ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

MÉTODO DE OPTIMIZACIÓN HEURÍSTICA PARA LA OPTIMIZACIÓN DE LA COORDINACIÓN DE PROTECCIONES ELÉCTRICAS DE SOBRE CORRIENTE

MÉTODO DE OPTIMIZACIÓN DE ALGORITMO CUCKOO OPTIMIZATION ALGORITHM (COA) APLICADO A LA COORDINACIÓN DE PROTECCIONES ELÉCTRICAS DE SOBRE CORRIENTE

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO

DANIEL SANTIAGO JIMÉNEZ SOLIZ

daniel.santiagodj@gmail.com

DIRECTOR: MSC. MAURICIO SANTIAGO SORIA COLINA

mauricio.soria@epn.edu.ec

DMQ, febrero 2024

CERTIFICACIONES

Yo, Daniel Santiago Jiménez Soliz declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

Daniel Santiago Jiménez Soliz

Certifico que el presente trabajo de integración curricular fue desarrollado por Daniel Santiago Jiménez Soliz, bajo mi supervisión.

_____

Msc. Mauricio Santiago Soria Colina
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

DANIEL SANTIAGO JIMÉNEZ SOLIZ

MSC. MAURICIO SANTIAGO SORIA COLINA

DEDICATORIA

Este trabajo dedico a Martha Soliz y Edgar Jiménez, mis padres, quienes han respaldado todas mis metas y me han acompañado en las adversidades, poniendo su tiempo y recursos con su mejor predisposición y sin esperar nada a cambio. De manera especial a mi madre que me ha ayudado incondicionalmente y me ha acompañado en los momentos más duros, hasta alcanzar mis sueños. Dedico este trabajo también a mi hermana Samantha Jiménez, a la que amo mucho y me enorgullezco de ser su hermano.

AGRADECIMIENTO

Doy gracias por sobre todas las cosas a Dios que me ha dado la vida, salud e inteligencia necesaria para cumplir esta meta, y por regalarme una hermosa familia que me ha apoyado y alentado en todo instante frente a las adversidades. Agradezco el apoyo de mis padres, mi abuelita Piedad Ibarra que ha sido parte de este logro, también agradezco a mi tío Tito Viscarra que le considero como un padre por darme tan valiosos consejos y ayudándome; a todos mis otros tíos que me aprecian mucho y siempre me han dado lo mejor de ellos.

Agradezco a mis profesores, con los cuales he compartido varias horas de clase, y los cuales he llegado a respetar por su valiosa labor, a más de admirar el trabajo que realizan en busca de formar buenos profesionales. Agradezco a mi tutor de tesis, el Msc. Mauricio Soria por guiarme en toda la realización de este trabajo, a más de haber compartido conmigo con la mejor predisposición otros conocimientos en los salones de clases para otras asignaturas.

ÍNDICE DE CONTENIDO

RESUMEN

En la elaboración del marco teórico se toman los fundamentos de la literatura para adaptarlos a este trabajo de la siguiente manera, inicialmente se presenta una introducción a las protecciones eléctricas de sobre corriente en Sistemas Eléctricos de Potencia. Seguido a esto, se menciona la teoría básica de los relés de sobre corriente y su clasificación. Posteriormente, se presenta el sistema de prueba de 13 nodos de la IEEE modelado en PowerFactory. Se continúa con la revisión bibliográfica del Algoritmo de Optimización Cucú (COA). Finalmente se presenta la función objetivo seleccionada para obtener la coordinación óptima entre relés 51.

En la metodología y análisis de los resultados se emplean los siguientes procedimientos: en primera instancia se incorporan protecciones de sobre corriente 50/51 al sistema de prueba y se presenta una propuesta de ajuste mediante criterios tradicionales a dichas protecciones. Después, se codifica el algoritmo COA en Python. Seguido a esto, se plantea el problema de coordinación óptima de relés 51. A continuación, se codifica en Python la función objetivo utilizada en el algoritmo COA y se consiguen resultados óptimos preliminares.

Finalmente se vincula el algoritmo COA (Python) con el sistema de prueba (PowerFactory) para mejorar la búsqueda de resultados óptimos, automatizar los cálculos y realizar el barrido de fallas. En resultados se analiza la mejora al implementar la optimización, revisando la disminución de tiempos de operación y coordinación entre relés. Se determina que el algoritmo COA consigue resolver este problema de optimización favorablemente.

PALABRAS CLAVE: Relé de sobre corriente, Algoritmo de Optimización Cucú, COA, barrido de fallas, Python, PowerFactory.

ABSTRACT

In developing the theoretical framework, this work takes the fundamentals of the literature to adapt them in the following way. Initially, an introduction to electrical overcurrent protections in Electrical Power Systems is presented. Following this, the document delves into the overcurrent relays and their classification. Subsequently, details about the IEEE 13-node test system modeled in PowerFactory are provided. The literature review of the Cuckoo Optimization Algorithm (COA) continues. Finally, it presents the selected objective function to obtain the optimal coordination between relays 51.

The following procedures are employed for the methodology and analysis of the results. In the first instance, 50/51 overcurrent protections are incorporated into the test system, and a proposal for adjustment using traditional criteria to these protections is presented. Then, the COA algorithm is coded into Python. Following this, it raises the problem of optimal coordination of overcurrent relays (51). The objective function used in the COA algorithm is then coded in Python, and preliminary optimal results are obtained.

Finally, the COA algorithm (Python) is linked to the test system (PowerFactory) to improve the search for optimal results, automate calculations, and perform fault sweeping. The results analyze the improvement when implementing the optimization, reviewing the reduction of operation times and coordination between relays. It is determined that the COA algorithm successfully resolves this optimization problem favorably.

KEYWORDS: Overcurrent relay, Cuckoo Optimization Algorithm, COA, fault sweeping, Python, PowerFactory.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El ajuste adecuado de las protecciones eléctricas es de gran importancia, porque al operar de forma coordinada y segura ante fallas se mantiene el suministro de energía en partes del sistema no afectado, se mantiene la calidad del servicio, se aumenta la confiabilidad y posibilita una adecuada estabilidad del sistema al producirse una perturbación. Una práctica común para ajustar protecciones eléctricas es calibrar los parámetros usando criterios proporcionados por la literatura, y por la experticia alcanzada de la operación del sistema eléctrico y el conocimiento de los elementos que se encuentran en campo.

Los relés deben actuar de forma coordinada para despejar únicamente el o los elementos en falla lo más rápido posible. El elemento en falla debe salir de servicio por la operación de sus protecciones, las cuales deben operar de forma coordinada una después de otra, dejando pasar un tiempo adecuado para actuar. Los tiempos que toman los relés de protección para actuar ante una falla deben ser mínimos, dado que los elementos en falla soportan elevadas corrientes durante todo este tiempo. Mientras mayor sea el tiempo de falla en el elemento, aumenta el riesgo para el personal que opera el sistema, además, mayores son los daños en el equipo por la circulación de corrientes elevadas.

Este trabajo busca fomentar la coordinación óptima de relés de sobre corriente temporizado (51) en los sistemas eléctricos del Ecuador para mejoren la confiabilidad, mantener la calidad y posibilitar una aceptable estabilidad del sistema ante perturbaciones. En Ecuador se realiza el ajuste de protecciones comúnmente con criterios tradicionales, en donde, las protecciones de sobre corriente son muy usadas debido a su bajo costo. Se implementa el Algoritmo de Optimización Cucú (Cuckoo Optimization Algorithm – COA) como método de optimización para resolver el problema de coordinación óptima entre relés 51.

Muchos de los ajustes de parámetros que posibilitan la coordinación óptima entre relés 51 no suelen hallarse de forma explícita mediante la utilización de criterios tradicionales obtenidos por la literatura. La optimización contempla un mayor número de alternativas para ajustar los relés 51, buscando la coordinación óptima entre relés. La metodología propuesta vincula el algoritmo de optimización (Python) y el sistema eléctrico (PowerFactory) con el fin de obtener los parámetros que ajustan los relés 51 para conseguir la coordinación óptima entre relés. El sistema de prueba seleccionado para coordinar los relés 51 es el de 13 nodos de la IEEE.

En el sistema de prueba se implementan relés de sobre corriente, los relés 51 son ajustados con el criterio tradicional y con el procedimiento de la optimización. Para determinar las mejoras obtenidas al ajustar los relés 51 con los parámetros óptimos, se realiza un barrido de fallas. Los tiempos de operación de cada relé 51 para cada punto de falla son comparados cuando son ajustados con el procedimiento de optimización con relación a cuando son ajustados mediante el criterio tradicional. Los relés deben operar de forma coordinada ante fallas de cortocircuito, cumpliendo con un Intervalo de Tiempo de Operación (CTI) de alrededor de 300 [ms] entre la operación de relés.

1.1 Objetivo general

Utilizando el Algoritmo de Optimización Cucú (Cuckoo Optimization Algorithm - COA) conseguir los valores de ajuste óptimo para los relés de sobre corriente, proporcionando una coordinación óptima de las protecciones implementadas en un sistema eléctrico propuesto en la literatura a través de simulación.

1.2 Objetivos específicos

1. Revisión de la bibliografía disponible y más relevante para la elaboración del estado del arte sobre el Algoritmo de Optimización Cucú implementado en la coordinación de protecciones.
2. Estructurar y codificar el Algoritmo de Optimización Cucú en el lenguaje de Python.
3. Implementar una función objetivo para la aplicación del Algoritmo de Optimización Cucú y definir un diagrama de flujo de la metodología que se pretende emplear.
4. Programar en DlgSILENT PowerFactory a través de Python la metodología propuesta y aplicarla a un sistema eléctrico propuesto en la literatura.
5. Analizar los resultados obtenidos, planteamiento de las conclusiones y recomendaciones y propuesta de posibles trabajos futuros.

1.3 Alcance

Se desarrollará la revisión de referencia bibliográfica con relación al Algoritmo de Optimización Cucú (Cuckoo Optimization Algorithm - COA), a las propuestas de función

objetivo que han sido usadas en la literatura para resolver problemas de optimización de protecciones eléctricas y la revisión bibliográfica de los relés de sobre corriente. Estas referencias se las implementará en este estudio para tomarlas como marco teórico e incorporarlas en el desarrollo del ajuste óptimo de protecciones de sobre corriente temporizada (51).

Se programará el algoritmo COA en lenguaje Python, siguiendo las referencias bibliográficas del algoritmo y comprobando su correcto funcionamiento con las funciones objetivo de comprobación encontradas en la literatura o también llamadas benchmarks test functions.

Se modelarán relés de sobre corriente para proteger el sistema de prueba de 13 nodos de la IEEE en el software de simulación de DlgSILENT PowerFactory y se propondrá un ajuste de las protecciones en base a las mediciones de corrientes de falla realizadas para utilizar los criterios tradicionales de ajuste y coordinación presentados por la literatura. Se implementará para cada relé el respectivo transformador de corriente (TC) para la medición de sobre corrientes, de modo que se pueda llevar a cabo el ajuste de los relés conforme a los datos tomados de sobre corriente siguiendo los criterios de ajuste y ocupando las herramientas tomadas del software PowerFactory.

En la revisión bibliográfica se identificarán las funciones objetivo implementadas para optimizar la coordinación entre protecciones de sobre corriente y se aplicará una función objetivo en el algoritmo COA codificado en Python. Además, una vez que se tenga disponible el funcionamiento del algoritmo y la función objetivo implementada con sus restricciones, se propondrá un diagrama de flujo en el cual conste la metodología seguida por el algoritmo para determinar valores optimizados de los ajustes de relés de sobre corriente implementados en el sistema de prueba.

En este trabajo, en base al alcance que se pretende llegar no se implementarán relés de sobre corriente direccional, por este motivo la direccionalidad de las corrientes no se analizará para la operación de los relés y, además, únicamente se optimizará la coordinación entre los relés 51 implementados en el sistema de prueba mediante la función objetivo propuesta en base a las variables de control tomadas, las cuales son la corriente pickup, el Dial y el tipo de curva.

Mediante la interfaz que presenta el software de PowerFactory, el cual se puede enlazar con la codificación en el lenguaje de Python, se realizará la programación de la metodología aplicada a la coordinación óptima de las protecciones de sobre corriente

temporizada (51) del sistema de 13 nodos de la IEEE usando el algoritmo COA como método de optimización.

Se realizará la comparación de los resultados conseguidos al ajustar los relés 51 mediante los criterios tradicionales tomados de la literatura con los resultados obtenidos del ajuste óptimo. Se analizarán los resultados para determinar si se han obtenido mejoras al implementar la optimización del ajuste de relés 51 que consigue el algoritmo COA a partir del ajuste con los criterios tradicionales, lo que corresponde al objetivo de este trabajo. Mediante el análisis de los resultados se presentan las conclusiones y recomendaciones a las cuales se llega, tomando en cuenta el alcance de este trabajo.

1.4 Marco teórico

Protecciones de sobre corriente en sistemas eléctricos de potencia

Con el aumento a la dependencia de energía que se presenta en la actualidad y que continua en aumento hacia los años futuros, se exige una mayor calidad y confiabilidad en el suministro de energía mediante estándares y normas que permitan ejecutar las condiciones pertinentes. El sistema de potencia puede sufrir fallas de cortocircuitos en cualquier lugar, cualquier instante y de maneras imprevistas. Las fallas deben ser despejadas de forma oportuna para que no sigan produciendo daño a los equipos afectados y evitar que el problema empeore [1].

Al producirse una falla en el sistema, la falla debe ser localizada y aislada en el menor tiempo posible, de esta manera se sigue suministrando energía al resto del sistema que no se encuentra con falla. El equipo afectado debe permanecer aislado hasta que se haya solucionado la falla. Al solucionar la falla, se conectan nuevamente los equipos que se han sacado de servicio y el sistema eléctrico vuelve a sus condiciones normales de operación. Con la operación oportuna de las protecciones eléctricas se puede evitar el colapso del sistema eléctrico [1].

Se debe monitorear, controlar y proteger de manera continua el sistema eléctrico ante fallas, para mantener su correcto funcionamiento. El núcleo y el cerebro de los sistemas de protecciones son los relés, dado que, estos son los encargados de detectar fallas y mandar la señal de operación hacia los interruptores asociados al relé, de esta manera los interruptores reciben la señal de apertura y despejan la falla. Es común la implementación de relés de sobre corriente para detectar y despejar fallas producidas por sobre corriente. Este relé detecta la corriente que sobrepasa la corriente pickup y opera después de un determinado tiempo [1].

La causa principal por la que se pueden producir apagones en cascada se debe al mal funcionamiento de los relés. El relé de sobre corriente es capaz de ser implementado como protección principal o como protección de respaldo, puede ser implementado en sistemas fuertemente mallados y además puede ser implementado en sistemas multi – máquinas. Las ventajas que presenta la implementación del relé de sobre corriente es su bajo costo y la facilidad de su implementación a diferencia de otros relés de protección. El ajuste adecuado de los relés de sobre corriente permite que actúen de forma coordinada entre ellos, en tiempos de operación apropiados [1].

Para ajustar las protecciones de sobre corriente se puede implementar diferentes criterios, entre estos se tienen criterios tradicionales obtenidos por la literatura o se pueden ocupar métodos de optimización. Para ajustar la protección de sobre corriente temporizada (51) se debe ajustar los parámetros que son corriente pickup (I_{set}), el Time Multiplier Setting (TMS) o Dial y el tipo de curva, en el caso de la protección de sobre corriente instantánea (50) solamente se ajusta la corriente pickup [1].

Relés de sobre corriente

Las protecciones de sobre corriente se han desarrollado a un inicio con una tecnología de construcción electromecánica. Con el paso del tiempo se han venido presentando nuevas tecnologías como son el relé de estado sólido, el híbrido y el numérico, que mejoran la fiabilidad, la rapidez, la simplicidad y la economía. Los relés de sobre corriente se clasifican según su tiempo de operación, en donde se tiene el relé de sobre corriente temporizado (51) y el relé de sobre corriente instantáneo (50) [2].

Relé de sobre corriente temporizado (51)

El relé de sobre corriente temporizado utiliza el principio de funcionamiento del disco de inducción. A pesar de que las tecnologías que se han implementado en la actualidad sean diferentes, su principio de funcionamiento es el mismo y se lo puede representar mediante la Figura 1.1. El relé 51 opere en un tiempo más rápido para corrientes más altas, mientras que opera en tiempos menores para corrientes más bajas. Por esta razón, a la curva de operación del relé de sobre corriente temporizado se lo llama curva característica de tiempo inverso [2].

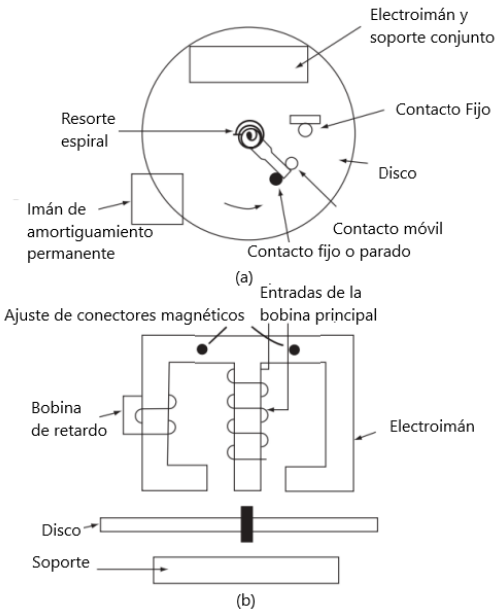


Figura 1.1 Típico disco de inducción tipo relé de sobre corriente: (a) Vista superior, (b) Vista inferior [2].

El principio de funcionamiento del relé 51 se detalla a continuación. Al pasar una corriente alterna en la bobina principal, se produce un flujo magnético, el cual, en su mayoría de veces pasa sobre el entrehierro y el disco hacia el contacto magnético del soporte y regresa sobre las patas laterales del electroimán. En un lado del electroimán se incorpora un pequeño bobinado de retardo que produce un cambio en el tiempo y en el ángulo del flujo hacia ese lado del disco, produciéndose la rotación del disco, esta rotación es amortiguada permanentemente por un electroimán. El resorte en espiral se comprime hasta que la corriente de operación sea mayor a la corriente pickup y se produce el reinicio de los contactos [2].

Los relés de sobre corriente se ajustan mediante su tipo de curva característica, su corriente pickup y su Dial. Se selecciona el tipo de curva del relé 51 más adecuado para cada aplicación, puede ser necesario un tiempo de respuesta más rápido o lento. Se han desarrollado diferentes tipos de curvas características para ajustar los relés 51 y coordinarles con otras protecciones. Los tipos de curvas características son de tiempo inverso, de tiempo muy inverso y de tiempo extremadamente inverso [2].

Se ajusta el tap y el DIAL del relé 51 para que opere de forma adecuada ante fallas de cortocircuito. Todos los relés tienen varios taps, cada tap representa el mínimo valor de corriente para que este opere, a este mínimo valor de corriente se lo llama corriente de

pickup. Se debe tomar en cuenta la tolerancia del fabricante para que en el ajuste se seleccione un tap que le permita actuar adecuadamente al relé ante fallas [2].

El espacio que recorren los contactos del relé para hacer contacto y operar es ajustable. Al ajuste que regula el tiempo que demora en recorrer todo este espacio se lo conoce como Dial o TMS. Con diferentes ajustes del TMS se consiguen diferentes tiempos de operación del relé 51 al utilizar la misma corriente de operación [2].

El ajuste correcto de las protecciones permite despejar las fallas de forma oportuna, sin embargo, el tiempo de operación de los relés de protección es un factor importante que debe ser minimizado para mitigar lo máximo posible las consecuencias de la falla en el sistema. En la literatura se han propuestos algunas alternativas para minimizar el tiempo de operación de los relés 51 mediante la coordinación óptima, usando métodos como: 1) Prueba y error, 2) Método de análisis estructural basado en la teoría de grafos, 3) Método de Optimización, 4) Inteligencia Artificial y 5) Algoritmos inspirados en la naturaleza [1].

Relé de sobre corriente Instantáneo (50)

A estos relés se les implementa en aplicaciones donde se necesita que operen de forma instantánea. A pesar de llamarse instantáneo, sí tiene un pequeño intervalo de tiempo de demora en la actuación, el cual, depende de su fabricante y de la tecnología que se encuentra hecho. Los tiempos típicos de operación son de 16 a 20 milisegundos [2].

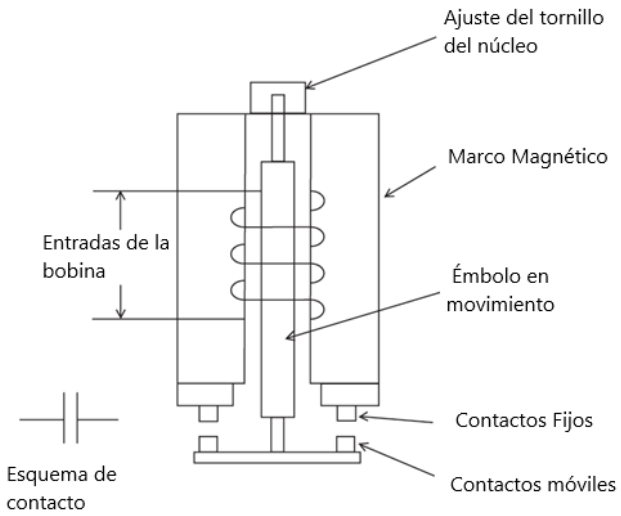


Figura 1.2 Principio de funcionamiento de un relé de sobre corriente instantáneo (50) [2].

La Figura 1.2 muestra el principio de funcionamiento del relé 50 descrito a continuación. Mediante la inyección de corriente continua por las entradas de la bobina, se energiza el relé de manera que se produce un flujo capaz de atraer el émbolo hacia la armadura. Si la corriente de operación supera el valor de la corriente pickup, se tocan los contactos del relé y este opera. En el relé 50 se ajusta únicamente la corriente pickup. El relé tiene varios taps para ajustar la corriente pickup, se selecciona un tap que permita operar el relé 50 adecuadamente, considerando la tolerancia dada por el fabricante [2].

Sistema de 13 nodos de la IEEE

El sistema eléctrico de 13 nodos de la IEEE es empleado en el presente trabajo como sistema de prueba. Este sistema de prueba se ha presentado por primera vez en [3] y se lo ha implementado en el software de PowerFactory en [4]. En [3] se menciona que este es un sistema radial que presenta circuitos trifásicos, bifásicos y monofásicos con cargas desbalanceadas. Se lo utiliza para validar el buen funcionamiento del análisis de sistemas desbalanceados en alimentadores de distribución y comparar los resultados con otros sistemas de prueba [4].

Este sistema eléctrico tiene una frecuencia de 60 [Hz] y se encuentra conformado por 13 nodos, un regulador de voltaje para cada fase, 9 cargas desbalanceadas, 10 líneas entre aéreas y subterráneas con 1, 2 y 3 fases asignadas correspondiente a cada línea según la configuración del sistema, dos bancos de capacitores, un transformador de distribución y un transformador para la subestación. En el Anexo I, mediante la Figura A1.1 se presenta el sistema de 13 nodos de la IEEE modelado en PowerFactory [4].

A continuación, se presenta una descripción de los elementos modelados en PowerFactory del sistema eléctrico. Los auto transformadores utilizados como reguladores de voltaje para cada fase se encuentran situados desde el nodo 650 siguiendo por la línea de compensación de caída de voltaje LOHL650-632 para mantener los voltajes en el nodo 632, se asumen reguladores de voltaje tipo paso. En los reguladores de voltaje se tiene una variación de voltaje por cambio de tap de 0,625 % de cada fase, este valor típico es proporcionado por [4], además se asume que no tienen pérdidas [4].

El sistema eléctrico tiene 9 cargas cuyos datos de P y Q están dados por [3] en kW y kVAr respectivamente para cada fase, estos valores son mostrados en la Tabla A2.1 del Anexo II. Las cargas se encuentran incorporadas mediante los modelos de cargas de

corriente constante, potencia contante e impedancia constante y tienen su configuración en delta o en estrella según corresponda a cada carga del sistema. La carga distribuida que se encuentra entre las barras 632 y 671 es modelada mediante cinco elementos de carga separados por distancias iguales a lo largo de la línea LOHL632-671 conectada entre dichos nodos [4]. Las líneas son modeladas según su tipo de conexión, el cual es identificado con el ID de las líneas que se muestran en la Tabla A2.2 del Anexo II [4].

En la Tabla A2.3 situada en el Anexo II se presentan los elementos conectados entre los nodos del sistema, los cuales son líneas, un transformador y un switch con sus respectivos parámetros. En la referencia [4] se presentan valores de resistencia en AC para las líneas aéreas a 50 grados Celsius, sin embargo, para modelar en PowerFactory se requiere la resistencia DC dada a 20 grados Celsius, de modo que en [4] se implementa los valores de 50 grados Celsius AC a los requeridos en PowerFactory de 20 grados Celsius DC. En la Tabla A2.4 del Anexo II se presentan los parámetros de las líneas según su calibre [4].

Se incorporan dos bancos de capacitores. Uno de estos es conectado en la barra 675 en estrella, inyectando 200 [kVAr] por cada fase en voltaje nominal. Mientras que, el segundo banco de capacitores se encuentra conectado en el nodo 611, es monofásico y entrega 100 [kVAr] a voltaje nominal [4].

En el sistema descrito se presentan también dos transformadores, se tiene implementado un transformador para la subestación cuya potencia es de 5 [MVA] y presenta una relación de transformación de 115/4,16 [kV] en configuración delta-estrella neutro con resistencia relativa de 1 % y reactancia relativa de 8 %, en donde el lado de alto voltaje va conectado en delta. Se tiene un segundo transformador para la distribución llamado XFM-1 de 500 [kVA] con una relación de 4,16/0,48 [kV] con configuración estrella neutro-estrella neutro, con una resistencia relativa de 1,1 % y una reactancia relativa de 2 % [4].

Se modela en el sistema eléctrico el transformador de la subestación. Al transformador de la subestación se lo implementa en la barra 650 a un nivel de voltaje de 1 [pu]. Este transformador se encuentra activado solamente para el caso de estudio "*Substation Transformers*". Este caso de estudio sirve para analizar el efecto que se produce al considerar la impedancia del transformador de la subestación en el sistema eléctrico para el flujo de potencias. [4].

En PowerFactory se tienen varios casos de estudio en la modelación del sistema eléctrico de 13 nodos de la IEEE. Los casos de estudio son implementados para hacer consideraciones en el sistema, que permita determinar el comportamiento de este al

tener la particularidad que señala el caso de estudio. Para este trabajo se selecciona el caso de estudio "*Study Detailed Network Model*", debido a que, se tiene la modelación del sistema eléctrico detallada.

Algoritmo de Optimización Cucú (Cuckoo Optimization Algorithm - COA)

El algoritmo COA es un algoritmo evolucionario inspirado en la naturaleza, su desarrollador es Ramin Rajabioun y ha sido publicado en el año 2011. El estilo de vida reproductiva de una familia de aves llamadas cucú ha inspirado este algoritmo. Las aves cucú practican el parasitismo, estas aves no cuidan a sus crías ni hacen sus propios nidos, sino que engañan a otras aves para que lo hagan por ellas [5].

El algoritmo está constituido por cucús maduros y huevos de cucú. Los cucús maduros ponen sus huevos en los nidos de las aves hospedadoras, haciéndolas creer que son sus huevos. El ave cucú hembra debe mimetizar los colores y la forma de los huevos del ave hospedadora al poner sus huevos, dado que, este es un factor muy importante para que sus huevos no sean reconocidos como parásitos y lanzados de los nidos, provocando su muerte. El huevo cucú al no ser reconocido como parásito por el ave hospedadora, es empollado hasta que eclosiona del huevo, después es alimentado por los padres adoptivos hasta alcanzar su madurez e irse del nido [5].

El algoritmo COA define como hábitat el arreglo de las variables de decisión del problema de optimización, es decir que, si el problema de optimización tiene N variables, entonces el hábitat tiene una dimensión de 1xN_{var}, como se observa mediante la Ecuación 1.1. Cada hábitat representa una posición del nido correspondiente a un cucú y la posición de varios hábitats representa una población de varios cucús. Para representar una población de aves cucú, el algoritmo crea algunos hábitats candidatos mediante el arreglo matricial de dimensiones N_{pop} xN_{var}, dando de forma aleatoria los hábitats o posiciones de los huevos de cada cucú [5].

$$\text{Hábitat} = [\text{X1}, \text{X2}, \text{X3}, \dots, \text{X}_{Nvar}]$$

Ecuación 1.1. Hábitat definido por algoritmo COA [5].

