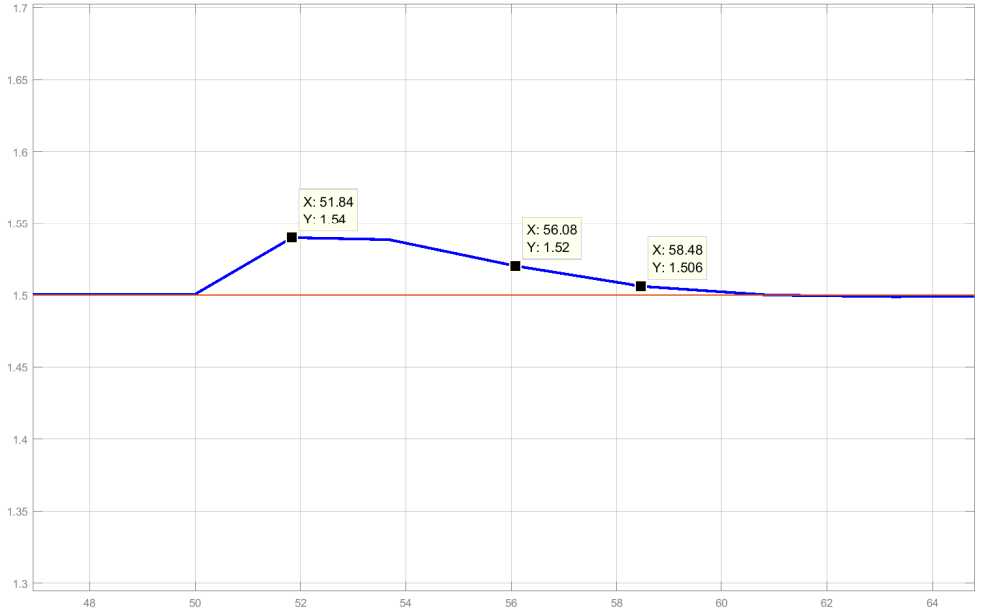


Figura 80. Simulación de la salida del sistema ante primera perturbación - Método PSO

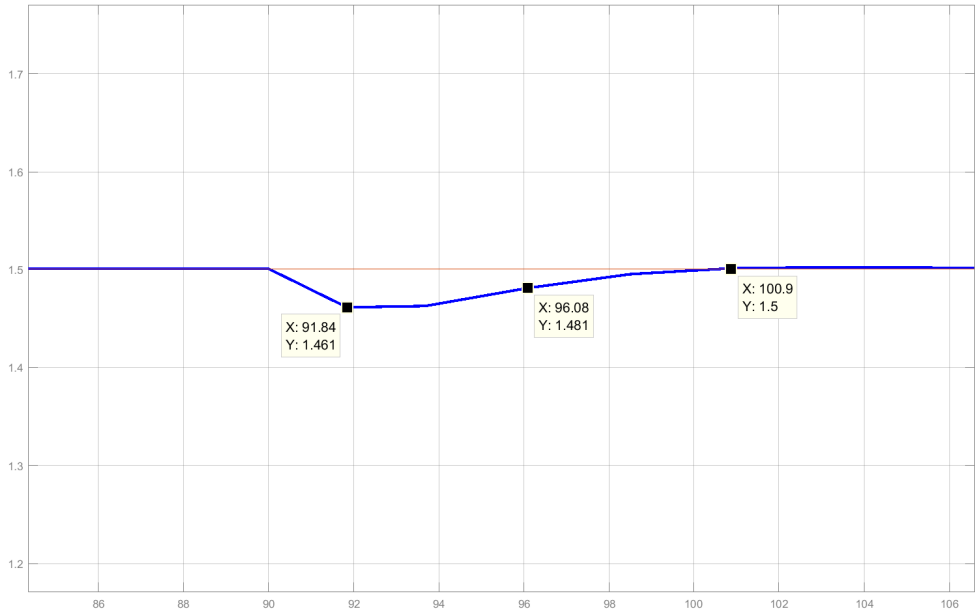


Figura 81. Simulación de la salida del sistema ante segunda perturbación - Método PSO

Siguiendo con la presentación de resultado, en la figura 82 se muestra la salida correspondiente del controlador diseñado mediante el algoritmo ABC. En este caso, se consiguió obtener una salida ligeramente subamortiguada. En la tabla 25 se especifica los valores alcanzados.

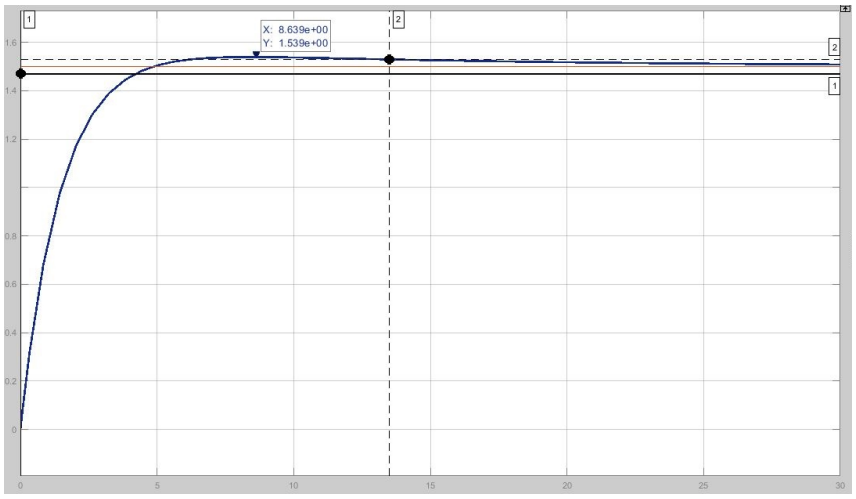


Figura 82. Simulación de la salida del sistema ante entrada escalón mediante el uso del controlador diseñado por ABC

Tabla 25
Respuesta Simulada - Método ABC

Parámetro	**Valor Alcanzado**
Valor máximo	1,539V
Tiempo de establecimiento	13,50s
Sobreimpulso	2,60 %
Error de estado estable	2 %

El sistema, además respondió ante las perturbaciones como se muestra en la figura 83. Con la primera perturbación (fig.84), el tiempo de recuperación fue de 13,09 segundos; y con la segunda perturbación (fig.85), el tiempo de recuperación fue de 15,5 segundos.

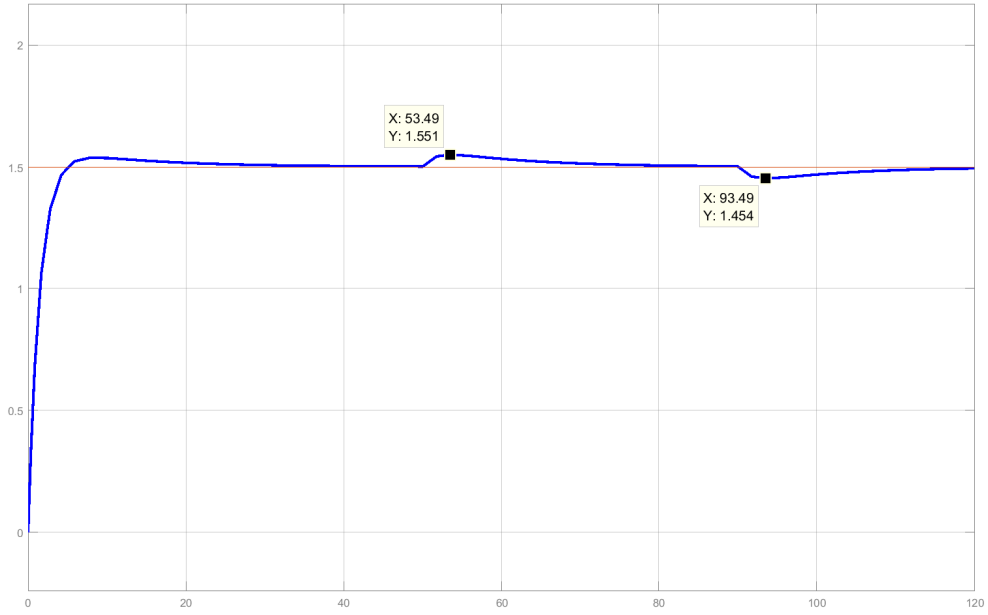


Figura 83. Simulación de la salida del sistema ante perturbaciones - Método ABC

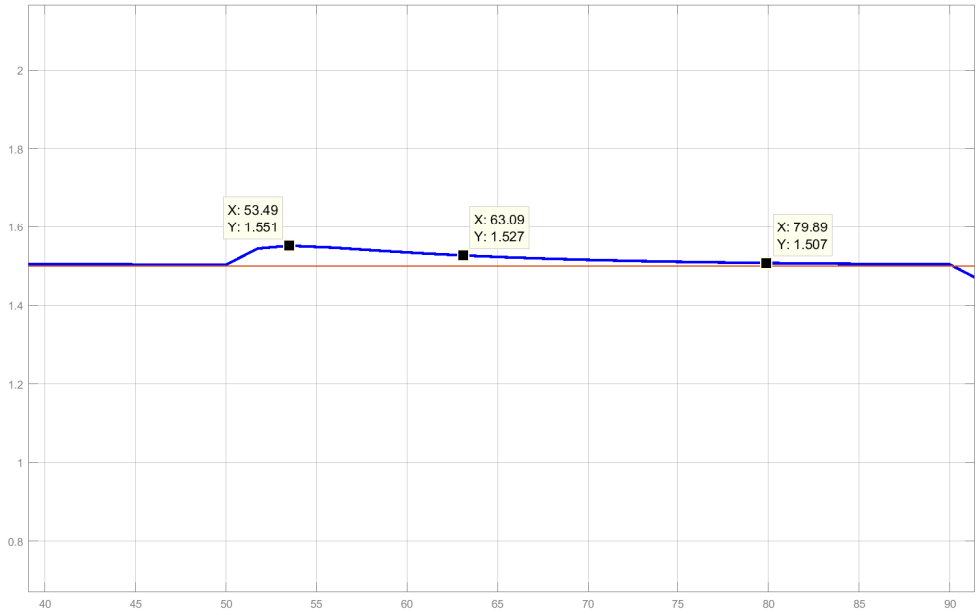


Figura 84. Simulación de la salida del sistema ante primera perturbación - Método ABC

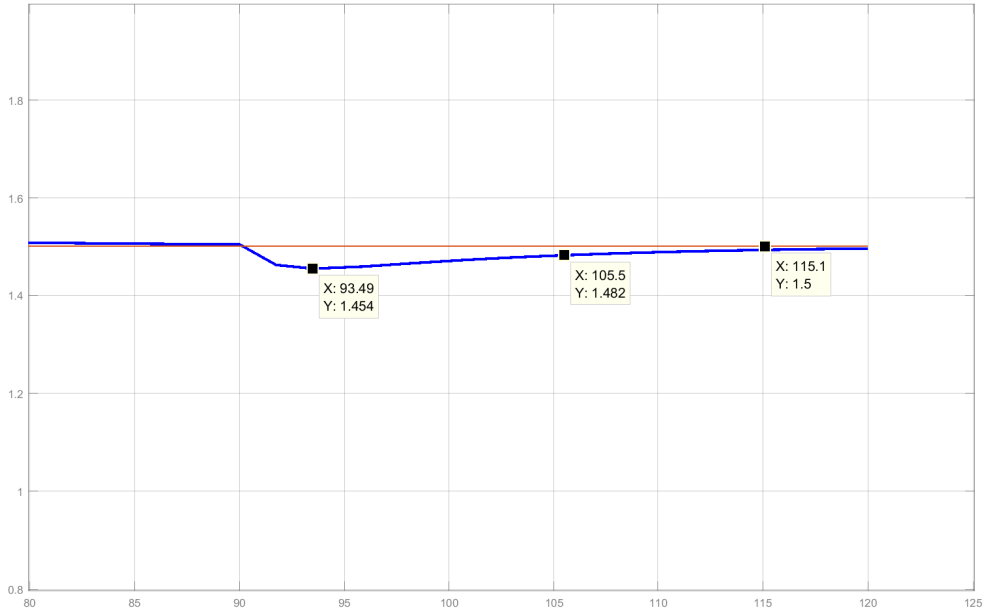


Figura 85. Simulación de la salida del sistema ante segunda perturbación - Método ABC

En base a las respuestas obtenidas, se presenta una tabla comparativa entre los resultados simulados alcanzados por los controladores diseñados a través de los métodos ZN, GA, PSO y ABC (tabla 26). Así mismo, en la tabla 27, se resume los tiempos de recuperación de cada uno de los controladores.

