Evaluación de Heurísticas para la Gestión Óptima de Recursos Energéticos Locales en un Sistema de Distribución

Anderson López Chaverra

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones
Bucaramanga
2016

Evaluación de Heurísticas para la Gestión Óptima de Recursos Energéticos Locales en un Sistema de Distribución

Anderson López Chaverra
Trabajo de grado para optar al título de
Ingenierio Electricista

Director
Iván David Serna Suárez
Magister en Ingeniería Eléctrica

Codirectores
Gabriel Ordóñez Plata
Doctor en Ingeniería Industrial

Gilberto Carrillo Caicedo
Doctor en Ingeniería Industrial

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones
Bucaramanga
2016

A mis padres Joaquín Emilio López López y Rosalba Chaverra Castaño, a mis hermanos Vladimir, Elkin y Judith, por ser la base de toda mi vida.

Todo este trabajo ha sido posible gracias a ustedes.

Anderson López

Agradecimientos

Agradezco a todos mis compañeros de la Universidad Industrial de Santander quienes me apoyaron tanto académicamente como en lo personal. Agradezco a la Rama Estudiantil IEEE-UIS por permitirme ser parte de esta gran familia y compartir tantos momentos de alegría. Agradezco a todos los profesores que me acompañaron en el camino del aprendizaje. Al profesor Iván Serna Suárez mi director de proyecto quién me guió y aportó su conocimiento para la realización de este proyecto. También agradezco a mi familia la cual es mi principal motivación para salir adelante. Finalmente un agradecimiento especial a Carolina Gómez Delgado, Aura Milena Rueda Díaz, Thomas Eduardo Medina Perea, Luis Alfonso Rada Alean, Jorge Barrios y Nestor Javier Díaz Díaz quienes me apoyaron incondicionalmente durante este largo camino.

Contenido

LISTA DE FIGURAS

Lista de tablas

Lista de algoritmos

Lista de anexos

RESUMEN

TÍTULO:
EVALUACIÓN DE HEURÍSTICAS PARA LA GESTIÓN ÓPTIMA DE RECURSOS ENERGÉTICOS LOCALES EN UN SISTEMA DE DISTRIBUCIÓN[1]

AUTOR:
ANDERSON LÓPEZ CHAVERRA[2]

PALABRAS CLAVE:
Heurísticas, optimización, fuentes de generación renovable, sistemas de almacenamiento, sistemas de distribución.

DESCRIPCIÓN:
Debido al alto grado de explotación de los recursos no renovables para cubrir la demanda energética se debe optar por soluciones más sostenibles que permitan un uso más eficiente de la canasta energética con la que se dispone; de esta manera es preciso el uso de fuentes de generación renovable apoyadas con sistemas de almacenamiento para aumentar la eficiencia y calidad de la energía de los actuales sistemas de distribución de energía eléctrica.
Para estos sistemas de distribución se requiere planificar en el tiempo la correcta operación de los elementos de almacenamiento pues este debe cargar, descargar o desconectarse del sistema de distribución tomando como principales criterios de operación los precios de compra/venta de la energía eléctrica, curva de demanda y perfiles de energía fotovoltaica.
Por consiguiente, este proyecto busca evaluar las heurísticas: Algoritmos genéticos, estrategias de optimización basadas en caos y búsqueda arborescente. De igual forma se evaluará dos esquemas en el que el almacenamiento estará concentrado en la cabecera del circuito y se comparará el desempeño con respecto al caso en el cual el almacenamiento se encuentra conectado en la cola del sistema de distribución y de este modo determinar que esquema es el más adecuado para gestionar los recursos energéticos locales en los sistemas de distribución.

[1] Trabajo de grado.
[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Iván David Serna Suárez. Codirector: Gabriel Ordoñez Pláta, Gilberto Carrillo Caicedo.

ABSTRACT

TITLE:

HEURISTIC EVALUATION FOR OPTIMAL MANAGEMENT OF LOCAL ENERGY RESOURCES IN A DISTRIBUTION SYSTEM. [3]

AUTHORS:

ANDERSON LÓPEZ CHAVERRA[4]

KEY WORDS:

Heuristic, optimization, sources of renewable generation, systems of storage, distribution systems.

DESCRIPTION:

Due to the high exploitation of nonrenewable resources to cover the energetic demand, it is necessary to choose more sustainable solutions that allow more efficient use of the available energy. That's why the use of renewable generation sources, supported by storage systems, is necessary to increase the efficiency and quality of energy of electric distribution systems. These electric distribution systems require planning in a short amount of time the correct operation of the storage elements because they must be loaded, unloaded or disconnected from the distribution system taking into account, as main criterion of operation, prices for buying and selling electricity, demand curve and photovoltaic power profiles.

Consequently, this project seeks to evaluate the heuristic: genetic Algorithms, strategies of optimization based on chaos for explample chaos based repetititve random search, particle swarm optimization and metaheuristic bat inspired algorithm and tree shaped search optimization. Also, two schemes in which storage will be concentrated at the head of the electric distribution system and the performance is compared with the case in which the storage is connected to the tail of the electric distribution system will be evaluated to determine which scheme is best suited to manage local energy resources in electric distribution systems.

[3] Research work.

[4] Faculty of Physical-Mechanic Engieneering. School of Electrical, Electronical and Telecommunications Engineering. Advisor: Iván David Serna Suárez. . Co-advisor: Gabriel Ordoñez Pláta, Gilberto Carrillo Caicedo.

Glosario

Elemento de almacenamiento de energía: Una batería o acumulador eléctrico es un dispositivo electroquímico que permite almacenar energía en forma química mediante el proceso de carga, y liberarla como energía eléctrica, durante la descarga, mediante reacciones químicas reversibles cuando se conecta con un circuito de consumo externo. Todas las baterías son similares en su construcción y están formadas por un número de celdas compuestas de electrodos positivos y negativos, separadores y de electrolito. El tamaño, el diseño interno y los materiales utilizados controlan la cantidad de energía disponible de cada celda. El tipo de acumulador más usado en la actualidad, dado su bajo costo, es la batería de plomo ácido. En ella, los dos electrodos están hechos de plomo y el electrolito es una solución de agua destilada y ácido sulfúrico.

Convergencia: En matemáticas, estadísticas, ciencias empíricas, ciencia de la computación, o economía, optimización matemática (o bien, optimización o programación matemática) es la selección del mejor elemento (con respecto a algún criterio) de un conjunto de elementos disponibles (**?**).

Capacidad: Es la cantidad de electricidad que puede obtenerse mediante la descarga total de una batería inicialmente cargada al máximo.

Energía solar fotovoltaica: La energía solar fotovoltaica es una fuente de energía que produce electricidad de origen renovable, obtenida directamente a partir de la radiación solar mediante un dispositivo semiconductor denominado célula fotovoltaica, o bien mediante una deposición de metales sobre un sustrato denominada célula solar de película fina (**?**).

Estado de carga: SOC - State of charge, por sus inglas en ingles se define como la capacidad

disponible de una batería y podría ser comparada con el depósito de combustible de un vehículo. Se representa como porcentaje de una referencia de carga completa, el cual debe ser actualizado periódicamente. Otras unidades pueden ser Culombios [C], Amperios-hora [Ah] o kilovatioshora [kWh].

Gasificación: Cuando una batería de plomo-ácido está próxima al 100 %, la cantidad de agua en el electrolito se ha reducido, esto impide la reacción química y como consecuencia el exceso de gases escapa del electrolito produciendo un intenso burbujeo. Por otro lado cuando el proceso de carga no es controlado, la gasificación excesiva contiene ácido sulfúrico que al escapar de los tapones de respiración daña los terminales (oxida los sostenes de plomo de las celdas) y disminuye la cantidad de ácido dentro de la batería.

Heurística: Se define como el conjunto de estrategias y técnicas para resolver problemas que conocemos y que estamos en capacidad de aplicar (**?**).

Sistema de distribución: Un sistema eléctrico de potencia incluye las etapas de generación, transmisión, distribución y utilización de la energía eléctrica, y su función primordial es la de llevar esta energía desde los centros de generación hasta los centros de consumo y por último entregarla al usuario en forma segura y con los niveles de calidad exigidos (**?**).

Sulfatación: Las baterías de plomo-ácido trae aparejado un dispositivo de sulfato de plomo en ambas placas; están constituidas por pequeños cristales, los cuales se descomponen fácilmente durante la carga. Si se tiene descargas por debajo del mínimo establecido, pobremente cargada o permanece descargada mucho tiempo el tamaño de los cristales crece y disminuye la superficie activa del electrodo que a su vez disminuye la capacidad de almacenaje de la batería

Vida útil: La vida útil de una batería en servicio corresponde al período de tiempo o al número de ciclos de carga/descarga que la batería puede soportar hasta que su capacidad sea insuficiente para cubrir las necesidades para las que fue diseñada. Se considera que una batería llega al fin de su vida útil cuando no puede entregar el 80 % de su capacidad nominal.

Introducción

El modelo actual de distribución de energía eléctrica está diseñado para garantizar la continuidad del servicio para todos los usuarios dentro de una región limitada, lo que puede ocasionar que los sistemas de distribución alejados de la red central a menudo presenten problemas en la calidad del servicio o que a pesar de tener un servicio continuo no contemplen la posibilidad de implementar fuentes de generación renovable.

De acuerdo con lo anterior, el problema guía de la investigación a adelantar, se puede encontrar diversas opciones para la solución del mismo (**??**). Una de las alternativas, es hacer un buen uso de las fuentes de generación renovable como lo es, la energía fotovoltaica. A saber, los paneles solares por su bajo costo son la opción más adecuada para implementarse en zonas con altos índices de radiación solar. Indudablemente, se hace necesario la inclusión de sistemas de almacenamiento como apoyo a la fuente de generación renovable para aprovechar la energía almacenada durante las horas de mayor radiación o en las horas donde el precio de la energía eléctrica es bajo y así disminuir la cantidad de energética que se extrae de la red centralizada. Ahora bien, la investigación se concentrará en el uso de heurísticas para determinar en qué momentos del día el sistema de almacenamiento debe cargar, descargar o desconectarse del sistema en función de los índices de radiación solar, y el precio de compra y venta de la energía eléctrica.

Este documento está organizado de la siguiente manera: en el *Capítulo 1* define el concepto de algoritmos genético, cúmulo de partículas, metaheurística del murciélago, búsqueda aleatoria repetitiva basada en caos y búsqueda arborescente; en la *Capítulo 2* se explica el método de *Newton-Raphson con sistema P.U. complejo* y *Flujo Radial TRX* para la solución de flujo de potencia; en la *Capítulo 3* se presenta la estructura de los algoritmos implementado; en la *Capítulo 4* se presenta el caso de aplicación, pruebas y análisis de resultados producto del desarrollo de todo el proyecto, seguido de las conclusiones que resumen los avances y logros

del mismo.

1

HEURÍSTICAS

En computación, una heurística es un algoritmo capaz de realizar búsquedas exhaustivas en comparación con los métodos tradicionales donde se necesitaría tiempos demasiados grandes para encontrar una buena solución o por el contrario no exista manera de hallar una solución por medio de técnicas analíticas (exactas o aproximadas). Sin embargo, uno de los principales inconvenientes con las heurísticas radica en que si encuentra una solución está no tenga las mejores propiedades, es decir, buenos tiempo y buenas soluciones; o si se ejecuta razonablemente rápido, puede que no exista prueba de que siempre será así (**?**).

En este capítulo se explicarán las diversas heurísticas que fueron implementadas y se comparará su desempeño basado en funciones de prueba generales, para así llevar a cabo la optimización del perfil de almacenamiento con las heurísticas más destacadas.

Este capítulo está organizado de la siguiente manera: en la *Sección 1.1* se define el algoritmo genético, explicando en detalle cada una de sus procesos como lo es la selección, cruce y mutación de los individuos; posteriormente en la *Sección 1.2* se define el concepto de las estrategias de optimización basadas en caos, particularizando en las heurísticas que imitan el comportamiento social de ciertas especies como es el caso de cúmulo de partículas y metaheurística del murciélago; en la *Sección 1.3* se explica el algoritmo búsqueda arborescente; y finalmente en la *Sección 1.4* se evalúa cada una de las heurísticas frente a funciones de prueba esándar.

1.1 Algoritmo genético

El término de algoritmo genético (AG) aparece en el (**?**) quien utilizó este método para buscar un conjunto de parámetros en funciones de los juegos evolutivos, comparándolos con los algoritmos de correlación y procedimientos de aprendizaje modelados. Pero, se considera como creador de este algoritmo al investigador Holland quien desarrolló los conceptos relacionados de este algoritmo con sus colegas y estudiantes de la universidad de Michigan y que fue publicado en (**?**). El libro se titula: "Adaptación en sistemas naturales y artificiales". El propósito original de Holland no era diseñar un sistema que resolviera problemas específicos, sino pretendía crear un algoritmo que permitiría resolver problemas de búsqueda. Por ello, los objetivos planteados por Holland y sus colegas son dos: en primera instancia, es abstraer y explicar rigurosamente el proceso adaptativo de los sistemas naturales, y en segunda medida, diseñar sistemas artificiales que tomarán los mecanismos más importantes de los sistemas naturales.

El algoritmo genético se asemeja al modelo biológico de los cromosomas y genes, el cual consiste en un conjunto de cromosomas que está conformado de genes y mediante los procesos de selección, cruce y mutación dichos genes crean nuevos cromosomas que heredan características de sus antecesores. Así mismo, cada cromosoma se ve como un individuo que pertenece a una población donde su adaptación al entorno depende de un proceso iterativo, que concluye cuando el mejor individuo no presenta un cambio significativo en un número determinado de generaciones.

En las siguientes subsecciones se explicará cada una de las etapas tomando como base el *Algoritmo* 1. En el *Anexo A.1* se encuentra la implementación del algoritmo el cual fue desarrollado en el lenguaje de programación Python.

Algoritmo 1. Algoritmo genético

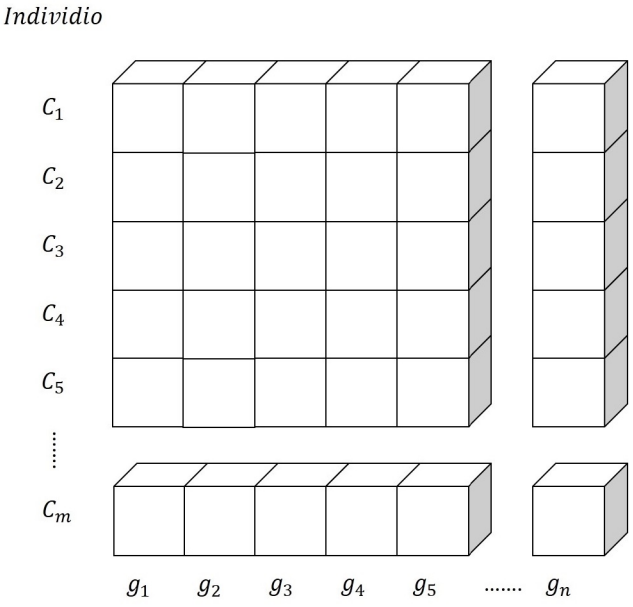
```
1  Generar, población inicial I
2  Evaluar, f(i) ∀i ∈ I, con f : I → ℝ
3      while convergencia de la población
4          foreach (i, j) ∈ I
5              if factibilidad del individuo
6                  Seleccionar, pareja de individuos (i, j)
7              else
8                  Reparar,individuono factible
9              d' ← Cruzar(i, j)
10             d ← Mutar(d')
11             Evaluar, f(d)
12             I ← Decendencia(d)
```

13 Insertar, $I \leftarrow I \cup \{d\}$

1.1.1 Población inicial Los algoritmos genéticos utilizan una analogía para trabajar con poblaciones de individuos. Donde cada uno de estos representan soluciones para un problema dado.

Computacionalmente los individuos se representan con un código genético C_m llamado cromosomas que a su vez se divide en genes g_n (ver *Figura* 1.1), típicamente una secuencia de bits en la que se encuentra codificada su información.

Figura 1.1.: Representación de la población en el algoritmo genético.