Para situar el centro de puesta de huevos, el cucú maduro fija una posición objetivo de llegada que le permita obtener un mayor número de crías sobrevivientes. Emprende su

vuelo desde el nido en el que se encuentre hacia la posición objetivo. La trayectoria de vuelo del cucú maduro se ve influenciada por la distancia que alcanzan a recorrer, la cual es una λ parte del total del recorrido que se debería recorrer hasta el punto objetivo. También tiene un desvío en la dirección de la trayectoria representado por un ángulo ϕ por influencia del viento u otros factores, como se lo puede apreciar en la Figura 1.3 [5].

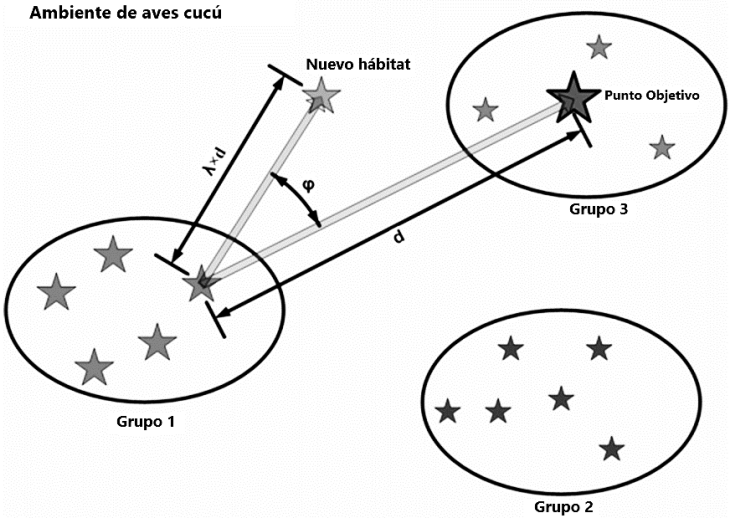


Figura 1.3. Migración de las aves cucú hacia el punto objetivo [5].

El porcentaje de la trayectoria que recorre el cucú hacia su nuevo hábitat representado por λ puede tomar valores entre 0 y 1 (λ~U [0,1]) y el desvío del ángulo de la trayectoria que toma el cucú respecto de la trayectoria que debería tomar hacia el punto objetivo representado por ϕ puede tomar valores entre -ω y ω (ϕ ~U [-ω, ω]). Se menciona en [5] que ω puede tomar un valor de $\frac{\pi}{6}$.y lograr una buena convergencia de la función objetivo planteada [5].

Cada cucú maduro tiene una distancia máxima a la que pone sus huevos la cual es conocida como radio de puesta de huevos, que en inglés toma el nombre de “Egg Laying Radious”. El algoritmo COA define al radio de puesta de huevos con sus siglas en ingles ELR, en la Figura 1.4 se muestra la representación gráfica del ELR para cada cucú maduro. Cada cucú maduro parte desde su centro de puesta de huevos hacia varios puntos obtenidos de forma aleatoria que representan los nidos de las aves hospedadoras, dentro del área que forma el ELR [5].

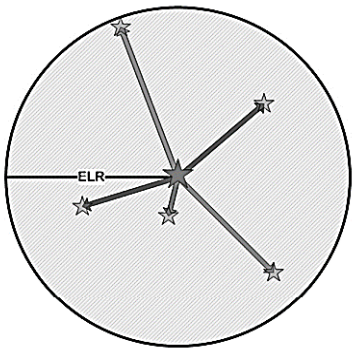


Figura 1.4. Radio de Puesta de Huevos [5].

El valor numérico del ELR, calculado por el algoritmo COA con la Ecuación 1.2, es proporcional al número de huevos que tiene cada cucú, el número total de huevos y la cantidad de huevos que puede poner en cada iteración un cucú. La cantidad de huevos que puede poner un cucú es de un valor entero entre un límite inferior y superior, estos límites son definidos por Var_{Low} y Var_{hi} respectivamente. Cada cucú puede poner de cinco a veinte huevos en cada una de sus puestas de huevos, estos valores son tomados como referencia al definir los límites de puesta de huevos de cada cucú dentro del algoritmo COA [5].

$$ELR = \alpha * \frac{N\acute{u}mero\ de\ huevos\ que\ tiene\ cada\ cuckoo}{N\acute{u}mero\ total\ de\ huevos} * (\mathrm{Var}_{Low}\ \mathrm{y}\ \mathrm{Var}_{hi})$$

Ecuación 1.2. Radio de puesta de huevos [5].

Donde α es un número entero que define un porcentaje del ELR máximo.

Debido a que la cantidad de alimento disponible en el medio ambiente es limitada, no es posible que todas las crías de cucú sobrevivan. El algoritmo COA limita su número de crías sobrevivientes considerando la evaluación calculada con la función objetivo a un número máximo de crías, definido por N_{max}. Las crías que no tienen la suficiente condición para sobrevivir mueren siendo devoradas por sus depredadores o son identificadas como parásitos por las aves hospedadoras y arrojadas de su nido. Las crías sobrevivientes crecen en su sociedad hasta llegar a su madurez [5].

Cuando las aves cucú se encuentran esparcidas por todo el ambiente, es difícil determinar a qué grupo o sociedad pertenecen según su posición. El algoritmo COA utiliza el método de Clustering K-means para agrupar los cucús en sociedades según sus

posiciones, con esto ubicar a la mejor sociedad según la media, y dentro de esta, determinar a la mejor posición de ave cucú según la evaluación con la función objetivo [5]. Según [5], en la implementación del algoritmo COA se recomienda realizar de entre tres a cinco clusters siendo k el número de clusters, para tener las aves cucú agrupadas en k clusters [5].

Se saca la media de las evaluaciones de cada sociedad. La sociedad que tiene la menor media se convierte en sociedad objetivo para la migración de otros grupos de aves cucú, dado que, esta sociedad se encuentra menos dispersa. La sociedad objetivo es la sociedad en donde los nuevos cucús maduros buscan reproducirse, porque las aves hospedadoras tienen huevos similares a los suyos y esa sociedad tiene mejores recursos que la sociedad de donde salieron [5].

En la Figura 1.3 se muestra la implementación del método de k-Means en el algoritmo COA. Se identifican tres clusters, estos son nombrados como Grupo 1, 2 y 3. En el Grupo 3 se tiene la sociedad con menor media, por lo tanto, se convierte en sociedad objetivo. En la sociedad objetivo se evalúa a los cucús mediante la función objetivo para encontrar el cucú que tiene la mejor evaluación. Al hábitat del cucú mejor evaluado de la sociedad objetivo se le determina como punto objetivo, donde los otros cucús quieren realizan su migración y llegar [5].

Clustering k - Means

Según indica [6], el método de clustering es ampliamente utilizado para minería de datos, recuperación de información, segmentación de imágenes y biología computacional. Este método se basa en la partición de un finito número de datos en grupos llamados clusters, los datos que conforman un cluster son similares entre ellos, mientras que son diferentes a los datos correspondientes a otros clusters. El método de cluster k - Means es uno de los más usados, dado que, en varias aplicaciones tradicionales se obtienen resultados satisfactorios [6].

Este método consiste en encontrar los centroides de cada cluster mediante la optimización de la función objetivo. En [6] se ocupa la función objetivo de error cuadrado que se muestra mediante la Ecuación 1.3. Al implementar esta función objetivo en el método de K - Means se busca determinar el centroide que se encuentre más cercano a todos los datos correspondientes al cluster [6].

$$J_{K-means} = \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - m_k)^2$$

Ecuación 1.3. Función objetivo del método Clustering k - Means [6].

Donde: K son los clusters.

m_k es el centroide del cluster k (C_k).

X = (x1, ..., xn) es la matriz de datos.

Este método inicializa los centroides una vez que ya se han definido los k clusters. Conforme el método va iterando en busca de los valores óptimos de los k centroides, los resultados siguen convergiendo. El método conforme itera hacia los resultados óptimos debe tener las siguientes consideraciones: 1) Decisión de las coordenadas del centroide, 2) Decisión de las distancias de cada item (dato) hacia el centroide y 3) Ubicación de cada item en un cluster seleccionado para la obtención de la menor distancia hacia el centroide.

Mediante la Figura 1.5 se presenta un ejemplo de forma gráfica en el cual se implementa el método de k – Means a un número de datos finitos, se define el número de k clusters con un valor de tres, de esta forma se puede observar que al aplicar el método de clustering k - Means se obtiene la división de tres clusters y cada cluster cuenta con su respectivo centroide. Se han desarrollado varias técnicas para mejorar el desempeño del método k - Means para aplicaciones en donde se maneja una elevada cantidad de datos, sin embargo, para este trabajo no se analiza el método k – Means sino que se lo aplica como parte del método COA [7].

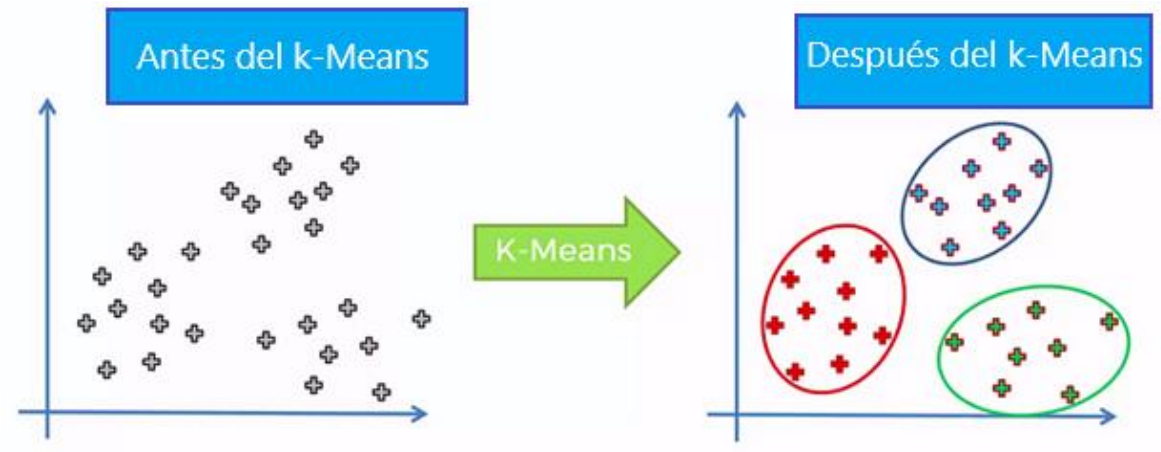


Figura 1.5 Método de K – Means [6].

Algoritmo COA implementado en otros trabajos revisados en la bibliografía

Según [8], con la implementación del algoritmo COA se realiza la estimación de la Demanda de Energía en Irán basándose en sus condiciones socioeconómicas, obteniendo resultados satisfactorios. Se utilizan cuatro modelos para determinar el modelo más preciso para estimar la demanda futura. Los modelos correspondientes a la optimización del problema se han implementado usando formas lineales y de potencia. Cada modelo es la combinación de variables socioeconómicas y demográficas. Los datos de variable socioeconómicas y demográficas que se han utilizado son tomadas desde el año 1972 hasta el año 2013, a los cuales se los implementa para ajustar los coeficientes que presentan las ecuaciones de cada modelo [8].

Una vez que ya se ha definido los coeficientes que componen los modelos, estos son capaces de predecir demandas futuras. En [8] se realiza la predicción de la demanda desde el año 2014 hasta el año 2030 con cada modelo, después se realiza una comparación entre modelos y se determinar cuál es el modelo que más se ajusta al problema de estimación de la demanda de energía. Los resultados arrojados sugieren que el modelo lineal que considera un mayor número de variables socioeconómicas es el que puede predecir la demanda futura de forma más precisa [8].

En [9] se menciona que el algoritmo COA es superior a varios algoritmos de optimización, por las funciones objetivo multimodal debido a la robustez frente a los cambios dinámicos y a su amplia aplicabilidad. En este estudio se combinan dos criterios para formar la función objetivo, el primero optimiza el intervalo de tiempo de carga (CIT) y el segundo optimiza las pérdidas de energía entre el intervalo de carga (EL) de una batería de litio-ion polimérico. Se aplica el algoritmo COA para conseguir un procedimiento eficiente en la carga rápida de baterías de litio – ion polimérico. El algoritmo COA es una técnica efectiva para solucionar este problema y obtener la carga óptima de baterías en intervalos de tiempo de carga óptimos con mínimas pérdidas [9].

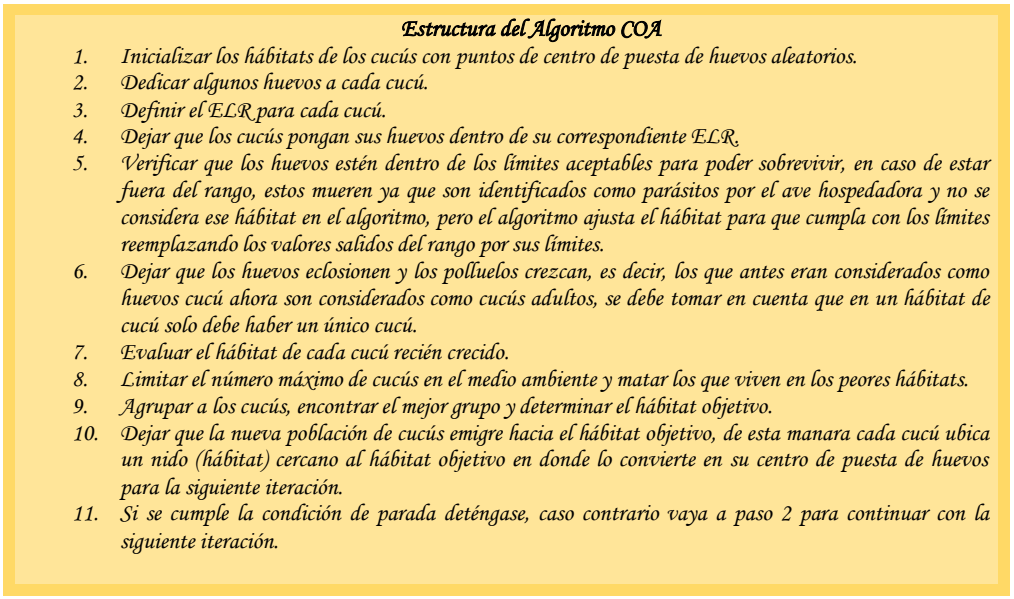
Principales pasos para estructurar el algoritmo COA

De manera general, el algoritmo COA se ejecuta de la siguiente forma:

- Es iniciada una población de aves cucú maduras con una cantidad de huevos, estos son colocados en diferentes nidos de forma aleatoria, una cantidad de huevos sobrevive, de forma que llega a su madurez y forman sociedades, mientras que los otros huevos mueren.

- Se evalúan las sociedades de cucús, de modo que aquella sociedad que presenta una mayor tasa de sobrevivientes es puesta como sociedad objetivo para otras sociedades y se calcula el mejor cucú dentro de la mejor sociedad para ponerla como punto objetivo de llegada para las otras aves cucú que migran.
- Las aves cucú de cada sociedad migran con miras a llegar al punto objetivo, buscando el hábitat óptimo para ellas.
- Se repiten varias iteraciones la migración y puesta de huevos de las sociedades de cucús, de modo que las sociedades de cucú siguen convergiendo en el punto óptimo.

Mediante la Figura 1.6 se presenta la estructura del algoritmo COA con los principales pasos que debe ser codificado:



Estructura del Algoritmo COA

1. *Inicializar los hábitats de los cucús con puntos de centro de puesta de huevos aleatorios.*
2. *Dedicar algunos huevos a cada cucú.*
3. *Definir el ELR para cada cucú.*
4. *Dejar que los cucús pongan sus huevos dentro de su correspondiente ELR.*
5. *Verificar que los huevos estén dentro de los límites aceptables para poder sobrevivir, en caso de estar fuera del rango, estos mueren ya que son identificados como parásitos por el ave hospedadora y no se considera ese hábitat en el algoritmo, pero el algoritmo ajusta el hábitat para que cumpla con los límites reemplazando los valores salidos del rango por sus límites.*
6. *Dejar que los huevos eclosionen y los polluelos crezcan, es decir, los que antes eran considerados como huevos cucú ahora son considerados como cucús adultos, se debe tomar en cuenta que en un hábitat de cucú solo debe haber un único cucú.*
7. *Evaluar el hábitat de cada cucú recién crecido.*
8. *Limitar el número máximo de cucús en el medio ambiente y matar los que viven en los peores hábitats.*
9. *Agrupar a los cucús, encontrar el mejor grupo y determinar el hábitat objetivo.*
10. *Dejar que la nueva población de cucús emigre hacia el hábitat objetivo, de esta manara cada cucú ubica un nido (hábitat) cercano al hábitat objetivo en donde lo convierte en su centro de puesta de huevos para la siguiente iteración.*
11. *Si se cumple la condición de parada deténgase, caso contrario vaya a paso 2 para continuar con la siguiente iteración.*

Figura 1.6. Estructura del algoritmo COA [5].

Función Objetivo del problema de optimización de relés de sobre corriente

El ajuste del relé de protección de sobre corriente temporizado (51) se lo realiza al determinar el ajuste de las variables de decisión según el estándar del tipo de curva característica del relé de sobre corriente (tiempo – corriente) de la IEC, el cual se menciona en [10]. Las variables de decisión que se deben ajustar en la curva característica de la IEC son el tipo de curva, la corriente pickup y el Dial. El ajuste de las variables de decisión debe procurar que el relé opere lo más rápido posible ante una falla

y se despeje el elemento fallado de forma oportuna para mitigar las consecuencias de la falla presentada [10].

Para obtener el ajuste del relé 51 que proporcione una coordinación óptima entre relés, se puede implementar diferentes metodologías utilizando la misma función objetivo. Para este trabajo se implementa el algoritmo COA como método de optimización para realizar la búsqueda del ajuste de los relés de sobre corriente en conjunto con la función objetivo que minimiza los tiempos de operación de cada relé propuesta en [10], para obtener la coordinación óptima de los relés de sobre corriente. En la Ecuación 1.4 se presenta la ecuación de la curva característica (tiempo – corriente) de los relés de sobre corriente del estándar IEC [10].

$$T = \frac{TMS\ x\ \alpha}{(I_{Sc}/I_{PickUp})^{n} - 1}$$

Ecuación 1.4. Curva característica según el estándar IEC [10].

Donde: *TMS* es el Time Multiplier Setting o Dial.

I_{PickUp} es la corriente pickup.

α y n representa el tipo de curva.

I_{Sc} es la corriente de cortocircuito, esta es una variable de prueba que se usa para determinar los tiempos de operación de los relés.

En la Ecuación 1.5 se presenta la función objetivo que minimiza el tiempo de operación total de todos los relés implementados y por ende se minimiza los tiempos de operación individuales de cada relé. Con la Ecuación 1.4 se calcula el tiempo de operación de cada relé. Las variables de decisión del relé 51 que se requiere ajustar en este trabajo son el tipo de curva, la corriente pickup y el Dial o TMS [10].

$$FO = \min \sum_{i=1}^{n} Ti$$

Ecuación 1.5. Función Objetivo para minimizar los tiempos de operación [10].

Esta función objetivo tiene la capacidad de minimizar los tiempos de operación de *n* relés que se encuentren en un sistema eléctrico. Al ser calculado el tiempo de operación de *n* relés, el algoritmo busca las combinaciones de variables de decisión de todos los relés

involucrados, para que al realizar el sumatorio de los tiempos de todos los relés resulte un tiempo total mínimo [10].

El estándar de la IEC del tipo de curva característica de los relés de sobre corriente definen los valores de α y n para cada tipo de curva, el cual se presenta mediante la Tabla 1.1 [10].

Tabla 1.1. Valores constantes para diferentes características de operación del relé de sobre corriente (Estándar IEC).

Tipo de característica	α	*n*
Largo tiempo inverso	120,0	1,0
Extremadamente inverso	80,0	2,0
Muy inverso	13,5	1,0
Normalmente inverso	0,14	0,02

Las variables de decisión necesitan tener restricciones para considerar todas las limitaciones que se tiene en este problema. Para la corriente pickup se debe implementar la restricción de ser menor a la corriente mínima de cortocircuito y al mismo tiempo debe ser mayor a la corriente de flujo de potencia en carga pico, para que tenga una sensibilidad correcta ante las fallas y no opere ante condiciones normales, o que opere ante una falla por más pequeña que esta sea, mientras esta falla esté considerada como falla dentro de los límites de sensibilidad del sistema. A esta restricción se la representa mediante la Ecuación 1.6 [10].

$$I_{PickUp}^{min} \leq I_{PickUp}^{i} \leq I_{PickUp}^{max}$$

Ecuación 1.6. Restricción del límite inferior y superior para el ajuste de corriente pickup de la curva característica de tiempo inverso [10].

El ajuste del Dial presenta de igual manera restricciones dadas por un límite máximo y un límite mínimo como se puede apreciar mediante la Ecuación 1.7. Los límites considerados en el software DlgSILENT PowerFactory se encuentran dados de entre 0,05 a 1, de manera que estos límites se consideran para la implementación de esta restricción [10].

$$TSM_{min} < TSM_i < TSM_{max}$$

Ecuación 1.7. Restricción para el ajuste del Dial de la curva característica de tiempo inverso [10].

Los ajustes que se realizan en el relé de sobre corriente no deben salirse de los límites establecidos para las variables de decisión tanto de corriente pickup como del dial, cuyos rangos son presentados mediante la Ecuación 1.6 y Ecuación 1.7 respectivamente. Las restricciones mostradas anteriormente son restricciones implementadas para el ajuste del relé, sin embargo, también se implementa una restricción para la coordinación entre relés, la cual se encuentra dada por la Ecuación 1.8 [10].

$$\hat{T}_j > T_i + \mathrm{CTI}$$

Ecuación 1.8. Restricción para coordinación entre relés [10].

Mediante la Ecuación 1.8 se restringen los valores del tiempo de operación de los relés de respaldo ($\hat{T}_j$) para que sean mayores a la suma de CTI con el tiempo de operación del relé principal (T_i). En donde, CTI es el intervalo de tiempo de coordinación (Coordination Time Interval), el cual se pueden asignar valores entre 0,2 a 0,5 segundos. Esta restricción asegura que se tenga coordinados relés primarios como relés de respaldo, en caso de no cumplirse esta restricción, el relé puede operar antes de lo esperado [10].

2 METODOLOGÍA

La metodología propuesta en este trabajo que permitirá determinar la coordinación óptima de relés 51, tiene las características mencionadas a continuación. Es de tipo deductiva, dado que las mejoras de los resultados obtenidos con el procedimiento de la optimización se deducen a partir de la comparación con los resultados conseguidos al aplicar los criterios tradicionales de ajuste de relés 51. Tiene un enfoque cuantitativo, debido a que se requiere cuantificar la disminución de los tiempos de operación de los relés 51 al aplicar el ajuste óptimo. Además, se determina de forma cuantitativa la eficiencia del algoritmo COA a través del tiempo de ejecución que se demora para conseguir la coordinación óptima de relés 51 en el sistema de prueba.

Este trabajo es experimental, debido a que busca aplicar el algoritmo COA con la función objetivo de minimización de tiempos de operación y las restricciones para obtener una coordinación óptima entre relés 51 en el sistema de prueba de 13 nodos de la IEEE. La información de los procedimientos para el ajuste tradicional y el ajuste óptimo de los relés 51 se obtiene de forma numérica con los resultados obtenidos del algoritmo COA (Python) y de las simulaciones realizadas en el sistema eléctrico de prueba (PowerFactory). Se almacena la información relevante en hojas de Excel para hacer un análisis comparativo con los resultados obtenidos entre procedimientos.

Se consideran algunos aspectos de programación y simulación para implementar el procedimiento de la optimización en el sistema de prueba seleccionado. Al sistema de prueba se le incorpora relés de sobre corriente dentro de dicha modelación como sistema de protección para supervisar, controlar y despejar fallas de cortocircuitos en los alimentadores LOHL650-632 y LOHL632-633. Se requiere realizar el ajuste de los relés, de tal forma que se toman criterios dados por la literatura para obtener el ajuste y la coordinación de los relés 50 y 51 en el sistema, este es considerado el criterio base de ajuste.

Con el procedimiento de la optimización se disminuyen los tiempos de operación de los relés ante fallas y al mismo tiempo se consigue que operen de forma coordinada. Se obtiene la coordinación óptima de los relés 51 en el alimentador LOHL632-633, debido a que en este alimentador se superponen las zonas de protección de los relés 51 incorporados. Las corrientes elevadas son despejadas mediante las protecciones instantáneas, considerando un alcance hasta del 80% del alimentador.

Los ajustes de la característica instantánea (50) se consideran como ajustes finales, es decir, no se realizará optimización de estos ajustes mientras que, los ajustes de la

característica temporizada (51) son tomados como referenciales, es decir que, si se empleará la optimización. Se establece un barrido de cortocircuitos por todo el alimentador LOHL632- 633, tanto con el ajuste mediante el criterio base como con el ajuste óptimo obtenido del procedimiento de optimización de los relés 51. Con el barrido de fallas realizado para los dos casos de ajustes se hace una comparación de los tiempos de operación y se verifica que dichos tiempos disminuyen con la optimización.

Para realizar la coordinación óptima de los relés 51 con el algoritmo COA, se codifica a la función objetivo en Python como parte del algoritmo de optimización. La función objetivo necesita obtener del sistema de prueba a las corrientes establecidas como límites de la restricción de la corriente pickup y los tiempos de operación de los relés 51 para verificar el cumplimiento del Intervalo de Tiempo de Coordinación (CTI) entre relés. Además, con los tiempos de operación de los relés 51 obtenidos por la corriente de prueba se realiza la sumatoria de los tiempos de operación.

En primera instancia, se resuelve el problema de coordinacion óptima de los relés 51 con el algoritmo COA. La ejecución del algoritmo COA para resolver el problema de coordinacion óptima es rápida y se puede determinar que los valores entregados del ajuste óptimo son coherentes. Se vincula entre el algoritmo de optimización (Python) y el sistema eléctrico de prueba (PowerFactory) para mejorar la optimización y automatizar la ejecución de los cálculos realizados en el sistema de pruebas y el intercambio de datos.

A continuación, se presenta la metodología seguida.

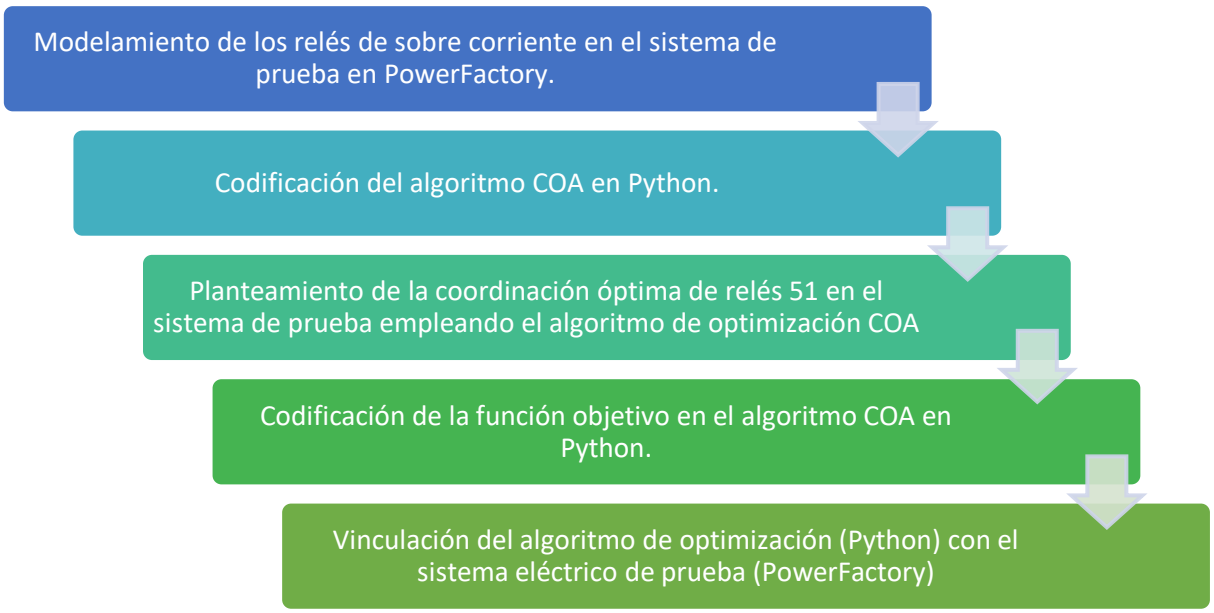


Figura 2.1. Metodología propuesta del proceso de optimización para la coordinación de los relés de sobre corriente temporizado (51).

La Figura 2.1 muestra la metodología seguida en este trabajo para optimizar la coordinación de relés 51. Esta metodología se establece de la siguiente manera, como primer paso, se modela el sistema de protecciones de sobre corriente dentro del sistema de prueba de 13 nodos de la IEEE en el software PowerFactory y se ajustan las protecciones mediante el criterio base. Como segundo paso, el algoritmo COA es codificado en el software de Python, su correcto funcionamiento es probado mediante distintas funciones de prueba que han sido tomadas de la literatura. Como tercer paso, se hace el planteamiento del problema de coordinación óptima de relés 51 en el sistema eléctrico de prueba, para poder resolverlo mediante el algoritmo COA.