Tabla 26
Comparación de las respuestas simuladas.

Controlador	**Valores característicos**		
Método	*Valor Máximo*	*Sobreimpulso*	*Tiempo de Estabilización*
ZN	1,934*V*	28,933 %	11,88*s*
GA	1,571*V*	4,734 %	18,00*s*
PSO	1,548*V*	3,201 %	16,50*s*
ABC	1,539*V*	2,602 %	13,50*s*

Tabla 27
Tiempo de recuperación ante perturbaciones simuladas.

Método de diseño del	**Tiempo de recuperación**	
Control PID	**Primera Perturbación**	**Segunda Perturbación**
Método ZN	5,43*s*	5,43*s*
Algoritmo GA	12,0*s*	14,7*s*
Algoritmo PSO	06,08*s*	6,08*s*
Algoritmo ABC	13,09*s*	15,5*s*

4.3.2. Resultados Medidos

Como se indica en el capítulo III, se empleó la tarjeta de control Teensy 3.6 y se acopló al módulo PCT-2 para lograr el control de flujo de aire caliente a una temperatura de 35°C (1,5$[V]$); la acción de control se ejecutó cada 25$[ms]$. En este apartado, se presentan las señales obtenidas de cada uno de los controladores ante una entrada escalón y el detalle de los valores característicos de las señales de salida en la tabla 28.

En cuanto al rechazo a perturbaciones, se realizó la interrupción de ingreso de aire frío al sistema controlado de manera parcial (dos diferentes posiciones de obstrucción). Normalmente, el aire ingresa por un área de 17,35cm^2 como se muestra en la figura 23; sin embargo, para crear una perturbación se limitó el ingreso del aire por un área de 8,67cm^2 para la primera perturbación y luego por un área de 13,01cm^2 para la segunda perturbación. El sistema respondió de acuerdo al controlador empleado. En las siguientes imágenes se muestra las señales de salida resultantes.

En el sistema asociado al método ZN, la salida del controlador (fig.86), tuvo un sobreimpulso de 19,067 %, un subimpulso de 2,93 % y un tiempo de estabilización de 16,77$[s]$; no obstante, presenta una ligera oscilación en su estado estable, pero se mantiene dentro del criterio del 2 %.

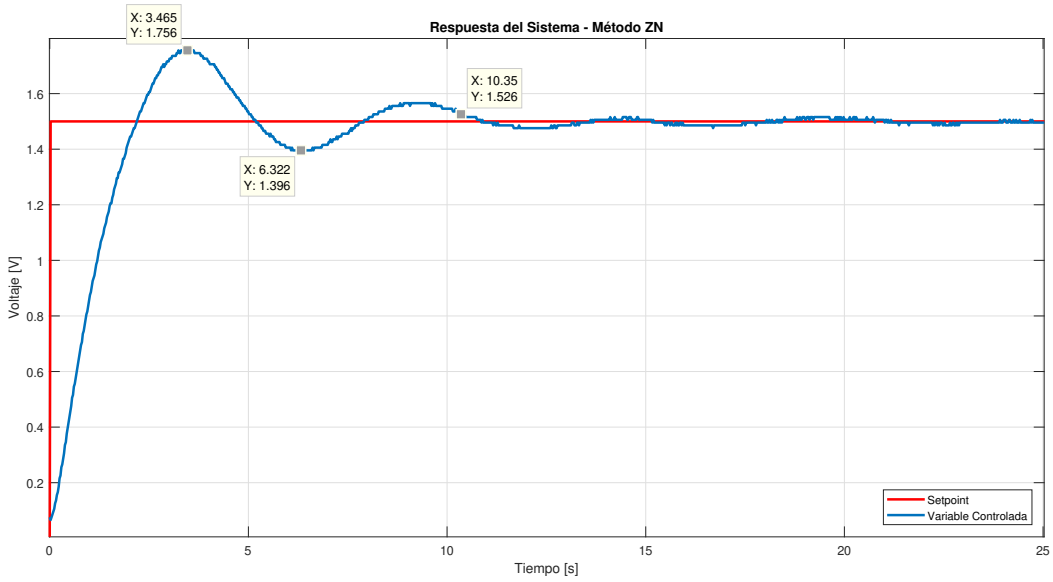


Figura 86. Respuesta del sistema con setpoint de 1,5V - Método ZN

Ante perturbaciones, el sistema tuvo un tiempo de recuperación promedio de aproximadamente 72,5$[ms]$ como se muestra en la figura 87. En la primera perturbación, alcanza un valor máximo de 1,52$[V]$ (elevación de un 1,33 % del setpoint) y en un tiempo de 63$[ms]$ logró la corrección de la variable controlada. En la segunda perturbación, la variable controlada disminuyó en un 4,67 %, y en un tiempo de 82$[ms]$ regresó a su estado estable con criterio de 2 % de error.

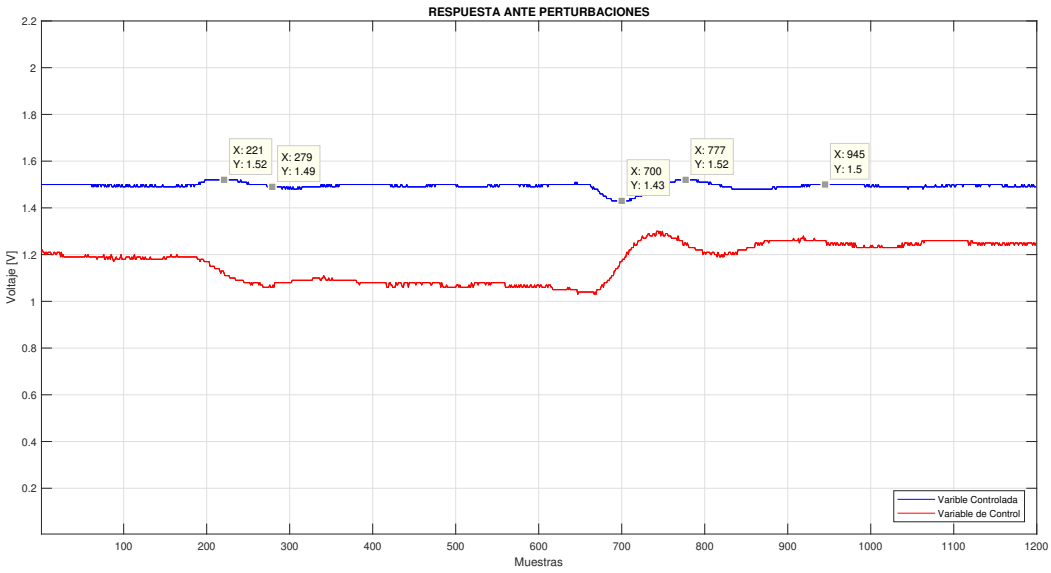


Figura 87. Respuesta ante perturbaciones - Método ZN

Con el uso del algoritmo GA para la sintonización del controlador, la salida del sistema (fig.88), no produjo ningún sobreimpulso, debido a que la ganancia Kd es baja. En cuanto al tiempo de estabilización, la señal alcanzó en 16,67$[s]$ igualar a la señal de referencia; a partir de ahí, esta señal al igual que en el caso anterior, presenta ligeras oscilaciones dentro del criterio de error del 2 %.

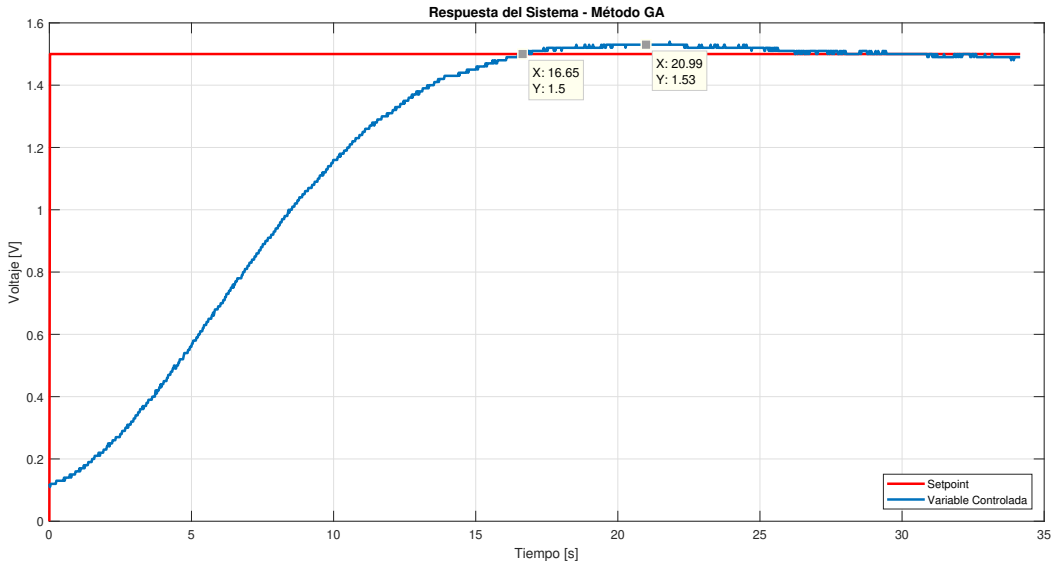


Figura 88. Respuesta del sistema con setpoint de 1,5V - Método GA

Y con la introducción de perturbaciones, el sistema controlado tardó 236,5$[ms]$ promedios, en recuperar el estado estable ante perturbaciones como se muestra en la figura 89. La introducción de la primera perturbación (fig.90), elevó la señal de salida en un 4 % y en un tiempo de 223$[ms]$ alcanzó el estado estable con un criterio del 2 % del error. Con la segunda perturbación (fig.91), la señal de salida disminuyó en un 7,33 % y tardó 250$[ms]$ en recuperar el estado estable.