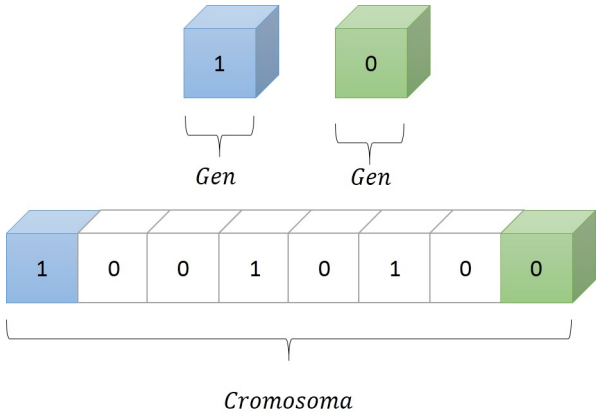


Uno de los principales problemas de los algoritmos genéticos está relacionado con la generación de la población inicial (ver *línea 1*). Así, como en la naturaleza las poblaciones muy grandes o muy pequeñas tenderán a desaparecer si no logran adaptarse al entorno donde viven. De igual forma un algoritmo genético con una población muy pequeña puede converger rápidamente hacia óptimos locales, por otro lado poblaciones muy grandes a pesar que puedan cubrir todo el espacio de búsqueda y posteriormente logre encontrar el óptimo global, puede acarrear problemas relacionados con el excesivo costo computacional. De acuerdo con estudios presentados por (**?**) el tamaño óptimo de la población con codificación binaria crece exponencialmente de acuerdo a la longitud del cromosoma. Estos resultados pueden indicar que los algoritmos genéticos tendrían una aplicabilidad limitada y poco competitiva frente a otras

técnicas de optimización combinatoria.

Codificación: Los cromosomas de alguna manera deberán contener información acerca de la solución que representa. La codificación se puede realizar de varias formas. La más utilizada es mediante una cadena de números binarios $C_i = [g_1, g_2, g_3, ..., g_n]$ con $g_n \in [0, 1]$ (ver *Figura 1.2*). Pero también se puede realizar la codificación mediante números enteros o incluso cadenas de palabras (**?**).

Figura 1.2.: Codificación binaria del cromosoma en el algoritmo genético.



Ahora bien, si se desea trabajar un problema continuo es necesario convertir el vector cromosoma a un valor acorde y dentro del rango establecido para la función objetivo. Para esto se emplea la *ecuación* 1.1 para pasar de un vector binario a un valor hexadecimal.

$$x_{hexa,i} = g_1 2^0 + g_2 2^1 + g_3 2^2 + ... + g_n 2^{n-1} \tag{1.1}$$

$x_{hexa,i}$, Representa el valor hexadecimal del *i-ésimo* individuo de la población.

Seguidamente, se toma la *ecuación* 1.2 que realiza la decodificación de $x_{hexa,i}$ a un valor comprendido en un rango previamente definido.

$$x_i = x_{hexa,i} \frac{Rango_{max} - Rango_{min}}{2^{bit} - 1} + Rango_{min} \tag{1.2}$$

x_i, Representa el valor decodificado del *i-ésimo* individuo de la población.

bit, Representa el número de genes en el cromosoma.

1.1.2 Función de evaluación

Uno de los aspectos más importantes en el algoritmo genético es la función de evaluación (ver *líneas 2 y* 11), ya que esta debe reflejar el valor de cada individuo de manera real, pues en la mayoría de los casos existe gran cantidad de restricciones que

hacen que buena parte de los puntos en el espacio de búsqueda sean descartados. De acuerdo a lo anterior, existen diversas soluciones para garantizar que lo individuos evaluados cumplan con todas las restricciones establecidas. Una de estas es donde se descartan los individuos que no cumplan con las condiciones dadas, por lo cual se les asignará un valor de cero en la función de evaluación, a este método se le denomina absolutista. Otro método empleado, consiste en reconstruir el individuo que no cumpla con las restricciones, para esto se suele usar un nuevo operador que se denominara reparador.

Por otro lado, la función de evaluación se utiliza para establecer que tan bueno es un individuo respecto a otro dentro de la población. Usualmente la función de evaluación pondera cada individuo en un rango entre 0 y 1, donde 0 representa un individuo poco adaptado y 1 representa un individuo muy adaptado.

1.1.3 Selección La población del algoritmo genético se somete a un proceso de selección (ver *línea* 6) que debe tender a favorecer la cantidad de individuos más adaptados. Este proceso se puede realizar de distintas maneras según como se plantea en (**?**), entre estas se tiene:

Selección proporcional o por ruleta: La probabilidad de seleccionar a un individuo x_i con este método es directamente proporcional a su adaptación relativa.

$$p_i = \frac{f_i}{\bar{f}} \tag{1.3}$$

$\bar{f}$, es la adaptación promedio de la población.

Posteriormente se necesita generar un número aleatorio que concuerde con la distribución obtenida en p_i, hecho esto se puede realizar el siguiente procedimiento:

1. Se define una puntuación acumulada de la siguiente forma:

$$\begin{aligned} q_o &= 0 \\ q_i &= p_1 + p_2 + p_3 + ... + p_n, \forall i \in \{1, 2, 3, ..., n\} \end{aligned} \tag{1.4}$$

2. Se genera un número aleatorio $a \in [0 : 1]$.

3. Se selecciona al individuo i que cumpla:

$$q_{i-1} < a < q_i \tag{1.5}$$

Este proceso se repite para cada individuo que se desea seleccionar.

Selección por torneo: En la selección por torneo se elige aleatoriamente una pequeña muestra de la población y a partir de esta se selecciona al mejor. Esta selección puede hacerse de forma determinista o probabilista.

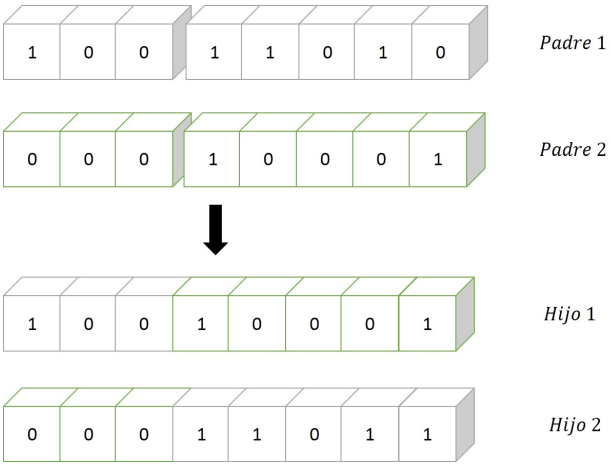
Determinista: Esta selección consiste en elegir aleatoriamente un número p de individuos de la población, regularmente el valor de p no es mayor a 2. Después se elige al que tenga la mayor adaptación entre los p individuos.

Probabilista: Se realiza al igual que el caso anterior, sin embargo la selección no favorece siempre al mejor, el proceso se realiza con cierta probabilidad. Si un número, que se genera aleatoriamente entre 0 y 1 es mayor a cierto umbral establecido para cada individuo, entonces se elige al mejor.

1.1.4 Cruce El objeto principal del proceso de cruce (ver *línea* 7) consiste en tomar dos individuos previamente seleccionados *padres* y a partir de estos se logre obtener *hijos* los cuales heredan las mejores características de los padres. Existen gran variedad de operadores de cruce, sin embargo los más empleados se detallan a continuación.

Cruce de un punto: Este método consiste en dividir el cromosoma de los padres en un punto generado aleatoriamente, obteniendo así dos segmentos diferentes en cada uno de ellos, posteriormente la cola y la cabeza se intercambia para dar nacimiento a dos hijos los cuales heredan información de los padres. En la bibliografía se refiere a este tipo de cruce con el nombre de SPX (Single Point Crossover).

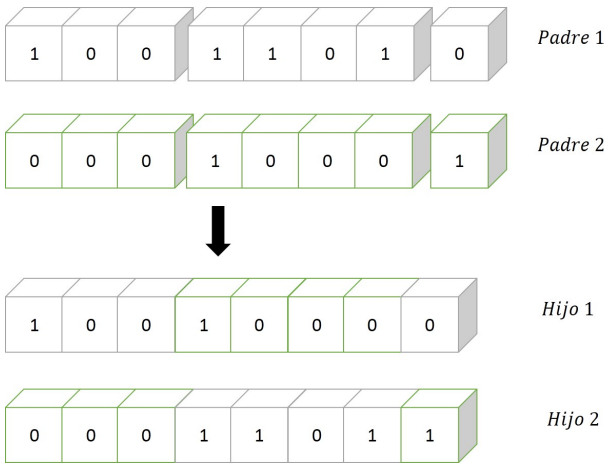
Figura 1.3.: Algoritmo Genético: cruce de un punto



Cruce de dos puntos: Este método consiste en dividir el cromosoma de los padres en dos puntos generados aleatoriamente, obteniendo así tres segmentos diferentes en cada uno de ellos,

posteriormente se intercambia el tronco para dar nacimiento a dos hijos los cuales heredan información de los padres, una ventaja de este método es que logra una mayor exploración del espacio de búsqueda respecto al cruce en un punto. En la bibliografía se refiere a este tipo de cruce con el nombre de DPX (Double Point Crossover).

Figura 1.4.: Algoritmo Genético: cruce de dos puntos

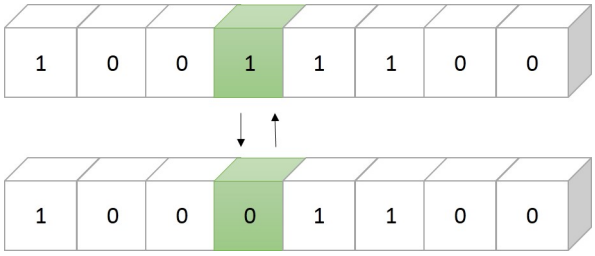


Siguiendo el mismo razonamiento se puede crear hijos realizando múltiples cortes en el cromosoma, sin embargo existen múltiples estudios que rechazan este método (**?**). Los estudios indicaron que el algoritmo genético mostraba una mejora notable cuando se realiza un corte en dos puntos en comparación con el corte de un punto, por otro lado más de dos cortes solo ocasionaba que el cromosoma pierda todas sus características y como consecuencia el algoritmo no logra converger.

1.1.5 Mutación La función principal de la mutación (ver *línea* 8) radica en proporcionar un pequeño elemento de aleatoriedad en un individuo dentro de la población. Generalmente la probabilidad de mutación es menor al 1 % esto se debe sobre todo a que los individuos muy mutados tienden a perder información heredada de los padres, provocando que el algoritmo nunca converja. Por otro lado, si una población tiene una probabilidad de mutación del 0 % esta tenderá a converger prematuramente hacia un óptimo local.

Computacionalmente la mutación consiste en variar aleatoriamente un gen de dentro del cromosoma (ver *Figura 1.5*), es decir si el gen que se desea mutar tiene un valor de 1 este será remplazado por un 0, ahora si el gen tiene un valor de 0 este será reemplazado por un 1.

Figura 1.5.: Algoritmo Genético: mutación



1.1.6 Descendencia En función de la cantidad de individuos reemplazados o insertados (ver *línea* 10) en la nueva población se pueden considerar 2 tipos de descendencia de acuerdo a como se describe en (**?**):

1. AG generacionales, la nueva población reemplaza por completo a la generación anterior.
2. AG con estado estacionario, se selecciona un número p de los mejores individuos de la población actual, y se incluyen en la nueva población, reemplazando a los p individuos según los siguientes criterios: reemplazo aleatorio, reemplazo de los individuos peor adaptados y reemplazo de individuos de adaptación similar.

1.2 Estrategias de optimización basadas en caos

Cuando se habla de caos, nos referimos a un comportamiento complejo, inestable y limitado. No obstante, una serie de números que se comporten de manera caótica pueden ser generados a través de lo que se conoce como mapa caótico. Uno de estos mapas es la parábola logística, este mapa fue aplicado en (**?**) para el algoritmo búsqueda aleatoria repetitiva donde los números creados aleatoriamente fueron reemplazados por números generados por medio del mapa caótico de la parábola logística:

$$x^{k+1} = rx^k \left(1 - x^k\right) \tag{1.6}$$

x^{k+1}, es un número entre cero y uno que representa a la fracción de individuos en un territorio, respecto de un k^0 supuesto máximo posible, en un instante k.

r, es un número positivo que representa la relación o tasa combinada entre la reproducción y la mortandad.

La estrategias de optimización basadas en caos tienen tres líneas de investigación en el campo de la optimización global de acuerdo con (**?**):

1. La solución de problemas combinatorios.

2. La sustitución de los números generados aleatoriamente en el algoritmo por números generados a partir de un mapa caótico.

3. La creación de algoritmos híbridos de optimización.

Para este proyecto se sigue el item's 2, tomando como base el mapa caótico de la parábola logística, ya que utilizando los números generados por dicho mapa mejoró los tiempos de solución en comparación con los números generados aleatoriamente(**???**).

1.2.1 Búsqueda aleatoria repetitiva basada en caos

La búsqueda aleatoria repetitiva basada en caos (RSS) (**?**) «es un método que se caracteriza por un componente aleatorio, que consiste en iniciar de un punto de arranque el cual desglosa un camino donde la dirección y el tamaño del paso, dependa de las decisiones que se encuentran en la iteración anterior. Asimismo, este algoritmo requiere de dos parámetros que son especificados por el usuario, pues afectan en gran medida el desempeño computacional del mismo». Por ello, este algoritmo se debe ajustar para cada problema que deseé desarrollar y así se logre obtener mejores resultados.

En las siguientes subsecciones se explica cada una de las etapas tomando como base el *Algoritmo* 2. En el *anexo A.2* se encuentra la implementación del algoritmo el cual fue desarrollado en el lenguaje de programación Python.

Algoritmo 2. Búsqueda aleatoria repetitiva basada en caos

```
1  Generar, población inicial X
2  Hacer, Z ← X
3  Inicializar, β, γ, λ ∈ [0, 1]
4  Evaluar, f(x) ∀x ∈ X, con f : X → ℝ
5      while convergencia de la población
6          foreach (x) ∈ X
7              β ← Caos(β)
8              γ ← Caos(γ)
9              u ← Rand.
10             △x ← Calcular(z, u, β, λ).
11             Hacer, y ← x + △x
12             if f(y) > f(x)
13                 Hacer, x ← y
14                 z ← Actualizar(z, △x, γ)
15                 Hacer, λ ← 1.2λ
16             else
17                 Hacer, λ ← 0.9λ
18                 if λ < 0.1
19                     Hacer, λ ← 0.1
```

Población inicial: La población inicial en el algoritmo búsqueda aleatoria repetitiva basada en caos (ver *línea 1*) consiste en crear un matriz cuyas filas representan una posible solución al problema, y las columnas representan las dimensiones o variables del problema (ver *Figura 1.6*).

Figura 1.6.: Búsqueda aleatoria repetitiva basada en caos: Población inicial

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \cdots & x_{m,n} \end{bmatrix}$$

Ecuaciones de actualización: Las *líneas 7, 8, 9, 10, y 11*, describe la dinámica de exploración de los puntos en busca de las mejores posiciones dentro del espacio de soluciones. Las *lineas 7 y 8* permiten por medio de la función caos generar dos números caóticos de acuerdo al mapa caótico de la *Ecuación* 1.6. La *línea 9* consiste en generar un vector unitario aleatorio que permite al algoritmo un mayor grado de ergodicidad. La *línea 10* evalúa la siguiente función:

$$\triangle x_i^{k+1} = \lambda_i^k \left[\beta_i^k \frac{z_i^k}{\|z_i^k\|} + (1 - \beta_i^k) u_i^k \right] \tag{1.7}$$

z_i, es una lista que guarda la dirección de las mejores posiciones encontrados hasta el momento.

u_i, es un lista aleatorio de magnitud unitaria.

β, es un factor de ponderación$\in [0, 1]$ que combina la dirección de z_i y u_i.

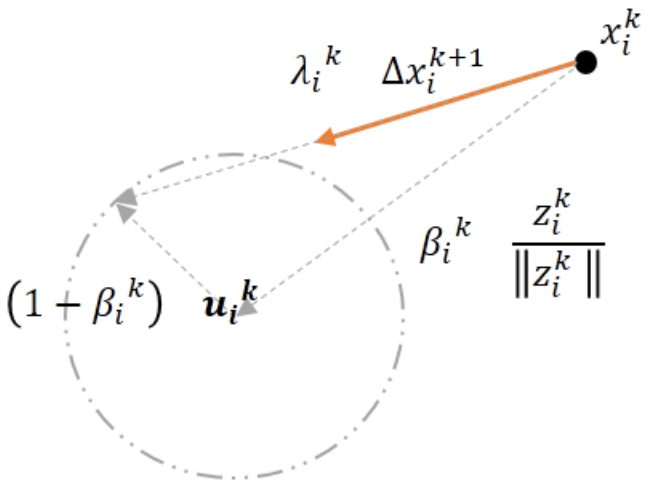
λ, es un escalar que controla la magnitud de la siguiente iteración.