Como cuarto paso, se codifica la función objetivo y las restricciones del problema de coordinación óptima de relés de sobre corriente temporizado (51) en Python. Con el algoritmo COA junto a la función objetivo se consiguen resultados óptimos preliminares del ajuste. Como quinto paso se vincula entre el código generado en Python y el sistema eléctrico de prueba modelado en el software PowerFactory para intercambiar información de forma automática entre estos programas y mejorar la búsqueda de valores óptimos, mejorando los resultados de la optimización conseguida en el cuarto paso. Los resultados relevantes son almacenados en hojas de Excel para analizar las mejoras obtenidas con el procedimiento de la optimización.

2.1 Modelación de los relés de sobre corriente en el sistema de prueba en PowerFactory

En el sistema de prueba de 13 nodos de la IEEE se implementan dos relés de protección de sobre corriente 50/51 de fase. Uno de estos se coloca en la entrada del alimentador troncal LOHL650-632 en el lado del nodo RG60, por facilidad se nombra a la característica instantánea como “50_RG60”, a la característica temporizada se la nombra como “51_RG60” y para referirse a ambas características se lo nombra relé “RG60”. Mientras que, al otro relé se lo coloca en la entrada del alimentador ramal LOHL632-633 en el lado del nodo 632, por facilidad se nombra a la característica instantánea como “50_632”, a la característica temporizada se la nombra como “51_632” y para referirse a ambas características se lo nombra relé “632”.

Se puede ver en la Figura 2.2 mostrada a continuación y en la Figura A1.1 mostrada en el Anexo I que en ambas se ha marcado de color rojo la modelación de los relés en el sistema de prueba con sus respectivos TCs. Se implementa un transformador de

corriente o TC de 25 VA 10P20 para pasar de Amperios primarios a Amperios secundarios la corriente de falla para cada relé 50/51. Teniendo en cuenta que los relés implementados son de 5 [A], se selecciona para el TC del relé RG60 una relación de transformación de 1000/5, mientras que para el relé 632 se selecciona una relación de 500/5. Mediante la modelación de los relés se puede simular con diferentes ajustes su respuesta ante varias corrientes de cortocircuito y de flujo de potencia. Con esto, se puede determinar el ajuste adecuado de los relés para conseguir que operen apropiadamente ante una falla.

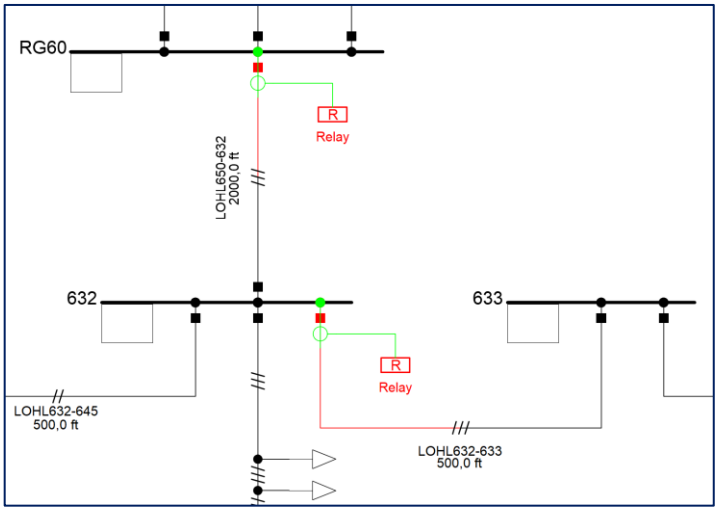


Figura 2.2. Relés de sobre corriente incorporados en el sistema de prueba.

Para realizar el ajuste del relé de sobre corriente instantáneo (50) se utiliza el criterio tradicional tomado de [12], en el que se realiza un estudio de cortocircuitos al 80% del alimentador protegido. Se ajusta la corriente pickup del relé 50 al 80% del alimentador que protege para asegurar la selectividad de la protección al ocurrir una falla, sin correr el riesgo de una inadecuada operación del relé por la falta de precisión de los instrumentos de medición. En el estudio de cortocircuitos se determina el cortocircuito de máxima corriente de falla en demanda máxima. Se ajusta la corriente pickup del relé 50 con el valor de la máxima corriente de falla obtenida por el estudio de cortocircuitos.

En el relé 50 únicamente se ajusta la corriente pickup mediante la máxima corriente de cortocircuito, dado que su tipo de curva es instantánea y el tiempo de operación depende de la tecnología del relé, la cual debe ser lo más rápida posible. En el sistema de prueba seleccionado, la máxima corriente de cortocircuito se produce por la falla trifásica a demanda máxima. Mediante la Tabla 2.1 se muestran los datos de ajuste obtenidos

usando el criterio tradicional para los relés 50 del sistema de prueba, correspondientes a los relés 50_RG60 y 50_632.

Tabla 2.1. Ajuste de los relés de sobre corriente instantáneo (50) modelados en el sistema de prueba.

Relé	Ajuste	Valor	Unidad
Relé 50_ 632	Pickup Current	17,04	pu
	Time setting	0,20	s
Relé 50_ RG60	Pickup Current	14,38	pu
	Time setting	0,20	s

Para realizar el ajuste del relé de sobre corriente temporizado (51) se sigue el criterio proporcionado por [12], en el que se realiza un estudio de cortocircuitos para determinar la corriente de falla mínima a demanda mínima en el sistema de prueba. Se utiliza la corriente de falla mínima para el ajuste de la corriente de pickup del relé 51. Se realiza el estudio de cortocircuitos en el punto más lejano del alimentador al que pueda ajustarse la protección.

El relé 51 es ajustado para proteger hasta el punto más alejado del alimentador seleccionado, ante las fallas menos sensibles, siempre y cuando la corriente de falla mínima sea mayor a la corriente de demanda máxima. El cumplimiento de la restricción mencionada anteriormente hace que los relés 51 no actúen en condiciones normales de operación y que sean capaces de censar y despejar fallas poco sensibles. En caso de no cumplir con la restricción, se selecciona otro punto que esté más cercano al relé 51 que el punto que no cumple, de esta forma, las corrientes de falla resultan ser mayores y se debe verificar que en ese caso si se cumpla con la restricción.

Para la coordinación entre las protecciones, se debe ajustar dichas protecciones desde la carga hacia la generación. [12] Se considera la curva característica más cercana a la carga como el punto de partida para el ajuste de las protecciones, a partir de esta protección las protecciones que se encuentran aguas arriba deben tener un tiempo de operación entre protecciones de aproximadamente 300 milisegundos. Para conseguir una coordinación adecuada entre relés, se debe seleccionar el tipo de curva y ajustar el Dial, para cada relé.

En la Figura 2.3 se observa el ajuste de los relés RG60 y 632 mediante las curvas características mostradas con color rojo y verde respectivamente, obtenidas del software de PowerFactory. Se selecciona para ambas curvas características el tipo de curva muy

inverso, se ajusta la corriente pickup para darle sensibilidad ante fallas y se ajusta el dial para que exista una coordinación adecuada entre relés. A manera de ejemplo del cálculo de corriente pickup, se muestra en la Figura 2.3 mediante la línea vertical roja el cálculo de la corriente de falla mínima vista por el relé 51_RG60, a esta corriente se la define como corriente pickup de la curva roja. De la misma manera se calcula la corriente pickup de la curva verde, mediante la corriente de falla mínima vista por el relé 51_632.

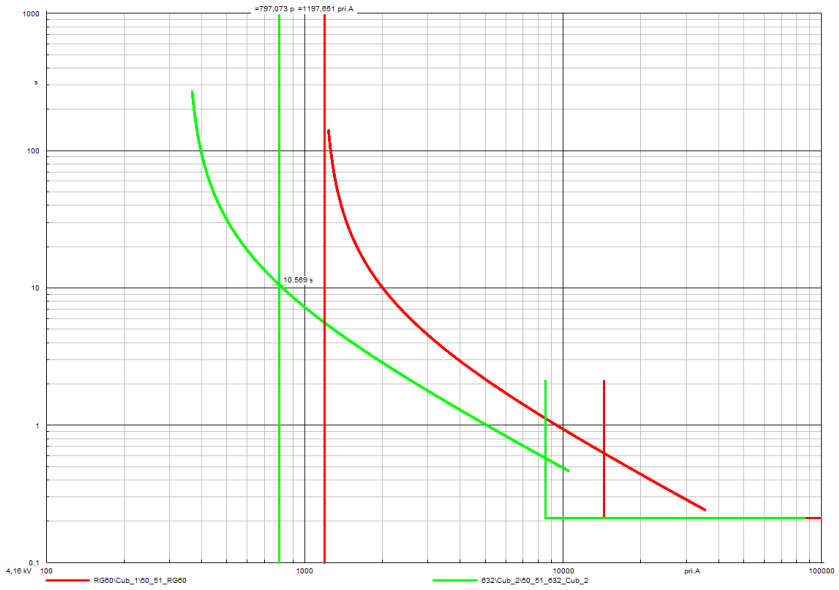


Figura 2.3. Curvas características de los relés RG60 y 632 con el cálculo de la corriente pickup del relé 51_RG60.

Mediante la Tabla 2.2 se presentan los ajustes obtenidos con el criterio tradicional para los relés 51 del sistema de prueba, correspondientes a los relés 51_RG60 y 51_632. La falla mínima considerada para ajustar la corriente pickup de los relés 51 corresponde a la falla fase a tierra en el nodo 633 a mínima corriente de falla con resistencia de falla, para los dos relés. Se considera una resistencia de falla de 8 ohmios para determinar la mínima corriente de falla del relé 51_632 mientras que para determinar la mínima corriente de falla del relé 51_RG60 se considera una resistencia de falla de 3 ohmios.

Tabla 2.2. Ajuste de los relés de sobre corriente temporizado (51) para el sistema de prueba.

Relé	Ajuste	Valor	Unidad
Relé 51_ 632	TimeDial	1,00	-
	Pickup Current	0,70	pu
	Characteristic	IEC class B (Very Inverse)	-
	Tiempo	0,560	s
Relé 51_RG60	TimeDial	0,52	-
	Pickup Current	1,18	pu
	Characteristic	IEC class B (Very Inverse)	-
	Tiempo	1,074	s

2.2 Codificación del algoritmo COA en Python

Para codificar el algoritmo COA en Python, se toma como referencia el código mostrado en [12] realizado en el software Matlab, el cual ha sido proporcionado por el mismo desarrollador del algoritmo COA, Ramin Rajabioun. En el Anexo III se presenta la explicación de la estructura del algoritmo COA implementado en Python. Se toma de la literatura a ocho funciones objetivo de prueba. Con los resultados obtenidos de las funciones objetivo se comprueba que el algoritmo en Python esté programado correctamente.

A continuación, se presentan los resultados obtenidos del algoritmo COA para las funciones objetivo de prueba, llevando a cabo ciento un iteraciones dentro de su proceso de optimización:

1. Primera función objetivo de prueba (Función de Rastrigin):

$$f(x) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Ecuación 2.1. Función de Rastrigin [13].

Esta función objetivo tiene un mínimo global de $f_* = 0$ dado por los parámetros óptimos (0,0,…,0) que se encuentran entre los límites $-5{,}12 \leq x_i \leq 5{,}12$, donde $i = 1{,}2, \ldots, d$ [13]. Para probar al algoritmo COA que se ha codificado para este trabajo en Python con esta función objetivo, se define d=9 para que el algoritmo optimice nueve variables de decisión. En la Tabla 2.3 se presentan los resultados óptimos obtenidos por el algoritmo COA.

Tabla 2.3. Resultados obtenidos del algoritmo COA para la función de Rastrigin con d=9.

Best Params	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**X8**	**X9**
	5,33E-09	-7,25E-11	2,76E-14	-1,99E-09	-9,22E-12	1,67E-10	-8,15E-10	-1,35E-10	-1,81E-10
Cost	0								

Donde Best Params representa el valor óptimo de las variables de decisión y Cost representa la evaluación de la función objetivo obtenida del hábitat de parámetros óptimos.

La Tabla 2.3 presenta los nueve parámetros que el algoritmo COA ha optimizado para obtener el mínimo global de cero, en donde, cada parámetro resulta ser aproximadamente cero. El algoritmo de optimización ha resuelto el problema dado por la función de Rastrigin en un tiempo de ejecución de 10,85 segundos.

2. Segunda función objetivo de prueba (Función de Rosenbrock para dos parámetros):

$$f(x,y) = 100(y - x^2)^2 + (1 - y)^2$$

Ecuación 2.2. Función de Rosenbrock para dos parámetros [14].

El mínimo global de la función de Rosenbrock es de $f_{(x,y)} = 0$ cuando los parámetros óptimos son $\mathrm{x} = 1$ y $\mathrm{y} = 1$ entre los límites de -2,048 y 2,048 para ambos parámetros [14]. En la Tabla 2.4 se presenta los resultados óptimos obtenidos del algoritmo COA después de un tiempo de ejecución de 7,35 segundos.

Tabla 2.4. Resultados obtenidos del algoritmo COA para la función de Rosenbrock.

Best Params	**x**	**y**
	1,0000014	0,9999246
Cost	6,19E-07	

3. Tercera función objetivo de prueba:

$$f(x,y) = \mathrm{x}\sin(4\mathrm{x}) + 1.1\mathrm{y} \times \sin(2\mathrm{y})$$

Ecuación 2.3. Tercera función objetivo de prueba usada con algoritmo COA [5].

La tercera función objetivo presenta un mínimo global de $f_{(x,y)} = -18{,}5547$ cuando los parámetros son $x = 9{,}039$ y $y = 8{,}668$ entre los límites de $x > 0 \wedge y > 0$ [5]. Mediante la Tabla 2.5 se muestran los resultados óptimos calculados por el algoritmo COA, obtenidos después de un tiempo de ejecución de 7,60 segundos.

Tabla 2.5. Resultados obtenidos del algoritmo COA para la tercera función objetivo de prueba.

Best Params	**x**	**y**
	9,03915992	8,6682699
Cost	-18,5547189	

4. Cuarta función objetivo de prueba:

$$f(x,y) = 0{,}5 + \frac{sin^2\left(\sqrt{x^2+y^2} - 0{,}5\right)}{1 + 0{,}1(x^2+y^2)}$$

Ecuación 2.4. Cuarta función objetivo de prueba usada con algoritmo COA [5].

La cuarta función objetivo presenta un mínimo global de $f_{(x,y)} = 0{,}5$ cuando los parámetros son $x = 0$ y $y = 0{,}5$ entre los límites de $x \geq 0 \wedge y < 2$ [5]. En la Tabla 2.6 se muestran los resultados óptimos calculados por el algoritmo COA, obtenidos después de un tiempo de ejecución de 7,91 segundos.

Tabla 2.6. Resultados obtenidos del algoritmo COA para la cuarta función objetivo de prueba.

Best Params	**x**	**y**
	0	0,5
Cost	0,5	

5. Quinta función objetivo de prueba:

$$f(x,y) = (x^2+y^2)^{0{,}25}\,\sin\{30[(x+0{,}5)^2+y^2]^{0{,}1}\} + |x| + |y|$$

Ecuación 2.5. Quinta función objetivo de prueba usada con algoritmo COA [5].

La quinta función objetivo presenta un mínimo global de $f_{(x,y)} = -0{,}2471$ cuando los parámetros son $x = -0{,}2$ y $y = 0$ entre los límites de $x > -\infty \wedge y < +\infty$ [5]. En la Tabla 2.7 se muestran los resultados óptimos calculados por el algoritmo COA, obtenidos después de un tiempo de ejecución de 8,06 segundos.

Tabla 2.7. Resultados obtenidos del algoritmo COA para la quinta función objetivo de prueba.

Best Params	**x**	**y**
	-2,02E-01	3,39E-05
Cost	-0,24735677	

6. Sexta función objetivo de prueba:

$$f(x,y) = J_0(x^2 + y^2) + 0{,}1|1 - x| + 0{,}1|1 - y|$$

Ecuación 2.6. Sexta función objetivo de prueba usada con algoritmo COA [5].

La sexta función objetivo presenta un mínimo global de $f_{(x,y)} = -0{,}3356$ cuando los parámetros son $x = 1$ y $y = 1{,}6606$ entre los límites de $x > -\infty \wedge y < +\infty$ [5]. La Tabla 2.8 muestra los resultados óptimos calculados por el algoritmo COA, obtenidos después de un tiempo de ejecución de 7,44 segundos.

Tabla 2.8. Resultados obtenidos del algoritmo COA para la sexta función objetivo de prueba.

Best Params	**x**	**y**
	1,00011111	1,66029528
Cost	-0,33558197	

7. Séptima función objetivo de prueba (Función generalizada de Rosenbrock):

$$f(x) = \sum_{i=1}^{d-1} [(1 - x_i)^2 + 100(x_{i+1} - {x_i}^2)^2]$$

Ecuación 2.7. Función generalizada de Rosenbrock [13].

Esta función objetivo tiene un mínimo global de $f_* = 0$ dado por los parámetros óptimos (1,1,…,1) que se encuentran entre los límites $-2{,}048 \leq x_i \leq 2{,}048$, donde $i = 1,2,\ldots,d$ [13]. Para probar al algoritmo COA que se ha codificado para este trabajo en Python con esta función objetivo, se define d=9 para que el algoritmo optimice nueve variables de decisión. En la Tabla 2.9 se presentan los resultados óptimos, obtenidos después de un tiempo de ejecución de 11,65 segundos.

Tabla 2.9. Resultados obtenidos del algoritmo COA para la función generalizada de Rosenbrock.

Best Params	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**
	0,9957	0,9979	0,9970	0,9965	0,9965	0,9998	1,0069	1,0167	1,0305
Cost	0,01840176								

8. Octava función objetivo de prueba (Primera función de Jong):

$$f(x) = \sum_{i=1}^{d} x_i^2$$

Ecuación 2.8. Octava función de prueba usada con algoritmo COA [13].

Esta función objetivo tiene un mínimo global de $f_* = 0$ dado por los parámetros óptimos (0,0,…,0) que se encuentran entre los límites $-5.12 \leq x_i \leq 5.12$ donde $i = 1,2,\dots,d$ [13]. Para probar al algoritmo COA que se ha codificado para este trabajo en Python con esta función objetivo, se define d=9 para que el algoritmo optimice nueve variables de decisión. Se obtienen los resultados óptimos mostrados en la Tabla 2.10 en un tiempo de ejecución de 11,75 segundos.

Tabla 2.10. Resultados obtenidos del algoritmo COA para la octava función objetivo de prueba.

Best Params	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**
	-3,11E-39	-7,82E-45	3,06E-39	-2,59E-49	6,22E-45	-1,01E+41	-7,90E-42	2,25E-42	3,51E-41
Cost	1,90E-77								

Los resultados obtenidos del algoritmo COA para las ocho funciones objetivo de prueba son aproximados a los valores óptimos teóricos proporcionados por las referencias señaladas de las funciones objetivo de prueba utilizadas. Se comprueba que la programación del algoritmo de optimización se encuentra correctamente realizada al obtener resultados óptimos coherentes para todas las funciones de prueba.

2.3 Planteamiento de la coordinación óptima de relés 51 en el sistema de prueba empleando el algoritmo de optimización COA.

Se requiere ajustar los parámetros de cada relé 51 implementados en el sistema de prueba con valores que consigan la coordinación óptima de los relés 51 mediante el algoritmo COA. Cada relé 51 tiene tres parámetros a los cuales se debe ajustar, estos son la corriente pickup, el Dial y el tipo de curva según el estándar IEC. El algoritmo COA tiene la capacidad de ajustar *n* relés, de modo que, el número de variables de decisión es de *3xn*. El algoritmo presenta las *3xn* variables de decisión que debe optimizar mediante un vector de dimensión *1xNvar* para los *n* relés, al cual lo define como hábitat.

El algoritmo COA inicializa varios hábitats de forma aleatoria mediante una matriz de dimensiones *NpopxNvar*, en donde, cada fila es a un hábitat. El algoritmo realiza la búsqueda de valores que se encuentran entre los límites VarLow y VarHi para las *3xn* variables. VarLow es el límite inferior y VarHi el límite superior de las variables. Es decir que, este algoritmo no realiza la búsqueda de variables que se encuentran fuera de estos límites y las *3xn* variables tienen los mismos límites. Se opta por definir los valores de los límites VarLow y VarHi con 0 y 1, respectivamente.

Considerando que, el algoritmo tiene a todas las variables entre los límites VarLow y VarHi, se necesita transformar cada variable hacia su nuevo límite requerido por el problema de optimización en la función objetivo, para que, la combinación de variables de decisión (hábitat) pueda ser evaluada. Mediante la Figura 2.4 se ilustra la estructura de una combinación o hábitat para optimizar la coordinación de *n* relés 51, considerando las tres variables para cada relé.

TMS1	**Ipickup1**	**Tipo de Curva1**	**TMS2**	**Ipickup2**	**Tipo de Curva2**	**…**	**TMS*n***	**Ipickup*n***	**Tipo de Curva*n***

Figura 2.4. Hábitat de las variables de decisión para *n* relés 51 en el algoritmo COA.

Para este trabajo se ha propuesto coordinar dos relés mediante el algoritmo de optimización COA, los cuales se han incorporado en el sistema de prueba modelado en el software PowerFactory, detallado en el numeral 2.1. De tal manera que, el número de variables de decisión que requiere optimizar el algoritmo COA es de seis. Por facilidad, se nombra al relé 51_RG60 como relé 1, mientras que al relé 51_632 se lo nombra relé 2. Al finalizar la optimización del algoritmo COA, se obtiene un hábitat entre los límites VarLow

y VarHi con los parámetros óptimos. A estos parámetros se los transforma hacia sus nuevos límites para el ajuste de los relés 51 en el sistema de prueba.

El algoritmo COA optimiza las variables de decisión procediendo de la forma mencionada a continuación. En cada iteración, para cada combinación inicializada por el algoritmo COA (hábitat de cucú maduros), se realiza una búsqueda de nuevas combinaciones, estas nuevas combinaciones (hábitats de cucús crías) tienen como centro al hábitat de cucú maduros y como radio máximo de búsqueda al ELR definido por el algoritmo. Los hábitats de cucús maduros y cucús crías se unen, formando una matriz de dimensiones *NpopxNvar*, de manera que, el nuevo número de población o número de combinaciones (*Npop*) ha incrementado y cada fila es un hábitat o combinación.

El algoritmo obtiene la evaluación de cada combinación de la función objetivo y ordena de mejor a peor combinación, priorizando las *Nmax* mejores combinaciones, las demás son descartadas. Al determinar la mejor combinación o hábitat, este valor es almacenado y guardado como mejor posición dado en la iteración, además, se almacena como mejor posición global si esta posición resulta ser la mejor posición dentro de todas las iteraciones realizadas hasta ese momento.

Para mejorar la búsqueda, el algoritmo COA separa la matriz de combinaciones en Knn clusters (K-means), a los cuales el algoritmo COA lo define como sociedades. Se calcula la media a las evaluaciones de las combinaciones correspondientes a cada sociedad, para determinar cuál es la mejor sociedad con menor media, y a esta sociedad se le halla la mejor posición. Todas las combinaciones o hábitats mueven su posición, buscando dirigirse hacia la mejor posición obtenida de la mejor sociedad.

Se unen todas las sociedades a una sola matriz de dimensiones *NpopxNvar*, se almacena en la última fila la mejor posición global para que no se pierda esa posición, mientras que se almacena en la penúltima fila la mejor posición global multiplicada por un número aleatorio para ver si aledaño a la mejor posición se mejora la actual mejor posición global. La matriz de posiciones obtenida es almacenada dentro de una lista y se la manda a la siguiente iteración como los nuevos hábitats de cucús maduros o centros de puesta de huevos como lo define el algoritmo COA.

Mientras se sigue iterando, el algoritmo sigue evaluando nuevas posiciones y consiguiendo las mejores evaluaciones para las posiciones hasta que se converge a soluciones óptimas. Los resultados óptimos a los que converge el algoritmo COA dependen de la función objetivo que se esté utilizando, ya que la función objetivo con sus restricciones representa un problema de optimización. Mediante la Figura 2.5 se presenta

el procedimiento del algoritmo COA para resolver el problema de coordinacion óptima de los dos relés 51 incorporados en el sistema de prueba.

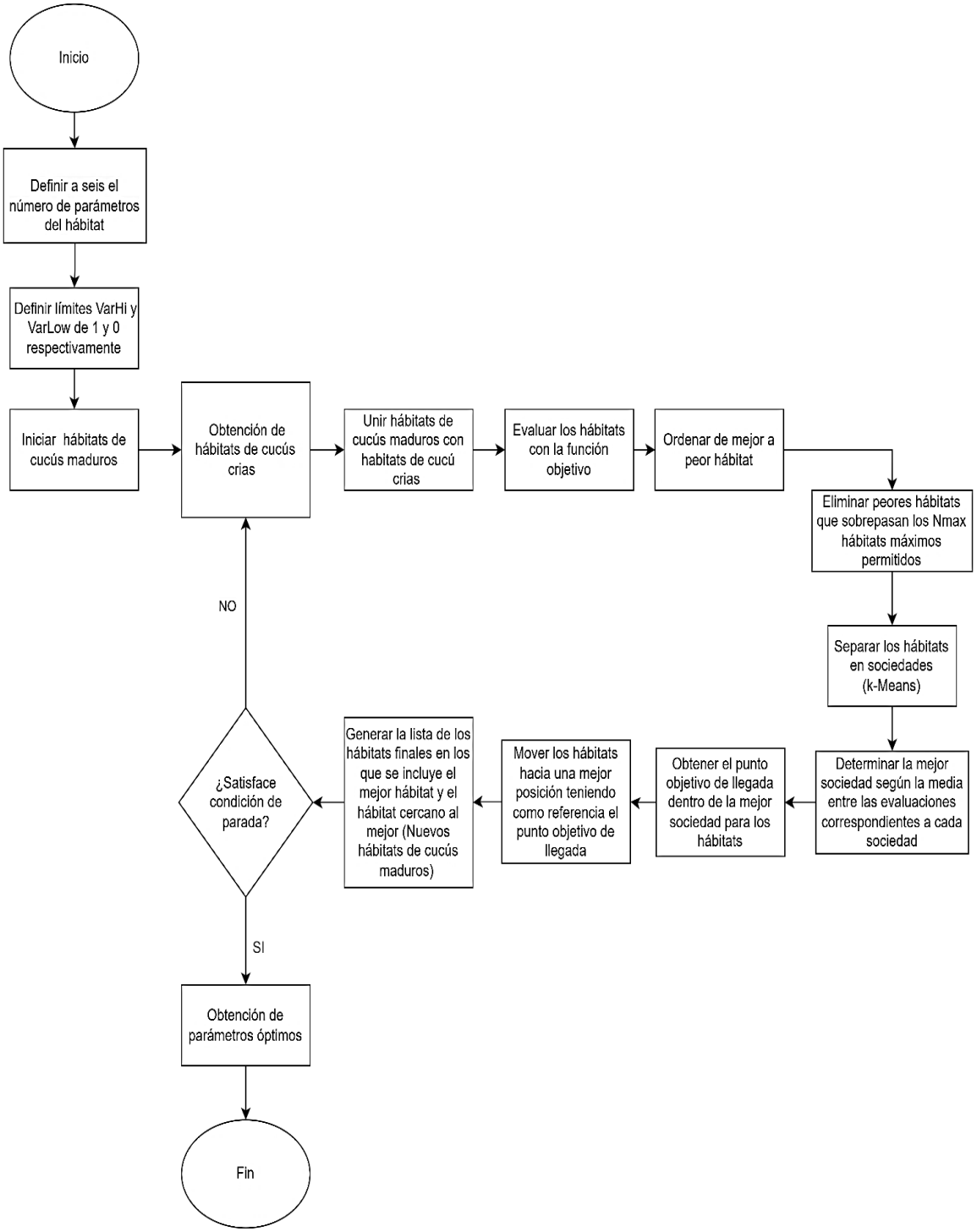


Figura 2.5. Planteamiento del uso del algoritmo COA para la obtención de la coordinación óptima de relés 51 del sistema de prueba.

2.4 Codificación de la función objetivo en el algoritmo COA en Python

Mediante la Ecuación 1.5 se define la función objetivo de minimización de tiempos de operación que se ha codificado en Python para resolver el problema de coordinación óptima de relés de sobre corriente temporizados (51). El tiempo de operación para cada relé ante una ocurrencia de falla se calcula con la Ecuación 1.4, debido a que los relés 51 del sistema de prueba usan el estándar IEC para definir sus curvas características. Se utiliza el algoritmo COA como método de optimización para resolver el problema definido por la función objetivo. Al incorporar esta función objetivo con sus restricciones en el algoritmo COA, el algoritmo se encarga de resolver la coordinación óptima de relés 51.