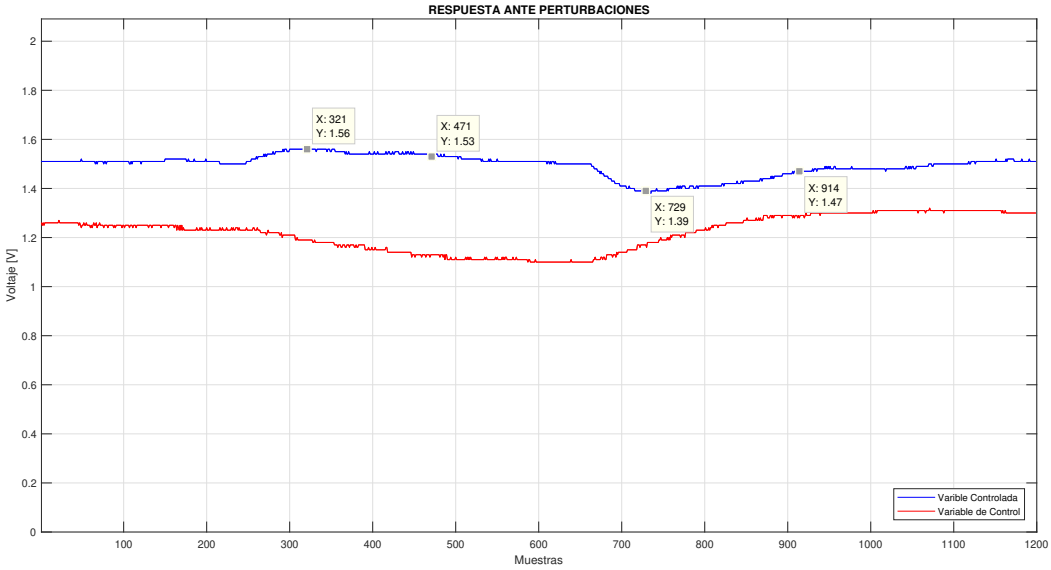


Figura 89. Respuesta ante perturbaciones - Método GA

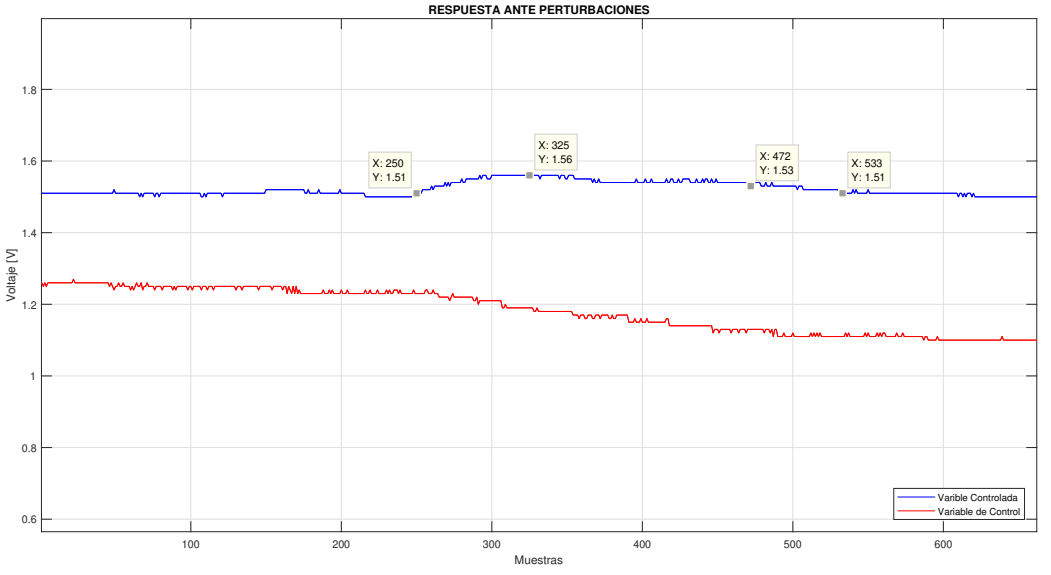


Figura 90. Respuesta ante primera perturbación - Método GA

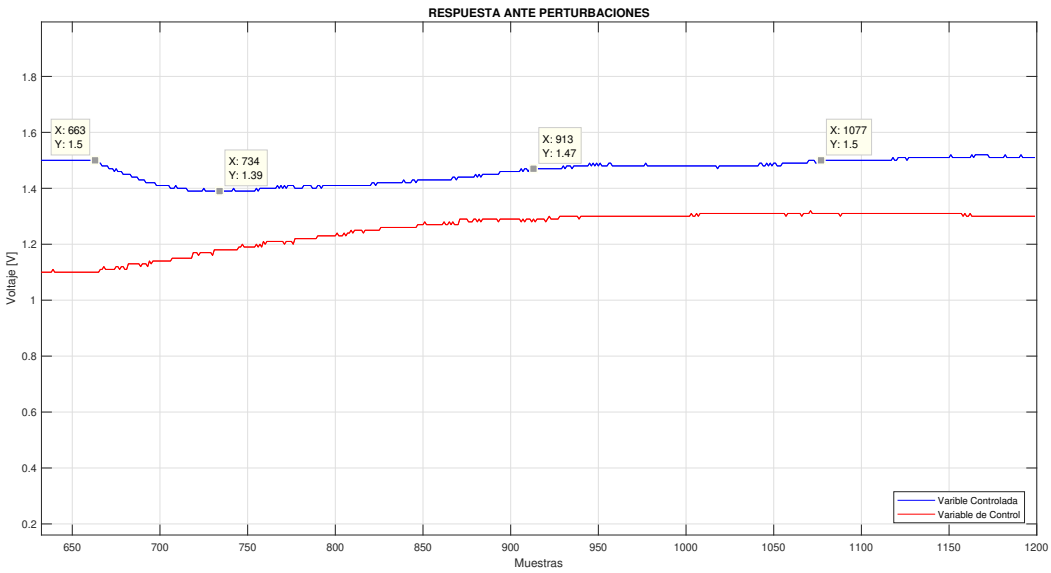


Figura 91. Respuesta ante segunda perturbación - Método GA

En otro caso, en la figura 92, se muestra la respuesta del controlador sintonizado mediante el uso del algoritmo PSO, ante una entrada escalón con un sobreimpulso del 19,07 %, un subimpulso del 2,93 % y un tiempo de estabilización de aproximadamente 16,77$[s]$ %.

Ante las perturbaciones, el sistema alcanzó la recuperación de la referencia en un tiempo promedio de 71$[ms]$ (fig93). La primera perturbación (fig.94), elevó la señal controlada en un 2,67 % y tardó 116$[ms]$ en eliminar el error; sin embargo, en 42$[ms]$ alcanzó su estado estable con un criterio del 2 % de error. La segunda perturbación (fig.95), disminuyó la señal en un 5,33 %, pero tardó 100$[ms]$ en alcanzar el estado estable, y en 135$[ms]$ eliminó el error entre la variable controlada y el setpoint.