En la *línea 11,* actualiza la posición dentro del espacio de soluciones de la siguiente manera:

$$x_i^{k+1} = x_i^k + \triangle x_i^{k+1} \tag{1.8}$$

Tal y como se muestra en la *Figura* 1.7, si se toma un valor de β_i muy cercano o igual a la unidad el resultado será un vector $\triangle x_i$ con la dirección de búsqueda de la mejor posición de la iteración anterior, ya que el producto de $(1 - \beta_i)u_i = 0$, caso contrario si β_i toma un valor muy cercano o igual a cero, caso en el cual la dirección de $\triangle x_i$ sera totalmente aleatoria. De este modo el parámetro β es el encargado de controlar la dirección y λ es un escalar asociado a cada punto el cual es elegido en un inicio por el usuario para establecer la magnitud del vector de dirección.

Figura 1.7.: Búsqueda aleatoria repetitiva basada en caos: determinación de la dirección de búsqueda



Criterio de aceptación: Tomando la *línea 12* se realiza una comparación de la posición actual respecto a la nueva posición calculada, de acuerdo a esto si la nueva posición es de mayor calidad que la posición actual se dice que cumple con el criterio de aceptación del algoritmo. Seguidamente, tras aprobar el criterio de aceptación en la *línea 12* la posición se actualiza al valor de la nueva posición y de igual forma en las *líneas 14 y 15* los valores de λ y z se actualizarán de acuerdo a las ecuaciones 1.9 y 1.10 respectivamente.

$$\lambda_i^{k+1} = a\lambda_i^k \tag{1.9}$$

$$z_i^{k+1} = \gamma_i^k z_i^k + (1-\gamma_i^k)\triangle x_i^k \tag{1.10}$$

γ, es un factor de ponderación $\in [0,1]$ definido por el usuario.

a, es un número constante mayor que cero.

En el caso contrario donde el criterio de aceptación no se satisfaga, es decir, cuando no se encuentra una posición de mejor calidad que el obtenido en la iteracion anterior, el valor de λ_i es reducido de acuerdo a la *Ecuación 1.11* (ver *líneas 18 y 20*).

$$\lambda_i^{k+1} = \begin{cases} b\lambda_i^k & \Rightarrow \lambda_i^k \geq a \\ a & \Rightarrow \lambda_i^k < a \end{cases} \tag{1.11}$$

b, es un número constante entre cero y uno.

a, es un número constante mayor que cero establecido por el usuario.

La *Ecuación* 1.11 asegura que el valor de λ_i sea distinto de cero, garantizando así que la magnitud de $\triangle x_i > 0$.

1.2.2 Cúmulo de partículas

La metaheurística PSO *(Particule Swarm Optimization)*, es una técnica inspirada en el comportamiento social de ciertos individuos, ya sea en el vuelo de las parvadas o bien el movimiento de los bancos de peces. El cúmulo de partículas es un sistema multiagente, es decir cada partícula se desplaza por todo el espacio establecido en búsqueda de las mejores posiciones. Esta metaheurística fue desarrollada por el Psicólogo-Sociólogo Jammes Kennedy y por el Ingeniero Electrónico Rusell Eberhart en 1995 . PSO se puede resumir de la siguiente manera : «*Los individuos que conviven en una sociedad tienen una opinión que es parte de un conjunto de creencias (el espacio de búsqueda) compartido por todos los posibles individuos*». (**?**, 18)

Cada individuo puede modificar su propia opinión basándose en tres factores (**?**) :

1. Su conocimiento sobre el entorno (su valor de fitness).
2. Su conocimiento histórico o experiencias anteriores (su memoria).
3. El conocimiento histórico o experiencias anteriores de los individuos situados en su vecindario.

En las siguientes subsecciones se explica cada una de las etapas tomando como base el *Algoritmo* 3. En el *Anexo A.12* se encuentra la implementación del algoritmo el cual fue desarrollado en el lenguaje de programación Python.

Algoritmo 3. CÚMULO DE PARTÍCULAS

1 Generar, posición y velocidad de cada individuo X, V
2 Inicializar, $\varphi_1^{\min}, \varphi_1^{\max}, \varphi_2^{\min}, \varphi_2^{\max}, w^{\min}, w^{\min} \in [0,1]$
3 **while** converjencia de la poblacion
4 **foreach** $(x) \in X$
5 $p_{Best} \leftarrow \text{Cognitivo}\,(f(x))$
6 $g_i \leftarrow \text{Social}\,(f(x)), \forall x \in X$
7 $w \leftarrow \text{Inercia}\left(w^{\min}, w^{\max}\right)$
8 $\varphi_1 \leftarrow \text{Ratio}\left(\varphi_1^{\min}, \varphi_1^{\max}\right)$
9 $\varphi_2 \leftarrow \text{Ratio}\left(\varphi_2^{\min}, \varphi_2^{\max}\right)$
10 $v \leftarrow \text{Calcular}\,(v, \varphi_1, \varphi_2, w, p_{Best}, g_i, rand_1, rand_2)$
11 Hacer, $x \leftarrow x + v$
12 Evaluar, $f\,(x)$

Población inicial: La población inicial en el algoritmo cúmulo de partículas (ver *línea 1*) consiste en crear dos matrices donde: la primera contiene la información de las posiciones de cada partícula y la segunda contiene las velocidades respectivamente. (ver *Figura* 1.8 y 1.9).

Figura 1.8.: Cúmulo de partículas: Posiciones de las partículas

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \cdots & x_{m,n} \end{bmatrix}$$

Figura 1.9.: Cúmulo de partículas: velocidad de las partículas

$$V = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & v_{2,3} & \cdots & v_{2,n} \\ v_{3,1} & v_{3,2} & v_{3,3} & \cdots & v_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m,1} & v_{m,2} & v_{m,3} & \cdots & v_{m,n} \end{bmatrix}$$

Ecuaciones de actualización: Las *línea 7, 8, 9, 10, y 11*, describe la dinámica de exploración de los partículas en busca de las mejores posiciones dentro del espacio de soluciones. La *linea 5* busca la mejor posición de la partícula actual (p_{Best}) respecto a sus posiciones en el pasado y en la *línea 6* se busca la mejor posición dentro del cúmulo (g_i). La *línea 7* consiste en generar el factor de inercia (ver *ecuación 1.12*). El factor de inercia, es una variable que establece un balance entre *Diversificación* e *Intensificación* en el cúmulo, por ejemplo valores

altos de w provoca una búsqueda exhaustiva (mas diversificada), por el contrario si se toman valores bajos de w provoca una búsqueda localizada (mas intensificada).

$$w^k = \frac{w_{min} - w_{max}}{K_{max}} k + w_{max} \tag{1.12}$$

w_{max}, es el peso inicial del factor de inercia.

w_{min}, es el peso final del factor de inercia.

K_{max}, es el número máximo de iteraciones.

k, es la iteración actual.

Al igual que el factor de inercia w, en las *líneas* 8 y 9 se establecen las ecuaciones de actualización para la razón de aprendizaje (ver *Ecuaciones 1.13 y 1.14*), de tal modo que en las primeras iteraciones el cúmulo tenga un comportamiento diversificado y con el paso de las iteraciones el comportamiento se intensifique.

$$\varphi_1^k = \frac{\varphi_1^{min} - \varphi_1^{max}}{K} k + \varphi_{1,max} \tag{1.13}$$

$$\varphi_2^k = \frac{\varphi_2^{max} - \varphi_2^{min}}{K} k + \varphi_{2,min} \tag{1.14}$$

φ_1^{max}, es el peso inicial cognitivo $\in [0, 1)$.

φ_2^{min}, es el peso inicial social $\in (0, 1]$, además se debe cumplir que $\varphi_2^{max} > \varphi_2^{min}$.

φ_1^{min}, es el peso final cognitivo $\in [0, 1)$, además se debe cumplir que $\varphi_1^{max} > \varphi_1^{min}$.

φ_2^{max}, es el peso final social $\in (0, 1]$.

K, es el número máximo de iteraciones.

k, es a iteración actual.

Adicionalmente, se puede presentar diferentes tipos de PSO, según la importancia de la razón de aprendizaje (**?**):

1. Modelo completo : $\varphi_1, \varphi_2 > 0$. Tanto el comportamiento cognitivo como el social intervienen en el movimiento.

2. Modelo solo cognitivo : $\varphi_1 > 0$ y $\varphi_2 = 0$. Únicamente el componente cognitivo interviene en el movimiento.

3. Modelo social : $\varphi_1 = 0$ y $\varphi_2 > 0$. Únicamente el componente social interviene en el movimiento.

4. Modelo sólo social : $\varphi_1 = 0$ y $\varphi_2 > 0$ y $g_i \neq x_i$. La posición de la partícula en sí no puede ser la mejor de su entorno.

Una vez que se establece la razón de aprendizaje, en la *línea 10* se calcula la velocidad de cada partícula de acuerdo a la siguiente ecuación:

$$v_i^{k+1} = w^k v_i^k + \varphi_1^k rand_1(pBest_i - x_i^k) + \varphi_2^k rand_2(g_i - x_i^k) \tag{1.15}$$

w, es el factor de inercia.

v_i^k, es la velocidad de la partícula i en la iteración k.

φ_1, es la razón de aprendizaje que controlan el componente cognitivo.

φ_2, es la razón de aprendizaje que controlan el componente social.

$rand_1$, es un números aleatorios entre0 y 1.

$rand_2$, es un números aleatorios entre0 y 1.

$pBest_i$, es la mejor posición encontrada por la partícula hasta el momento.

g_i, es la mejor posición del cúmulo encontrada hasta el momento.

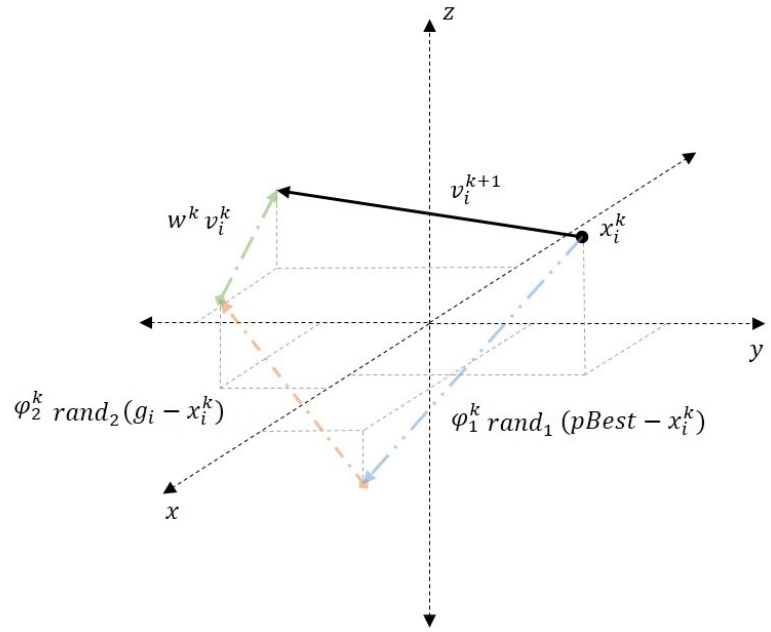
Posteriormente, en la *línea 11* se calcula la nueva posición de la partícula de acuerdo a la siguiente ecuación:

$$x_{i+1}^k = x_i^k + v_i^{k+1} \tag{1.16}$$

x_i^k, es la posición de la partícula i en la iteración k.

Finalmente, en la *Figura* 1.10 se observa la dinámica del movimiento de cada partícula donde: la linea azul corresponde el movimiento particular, la línea roja al movimiento global , la línea verde al factor de inercia y la línea negra corresponde la suma geométrica de cada uno de los movimientos de la cual corresponde a la nueva posición de la partícula.

Figura 1.10.: Determinación de la dirección de búsqueda para el algoritmo cúmulo de partículas



Modificación: Las modificación propuestas se realizó con base en la *Algoritmo* 3, donde las variables aleatorios $rand_1$ y $rand_2$ de las *línea* 10 son reemplazados por β y γ los cuales son valores generados a partir del mapa caótico de la *ecuación* 1.6. La modificaciones propuestas se presentan el *Algoritmo 4*.

Algoritmo 4. Cúmulo de partículas basado en caos

1 Generar, posición y velocidad de cada individuo X, V
2 Inicializar, $\varphi_1^{\min}, \varphi_1^{\max}, \varphi_2^{\min}, \varphi_2^{\max}, w^{\min}, w^{\min}, \beta, \gamma \in [0, 1]$
3 **while** convergencia de la población
4 **foreach** $(x) \in X$
5 $\beta \leftarrow \text{Caos}\,(\beta)$
6 $\gamma \leftarrow \text{Caos}\,(\gamma)$
7 $p_{Best} \leftarrow \text{Cognitivo}\,(f(x))$
8 $g_i \leftarrow \text{Social}\,(f(x)), \forall x \in X$
9 $w \leftarrow \text{Inercia}\left(w^{\min}, w^{\max}\right)$
10 $\varphi_1 \leftarrow \text{Ratio}\left(\varphi_1^{\min}, \varphi_1^{\max}\right)$
11 $\varphi_2 \leftarrow \text{Ratio}\left(\varphi_2^{\min}, \varphi_2^{\max}\right)$
12 $v \leftarrow \text{Calcular}\,(v, \varphi_1, \varphi_2, w, p_{Best}, g_i, \beta, \gamma)$
13 Hacer, $x \leftarrow x + v$
14 Evaluar, $f\,(x)$

1.2.3 Metaheurística del murciélago

La metaheurística del murciélago, es una técnica inspirada en el comportamiento social de los murciélagos (**?**), en el cual se definen los siguientes criterios para el algoritmo.

1. Todos los murciélagos utilizan la ecolocalización para sensar la distancia y también establecer la diferencia entre los alimentos y/o presas y barreras de fondo.
2. Los murciélagos vuelan al azar con velocidad v_i, en la posición x_i a una frecuencia f. A_i y r_i corresponde al sonido y pulso emitido por el murciélago respectivamente.
3. El volumen A_i, empieza con un valor alto A_{max} y a medida que se acerca a la presa este volumen decae hasta llegar a un valor contante A_{min}.
4. El pulso r_i, empieza en un valor mínimo r_{min} y a medida que se acerca a la presa este pulso aumenta hasta llegar a un valor máximo r_{max}.

En las siguientes subsecciones se explica cada una de las etapas tomando como base el *Algoritmo* 5. En el *Anexo A.13* se encuentra la implementación del algoritmo el cual fue desarrollado en el lenguaje de programación Python.

Algoritmo 5. METAHEURÍSTICA DEL MURCIÉLAGO

Generar, posición y velocidad inicial X y V
Inicializar, $r, f^{\min}, f^{\max}, \epsilon, A, \beta, \alpha, \varphi \in [0, 1]$
    while convergencia de la población
        foreach $(x) \in X$
            $f \leftarrow$ Calcular $\left(f^{\min}, f^{\max}, \beta\right)$
            $v_{aux} \leftarrow$ Calcular (x, x_{optimo}, f, v)
            if $rand > r$
                $x_{local} \leftarrow$ Local $\left(x_{optimo}, \epsilon, \overline{A}\right)$
            hacer, $x_{aux} \leftarrow x + v_{aux} + x_{local}$
            if $rand < A$ and $f(x) < f(x_{optimo})$
                Hacer, $x \leftarrow x_{aux}$
                Hacer, $v \leftarrow v_{aux}$
                $A \leftarrow$ Actualizar (A, α)
                $r \leftarrow$ Actualizar (r, φ)

Población inicial: La población inicial en el algoritmo (ver *línea 1*) consiste en crear dos matrices donde: la primera contiene la información de las posiciones de cada partícula y la segunda contiene las velocidades respectivamente. (ver *Figura* 1.11 y 1.12).

Figura 1.11.: Metaheurística del murciélago, Posiciones de los murciélagos

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \cdots & x_{m,n} \end{bmatrix}$$

Figura 1.12.: Metaheurística del murciélago, velocidad de los murciélagos

$$V = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & v_{2,3} & \cdots & v_{2,n} \\ v_{3,1} & v_{3,2} & v_{3,3} & \cdots & v_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m,1} & v_{m,2} & v_{m,3} & \cdots & v_{m,n} \end{bmatrix}$$

Ecuaciones de actualización: Para el cálculo de la velocidad de los murciélagos, es necesario definir la frecuencia (f) *(ver línea 5* y *Ecuación 1.17),* cuya objetivo consiste en evitar que el murciélago alcance la misma posición del mejor murciélago dentro de la colonia. Conociendo este valor en la *Ecuación 1.18* se describe la actualización para las velocidades de los murciélagos (ver *línea* 6).

$$f_i^{k+1} = f_{min} + (f_{max} - f_{min})\,\beta^k \tag{1.17}$$

$$v_i^{k+1} = v_i^k + \left(x_{óptimo} - x_i^k\right) f_i^{k+1} \tag{1.18}$$

β^k, es una variable aleatoria con $\beta \in [0:1]$.

f_i^{k+1}, es la frecuencia asociada al murciélago i en la iteración $k+1$.

v_i^{k+1}, es la velocidad asociada al murciélago i en la iteración $k+1$.

