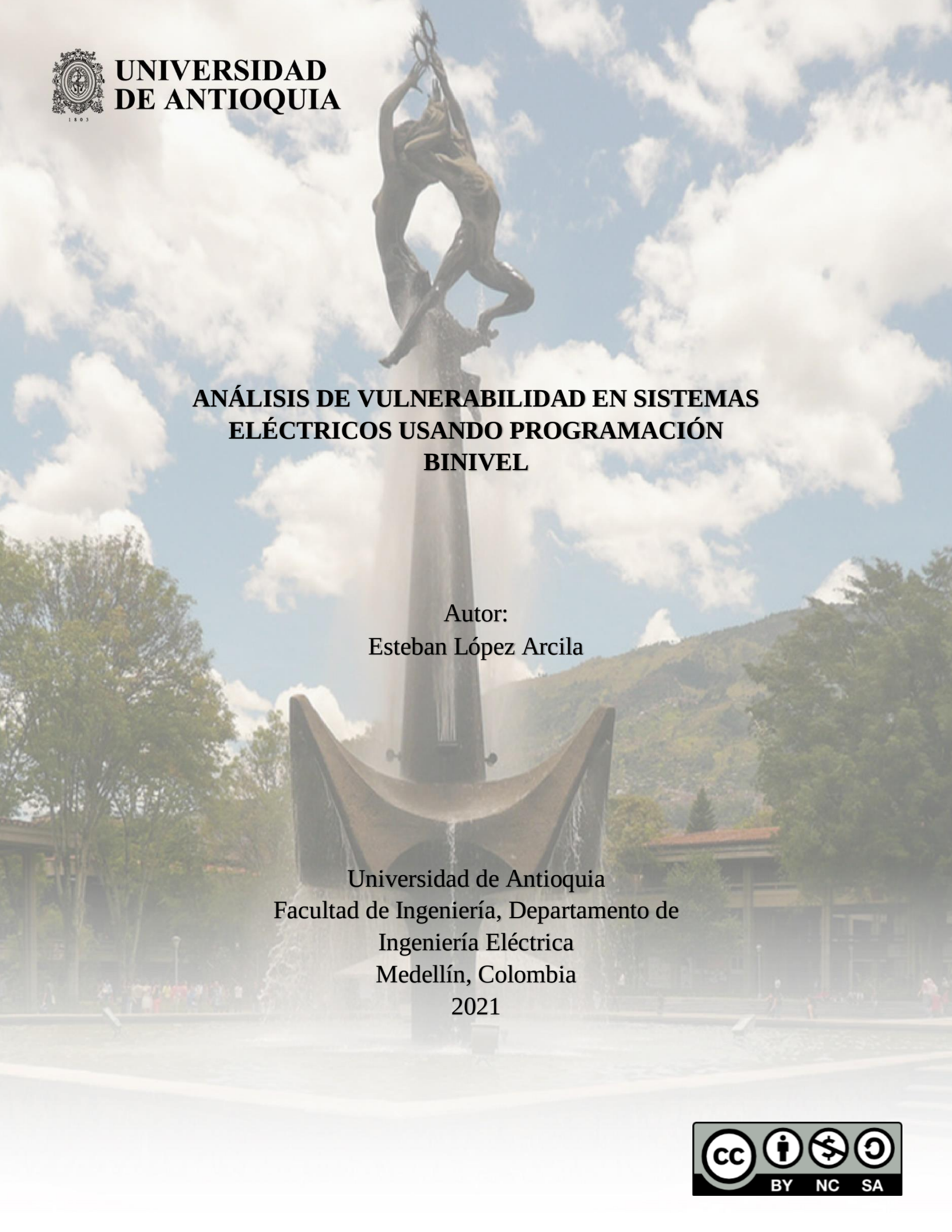

UNIVERSIDAD DE ANTIOQUIA

ANÁLISIS DE VULNERABILIDAD EN SISTEMAS ELÉCTRICOS USANDO PROGRAMACIÓN BINIVEL

Autor:
Esteban López Arcila

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica
Medellín, Colombia

2021

Análisis de Vulnerabilidad en Sistemas Eléctricos Usando Programación Binivel

Esteban López Arcila

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:
Ingeniero Electricista

Asesor:
Jesús María López Lezama, Doctor en Ingeniería Eléctrica

Línea de Investigación:
Sistemas de Potencia
Grupo de Investigación:
Grupo de Investigación en el Manejo Eficiente de la Energía (GIMEL)

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica
Medellín, Colombia
2021

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ANÁLISIS DE VULNERABILIDAD EN SISTEMAS ELÉCTRICOS USANDO PROGRAMACIÓN BINIVEL

Resumen

En esta investigación se analizó la vulnerabilidad de distintos sistemas eléctricos tanto de potencia como de distribución. Estos sistemas actuaban bajo escenarios de operación segura, cada sistema constaba de 24, 32 y 69 barras y, 38, 37 y 74 líneas de transmisión correspondientes, estos sistemas son conocidos en la literatura como: sistema de potencia IEEE RTS de 24 barras, sistema de distribución IEEE de 32 barras y sistema de distribución IEEE de 69 barras. Cada uno de estos sistemas actuaban bajo un escenario de operación segura diferente, para maniobrar en casos de contingencias. Es preciso mencionar que no se tuvieron restricciones de línea; es decir, no se tomaron en cuenta límites de potencia para la transmisión de energía en las líneas.

En el primer nivel se programó un agente mal intencionado que simuló ataques a líneas teniendo en cuenta una cantidad de recursos disponibles; esto es, se consideró un límite de líneas que podían ser atacadas. Estos ataques se hacían de forma aleatoria debido que la cantidad de casos posibles de irrupciones se volvían ingentes e imposibles de abordar. El agente generó diferentes ataques que pretendían buscar el escenario con mayor deslastre.

En segundo nivel se programó un agente de defensa que intenta maniobrar recursos disponibles para minimizar el daño causado por el ataque del agente maligno. Al agente defensor se le entregó cada uno de los escenarios de ataque generados por el agente maligno para crear una respuesta con los recursos disponibles operables; estos eran, las líneas que estaban en principio inactivas o normalmente abiertas. Se escogieron de forma aleatoria las líneas que se ponían en operación. El defensor calculó de nuevo el deslastre para el escenario correspondiente, buscando el caso posible con menor carga desatendida y mitigar el daño en respuesta al ataque.

Como resultado de esta investigación se obtuvo una serie de vectores de ataque que maximizan el deslastre de carga y una serie de vectores de reacción que intentaron mitigar estos ataques sobre los sistemas IEEE RTS 24 de potencia, IEEE 32 de distribución e IEEE 69 de distribución. Estos se obtuvieron mediante dos heurísticas, algoritmo genético y un torneo.

Introducción

En la actualidad, la electricidad se ha vuelto un recurso indispensable para la cotidianidad; esta ha facilitado innumerables aplicaciones agrícolas e industriales entre otras, permitiendo un mejor aprovechamiento del tiempo. Garantizar el suministro de este recurso se ha vuelto una tarea de interés público y gubernamental.

Dada la importancia de mantener la continuidad del suministro de fluido eléctrico, surge la necesidad de analizar la vulnerabilidad de los sistemas eléctricos ante pérdidas de elementos como líneas de transmisión, transformadores, generación, etc. Ante estos eventos surgen las siguientes preguntas: ¿Cuál es la mejor operación del sistema para evitar el máximo deslastre de carga? ¿Qué elementos se deben reforzar para mejorar la seguridad del sistema? ¿Cuáles pueden ser las posibles medidas a tomar ante la pérdida de un elemento del sistema? Este proyecto se centró, específicamente en dar respuesta a la vulnerabilidad ocasionada por la pérdida de líneas y transformadores y, también en dar los posibles planes de mitigación de ésta al Operador del Sistema (OS), por ejemplo, determinar las líneas que al salir del sistema causen el máximo deslastre de carga, para dar posibles sugerencias al OS.