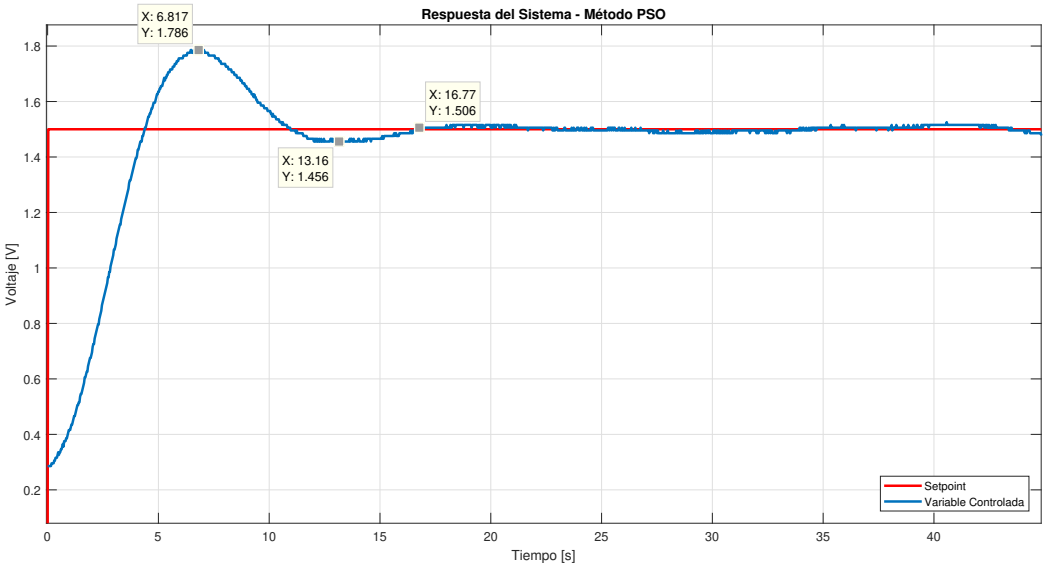


Figura 92. Respuesta del sistema con setpoint de 1,5V - Método PSO

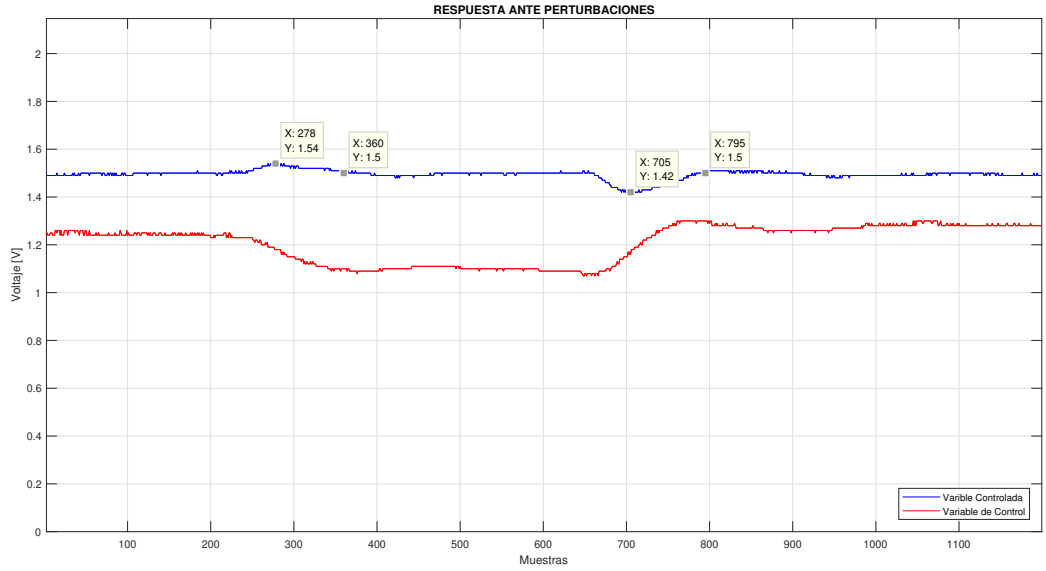


Figura 93. Respuesta ante perturbaciones - Método PSO

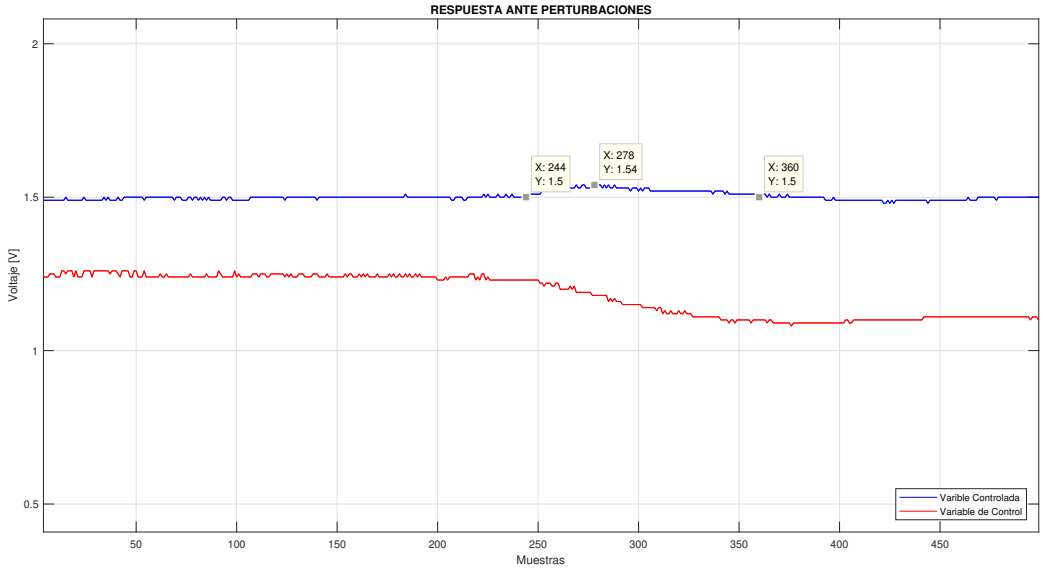


Figura 94. Respuesta ante primera perturbación - Método PSO

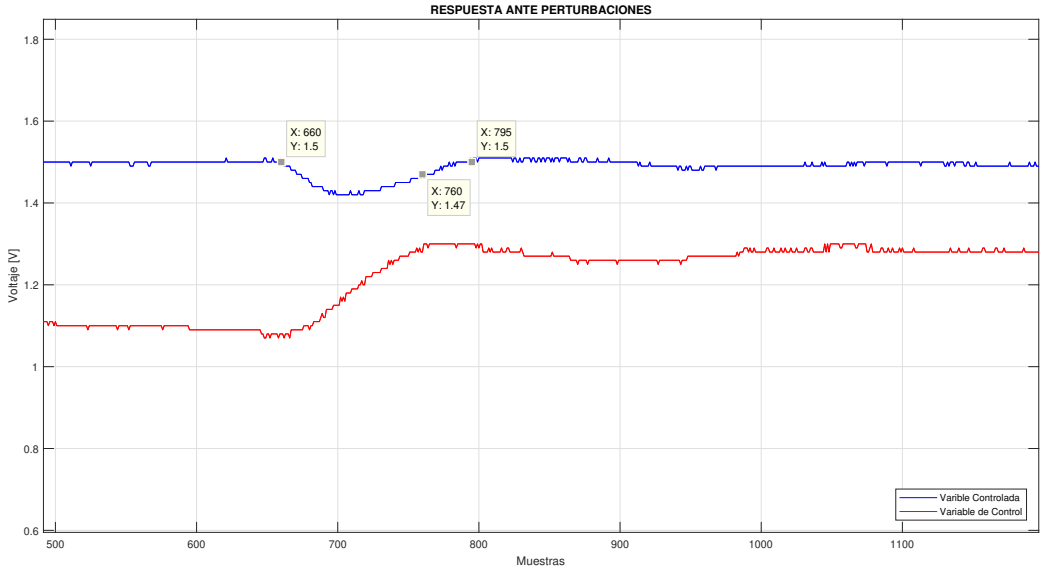


Figura 95. Respuesta ante segunda perturbación - Método PSO

La respuesta del controlador sintonizado mediante el algoritmo ABC (fig.96), produjo una señal críticamente amortiguada; ya que, los valores de las ganancias encontrada produjeron un controlador tipo PI. Como se puede observar en la tabla 21, la ganancia Kd es aproximadamente cero. Esto conllevó a que, el tiempo de estabilización sea de aproximadamente $20[s]$.

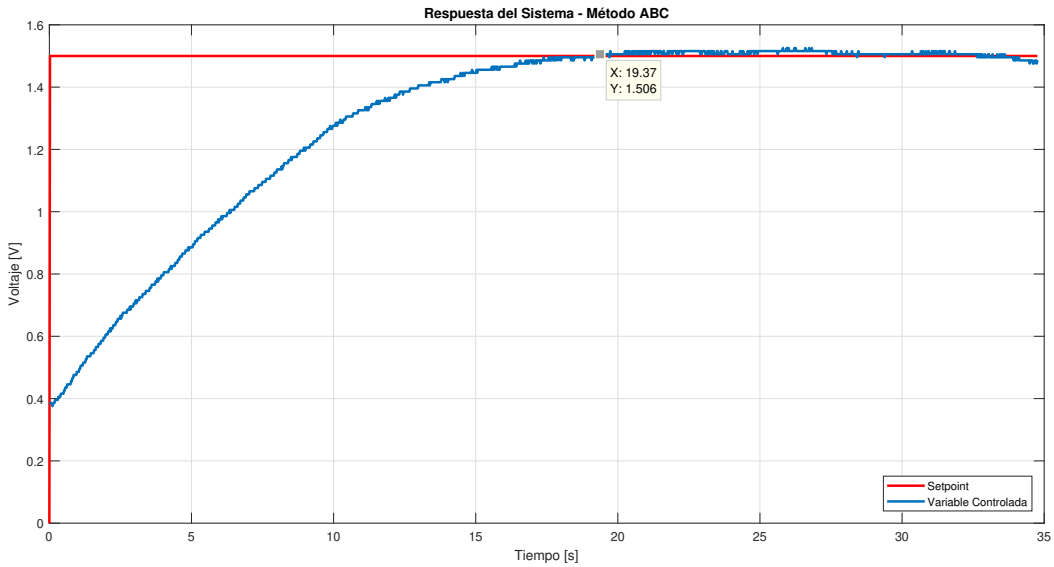


Figura 96. Respuesta del sistema con setpoint de $1{,}5V$ - Método ABC

Mientras que, el sistema con la primera perturbación (fig.97), obtuvo una elevación del $4{,}67\,\%$ de la señal de salida a partir de la señal de referencia; el tiempo de recuperación fue de $155[ms]$ (con el criterio del $2\,\%$ del error), mientras que para eliminar el error tardó $406[ms]$. En cambio, con la segunda perturbación (fig.98), tardó aproximadamente $187[ms]$ en alcanzar estado estable.

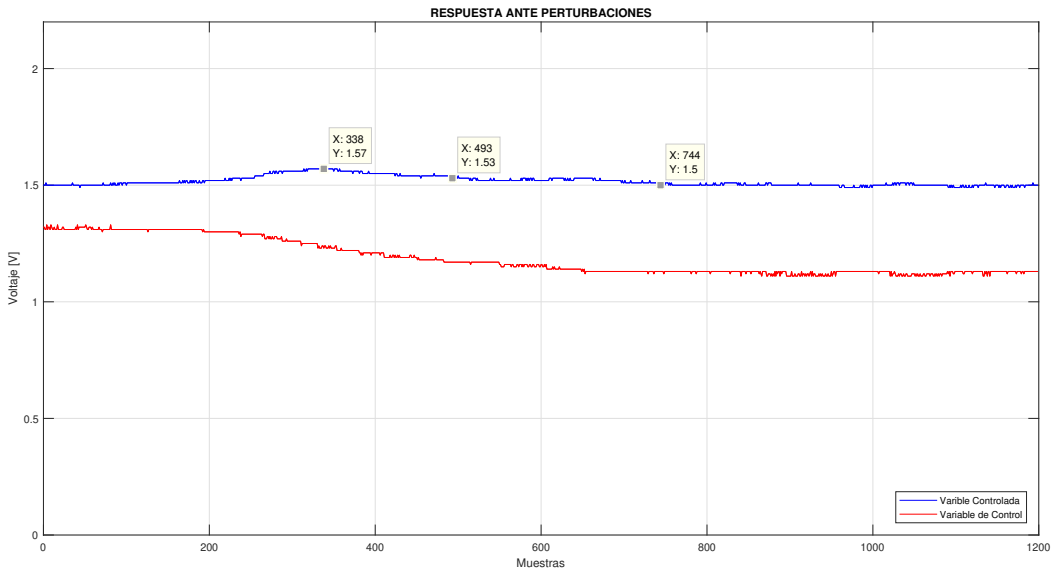


Figura 97. Respuesta ante primera perturbación - Método ABC

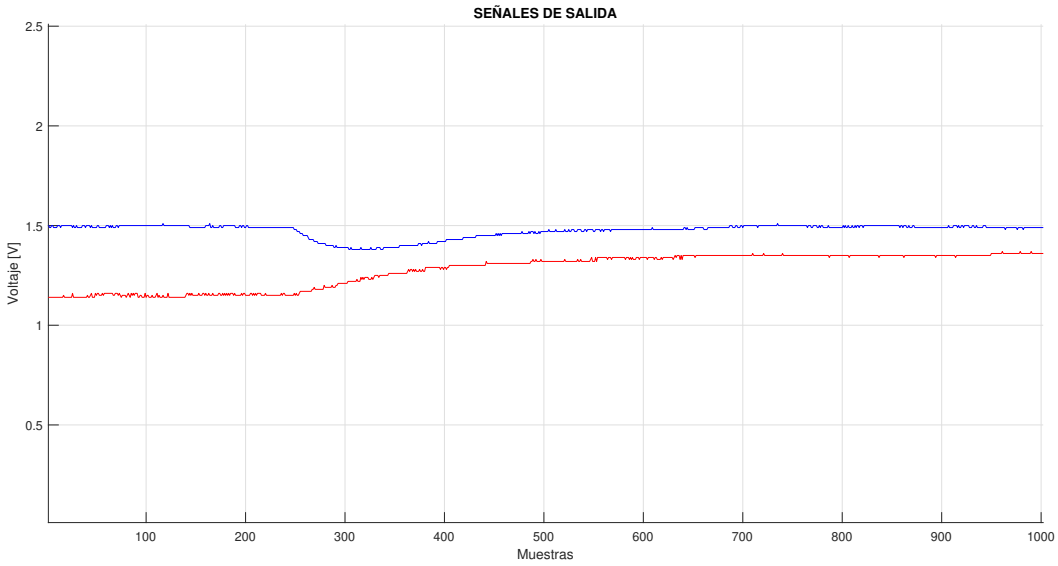


Figura 98. Respuesta ante segunda perturbación - Método ABC

A continuación, se muestra una figura comparativa de las señales medidas (fig.99) y en la tabla 28 se presenta los valores característicos de las señales medidas. Seguidamente, en la tabla 29, se resume los tiempos que tardó cada uno de los controladores, en responder la introducción de las perturbaciones.