Al ingresar en la función objetivo las variables de decisión para el ajuste de los *n* relés 51 y el correspondiente valor de la corriente de prueba, se calcula el tiempo de operación para los relés y estos tiempos son sumados. Son ingresados en la función objetivo los valores obtenidos del PowerFactory al hacer el cálculo de la corriente de prueba en el sistema de prueba. Se ingresa en la función objetivo la matriz de combinaciones *NpopxNvar* del algoritmo COA, en donde cada fila corresponde a una combinación de variables de decisión para el ajuste de los *n* relés. La función objetivo debe evaluar a todas las combinaciones dentro de la matriz y devolver todas las evaluaciones al algoritmo COA.

Para evaluar cada combinación, se requiere implementar restricciones a las variables de decisión, permitiendo que el algoritmo descarte aquellas combinaciones que no cumplan con lo requerido. En caso de infringir con las restricciones, se suma en los tiempos de operación total a los valores obtenidos por los pesos de las restricciones, consiguiendo una mala evaluación por el elevado valor obtenido. Si la combinación cumple con las restricciones no se suma ningún valor al sumatorio de tiempos de operación. El algoritmo COA minimiza la evaluación con la búsqueda de nuevas combinaciones que se acercan a la mejor evaluación conseguida hasta el momento y descartando las peores combinaciones.

Cada variable de decisión dada por el algoritmo COA es un número que se encuentra dentro de los límites VarLow y VarHi, por lo que, es necesario transformarlos hacia sus nuevos límites requeridos para resolver el problema de optimización y aplicar las restricciones dentro de la función objetivo.

Normalización de Parámetros

Para cambiar de límites al valor de cada variable de decisión proporcionada por el algoritmo COA, se implementa la ecuación de normalización Min-Max definida en Ecuación 2.9.

$$v' = \frac{v - mín}{máx - mín}(nuevo_{máx} - nuevo_{mín}) + nuevo_{mín}$$

Ecuación 2.9. Ecuación de normalización Min-Max [15].

Donde: v' es el nuevo valor normalizado.

v es el valor sin normalizar

mín es el límite inferior en unidades iniciales.

máx es el límite superior en unidades iniciales.

$nuevo_{mín}$ es el límite inferior de la normalización.

$nuevo_{máx}$ es el límite superior de la normalización.

Tomando en cuenta que los límites VarLow y VarHi de las variables de decisión proporcionadas por el algoritmo COA están definidos entre 0 y 1 respectivamente, cada variable debe ser transformada hacia sus nuevos límites requeridos por los relés. Los nuevos límites para cada variable de decisión son mostrados en la Tabla 2.11.

Tabla 2.11. Nuevos límites para la normalización de las variables de decisión.

Parámetros	Límite Inferior	Límite Superior
TMS	0,05	1,0
IpickUp	ILoadFlow_Max_Dem	1Fault_Rf_3_Min_Dem
Tipo de Curva	0	3

Donde: **ILoadFlow_Max_Dem** es la corriente de flujo de potencia dada en demanda máxima.

1Fault_Rf_3_Min_Dem es la corriente de falla monofásica con resistencia de falla de 3 ohmios dada a demanda mínima en el nodo 633.

Los nuevos límites de las variables se mencionan a continuación. Para el TMS se definen los mismos límites del software de PowerFactory dentro de su configuración de 0,05 a 1,0. Los límites de la corriente pickup son definidos por los criterios de optimización, donde se tiene como límite inferior a la corriente de flujo de potencia vista por el relé en demanda máxima, mientras que para el límite superior se asume que la corriente de falla mínima se da en la corriente de falla monofásica con una resistencia de falla de 3 ohmios a demanda mínima. Finalmente, el tipo de curva se encuentra definido con límites de 0 y 3, considerando que este parámetro toma valores enteros, en donde, cada valor entero es un índice que representa un tipo de curva del estándar IEC mostrado en la Tabla 2.12.

Tabla 2.12. Valores enteros que definen el tipo de curva IEC dentro del algoritmo COA.

Tipo de característica	Índice
Largo tiempo inverso	3
Extremadamente inverso	2
Muy inverso	1
Normalmente inverso	0

A manera de ejemplo, se presenta el cambio de límites de la variable de decisión del tipo de curva, de modo que, en el algoritmo COA se obtienen valores que se encuentran comprendidos entre un valor mínimo (VarLow) y un valor máximo (VarHi), en donde anteriormente se ha definido límites de 0 y 1, respectivamente. Sin embargo, se requiere que la variable de decisión correspondiente al tipo de curva del relé de sobre corriente sea expresada con un número que se encuentre comprendido entre 0 y 3, como se puede ver en la Figura 2.6, para lo cual se aplica la Ecuación 2.9. De esta misma manera, se aplica el cambio de límites para las demás variables de decisión.

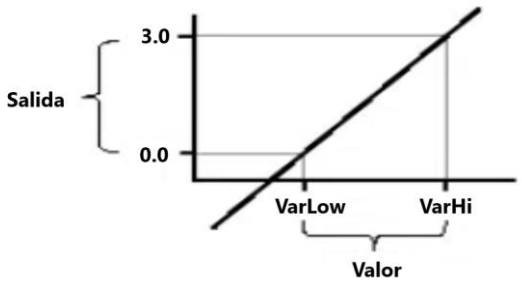


Figura 2.6. Normalización de la variable correspondiente al tipo de curva [16].

Posterior a que la variable del tipo de curva ha sido normalizada dentro de sus nuevos límites, su valor numérico debe ser redondeado al entero más cercano para que sea

reconocido como índice y se pueda identificar el tipo de curva que se ha considerado en la evaluación de la función objetivo para el relé correspondiente. Con el índice del tipo de curva se establecen los valores de α y n mostrados en la Tabla 1.1, para calcular el tiempo de operación de los relés según la Ecuación 1.4.

Restricciones implementadas

Las restricciones principales que se aplican son los límites de las variables de decisión presentadas en la Tabla 2.11 para ajustar los n relés y la restricción correspondiente a la coordinación entre relés, mostrada en la Ecuación 1.8. Mediante la codificación de restricciones dentro de la función objetivo, se establecen límites para los valores que pueden tomar las variables de la combinación. Se puede aumentar restricciones, según requiera el algoritmo COA para obtener valores coherentes de las variables de decisión que resuelven el problema de optimización.

Los parámetros no pueden tomar cualquier valor para resolver el problema de coordinación óptima de relés de sobre corriente instantáneos (51). Dado que, aun cuando al ajustar los relés 51 con parámetros que están fuera de sus límites, pueden disminuir el tiempo de operación, estos ajustes en la práctica no se pueden emplear, porque conlleva a un mal funcionamiento del sistema de protecciones o el relé no admite estos ajustes.

La Tabla 2.13 muestra los valores de las corrientes calculadas para restringir el ajuste de la corriente pickup de cada relé en el sistema de prueba de 13 nodos de la IEEE. Estos valores de corrientes son tomadas de la ejecución de la falla en el sistema de prueba modelado en PowerFactory e ingresados en el código programado en Python para introducir las restricciones en la función objetivo.

Tabla 2.13. Corrientes usadas para restringir los valores de ajuste de la corriente pickup de cada relé.

ILoadFlow_Max_Dem [A]		**1Fault_Rf_3_Min_Dem [A]**	
Relé 51_RG60	**Relé 51_632**	**Relé 51_RG60**	**Relé 51_632**
586,61	80,99	1197,65	797,07

Para cumplir la coordinación entre relés, el CTI entre curvas características de los n relés debe tener una diferencia de tiempos de alrededor de 300 [ms]. Para este trabajo, se establece como restricción que el CTI debe ser mayor a 300 [ms] y no debe superar los 350 [ms]. De esta forma se tienen un intervalo de tiempo de coordinación (CTI) adecuado

para evitar que entre relés operen antes de tiempo o que operen después de mucho tiempo.

En la Tabla 2.11 se considera como límite inferior de la restricción para la corriente pickup a la corriente de flujo dada en demanda máxima, sin embargo, se debe considerar el crecimiento de demanda. En este trabajo se asume un crecimiento del 50% de la demanda en el sistema de prueba, siendo este un sistema de distribución. Por lo tanto, se debe establecer como límite inferior para cada relé a la corriente calculada del flujo de potencia en demanda máxima multiplicada por 1,5 veces su valor. De esta forma se garantiza que, al haber un crecimiento menor al 50 % de la demanda actual, los relés que se han ajustado no operen en condiciones normales del sistema.

Las restricciones deben ser implementadas dentro de la función objetivo que se encuentra codificada en Python. Cada término de violación o peso del correspondiente límite de cada restricción se calcula con la Ecuación 2.10 y se suma al tiempo de operación total. El término de violación incrementa si la variable sale del límite de la correspondiente restricción y es cero si está dentro de su límite.

$$Término\_de\_Viololación = \frac{(G(X) - A + |G(X) - A|)}{2}$$

Ecuación 2.10. Ecuación del término de violación [17].

Para calcular el término de violación de cada límite para cada restricción usando la Ecuación 2.10, se define la variable de la restricción como $G(X)$ y al límite de dicha restricción como A. Para definir si dicho límite corresponde al límite superior o al límite inferior se lo realiza con el signo que se pone a la variable de la restricción y el límite de la restricción. Para restringir un límite inferior, el signo de $G(X)$ y de A son negativos, mientras que para restringir un límite superior el signo de $G(X)$ y de A son positivos [17].

A continuación, a manera de ejemplo se calculan los términos de violación correspondientes al límite superior e inferior para restringir los valores que puede tomar el Dial o TMS del primer relé. Se realiza el mismo procedimiento para restringir las demás variables de decisión con sus respectivos límites.

Si: TMS1 > 0,05 entonces

$$G_{TSM\_1} = -TSM1$$

$$A_{TSM\_1} = -0{,}05$$

$$TV_1 = \frac{((G_{TSM_1} - A_{TSM_1}) + |G_{TSM\_1} - A_{TSM\_1}|)}{2}$$

Si: TMS1 < 1,0 entonces

$$G_{TSM\_2} = TSM1$$

$$A_{TSM\_2} = 1{,}0$$

$$TV_2 = \frac{((G_{TSM_2} - A_{TSM_2}) + |G_{TSM\_2} - A_{TSM\_2}|)}{2}$$

Los términos de violación de las restricciones son sumados a los tiempos de operación total en la función objetivo. La función objetivo tiene que minimizar el valor de los términos de violación de las restricciones y los tiempos de operación de los relés para obtener resultados óptimos. Cada término de violación puede ser multiplicado por un término muy alto para que el algoritmo optimice a valores de los términos de violación que se acerquen a cero. El valor probado para multiplicar a los términos es de 10^5, sin embargo, se puede aumentar o disminuir este valor según convenga en los resultados que obtiene el algoritmo [17].

A continuación, mediante la Ecuación 2.11 se presenta la función objetivo del tiempo de operación total con los términos de violación de las restricciones en forma de pesos para optimizar dos relés 51.

$$FO = min\left(\frac{(TSM1 * \alpha 1)}{\left(\frac{Isc1}{Ipickup1}\right)^{n1} - 1} + \frac{(TSM2 * \alpha 2)}{\left(\frac{Isc2}{Ipickup2}\right)^{n2} - 1} + (TV_1 + TV_2 + \cdots + TV_i) * 10e5\right)$$

Ecuación 2.11. Función objetivo con las restricciones para dos relés.

Donde TV_i representa el término de violación *i* de la función objetivo.

La función objetivo mostrada en la Ecuación 2.11 se ha implementado en el algoritmo COA para evaluar las combinaciones de ajustes de relés 51. Las variables transformadas a sus nuevos límites consideran la restricción inferior y superior mediante los términos de violación correspondientes. Para calcular el tiempo de operación de cada relé en la función objetivo, se obtiene la corriente de prueba (I_{Sc}) del sistema eléctrico y las variables de ajuste de relés 51 del algoritmo COA.

La corriente de prueba seleccionada para optimizar la coordinación entre relés 51 es la corriente de falla monofásica sin impedancia de falla a mínima demanda. Se selecciona este tipo de falla por ser una falla que comúnmente suele darse en sistemas de distribución. Se realiza el cálculo de la corriente de prueba en el nodo 633 del alimentador LOHL632- 633 del sistema de prueba, por ser el punto más lejano de la zona protegida. Al considerar que el sistema de prueba de 13 nodos de la IEEE es un sistema radial, las corrientes que circulan aguas arriba son mayores que las corrientes que circulan aguas abajo, cumpliendo la ley de corrientes de Kirchoff. Por tal razón, la corriente de prueba vista por el relé _RG60 es mayor a la corriente vista por el relé _632.

Se consigue el ajuste óptimo para los relés 51 del sistema eléctrico con el algoritmo COA, cuando se incluye a la función objetivo dentro de su codificación en Python. En el Anexo IV se muestra la función objetivo con sus restricciones. El algoritmo COA se ejecuta desde el mismo Python de manera independiente al sistema de prueba modelado en PowerFactory. Los parámetros óptimos obtenidos se encuentran entre los límites VarLow y VarHi, se les transforma hacia sus nuevos límites para ajustar los relés 51.

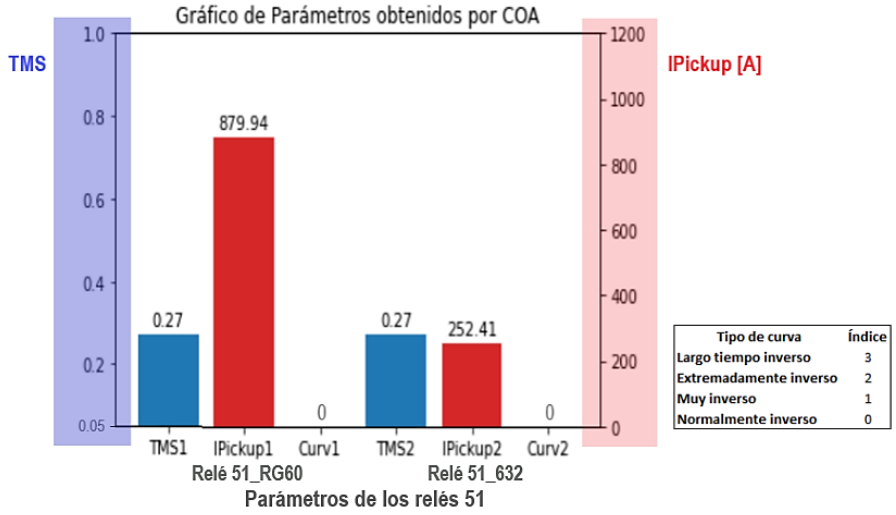


Figura 2.7. Resultados de los parámetros obtenidos del algoritmo COA.

Para ajustar los dos relés 51 incorporados en el sistema de prueba y resolver el problema de coordinación óptima entre estos relés, se emplean los resultados obtenidos por el algoritmo COA mostrados en la Figura 2.7. Se debe tener en cuenta que los parámetros de las corrientes pickup se ajustan en el sistema de prueba con valores en por unidad, por lo que, en la Tabla 2.14 se presentan los ajustes óptimos ingresados en los relés del sistema eléctrico.

Tabla 2.14. Ajustes óptimos de ambos relés cumpliendo restricciones.

Relés	Relé 51_RG60			Relé 51_632		
Best Params	TSM1	Ipickup1 [pu]	Tipo de Curva1	TSM2	Ipickup2 [pu]	Tipo de Curva2
	0,2742	0,8799	0	0,2684	0,5048	0
Cost	1,7521					

Donde Best Params representa el valor óptimo de las variables de decisión y Cost representa la evaluación de la función objetivo obtenida del hábitat de parámetros óptimos.

En la Figura 2.8 se aprecia que, al realizar la evaluación de los parámetros óptimos con la corriente de prueba mediante la función objetivo, se obtiene un tiempo total de operación de 1,7521 segundos. Los ajustes óptimos se consiguen a las ciento un iteraciones en un tiempo de ejecución de 18,67 segundos.

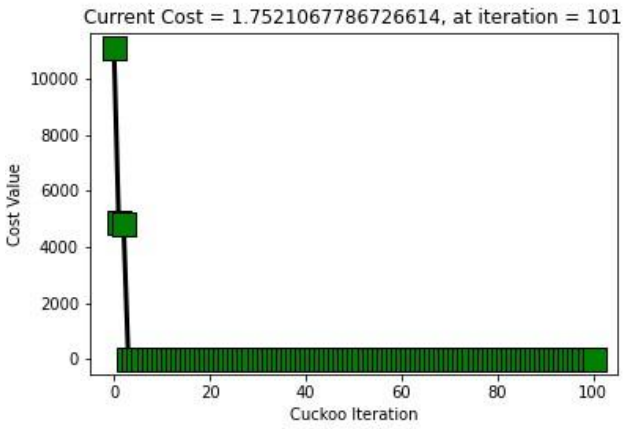


Figura 2.8. Obtención de los parámetros óptimos con el algoritmo COA a las 101 iteraciones.

Donde Current Cost representa la evaluación de la función objetivo obtenida del hábitat de parámetros óptimos e iterations es el número de iteraciones en las que el algoritmo consigue los ajustes óptimos.

Se calcula la corriente de prueba en el sistema eléctrico al tener ajustados los dos relés 51 con los parámetros óptimos. Se puede ver en la Figura 2.9 que los relés 51_RG60 y 51_633 operan a los 1,107 [s] y 0,634 [s], respectivamente. Al sumar los tiempos de operación se obtiene un valor de 1,741 [s], se aproxima al tiempo de operación total calculado por la función objetivo.

La diferencia correspondiente a la sumatoria de los tiempos de operación se atribuye al número de decimales que se puede introducir en los parámetros de ajuste de los relés 51 en el software de PowerFactory, debido a que, Python calcula los tiempos de operación con todos los decimales mediante la Ecuación 1.4, mientras que el software de PowerFactory solo permite introducir dos decimales.

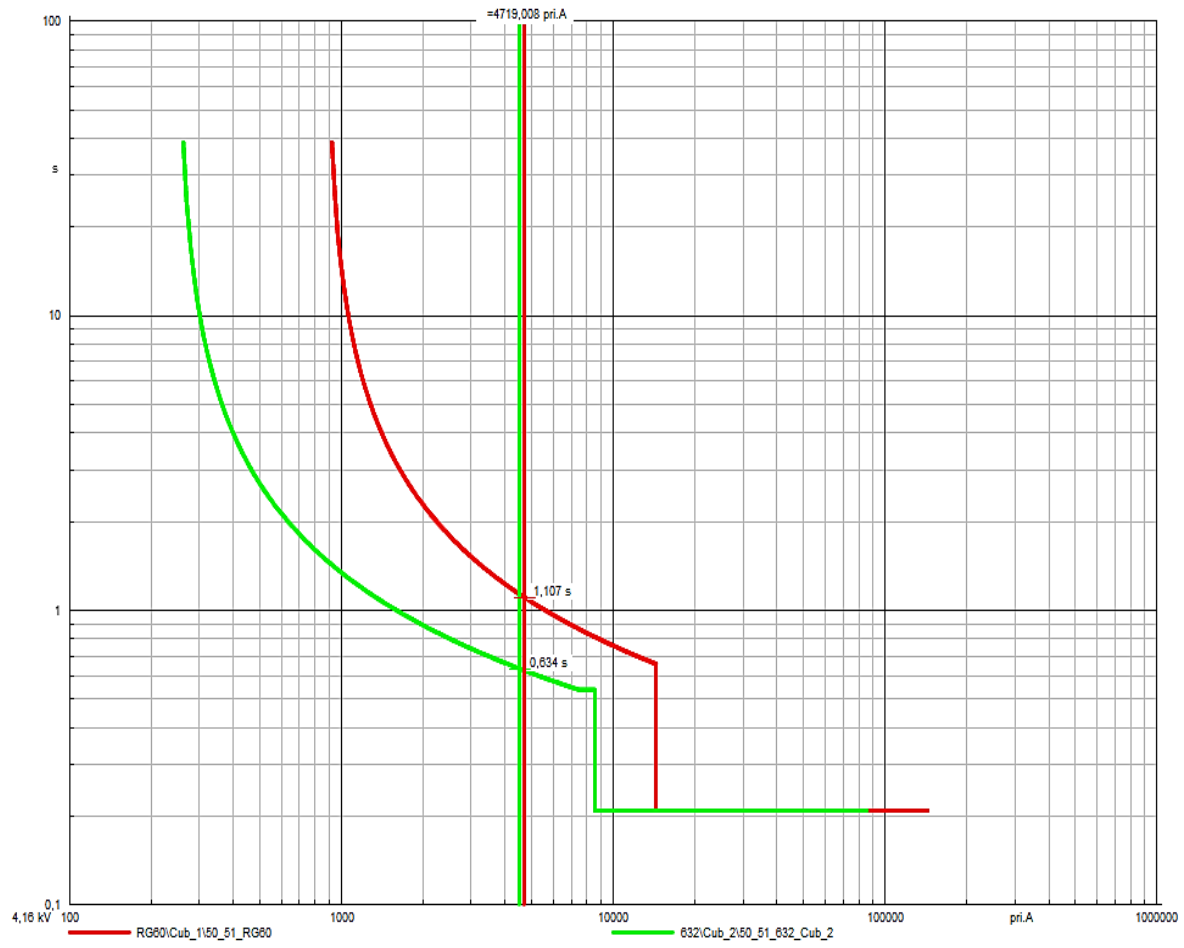


Figura 2.9. Cálculo de la corriente de prueba en el sistema eléctrico con el ajuste óptimos de los relés 51.

Al verificar si se cumplen las restricciones, la corriente pickup y el Dial de ambos relés 51 se encuentra dentro de los respectivos límites. Sin embargo, no logra cumplir con la restricción del CTI entre curvas características, el cual debe ser mayor o igual a los 300 [ms]. Al calcular los tiempos de operación de los relés 51_RG60 y 51_632, considerando la falla más severa en el alimentador se obtienen los tiempos de 798 [ms] y 537 [ms] respectivamente, como se observa en la Figura 2.10. El CTI entre las curvas características de estos relés es la diferencia entre estos tiempos, que resulta ser de 261 [ms].

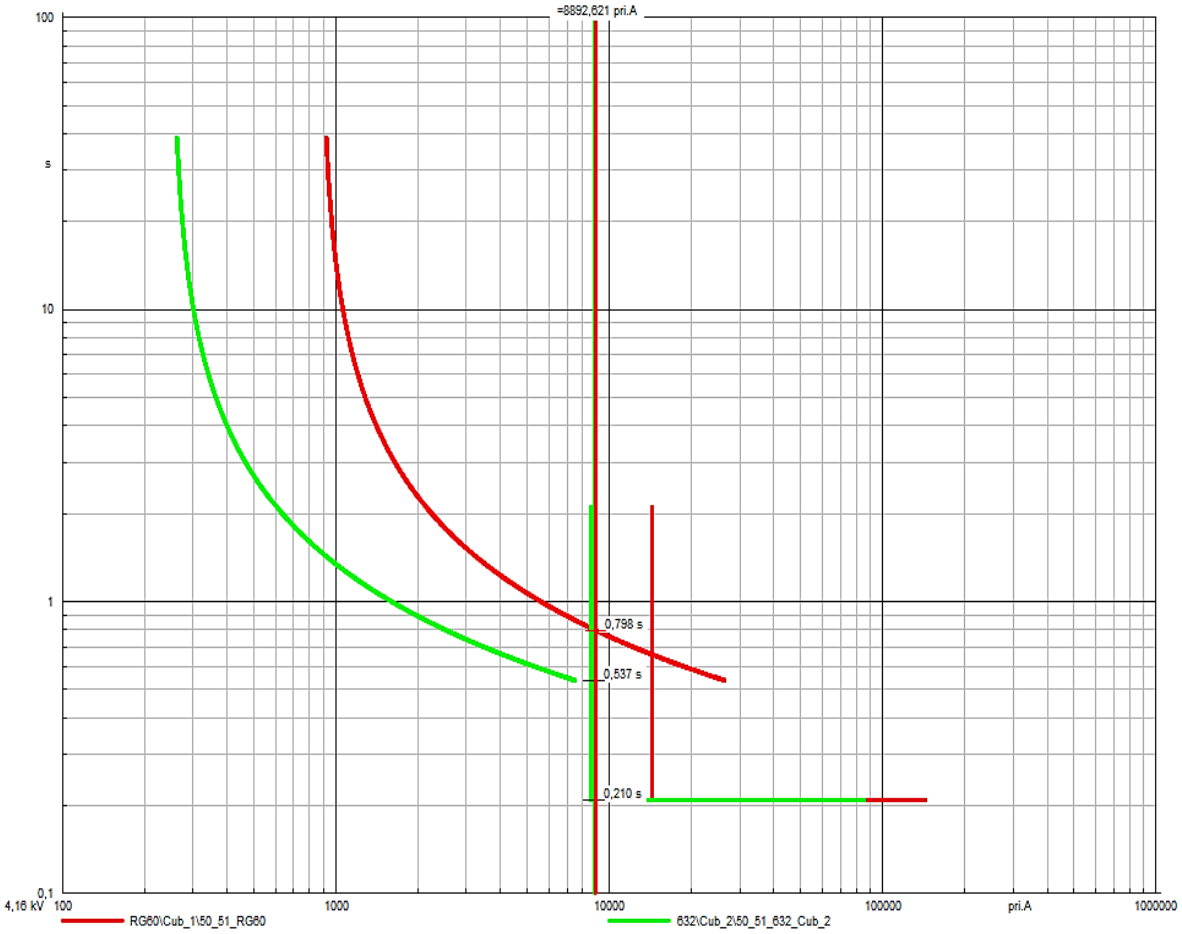


Figura 2.10. Cálculo de falla para verificar cumplimiento del CTI entre curvas características.

Esta diferencia se atribuye a la ecuación de la curva característica IEC por no ser considerada como una función por partes. No obstante, los valores optimizados de las variables de decisión resultan ser coherentes, esto indica el buen funcionamiento del

código de Python realizado. A pesar de no tener un CTI mayor o igual a 300 [ms], el ajuste obtenido es correcto, dado que el CTI es aceptable en rangos de 0,2 a 0,5 segundos según [10].

El algoritmo COA mejora el ajuste de los parámetros de los relés sin infringir la condición del CTI al realizar la vinculación del algoritmo que se encuentra codificado en Python con el sistema eléctrico de prueba que se encuentra modelado en PowerFactory. Se consigue cumplir con todas las restricciones para asegurar el correcto funcionamiento del sistema eléctrico. Se obtienen los tiempos de operación de los cálculos realizados en el sistema de prueba con PowerFactory directamente. Con esto, no se tiene el inconveniente de considerar la función que representa la curva característica del estándar IEC como una función por partes.

2.5 Vinculación del algoritmo de optimización (En Python) con el sistema eléctrico de prueba (En PowerFactory)

Se dispone de los relés de sobre corriente en el sistema de prueba de 13 nodos de la IEEE, que se encuentra modelado en el software de PowerFactory, y la codificación del algoritmo con la función objetivo para minimizar los tiempos de operación de los relés 51. El algoritmo COA es capaz de resolver el problema de coordinación óptima de relés 51 para el sistema de prueba. Dicho algoritmo es independiente y no requiere intercambiar información con el sistema de prueba modelado en PowerFactory, dado que en el algoritmo de optimización se ingresan las corrientes obtenidas en el sistema de prueba y se obtienen los tiempos de operación mediante la Ecuación 1.4.

Al aplicar la Ecuación 1.4 para el cálculo del tiempo de operación de los relés 51, los resultados de los tiempos de operación obtenidos corresponden a una curva característica de la IEC la cual es una función por partes, como se evidencia en la Figura 2.11. La Ecuación 1.4 corresponde a la primera parte de la curva característica de la IEC. En los casos donde la corriente de prueba cae en la parte 2, el tiempo de operación del relé es una constante.