Dado que los sistemas eléctricos representan una gran inversión pública, el deslastre de carga representa enormes pérdidas monetarias, no solo por la pérdida de los equipos, sino también por las pérdidas económicas debidas a la desatención de la carga desconectada [1]. El desarrollo de estrategias para minimizar la carga desatendida ante eventuales fallas o ataques intencionales es de vital interés para los OS. Los análisis de vulnerabilidad son una forma de elaborar dichas estrategias; la importancia de su elaboración radica en la oportuna intervención del OS, mejorando o reforzando los elementos más vulnerables antes de que fallen o sean dañados por un agente atacante.

El problema de interdicción en los sistemas de potencia fue propuesto por primera vez en [2], el cual consiste en un análisis de vulnerabilidad introduciendo un nuevo concepto: la intencionalidad. En este caso existen dos agentes con funciones objetivos en conflicto. El objetivo del agente disruptor es causar el mayor deslastre de carga posible al sistema; mientras que el OS deberá responder a cualquier ataque para minimizar el deslastre de carga. El agente disruptor posee recursos limitados y debe escoger el conjunto de elementos a atacar, anticipándose a la reacción del OS. La interacción de estos dos agentes se puede modelar de forma jerárquica, dando origen a un problema de programación binivel [2].

El problema de interdicción de la red eléctrica (EGIP), es altamente no convexo y no lineal por naturaleza, algunos autores abordan este problema usando modelos lineales de la red y replanteando equivalentes matemáticos del problema de un solo nivel, solucionando un problema de programación

de enteros mixtos. En este trabajo se propuso utilizar un modelo topológicamente igual a cada circuito eléctrico de prueba, considerando conectividad en la red y generación distribuida para mitigar el deslastre de carga. La no convexidad y no linealidad del problema en cuestión dificulta su solución por métodos exactos, lo que lleva a emplear métodos heurísticos que, si bien no entregan la solución más óptima o general, si dan respuestas adecuadas al problema en tiempos de computo razonables.

Objetivos

1. Objetivo general

Solucionar el problema de vulnerabilidad de un sistema eléctrico de N barras determinando los elementos que al fallar ocasionan el mayor impacto (racionamiento), para sugerir los posibles refuerzos al sistema.

2. Objetivos específicos

- Revisar los modelos y métodos de solución aplicados al problema de vulnerabilidad en la literatura técnica.
- Validar diferentes técnicas heurísticas y meta-heurísticas de solución para el problema de vulnerabilidad.
- Aplicar una técnica meta-heurística de solución al problema de vulnerabilidad y comparar con resultados publicados.

Marco Teórico

Los sistemas eléctricos de potencia pueden estar expuestos a ataques de origen intencional. El sistema eléctrico colombiano ha sufrido más de 200 ataques terroristas desde el año 1999 [3]. Por esto, determinar la vulnerabilidad de los sistemas eléctricos adquiere cada día mayor importancia.

Mediante la implementación de metodologías para su análisis, es posible estudiar las consecuencias de diferentes eventos. Una contingencia se produce cuando uno o más elementos salen de operación. Normalmente los sistemas deben soportar este tipo de eventos conocidos como contingencias N-1 y N-2. Es decir, después de la salida de un elemento del sistema (o dos), este debe regresar a un estado de estabilidad en un nuevo punto de operación [2].

Por su naturaleza, los problemas de interdicción son abordados desde una perspectiva de programación binivel, la cual permite definir funciones objetivo diferentes para el agente atacante y defensor. Adicionalmente, las restricciones del nivel superior, están en función de las variables de optimización del nivel inferior. En general, un modelo binivel es un problema

de optimización en el cual una de sus restricciones es otro problema de optimización [4].

En [5] se aborda la problemática de interdicción hallando las líneas más críticas para un sistema eléctrico mediante la implementación de un algoritmo genético, la efectividad de este se midió con el deslastre de carga que se dejó de atender y, considerando que el agente atacante posee un recurso económico limitado para hacerlo. En [2] se propone un modelo de interdicción distinta el cual le permite al agente atacante variar su función objetivo, para así verificar la efectividad de estas.
Los criterios tradicionales de tipo N-1 y N-2, permiten evaluar escenarios de operación segura (dentro de sus márgenes permitidos). El inconveniente radica en que solo los cortes al azar de origen natural son considerados en el conjunto de contingencias. Desafortunadamente, las múltiples contingencias son cada vez más frecuentes. Además, los sistemas de energía están expuestos no solo a fallos aleatorios, sino además provocados (ataques terroristas) que se caracterizan distintivamente por sus malas intenciones. Por lo tanto, estos criterios no son suficientes para evaluar la vulnerabilidad [6].

En [4], se plantea un modelo Min-Min, en donde la función objetivo del agente atacante consiste en minimizar el número de elementos de la red a atacar, sujeto a que la demanda desatendida sea igual o mayor a un valor esperado. Este modelo de interdicción es resuelto mediante programación lineal entera mixta (PLEM), al convertir las no linealidades del nivel inferior en restricciones lineales, mediante las condiciones de optimalidad de Karush Kuhn Tucker (KKT).
En [7], se plantea el problema de interdicción mediante un modelo de optimización Max-Max, donde la función objetivo del agente atacante es maximizar el deslastre de carga en el sistema. En este problema se vuelven lineales las restricciones del nivel inferior mediante técnicas de dualidad y se plantea un problema equivalente de PLEM.

En [8], se propone un nuevo modelo de interdicción que permite definir diferentes funciones objetivo para el agente atacante y defensor. Además, dicho modelo permite incluir restricciones en el nivel superior que dependan de las variables del nivel inferior.

En general el objetivo del agente disruptor es maximizar el deslastre de carga como se indica en la ecuación (1); donde ΔP_n^d es la carga deslastrada en la barra *n*. El índice inferior *n* indica el número de la barra, mientras que el índice superior *d*, se refiere a la demanda, *IV* es el vector de interdicción binaria y *N* es el conjunto de barras. El tamaño del vector de interdicción es igual al número de líneas. Las entradas del vector de interdicción toman el valor uno si la línea correspondiente está en servicio y cero si está bajo ataque. La ecuación (2) indica el límite de recursos destructivos, donde *M* es el número de líneas bajo ataque e *IV* es la primera entrada del vector de interdicción. La

ecuación (3) indica la naturaleza del vector de interdicción de entrada y la ecuación (4) representa la reacción del operador del sistema.

$$\underset{IV}{\text{Max}} \sum_{n \in N} \Delta P_n^d ; \qquad \forall n \in N \tag{1}$$

Restricciones:

$$\sum_{l \in L} (1 - IV_l) = M; \qquad \forall l \in L \tag{2}$$

$$IV_l \in \{0,1\} \tag{3}$$

$$\text{Min} \sum_{n \in N} \Delta P_n^d ; \qquad \forall n \in N \tag{4}$$

Metodología

Las meta-heurísticas, en su definición original, son métodos de solución que dirigen una interacción entre los procedimientos locales de mejora y estrategias de alto nivel, para generar procesos capaces de evitar los óptimos locales y realizar búsquedas robustas de un espacio de soluciones. Varias de las herramientas y mecanismos que han surgido de la creación de métodos meta-heurísticos han demostrado ser efectivas, tanto es así que las meta-heurísticas se han vuelto el centro de atención en los últimos años como la línea de ataque preferido para resolver muchos tipos de problemas complejos, principalmente los de naturaleza combinatoria.

Cuando se trata de problemas no convexos multimodales, se prefiere el uso de técnicas meta-heurísticas sobre metodologías de optimización clásicas. Ejemplos de aplicaciones exitosas de meta-heurísticas empleadas para resolver problemas de programación de dos niveles se pueden encontrar en [9] y [10].

Además, algunas de las aplicaciones más exitosas de métodos exactos se han producido mediante la incorporación de estrategias meta-heurísticas dentro de ellos. Estos resultados han motivado la investigación y aplicación de nuevas metodologías y meta-heurísticas mejoradas [11].