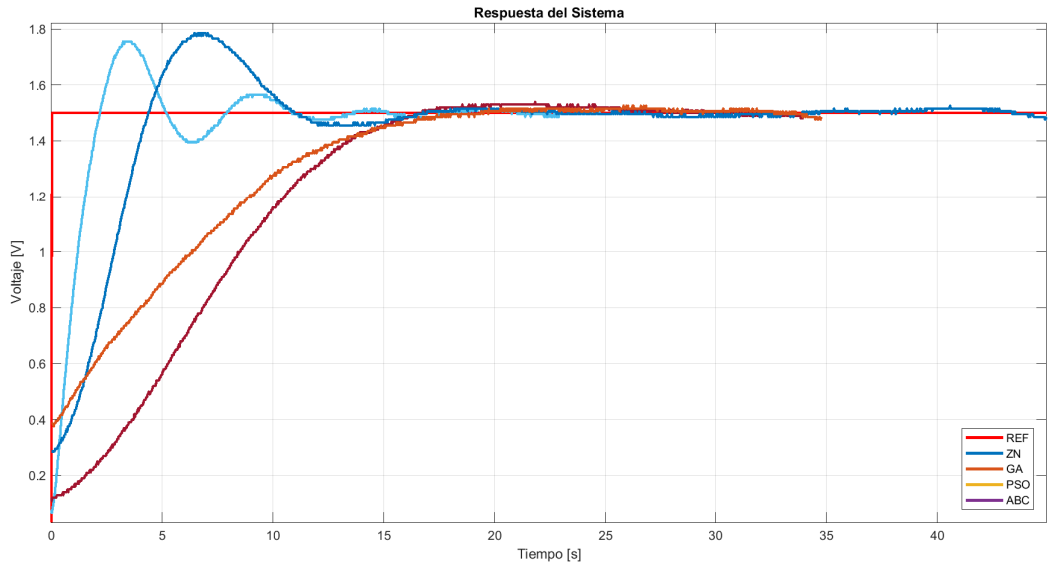


Figura 99. Salidas del sistema real ante una entrada escalón mediante el uso de los controladores diseñados por algoritmos bio-inspirados.

Tabla 28
Comparación de las respuestas medidas.

Controlador		**Valores característicos**	
Método	*Valor Máximo*	*Sobreimpulso*	*Tiempo de Estabilización*
ZN	1,870*V*	24,667 %	15,00*s*
GA	–	8,533 %	15,00*s*
PSO	1,780*V*	18,667 %	13,20*s*
ABC	–	–	13,40*s*

Tabla 29
Tiempo de recuperación ante perturbaciones reales.

Método de diseño del	**Tiempo de recuperación**	
Control PID	**Primera Perturbación**	**Segunda Perturbación**
Método ZN	63,0*ms*	82,0*ms*
Algoritmo GA	223*ms*	250*ms*
Algoritmo PSO	42*ms*	100*ms*
Algoritmo ABC	155*ms*	187*ms*

4.4. Discusión de Resultados

Durante la ejecución de los algoritmos bio-inpirados, con el fin de hallar las ganancias más óptimas para el controlador tipo PID, se realizó la evaluación de costo y desempeño de las posibles soluciones. Las soluciones finales, de acuerdo a cada método, alcanzaron los costos y desempeños representados en la figura 100 y 101 respectivamente.

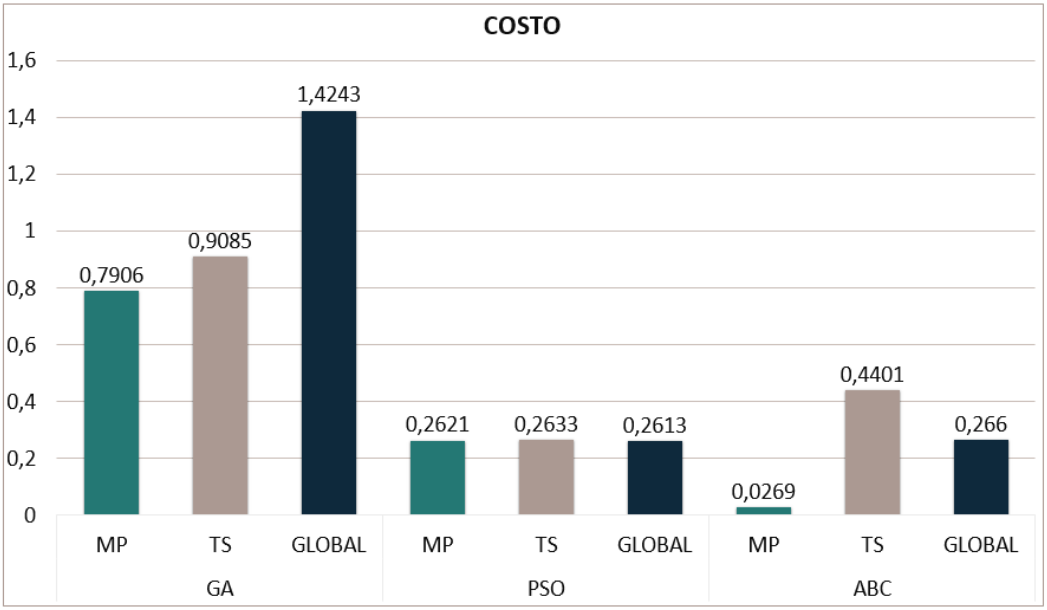


Figura 100. Análisis comparativo del costo alcanzado.

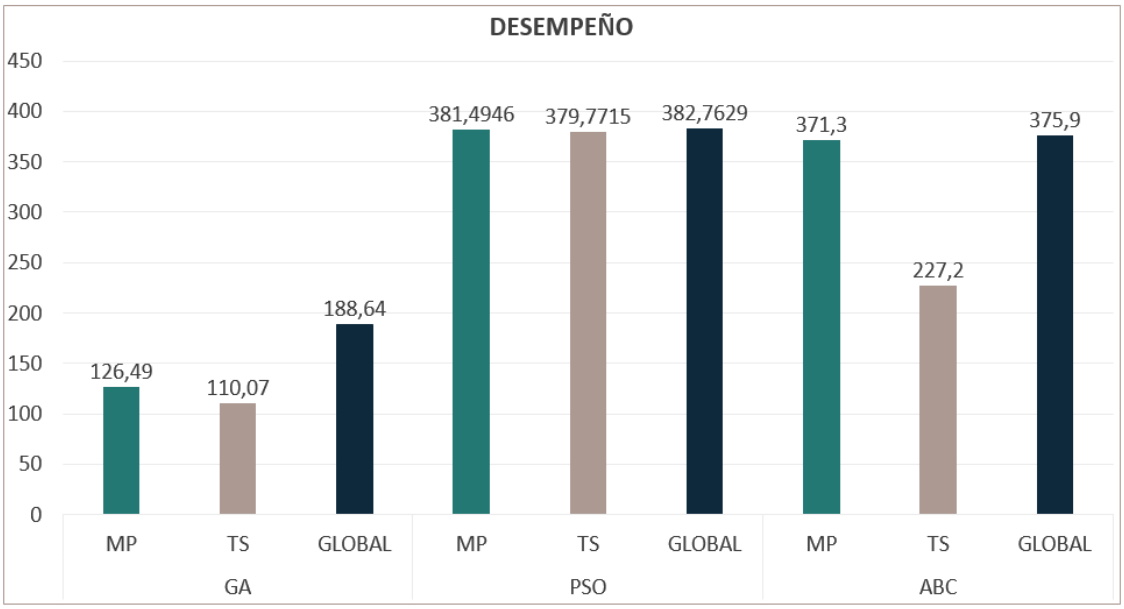


Figura 101. Análisis comparativo del desempeño alcanzado.

Como se puede observar, el método con el mínimo costo generado, en base al criterio global, fue el algoritmo PSO,y este a la vez, el algoritmo con el mayor desempeño alcanzado frente al algoritmo ABC y GA. Por el contrario, el uso de algoritmo GA, generó el mayor costo y menor desempeño frente a los algoritmos PSO y ABC.

En lo concerniente a la implementación de los valores encontrados, todos los controladores presentaron respuestas aceptables dentro de los requerimientos de control planteados. No obstante, en la tabla 26 se verificó en las pruebas simuladas que, el controlador sintonizado mediante el uso del algoritmo ABC alcanzó los mínimos valores característicos en la señal de salida (fig.102).

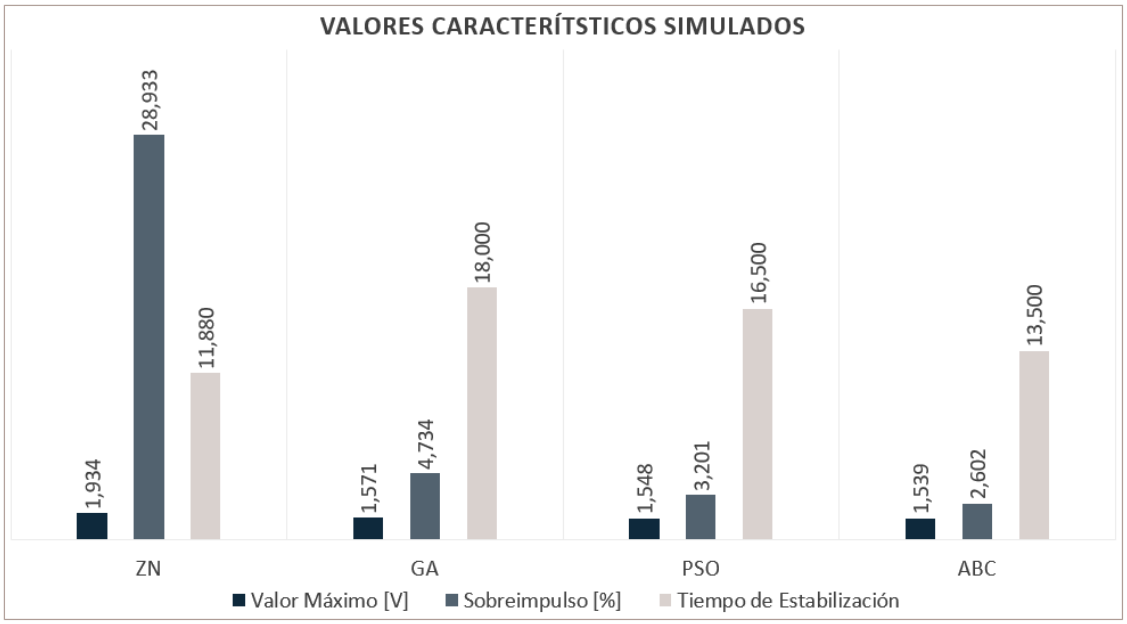


Figura 102. Análisis comparativo de las respuestas simuladas.