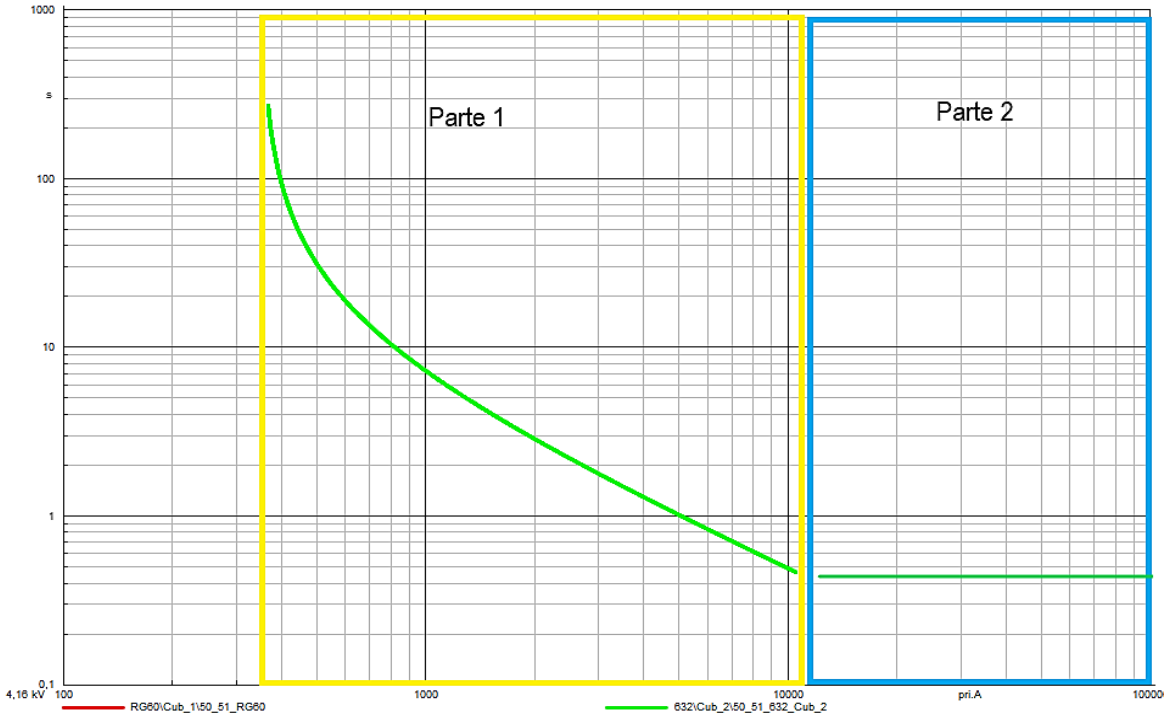


Figura 2.11. Función por partes de la curva característica IEC.

Se realiza de manera automática el cálculo de los tiempos de operación de cada relé a través de la vinculación del algoritmo COA (Python) y el sistema de prueba (PowerFactory). Esta vinculación permite el intercambio de información de manera versátil, para lo cual, se ha desarrollado una aplicación que permite manipular los objetos de PowerFactory utilizando Python [18]. De esta manera, se consigue calcular las corrientes y los tiempos de operación directamente desde el sistema de prueba, y esta información disponerla también en Python. Al realizar cambios en el sistema de prueba, las corrientes son actualizadas automáticamente en la codificación del algoritmo de optimización.

Mediante la Tabla 2.15 se presentan los principales comandos para acceder desde Python a PowerFactory.

Tabla 2.15. Principales comandos para acceder a PowerFactory desde Python.

Comando	Función
import powerfactory as pf	Importar librería de PowerFactory
pf.GetApplication()	Aplicación de acceso a pf
GetCurrentScript()	Acceso del script

Barrido de fallas en alimentador más alejado

Se realiza el barrido de fallas para validar y cuantificar la mejora que se obtiene al calibrar los relés 51 con el ajuste óptimo sobre todo el alimentador protegido más alejado LOHL632-633, tomando como referencia el ajuste base. La falla considerada para ejecutar el barrido es la monofásica sin impedancia de falla a demanda mínima, por ser un tipo de falla común en sistemas de distribución. El barrido calcula la falla en todos los puntos del alimentador que están separados entre sí por una distancia que corresponde al 10% de su longitud total, tanto con el ajuste base como con el ajuste óptimo de los relés 51. Se considera como punto de referencia el nodo 632 para determinar la distancia del punto de referencia hacia el punto de falla en el sistema eléctrico.

Mediante la vinculación del sistema de prueba con el algoritmo de optimización, se automatizan los cálculos del barrido de fallas. La codificación realizada en Python para el barrido de fallas que vincula ambos programas permite obtener directamente los tiempos de operación de los dos relés 51 para todos los puntos de falla calculados para ambos casos de ajuste. Los tiempos de operación obtenidos al calibrar los relés 51 con el ajuste tradicional y el ajuste óptimo son almacenados en una tabla de Excel, para ser comparados. En el Anexo V se muestra la codificación en Python del barrido de fallas.

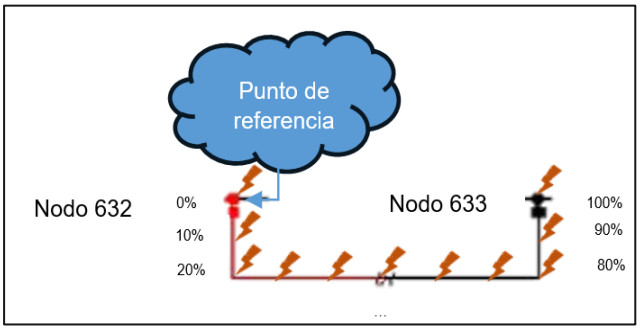


Figura 2.12. Barrido de cortocircuitos en el alimentador LOHL632-633.

Procedimiento para la optimización de la coordinación de los relés 51 entre Python - PowerFactory

La Figura 2.13 muestra el procedimiento de la optimización que se obtiene al vincular el sistema de prueba con el algoritmo de optimización. Este procedimiento optimiza las variables de decisión a valores que cumplen con todas las restricciones. Mejora la evaluación de los ajustes óptimos conseguidos con el algoritmo COA independiente del sistema de prueba obtenido en el numeral 2.4 gracias al intercambio de información por

la vinculación entre ambos programas. La vinculación permite que el algoritmo COA reciba los valores de los cálculos realizados de corrientes y tiempos de operación de los relés 51 directamente del sistema eléctrico.

Inicialmente se ejecuta el barrido de cortocircuitos en el alimentador para obtener los tiempos de operación de los relés 51 al estar ajustados con el criterio tradicional. Después, se calcula en el sistema de prueba las corrientes que definen los límites de la restricción de corriente pickup y se transmite automáticamente los cálculos a la función objetivo para establecer esta restricción. Considerando que estas corrientes dependen del sistema de prueba, las corrientes permanecen constantes porque no se modifica el sistema de prueba durante la optimización, por esto se calculan una sola vez. Seguido a esto, el algoritmo COA inicializa las combinaciones de las variables de decisión en una lista de Python.

Se continúa con el cálculo de los tiempos de operación de los relés 51 en el sistema de prueba a cada combinación ante la falla más severa en el alimentador y se transmite automáticamente al algoritmo, para determinar el peso dado por la restricción del CTI que se suma al tiempo total de operación. Posterior a esto, se calculan los tiempos de operación de los relés 51 ante la falla de prueba en el sistema de eléctrico y se transfieren automáticamente estos cálculos mediante el vínculo al algoritmo de optimización para ser sumados con los pesos en la función objetivo. El algoritmo COA ordena de mejor a peor combinación según su resultado de evaluación, descarta a las peores combinaciones y hace que las demás combinaciones se muevan hacia la mejor posición de cada iteración. El procedimiento de este párrafo se repite varias iteraciones hasta converger en resultados óptimos.

Al tener el ajuste óptimo de los relés 51, se realiza un barrido de cortocircuitos en el alimentador para comparar con los resultados obtenidos con el criterio de ajuste base. Para mostrar el mejor tiempo de operación hallado por el algoritmo COA, se calcula la corriente de prueba en PowerFactory, de esta manera se observa la disminución del tiempo de operación de los dos relés de forma gráfica.

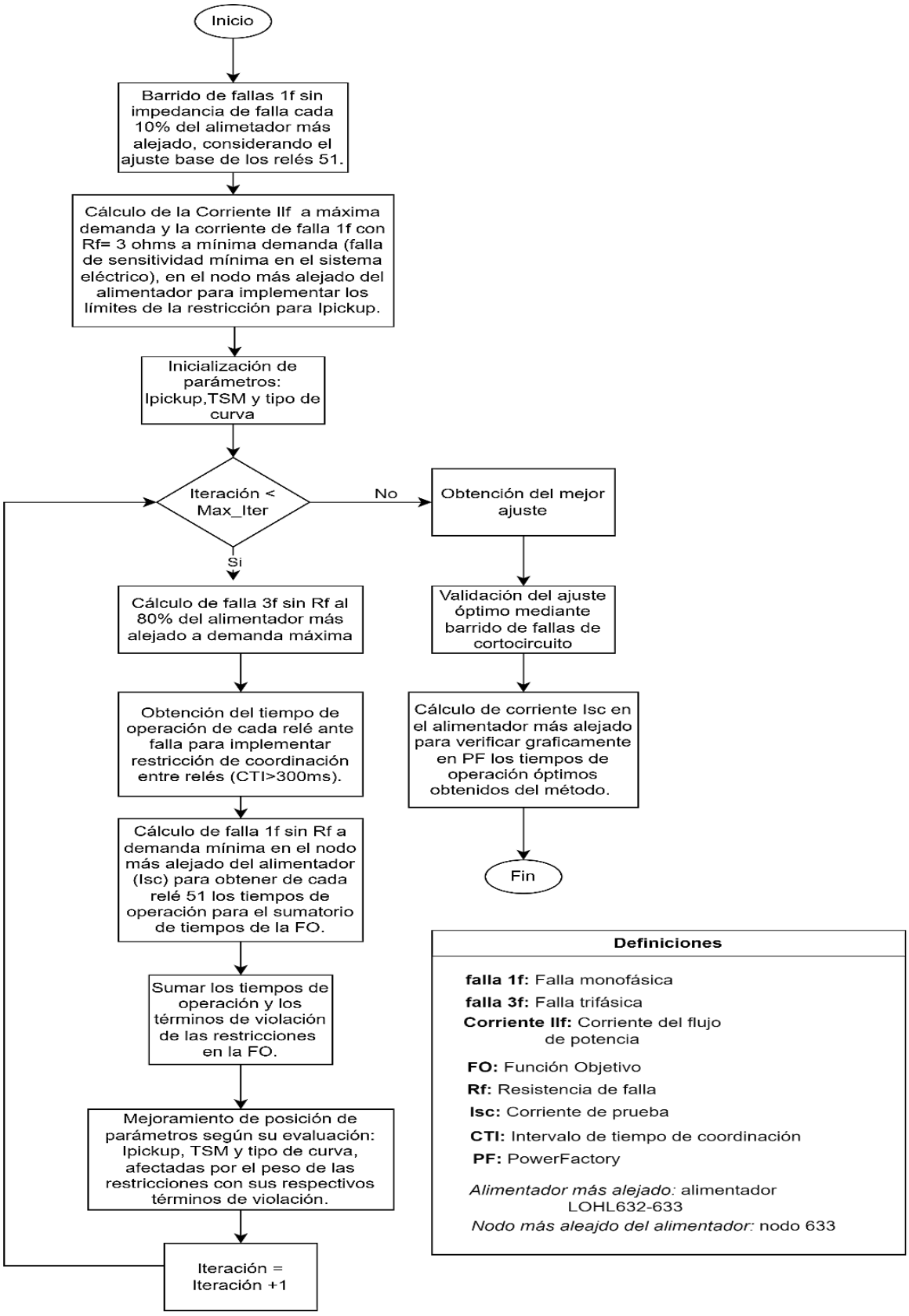


Figura 2.13. Procedimiento para la optimización de la coordinación de relés 51 entre Python - PowerFactory.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Se obtienen resultados satisfactorios en el ajuste de los relés 51 del sistema de prueba de 13 nodos de la IEEE al implementar la optimización. El ajuste óptimo de los relés 51 se obtiene del procedimiento de la optimización propuesto en este trabajo, basada en el algoritmo COA con la vinculación al sistema eléctrico. Se comprueba la reducción de los tiempos de operación al ser comparados antes y después de realizar la optimización. Además, los ajustes óptimos cumplen con la operación coordinada entre relés 51 ante una falla de cortocircuito en la zona de protección.

El criterio base de ajuste, obtenido por la literatura, se establece como punto de comparación con el procedimiento de la optimización. Este criterio sirve como punto de comparación por ser el más empleado para el ajuste y la coordinación de las protecciones. Mediante la comparación, se determinan las ventajas brindadas por la optimización de la coordinación de los relés 51 en el sistema de prueba.

Se considera que las corrientes de falla dependen de las condiciones específicas de la red y de la naturaleza de la falla. En el sistema de prueba de 13 nodos de la IEEE, al tratarse de un sistema de distribución eléctrico, las fallas más comunes resultan ser de cortocircuito fase a tierra. Este tipo de falla se puede dar típicamente por el aislamiento defectuoso, daños mecánicos de conductores o equipos, desgaste y envejecimiento de cables y equipos, condiciones climáticas adversas o errores de instalación o mantenimiento de los equipos. Con los ajustes óptimos, se protege al sistema de prueba de este tipo de fallas con tiempos de operación mínimos y manteniendo la coordinación entre relés, con lo que se consigue mejorar la confiabilidad del sistema eléctrico de prueba.

Se optimizan los parámetros mediante la corriente de prueba, que se trata de una falla monofásica sin impedancia de falla a demanda mínima en el nodo 633 del alimentador LOHL632-633. Siendo esta, una corriente poco sensible que permite realizar la búsqueda de parámetros que hacen operar a los relés 51 rápidamente y de forma coordinada ante este tipo de fallas. El sistema de prueba al ser un sistema radial debe detectar rápidamente las fallas de los alimentadores aguas abajo, para que se abra el elemento fallido y el resto del sistema pueda seguir suministrando energía. Por este motivo, se

prioriza la optimización de los tiempos de operación de los relés 51 en el alimentador LOHL632-633, al ser este un alimentador ramal del alimentador LOHL650-632.

En este caso, se consigue disminuir los tiempos de operación a lo largo del alimentador LOHL632-633, que es donde operan los relés 51_RG60 y 51_632 por ser el alimentador más alejado. Además, en este alimentador es donde ambos relés participan, por tal motivo, se puede realizar la búsqueda de ajustes óptimos para la coordinacion óptima entre relés. Las corrientes elevadas son despejadas mediante las protecciones instantáneas, considerando una protección hasta del 80% del alimentador a corriente de falla máxima.

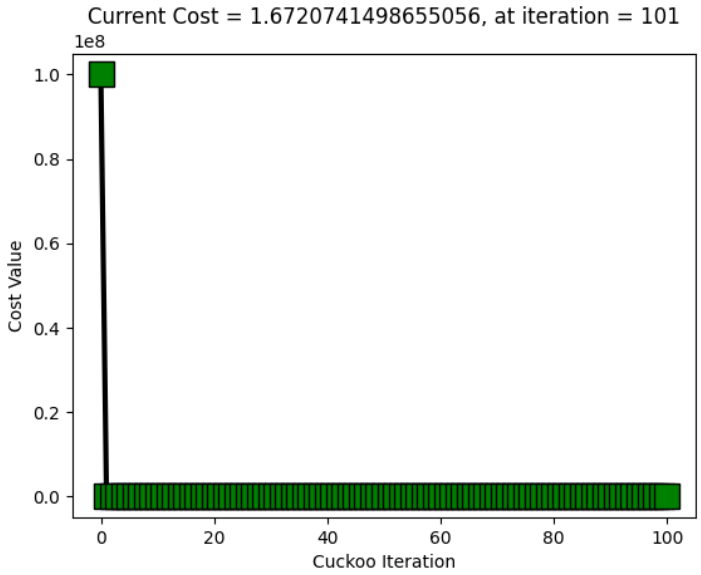


Figura 3.1. Reducción de tiempos de operación para relés 51 con algoritmo COA en Python, por iteración.

Usando la corriente de prueba, el algoritmo COA consigue minimizar la sumatoria de tiempos de operación de los relés 51 mediante la función objetivo a un tiempo total aproximado de 1,672 [s] con ciento un iteraciones, como se observa en la Figura 3.1. En la Tabla 3.1 se muestran los parámetros que consiguen la coordinación óptima de los relés 51_RG60 y 51_632.

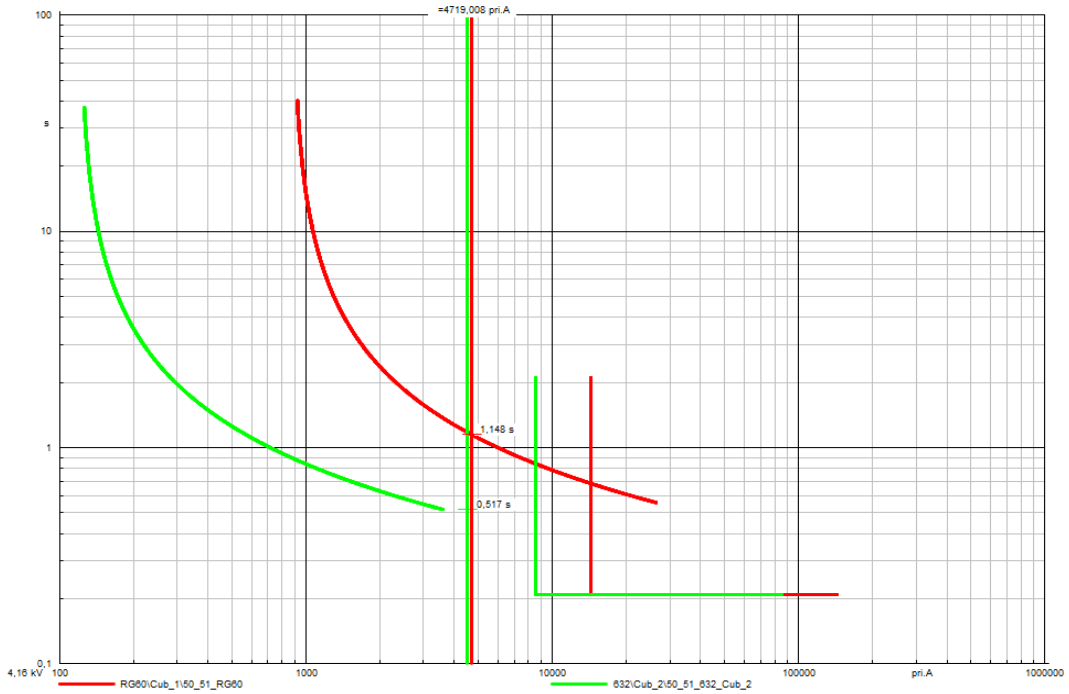
Tabla 3.1. Ajuste de relés 51_RG60 y 51_632 obtenidos con el procedimiento de la optimización basado en algoritmo COA.

Relé	Ajuste	Valor	Unidad
Relé 51_632	TimeDial	0,26	-
	Pickup Current	0,24	pu
	Characteristic	IEC class A (Standard Inverse)	-
	Tiempo	0,517	s
Relé 51_RG60	TimeDial	0,28	-
	Pickup Current	0,88	pu
	Characteristic	IEC class A (Standard Inverse)	-
	Tiempo	0,828	s

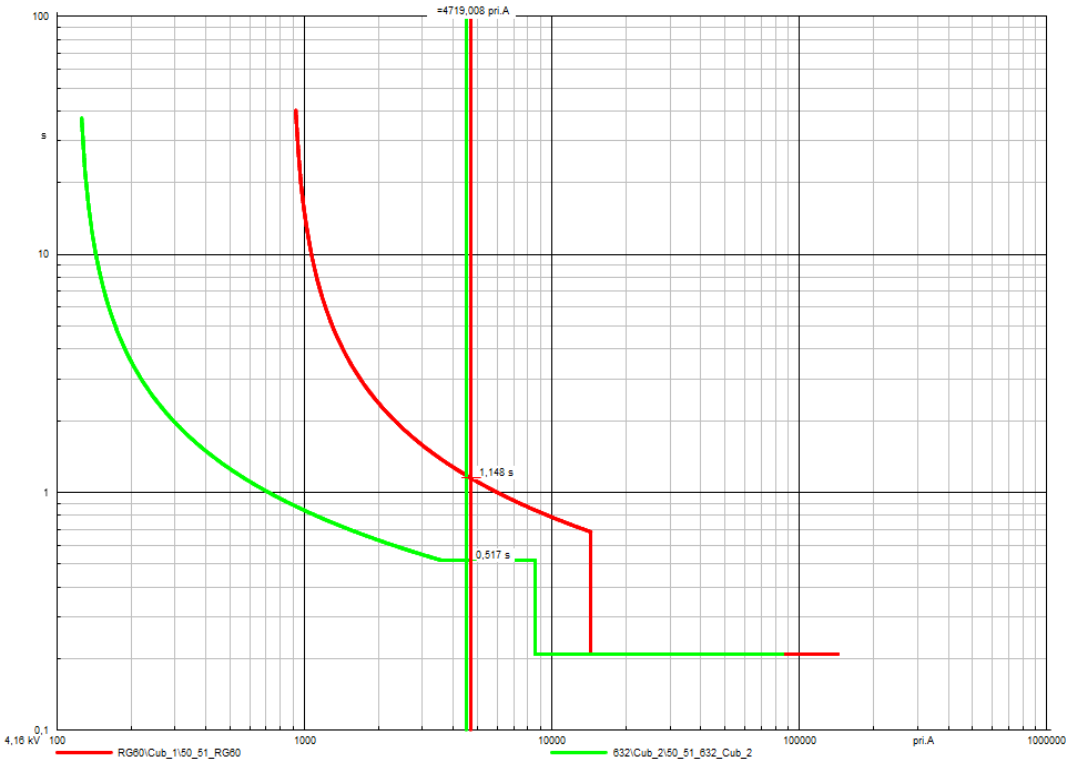
Según lo señalado en el numeral 2.5, la curva característica del relé 51 según la IEC es una función compuesta por dos partes. PowerFactory muestra en sus gráficas únicamente las curvas obtenidas por la parte 1 de la función compuesta de las curvas IEC cuando se presenta la curva instantánea separada de la temporizada. Sin embargo, al juntar la curva característica instantánea con la temporizada de la IEC se puede observar la curva representada por la Ecuación 1.4 y parte de la curva representada por la constante, correspondientes a parte 1 y parte 2 que se señala en el numeral 2.5.

En la Figura 3.2 se observan las curvas características de los relés 51_RG60 y 51_632 con color rojo y verde respectivamente. Las curvas características han sido ajustadas con los parámetros óptimos en el software de PowerFactory. La gráfica del literal (a) muestra la curva temporizada sin tomar en cuenta la parte 2 de la función compuesta. Mientras que, en la gráfica del literal (b), con la curva temporizada e instantánea juntas se logra apreciar la parte 2 de la función compuesta del relé 51, hasta que pasa a ser protección de respaldo del relé 50. Por lo que se logra apreciar de mejor manera el comportamiento de la parte 2 de la función compuesta en el relé 51.

Se muestran los tiempos de operación de 1,148 [s] y 0,517 [s] correspondiente a cada relé al calcular la corriente de prueba, obteniendo un tiempo total de 1,665 [s]. Este tiempo se aproxima a la sumatoria de tiempos de operación obtenido por el algoritmo COA (Python) de 1,672 [s], mostrado anteriormente en la Figura 3.1. Sin embargo, la diferencia en el cálculo del tiempo total se atribuye al número de decimales (dos decimales) que se puede introducir en el PowerFactory. Dado que, al no introducir todos los decimales de las variables con los que trabaja Python, los tiempos de operación de los relés obtenidos en el sistema de prueba presentan una pequeña variación.



a)



b)

Figura 3.2. Curvas características ajustadas con los parámetros óptimos de los relés 51_RG60 y 51_632 632 al calcular la corriente de prueba. a) Curva instantánea separada a la curva temporizada, b) Curva instantánea junta a la curva temporizada.

Al calcular la corriente de prueba en el sistema eléctrico con el ajuste base de los relés 51_RG60 y 51_632, se presenta en la Figura 3.3 los tiempos de operación de 2,341 [s] y 1,130 [s], respectivamente. Resultando un tiempo total de operación de 3,471 [s]. Estos tiempos resultan ser mayores que los obtenidos con los ajustes óptimos. Se obtiene una mejora en los tiempos de operación al implementar el procedimiento de la optimización. Los tiempos de operación disminuyen de 2,341 a 1,148 [s] y de 1,130 a 0,517 [s], reduciendo el tiempo de operación en 1,193 [s] (reducción del 50,97%) y 0,61 [s] (reducción del 54,23 %) para cada relé.

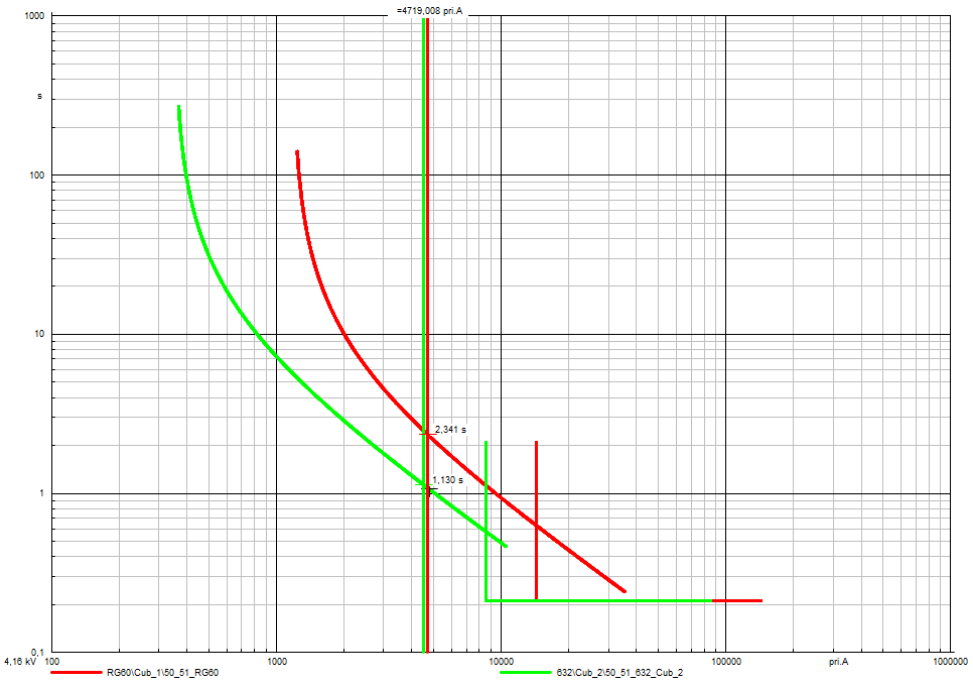


Figura 3.3. Curvas características ajustadas con el criterio base de los relés 51_RG60 y 51_632 al calcular la corriente de prueba.

Se comprueba que las restricciones establecidas no sean infringidas en los ajustes óptimos, por lo que se procede de la siguiente manera. Primeramente, se verifica que el CTI tenga un tiempo alrededor de 300 [ms]. Seguido a esto, se revisa que la corriente pickup no salga de los límites establecidos. Con el cumplimiento de las restricciones, se garantiza una operación adecuada de las protecciones de sobre corriente implementadas en el sistema de prueba.

Para comprobar que el CTI tiene un tiempo entre los 300 [ms], se calcula la falla más grave en la que actúa el relé 51_632 como protección principal para el alimentador LOHL632- 633 del sistema de prueba, para obtener los tiempos de operación. Se calcula

la diferencia de tiempos de operación entre relés 51 y se comprueba si cumple el CTI, como se indica mediante la Figura A6.1 en ANEXO VI. Los tiempos obtenidos resultan ser de 517 [ms] y 828 [ms] para los relés 51_632 y 51_RG60 respectivamente, de modo que, se obtiene un CTI de 311 [ms], con lo que se cumple la restricción de coordinación entre los relés 51.

Para revisar que la corriente pickup no salga de los límites se calculan las corrientes seleccionadas como límites. Las corrientes obtenidas del flujo de potencia a demanda máxima son tomadas como límite inferior para el ajuste de la corriente pickup de cada relé. Estas corrientes son vistas por los relés 51_632 y 51_RG60 con un valor de 80,99 [A] y 586,61 [A], respectivamente. Las corrientes tomadas como límites superiores son obtenidas por el cálculo de una corriente monofásica con resistencia de falla de 3 ohmios a demanda mínima en el nodo 633, cuyos valores son 797,07 [A] y 1197,65 [A] para cada relé.

Al tener un ajuste óptimo de corriente pickup de 120 [A] y 880 [A] para cada relé, se puede verificar que se encuentra dentro de los límites establecidos. Además, los valores de la corriente pickup encontrados también aseguran la no operación de los relés 51 ante un crecimiento de hasta aproximadamente un 50% de su demanda actual. Mediante la Figura A6.2 y la Figura A6.3 ubicadas en el ANEXO VI, se demuestra gráficamente que el ajuste obtenido para cada relé 51 no sale de su respectivo límite inferior ni superior.

Para verificar la operación adecuada de las protecciones, se realizar el cálculo de una corriente trifásica sin impedancia de falla a demanda máxima al 10% del alimentador. Se observa mediante la Figura 3.4 que el relé 50_632 opera a los 210 [ms] como protección principal, a los 517 [ms] opera el relé 51_632 como primera protección de respaldo y a los 762 [ms] actúa el relé 51_RG60 como segunda protección de respaldo. Los tiempos de operación obtenidos corroboran la operación adecuada de los relés implementados en el sistema de prueba ajustados con los parámetros óptimos.