Por todo lo anterior, las heurísticas son una de las formas de abordar problemas no convexos, dando buenos resultados siempre y cuando sean bien ajustados estos algoritmos, para poder implementar alguna metodología de análisis, primero se debe tener la forma de calcular el deslastre de carga en un sistema eléctrico. Esquemáticamente hablando se podría decir que un grafo es similar

a un sistema eléctrico, el cual transporta energía en forma de electricidad, siendo así; en la primera etapa de este trabajo se investigaron algoritmos que especifiquen la forma de recorrer un grafo, dentro de los cuales se planteó la implementación del algoritmo "Búsqueda primero en amplitud" (BFS por sus siglas en inglés *Breadth First Search*). Este algoritmo permitió hacer el recorrido de los distintos grafos que representaban cada sistema eléctrico.

Posteriormente se programaron una serie de métodos necesarios para simular las funciones del agente mal intencionado; los cuales se encargaban de Generar ataques aleatorios y hacer búsquedas locales para mejorar cada solución planteada, la búsqueda local consiste en la elección de dos individuos de cada vector de ataque, los cuales se reemplazan por sus vecinos y se calcula nuevamente el deslastre, esto se ilustra mejor en la figura 1. Luego se verifico que cada uno de los algoritmos cumpliera su función tanto en Python como en Matlab y que estos a su vez tuvieran coherencia entre ellos.

Más adelante se programaron los métodos que componen las funciones del agente defensor, los cuales se encargaban de generar vectores aleatorios de activación de líneas y, hacer búsquedas locales para mejorar cada solución planteada, en este caso la búsqueda local consiste en la elección de dos individuos de cada vector de activación o defensa, los cuales se reemplazan por sus vecinos y se calcula nuevamente el deslastre, intentando disminuir la carga desatendida, esto se ilustra mejor en la figura 1. Se armonizó este grupo de métodos para dar en conjunto una respuesta a los ataques dados como entrada. Luego se verifico su validez comparando las respuestas de cada lenguaje y así garantizando la coherencia entre los algoritmos de cada lenguaje.

Se programó una heurística llamada "Torneo", encargada de acoplar en su conjunto la totalidad de los algoritmos de cada agente. Este algoritmo basó su lógica en la estructura de un torneo, en la que se colocan a competir posibles soluciones al problema de la búsqueda del mejor vector de interdicción y, posibles candidatos a mejor respuesta de defensa. En la primera parte del torneo se generan posibles candidatos a mejor vector de interdicción, luego se hace una búsqueda local para cada candidato y, se escoge el mejor candidato, después, se generan de forma aleatoria defensas para este vector de interdicción, se realiza una búsqueda local de los candidatos a mejor defensa y, se escoge el candidato que logra reducir en mayor medida el deslastre de carga, termina el torneo y se almacena la respuesta de los candidatos. Se realizan varios torneos y se retornan los finalistas de estos torneos. En total se guardaron y se analizaron para su comparación en cada uno de los lenguajes (Python y Matlab) 10 candidatos en cada sistema de prueba.

Por último, se compararon gráficamente las respuestas de cada uno de los algoritmos implementados en cada lenguaje en el sistema de potencia IEEE-RTS-24 y sistemas de distribución IEEE-32 e IEEE-69; en Visio se implementaros las configuraciones de cada uno de los sistemas con sus correspondientes medidas preliminares.

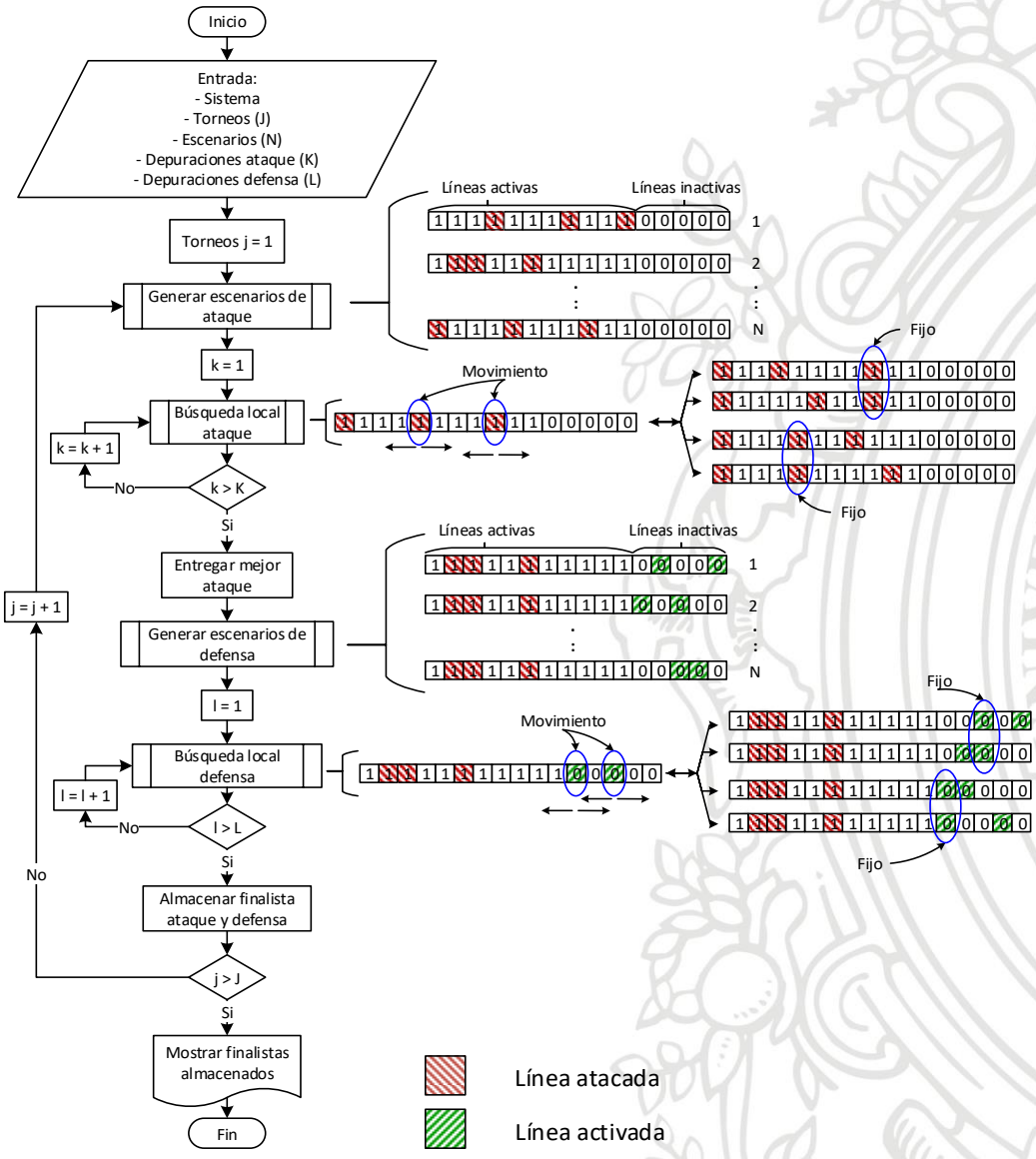

Fig. 1. Método "Torneo".

Resultados y análisis

Rendimiento de los métodos que componen al agente mal intencionado, para esta comparación se realizaron pruebas en el sistema eléctrico IEEE -RTS-24, el cual consta de 24 barras y 38 líneas. Este sistema se encuentra bajo un

escenario de operación segura, en el cual tiene normalmente abiertas las líneas 8, 12, 14, 15, 18, 21, 23, 24, 25, 31, 32, 34, 36, 11, 17, 20, 5, 22 y 1, las cuales se encuentran marcadas con una línea punteada azul en la figura 2. A este agente se le dio como insumo un máximo de 4 ataques y, se le pidió que generase 40 escenarios posibles de ataque, de estos se seleccionó el ocasionador de mayor deslastre; el resultado de esta comparación se plasmó en la tabla 1.