En cambio, los valores medidos a la salida de la planta real, se presentan gráficamente en la figura 103 (correspondiente a la tabla 28). Con estos datos, se verifica que el algoritmo PSO obtuvo el mejor tiempo de estabilización, y un sobreimpulso dentro del rango establecido pero mayor a comparación de los algoritmos GA y ABC; estos dos últimos, generó una respuesta críticamente amortiguada ($Mp = 0\,\%$); ya que, las ganancias generadas para la parte derivativa tendían a cero.

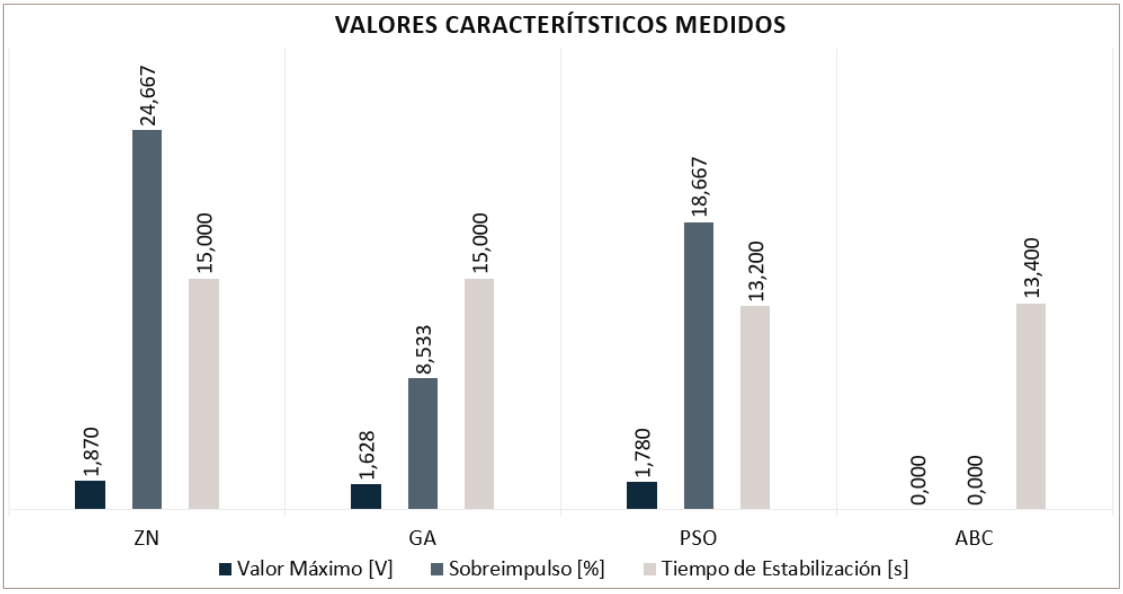


Figura 103. Análisis comparativo de las respuestas medidas.

En cuanto al rechazo a las perturbaciones, entre los algoritmos bio-inspirados, el uso del algoritmo PSO generó el controlador con mayor rapidez de respuesta para la corrección de las perturbaciones introducidas, como se puede apreciar en la figura 104. Tanto GA como ABC, presentan una respuesta más lenta por sus ganancias bajas en la parte integral y derivativa.

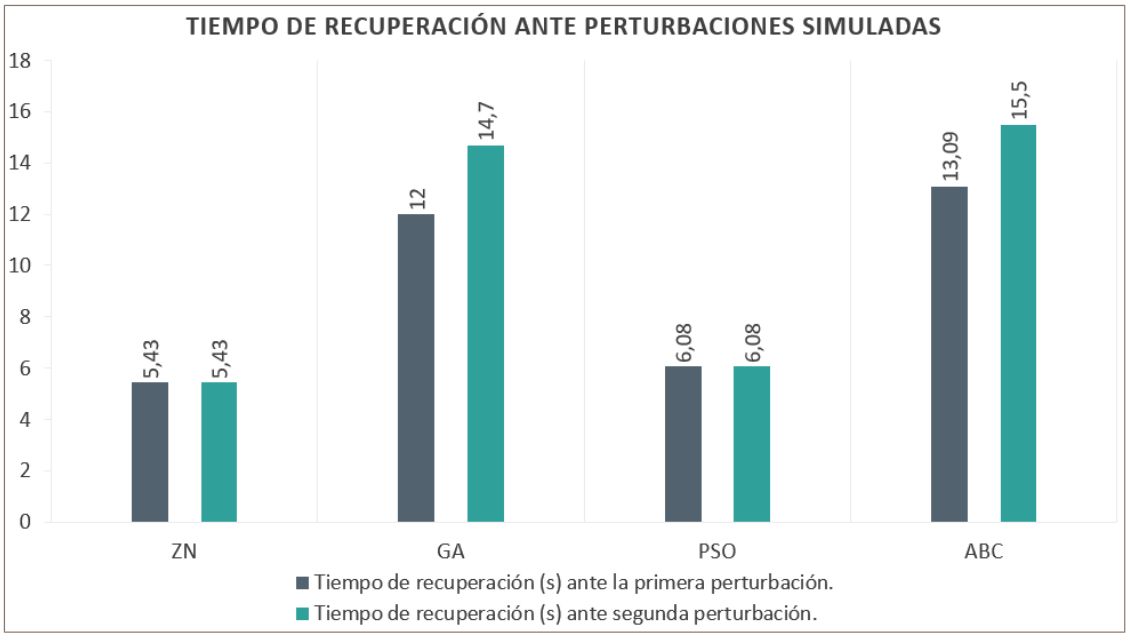


Figura 104. Análisis comparativo de los tiempos de respuesta simulados ante perturbaciones.

En definitiva, considerando los algoritmos bio-inspirados GA, PSO y ABC, la mejor respuesta del sistema se tuvo en base al controlador diseñado mediante el algoritmo PSO, seguido del controlador diseña mediante ABC y finalmente el controlador diseñado mediante GA. Adicionalmente, hay que considera que con PSO se logró una acción más rápida para lograr un rechazo a perturbaciones.

Capítulo 5

Conclusiones y Recomendaciones

Finalmente, en este capítulo se dan a conocer las conclusiones más relevantes del presente proyecto investigativo. Y en base a la experiencia obtenida durante el proceso de desarrollo del mismo, se presenta algunas recomendaciones que servirán de contribución para proyectos futuros.

5.1. Conclusiones

Para el desarrollo de la sintonización de controladores tipo PID mediante el uso de los algoritmos de optimización GA, PSO y ABC, se definieron tres criterios objetivos para usar dichos algoritmos como herramientas específicas de búsqueda y optimización de esta aplicación. Los criterios empleados fueron porcentaje de sobreimpulso, tiempo de estabilización y porcentaje de error en estado estable; cada uno junto a un peso ponderado del 50 %.

La incorporación de estos criterios en la construcción de los códigos, se realizó en forma de costo y desempeño; permitiendo así, a través de su cuantificación, la discriminación de las posibles mejores soluciones, según los objetivos de control planteados.

La puesta en marcha de los métodos de sintonización desarrollados, arrogó resultados prometedores; los cuales, tras ser implementados, confirmaron la efectividad de los controladores como parte del sistema de flujo de aire caliente, módulo PCT-2.

Se determinó la diferencia de desempeño de cada uno de los algoritmos bio-inspirados descritos en el presente trabajo, a través de un análisis comparativo de resultado. Sin embargo, no se realizó una comparación entre el desempeño progresivo (en cada ciclo de ejecución) que alcanza cada uno de los algoritmos bio-inspirados implementados, puesto que, los procesos que comprenden cada ciclo difieren notoriamente uno del otro. Especialmente en el algoritmo ABC, en el cual, un ciclo de ejecución está comprendido por tres exploraciones de todo el espacio de búsqueda (realizadas por los grupos de abejas empleada, observadoras y exploradoras); en otras palabras, en cada ciclo de ejecución se realiza tres veces la evaluación de aptitud de todo el conjunto de posibles soluciones que, a diferencia del algoritmo GA y PSO, solo se realiza una sólo evaluación de aptitud del conjunto de posibles soluciones (población, enjambre y colonia respectivamente). Pero, para una comparación correcta del desempeño final alcanzado por cada uno de los tres algoritmos bio-inspirados, se consideró en número de evaluaciones total que se realiza durante ejecución toral de los algoritmos GA, PSO y ABC, como lo sugiere Mernik et al. (2015).

De análisis de resultados mencionado, se determinó que tomando en cuanto el tiempo de estabilización del sistema simulado y obtenido mediante la aplicación del algoritmo GA en la sintonización del controlador, se pudo mejorar en un 32,38 % con la aplicación del algoritmo PSO; mientras que, con la aplicación del algoritmo ABC se mejoró en un 45,04 %. Asimismo, el sobreimpulso, partiendo de los resultados obtenido mediante la aplicación de GA, se pudo mejor en un 8,33 % con el algoritmo PSO y un 25 % con el algoritmo ABC.

En comparación entre el algoritmo PSO y ABC, la aplicación del algoritmo de ABC mejora la respuesta del algoritmo PSO en un 18,71 % en cuanto al tiempo de estabilización, y en un 18,18 % en cuanto a sobreimpulso.

De los datos medidos a las señales de salida reales de la planta, con el algoritmo PSO se alcanzó un tiempo de estabilización menor al 12 % de lo conseguido con el algoritmo GA, y también menor al 1,51 % del tiempo generado con la aplicación del algoritmo ABC. Pero, en cuanto al sobreimpulso generado en régimen transitorio, el algoritmo PSO generó mayor un valor superior al 100 % en comparación a la respuesta del algoritmo GA. En este caso, el algoritmo ABC no generó ningún sobreimpulso en la salida del sistema.

5.2. Recomendaciones

Para garantizar la convergencia de la búsqueda de los algoritmos, se debe fijar correctamente los parámetros de ajuste del del algoritmo, dependiendo de los requerimientos de problema. Por lo cual, se debe conocer la naturaleza el problema a tratarse para aplicar de forma correcta los logaritmos de búsqueda.

Además, en la aplicación del algoritmo bio-inspirado evolutivo GA y los algoritmos basados en inteligencia de enjambre PSO y ABC, no se recomienda hacer una comparación referente al tiempo de ejecución; puesto que cada uno de los algoritmos cuentan con un conjunto de diferentes operaciones para lograr la exploración de soluciones dentro del espacio de búsqueda; como es el caso del algoritmo ABC que alcanza su búsqueda en un tiempo de procesamiento (minutos) mayor al que requerido por el algoritmo PSO o GA (segundos).