Un aspecto muy importante en este algoritmo es la capacidad de realizar una búsqueda local, en cuanto cumpla la condición $rand > r$ (ver *línea 7)*. Posteriormente si esta condición se cumple, la búsqueda local se calculará de acuerdo a la *Ecuación* 1.19 (ver *línea* 8).

$$x_{local} = x_{óptimo} + \epsilon \bar{A} \tag{1.19}$$

$\overline{A}$, es la media del vector A.

$x_{óptimo}$, es la posición del murciélago mejor adaptado.

ϵ, es un escalar definida por el usuario con $\epsilon \in [0:1]$.

Seguidamente la *Ecuación 1.20* calcula la nueva posición del murciélago en función de la velocidad y búsqueda local halladas con anterioridad (ver *línea* 9).

$$x_i^{k+1} = x_i^k + v_i^{k+1} + x_{local} \tag{1.20}$$

x_i^{k+1} : Posición asociada al murciélago i en la iteración $k+1$.

Criterio de aceptación: Caso contrario al cúmulo de partículas donde no existe criterio de aceptación, pues cada partícula puede empeorar su posición tras el paso de las iteraciones, este algoritmo presenta dos criterios de aceptación que consiste en:

1. Que un número generado aleatoriamente en el rango de [0, 1] sea menor que el volumen (A) emitido por el murciélago (ver *línea 10*).

2. Que la posición actual del murciélago sea de mayor calidad que la del mejor murciélago de la colonia en la iteración anterior (ver *línea 10*).

Si estos dos criterios se cumplen la posición y velocidad del murciélago se actualiza tomando los resultados obtenidos en las *ecuaciones 1.18 y 1.20* (ver *líneas 11 y 12*). Posteriormente los valores del volumen (A) y del pulso (r) se actualiza de acuerdo a las *ecuaciones 1.21 y 1.22 (*ver *líneas 13 y 1*4)

$$A_i^{k+1} = \alpha A_i^k \tag{1.21}$$

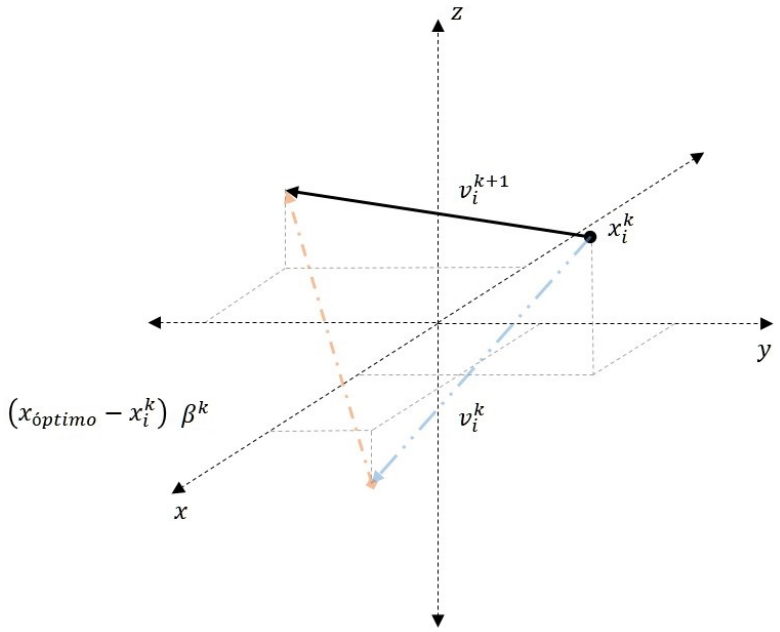
$$r_i^{k+1} = r_i^o \left(1 - e^{-\varphi * k}\right) \tag{1.22}$$

α: Constante definida por el usuario con $\alpha \in [0, 1]$.

φ: Constante definida por el usuario con $\varphi > 0$.

Finalmente, en la *Figura* 1.13 se observa la dinámica del movimiento de cada murciélago donde: la linea azul corresponde la velocidad actual del murciélago, la línea roja a la búsqueda local y la línea negra corresponde a la suma geométrica de cada uno de los movimientos la cual nos indica la nueva posición del murciélago.

Figura 1.13.: Determinación de la dirección de búsqueda para la metaheurística del murciélago.



Modificación: Las modificaciones propuestas se realizaron con base al *Algoritmo* 5, donde las variables generadas aleatoriamente en las *líneas 7 y 10* por γ y λ los cuales son valores generados a partir del mapa caótico de la *Ecuación* 1.6. La modificaciones propuestas se presentan el *Algoritmo 6*.

Algoritmo 6. Metaheurística del murciélago basada en caos

1 Generar, posición y velocidad inicial X y V
2 Inicializar, $r, f^{\min}, f^{\max}, \epsilon, A, \beta, \alpha, \varphi, \gamma, \lambda \in [0, 1]$,
3 **while** convergencia de la población
4 **foreach** $(x) \in X$
5 $\lambda \leftarrow \text{Caos}\,(\lambda)$
6 $\gamma \leftarrow \text{Caos}\,(\gamma)$
7 $f \leftarrow \text{Calcular}\,(f_{min}, f_{max}, \beta)$
8 $v_{aux} \leftarrow \text{Calcular}\,(x, x_{optimo}, f, v)$
9 **if** $\gamma > r$
10 $x_{local} \leftarrow \text{Local}\left(x_{optimo}, \epsilon, \overline{A}\right)$
11 hacer, $x_{aux} \leftarrow x + v_{aux} + x_{local}$
12 **if** $\lambda < A$ and $f(x) < f(x_{optimo})$
13 Hacer, $x \leftarrow x_{aux}$
14 Hacer, $v \leftarrow v_{aux}$

15 $A \leftarrow \text{Actualizar}\,(A, \alpha)$
16 $r \leftarrow \text{Actualizar}\,(r, \varphi)$

1.3 BÚSQUEDA ARBORESCENTE

En la tesis doctoral (**?**) se presenta el algoritmo de reconfiguración el cual busca por medio de cambios unitarios a lo largo y ancho de un vector de decisiones encontrar la mejor topología para la red de un sistema de distribución. No obstante, en este trabajo no es de intereses realizar cambios en la topología del sistema, si no que se requiere adaptar al problema la estrategia de búsqueda que emplea el algoritmo reconfiguración. Esta estrategia se llamará *búsqueda arborescente* consta de tres etapas: árbol inicial de búsqueda, búsqueda local y búsqueda global.

En las siguientes subsecciones se explica cada una de las etapas tomando como base el *Algoritmo* 7. En el *Anexo A.14* se encuentra la implementación del algoritmo el cual fue desarrollado en el lenguaje de programación Python.

Algoritmo 7. BÚSQUEDA ARBORESCENTE

1 Generar, árbol inicial de búsqueda X
2 **while** convergencia del arbol
3 Evaluar, $f\,(x)\,\forall x \in X$
4 **foreach** $(x) \in X$
5 $x' \leftarrow \text{Local}\,(x)$
6 Evaluar, $f\,(x')$
7 $x \leftarrow \text{Global}\,(x', f(x)`, x, f(x)))$

1.3.1 Árbol inicial de búsqueda

Este árbol inicial de búsqueda radica (ver *línea 1*) en la construcción de una lista que permita reducir el espacio de soluciones y ademas evite la exploración en zonas no factibles que conlleven a mínimos locales o a mayores tiempos de simulación. Es decir, se debe identificar zonas factibles con base a estudios previos del problema y de acuerdo a esta información construir dicha lista.

1.3.2 Búsqueda local La función Local (ver *línea 5*) comienza evaluando el árbol inicial, realizando cambios unitarios a lo largo de la lista y registrando los mejores cambios observados (**?**).

1.3.3 Búsqueda global La función Global (ver *línea* 8) toma decisiones que permitan organizar la información con el fin de mejorar la función de evaluación a partir de la percepción que se obtenga de la búsqueda local. De acuerdo con (**?**, 89) esta búsqueda se realiza organizando las decisiones en dos formas :

En profundidad: Se toma el mejor cambio realizado en la búsqueda local y se congela dicho cambio, posteriormente se realiza una nueva búsqueda local; al finalizar se compara la mejor solución obtenida en curso con la solución obtenida para la búsqueda local anterior, si esta llegase hacer igual o de peor calidad se puede decir que el proceso ha terminado.

En anchura: Descongelando la decisión tomada en profundidad y congelando la peor decisión encontrada en la búsqueda local con el objetivo de seguir buscando en profundidad. Este tipo de búsqueda permite un mayor grado de ergodicidad pues evita converger prematuramente en óptimos locales.

1.4 Funciones de prueba

Para analizar el comportamiento de los heurísticas descritas en secciones anteriores, se utilizaron 5 funciones de prueba que tiene un único mínimo global. En la *Tabla* 1.1 se muestra la información correspondiente a la posición del óptimo global, el valor de la función en el óptimo global y el rango en el espacio de búsqueda.

Tabla 1.1.: Funciones de prueba

Nombre	***Definición***	***X_ opt***	***f(X_ opt)***	***Rango***
N.N	$f(X) = x_1 * sin(4 * x_1) + 1{,}1 * x_2 * cos(2 * x_2)$	(9,038; 8,66)	$-18,54$	$0 \leq x_i \leq 10$
Beale's	$f(X) = (1{,}5 - a + a * b)^2 + (2{,}25 - a + a * b^2)^2$ $+(2{,}625 - a + a * b^3)^2$	(3; 0,5)	0	$-4{,}5 \leq x_i \leq 4{,}5$
Matyas	$f(X) = 0{,}26 * \left(x_1^2 + x_2^2\right) - 0{,}48 * x_1 * x_2$	(0; 0)	0	$-10 \leq x_i \leq 10$
Booth's	$f(X) = (x_1 + 2 * x_2 - 7)^2 + (2 * x_1 + x_2 - 5)^5$	(1; 3)	0	$-10 \leq x_i \leq 10$
Ackley's	$f(X) = -20 * e^{-0{,}2 * \sqrt{0{,}5 * (X_1^2 + X_2^2)}}$ $-e^{0{,}5 * (cos(2 * \pi * x_1) + cos(2 * \pi * x_2))} + e + 20$	$(0, 0)$	0	$-10 \leq x_i \leq 10$

Para estas pruebas se tomó los siguientes criterios :

1. La población inicial se establece en 100 individuos.

2. La búsqueda se detendrá cuando el mejor individuo de la población no presente un cambio significativo en 10 iteraciones consecutivas.

3. Las parámetros considerados para determinar el mejor desempeño computacional son: la precisión de la respuesta y el tiempo de simulación.

TABLA 1.2.: Funciones de pruebas: resultados

Función	***Heurística***	***X_ opt***	***f(X_ opt)***	***t [s]***	k
N.N	Algoritmo genético	(9,042; 8,670)	$-18,553$	14,1	20
	Cúmulo de partículas	(9,038; 8,668)	$-18,554$	6,3	29
	Metaheurística del murciélago	(9,040; 8,662)	$-18,553$	10,6	32
	Búsqueda Caótica repetitiva	(9,104; 8,748)	$-18,118$	5,2	5
Beale's	Algoritmo genético	(2,999; 0,497)	0,000	11,4	13
	Cúmulo de partículas	(2,999; 0,499)	0,000	5,4	21
	Metaheurística del murciélago	(3,001; 0,500)	0,000	14,6	73
	Búsqueda Caótica Repetitiva	(3,040; 0,523)	0,000	4,9	5
Matyas	Algoritmo Genético	(0,205; 0,205)	0,001	22,4	30
	Cúmulo de Partículas	(0,000; 0,000)	0,000	8,3	18
	Metaheurística del murciélago	(0,000; 0,000)	0,000	15	18
	Búsqueda Caótica Repetitiva	$(-0,085; -0,099)$	0,000	4,5	16
Booth's	Algoritmo Genético	(0,967; 2,903)	0,007	9,8	3
	Cúmulo de Partículas	(1,000; 2,999)	0,000	4,5	20
	Metaheurística del murciélago	(1,000; 2,999)	0,000	6,5	40
	Búsqueda Caótica Repetitiva	(1,1624; 2,825)	0,0572	9,2	44
Ackley's	Algoritmo Genético	(0,009; 0,009)	0,0327	14,6	13
	Cúmulo de Partículas	(0,000; 0,000)	0,000	6,0	34
	Metaheurística del murciélago	(0,000; 0,000)	0,000	9,0	49
	Búsqueda Caótica Repetitiva	$(-0,105; -0,105)$	0,815	11,4	54

Para todos los casos considerados, las heurísticas implementadas fueron capaces de encontrar el óptimo global. Igualmente, los algoritmos convergen rápidamente y escapan con facilidad de los óptimos locales.

2

Flujo de potencia

En este capítulo se describe los algoritmos implementados para resolver el flujo de potencia, tomando como referencia en primer lugar el método de *Newton-Raphson con sistema P.U. complejo,* propuesto en (**?**) el cual se caracteriza por resolver sistemas de distribución con una alta relación R/X, en segundo lugar el método *Fujo Radial TRX,* propuesto por (**?**), el cual se caracteriza por resolver sistemas de distribución radiales de gran tamaño.

La organización de este capítulo se compone de la siguiente manera: en la *Sección 2.1* se explica la estructura del algoritmo *Newton Rapson con sistema P.U. complejo* y en la *Sección 2.2* se explica la estructura del algoritmo *Flujo Radial TRX*.

2.1 Newton Raphson con sistema P.U. complejo

El método de Newtoh Raphson (NR) es un método basado en un fundamento matemático. Este método utiliza un proceso iterativo mediante el cual se aproxima a la solución linealizando las ecuaciones de potencia:.

$$P_i = |V_i|^2 G_{ii} + \sum_{j=1}^{n} |V_i|\,|V_j|\,|Y_{ij}|\,cos(\theta_{ij} + \delta_j - \delta_i) \tag{2.1}$$

$$Q_i = |-V_i|^2 B_{ii} - \sum_{j=1}^{n} |V_i|\,|V_j|\,|Y_{ij}|\,sin(\theta_{ij} + \delta_j - \delta_i) \tag{2.2}$$

P_i, es la potencia activa inyectada a la barra i.

Q_i, es la potencia reactiva inyectada a la barra i.

V_i, es la tensión en la barra i.

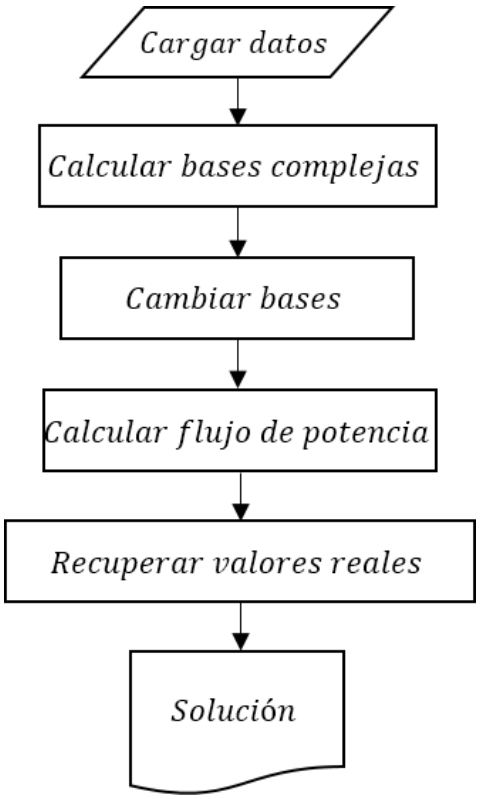
$Y_{i,j}$, es la admitancia entre la barra i y la barra j.

Este método es muy efectivo para resolver grandes sistemas de potencias, pero presenta problemas de convergencia o tiempos ineficientes de solución para sistemas de distribución con una alta relación R/X (**?**).