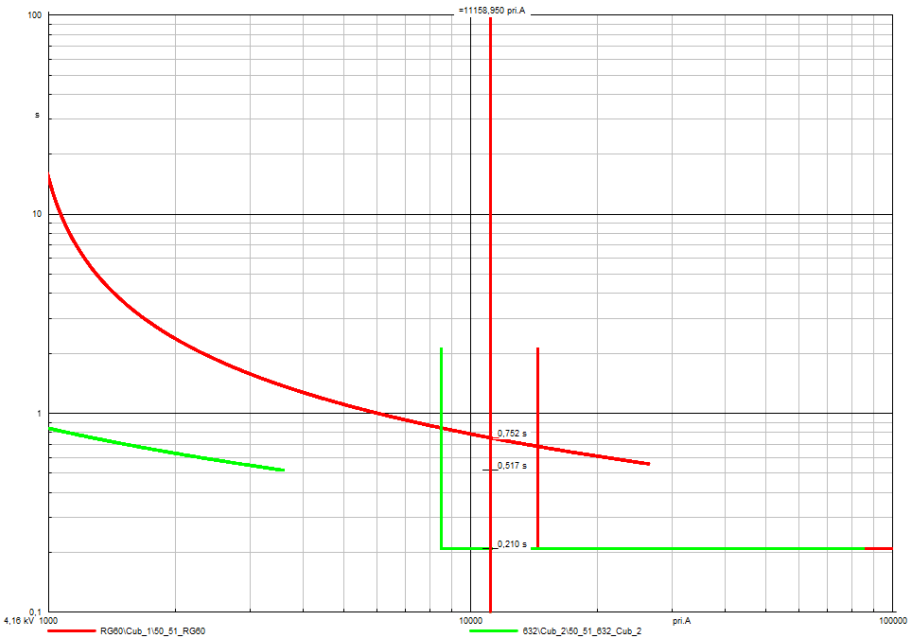


Figura 3.4. Cálculo de corriente de falla trifásica sin impedancia de falla a demanda máxima al 10% del alimentador LOHL632-633.

Se toma la siguiente consideración para determinar los beneficios alcanzados con los parámetros optimizados de los relés 51. Los relés 50_RG60 y 50_632 son los encargados de operar de forma instantánea las fallas que presentan corrientes muy altas, en donde, los relés 51 actúan como protección de respaldo. Por esta razón, se requiere comprobar la disminución de los tiempos de operación de los relés 51 ante corrientes de falla que hagan actuar a los relés 51 como protección principal.

A fin de presentar las mejoras obtenidas en cuanto a los tiempos de operación de los relés 51 ajustados con los parámetros óptimos, se presenta un barrido de fallas. Mediante la Tabla 3.2 se muestran los tiempos de operación de los relés 51 al realizar el barrido con la falla monofásica a mínima demanda sin impedancia de falla. Se realiza el barrido de fallas en el alimentador LOHL632-633 por ser el más lejano de la zona de protección. Los tiempos de operación del barrido han sido obtenidos para los dos relés 51 implementados en el sistema de prueba.

Se compara entre los tiempos de operación obtenidos con el criterio base y el procedimiento de la optimización propuesto. Al ajustar los relés 51 con los parámetros óptimos se disminuyen los tiempos de operación obtenidos del criterio base en todos los puntos del alimentador. En promedio, en todo el alimentador existe una reducción de

46,96% y 47,89% del tiempo de operación en los relés 51_RG60 y 51_632, respectivamente.

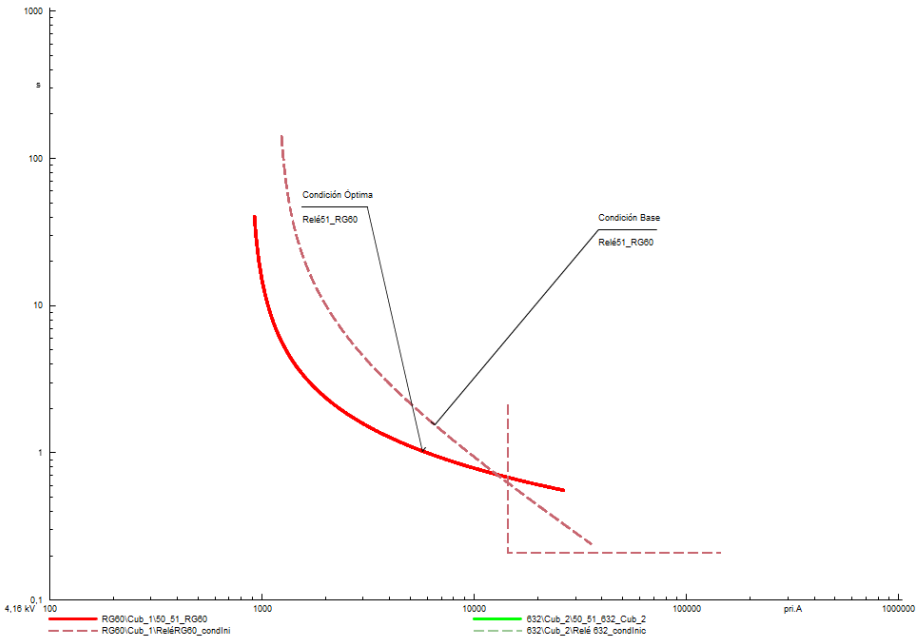
Tabla 3.2. Barrido de fallas realizado en el alimentador LOHL632-633.

Alimentador LOHL632-633	Ajuste criterio base		Ajuste criterio óptimo		Reducción del tiempo de operación	
Longitud nodo 632- punto de falla	Relé 51-RG60	Relé 51-632	Relé 51-RG60	Relé 51-632	Relé 51-RG60	Relé 51-632
[%]	[s]*	[s]*	[s]*	[s]*	[%]	[%]
0%	1,746218	0,865971	1,008424	0,517112	-42,25	-40,29
10%	1,806491	0,893293	1,023232	0,517112	-43,36	-42,11
20%	1,86661	0,920409	1,037826	0,517112	-44,40	-43,82
30%	1,926564	0,947316	1,052214	0,517112	-45,38	-45,41
40%	1,98634	0,974014	1,066401	0,517112	-46,31	-46,91
50%	2,045925	1,0005	1,080392	0,517112	-47,19	-48,31
60%	2,105311	1,026775	1,094191	0,517112	-48,03	-49,64
70%	2,164485	1,052838	1,107805	0,517112	-48,82	-50,88
80%	2,22344	1,078688	1,121237	0,517112	-49,57	-52,06
90%	2,282166	1,104326	1,134491	0,517112	-50,29	-53,17
100%	2,340655	1,129752	1,147571	0,517112	-50,97	-54,23
Promedio [%]					-46,96	-47,89
[s]*: segundos						

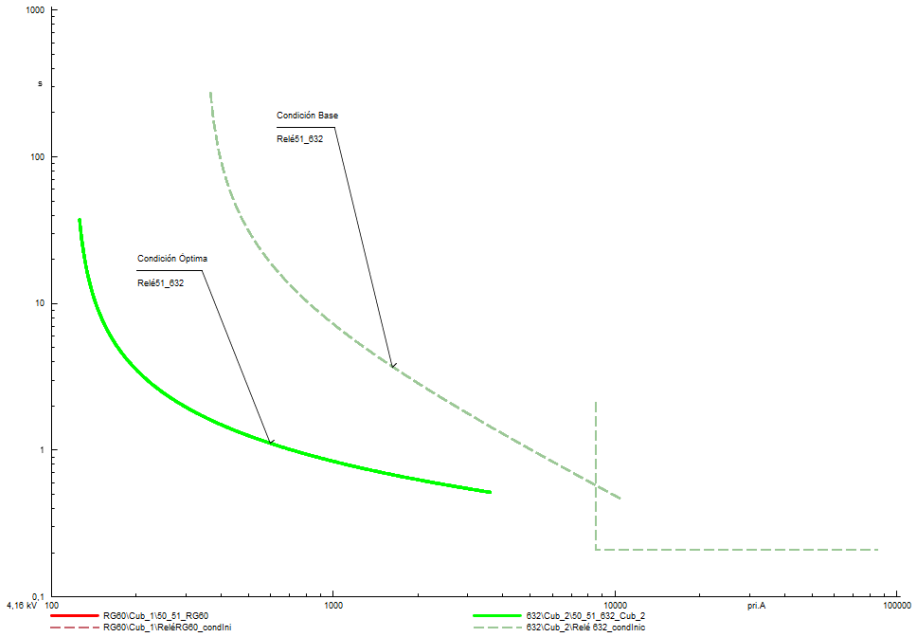
Mediante la Figura 3.5 se presentan las curvas características de los relés 50/51 incorporados en el sistema de prueba. Se muestra el ajuste realizado para los relés 51 tanto por criterio base, así como con el procedimiento de la optimización. Con la finalidad de distinguir entre cada caso de ajuste, se adopta la convención de representar la curva continua como la condición óptima y la curva entrecortada como condición base. De manera que, las curvas continuas representan el comportamiento de los relés 51 al ser ajustados mediante el procedimiento para optimización, mientras que, las curvas entrecortadas representan el comportamiento de los relés 50/51 al ser ajustados con el criterio base. Se debe tomar en cuenta que, los relés 50 únicamente son ajustados con el criterio base, como se observa en todas las gráficas de la Figura 3.5.

En el literal (a) de la Figura 3.5 se presentan las curvas características del relé _RG60, tanto en condición base como en condición óptima, a estas curvas se las identifica con color rojo. En el caso del literal (b), se presenta de la misma manera las curvas

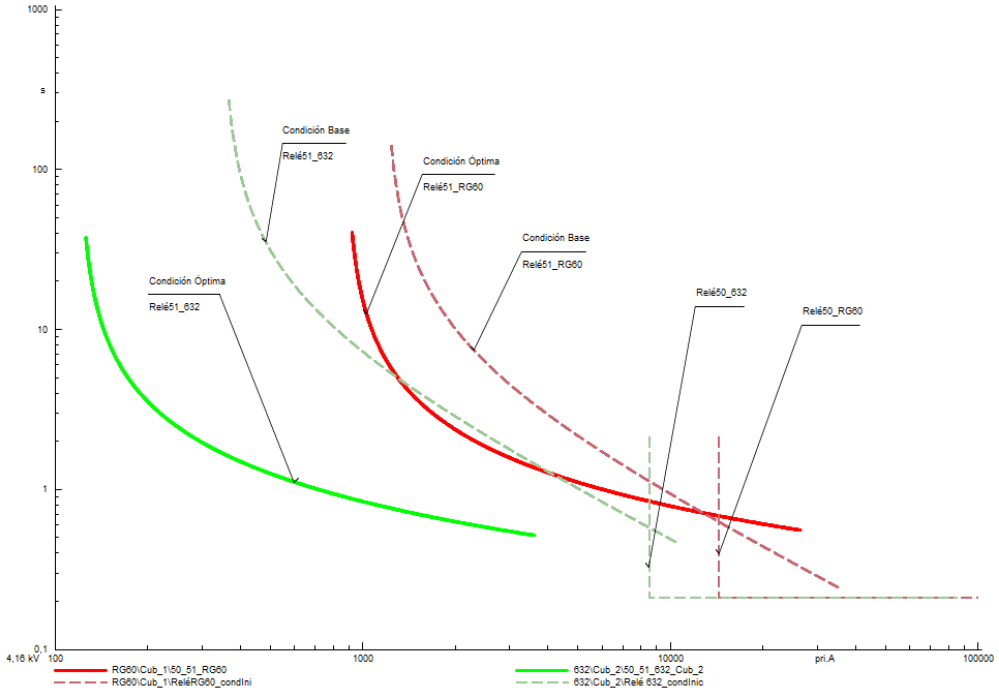
características para el relé _632, en donde se las identifica con color verde. Finalmente, en el literal (c) se presentan las curvas características de los dos relés, según su condición base y su condición óptima, como se ha planteado en los anteriores literales. Se observa el cambio que ocurre en la curva característica, en donde, en un comienzo se encuentran en condición base y pasan a condición óptima.



a)



b)



c)

Figura 3.5. Curvas características - Ajuste de parámetros de relés 50/51. a) Ajuste base y ajuste óptimo para relé 51_RG60, b) Ajuste base y ajuste óptimo para relé 51_632 y c) Ajuste base y ajuste óptimo para relés 51_RG60 y 51_632.

Al calcular una falla menos sensible que la empleada para ejecutar el barrido de fallas, se sigue obteniendo una disminución de los tiempos de operación obtenidos con los ajustes base de los relés 51 al aplicar los ajustes óptimos. Se demuestra lo mencionado con anterioridad al calcular una falla monofásica al 10% de distancia del alimentador LOHL632 -633 a demanda mínima con una resistencia de falla de 2 ohmios en el sistema eléctrico. La Figura 3.6 muestra los resultados obtenidos del cálculo de la corriente de falla mencionada anteriormente.

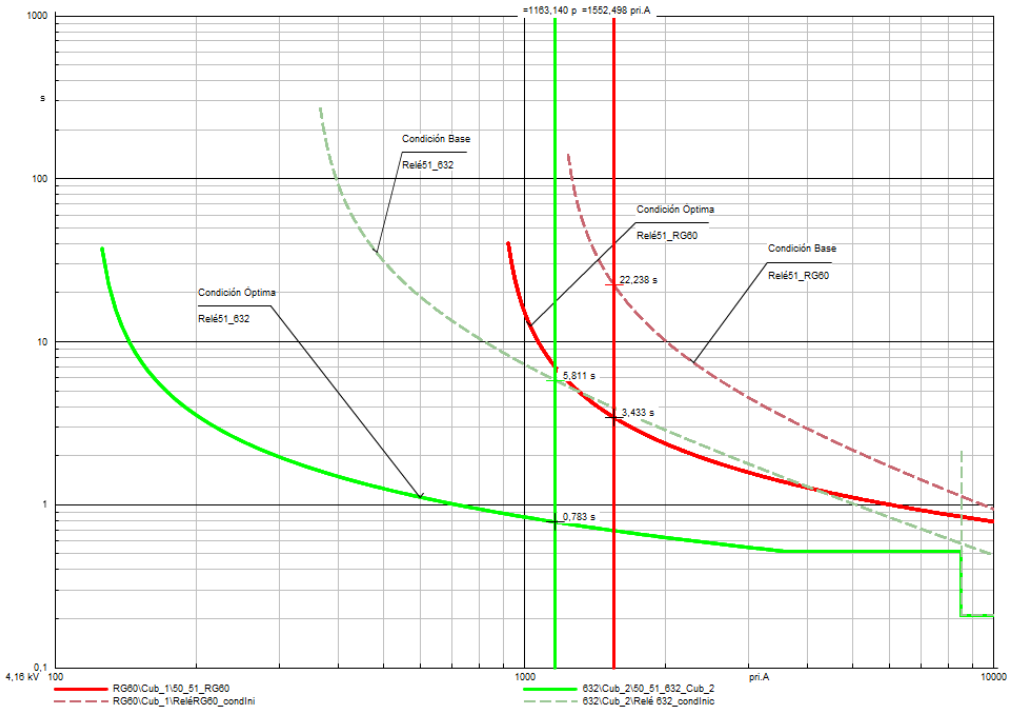


Figura 3.6. Tiempos de operación al calcular una falla monofásica con resistencia de falla de 2 ohmios a demanda mínima al 10% del alimentador LOHL632-633.

En la Figura 3.6 se observa que, al realizar el cálculo de la falla mencionada anteriormente, los relés 51_RG60 y 51_632 disminuyen el tiempo de operación con el ajuste óptimo de 22,238 [s] a 3,433 [s] y de 5,811 [s] a 0,783 [s], respectivamente. En donde, existe una reducción del tiempo de operación en 18,805 [s] (reducción del 84,562%) y 5,028 [s] (reducción del 86,526%) para cada relé. Se observa que, mientras menos sensible es la corriente de falla se tiene una mayor disminución del tiempo de operación de los relés 51 ajustados con los parámetros óptimos.

Con el ajuste óptimo se logra reducir los tiempos de operación de los relés 51 hasta ciertos valores de corrientes. Es decir, para valores de corrientes menores al punto de cruce entre curvas características de condición óptima y condición base, el ajuste óptimo reduce los tiempos de operación. Ya que los tipos de curvas más inclinadas hacen que el relé 51 operé más rápido para corrientes elevadas. Mediante la Figura 3.7 se presentan los puntos de cruce P1 y P2 de las curvas características en condición base y en condición óptima para los relés 51_RG60 y 51_632, respectivamente.

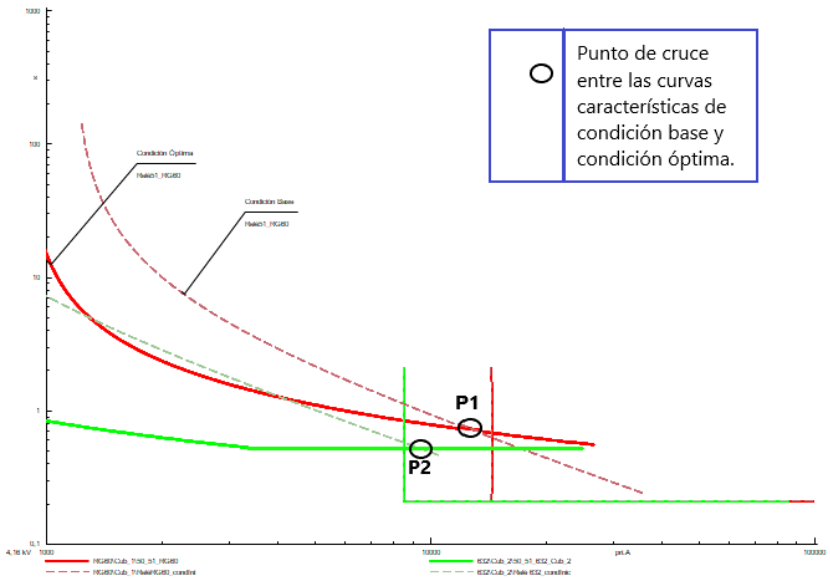


Figura 3.7. Punto de cruce entre curvas características de condición óptima y condición base para los relés 51_RG60 y 51_632.

En la Figura 3.7 se observa que, para el ajuste óptimo de los relés 51, mientras las corrientes más se acercan al tramo de protección de corrientes muy altas, sus tiempos son cada vez más similares a los obtenidos mediante el ajuste con el criterio base. Mientras que, al alejarse del tramo de corrientes altas, en donde se presentan corrientes muy poco sensibles, el ajuste óptimo consigue un rendimiento mejorado al ajuste usando el criterio base, como ya se lo ha analizado con el caso de falla presentado mediante la Figura 3.6. Además, se espera que opere el relé 50 en el caso de producirse corrientes de falla mayores a la corriente pickup del mismo, ya que se encuentra en su zona de protección.

Al calcular una falla trifásica sin impedancia de falla a demanda máxima al 50% del alimentador LOHL632-633, se obtienen los resultados mostrados en la Figura 3.8. Se observa que el relé 50_632 opera de forma instantánea a los 210 [ms]. El relé 51_632 opera como primera protección de respaldo a los 507 [ms], este tiempo de operación es similar tanto con el ajuste óptimo, como con el ajuste base de los relés 51. Finalmente, el relé 51_RG60 actúa como segunda protección de respaldo, llegando a operar en 796 [ms] y 968 [ms] para el ajuste óptimo y el ajuste base, respectivamente; disminuyendo 172 [ms] (reducción del 17,77%) en la operación del relé 51_RG60.

Figura 3.8. Tiempos de operación al calcular una falla trifásica sin impedancia de falla a demanda máxima al 50% de la línea LOHL632-633.

Con los resultados obtenidos, se aprecian las mejoras conseguidas con el procedimiento de la optimización realizado, tomando como referencia el criterio base. Se debe tomar en cuenta que el ajuste mediante el criterio base es adecuado para proteger a los sistemas eléctricos, sin embargo, el procedimiento de optimización consigue mejores ajustes gracias a la exploración eficiente del espacio de soluciones. Se observa una mejoría en el tiempo de operación de los relés 51, manteniendo una coordinación entre los relés del sistema eléctrico de prueba.

3.2 Conclusiones

Al realizar la comparación entre el ajuste obtenido de los criterios proporcionados por la literatura y el ajuste dado por el procedimiento de la optimización, se determina que los tiempos de operación de los relés 51_RG60 y 51_632, calculados con la corriente de prueba en el nodo 633, disminuyen de 2,341 [s] a 1,148 [s] y de 1,13 [s] a 0,517 [s] respectivamente, reduciendo el tiempo de operación en 1,193 [s] (reducción del 50,97 %) y 0,61 [s] (reducción del 54,23 %) para cada relé.

Se verifica que con el ajuste de los relés 51 mediante el procedimiento de la optimización propuesto, al realizar el barrido de fallas en el alimentador LOHL632-633, los tiempos de operación de todos los relés 51 obtenidos para cada punto de falla en el alimentador son menores que los tiempos de operación obtenidos por criterios tradicionales de ajuste. Se tiene una reducción promedio del tiempo de operación en todo el alimentador de 46,96 % y 47,87 %, para cada relé. Esto favorece a la mitigación de las consecuencias contraproducentes causadas por fallas de cortocircuito, como por ejemplo el daño de equipos y la pérdida de vida útil, en toda la zona de protección.

Se observa en todos los puntos de falla que los parámetros conseguidos por el procedimiento de la optimización para el ajuste de los relés 51, además de que permiten obtener tiempos de operación menores, también permiten una operación coordinada entre relés, actuando primeramente el relé principal y después de un intervalo de tiempo (Coordination Time Interval - CTI) alrededor de 300 [ms], actúa el relé de respaldo. Con esto se asegura una operación selectiva de los relés ante una falla en el sistema, cumpliendo la condición del CTI entre los tiempos de operación de los relés del sistema de prueba.

Durante la experimentación se evidenció la existencia de óptimos locales que hacen complejo el problema de optimización. Tomando en cuenta que el problema de coordinación óptima de relés 51 es complejo y puede tener varios óptimos locales, la metodología propuesta, al estar basada en un algoritmo inspirado en la naturaleza, tiene la capacidad de determinar de forma adecuada el óptimo global del problema de optimización.

La función objetivo de minimización del sumatorio de los tiempos de operación para los relés 51 con sus respectivas restricciones implementadas, es adecuada para optimizar el ajuste de los relés 51 gracias a la obtención de la evaluación para cada combinación, permitiendo que el algoritmo COA ordene de mejor a peor combinación para descartar las peores combinaciones y realizar la búsqueda de nuevas combinaciones tomando como punto objetivo de llegada las mejores combinaciones actuales de cada iteración y la mejor combinación. Al resolver la coordinación óptima de relés de sobre corriente

temporizados (51) para el sistema de prueba, se ha obtenido un óptimo global de tiempo de operación de 1,672 [s].

Se puede ver que el algoritmo COA logra converger con pocas iteraciones a valores óptimos. Con aproximadamente 20 iteraciones, el algoritmo ya se aproxima al valor óptimo global teniendo una población considerable de combinaciones. Cuando el algoritmo COA alcanza el número total de iteraciones se consiguen los parámetros óptimos para el ajuste de coordinación óptima para relés 51 en un tiempo de ejecución de aproximadamente 18 [s].

Además, se evidencia una mejora en la sensibilidad del relé ante la ocurrencia de fallas cercanas pero mayores a la corriente de demanda máxima, por ser estas las que mayormente se presentan en sistemas de distribución.

3.3 Recomendaciones y trabajos futuros

Se recomienda considerar la utilización de otros métodos de optimización para comparar resultados y validar que se han obtenido los óptimos globales de la minimización de tiempos de operación a través del método COA.

Se recomienda tomar en cuenta todas las restricciones necesarias para resolver el problema de optimización y obtener parámetros de ajuste que permitan operar de manera adecuada a los relés 51 al disminuir sus tiempos de operación con respecto al ajuste tradicional y operar de forma coordinada entre relés ante una falla de cortocircuito.

Como trabajo futuro se puede aplicar esta metodología de optimización a otras secciones del sistema eléctrico que pueden ser sistemas de subtransmisión, transmisión y otros sistemas de distribución que tengan protecciones de sobre corriente.

Considerar en un trabajo futuro la característica de direccionalidad de los relés de sobre corriente, especialmente en las secciones de subtransmisión y transmisión, que son

donde mayormente se aplican relés de sobre corriente direccionales, o en sistemas de distribución con generación distribuida.

En un trabajo futuro se puede implementar más relés de sobre corriente en el sistema de prueba, para verificar que el algoritmo consigue resultados óptimos al aumentar el número de parámetros en su optimización.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] S. Samadinasab, F. Namdari, y N. Shojaei, “A New Method for Optimal Coordination of Overcurrent Relays in Power System Networks Earthing System Design”, *International Electrical Engineering Journal (IEEJ)*, vol. 6, núm. 11, 2015, pp. 2066–2073.

[2] J. Blackburn y T. Domin, “Protection Fundamentals and Basic Design Principles”, en *Protective Relaying Principles and Applications*, 3a ed., H. Willis y M. Rashid, Eds., New York, 2006, pp. 198–205.

[3] W. H. Kersting, “Radial Distribution Test Feeders”, en *2001 IEEE Power Engineering Society Winter Meeting*, Columbia, jun. 2001, pp. 908–912. doi: 10.1109/PESW.2001.916993.

[4] DIgSILENT Powerfactory. *IEEE 13 Node Test Feeder*, Alemania. [En línea]. Disponible en: www.digsilent.de.

[5] R. Rajabioun, “Cuckoo Optimization Algorithm”, *Applied Soft Computing Journal*, vol. 11, núm. 8, dic. 2011, pp. 5508–5518, doi: 10.1016/j.asoc.2011.05.008.

[6] M. Li y Y. Da Xue, “K-means Clustering with Manifold”, en *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010* IEEE*)*, China, ago. 2010, pp. 2095–2099.

[7] D. Usman y I. Bin Mohamad, “Outlier removal approach as a continuous process in basic K-means clustering algorithm”, *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, núm. 4, 2014, pp. 771–777, doi: 10.19026/rjaset.7.315.

[8] S. A. Jalaee, A. Ghaseminejad, M. Lashkary, y M. Rezaee Jafari, “Forecasting Iran’s energy demand using cuckoo optimization algorithm”, *Mathematical Problems in Engineering*, vol. 2019, 2019, doi: 10.1155/2019/2041756.

[9] P. Makeen, H. A. Ghali, y S. Memon, “Experimental and Theoretical Analysis of the Fast Charging Polymer Lithium-Ion Battery Based on Cuckoo Optimization Algorithm (COA)”, *IEEE Access*, vol. 8, 2020, pp. 140486–140496, doi: 10.1109/ACCESS.2020.3012913.

[10] A. Mahari y H. Seyedi, “An analytic approach for optimal coordination of overcurrent relays”, *IET Generation, Transmission and Distribution*, vol. 7, núm. 7, 2013, pp. 674 –680, doi: 10.1049/iet-gtd.2012.0721.

[11] J. C. Guamán, “Estudio de coordinación de protecciones del sistema de subtransmisión de CNEL EP Sucumbíos ante un nuevo punto de conexión al Sistema Nacional Interconectado”, *Tesis de pregrado, EPN*, Quito, Ec., 2017. [En línea]. Disponible en: http://bibdigital.epn.edu.ec/handle/15000/17463.

[12] R. Rajabioun, “Cuckoo Optimization Algorithm”, *MathWorks*, 2011. Consultado: el 10 de diciembre de 2023. [En línea]. Disponible en: https://la.mathworks.com/matlabcentral/fileexchange/35635-cuckoo-optimization-algorithm.

[13] X. Yang y S. Deb, “Cuckoo Search via Lévy Flights”, *IEEE*, 2009, pp. 210–214. doi: 10.1109/NABIC.2009.5393690.

[14] L. M. Moreno, J. David, V. Camilo, A. Figueroa, y J. Sebastián Jiménez, “Actividad 07 : Optimización heurística Planteamiento del problema Optimización de funciones de prueba”, Medellín, ene. 2021. [En línea]. Disponible en: https://rpubs.com/josjimenezja/optimizacion_heuristicaDecisionesbajoincertidumbre.

[15] D. Jain, “Data Normalization in Data Mining”, *geeksforgeeks*. [En línea]. Disponible en: https://www.geeksforgeeks.org/data-normalization-in-data-mining/.

[16] Gútiez, “Señal analógica en TIA Portal – Escalado”, *PROGRAMACIÓNSIEMENS*. Consultado: el 5 de diciembre de 2023. [En línea]. Disponible en: https://programacionsiemens.com/escalado-de-una-senal-analogica-en-tia-portal/.

[17] Unique Simulations, EE.UU. Satisfying Constraints by COA. (enero 21, 2020).

Consultado: el 5 de diciembre de 2023. [En línea]. Disponible en: https://youtu.be/PyKQzh-88ps?si=Bta4TKwz6qU1AKgp.