Tabla 1. Rendimiento algoritmo ataque.

COPARACION ATAQUE	**DESLASTRE [MW]**	**LINEAS ATACADAS**	**TIEMPO DE EJECUCION [s]**
PYTHON	1312	10, 27, 37, 30	0.0070
MATLAB	1414	9, 26, 30, 37	0.9160

Como se observa de la anterior tabla ambos algoritmos encontraron resultados similares en cuanto al deslastre, pero el algoritmo en Python lo hizo en un tiempo muchísimo menor (aproximadamente 7 [ms]), esto puede deberse a la arquitectura del algoritmo empleada en cada lenguaje y al rendimiento de cada lenguaje.

El rendimiento de los métodos que componen el agente defensor, bajo las mismas condiciones en el sistema IEEE-RTS-24 anteriores, se evaluó dándole como insumo cada uno de los ataques generados aleatoriamente por el agente mal intencionado de la primera parte, incluyendo el candidato que generó el mayor deslastre para cada lenguaje, a este agente solo se le permitió poner en funcionamiento 2 líneas dando como resultados los mostrados en la tabla 2.

Tabla 2. Rendimiento algoritmo defensa.

COPARACION DEFENSA	**NUEVO DESLASTRE [MW]**	**LINEAS PUESTAS EN OPERACION**	**TIEMPO DE EJECUCION [s]**
PYTHON	1312	17, 32	0.0060
MATLAB	1045	11, 31	0.7987

De la anterior tabla podemos extraer que, ambos algoritmos no retornan una buena respuesta ante los ataques generados anteriormente, pero lo importante de este resultado es observar el rendimiento de cada algoritmo, dando como resultado una clara ventaja en el algoritmo implementado en Python, debido a su menor tiempo de computo.

Para los resultados siguientes como criterio de validez de estos se le dio mayor importancia primero que todo al máximo deslastre después de generada la defensa (figuras 6, 7 y 8); este resultado nos indicaría que a pesar de las

acciones de defensa el algoritmo no logro encontrar una mejor estrategia de defensa (optimo local) o, que no hay mejor estrategia y, segundo que todo, se le dio importancia a la mayor reducción del deslastre (figuras 3, 4 y 5), este resultado nos indicaría la mejor estrategia de defensa para un determinado ataque.

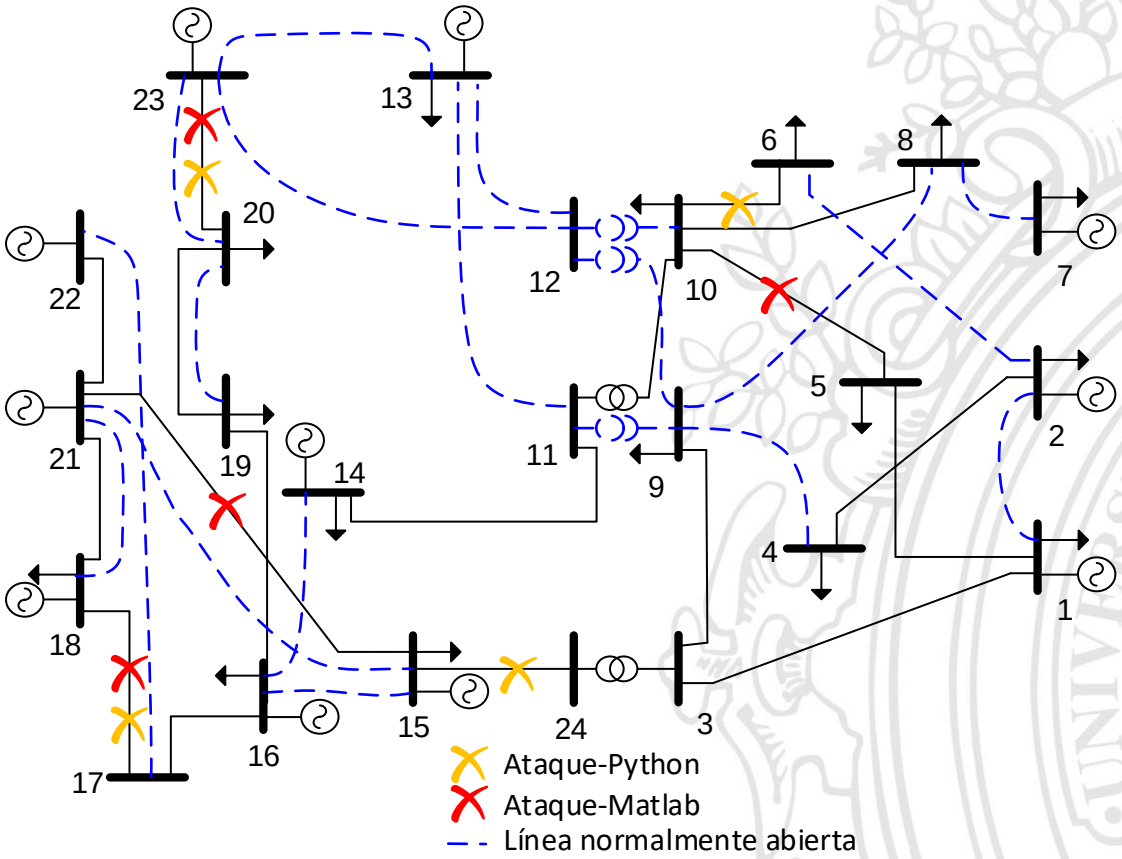

Fig. 2. Prueba de agentes malintencionados.

En la implementación de los algoritmos en los sistemas IEEE-RTS-24 de potencia, IEEE-32 de distribución e IEEE-69 de distribución; se dio como insumos al método "Torneo", un máximo de 6 ataques al sistema, 3 operaciones de defensa y, que generase en cada torneo 20 participantes, también se le dijo que realizara 2 búsquedas locales para todos los participantes de cada torneo, tanto en el ataque como en la defensa, en total se realizaron 10 torneos, de los finalistas se escogió aquel que generó una mayor reducción del deslastre para cada lenguaje (tablas 3, 4 y 5) y aquel que a pesar de conmutar líneas inoperantes a operantes, no logró reducir en mayor medida el deslastre ocasionado por el ataque para cada sistema de prueba (tablas 7, 8 y 9).

Tabla 3. Mayor reducción de deslastre en sistema IEEE-RTS-24.

SISTEMA-POTENCIA IEEE-RTS-24	PYTHON	MATLAB
LINEAS ATACADAS	37, 10, 26, 29, 30	4, 10, 26, 29, 37
DESLASTRE EN ATAQUE [MW]	1469	1543
RESPUESTA DEFENSA	25, 11, 36	11, 23, 36
DESLASTRE EN DEFENSA [MW]	278	297
REDUCCION DESLASTRE [MW]	1191	1246
TIEMPO DE EJECUCION [s]	0.4667	59.7378