Una solución para este problema se plantea en (**?**), donde se propone cambiar las bases reales por bases complejas. Este cambio de base permite reducir la relación R/X y así ejecutar el método de NR, pero con la salvedad que la solución encontrada no corresponde a los valores verdaderos. Para recuperar estos valores se debe hacer el cambio de base inverso.

A continuación en la *Figura 2.1* se presenta el diagrama de flujo correspondiente al flujo Newton Raphon con sistema P.U. complejo, tomando como base la metodología empleada en (**?**). Ademas, en las siguientes subsecciones se da una explicación del mismo acompañada de un ejemplo de aplicación y validación.

Figura 2.1.: Diagrama de flujo, Newton Raphson con sistema P.U. complejo



2.1.1 Cargar datos

En los *Anexos* A.3, A.4, A.5, A.6 y A.7, se encuentra la información en detalle de las características eléctricas de las líneas, cargas y generadores que fueron empleados para cada caso de prueba.

2.1.2 Calculo de bases complejas

La bases complejas son calculadas en cuatro pasos:

1. Se calcula el promedio de los ángulos de la impedancias serie para cada línea k del sistema (sindo k, el número total de líneas).

$$\alpha_{avg} = \frac{\sum_{k=1}^{K} tan^{-1}\left(\frac{X_k}{R_k}\right)}{K} \tag{2.3}$$

2. Se calcula el promedio entre el menor y mayor angulo de la impedancia serie de las líneas:

$$\gamma_{avg} = \frac{tan^{-1}\left(\frac{X_k}{R_k}\right)_{máx} + tan^{-1}\left(\frac{X_k}{R_k}\right)_{mín}}{2} \tag{2.4}$$

3. Se calcula la corrección con base al promedio de los factores de potencias instalada para cada carga d (siendo D, el número total de cargas):

$$\varepsilon = 1 - \frac{\sum_{d=1}^{D} cos\left(tan^{-1}\left(\frac{Q_d}{P_d}\right)\right)}{D} \tag{2.5}$$

4. Finalmente, se calcula en ángulo base de acuerdo a la siguiente ecuación:

$$\phi_{base} = \left(\frac{\pi}{2} - \frac{\alpha_{avg} + \gamma_{avg}}{2}\right)(1 + \varepsilon) \tag{2.6}$$

A partir de ϕ calculado y tomando como supuesto que las bases originales del sistema de potencia que pertenecen a los números reales, se prosigue a calcular las bases complejas de acuerdo a las siguientes ecuaciones.

$$S_{cplx} = S_{base}e^{-j\phi_{base}} = |S_{cplx}|\, e^{-j\phi_{base}} \tag{2.7}$$

$$V_{cplx} = V_{base}e^{j0} = V_{base} \tag{2.8}$$

$$Z_{cplx} = Z_{base}e^{-j\phi_{base}} = \frac{V_{cplx}^2}{S_{cplx}^*} = |Z_{cplx}|\, e^{-j\phi_{base}} \tag{2.9}$$

$$I_{cplx} = I_{base}Z_{base}e^{-j\phi_{base}} = \frac{V_{cplx}}{Z_{cplx}} = |I_{cplx}|\, e^{-j\phi_{base}} \tag{2.10}$$

Para el cálculo de las bases complejas se creó la función *cplx_ base(),* la cual se encuentra en el *Anexo A.18.* Esta función tiene como dato de entra da la lista *mpc* y como salida S_{cplx}, V_{cplx}, Z_{cplx} e I_{cplx} dentro de la lista *base_ compleja.*

2.1.3 Cambio de bases Esta etapa consiste en realizar el cambio de base ordinario y considerando que la lista *bus (*ver *subsección 3.1.1)* del caso del sistema de distribución se encuentra en pu, se prosigue a calcular los nuevos valores de R_{cpu}, X_{cpu}, P_{cpu} y Q_{cpu} de acuerdo a las ecuaciones 2.12, 2.13, 2.15 y 2.16 respectivamente:

$$Z_{cpu} = (R + jX)\frac{Z_{base}}{Z_{cplx}} = |Z_{cpu}|\, e^{j(\theta+\phi_{base})} \tag{2.11}$$

$$R_{cpu} = |Z_{cpu}|\, cos(\theta + \phi_{base}) \tag{2.12}$$

$$X_{cpu} = |Z_{cpu}|\, sin(\theta + \phi_{base}) \tag{2.13}$$

$$S_{cpu} = (P + jQ)\frac{S_{base}}{S_{cplx}} = |S_{cpu}|\, e^{j(\theta+\phi_{base})} \tag{2.14}$$

$$P_{cpu} = |S_{cpu}|\, cos(\theta + \phi_{base}) \tag{2.15}$$

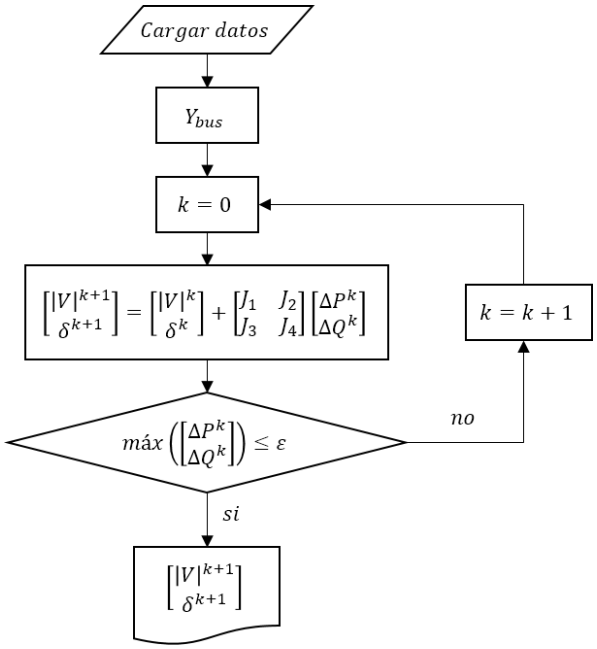
$$Q_{cpu} = |S_{cpu}|\, sin(\theta + \phi_{base}) \tag{2.16}$$

Para el calculo del cambio de base se creó la función *cpu_ normalization(),* la cual se encuentra en el *Anexo A.18.* Esta función tiene como datos de entrada las listas *mpc* y *base_ compleja* *y* como salida devuelve la lista mpc_{cpu} con los cambios de bases correspondiente.

2.1.4 Calculo del flujo de potencia

El algoritmo de Newton Raphson implementado sigue la metodología propuesta en (**?**).

Figura 2.2.: Diagrama de flujo para el método de Newton Rapshon



Para esto se creó la función *runpf(),* la cual tiene como parámetros de entrada la lista *mpc* y como parámetro de salida la lista r[1]. El algoritmo se presenta en el *Anexo A.18*

A continuación en la *Figura 2.3*, se presenta un ejemplo para verificar la validez de los resultados del método de NR implementado. Para esto tomaremos el caso de prueba *'case_ 4gs'*[2] propuesto en (**?**), el cual consta de 2 generados, 4 barras y tres cargas.

[1] r, es semejante a *mpc,* pero con los valores de la lista *bus* actualizados para la tensión y potencia calculadas.
[2] Este caso corresponde a un sistema de potencia por lo cual al relación de R/X es baja y el método de NR lo resuelve fácilmente.

Figura 2.3.: Solución del flujo de potencia mediante el método de Newton Raphson para el sistema de potencia de 4 barras, *'caso_4gs'*

```
from newton_raphson import runpf
from resultados import solution
from case import loadcase

mpc = loadcase('case_4gs')
r = runpf(mpc)
solution(r)
```

Bus	Tipo	P_g [MW]	Q_g [MVAR]	P_d [MW]	Q_d [MVAR]	Q_s [MVAR]	Vpu	Deg
1	Slack	186.809	114.5	50.0	30.99	0.0	1.0	0.0
2	Carga	0.0	0.001	170.0	105.35	0.0	0.982	-0.976
3	Carga	0.0	0.0	200.0	123.94	0.0	0.969	-1.872
4	Voltaje	318.0	181.429	80.0	49.58	0.0	1.02	1.523
		504.809	295.93	500.0	309.86			

2.1.5 Cambio de base inverso

Esta operación permite recuperar la solución del caso tras resolver el flujo de potencia. Para esto se creó la función *cplx_real()* (ver *Anexo*A.18), la cual tiene como parámetro de entrada la lista r_{cpu} y como parámetro de salida la lista r con valores reales.

$$S = (P_{cpu} + jQ_{cpu})\, S_{cplx} \tag{2.17}$$

$$P = |S|\, cos(\theta) \tag{2.18}$$

$$Q = |S|\, sin(\theta) \tag{2.19}$$

$$Z = (R_{cpu} + jX_{cpu})\, Z_{cplx} \tag{2.20}$$

$$R = |Z|\, cos(\theta) \tag{2.21}$$

$$X = |Z|\, sin(\theta) \tag{2.22}$$

2.1.6 Ejemplo para el sistema de distribución de 7 barras

A continuación se presenta un caso de prueba para dar un mayor entendimiento del método propuesto con anterioridad. Para esto, se resuelve el flujo de potencia para el sistema de distribución propuesto en (**?**) el cual se compone 7 barras, 6 líneas, 6 cargas y 1 generador (ver *Figura* 2.4). Los datos de buses y líneas se presentan en las *Tablas 2.1 y 2.2,* respectivamente (ver *Anexo A.4).*

Figura 2.4.: Diagrama unifilar para el sistema de distribución de 7 barras.

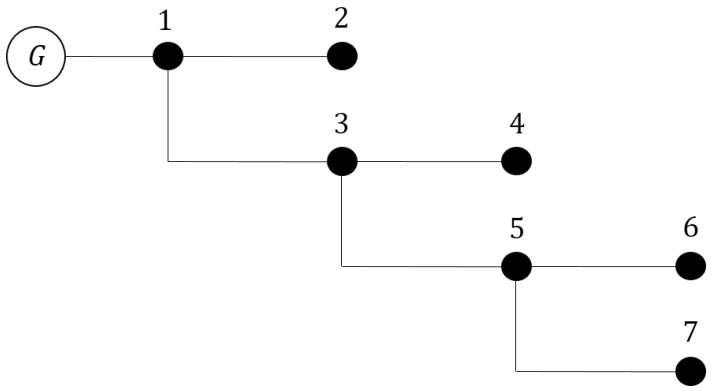


TABLA 2.1.: Datos de barra para el sistema de distribución de 7 barras.

Barra	***Generación***		***Carga***		***V [p.u]***	***Tipo***
	P [pu]	***Q [pu]***	***P [pu]***	***Q [pu]***		
1	–	–	0	0	$1\angle 0$	Slack
2	0	0	0,1017	0,0635	$1\angle 0$	Carga
3	0	0	0,0547	0,0342	$1\angle 0$	Carga
4	0	0	0,0809	0,0596	$1\angle 0$	Carga
5	0	0	0,1017	0,0635	$1\angle 0$	Carga
6	0	0	0,0544	0,0342	$1\angle 0$	Carga
7	0	0	0,0809	0,0596	$1\angle 0$	Carga

TABLA 2.2.: Datos de línea para el sistema de distribución de 7 barras.

Barra	***Serie Z***		***Y en paralelo***
	R [p.u]	***X [p.u]***	***Y/2 [p.u]***
1-2	0,0265	0,0462	0
1-3	0,1005	0,0693	0
3-4	0,0670	0,0462	0
3-5	0,0265	0,0462	0
5-6	0,1005	0,0693	0
5-7	0,0670	0,0462	0

La bases utilizadas para este caso son las siguiente: $V_{base} = 12{,}47\,[kV]$ y $S_{base} = 1\,[MVA]$.

El primer paso consiste en calcular el ángulo base y bases complejas de acuerdo a los descrito en *sección 2.1.2* (en la *Tabla 2.3* se presentan los resultados).

TABLA 2.3.: Ángulo base y bases complejas

Parámetro		
α_{avg}	43,1127	*Ecuación 2.3*
γ_{avg}	47,3750	*Ecuación 2.4*
ε	0,2119	*Ecuación 2.5*
ϕ_{base}	54,2436	*Ecuación 2.6*
S_{cplx}	$1e^{-j54,2587}$	*Ecuación* 2.7
V_{cplx}	12,47	*Ecuación 2.8*
Z_{cplx}	$1{,}555e^{-j54,2587}$	*Ecuación 2.9*
I_{cplx}	$8{,}0192e^{-j54,2587}$	*Ecuación 2.10*

El segundo paso consiste en calcular los nuevos valores para R y X (de acuerdo a lo descrito en la *subsección* 2.1.3). En la *Tabla 2.4* se presentan los cálculos en el siguiente orden: primera y segunda columna corresponde a R_{pu} y X_{pu}, la columna 3 corresponde a al relación R/X, la columna cuatro y cinco corresponde a R_{cpu}y X_{cpu} tras aplicar el cambio de base complejo y la columna 6 corresponde a la nueva relación R/X.

TABLA 2.4.: Impedancia serie - *cpu normalization*

Branch	***R [pu]***	***X [pu]***	***R/X***	***R [cpu]***	***X [cpu]***	***R/X***
1	0.0265	0.0462	0.5735	-0.0220	0.04849	0.4540
2	0.1005	0.0693	1.4502	0.0024	0.1220	0.0201
3	0.0670	0.0462	1.4502	0.0016	0.0813	0.0201
4	0.0265	0.0462	0.5735	-0.0220	0.0484	0.4540
5	0.1005	0.0693	1.4502	0.0024	0.1220	0.0201
6	0.0670	0.0462	1.4502	0.0016	0.0813	0.0201

Como se observa en la *Tabla 2.4* la relación R/X de la columna 7 es menor en comparación a la relación de la columna 4, lo cual nos garantiza el poder aplicar el método de NR.

De igual forma en la *Tabla 2.5* se presenta los nuevos valores para la potencia activa y reactiva.

TABLA 2.5.: Potencia activa y reactiva con bases reales y complejas.

P[pu]	***Q[pu]***	***P[cpu]***	***Q[cpu]***
0.1017	0.0635	0.0079	0.1196
0.0547	0.0342	0.0042	0.0644
0.0809	0.0596	-0.0011	0.1005
0.1017	0.0342	0.0079	0.1196
0.0544	0.0342	0.0042	0.0643
0.0809	0.0596	-0.0011	0.1005

El tercer paso consiste en resolver el flujo de potencia con los nuevos valores de R, X, P y Q calculados.

En la *Figura 2.5,* se presenta el ejemplo para el cálculo del flujo de potencia con los nuevos parámetros calculados. Es importante resaltar que sin el cambio de base complejo de, no es posible resolver el flujo de potencia.

Figura 2.5.: Solución del flujo de potencia mediante el método de Newton Raphson con sistema P.U. complejo.

```
from case import loadcase
from newton_raphson import runpf, cplx_bases, cpu_normalization, reverse_cpu_normalization
from resultados import solution

mpc = loadcase('case_7')
base_complejas = cplx_bases(mpc)
mpc_cplx = cpu_normalization(mpc, base_complejas)
print 'cplx'
r_cpu = runpf(mpc_cplx)
solution(r_cpu)
print 'Real'
r = reverse_cpu_normalization(r_cpu)
solution(r)
```

```
Bus    Tipo      P_g [MW]    Q_g [MVAR]    P_d [MW]    Q_d [MVAR]    Q_s [MVAR]    Vpu      Deg
-----  ------  ----------  ------------  ----------  ------------  ------------  -------  -------
  1    Slack         0.02         0.606        -0.0           0.0           0.0      1.0      0.0
  2    Carga          0.0           0.0       0.008          0.12           0.0    0.994   -0.174
  3    Carga          0.0           0.0       0.004         0.064           0.0    0.941   -0.021
  4    Carga          0.0           0.0      -0.001           0.1           0.0    0.932   -0.004
  5    Carga          0.0           0.0       0.008          0.12           0.0    0.926    -0.47
  6    Carga          0.0           0.0       0.004         0.064           0.0    0.917   -0.494
  7    Carga          0.0           0.0      -0.001           0.1           0.0    0.917   -0.453
-----  ------  ----------  ------------  ----------  ------------  ------------  -------  -------
                      0.02         0.606       0.022         0.568
-----  ------  ----------  ------------  ----------  ------------  ------------  -------  -------

Real

Bus    Tipo      P_g [MW]    Q_g [MVAR]    P_d [MW]    Q_d [MVAR]    Q_s [MVAR]    Vpu      Deg
-----  ------  ----------  ------------  ----------  ------------  ------------  -------  -------
  1    Slack        0.503         0.337         0.0           0.0           0.0      1.0      0.0
  2    Carga          0.0           0.0       0.102         0.063           0.0    0.994   -0.174
  3    Carga          0.0           0.0       0.055         0.034           0.0    0.941   -0.021
  4    Carga          0.0           0.0       0.081          0.06           0.0    0.932   -0.004
  5    Carga          0.0           0.0       0.102         0.063           0.0    0.926    -0.47
  6    Carga          0.0           0.0       0.055         0.034           0.0    0.917   -0.494
  7    Carga          0.0           0.0       0.081          0.06           0.0    0.917   -0.453
-----  ------  ----------  ------------  ----------  ------------  ------------  -------  -------
                     0.503         0.337       0.476         0.314
```

Para validar los resultados en la *Tabla 2.6* se calcula el error de las magnitudes de las tensiones y del ángulo de los buses respecto a los valores teóricos que se presentan en (**?**).