Por último, para las futuras investigaciones aplicativas que requieran el uso del módulo PCT-2, se recomienda considera las variables externas que modifiquen el funcionamiento del control de flujo de aire caliente de dicho módulo; especialmente la temperatura ambiente a la que se trabaje y el tiempo de funcionamiento de la misma.

Referencias

Abazari, A., Monsef, H., y Wu, B. (2018). Load frequency control by de-loaded wind farm using the optimal fuzzy-based PID droop controller. *IET Renewable Power Generation*, *13*(1), 180–190. Descargado de https://digital-library.theiet.org/content/journals/10.1049/iet-rpg.2018.5392 doi: 10.1049/iet-rpg.2018.5392

Abdel-Basset, M., y Shawky, L. A. (2018). Flower pollination algorithm: a comprehensive review. *Artificial Intelligence Review*, 1–25. Descargado de https://doi.org/10.1007/s10462-018-9624-4 doi: 10.1007/s10462-018-9624-4

Agarwal, P., y Mehta, S. (2014). Nature-Inspired Algorithms: State-of-Art, Problems and Prospects. *International Journal of Computer Applications*, *100*(14), 14–21. Descargado de http://research.ijcaonline.org/volume100/number14/pxc3898331.pdf doi: 10.5120/17593-8331

Ahmed, H., y Glasgow, J. (2012). Swarm Intelligence : Concepts , Models and Applications. *School of Computing Queen's University Kingston, Ontario, Canada*, *585*(February), 1–50. doi: 10.13140/2.1.1320.2568

Akay, B., y Karaboga, D. (2012, jun). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, *192*, 120–142. Descargado de http://dx.doi.org/10.1016/j.ins.2010.07.015 doi: 10.1016/j.ins.2010.07.015

Aksoy, A., Aslan, S., y Karaboga, D. (2018). Order based emigrant creation strategy for parallel artificial bee colony algorithm. En *International conference on engineering technologies (icente'17)* (Vol. 2018, pp. 59–75).

Alswaitti, M., Albughdadi, M., y Isa, N. A. M. (2018). Density-based particle swarm optimization algorithm for data clustering. *Expert Systems with Applications*, *91*, 170–186.

AminShokravi, A., Eskandar, H., Derakhsh, A. M., Rad, H. N., y Ghanadi, A. (2018). The potential application of particle swarm optimization algorithm for forecasting the air-overpressure induced by mine blasting. *Engineering with Computers*, *34*(2), 277–285.

Arana, N., López, C., y Alanis, A. (2018). *Bio-inspired Algorithms for Engineering*. Elsevier Science. Descargado de https://books.google.com.ec/books?id=bB8wDwAAQBAJ

Arian, S. A. (2010). Control PID, sintonización. , 1–13. Descargado de http://www.arian.cl/downloads/nt-013.pdf

Askarzadeh, A. (2018). A Memory-Based Genetic Algorithm for Optimization of Power Generation in a Microgrid. *IEEE Transactions on Sustainable Energy*, *9*(3), 1081–1089. doi: 10.1109/TSTE.2017.2765483

Aslan, S., Badem, H., y Karaboga, D. (2019). Improved quick artificial bee colony (iqabc) algorithm for global optimization. *Soft Computing*, 1–22.

Back, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press. Descargado de https://books.google.com.ec/books?id=htJHI1UrL7IC

Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., ... Zdravkovic, V. (2008, jan). Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the National Academy of Sciences*, *105*(4), 1232–1237. Descargado de http://www.pnas.org/cgi/doi/10.1073/pnas.0711437105 doi: 10.1073/pnas.0711437105

Binitha, S., y S Siva, S. (2012, may). A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSCE)*, *2*(2), 137–151. Descargado de https://pdfs.semanticscholar.org/4561/8bef1e4be5cfd2c999d42398166621fd7649.pdf

Bitam, S., Batouche, M., y Talbi, E.-g. (2010, apr). A survey on bee colony algorithms. En *2010 ieee international symposium on parallel & distributed processing, workshops and phd forum (ipdpsw)* (pp. 1–8). Atlanta: IEEE. Descargado de http://ieeexplore.ieee.org/document/5470701/ doi: 10.1109/IPDPSW.2010.5470701

Caceres, C., Rosario, J. M., y Amaya, D. (2017, July). Approach of kinematic control for a nonholonomic wheeled robot using artificial neural networks and genetic algorithms. En *2017 international conference and workshop on bioinspired intelligence (iwobi)* (p. 1-6). doi: 10 .1109/IWOBI.2017.7985533

César R. López M. (2014). Evaluación de desempeño de dos técnicas de optimización bio-inspiradas: Algoritmos Genéticos y Enjambre de Partículas. *Tekhnê*, *11*(1), 49–58.

Chen, J., Yu, W., Tian, J., Chen, L., y Zhou, Z. (2018, feb). Image contrast enhancement using an artificial bee colony algorithm. *Swarm and Evolutionary Computation*, *38*(September), 287–294. Descargado de http://dx.doi.org/10.1016/j.swevo.2017.09.002 doi: 10.1016/j.swevo.2017.09.002

Chong, A., Lam, K. P., Pozzi, M., y Yang, J. (2017). Bayesian calibration of building energy models with large datasets. *Energy and Buildings*, *154*, 343–355.

Darwish, A. (2018). Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. *Future Computing and Informatics Journal*, *3*(2), 231–246. Descargado de https://doi.org/10.1016/j.fcij.2018.06.001 doi: 10.1016/j.fcij.2018.06.001

Dehkordi, M. N., y Amiri, E. (2018). Dynamic data clustering by combining improved discrete artificial bee colony algorithm with fuzzy logic. *International Journal of Bio-Inspired Computation*, *12*(3), 164. doi: 10.1504/ijbic.2018.10015870

Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., y Harley, R. G. (2008, apr). Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. *IEEE Transactions on Evolutionary Computation*, *12*(2), 171–195. Descargado de http://ieeexplore.ieee.org/document/4358769/ doi: 10.1109/TEVC.2007.896686

Eberhart, y Yuhui Shi. (2002). Particle swarm optimization: developments, applications and resources. , 81–86. doi: 10.1109/cec.2001.934374

Eberhart, R., y Kennedy, J. (2002). A new optimizer using particle swarm theory. En *Mhs'95. proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43). IEEE. Descargado de http://ieeexplore.ieee.org/document/494215/ doi: 10.1109/MHS.1995.494215

Fallah, S. N., Deo, R. C., Shojafar, M., Conti, M., y Shamshirband, S. (2018). Computational intelligence approaches for energy load forecasting in smart energy management grids: State of the art, future challenges, and research directions. *Energies*, *11*(3), 1–31. doi: 10.3390/en11030596

Fetouh, T., y Zaky, M. S. (2017, January). New approach to design svc-based stabiliser using genetic algorithm and rough set theory. *IET Generation, Transmission & Distribution*, *11*(2), 372–382. Descargado de https://digital-library.theiet.org/content/journals/10.1049/iet-gtd.2016.0701

Fister, I., Yang, X. S., Brest, J., y Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *Elektrotehniski Vestnik/Electrotechnical Review*, *80*(3), 116–122.

Gong, D., Han, Y., y Sun, J. (2018). A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems. *Knowledge-Based Systems*, *148*, 115–130.

Hajihassani, M., Armaghani, D. J., y Kalatehjari, R. (2018). Applications of particle swarm optimization in geotechnical engineering: a comprehensive review. *Geotechnical and Geological Engineering*, *36*(2), 705–722.

Hu, X., y Eberhart, R. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. En *Proceedings of the 2002 congress on evolutionary computation. cec'02 (cat. no.02th8600)* (Vol. 2, pp. 1677–1681 vol.2). doi: 10.1109/CEC.2002.1004494

Hussain, K., Mohd Salleh, M. N., Cheng, S., Shi, Y., y Naseem, R. (2018). Artificial bee colony algorithm: A component-wise analysis using diversity measurement. *Journal of King Saud University - Computer and Information Sciences*. Descargado de https://doi.org/10.1016/j.jksuci.2018.09.017 doi: 10.1016/j.jksuci.2018.09.017

Karaboga, D. (2005). *An Idea Based On Honey Bee Swarm For Numerical Optimization* (Inf. Téc.). Erciyes University.

Karaboga, D., Gorkemli, B., Ozturk, C., y Karaboga, N. (2014, jun). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57. Descargado de http://link.springer.com/10.1007/s10462-012-9328-0 doi: 10.1007/s10462-012-9328-0

Kennedy, J. (2006). Swarm Intelligence. En M. Springer, Boston (Ed.), *Handbook of nature-inspired and innovative computing* (pp. 187–219). Boston: Kluwer Academic Publishers. Descargado de http://link.springer.com/10.1007/0-387-27705-6{_}6 doi: 10.1007/0-387-27705-6_6

Kennedy, J. (2010). Particle Swarm Optimization. En C. Sammut y G. I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 760–766). Boston, MA: Springer US. Descargado de https://doi.org/10.1007/978-0-387-30164-8{_}630 doi: 10.1007/978-0-387-30164-8_630

Kirchner, W. H., Lindauer, M., y Michelsen, A. (1988). Honeybee dance communication - Acoustical indication of direction in round dances. *Naturwissenschaften*, *75*(12), 629–630. doi: 10.1007/BF00366482

Kong, D., Chang, T., Dai, W., Wang, Q., y Sun, H. (2018). An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy. *Information Sciences*, *442*, 54–71.

Kulkarni, V., y Desai, V. (2016). ABC and PSO: A comparative analysis. En *Ieee international conference on computational intelligence and computing research.* Chennai. doi: 10.1109/ICCIC.2016.7919625

Kumar, S., Nayyar, A., y Kumari, R. (2019). Arrhenius artificial bee colony algorithm. En *International conference on innovative computing and communications* (pp. 187–195).

Kumari, S., Prince, P., Verma, V. K., Appasani, B., y Ranjan, R. K. (2018, Feb). Ga based design of current conveyor pld controller for the speed control of bldc motor. En *2018 4th international conference on computational intelligence communication technology (cict)* (p. 1-3). doi: 10.1109/CIACT.2018.8480149

Kurdi, H., y How, J. (2016). Bio-Inspired Algorithm for Task Allocation in Multi-UAV Search and Rescue Missions. En *Aiaa guidance, navigation, and control conference* (pp. 1–7). California. Descargado de http://arc.aiaa.org/doi/10.2514/6.2016-1377 doi: 10.2514/6.2016-1377

L. Li, X., J. Shao, Z., y X. Qian, J. I. (2002). *An optimizing method based on autonomous animate: Fish swarm algorithm* (Vol. 22).