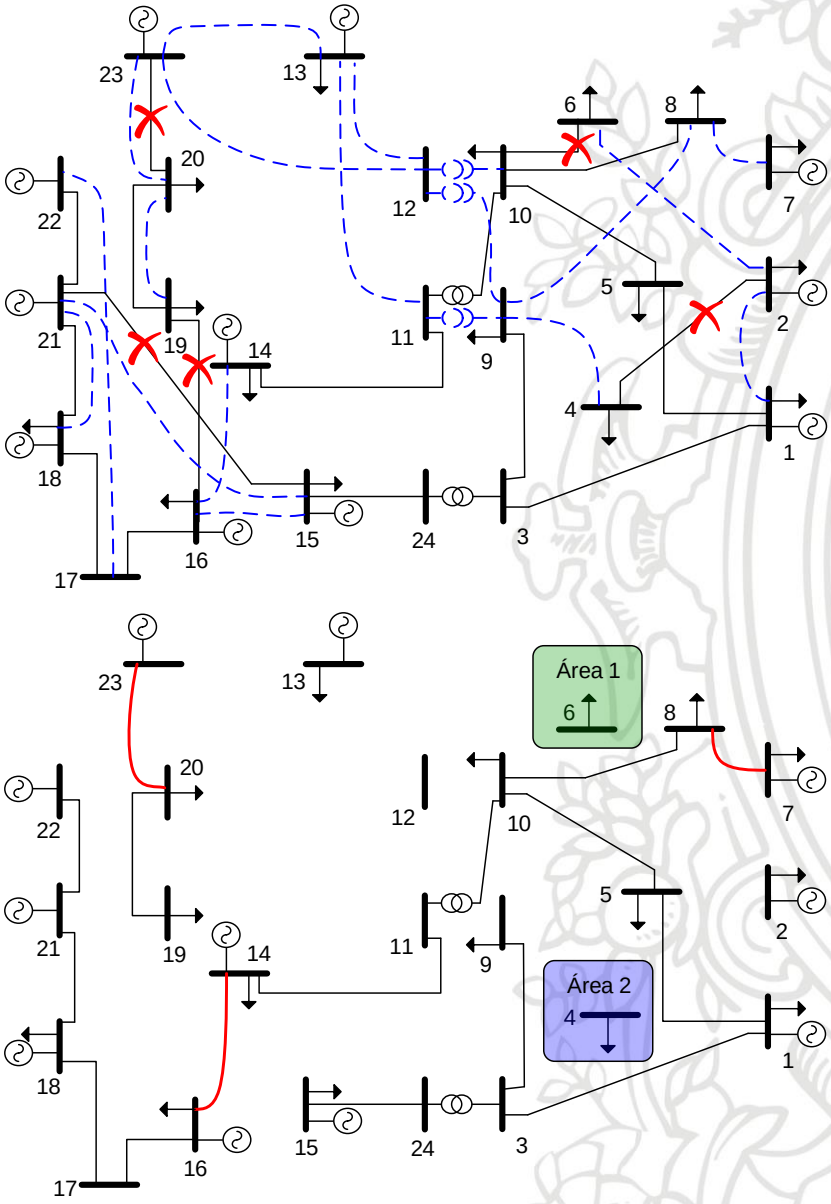

Fig. 3. Mayor reducción de deslastre IEEE-RTS-24.

De las gráficas podemos evidenciar que algunas áreas han quedado por completo desconectadas de la red, el algoritmo ha visto que es mejor

reconectar el resto del sistema, antes que estas zonas, y es lógico, son solo unas cargas en comparación con el sistema entero (fig. 3).

Tabla 4. Mayor reducción de deslastre en sistema IEEE-32.

SISTEMA-DISTRIBUCIÓN 32 BARRAS	PYTHON	MATLAB
LINEAS ATACADAS	3, 8, 15, 22, 29, 30	1, 4, 6, 21, 25, 27
DESLASTRE EN ATAQUE [KW]	3255.4841	3635.7
RESPUESTA DEFENSA	33, 35, 37	35
DESLASTRE EN DEFENSA [KW]	1327.6698	3392.2
REDUCCION DESLASTRE [KW]	1927.8143	243.5114
TIEMPO DE EJECUCION [s]	0.3748	59.9106

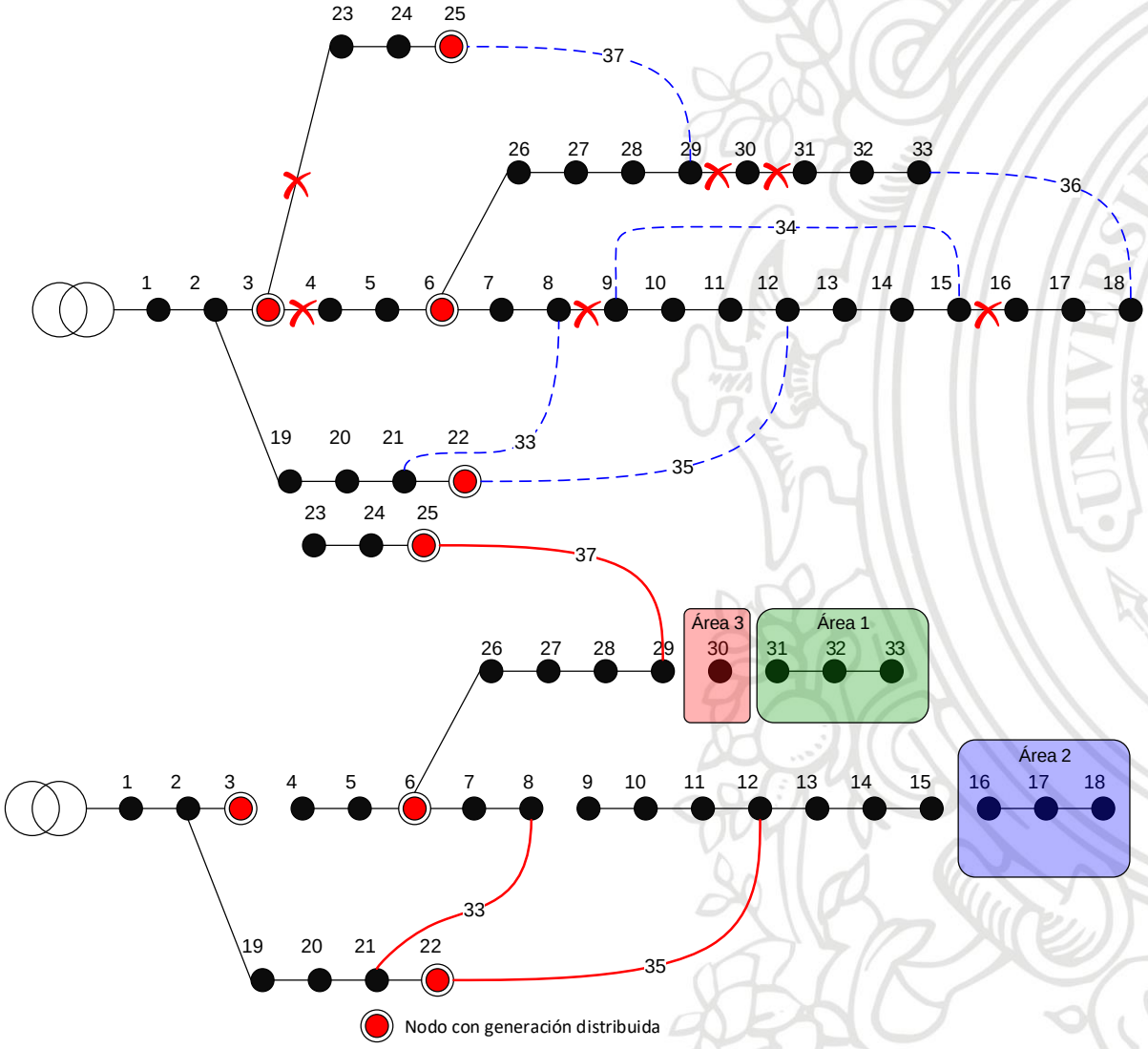

Fig. 4. Mayor reducción de deslastre IEEE-32.

Para el sistema IEEE-32 el algoritmo ha encontrado la forma de reconectar la mayor parte de la red con el nodo de alimentación (nodo 1). Las áreas 1 y 2 (verde y azul) a pesar de que, si se reconectan, no se disminuiría el deslastre.

Tabla 5. Mayor reducción de deslastre en sistema IEEE-69.