TABLA 2.6.: Validación de los resultados para el método de NR-C aplicado al caso de 7 barras.

Bus	***Teórico***		***Calculado***		***Error***	
	$\|V\|\ [pu]$	$\angle [Deg]$	$\|V\|\ [pu]$	$\angle [Deg]$	$\|V\|\ [\%]$	$\angle [\%]$
1	1	0	1	0	N.A	N.A
2	0.9943	-0.1738	0.9943	-0.1738	0	0
3	0.9407	-0.0206	0.9407	-0.0206	0	0
4	0.9320	-0.0039	0.9320	-0.0039	0	0
5	0.9260	-0.4699	0.9260	-0.4699	0	0
6	0.91774	-0.4938	0.9174	-0.4938	0	0
7	0.9171	-0.4527	0.9171	-0.4527	0	0

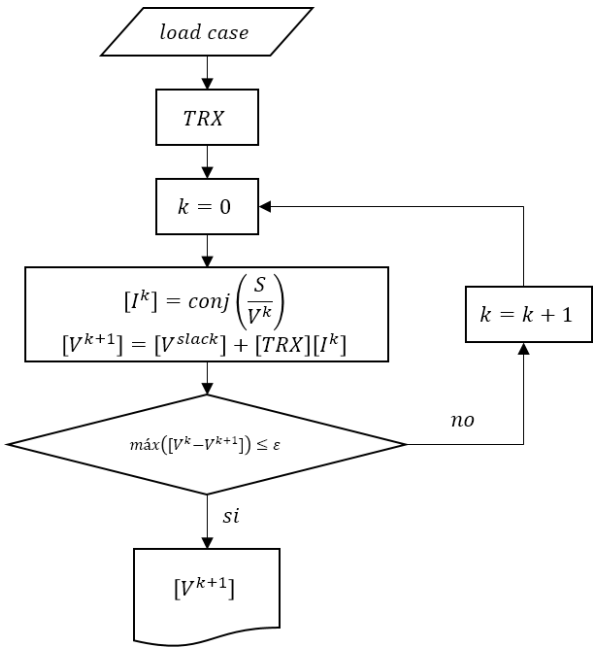
2.2 Flujo radial TRX

Este algoritmo es útil para ser aplicado con propósitos de planificación y evaluación del sistema de distribución en tiempo real. Ademas, este algoritmo se caracteriza por utilizar una matriz única T para determinar el estado del sistema y una matriz Z que refleja la impedancia del sistema asociado.

Una de las característica a resaltar de este algoritmo es el de no necesitar del cálculo de la matriz de admitancias ni la matriz jacobiana, por lo cual la cantidad de cálculos en comparación al método de Newton Raphson es menor, lo que se traduce en un menor tiempo de computo. Pero, a pesar de tener buenos tiempos de simulación, este método tiene un margen de error a medida que se aumenta el número de barras en el sistema de distribución (?).

A continuación en la *Figura 2.6* se presenta el diagrama de flujo correspondiente al flujo radial TRX, tomando como base la metodología empleada en (?). Ademas, en las siguientes subsecciones se da una explicación del mismo acompañada de un ejemplo de aplicación y validación.

Figura 2.6.: Diagrama de flujo, flujo Radial TRX



2.2.1 Cargar caso En los *Anexos* A.4, A.5 y A.6, se encuentra la información en detalle de las características eléctricas de las líneas, cargas y generadores que fueron empleados para cada caso de prueba.

2.2.2 Cálculo de la matriz TRX

Para el cálculo de la matriz TRX, se toma como ejemplo el sistema de distribución de 7 barras. Partiendo de la ley de Kirchhoft se calcula las corrientes de rama en función de las corrientes inyectadas, tal como se muestra a continuación.

$$B_1 = I_1$$

$$B_2 = I_2 + I_3 + I_4 + I_5 + I_6 + I_7$$

$$B_3 = I_3$$

$$B_4 = I_5 + I_6 + I_7$$

$$B_5 = I_6$$

$$B_6 = I_7$$

De esta forma, la relación entre las corrientes de rama y corrientes inyectadas se pueden escribir de forma matricial como se muestra en la *Ecuación 2.23.*

$$B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{bmatrix} \tag{2.23}$$

La *Ecuación* 2.23 se puede reescribir en forma general como se muestra a continuación:

$$[B] = [T]\,[I] \tag{2.24}$$

Por lo tanto la matriz T que representa al sistema de distribución de 7 barras corresponde a:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

Ahora, para calcular la matriz la matriz Z de impedancias, solo basta con construir una matriz diagonal cuya diagonal sea las impedancias de líneas.

$$Z = \begin{bmatrix} 0,0265 + j0,0462 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,1005 + j0,0693 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,0670 + j0,0462 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,0265 + 0,0462 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,1005 + j0,0693 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,0670 + j0,0462 \end{bmatrix}$$

Finalmente, la matriz TRX se calcula de acuerdo a la *Ecuación 2.26.*

$$TRX = \left[T^T\right][Z][T] \tag{2.26}$$

2.2.3 Proceso iterativo

Una vez calculada al matriz TRX, se prosigue a calcular el vector de corrientes inyectadas de acuerdo con:

$$I = conj\left(\frac{S}{V}\right)$$

S, Vector de demanda del sistema de distribución.

V, Vector de voltajes correspondiente a cada barra.

Finalmente, se calcula las tensiones en las barras de acuerdo a la *Ecuación 2.27.*

$$[V] = [V_0] - [TRX][I] \tag{2.27}$$

El proceso se detiene cuando la diferencia del voltaje entre iteraciones es menor a la tolerancia establecida.

$$\left|V^{k+1} - V^k\right| \leq e$$

2.2.4 Ejemplo para el sistema de distribución de 7 barras

En la *Figura 2.7,* se presenta el ejemplo para el cálculo del flujo radial para el sistema de distribución de 7 barras. Los datos y características de este ejemplo son los mismos que se utilizaron en la subsección 2.1.6.

Figure 2.7.: Solución del flujo de potencia mediante el método de flujo radial TRX

```
from case import loadcase
from flujo_radial import runTRX
from resultados import solution

mpc = loadcase('case_7')
r = runTRX(mpc)
solution(r)
```

Bus	Tipo	P_g [MW]	Q_g [MVAR]	P_d [MW]	Q_d [MVAR]	Q_s [MVAR]	Vpu	Deg
1	Slack	0.503	0.337	0.0	0.0	0.0	1.0	0.0
2	Carga	0.0	0.0	0.102	0.064	0.0	0.994	-0.174
3	Carga	0.0	0.0	0.055	0.034	0.0	0.941	-0.021
4	Carga	0.0	0.0	0.081	0.06	0.0	0.932	-0.004
5	Carga	0.0	0.0	0.102	0.064	0.0	0.926	-0.47
6	Carga	0.0	0.0	0.055	0.034	0.0	0.917	-0.494
7	Carga	0.0	0.0	0.081	0.06	0.0	0.917	-0.453
		0.503	0.337	0.476	0.316			

Para validar los resultados, en la *Tabla* 2.7 se calcula el error de las magnitudes de las tensiones y del ángulo de los buses respecto a los valores teóricos que se presentan en (**?**).

Table 2.7.: Validación de los resultados para el método de flujo radial TRX, aplicado al caso de 7 barras

Bus	***Teórico***		***Calculado***		***Error***	
	$\|V\|\ [pu]$	$\angle [Deg]$	$\|V\|\ [pu]$	$\angle [Deg]$	$\|V\|\ [\%]$	$\angle [\%]$
1	1	0	1	0	N.A	N.A
2	0.9943	-0.1738	0.9943	-0.1738	0	0
3	0.9407	-0.0206	0.9407	-0.0206	0	0
4	0.9320	-0.0039	0.9320	-0.0039	0	0
5	0.9260	-0.4699	0.9260	-0.4699	0	0
6	0.91774	-0.4938	0.9174	-0.4938	0	0
7	0.9171	-0.4527	0.9171	-0.4527	0	0

3

IMPLEMENTACIÓN

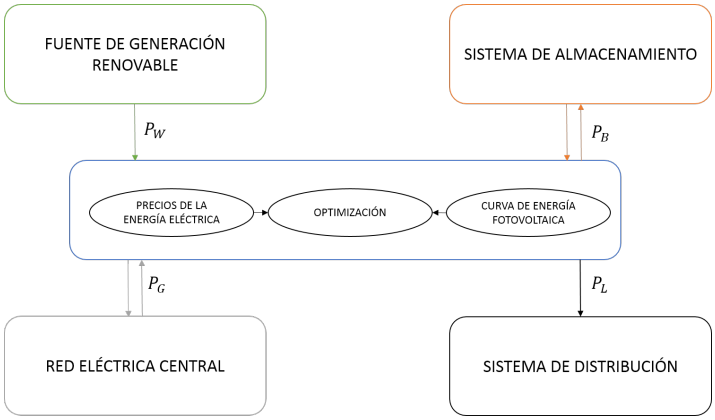
En este proyecto se propone un esquema de optimización para un sistema de distribución basado en perfiles de energía fotovoltaica, demanda y precio de compra/venta de la energía eléctrica para una ventana de tiempo de 24 horas. Por otro lado, dado a que las fuentes de generación renovable en su mayoría son de naturaleza intermitente, se propone el algoritmo de optimización con la intención de tener un sistema autónomo que gestione los recursos energéticos en el sistema de distribución, de manera que se almacene o se extraiga energía de un elemento de almacenamiento, o bien, se compre o venda energía eléctrica a la red centralizada de acuerdo a los precios de compra/venta.

La organización de este capítulo se compone de la siguiente manera: en la *Sección 3.1* se explica la estructura del programa y en la *Sección 3.2* se presenta un ejemplo del proceso de optimización.

3.1 Estructura del programa

El sistema eléctrico para la cual se propone el esquema de optimización se muestra en la *Figura* 3.1. Esta figura ilustra que el flujo de energía entre los diferentes componentes del sistema está determinado mediante un algoritmos de optimización que considera perfiles de energía a fotovoltaica, demanda y precio de compra/venta de energía eléctrica.

Figura 3.1.: Esquema de optimización para un sistema de distribución rural



El algoritmo implementado se compone de tres etapas principales: la primera corresponde a los datos de entrada, de la cual se debe elegir el sistema de distribución de prueba, el método de solución del flujo de potencia, el método heurístico para la optimización y finalmente se elige un día dentro de la base de datos para cargar las curvas de demanda, perfil de energía fotovoltaica y precio de compra/venta de la energía eléctrica. La segunda etapa corresponde al proceso interactivo el cual se ejecuta de acuerdo a la heurística y parámetros de entrada seleccionados. Adicionalmente, en esta etapa se registra el consumo energético del sistema de distribución y tiempo de computo o tiempo CPU. En la última etapa se realiza un análisis de los resultados los cuales se presentan en tablas y gráficos para mayor comprensión.

El objetivo principal del esquema de optimización es determinar en que horas del día el sistema de almacenamiento debe cargar, descargar o desconectarse del sistema de distribución, con el fin que las inyecciones de potencia eléctrica de la red central sea mínima y al mismo tiempo que se maximice las inyecciones de potencia por parte del sistema de almacenamiento hacia el sistema de distribución.

3.1.1 Datos de entrada

Caso de prueba: Para este proyecto se trabajó con 6 casos de prueba los cuales se enuncian a continuación:

1. 'caso_4gs', sistema de potencia de 4 barras, 4 líneas, 3 cargas y 1 generador.
2. 'caso_7', sistema de distribución de 7 barras, 6 lineas, 6 cargas y 1 generador.
3. 'caso_13', sistema de distribución de 13 barras, 12 lineas, 12 cargas y 1 generador.
4. 'caso_33', sistema de distribución de 33 barras, 32 líneas, 32 cargas y 1 generador.
5. 'caso_30', sistema de distribución de 30 barras, 41 líneas, 20 cargas y 6 generador.

En los *Anexos* A.3, A.4, A.5, A.6 y A.7, se encuentra la información en detalle de las características eléctricas de las líneas y datos de barra. Adicionalmente en el *Anexo* A.2, se encuentra la función *loadcase()*, la cual permite cargar cada uno de los casos ya mencionados. Esta función tiene como salida seis parámetros los cuales se describen a continuación:

1. baseMVA, base de potencia en megavolteoamperio.
2. bus, esta lista contiene la información de las barras:
   a) Tipo de barra (Slack = 0, PQ = 1, PV = 2).
   b) Potencia activa generada ($P_G\,[pu]$).
   c) Potencia reactiva generada ($Q_G\,[pu]$).
   d) Potencia activa demandada ($P_D\,[pu]$).
   e) Potencia reactiva demandada ($Q_{D\,[pu]}$).
   f) Potencia shunt ($P_{S\,[pu]}$).
   g) Magnitud del voltaje en la barra ($|V|\ [pu]$).
   h) Angulo del voltaje en la barra ($\delta\,[deg]$).
   i) Tensión base.
3. branch, esta lista contiene la información de las características eléctricas de las líneas:
   a) Barra inicial.
   b) Barra final.
   c) Resistencia serie ($R\,[pu]$).
   d) Reactancias. serie ($X\,[pu]$).
   e) Admitancia paralelo ($B/2\,[pu]$).
4. N_bus, número de barras.
5. N_branch, número de líneas.
6. basekV, tensión base en kilovolteos.
7. N_gen, número de generadores.

Método para solucionar el flujo de potencia: De acuerdo a lo descrito en el *Capítulo* 2, para este proyecto se cuenta con 3 métodos para resolver flujos de potencia los cuales se enuncian a continuación:

1. 'NR', Newton Rapson.
2. 'NR_CPLX', Newton Rapso complex pu normalization.
3. 'TRX', Flujo radial TRX.

Método heuristico: De acuerdo a lo descrito en el *Capítulo* 1, para este proyecto se cuenta con 5 métodos heurísticos los cuales se enuncian a continuación:

1. 'AG', algoritmos genético.
2. 'RSS', búsqueda aleatoria repetitiva.
3. 'PSO', cúmulo de partículas.
4. 'H_BAT', metaheurística del murciélago.
5. 'B_A', búsqueda arborescente.

Base de datos: Para este proyecto se cuenta con una amplia base datos, de la cual se tiene 264 días para precio de compra/venta de la energía eléctrica, demanda y perfil de energía fotovoltaica. Para cargar la información se creó la función *pronostico()* (ver *Anexo* A.8), esta función permite elegir entre los 264 días y con ventanas máximas de 24 horas.

Demanda del sistema de distribución y potencia de la fuente de generación renovable: De acuerdo a lo descrito anteriormente, la función *pronosticos()* carga tres listas donde: C_G corresponde al precio de compra/venta de la energía eléctrica; P_w y $P_l \in [0, 1]$ (adimensional), corresponden al perfil de energía fotovoltaico y demanda. Ahora bien, para el cálculo de la demanda en la ventana de tiempo seleccionado se realiza de la siguiente manera:

$$\begin{aligned} P_L &= -\left(P_D\right)\left(P_l\right)\ [MW] \\ Q_L &= -\left(Q_D\right)\left(P_l\right)\ [Mvar] \end{aligned} \tag{3.1}$$

De igual forma, para el cálculo de la curva de potencia en la fuente de generación renovable se realiza de la siguiente manera:

$$P_W = (K)(P_w)\,[MW] \tag{3.2}$$

K, es un factor que indica la máxima potencia de generación.

Potencia y capacidad del sistema de almacenamiento: El sistema de almacenamiento para este proyecto tiene dos parámetros de entrada: en primer lugar la capacidad máxima de almacenamiento, y en segundo lugar la potencia, la cual indica cuanto de la capacidad máxima de almacenamiento se puede inyectar.