Li, T., Shao, G., Zuo, W., y Huang, S. (2017). Genetic Algorithm for Building Optimization. En *9th international conference on machine learning and computing - icmlc 2017* (pp. 205–210). Singapore. Descargado de http://dl.acm.org/citation.cfm?doid=3055635.3056591 doi: 10.1145/3055635.3056591

Lukeman, R., Li, Y.-X., y Edelstein-Keshet, L. (2010, jul). Inferring individual rules from collective behavior. *Proceedings of the National Academy of Sciences*, *107*(28), 12576–12580. Descargado de http://www.pnas.org/lookup/doi/10.1073/pnas.1001763107 doi: 10.1073/pnas.1001763107

Martín-Moreno, R., y Vega-Rodríguez, M. A. (2018). Multi-Objective Artificial Bee Colony algorithm applied to the bi-objective orienteering problem. *Knowledge-Based Systems*, *154*(November 2017), 93–101. Descargado de http://dx.doi.org/10.1016/j.knosys.2018.05.005 doi: 10.1016/j.knosys.2018.05.005

Melanie, M. (1998). An Introduction to Genetic Algorithms. *A Bradford Book The MIT Press*, 158.

Mernik, M., Liu, S. H., Karaboga, D., y Črepinšek, M. (2015). On clarifying misconceptions when comparing variants of the Artificial Bee Colony Algorithm by offering a new implementation. *Information Sciences*, *291*(C), 115–127. doi: 10.1016/j.ins.2014.08.040

Mills, A. J., y Ashton, R. W. (2017, June). Genetic algorithm design of an adaptive, multirate lqr controller for a multi-machine mvdc shipboard electric distribution system with constant power loads. En *2017 ieee transportation electrification conference and expo (itec)* (p. 381-386). doi: 10.1109/ITEC.2017.7993301

Miniguano, H. (2008). *Diseño e implementación de un controlador robusto para el módulo experimental de control de temperatura de un flujo de aire pct-2.* (Tesis Doctoral, Escuela Politécnica del Ejercito). Descargado de http://repositorio.espe.edu.ec/handle/21000/448

Mishra, A. (2017). Nature Inspired Algorithms : A Survey of the State of the Art. *International Journal*, *5*(9), 16–21.

Mishra, B., Dehuri, S., y Wang, G. (2013). A State of the Art Review of Artificial Bee Colony in the Optimization of Single and Multiple Criteria. *International Journal of Applied Metaheuristic Computing*, *4*(4), 23–45. Descargado de http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/ijamc.2013100102 doi: 10.4018/ijamc.2013100102

Mitchell, M. (1998a). Book Review: Handbook of genetic algorithms. *Artificial Intelligence*, *100*(1-2), 325–330. Descargado de http://tocs.ulb.tu-darmstadt.de/28323289.pdf

Mitchell, M. (1998b). Elements of Generic Algorithms. *An Introduction to Generic Algorithms*, 158. Descargado de https://mitpress.mit.edu/books/introduction-genetic-algorithms doi: 10.1016/S0898-1221(96)90227-8

Mousavi-Avval, S. H., Rafiee, S., Sharifi, M., Hosseinpour, S., Notarnicola, B., Tassielli, G., y Renzulli, P. A. (2017). Application of multi-objective genetic algorithms for optimization of energy, economics and environmental life cycle assessment in oilseed production. *Journal of Cleaner Production*, *140*, 804–815. Descargado de http://dx.doi.org/10.1016/j.jclepro.2016.03.075 doi: 10.1016/j.jclepro.2016.03.075

Naghibi, S. A., Ahmadi, K., y Daneshi, A. (2017). Application of support vector machine, random forest, and genetic algorithm optimized random forest models in groundwater potential mapping. *Water Resources Management*, *31*(9), 2761–2775.

Nisi, K., Nagaraj, B., y Agalya, A. (2018). Tuning of a PID controller using evolutionary multi objective optimization methodologies and application to the pulp and paper industry. *International Journal of Machine Learning and Cybernetics*, *0*(0), 0. Descargado de http://link.springer.com/10.1007/s13042-018-0831-8 doi: 10.1007/s13042-018-0831-8

İnkaya, T., Kayalıgil, S., y Özdemirel, N. E. (2016). Swarm Intelligence-Based Clustering Algorithms: A Survey. En *Unsupervised learning algorithms* (pp. 303–341). Cham: Springer

International Publishing. Descargado de http://link.springer.com/10.1007/978-3-319-24211-8{_}12 doi: 10.1007/978-3-319-24211-8_12

O'Dwyer, A. (2009). *Handbook of PI and PID Controller Tuning Rules* (3rd ed., Vol. 106; Imperial College Press, Ed.) (n.º 11).

Ogata, K. (2003). *Ingeniería de control moderna*. Pearson Educación. Descargado de https://books.google.com.ec/books?id=QK148EPC_m0C

Parvin, K., Kit, Y. K., Jern, K. P., Hoque, M., y Hannan, M. (2019, 03). Particle swarm optimization based fuzzy logic mppt inverter controller for grid connected wind turbine. *International Journal of Renewable Energy Research*, *9*(1), 164-174.

Patil, M. B., Naidu, M. N., Vasan, A., y Varma, M. R. (2019). Water distribution system design using multi-objective particle swarm optimisation. *arXiv preprint arXiv:1903.06127*.

Salahshour, E., Malekzadeh, M., Gordillo, F., y Ghasemi, J. (2019). Quantum neural network-based intelligent controller design for cstr using modified particle swarm optimization algorithm. *Transactions of the Institute of Measurement and Control*, *41*(2), 392–404.

Sánchez, V., y Pizarro, D. (2010). Diagnóstico del nivel de automatización en las pequeñas y medianas industrias de la ciudad de Cuenca. *Ingenius*(4), 44–56. Descargado de http://revistas.ups.edu.ec/index.php/ingenius/article/view/4.2010.05 doi: 10.17163/ings.n4.2010.05

Secretaría Nacional de Planificación y Desarrollo - Se, y Nplades. (2017). *Plan Nacional de Desarrollo 2017-2021 "Toda una vida"*. Descargado de http://www.inglaterra.net/economia-de-inglaterra/

Senberber, H., y Bagis, A. (2017). Fractional PID controller design for fractional order systems using ABC algorithm. *Proceedings of the 21st International Conference on Electronics*, *6*, 4–10. doi: 10.1109/ELECTRONICS.2017.7995218

Sengupta, S., Basak, S., y Peters, R. (2018, oct). Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. *Machine Learning and Knowledge Extraction*, *1*(1), 157–191. Descargado de http://arxiv.org/abs/

1804.05319{%}0Ahttp://dx.doi.org/10.3390/make1010010http://www.mdpi.com/2504-4990/1/1/10 doi: 10.3390/make1010010

Shi, Y., y Eberhart, R. (1999). Empirical study of particle swarm optimization. En *Proceedings of the 1999 congress on evolutionary computation-cec99 (cat. no. 99th8406)* (Vol. 3, pp. 1945–1950). IEEE. Descargado de http://ieeexplore.ieee.org/document/785511/ doi: 10.1109/CEC.1999.785511

Shi, Y., y Eberhart, R. (2004). Feature Article Particle Swarm Optimization Feature Article (Cont .). *Neural Networks*(February), 69–73.

Sumpter, D. J. T. (2010). *Collective Animal Behavior*. Princeton University Press. Descargado de https://books.google.com.ec/books?id=JwdOrSMmdkUC

Taetragool, U., Sirinaovakul, B., y Achalakul, T. (2018). Ness: A modified artificial bee colony approach based on nest site selection behavior. *Applied Soft Computing*, *71*, 659–671.

Tian, Y., Huang, L., y Xiong, Y. (2017). A General Technical Route for Parameter Optimization of Ship Motion Controller Based on Artificial Bee Colony Algorithm. *International Journal of Engineering and Technology*, *9*(2), 133–137. Descargado de http://www.ijetch.org/index.php?m=content{&}c=index{&}a=show{&}catid=85{&}id=1127 doi: 10.7763/IJET.2017.V9.958

Whitley, D. C. S. U. (1994). A Genetic Algorithm Tutorial by Darrell Whitley. *Statistics and Computing*(4), 65–85. Descargado de http://samizdat.mines.edu/ga{_}tutorial/

Xiang, W.-l., Meng, X.-l., Li, Y.-z., He, R.-c., y An, M.-q. (2018). An improved artificial bee colony algorithm based on the gravity model. *Information Sciences*, *429*, 49–71.

Yan, F., Wang, Y., Xu, W., y Chen, B. (2018, jun). Time delay control of cable-driven manipulators with artificial bee colony algorithm. *Transactions of the Canadian Society for Mechanical Engineering*, *42*(2), 177–186. Descargado de http://www.nrcresearchpress.com/doi/10.1139/tcsme-2017-0043 doi: 10.1139/tcsme-2017-0043

Yan, G., y Li, C. (2011). An Effective Refinement Artificial Bee Colony Optimization Algorithm Based On Chaotic Search and Application for PID Control Tuning. *Journal of Computational Information Systems*, *7*(9), 3309–3316.

Yang, M., Li, Y., Du, H., Li, C., y He, Z. (2018). Hierarchical multiobjective h-infinity robust control design for wireless power transfer system using genetic algorithm. *IEEE Transactions on Control Systems Technology*(99), 1-9. doi: 10.1109/TCST.2018.2814589

Yang, X.-S. (2005). Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. En (pp. 317–323). Descargado de http://link.springer.com/10.1007/11499305{_}33 doi: 10.1007/11499305_33

Yang, X.-S. (2011). Review of Metaheuristics and Generalized Evolutionary Walk Algorithm. , *3*(2), 77–84. Descargado de http://arxiv.org/abs/1105.3668 doi: 10.1504/IJBIC.2011.039907

Yang, X.-S. (2014). *Nature-Inspired Optimization Algorithms* (Elsevier ed., Vol. 1). London: Elsevier. Descargado de https://linkinghub.elsevier.com/retrieve/pii/C20130013680 doi: 10.1016/C2013-0-01368-0

Yang, X. S., Cui, Z., Xiao, R., Gandomi, A. H., y Karamanoglu, M. (2013). *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*. Elsevier Science. Descargado de https://books.google.com.ec/books?id=J0VcBQxtcwsC

Yang, Y., Zhou, Y., y Gong, Q. (2010). Hybrid Artificial Glowworm Swarm Optimization Algorithm for Solving System of Nonlinear Equations. *Journal of Computational Information Systems*, *6*(10), 3431–3438.

Yurtkuran, A., Yagmahan, B., y Emel, E. (2018). A novel artificial bee colony algorithm for the workforce scheduling and balancing problem in sub-assembly lines with limited buffers. *Applied Soft Computing Journal*, *73*, 767–782. Descargado de https://doi.org/10.1016/j.asoc.2018.09.016 doi: 10.1016/j.asoc.2018.09.016

Zhang, Q., Ogren, R. M., y Kong, S.-C. (2017). Application of Improved Artificial Bee Colony Algorithm to the Parameter Optimization of a Diesel Engine With Pilot Fuel Injections. *Journal of Engineering for Gas Turbines and Power*, *139*(11), 112801. Descargado de http://gasturbinespower.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4036766 doi: 10.1115/1.4036766

Zomaya, S. O., y Y., A. (2006). *Handbook of bioinspired algorithms and applications* (Ilustrada ed., Vol. 5; S. Olariu y A. Zomaya, Eds.). doi: 10.1186/1475-925X-5-47