SISTEMA-DISTRIBUCIÓN 69 BARRAS	PYTHON	MATLAB
LINEAS ATACADAS	4, 19, 51, 55, 27, 61	7, 27, 31, 33, 47, 51
DESLASTRE EN ATAQUE [KW]	1203.2688	1142.5
RESPUESTA DEFENSA	72, 74, 71	70, 73
DESLASTRE EN DEFENSA [KW]	553.2235	34.4658
REDUCCION DESLASTRE [KW]	650.0453	1108
TIEMPO DE EJECUCION [s]	0.6416	109.1195

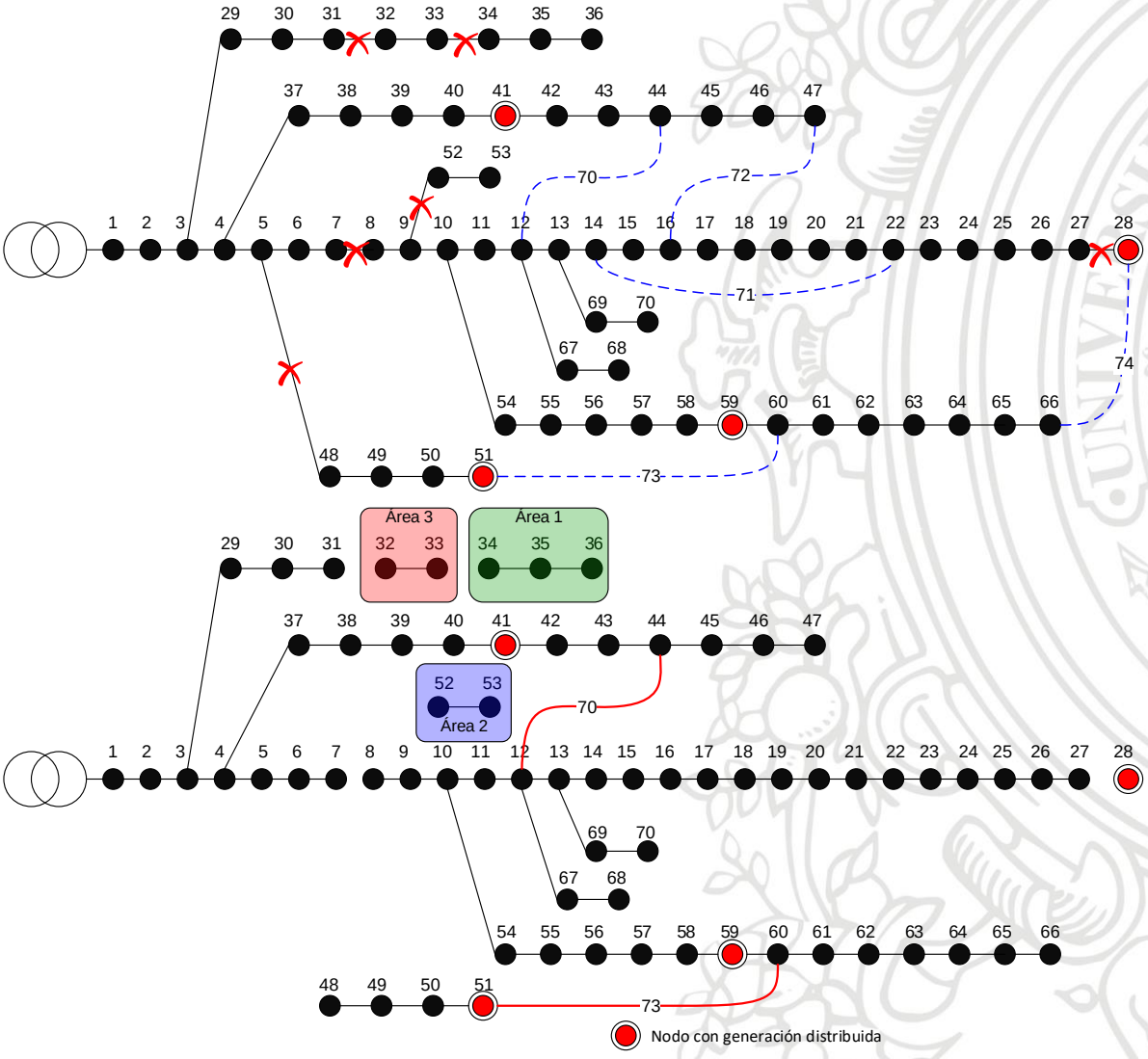

Fig. 5. Mayor reducción de deslastre IEEE-69.

De la figura 5 vemos que no hay forma de reconectar las áreas 1, 2 y 3, el efecto de las líneas 70 y 72 es equivalente, la línea 71 no tendría utilidad en operación para este caso, ya que une dos zonas del sistema que no están desconectadas; esta sería una solución viable (operar línea 71) para el problema de interdicción considerando restricciones de línea. El nodo 28 ha quedado aislado por completo de la red, pero este nodo posee generación distribuida y esta es capaz de suplir su demanda, por lo tanto, el algoritmo no vio necesario operar la línea 74, ya que las barras que une esta línea con la barra 28, están siendo suplidas por el nodo de alimentación (nodo 1).

En la tabla siguiente se muestra un resumen de las estadísticas de los algoritmos en cada lenguaje y para cada sistema de prueba, en general se calculó la reducción promedio del deslastre, entre el ataque y la defensa. Como podemos ver, en promedio el algoritmo en Matlab logro obtener una mayor reducción del deslastre en promedio, pero Python tardo en todos los casos menos tiempo en ejecutar la heurística.

Tabla 6. Reducción promedio del deslastre.

ALGORITMO / SISTEMA ELÉCTRICO	PYTHON	MATLAB
IEEE-24	971.23 [MW]	1024.3 [MW]
32 BARRAS	361.4777 [KW]	243.5114 [KW]
69 BARRAS	249.5792 [KW]	326.9768 [KW]

Para las siguientes pruebas (Tablas 7, 8 y 9, figuras 6, 7 y 8) se consideró el mayor deslastre sobre cada sistema, después de defenderse como se mencionó anteriormente. Este criterio de medición nos permitió encontrar aquellos escenarios para los que no se pudo defender el sistema a pesar del ataque, ayudándonos a determinar qué líneas son críticas para el sistema.

Tabla 7. Mayor deslastre en sistema IEEE-RTS-24 ante defensa.

SISTEMA-POTENCIA IEEE-24	PYTHON	MATLAB
LINEAS ATACADAS	4, 37, 7, 26, 28	4, 9, 13, 26, 30, 37
DESLASTRE EN ATAQUE [MW]	1488	1488
RESPUESTA DEFENSA	17, 21, 25	8, 24, 36
DESLASTRE EN DEFENSA [MW]	726	828
REDUCCION DESLASTRE [MW]	762	660
TIEMPO DE EJECUCION [s]	0.4667	59.7378