Barra para implementar fuente de generación renovable y sistema de almacenamiento: Esta información permite elegir en que barra se implementa el sistema de almacenamiento o fuente de generación renovable. Ademas, permite elegir el porcentaje de la de potencia para instalar en la barra. Por ejemplo, en un sistema de distribución de 13 barras se tiene una de potencia de 3 [MW] para el sistema de almacenamiento, y se desea instalar en la barra 13 el 70 % de esta potencia y en la barra 10 el 30 % restante, entonces la forma de ingresar esta información es de la siguiente manera:

$$bus = \begin{bmatrix} 13 & 10 \\ 0{,}7 & 0{,}3 \end{bmatrix} \tag{3.3}$$

La primera fila hace referencia a la posición que tendrá el sistema de almacenamiento o fuente de generación renovable en el sistema de distribución, y la segunda fila corresponde al porcentaje de la capacidad de potencia que se tendrá.

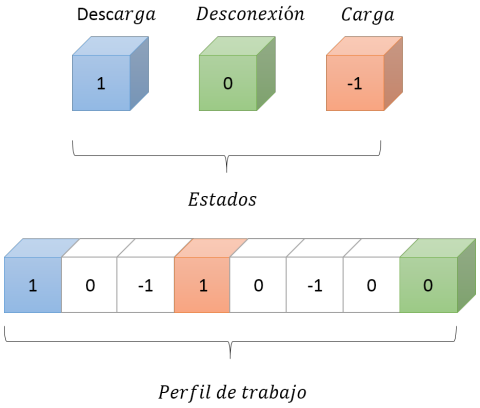
3.1.2 Procesos iterativo

Cálculo del tiempo CPU: Para este proyecto el tiempo de computo se considera como uno de los parámetros principales de rendimiento, por tal motivo es necesario tener un registro preciso de este tiempo. Este cálculo de tiempo solo considerará la etapa iterativa, es decir solo se tomará registro del tiempo que le toma a la heurística encontrar la respuesta (ver *Anexo A.9*)

Dinámica del sistema de almacenamiento: Para la solución de este problema se requiere planificar en el tiempo el correcto funcionamiento del sistema de almacenamiento. Esta planificación tendrá como resultado un perfil de trabajo el cual indicará si el sistema de almacenamiento debe cargar, descargar o desconectarse de la red para una ventana de tiempo dada.

Como se observa en la *Figura* 3.2, el perfil de trabajo se modela como un arreglo de longitud $L = [l_1, l_2, l_{3,} ...l_n]$, donde cada componente o posición, tiene tres posibles estados. Este perfil de trabajo esta codificado de la siguiente manera: 1 representa la descarga, 0 representa la desconexión y -1 representa la carga del sistema de almacenamiento.

Figura 3.2.: Codificación del perfil de trabajo, con sus respectivos estados de operación.



A continuación se describe cada uno de los posibles estados del perfil de trabajo, ya que cada uno de estos tiene un efecto positivo, negativo o neutral en el esquema de optimización.

Carga: Este estado se realizará en primera medida con base al precio actual de la energía eléctrica, es decir, si el precio es bajo se contempla la posibilidad de cargar, pero con la salvedad de que esta acción no repercuta en las horas siguientes del día, pues cabe la posibilidad de que en ese momento se requiera de la mayor cantidad de potencia, por lo tanto conectar una carga mas al sistema puede ser desfavorable para la estabilidad del mismo.

Descarga: Este estado se realizará contrario al estado de carga pues se requiere que este se realice teniendo en cuenta dos escenarios: Primero, cuando el precio de la energía eléctrica sea alto para inyectar a la red y segundo para ser utilizada en horas de la noche tras haber sido cargada en el trascurso del día.

Desconexión: Está operación sirve para aquellas situaciones donde no sea conveniente cargar o descargar, ya que si se realiza alguna de estas dos operaciones el único efecto que se lograría en el sistema seria empeorarlo.

Hasta este punto el perfil de trabajo tiene codificada la información que corresponde a un conjunto finito estados del sistema de almacenamiento. Por lo tanto, se hace necesario decodificar la información y así lograr evaluar el perfil en la función de evaluación.

La decodificación del perfil de trabajo se realizará en 5 paso los cuales se describen a continuación (ver *Anexo A.15*):

Paso 1: Establecer los límites de carga para el sistema de almacenamiento $SOC_{máx} \in [0{,}1; 1]$ y $SOC_{mín} \in [0{,}1; 1]$.

Paso 2: Establecer la taza máxima y mínima de carga/descarga en función del $SOC_{máx}$ del sistema de almacenamiento (ver *Ecuación* 3.4). Con el objeto de garantizar la longevidad

de la batería es recomendable tomar valores constante de $\triangle SOC$, pues cargas y descargas no controladas pueden ocasionar gasificación, sulfatación o en el peor de los casos a una muerte súbita del sistema de almacenamient0.

$$\triangle SOC = a\,(SOC_{máx} - SOC_{mín}) \tag{3.4}$$

a, es un número constante $\in [0, 1]$

Paso 3: Establecer la dinámica para el sistema de almacenamiento, donde la *Ecuación* 3.5 representa la carga; la *Ecuación* 3.6 representa la descarga, la *Ecuación* 3.7 representa la desconexión y la *Ecuación* 3.8 representa el estado de carga en el tiempo del sistema de almacenamiento.

$$SOC^{-} = SOC(t-1) + \triangle SOC \tag{3.5}$$

$$SOC^{+} = SOC(t-1) - \triangle SOC \tag{3.6}$$

$$SOC^{0} = SOC(t-1) \tag{3.7}$$

$$SOC(t) = \begin{cases} SOC^{-} & si,\ L(t) = -1 \\ SOC^{0} & si,\ L(t) = 0 \\ SOC^{+} & si,\ L(t) = 1 \end{cases} \tag{3.8}$$

Paso 4: Establecer la potencia del sistema de almacenamiento en función del SOC a partir de la *Ecuación* 3.9.

$$P_B(t) = [SOC(t) - SOC(t-1)]\,\frac{P_{base}}{\triangle t} \tag{3.9}$$

P_{base}, es la capacidad máxima de almacenamiento.

$\triangle t$, es el paso de tiempo entre una descarga y otra.

De acuerdo a lo descrito, en el *Anexo A.15,* se presenta la función *dinamica_ de_ la_ bateria(),* la cual permite crear el vector de potencia del sistema de almacenamiento a partir del perfil de trabajo ingresado.

Flujo de potencia: La metodología pare correr el flujo de potencia se presenta a continuación:

1. Cargar la potencia de la fuente de generación renovable para el instante k.
2. Cargar la potencia del sistema de almacenamiento.

3. Cargar las demanda del sistema de distribución para el instante k.
4. Conectar la batería y la fuente de generación renovable en la barra o barras seleccionadas.
5. Resolver el flujo de potencia para el instante k.
6. Guardar la potencia generada, potencia demanda, magnitud y angulo de la tensión.
7. Repetir el proceso a partir de 5, para el instante de tiempo k+1 hasta T.

De acuerdo a lo descrito, en el *Anexo A.16,* se presenta la función *power_flow()* la cual permite resolver el flujo de potencia teniendo en cuenta la metodología propuesta anteriormente.

Reparador: Esta acción se ejecuta internamente en la función *dinamica_de_la_batería().* Si la tensión calculada en alguna de las barras sobre pasa el limite superior de tensión, la inyección de potencia de la batería hacia el sistema de distribución se reduce en un 90 %. Pero, si la tensión de la barra queda por debajo del limite inferior de tensión, la potencia inyectada del sistema de distribución hacia la batería se reduce en un 90 %. Esta acción se repite hasta que la tensión de la barra no sobrepase ninguno de los limites de tensión establecidos.

función de evaluación: Con el fin de disminuir las inyecciones de potencia por parte de la red eléctrica central hacia el sistema de distribución y a su vez aprovechando al máximo la energía disponible del sistema de almacenamiento, se hace necesario establecer dos funciones que permitan optimizar el problema de la gestión de recursos. La *Ecuación* 3.10 representa el consumo de la energía eléctrica por parte de la red central y la *Ecuación* 3.11 representa el consumo por parte del sistema de almacenamiento.

$$f_1 = \sum_{k=1}^{T} C_{G,k} P_{G,k} \,[\$] \rightarrow (mín) \tag{3.10}$$

$$f_2 = \sum_{k=1}^{T} C_{G,k} P_{Bk} \,[\$] \rightarrow (máx) \tag{3.11}$$

$C_{G,k}$, es el pronóstico de compra/venta de energía eléctrica para un instante k de tiempo.

$P_{G,k}$, es la potencia de la red central para un instante k de tiempo.

$P_{B,k}$, es la potencia del sistema de almacenamiento para un instante k de tiempo.

T, es el intervalo de tiempo seleccionado para la optimización.

Este problema es de naturaleza multi-objetivo lo cual conlleva a un proceso computacional de mayor complejidad, por lo cual se define la *Ecuación 3.12* que reúne las características de las *ecuaciones* 3.10 y 3.11.

$$f = \sum_{k=1}^{T} C_{G,k}(P_{G,k} - P_{B,k})\,[\$] \rightarrow (mín) \tag{3.12}$$

Sujeta a:

$$0{,}1 \leq SOC \leq 1 \tag{3.13}$$

$$P_G + P_W + P_B + P_L = 0 \tag{3.14}$$

$$0{,}9 \leq V \leq 1{,}1 \tag{3.15}$$

De acuerdo a lo descrito, en el *Anexo* A.17*,* se presenta la función *funcion_ evaluacion()* la cual permite evaluar el perfil de trabajo para el sistema de almacenamiento.

3.1.3 Datos de salida En el *Anexo B,* se presenta los resultados o datos de salida correspondientes a cada uno de los sistemas de prueba mencionados con anterioridad.

Gráficas: Para los casos de prueba, se presentan 4 gráficos de la siguiente manera: la primera gráfica presenta la potencia activa de la red central, sistema de almacenamiento, demanda y fuente de generación renovable. La segunda gráfica presenta la potencia reactiva de la red central y demanda. La tercera presenta la curva de estado de carga SOC y precio de compra/venta de la energía eléctrica. Por ultimo en la gráfica 4 se presenta el perfil de tensiones del sistema de distribución.

Resumen: Para los casos de prueba, se presentan 4 tablas de la siguiente manera: la primera tabla *características del sistema,* presenta un resumen de los parámetros de entrada.

La segunda tabla *flujo de potencia,* presenta los valores de potencia de la red central, sistema de almacenamiento y fuente de generación renovable para cada hora de la ventana de tiempo seleccionada. La tercera tabla *resumen flujo de potencia,* presenta la potencia total para cada unos de los sistemas intervenidos. Finalmente en la tabla *parámetros de rendimiento,*se presentan las perdidas de potencia, tiempo CPU y consumo energético.

3.2 Ejemplo de optimización

En esta sección se explica la implementación de las heurísticas mencionas en el *capitulo 1,* particularizando en como generar la población inicial, la evaluación de los individuos y la descendencia de estos.

3.2.1 Algoritmo genético

Población inicial y parámetros de entrada: Para generar la población es necesario tener en cuentas las siguientes consideraciones. Primero, el tamaño del cromosoma estará compuesto por 12 genes, esto equivale a tener una ventana de tiempo para el sistema de almacenamiento de 12 horas. Segundo, cada gen tendrá por defecto tres estados, representados de la siguiente manera: *descarga = 1*, *desconexión = 0* y *carga = -1* lo cual corresponde con la *Figura 3.2.* Tercero, el tamaño de la población será de 2 individuos para agilidad del proceso y se generan aleatoriamente.

$$Padre = \begin{bmatrix} 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 1 \end{bmatrix}$$

$$Madre = \begin{bmatrix} -1 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 1 \end{bmatrix}$$

Adicionalmente se

Decodificación: Tomando las *ecuaciones 3.5* a *3.9.* y estableciendo los valores de $SOC_{máx} = 1$, $SOC_{mín} = 0$, $\triangle SOC = 0{,}25$ y $P_{Base} = 40[W]$, se obtiene lo siguientes resultados.

$$\begin{matrix} Padre \\ SOC_{padre}(t) \\ P_{padre}(t) \end{matrix} = \begin{bmatrix} 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0{,}25 & 0{,}5 & 0{,}5 & 0{,}25 & 0 & 0 & 0 & 0{,}25 & 0 & 0{,}25 & 0 & 0 & 0{,}25 \\ 0 & 0 & 0 & -10 & -10 & 0 & 10 & 10 & 0 & 0 & -10 & 10 & -10 & 10 & 0 & -10 \end{bmatrix}$$

$$\begin{matrix} Madre \\ SOC_{padre}(t) \\ P_{madre}(t) \end{matrix} = \begin{bmatrix} -1 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 1 \\ 0{,}25 & 0{,}5 & 0{,}75 & 0{,}75 & 1 & 1 & 1 & 1 & 1 & 1 & 0{,}75 & 0{,}5 & 0{,}75 & 1 & 1 & 0{,}75 \\ -10 & -10 & -10 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 10 & 10 & -10 & -10 & -10 & 10 \end{bmatrix}$$

Evaluación: Para la correcta evaluación de cada individuos se debe realizar los siguientes pasos. Primero, tomar cada una de las potencia del individuo a evaluar, y resolver el flujo de potencia para cada hora del día. Segundo, guardar las potencias calculadas para P_G y P_B. Tercero, Evaluar al individuo de acuerdo a la *Ecuación 3.12* y establecer su ponderación.

Selección: Con base en los resultados de la etapa de evaluación, se selecciona a dos individuos con cierta probabilidad (TC) de acuerdo a lo descrito en la *sección 1.1.3* .

Cruce: Con base en los descrito en la *sección 1.1.4* el cruce consiste en hacer un corte en el cromosoma de los padre y al intercambiar secciones del cromosoma se dará nacimiento a dos hijos.

$$Padre = \begin{bmatrix} 0 & 0 & 0 & -1 & -1 & 0 & 1 & \leftarrow & \vdots & \rightarrow & 1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 1 \end{bmatrix}$$

$$Madre = \left[\begin{array}{cccccccccccccccc} -1 & -1 & -1 & 0 & -1 & 0 & 0 & \leftarrow & \vdots & \rightarrow & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 1 \end{array} \right]$$

$$\Downarrow$$

$$Hijo_1 = \left[\begin{array}{cccccccccccccccc} 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 1 \end{array} \right]$$

$$Hijo_2 = \left[\begin{array}{cccccccccccccccc} -1 & -1 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 1 \end{array} \right]$$

Mutación: Con base en los descrito en la *sección 1.1.5* la mutación consiste en alterar determinados genes en los hijos, cambiando el valor con cierta probabilidad (TM), tal como se muestra a continuación:

$$Hijo_1 = \left[\begin{array}{cccccccccccccccc} 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 1 \end{array} \right]$$

$$\Downarrow$$

$$Hijo_{1,mutado} = \left[\begin{array}{cccccccccccccccc} 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & -1 & -1 & 0 & 1 \end{array} \right]$$

$$Hijo_2 = \left[\begin{array}{cccccccccccccccc} -1 & -1 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 1 \end{array} \right]$$

$$\Downarrow$$

$$Hijo_{2,mutado} \left[\begin{array}{cccccccccccccccc} -1 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 1 \end{array} \right]$$

Descendencia: De acuerdo a lo descrito en la *sección 1.1.6*, la descendencia se puede realizar por el método generacional o por el método estacionario. Después de escoger el método de descendencia los proceso de selección, cruce, mutación se repite con la nueva generación una y otra vez hasta que un individuo sea lo suficientemente bueno para terminar el proceso.

3.2.2 Búsqueda aleatoria repetitiva basada en caos

Población inicial: La población inicial sigue la misma metodología del algoritmo genético, donde x_i^k, representa un individuo de la población y z_i^k es un registro de la mejor posición alcanzada por x_i hasta la iteración actual.

Ecuación de actualización: La primera acción consiste generar un individuo aleatorio u, el cual se combina en proporción a un número caótico ($\beta \in [0, 1]$) con z_i^k dando origen a dx_i. Esta combinación al igual que el algoritmo genético consiste en hacer un corte e intercambiar

secciones del individuo. Seguidamente, se calcula la posible posición del individuo (y_i^k) el cual resulta de combinardx_i con x_i^k.

Criterio de aceptación: Esta etapa consiste en decodificar x_i^k y y_i^k evaluar su posición (la decodificación y evaluación se realiza de igual manera que en el algoritmo genético). Si $f(y_i^k) > f(x_i^k)$, entonces $x_i^{k+1} = y_i^k$ y z_i^{k+1}se actualiza combinando z_i^kcon dx_i, esta combinación al igual que la anterior se realiza en proporción a un número caótico $\gamma \in [0, 1]$.