[18] DIgSILENT PowerFactory. Python Function Reference (2023), ed. 2013. Consultado: el 9 de enero de 2024. [En línea]. Disponible en: https://www.digsilent.de/.

5 ANEXOS

ANEXO I. Sistema de 13 nodos de la IEEE modelado en PowerFactory.

ANEXO II. Datos del sistema IEEE de 13 nodos.

ANEXO III. Explicación de la estructura del algoritmo COA en Python.

ANEXO IV. Función objetivo con sus restricciones codificada en Python.

ANEXO V. Barrido de fallas en un alimentador del sistema de prueba.

ANEXO VI. Gráficas de comprobación de restricciones.

ANEXO VII. Manual de usuario de la metodología propuesta generada.

ANEXO I

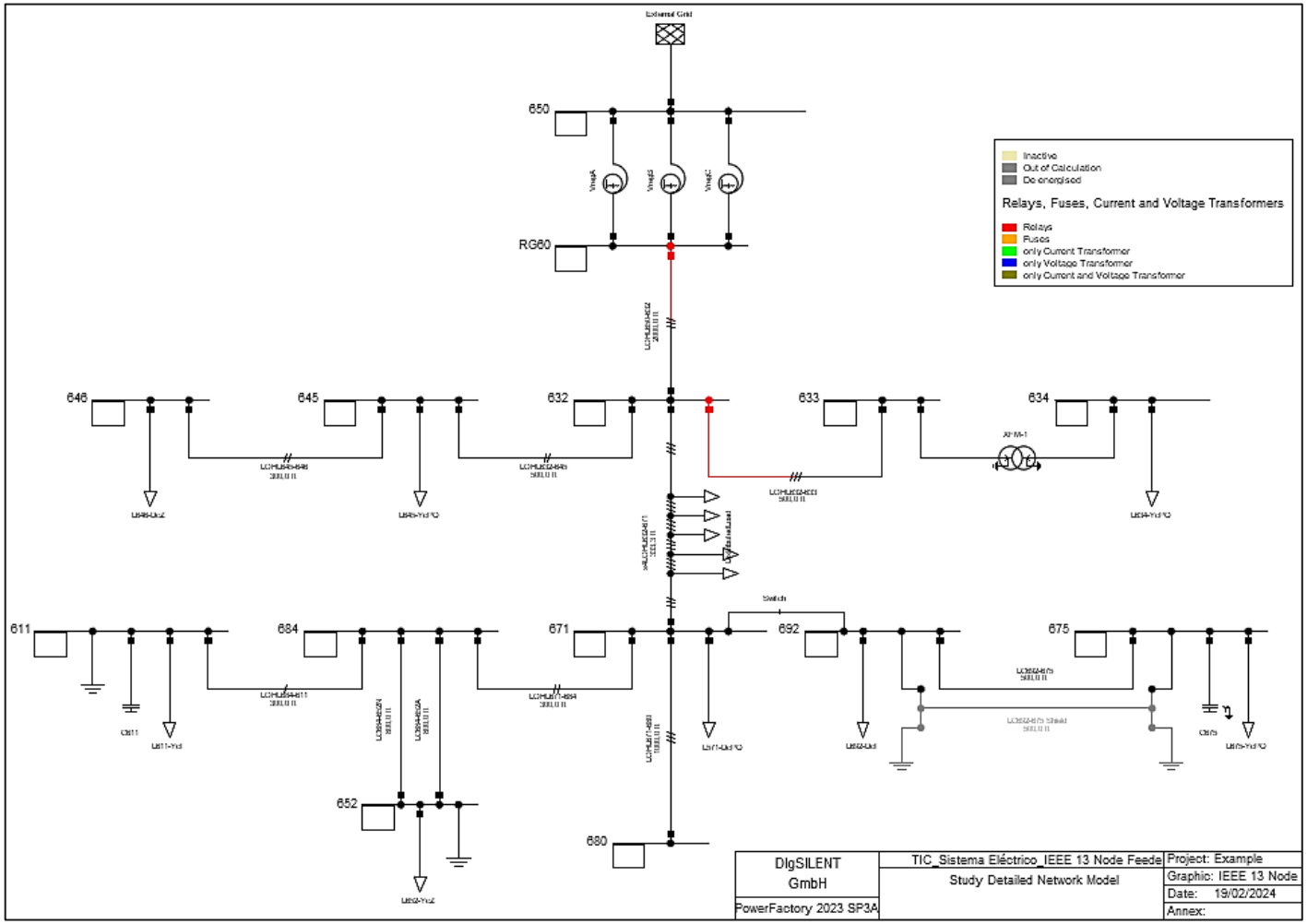


Figura A1.1. Sistema eléctrico de prueba IEEE 13 Node Feeder *[4]*

ANEXO II

Tabla A2.1. Cargas desbalanceadas del sistema IEEE de 13 nodos *[4]*.

Nodo	Conexión	Modelo Constante	Fase 1 / A-B		Fase 2 / B-C		Fase 3 / C-A		∑P	∑Q
			P	Q	P	Q	P	Q		
			kW	kVAr	kW	kVAr	kW	kVAr	kW	kVAr
632-671*	Y	PQ	17	10	66	38	117	68	200	116
634	Y	PQ	160	110	120	90	120	90	400	290
645	Y	PQ	0	0	170	125	0	0	170	125
646	D	Z	0	0	230	132	0	0	230	132
652	Y	Z	128	86	0	0	0	0	128	86
671	D	PQ	385	220	385	220	385	220	1155	660
675	Y	PQ	485	190	68	60	290	212	843	462
692	D	I	0	0	0	0	170	151	170	151
611	Y	I	0	0	0	0	170	80	170	80
		∑	1175	616	1039	665	1252	821	3466	2102
		S (kVA)		1327		1234		1497		4054

Tabla A2.2. Identificador de los tipos de líneas de distribución *[3]*.

ID de línea	Tipo	Línea
500	3 fases + neutro	Aérea
505	2 fases + neutro	Aérea
510	1 fase + neutro	Aérea
515	3 fases	Subterránea
520	1 fase	Subterránea

Tabla A2.3. Conexión de nodos en el sistema *[4]*.

Nodo A	Nodo B	Distancia en pies	Fases	Calibre por fase	Calibre por neutro	ID de línea
632	645	500	CBN	1/0	1/0	505
632	633	500	CABN	4/0	4/0	500
633	634	XFM-1 - Transformador 4,16/0,48 kV				
645	646	300	CBN	1/0	1/0	505
650	632	2000	BACN	556,5	4/0	500
684	652	800	AN	1/0 AA, TS	1/0 Cu	520
632	671	2000	BACN	556,5	4/0	500
671	684	300	ACN	1/0	1/0	505
671	680	1000	BACN	556,5	4/0	500
671	692	Switch				
684	611	300	CN	1/0	1/0	510
692	675	500	ABCN	250 AA, CN	Ninguno	515

Tabla A2.4. Datos del conductor para líneas aéreas *[3] [4]*.

Conductor AWG	Material	R [Ω/milla]	Diámetro [pulgadas]	GMR [pies]	Ampacidad [A]	R (DC a 20° C) [Ω/miles de pies]
556,5	ACSR	0,1859	0,927	0,0313	730	0,03520833
4/0	ACSR	0,592	0,563	0,00814	340	0,1121212
1/0	ACSR	1,12	0,398	0,00446	230	0,2121212

ANEXO III

1. Inicializar el tamaño de la población, el número máximo de iteraciones (maxIter), número máximo de cucús (maxNumOfCuckoos), el número de clusters requeridos mediante el método K-means (knnClusterNum) y los límites superiores e inferiores para cada dimensión del problema.
    - El tamaño de la población es iniciado mediante los siguientes parámetros: número de variables de optimización (npar) y número de población inicial (numCuckooS).
    - Mediante la definición de (maxIter) se define el número de iteraciones que va a realizar el algoritmo para llegar a sus variables optimizadas.
    - El número máximo de cucús (maxNumOfCuckoos) limita el número de cucús que pueden vivir al mismo tiempo en el ambiente. Este parámetro también determina el tamaño de la población que el algoritmo utiliza para buscar soluciones óptimas, mientras más cucús sobreviven en el ambiente, hay una mayor población que busca soluciones óptimas.
    - Mediante (knnClusterNum) se definen los números de clusters que se determina mediante el método K-means para agrupar los cucús.
    - Se presentan límites para los valores de las variables que están dados por valores mínimos (varLo) y valores máximos (varHi), además se presentan límites para el número de huevos, los cuales son mínimo número de huevos (minNumberOfEggs) y máximo número de huevos (maxNumberOfEggs) por cada cucú, estos límites también determinan el tamaño de la población de búsqueda del algoritmo COA.
    - Se presentan los parámetros radiusCoeff, motionCoeff y accuracy que son ajustables según el problema de optimización que esté resolviendo el algoritmo COA para mejorar la búsqueda de los resultados óptimos.

```
"""
                    Escuela Politécnica Nacional
          Facultad de Ingeniería Eléctrica y Electrónica

Realizado por: Daniel Jiménez
Título: Coordinación óptima de relés de sobre corriente
        temporizado (51) en el sistema de prueba de
        13 nodos de la IEEE.

Mediante el código realizado en Python, se implementa el algoritmo
COA como método de optimización para resolver el problema de
coordinación óptima de relés 51 en el sistema de prueba. El
algoritmo de optimización con la función objetivo consigue
parámetros óptimos para el ajuste de los relés 51. Con el ajuste
```

```
óptimo se disminuyen los tiempos de operación obtenidos del ajuste
tradicional de los relés 51 y al mismo tiempo se efectúa la operan
de forma coordinada entre relés.
"""

#----------Algoritmo COA-----
######################## Parámetros del algoritmo COA #############
npar = 9  # Número de variables de optimización
varLo = -0.5  # Límite inferior de parámetros
varHi = 0.5 # Límite superior de parámetros
# Poner parámetros del algoritmo COA

numCuckooS = 5                # Inicialización de población inicial
minNumberOfEggs = 5           # Número mínimo de huevos cucú
maxNumberOfEggs = 10           # Número máximo de huevos cucú
maxIter = 100                 # Número de iteraciones máximas
knnClusterNum = 4            # Número de clusters que se quiere
                                hacer
motionCoeff = 9               # Variable Lambda en paper de COA,
                                default=2
accuracy = float('-inf')      # La precisión de la respuesta que se
                                necesita
maxNumOfCuckoos = 10          # Número máximo de cucús que pueden
                                vivir al mismo tiempo
radiusCoeff = 2               # Parámetro de control de la puesta de
                                huevos
cuckooPopVariance = 1e-13     # Varianza de la población que
                                finaliza la optimización
```

2. Generar una población inicial aleatoria de (numCuckooS) nidos de Cuckoo.
   for cuckooNumber in range(numCuckooS):
   - Se realiza la generación de los centros de puesta de huevos para cada cucú, denotando como centro de puesta de huevos a la posición en la que se encuentra el cucú inicialmente antes de su movimiento hacia otros nidos para poner sus huevos.

```
for cuckooNumber in range(numCuckooS):
    center = [(varHi - varLo) * random.random() + varLo for _
              in range(npar)]
    cuckoo = {"center": center}
    cuckooPop.append(cuckoo)
```

3. Inicialización del punto objetivo (goalPoint) y el lazo de iteraciones
   - Se genera un punto objetivo dando valores aleatorios que se encuentren entre los rangos de las variables para la inicialización del algoritmo COA. El punto objetivo es el lugar donde otros cucús quieren llegar para poner sus huevos debido a que en este punto se tiene una alta tasa de huevos sobrevivientes. Además, se inicializa el mejor cucú y se inicializan las variables de control del lazo de iteraciones.

```
# Inicialización del punto objetivo de llegada y la mejor posición
de cucú
iteration = 0
maxProfit = -1e20  # Se pone un numero negativo para iniciar
```

```
goalPoint = (varHi - varLo) * np.random.rand(1, npar) + varLo  # Un
          número aleatorio es puesto como punto objetivo
goalPoint= np.squeeze(goalPoint) #Se tiene solo un vector no una
                                  matriz
globalBestCuckoo = goalPoint
globalMaxProfit = maxProfit
profitVector = []
```

4. Repetir hasta que se alcance el número máximo de iteraciones:
   a) Establecer el número de huevos de cada cucú.

```
# Inicializar número de huevos por cada cucú
   for cuckooNumber in range(numCuckooS):
       cuckooPop[cuckooNumber]['numberOfEggs'] =
             np.floor((maxNumberOfEggs - minNumberOfEggs) *
             np.random.rand() + minNumberOfEggs)
```

   b) Cálculo del total de huevos de todos los cucús (cuckooNumber).

```
summ = 0
   for cuckooNumber in range(numCuckooS):
       summ = summ + cuckooPop[cuckooNumber]['numberOfEggs']
```

   c) Se calcula el radio de puesta de huevos (Egg Laying Radius- ELR) por cada cucú, este valor indica el radio máximo que puede recorrer cada cucú para la puesta de sus huevos.

```
for cuckooNumber in range(numCuckooS):
       cuckooPop[cuckooNumber]['eggLayingRadius'] =
           cuckooPop[cuckooNumber]['numberOfEggs'] / summ *
           (radiusCoeff * (varHi - varLo))
```

   d) En base al "eggLayingRadius" se genera el radio para la puesta de cada huevo de cada cucú, y a estos radios, los cuales no pueden ser mayores al "Egg Laying Radius" respectivo de cada cucú se los almacena en (eggLayingRadiuses) de cada cucú.

```
for cuckooNumber in range(numCuckooS):
       cuckooPop[cuckooNumber]['eggLayingRadiuses'] =
                 cuckooPop[cuckooNumber]['eggLayingRadius'] *
                 np.random.rand(int(cuckooPop[cuckooNumber]['numbe
                 rOfEggs']), 1)
```

   e) Puesta de huevos del cucú en puntos aleatorios:
   for cuckooNumber in range(numCuckooS):
   - Se genera de forma aleatoria la posición de la puesta de cada huevo perteneciente a cada cucú tomando en cuenta el radio de puesta de cada huevo almacenado en (eggLayingRadiuses) de su respectivo cucú y su ángulo de desvío respecto a la trayectoria objetivo dado por la variable (angle) el cual se le otorga un ángulo diferente para cada huevo de un cucú y está dado entre 0 a 2*π. Con el ángulo y el módulo (radio) se aplica la suma de Euler añadiendo un término aleatorio en el coseno para darle aleatoriedad a la trayectoria que recorre el cucú y se almacena en (addingValue), esto se

realiza a todas las variables de optimización de todos los huevos de todos los cucús.

```
for iii in range(npar):
    addingValue[iii] = (-1) ** randNum * tmpRadiuses[cnt] *
        np.cos(angles[cnt]) + tmpRadiuses[cnt] * np.sin(angles[cnt])
```

Donde: randNum es el número aleatorio de la trayectoria que toma el cucú.
tmpRadiuses es la matriz que almacena temporalmente los radios de cada huevo.
cnt corresponde a la posición de cada huevo de cada cucú.

- La posición final de todos los huevos se almacena en la variable "newParams", esta posición final es la suma de la posición inicial del respectivo cucú (centro del cucú o nido) y la trayectoria que viaja el cucú para poner dicho huevo (addingValue), de esta manera se almacenan las nuevas posiciones de todos los cucús en "newParams" apilando todas las posiciones de todos los huevos en dicha matriz.

```
newParams = np.vstack((newParams, params + addingValue))
```

- Se revisa si los valores de las posiciones de los huevos cucús se encuentran dentro de los límites (newParams), si no es así se remplaza por el límite más cercano. Es decir, se matan a los huevos cucú que son descubiertos como paracitos.

```
newParams[newParams > varHi] = varHi
newParams[newParams < varLo] = varLo
```

- Se almacenan los resultados de las nuevas posiciones que cumplen con los límites de las variables en "newPosition4Egg" para cada cucú.

```
    for cuckooNumber in range(numCuckooS):
        params = cuckooPop[cuckooNumber]['center']  # Obtener
                                  centros de puesta de huevos
        tmpRadiuses = cuckooPop[cuckooNumber]['eggLayingRadiuses']
        numRadiuses = tmpRadiuses.size

        # División de la circunferencia en 'numRadiuses' segmentos
        # Para buscar en todos los hábitats posibles la solución
                                                    óptima
        angles = np.linspace(0, 2 * np.pi, numRadiuses)  # En
                                                      radianes
        newParams = np.empty((0, npar))

        for cnt in range(numRadiuses):
            addingValue = np.zeros(npar)
            for iii in range(npar):
                randNum = np.floor(2 * np.random.rand()) + 1

                addingValue[iii] = (-1) ** randNum *
                    tmpRadiuses[cnt] * np.cos(angles[cnt]) +
                    tmpRadiuses[cnt] * np.sin(angles[cnt])
            newParams = np.vstack((newParams, params +
                                       addingValue))

        # Chequear límites de los parámetros
```

```
        newParams[newParams > varHi] = varHi
        newParams[newParams < varLo] = varLo

        newParams = np.array(newParams)
        cuckooPop[cuckooNumber]['newPosition4Egg'] = newParams
```

f) Se revisa si en la posición los huevos de cada cucú no se encuentran posiciones de huevos repetidas, si se da el caso, se eliminan las posiciones repetidas, esto se debe a que los cucús solamente ponen un huevo en cada nido. Los huevos no repetidos eclosionan y crecen.

```
    for cuckooNumber in range(numCuckooS):
        tmpPopulation = cuckooPop[cuckooNumber]['newPosition4Egg']
        tmpPopulation = np.floor(tmpPopulation * 1e20) / 1e20
        ii = 2
        cntt = 1
        while ii <= tmpPopulation.shape[0] or cntt <=
                                    tmpPopulation.shape[0]:
            if np.all(tmpPopulation[cntt-1, :] ==
                          tmpPopulation[ii-1, :]):
                tmpPopulation = np.delete(tmpPopulation, ii-1,
                                          axis=0)


            ii += 1
            if ii > tmpPopulation.shape[0] and cntt <=
                                        tmpPopulation.shape[0]:
                cntt += 1
                ii = cntt + 1
                if ii > tmpPopulation.shape[0]:
                    break
        cuckooPop[cuckooNumber]['newPosition4Egg'] = tmpPopulation
```

g) Se evalúa el hábitat de cada cucú recién crecido y los centros de puesta de huevos:
for cuckooNumber in range(numCuckooS):
…
profitValues = -costFunction(combinedPositions) #La población es la unión del centro de puesta de huevos y la posición de los cucús recién crecidos (combinedPositions).

```
for cuckooNumber in range(numCuckooS):
        center = cuckooPop[cuckooNumber]['center']
        newPosition4Egg =
                        cuckooPop[cuckooNumber]['newPosition4Egg']
        combinedPositions = np.vstack((center, newPosition4Egg))
        profitValues = -costFunction(combinedPositions)
        cuckooPop[cuckooNumber]['profitValues'] = profitValues
```

h) Se limita el número máximo de cucús que pueden vivir en el ambiente al mismo tiempo con la variable “maxNumOfCuckoos”:
if numCuckooS > maxNumOfCuckoos:

- Se determina si el número de cucús es mayor al máximo aceptable, en el caso

de cumplirse lo anteriormente mencionado se ordena en función de la evaluación a todos los cucús recién crecidos en orden descendente y solamente se conservan los “maxNumOfCuckoos” primeros y mejores cucús y se guarda al mejor cucú como (bestCuckooCenter).

else:

- Caso contrario se ordena en función de la evaluación a todos los cucús recién crecidos en orden descendente y se guarda al mejor cucú como (bestCuckooCenter)

```
    if numCuckooS > maxNumOfCuckoos:
        for cuckooNumber in range(numCuckooS):
            tmpProfits.append(cuckooPop[cuckooNumber]
                              ['profitValues'])
            allPositions.append(np.vstack((cuckooPop
                  [cuckooNumber]['center'],cuckooPop[cuckooNumber]
                   ['newPosition4Egg']
                               [:, :npar])))
            whichCuckooPopTheEggBelongs.append(np.full(
                   (cuckooPop[cuckooNumber]['newPosition4Egg'][:,
                   :npar].shape[0],), cuckooNumber+1))
        tmpProfits = np.concatenate(tmpProfits)
        allPositions = np.concatenate(allPositions, axis=0)
        whichCuckooPopTheEggBelongs =
                   np.concatenate(whichCuckooPopTheEggBelongs)

        # Ordenar posiciones de cucús de mejor a peor
          evaluación
        sortingIndex = np.argsort(tmpProfits, axis=0)[::-1]
        sortedProfits = tmpProfits[sortingIndex].squeeze()
        sortedProfits = np.reshape(sortedProfits, (-1, 1))
                        # Para pasarle de una fila a una columna
        # Mejor cucú que será almacenado en la siguiente
          iteración
        bestCuckooCenter = allPositions[sortingIndex[0]
                           , :npar]
        # Almacenar los Nmax mejores cucús y a los demás se
          descartar
        sortedProfits = sortedProfits[:maxNumOfCuckoos]
        allPositions =
                   allPositions[sortingIndex[:maxNumOfCuckoos],
                   :].squeeze()

        # Almacenamiento de cucús en 'CuckooPop'
        cuckooPop = []

        for ii in range(maxNumOfCuckoos):
            cuckoo1 = {
                'newPosition4Egg': allPositions[ii]
                         .reshape(1,-1),
                'center': allPositions[ii],
                'profitValues': sortedProfits[ii]
                }
            cuckooPop.append(cuckoo1)
        del cuckoo1
        numCuckooS = maxNumOfCuckoos
```

```
32
33      # Si no excede Nmax se almacena toda la población de cucús
34      else:
35
36          for cuckooNumber in range(1, numCuckooS + 1):
37              tmpProfits.extend(cuckooPop[cuckooNumber -
                        1]['profitValues'])
38              allPositions.extend([np.array
                       (cuckooPop[cuckooNumber - 1]
                       ['center']).reshape(1,npar
                        ), cuckooPop[cuckooNumber - 1]
                       ['newPosition4Egg'][:, :npar]])
39              whichCuckooPopTheEggBelongs.extend(cuckooNumber *
                       np.ones((cuckooPop[cuckooNumber
                       - 1]['newPosition4Egg']
                       [:, :npar].shape[0], 1)))
40          tmpProfits = np.array(tmpProfits)
41          allPositions = np.concatenate(allPositions)
42          whichCuckooPopTheEggBelongs =
                        np.array(whichCuckooPopTheEggBelongs)
43
44          sortingIndex = np.argsort(tmpProfits, axis=0)[::-1]  #
                              Índice de ordenamiento descendente
45          sortedProfits = np.concatenate(sorted(tmpProfits,
                         reverse=True))  # Valores ordenados
                                      de las evaluaciones
46          sortedProfits = np.reshape(sortedProfits
                         ,(-1, 1)) #Esta función le pasa de
                                    ser una fila a una columna
47          # Mejor cucú que será almacenado en la siguiente
              iteración
48          bestCuckooCenter = allPositions[sortingIndex[0], :npar]
```

i) Identificación del mejor cucú global:
if currentMaxProfit > globalMaxProfit:

- Se obtiene la evaluación del mejor cucú obtenido en la correspondiente iteración (currentMaxProfit) y se compara con la evaluación del mejor cucú global (globalMaxProfit), en el caso de tener mejor profit el mejor cucú actual que el global, entonces se actualiza el mejor cucú global por el mejor cucú actual.

```
1     if currentMaxProfit > globalMaxProfit:
2         globalBestCuckoo = currentBestCuckoo
3         globalMaxProfit = currentMaxProfit
```

j) Agrupación por clusters mediante método k-means a las posiciones de los cucús, determinación del mejor cluster y determinación del mejor hábitat:

Se unen todas las posiciones de todos los cucús en una sola matriz (allPositions) para proceder a agrupar los cucús según su centro estimado correspondiente a cada cluster obtenido por el método K-means.
if np.sum(np.var(allPositions)) < cuckooPopVariance:
break

- Si la suma de la varianza de la matriz de posiciones de todos los cucús "allPositions" es menor a la varianza aceptable de la población de cucús (cuckooPopVariance) entonces se termina el algoritmo, debido a que, esto significa que toda la población converge en un mismo punto y ya se ha determinado el óptimo global.

else:

- Caso contrario se determina el cluster al que corresponde cada cucú según su posición (clusterNumbers) y el centro de los clusters (clusterCenters) mediante el método K-means.

for cnt in range(len(clusterNumbers)):

- Se agrupan las posiciones en una matriz y las evaluaciones de los cucús en otra según el cluster al que corresponde cada cucú evaluado (hábitat).

for cnt in range(knnClusterNum):
    f_mean[cnt] = np.mean(cluster[cnt]['profits'])

- Se evalúa la media de cada cluster, de modo que el cluster con los datos menos dispersos (menor media) es el mejor cluster.
- Del mejor cluster se obtiene el "goalPoint", que es la posición del mejor cucú que ha sido agrupado a ese cluster.

```
#  Determinación del mejor cluster
f_mean = np.zeros((knnClusterNum, 1))
for cnt in range(knnClusterNum):
    f_mean[cnt] = np.mean(cluster[cnt]['profits'])

sortingIndex_f_mean = np.argsort(-f_mean, axis=0)  # Ordenar
                                  los índices de forma descendente
sorted_f_mean = np.sort(f_mean, axis=0)[::-1]  # Ordenar media
                         de clusters de forma descendente
maxFmean = sorted_f_mean[0]; indexOfBestCluster =
                                        sortingIndex_f_mean[0]

# Se conoce el mejor cluster por 'indexOfBestCluster'
# Se selecciona del mejor cluster el punto objetivo
                                                'GoalPoint'
maxProfitInBestCluster =
          max(cluster[indexOfBestCluster.item()]["profits"])
indexOfBestEggPosition =
          max(enumerate(cluster[indexOfBestCluster.item()]
                  ["profits"]), key=lambda x: x[1])[0]
goalPoint = cluster[indexOfBestCluster.item()]
                  ["positions"][indexOfBestEggPosition, 0:npar]
```

k) Migración de cucús maduros hacia el punto objetivo (goalPoint):

tmpPositions[cntPosition, :] = tmpPositions[cntPosition, :] + \
    motionCoeff * np.random.rand(1, npar) * (goalPoint - tmpPositions[cntPosition, :])

- La variable (tmpPositions) almacena la nueva posición de los cucús al realizar su vuelo hacia la posición cercana al punto objetivo, este vuelo tiene una distancia aleatoria. La distancia que recorren los cucús es proporcional al coeficiente que controla la magnitud del vuelo de los cucús (motionCoeff) y la diferencia entre el punto objetivo y el punto de inicio del cucú. La variable "cntPosition" señala la fila de la matriz que almacena las

posiciones de los cucús, de esta manera se genera la nueva posición de todos los cucús después de su vuelo hacia el punto cercano aleatorio al punto objetivo.

tmpPositions[tmpPositions > varHi] = varHi
tmpPositions[tmpPositions < varLo] = varLo

- Se verifica que los valores de las nuevas posiciones de los cucús (tmpPositions) se encuentren dentro de los límites, caso contrario se reemplazan por los límites más cercanos.

```
    numNewCuckooS = 0
    for cntClstr in range(len(cluster)):
        tmpCluster = cluster[cntClstr]
        tmpPositions = tmpCluster['positions']
        for cntPosition in range(tmpPositions.shape[0]):
            tmpPositions[cntPosition, :] =
                  tmpPositions[cntPosition, :] +
                  motionCoeff * np.random.rand(1, npar)
                  * (goalPoint - tmpPositions[cntPosition, :])
        # Chequear si los parámetros se encuentran dentro de los
          límites
        tmpPositions[tmpPositions > varHi] = varHi
        tmpPositions[tmpPositions < varLo] = varLo

        # Actualizar posición de clusters
        cluster[cntClstr]['positions'] = tmpPositions
        cluster[cntClstr]['center'] = np.mean(tmpPositions,
                                      axis=0)
        # Actualizar número de cucús 'numCuckooS'
        numNewCuckooS += cluster[cntClstr]['positions'].shape[0]
```

l) Implementación de nuevos centros de puesta de huevos (nidos) para nuevas generaciones de huevos cucú:
Las posiciones de los cucús se almacenan como centros de puesta de huevos de la nueva generación de cucús.

```
    numNewCuckooS = 0
    for cntClstr in range(len(cluster)):
        tmpCluster = cluster[cntClstr]
        tmpPositions = tmpCluster['positions']
        for cntPosition in range(tmpPositions.shape[0]):
            tmpPositions[cntPosition, :] =
                   tmpPositions[cntPosition, :] +
                  motionCoeff * np.random.rand(1, npar) *
                   (goalPoint - tmpPositions[cntPosition, :])
        # Chequear si los parámetros se encuentran
          dentro de los límites
        tmpPositions[tmpPositions > varHi] = varHi
        tmpPositions[tmpPositions < varLo] = varLo

        # Actualizar posición de clusters
        cluster[cntClstr]['positions'] = tmpPositions
        cluster[cntClstr]['center'] = np.mean
                    (tmpPositions, axis=0)
        # Actualizar número de cucús 'numCuckooS'
        numNewCuckooS += cluster[cntClstr]
```

```
16          ['positions'].shape[0]
```

m) Copiar mejor cucú hacia la siguiente generación:

Se copia el mejor cucú (globalBestCuckoo) en la matriz de centros de cucús en la última fila de dicha matriz, además, en la penúltima fila se copia un número cercano del "globalBestCuckoo" seleccionado aleatoriamente que se encuentre dentro de los límites de las variables, a estos puntos se les calcula el profit. De esta manera se tienen los centros de los nuevos cucús que van a poner sus huevos en la siguiente iteración almacenados por la matriz de centros del paso anterior con las dos posiciones anteriormente mencionadas también incorporadas, dado que, al conservarse el centro de la mejor posición de la iteración actual para la siguiente generación, se logra que siga permaneciendo en otras generaciones esa posición que da buenos resultados y vaya optimizándose, además se conserva el punto cercano al mejor cucú actual dado aleatoriamente para que el algoritmo siga identificando mejores resultados alrededor del mejor cucú actual.

```
1    cuckooPop[-1]['center'] = globalBestCuckoo # Este es el mejor
                                                    cucú
2    cuckooPop[-1]['profitValues'] = -
          costFunction(np.reshape(cuckooPop[-1]['center'],(1,-1)))
3
4    tmp = np.random.rand(1, npar) * globalBestCuckoo
5    tmp[tmp > varHi] = varHi
6    tmp[tmp < varLo] = varLo
7    cuckooPop[-2]['center'] = tmp# Este es el cucú cercano al mejor
                                    cucú
8    cuckooPop[-2]['profitValues'] = -
          costFunction(np.reshape(cuckooPop[-2]['center'],(1,-1)))
```

ANEXO IV

```
#################### Función Objetivo ##########
"""
Con la función objetivo de minimización de tiempos de operación, se
evalúan los hábitats que contienen a las variables de decisión para
ajustar los relés 51. Los peores hábitats son descartados por el
algoritmo COA y los mejores son considerados en la búsqueda de nuevos
mejores hábitats en el procedimiento de la optimización.
"""


def YourCostFunctionName(populationMatrix):
    numOfCuckoos = populationMatrix.shape[0]
    cost = np.zeros((numOfCuckoos, 1))

    for i in range(numOfCuckoos):
        currentCuckoo = populationMatrix[i, :]

        x1 = currentCuckoo[0] #TDS1
        y1 = currentCuckoo[1] #I_PickUp1
        x2 = currentCuckoo[2] #TDS2
        y2 = currentCuckoo[3] #I_PickUp2
        z1 = currentCuckoo[4] #Curva rojo1
        z2 = currentCuckoo[5] #Curva verde2


        # Flujo de potencia
        Ilf0 = 586.61 # Relé-RG60(curva característica color roja)
        Ilf1 = 80.99 # Relé-632(curva característica color verde)
        # CC monofásico a 3 Ohmios de Rf
        Isc0_Rf3 = 1197.65 # Relé-RG60
        Isc1_Rf3 = 797.07 # Relé-632

        # CC monofásico sin Rf
        Isc0 = 4744 # Relé-RG60
        Isc1 = 4535 # Relé-632
        #I CC trifásico
        Isc2 = 8892.621 # Relé-RG60
        Isc3 = 8791 # Relé-632

        ################ Restricciones #
        # Límites del Dial Relé 1
        x1 = ( (  (x1-0)/(1-0)  )*(1-0)+0 )
        G_x1 = -x1
        A_1 = -0.05
        ViolationTerm_1 = (  (G_x1-A_1)+abs(G_x1-A_1)  )/2

        G_x2 = x1
        A_2 = 1.0
        ViolationTerm_2 = (  (G_x2-A_2)+abs(G_x2-A_2)  )/2
```

```
50
51          # Límites del Dial Relé 2
52          x2 = ( (  (x2-0)/(1-0)  )*(1-0)+0 )
53          G_x3 = -x2
54          A_3 = -0.05
55          ViolationTerm_3 = (  (G_x3-A_3)+abs(G_x3-A_3)  )/2
56
57          G_x4 = x2
58          A_4 = 1.0
59          ViolationTerm_4 = (  (G_x4-A_4)+abs(G_x4-A_4)  )/2
60
61          # Límites de corriente Pickup Relé 1
62          y1 = ( (  (y1-0)/(1-0)  )*(Isc0_Rf3-Ilf0*1.5)
                                          +Ilf0*1.5 )
63          G_x5 = y1
64          A_5 =Isc0_Rf3
65          ViolationTerm_5 = (  (G_x5-A_5)+abs(G_x5-A_5)  )/2
66
67          G_x6 = -y1
68          A_6 = -Ilf0
69          ViolationTerm_6 = (  (G_x6-A_6)+abs(G_x6-A_6)  )/2
70
71          # Límites de corriente Pickup Relé 2
72          y2 = ( (  (y2-0)/(1-0)  )*(Isc1_Rf3- Ilf1*1.5)
                                        + Ilf1*1.5 )
73          G_x7 = y2
74          A_7 = Isc1_Rf3
75          ViolationTerm_7 = (  (G_x7-A_7)+abs(G_x7-A_7)  )/2
76
77          G_x8 = -y2
78          A_8 = -Ilf1
79          ViolationTerm_8 = (  (G_x8-A_8)+abs(G_x8-A_8)  )/2
80
81
82          #Curvas-Normalización
83          # Rojo
84          z1 = round( (  (z1-0)/(1-0)  )*(3-0)+0 ) #Ecuación de
                        normalización
85          cur1 = IEC(z1)# función que devuelve α y n de z1
86          ar1 = cur1[0]# valor de α de curva característica del Relé-RG60
87          ar2 = cur1[1]# valor de n de curva característica del Relé-RG60
88
89          # Verde
90          z2 = round( (  (z2-0)/(1-0)  )*(3-0)+0 ) #Ecuación de
                        normalización
91          cur2 = IEC(z2) # función que devuelve α y n de z2
92          av1 = cur2[0]# valor de α de curva característica del Relé-632
93          av2 = cur2[1]# valor de n de curva característica del Relé-632
94
95          # CTI > 300 [ms]
96          CTI = 0.3
97          G_x9 = -(-(x2*av1)/((Isc3/y2)**av2-1) +
                     (x1*ar1)/((Isc2/y1)**ar2-1))
98          A_9 = -CTI
```

```
 99          ViolationTerm_9 = (  (G_x9-A_9)+abs(G_x9-A_9)  )/2
100
101          # Tiempo de operación del relé 2 > 510 [ms]
102          G_x10 = -((x2*av1)/((Isc3/y2)**av2-1))#Ecuación 1.4
103          A_10 = -0.510
104          ViolationTerm_10 = (  (G_x10-A_10)+abs(G_x10-A_10)  )/2
105
106          # Curva verde mayor a roja
107          G_x11 = -z2
108          A_11 = -z1
109          ViolationTerm_11 = (  (G_x11-A_11)+abs(G_x11-A_11)  )/2
110          # Tiempo de operación del relé 2 < 550 [ms]
111          G_x12 = ((x2*av1)/((Isc3/y2)**av2-1))
112          A_12 = 0.550
113          ViolationTerm_12 = (  (G_x12-A_12)+abs(G_x12-A_12)  )/2
114
115          # CTI < 350 [ms]
116          G_x13 = (-(x2*av1)/((Isc3/y2)**av2-1) +
                      (x1*ar1)/((Isc2/y1)**ar2-1))
117          A_13 = 0.350
118          ViolationTerm_13 = (  (G_x13-A_13)+abs(G_x13-A_13)  )/2
119
120          # y1 > y2-->Corrientes pickup Relé 1 mayor a Relé 2
121          G_x14 = -y1
122          A_14 = -y2
123          ViolationTerm_14 = (  (G_x14-A_14)+abs(G_x14-A_14)  )/2
124
125          # El dial de verde debe ser menor al dial de rojo
126          G_x15 = -x1
127          A_15 = -x2
128          ViolationTerm_15 = (  (G_x15-A_15)+abs(G_x15-A_15)  )/2
129
130
131          cost[i] = (x1*ar1)/((Isc0/y1)**ar21)+
                      (x2*av1)/((Isc1/y2)**av2-1)
                       +(ViolationTerm_1+ViolationTerm_2+
                       ViolationTerm_3+ViolationTerm_4+
             ViolationTerm_5+ViolationTerm_6+ViolationTerm_7
                       +ViolationTerm_8+ViolationTerm_9+
                       ViolationTerm_10+ViolationTerm_11+
                       ViolationTerm_12+ViolationTerm_13+
                       ViolationTerm_14+ViolationTerm_15)*1e5




132      return cost
```

ANEXO V

PowerFactory presenta un manual de comandos en el lenguaje de Python [18]. Este manual permite enlazar PowerFactory con scripts de Python y de esta forma manipular las variables de los objetos codificados en su programación orientada a objetos y realizar acciones que pueden ser repetidas. El barrido de fallas es realizado en el alimentador LOHL632-633 del sistema de prueba, el cual se encuentra modelado en el software de PowerFactory. Mediante la Tabla A5.1 se presenta la función de los principales comandos usados para realizar el barrido de fallas en el sistema de prueba.

Tabla A5.1. Principales comandos usados para simulación de barrido de fallas.

Comando	Función
GetFromStudyCase('ComFunction')	Ejecución de cualquier tipo de simulación
GetCurrentScript()	Acceso del script
ComShc	Cortocircuito
Execute()	Ejecución del cálculo
Object.GetAttribute('Result_variable_name')	Acceso al resultado
PrintPlain(str message)	Impresión en el Output Window

A continuación, se presenta el código de Python que permite realizar el barrido de fallas en el alimentador seleccionado dentro del sistema de prueba en el software de PowerFactory usando los comandos proporcionados por [18].

```
################## Barrido de fallas ###############
"""
Con el barrido de fallas se calcula el tiempo de operación de los dos
relés 51 del sistema de prueba. El barrido calcula la falla en todos los
puntos del alimentador que están separados entre sí por una distancia que
corresponde al 10% de su longitud total.
"""

i = 0
time1 = 0.0
time2 = 0.0
for i in range(11):
    shc = app.GetFromStudyCase("ComShc")
    script = app.GetCurrentScript()
    line =  script.line
    shc.shcobj = line # Selección del alimentador para realizar barrido
    shc.iopt_mde = 3 # Escojo el método
```

```
    shc.iopt_shc = "spgf" # Escojo tipo de falla
    ppro = shc.ppro = i*10 # Escojo la falla cada 10% del alimentador
    shc.Execute() # Se ejecuta falla
    tmpRed= script.tmpRed # Relé 1
    tmpGreen= script.tmpGreen # Relé 2
    time1 = tmpRed.GetAttribute("s:yout") #Tiempo de operación de curva
                                          Relé 1
    app.PrintPlain("Falla "+ str(i+1)+": "+str(ppro)+"%") # distancia de
                                  falla en el alimentador [%]
    app.PrintPlain("Top_R: "+str(time1)) # Se imprime tiempo del relé 1
    time2 = tmpGreen.GetAttribute("s:yout") # Tiempo de operación de curva
                                            Relé 2
    app.PrintPlain("Top_V: "+str(time2)) # Se imprime tiempo relé 2
```

ANEXO VI

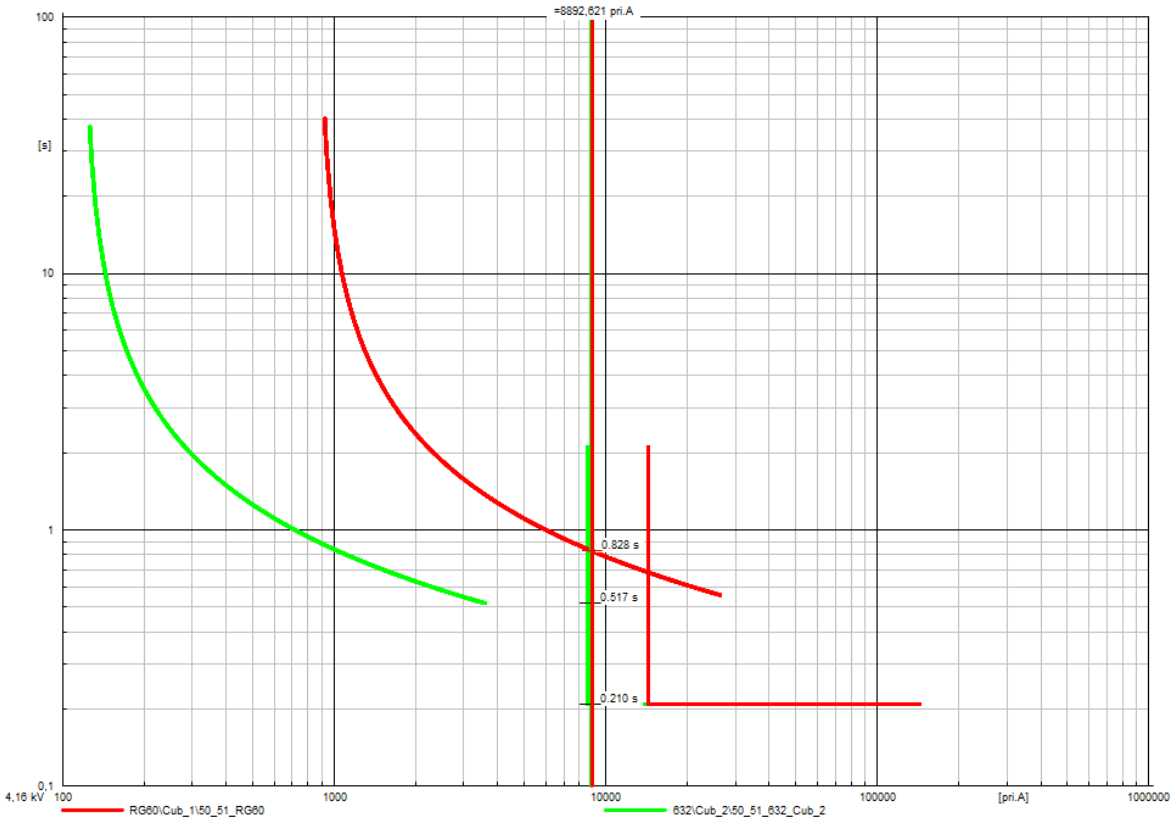


Figura A6.1. Cálculo de falla trifásica sin impedancia de falla al 80% del alimentador LOHL632-633.

Se calcula la corriente de falla trifásica sin impedancia de falla hecha al 80% del alimentador LOHL632- 633, debido a que es la falla más grave en donde el relé _632

actúa como protección principal. En la Figura A6.1 se observan los tiempos de operación de los relés 51_RG60, 51_632 y 50_632 de 828, 517 y 210 [ms], respectivamente. Se observa que se cumple la operación coordinada entre relés al operar primeramente el relé 50_632 como protección principal, después de 307 [ms] opera el relé 51_632 y por último opera el relé 51_RG60 después de 311 [ms]. Cumpliéndose el CTI entre la operación de los relés.

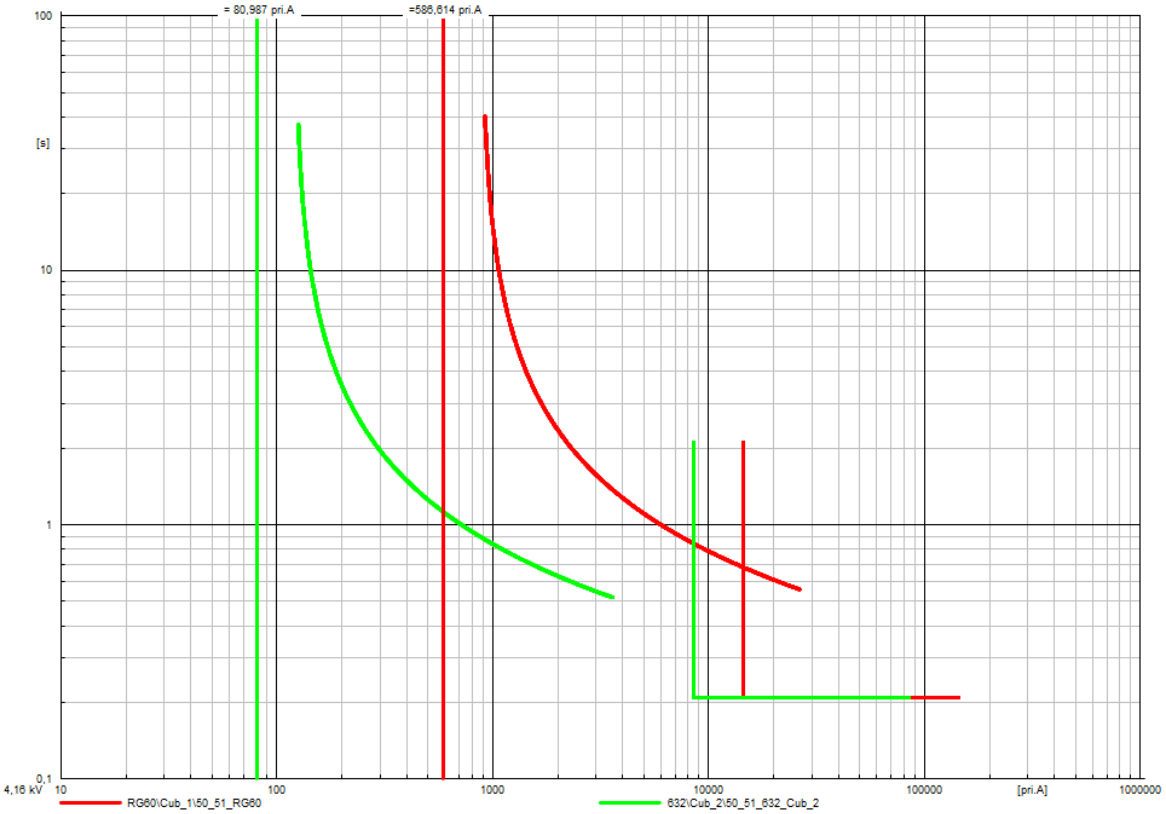


Figura A6.2. Cálculo del flujo de potencia a demanda máxima para implementar límite inferior del ajuste de corriente pickup.

Mediante la Figura A6.2 se muestran las curvas características según el estándar IEC de los dos relés 50/51 incorporados en el sistema de prueba modelado en PowerFactory al calcular las corrientes del flujo de potencias. Las corrientes resultantes del flujo de potencias que pasa por cada relé se muestran mediante líneas verticales de color rojo y verde para el relé RG60 y 632 respectivamente. Estas corrientes del flujo de potencias son usadas como límite inferior de la corriente pickup que se busca optimizar con el algoritmo COA, sin tomar en cuenta el crecimiento de demanda en el sistema de prueba.

En la optimización se toma en cuenta un creciente de demanda del 50% de la demanda actual. Esta consideración se la puede observar gráficamente con las curvas características, las cuales tienen una separación hacia la derecha de la línea vertical de

la corriente de flujo respectiva de cada relé. Y numéricamente se comprueba el cumplimiento de la consideración del crecimiento de demanda mediante el ajuste de la corriente pickup. De modo que, para la curva verde se ajusta la corriente pickup a 120 [A] para una corriente de flujo de aproximadamente 81 [A] y para la curva roja se ajusta la corriente pickup a 880 [A] para una corriente de flujo de aproximadamente 587 [A]. Asegurando la no operación de las protecciones al existir un crecimiento de la demanda de hasta aproximadamente el 50% de su demanda actual.

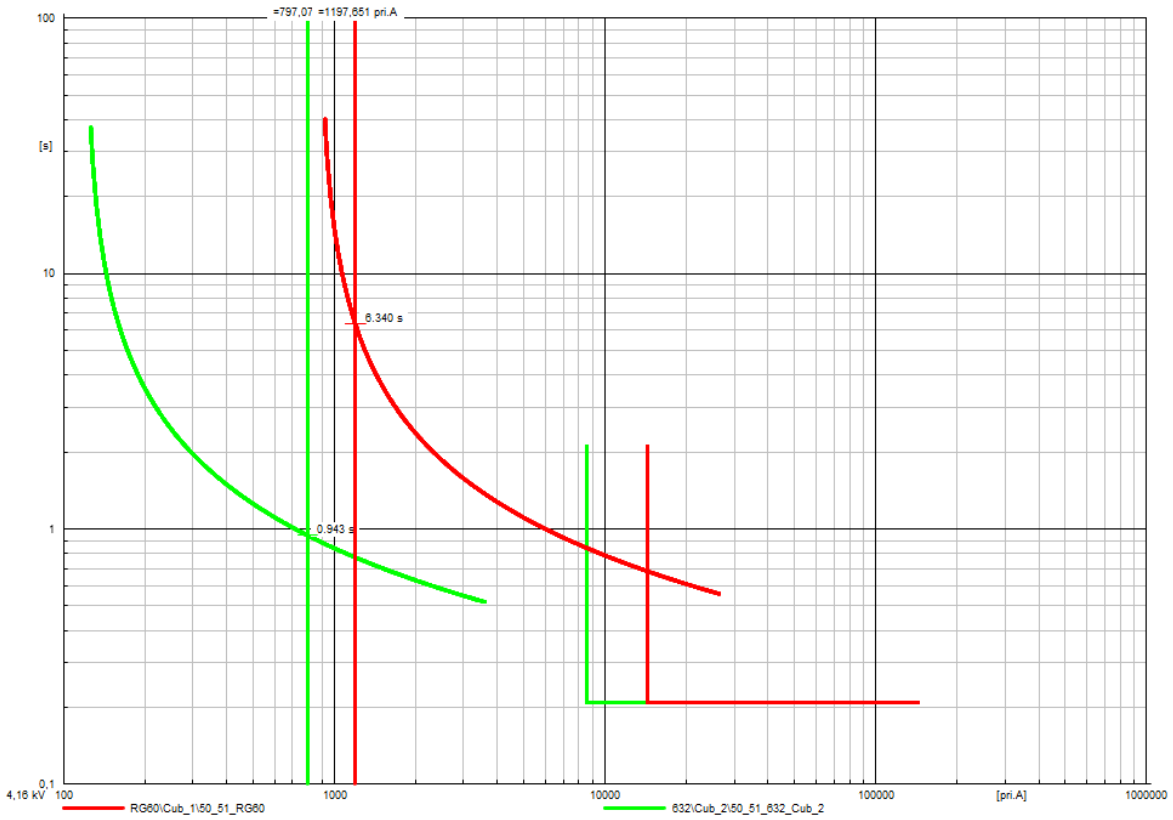


Figura A6.3. Cálculo de falla monofásica con una resistencia de falla de 3 ohmios a demanda mínima para implementar límite superior del ajuste de corriente pickup.

Mediante la Figura A6.3 se observa que el ajuste de las corrientes pickup establecidas para cada relé 51 no superan el límite superior que se ha establecido por el cálculo dado de la falla monofásica con una resistencia de falla de 3 ohmios a demanda mínima al 100% del alimentador LOHL632-633.

ANEXO VII

Escuela Politécnica Nacional
Facultad de Ingeniería Eléctrica y Electrónica

Manual de usuario para la coordinación óptima de relés 51 con la Metodología propuesta.

Realizado por: Daniel Jiménez

Objetivo:

Con este manual de usuario se pretende aclarar los pasos implementados de la metodología propuesta para resolver el problema de optimización, correspondiente a la coordinación óptima de relés 51, usando como método de optimización al algoritmo COA.

Definiciones:

COA (Cuckoo Optimization Algorithm): Este es un algoritmo evolutivo inspirado en la naturaleza que busca variables de decisión que optimicen la solución del problema.

Requisitos:

1. Software de simulación PowerFactory desde la versión PowerFactory 2023 SP3A en adelante.
2. Software de Python con la versión 3.10.

Pasos para ejecutar la metodología:

La carpeta comprimida "*Anexos_TIC_Jiménez_Daniel*" contiene el sistema eléctrico de prueba de 13 nodos de la IEEE con el nombre "*TIC_Sistema Eléctrico_IEEE 13 Node Feede*", modelado en el software PowerFactory. La optimización de la coordinación entre los relés 51 se realiza en el caso de estudio "*Study Detailed Network Model*".

Para que el PowerFactory pueda ejecutar los scripts de Python en la modelación del sistema eléctrico, es necesario realizar la siguiente configuración:

1. En la lista de opciones de la parte superior de la ventana de PowerFactory se selecciona la opción "Tools".
2. Se selecciona la opción "Configurations" en la lista de opciones desplazadas.
3. En la nueva ventana abierta, se elige la opción "External Applications".
4. Se verifica que en "Interpreter" se encuentre seleccionado "selectec by version" y que en "version" se encuentre seleccionado "3.10" dentro del recuadro de Python.
5. En caso de no tener seleccionado como se indica en 4, se procede a hacerlo y se pone "OK".

La carpeta comprimita "*Anexos_TIC_Jiménez_Daniel*" contiene los scripts "*Presentación_Paso2*", "*Presentación_Paso4*" y "*Presentación_Paso5*", desarrollados en Python 3.10. Los tres scripts utilizan dentro de su programación al script que se encuentra con el nombre "*ParametroCurva_IEC*". El script "*ParametroCurva_IEC*" debe estar en la misma carpeta que los tres scripts principales correspondientes al paso dos, paso cuatro y paso cinco, para que puedan ser ejecutados.

Los tres scripts principales utilizan varias librerias dentro de la programación. Es necesario descargarse las librerías dentro de la carpeta de librerías del Python con el comando "pip install Nombre_Librería" en el cmd del computador. Las librerías descargadas para codificar la metodología seguida son las siguientes:

-matplotlib
-numpy
-openpyxl
-pandas
-scikit-learn
-scipy
-sklearn

En el escrito del TIC se presenta la metodología seguida para conseguir el procedimiento de la optimización mediante los pasos que indica la Figura 2.1 en el escrito. El paso uno corresponde a la modelación de los relés de sobre corriente en el sistema de prueba y el ajuste con criterios tradicionales, el documento generado para este paso es el de "*TIC_Sistema Eléctrico_IEEE 13 Node Feede*", realizado en PowerFactory. El paso dos es la programación del script "*Presentación_Paso2*" correspondiente algoritmo COA codificado en Python y puesto a prueba por las funciones objetivo de prueba.

Se consigue resolver el problema de coordinación óptima de relés de sobre corriente en el script "*Presentación_Paso4*". Se vincula el sistema de prueba (Python) con el algoritmo COA(PowerFactory) en el script "*Presentación_Paso5*" para automatizar los cálculos y obtener mejores resultados de optimización que en el paso cuatro. El paso tres corresponde al planteamiento del problema de optimización, por esta razón, este paso está implícito en el cuarto y quinto paso.

--- Script "*Presentación_Paso2*": Permite ejecutar el código del algoritmo COA para resolver los ocho problemas de optimización de las funciones de prueba propuestas en la sección 2.2 del escrito del TIC, desde Python.
--- Script "*Presentación_Paso4*": Permite ejecutar el código del algoritmo COA para resolver el problema de optimización de la función objetivo que resuelve la coordinación óptima de relés 51 propuesta, mostrada en la sección 2.4 del escrito del TIC. Este script se ejecuta desde Python.
--- Script "*Presentación_Paso5*": Permite ejecutar el código del algoritmo COA para resolver el problema de optimización de la función objetivo que resuelve la coordinación óptima de relés 51 propuesta con la vinculación del sistema eléctrico, el procedimiento de la optimización es mostrado en la sección 2.5 del escrito del TIC. Este script se ejecuta desde PowerFactory.

Nota: PowerFactory debe tener configurado como aplicación externa a Python 3.10 para que pueda ser ejecutado el script de Python desde PowerFactory.

Partes del código que debe ser cambiado para ser ejecutado en su computador:

- En los scripts del paso 4 y paso 5 se presentan gráficas de los resultados obtenidos, estas gráficas almacenadas en una carpeta correspondiente al computador que esté ejecutando el programa. Caso contrario el código tendrá errores en su ejecución. En el caso del paso 4 se debe cambiar la dirección de almacenamiento en las líneas del código 503 y 520. Para el caso del paso 5 se debe cambiar la dirección de almacenamiento en las líneas del código 795 y 819.

- En el script del paso 5 se debe cambiar la dirección del PowerFactory que se encuentra instalado en el computador, se encuentra en la línea 26 del código correspondiente.

Para introducir los parámetros de ajuste óptimo del paso 4 en el sistema eléctrico en PowerFactory, se realiza un script que pone los parámetros al ejecutar el script "Ajuste_SistemaEléctrico_Paso4", dentro de Powerfactory.

Los parámetros óptimos conseguidos del procedimiento de la optimización del paso 5 son graficados mediante el script "*GráficaResultados_COA*". El script "*GráficaResultados_COA*" se ejecuta solo en el mismo Python.