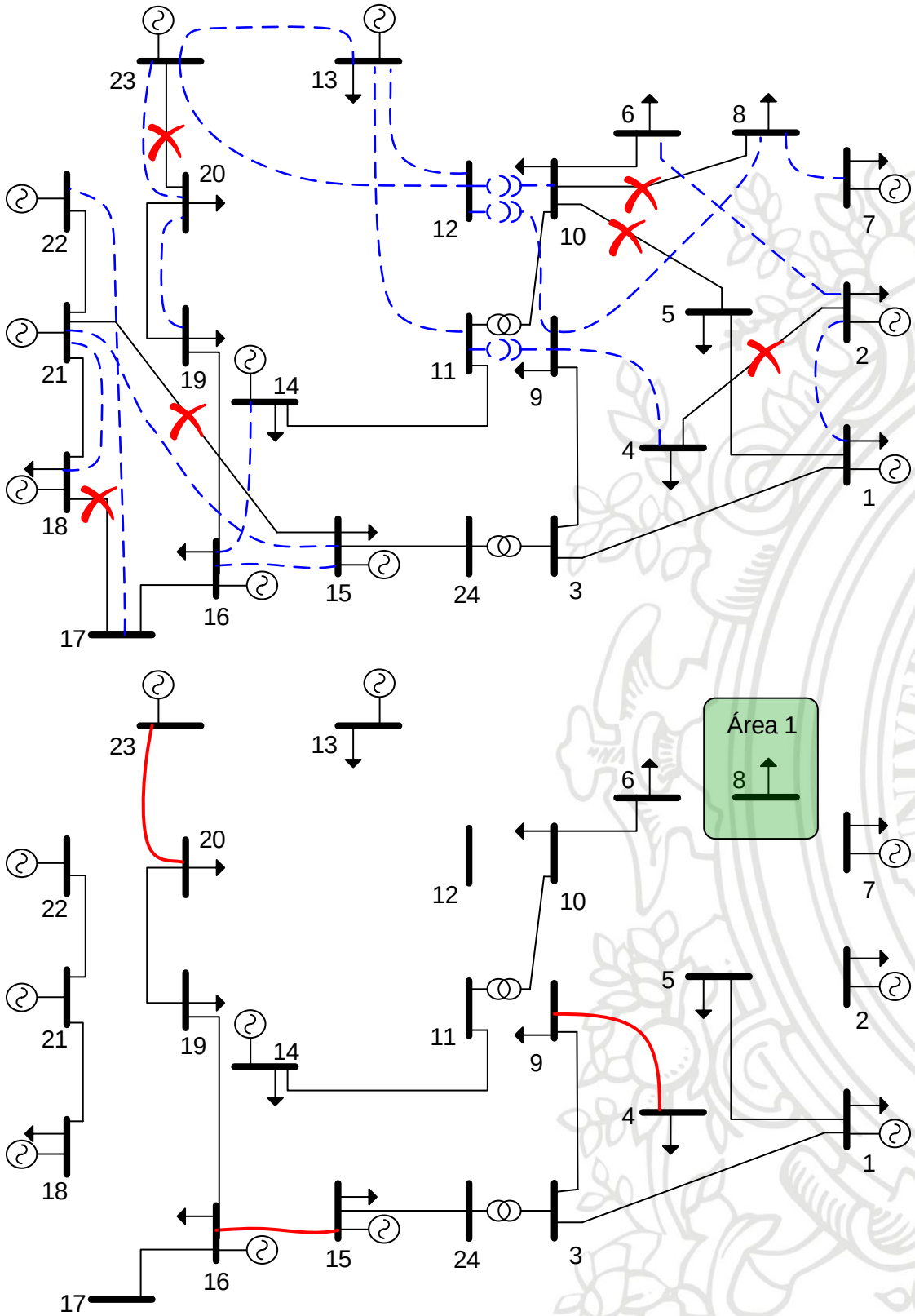

Fig. 6. Mayor deslastre después de defender sistema IEEE-RTS-24.

Como se observa de la anterior figura, para este caso el algoritmo no fue capaz de encontrar una ruta para unir la generación en la parte oeste de la

vista del sistema, con la demanda al este, lo que indica que ha quedado en un óptimo local en la defensa.

Tabla 8. Mayor deslastre en sistema IEEE-32 ante defensa.

SISTEMA-DISTRIBUCIÓN 32 BARRAS	PYTHON	MATLAB
LINEAS ATACADAS	1, 2, 3, 8, 20, 23	1, 4, 6, 21, 25, 27
DESLASTRE EN ATAQUE [KW]	3466.1239	3635.7
RESPUESTA DEFENSA	33	35
DESLASTRE EN DEFENSA [KW]	3321.1011	3392.2
REDUCCION DESLASTRE [KW]	145.0228	243.5114
TIEMPO DE EJECUCION [s]	0.3748	59.9106

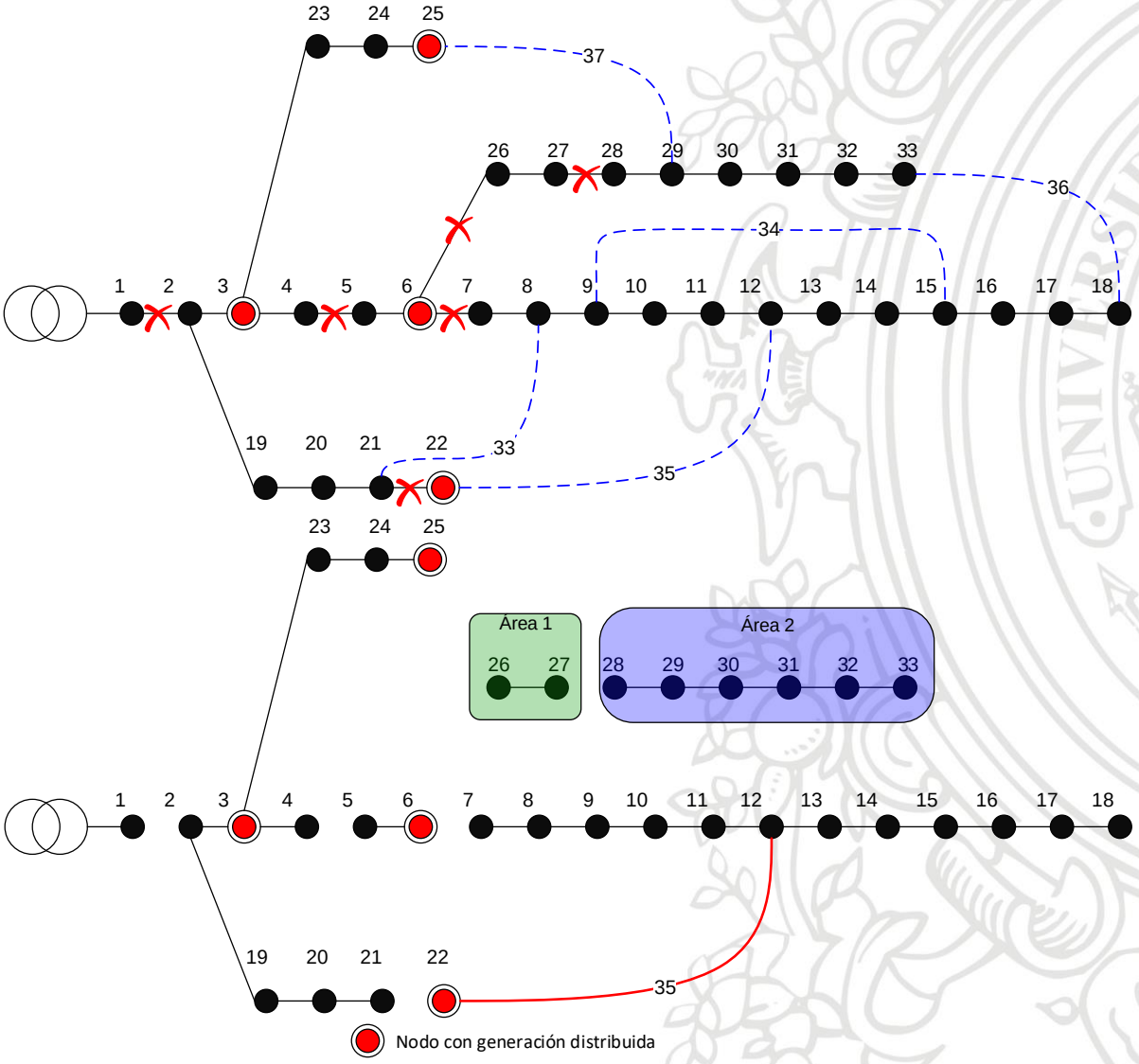

Fig. 7. Mayor deslastre después de defender sistema IEEE-32.

El resultado de la figura 7, pertenece al algoritmo implementado en Python, como se observa en este caso, si pudo encontrar una buena solución. El agente atacante ha logrado aislar dos recursos de generación distribuida y, aisló el sistema del nodo de alimentación, la mejor opción en este caso es poner en funcionamiento la línea 35 como lo muestra la figura 7.

Tabla 9. Mayor deslastre en sistema IEEE-69 ante defensa.

SISTEMA-DISTRIBUCIÓN 69 BARRAS	PYTHON	MATLAB
LINEAS ATACADAS	65, 4, 41, 48, 22, 28	2, 4, 26, 27, 37, 41
DESLASTRE EN ATAQUE [KW]	1167.1213	1197.1
RESPUESTA DEFENSA	72, 74	1, 74
DESLASTRE EN DEFENSA [KW]	1143.0287	1092.9
REDUCCION DESLASTRE [KW]	24.0926	104.2667
TIEMPO DE EJECUCION [s]	0.6416	109.1195

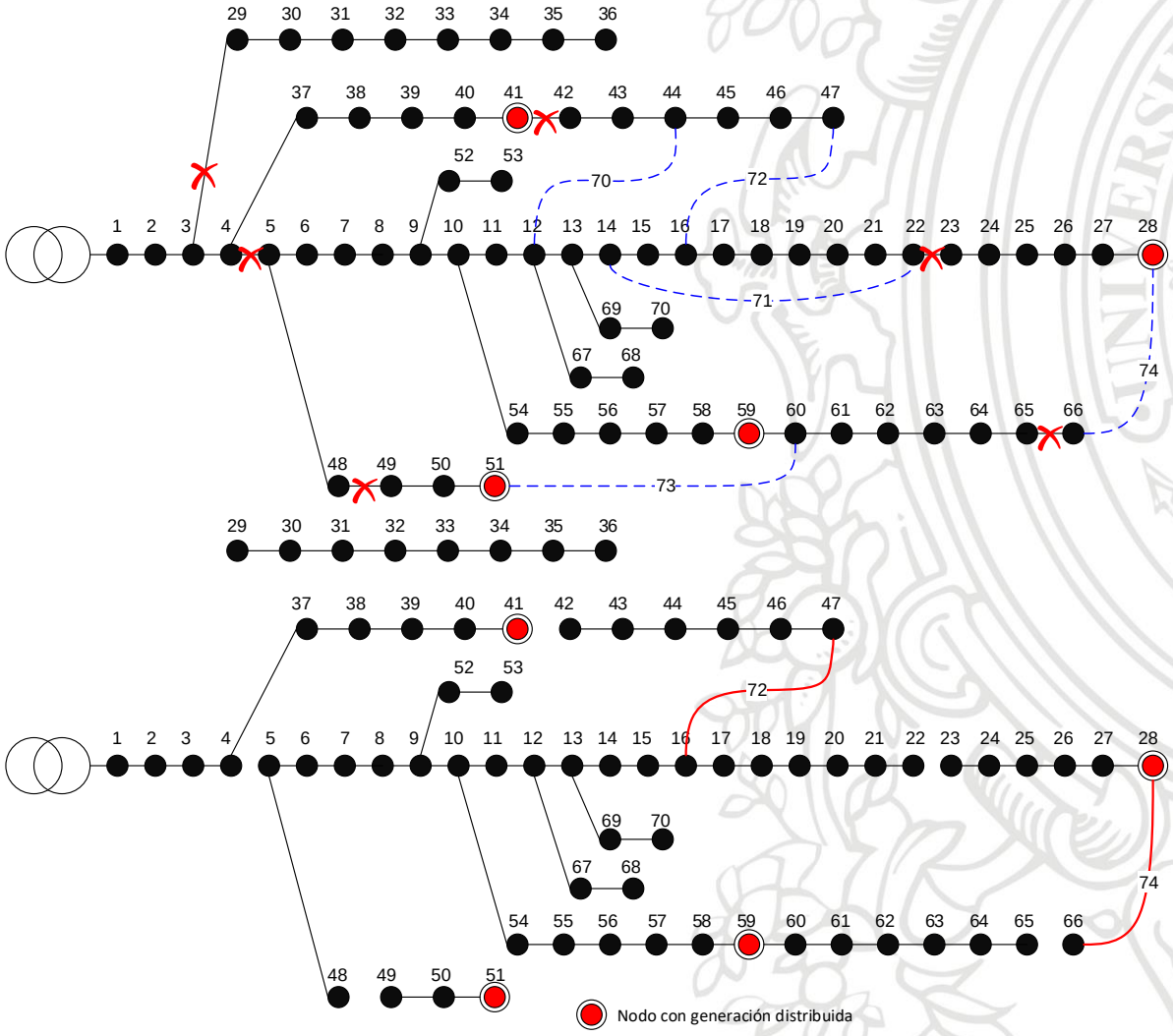

Fig. 8. Mayor deslastre después de defender sistema IEEE-69.

De la figura 8 vemos que el agente agresor da una solución similar en aspecto a la del sistema IEEE-32, este intenta aislar el sistema del nodo de alimentación en los primeros corredores de potencia (ataque en la línea que va del nodo 4 al nodo 5), luego intenta aislar la generación distribuida (ataque cercano a los nodos 22 y 65). El agente defensor intenta mitigar el daño conectando el nodo de generación distribuida (28) con la demanda en el nodo 65 y, también inútilmente opera la línea 72 con la generación distribuida en el nodo 59 pero, sin embargo, esta generación no suple la demanda total.

Conclusiones

De los resultados mostrados en la anterior sección se puede decir que la implementación de adecuadas heurísticas puede entregar soluciones satisfactorias para el análisis de vulnerabilidad, estas deben ser sintonizadas dependiendo del sistema eléctrico, para nuestro caso se usó la misma en cada sistema de prueba.

Para el análisis de vulnerabilidad, en los sistemas radiales, como es el caso en distribución, se observó que los mejores ataques contemplaban líneas cercanas al nodo de alimentación y, también se buscaba aislar los nodos con generación distribuida.

Se concluye también que el lenguaje de programación Python, en general tubo mayor rendimiento en la ejecución y, como resultado de este trabajo se obtuvo una herramienta multipropósito para el análisis de la vulnerabilidad en los sistemas eléctricos, tanto de distribución como de potencia.

Referencias Bibliográficas

[1] J. J. Cortina, "ANÁLISIS DE VULNERABILIDAD EN SISTEMAS DE POTENCIA USANDO TÉCNICAS DE OPTIMIZACIÓN METAHEURÍSTICA.", M.S. thesis, Dept. Electrical Eng., Univ. of Antioquia, Medellin, Colombia, 2016.

[2] J. Salmeron, K. Wood, and R. Baldick, "Analysis of Electric Grid Security Under Terrorist Threat," *IEEE Trans. Power Syst.*, vol. 19, no. 2, pp. 905–912, May 2004.

[3] P. Corredor and M. Ruiz, "Against All Odds," in *IEEE Power and Energy Magazine*, vol. 9, no. 2, pp. 59-66, March-April 2011.

[4] J. M. Arroyo and F. D. Galiana, "On the Solution of the Bilevel Programming Formulation of the Terrorist Threat Problem," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 789–797, May 2005.

[5] L. Agudelo, J. M. López-Lezama, and N. Muñoz, "Análisis de Vulnerabilidad de Sistemas de Potencia Mediante Programación Binivel," *Inf. tecnológica*, vol. 25, no. 3, pp. 103–114, Feb. 2014.

[6] J. M. Arroyo, "Bilevel programming applied to power system vulnerability analysis under multiple contingencies," in *IET Generation, Transmission & Distribution*, vol. 4, no. 2, pp. 178-190, February 2010

[7] J. M. Arroyo, "Bilevel programming applied to power system vulnerability analysis under multiple contingencies," *IET Gener. Transm. Distrib.*, vol. 4, no. 2, pp. 178-190, 2010.

[8] J. M. Arroyo and F. J. Fernandez, "A Genetic Algorithm Approach for the Analysis of Electric Grid Interdiction with Line Switching," in 15th International Conference on Intelligent System Applications to Power Systems, Curitiba, Brazil, 2009, pp. 1-6.

[9] H. Li, L. Zhang, and Y. Jiao, "Solution for integer linear bilevel programming problems using orthogonal genetic algorithm," *J. Syst. Eng. Electron.*, vol. 25, no. 3, pp. 443–451, Jun. 2014.

[10] K. C. Almeida and F. S. Senna, "Optimal Active-Reactive Power Dispatch Under Competition via Bilevel Programming," *IEEE Trans. Power Syst.*, vol. 26, no. 4, pp. 2345–2354, Nov. 2011.

[11] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*. 3rd ed. New York: International Series in Operations Research & Management Science – Springer, 2019.