Una vez realizada las correspondientes actualizaciones el procesos se repite hasta que el mejor individuo no presente cambios significativos en un número determinado de iteraciones consecutivas.

3.2.3 Cúmulo de partículas

Población inicial: La población inicial sigue la misma metodología del algoritmo genético, donde x_i^k corresponde a la posición actual de la partícula. Adicionalmente cada partícula tendrá asociada una velocidad v_i^k.

Ecuación de actualización: La primera acción consiste en decodificar y evaluar la posición de cada partícula y determinar la mejor posición dentro del cúmulo (g_i) y la mejor posición de la partícula teniendo en cuenta todas las posiciones que ha tenido hasta llegar a la posición actual (p_{Best}). Una vez determinado estos dos parámetros se realiza un corte en ambas partículas, este corte empezará en la cola de $pBest$ con el fin de que en las primeras iteraciones la búsqueda sea *diversificada* y con el paso de las iteraciones este corte acabe en la cabeza de p_{Best} haciendo que la búsqueda sea *intensificada* (este metodología de empezar con una búsqueda diversificada y terminar con una búsqueda intensificada corresponde al modelo completo, cuando $\varphi_1, \varphi_2 > 0$). Posteriormente se calcula la nueva posición de la partícula (x_i^{k+1}) la cual consiste en combinar x_i^kcon v_i^k (este corte se hace en proporción a un número generado a partir del mapa caótico de la parábola logística). En este modelo completo no se considera criterio de aceptación pues cada partícula tiene la opción de empeorar su posición respecto a la anterior con el fin de evitar caer en óptimos locales y a su vez logrando mayor ergodicidad dentro del espacio de búsqueda.

Una vez realizada las correspondientes actualizaciones el procesos se repite hasta que la mejor partícula no presente cambios significativos en un número determinado de iteraciones consecutivas.

3.2.4 Metaheurística del murciélago

Población inicial y parámetros de entrada: La generación de la población inicial sigue la misma metodología del algoritmo genético, donde x_i^k y v_i^k representa la posición y velocidad actual de la partícula respectivamente. Adicionalmente, cada partícula tendrá asociado un pulso ($r \in [0,1]$) que al pasar de las iteraciones ira aumentando y un sonido $A \in [0,1]$ que al pasar de las iteraciones ira disminuyendo.

Ecuación de actualización: La primera acción consiste en determinar la mejor posición dentro de la colonia ($x_{óptimo}$). Posteriormente se realiza la búsqueda local (x_{local}), esta búsqueda solo se realiza si el pulso emitido por el murciélago es menor que un número generado a partir del mapa caótico de la parábola logística (γ); si este criterio se cumple la búsqueda local consistirá en tomar la posición actual del murciélago x_i^k y cambiar un valor dentro del vector (lo equivalente a la mutación en el algoritmo genético). Seguidamente se calcula la posible velocidad de la partícula (v_{aux}), la cual consiste en combinar $x_{óptimo}$, x_i^k y v_i^k. Finalmente, se calcula la posible posición del murciélago (x_{aux}) combinando x_i^kcon v_{aux} (los cortes se hacen en proporción a un número generado a partir del mapa caótico de la parábola logística).

Criterio de aceptación Esta etapa consta de dos criterios de aceptación donde la primera consiste en comparar el sonido emitido por el murciélago con un número caótico (λ) y la segunda consiste en decodificar y evaluar a x_i^k y x_{aux}. Ahora bien, si $A_i^k < \lambda$ y $f(x_{aux}) > f(x_i^k)$ se dice que el criterio de aceptación se cumple y por lo tanto la posición y velocidad del murciélago se actualizan $x_i^{k+1} = x_{aux}$ y $v_i^k = v_{aux}$, de igual forma el pulso y el sonido se actualizan a partir de las *ecuaciones 1.22 y 1.21* respectivamente.

Una vez realizada las correspondientes actualizaciones el procesos se repite hasta que la mejor partícula no presente cambios significativos en un número determinado de iteraciones consecutivas.

3.2.5 Búsqueda arborescente

Árbol inicial de búsqueda: Para esta heurística se debe construir un árbol inicial que permita acelerar la exploración y a su vez evite caer en zonas no factibles. De acuerdo con los resultado de (**?**) la curva de SOC tiene un comportamiento el cual consiste en empezar con el mínimo de carga y terminar el día con la misma cantidad de carga que empezó. Es decir, si en la primer hora del día el SOC del sistema de almacenamiento es 0.1 en la ultima hora del día debe ser 0.1. Por lo tanto, un árbol inicial de búsqueda que cumpla esta condición es aquel que durante las primeras horas del día a cargue al máximo, seguido a esto desconecta y finalmente descargue la energía almacenada en el mismo lapso de tiempo que le tomo cargar.

$$x_i = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Búsqueda local: La búsqueda local comienza evaluando el árbol inicial realizando cambios unitario hasta llegara a la última posición. Cada solución obtenida se registra en un vector f el cual servirá como referencia para la búsqueda global (para la evaluación del árbol se sigue la misma metodología empleada por el algoritmo genético).

$$
\begin{array}{cc}
x_i & \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad f(x_i) \\
x_{i,1} & \begin{bmatrix} 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad f(x_{i,1}) \\
x_{i,2} & \begin{bmatrix} 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad f(x_{i,2}) \\
\vdots & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
x_{i,(n-2)} & \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad f(x_{i,(n-2)}) \\
x_{i,(n-1)} & \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad f(x_{i,(n-1)}) \\
x_{i,n} & \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 \end{bmatrix} \quad f(x_{i,n})
\end{array}
$$

Búsqueda global: A partir de los resultados que se obtengan de la búsqueda local, se conserva el mejor cambio unitario observado respecto al árbol base (x_i). Es decir si $f(x_{i,n}) > f(x_i)$ entonces $x_i = x_{i,n}$. Tomando como ejemplo que el mejor cambio observado en el árbol es $x_{i,2}$, entonces el árbol cambia de la siguiente manera:

$$
x_i = \begin{bmatrix} 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}
$$

Posteriormente el proceso de búsqueda local/global se repite una y otra vez hasta que no se observe una mejora en la búsqueda local respecto al árbol base.

4

Caso de aplicación

Este capítulo centra el estudio en dos pruebas: la primera consiste en determinar que heurística obtiene el mejor desempeño para resolver el problema de la gestión óptima de recursos energéticos; la segunda con base a la mejor heurística de la primer prueba consistirá en tomar el modelo centralizado y distribuido para el sistema de almacenamiento y analizar los pro y contras correspondiente.

La organización de este capítulo se compone de la siguiente manera: en la *Sección 4.1* se menciona las caracterizaras del equipo de computo que se utilizó para la ejecución de las heurísticas; en la *Sección 4.2* se da la explicación del análisis de sensibilidad; en la *Sección 4.3* se presenta las características del sistema de distribución de prueba; en la *Sección 4.4* se presenta la primer prueba la cual consiste en determinar que heurística obtiene los mejores resultados en cuanto a precisión en la solución y tiempo de simulación y finalmente en la *Sección 4.5* se realiza una segunda prueba donde se plantea dos modelos uno centralizado y otro distribuido para el sistema de almacenamiento con el fin de establecer que modelo es el mas adecuado.

4.1 Entorno computacional

Las pruebas de los algoritmos fueron ejecutados en un computador con procesador AMD A8-3500M APU a 1.50 GHz con 4 GB de RAM. El software se compiló para el sistema operativo Windows 8.1 y el código fue desarrollado en el lenguaje de programación Python.

4.2 Análisis de sensibilidad

El análisis de sensibilidad consiste en buscar empíricamente los criterios de optimización, es decir, los parámetros de entrada de los algoritmos que generen soluciones óptimas. Este proceso se realizo en la *Sección 1.1* por medio de las funciones de prueba utilizadas.

En los distintos autores revisados en este trabajo, se halló que la forma en que se seleccionaron los parámetros es por prueba y error; ya que en la literatura no hay una forma exacta para determinar las soluciones óptimas en la implementación de los algoritmos. En cuanto al tamaño de la población, se definió arbitrariamente un tamaño de 100 individuos ya que es un valor típico usado en la literatura (**?**).

4.3 Sistema de prueba

El sistema de distribución que se utilizará para la realizar las siguientes pruebas corresponde al sistema de distribución de 7 barras empelado en el *capitulo* 2. Adicionalmente, se hicieron las mismas pruebas para 2 sistemas radiales (ver *Anexo B.1* y B.2) y 1 sistema mallado (ver *Anexo B.3*).

4.4 Prueba 1

Para la realización de la prueba se tendrá en cuenta la siguiente metodología:

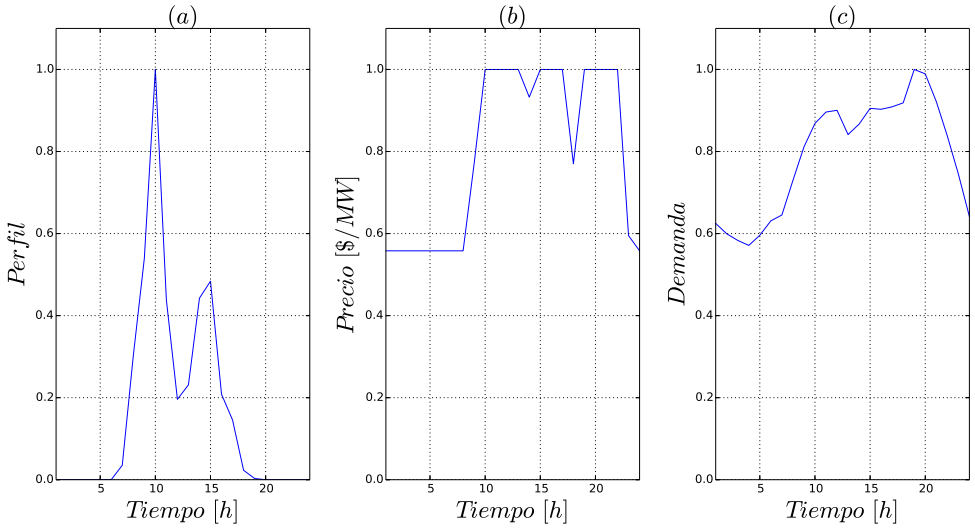
1. Escoger un día en la base de datos para cargar la curva de demanda, radiación solar y precio de compra/venta de la energía eléctrica..
2. Escoger el punto de conexión para el sistema de almacenamiento y fuente de generación renovable.
3. Escoger la potencia del sistema de almacenamiento y de la fuente de generación renovable.
4. Correr el flujo de potencia NR_CPLX utilizando una de las heurísticas estudiadas en el *Capítulo 1*, y registrar el consumo energético, pérdidas y tiempo de simulación.
5. Repetir 20 veces los paso 6 y 7 para cada una de las heurísticas estudiadas.

6. Calcular promedio y desviación estándar para el consumo y tiempo de simulación.

7. Repetir los pasos 5, 6 y 7 utilizando el flujo de radial TRX.

De acuerdo a lo descrito en la metodología, para el paso 1 se selecciona el día 0 de la base datos. En la *Figura* 4.1, se presenta las curvas correspondientes al día seleccionado.

Figura 4.1.: Base de datos correspondiente al día cero.



a, corresponde al perfil de energía fotovoltaico (adimensional).

b, corresponde a la curva de precio de compra/venta de la energía eléctrica.

c, corresponde a la curva de demanda para sistema de distribución (adimensional).

Para el paso 2 se corre el flujo de potencia NR-C para la demanda seleccionada y sin considerar la fuente de generación renovable. Posteriormente, se registra el consumo energético y perdidas del sistema de distribución para una ventana de tiempo de 24 horas.

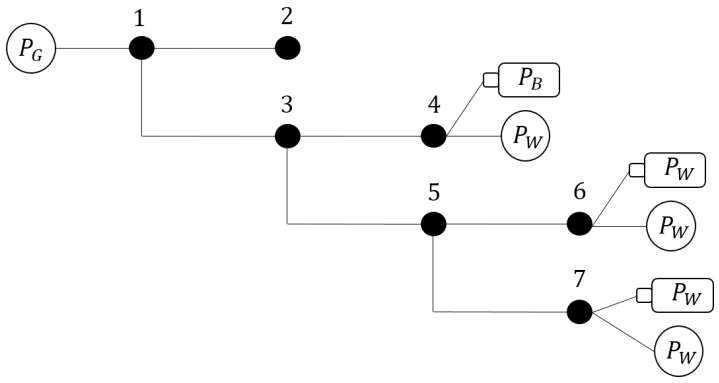
$$C_{base} = 7,7793\,[\$] \tag{4.1}$$

$$P_{perdidas} = 0{,}3849\,[MW] \tag{4.2}$$

En el paso 3, se eligen las barras 4, 6 y 7 para la conexión de la fuente de generación renovable ya que como vemos en la *Figura* 4.2 estas corresponden a las colas del sistema de distribución y por lo tanto tendrán la mayor caída de tensión. Esta conexión corresponde a un modelo distribuido por lo cual al inyectar potencia en estas barras mejorará el perfil de

tensiones y a su vez permitirá reducir las perdidas de potencia en el sistema de distribución (**?**). De igual manera se elige estas barras para la conexión del sistema de almacenamiento.

Figura 4.2.: Modelo con almacenamiento distribuido, para el sistema de distribución de 7 barras.



Para el paso 4, se elige una capacidad máxima de almacenamiento equivalente a 1 [MW] (ver *Ecuación* 4.3), con potencia de carga/descarga equivalente a 1/6 de la capacidad máxima de almacenamiento y para la fuente de generación renovable se elige un valor de K igual a 0.3 [MW] (ver *Ecuación y* 4.4). Estos valores se tomaron empíricamente considerando que la inyección de potencia de estos dos elementos no genere un *Black Out* al momento de resolver el flujo de potencia.

$$P_{base} = 1\,[MW] \tag{4.3}$$

$$K = 0{,}3\,[MW] \tag{4.4}$$

Finalmente, para el paso los 5, 6, 7 y 8 en las *Tablas* 4.1, 4.2, 4.3, 4.4 *y* 4.5 se presentan los resultados de las 20 ejecuciones por cada heurística y para cada flujo de potencia. La información de las tablas se da de la siguiente manera: la columna 1 corresponde al número del registro de la prueba, la columna 2 corresponde al consumo energético, la columna 3 corresponde al tiempo del CPU y la columna 4 corresponde a las perdidas de potencia.

TABLA 4.1.: Registro de 20 ejecuciones utilizando la heurística algoritmo genético.

Registro	***N.R con sistema P.U. complejo***			***Flujo radial TRX***		
	C [$]	t_{CPU} ***[s]***	P_{per} ***[MW]***	***C [$]***	t_{CPU} ***[s]***	P_{per} ***[MW]***
1	6,1390	186,253	0,3469	6,138	51,089	0,3455
2	6,1390	273,508	0,3469	6,1625	40,350	0,3466
3	6,1390	241,632	0,3469	6,1381	66,522	0,3455
4	6,1632	179,613	0,3496	6,1414	53,824	0,3489
5	6,1393	205,776	0,3474	6,1381	60,725	0,3455
6	6,1390	256,987	0,3469	6,1461	39,614	0,3536
7	6,1632	327,353	0,3496	6,1381	81,245	0,3455
8	6,1424	323,356	0,3503	6,1414	55,956	0,3489
9	6,1390	352,041	0,3469	6,1381	50,403	0,3455
10	6,1471	225,608	0,355	6,1381	41,287	0,3455
11	6,1390	318,373	0,3469	6,1697	42,787	0,3538
12	6,1390	224,255	0,3469	6,1381	51,018	0,3455
13	6,1390	262,655	0,3469	6,1414	61,929	0,3489
14	6,1414	189,506	0,3494	6,1381	50,787	0,3455
15	6,1390	277,317	0,3469	6,1381	53,759	0,3455
16	6,1625	270,095	0,3471	6,1464	39,643	0,3542
17	6,1471	184,712	0,355	6,1406	42,893	0,348
18	6,1540	177,708	0,3621	6,1381	45,621	0,3455
19	6,1390	295,714	0,3469	6,1414	39,750	0,3489
20	6,1390	228,395	0,3469	6,1381	53,896	0,3455

TABLA 4.2.: Registro de 20 ejecuciones utilizando la heurística búsqueda aleatoria repetitiva.

Registro	***N.R con sistema P.U. complejo***			***Flujo radial TRX***		
	C [$]	t_{CPU} ***[s]***	P_{per} ***[MW]***	***C [$]***	t_{CPU} ***[s]***	P_{per} ***[MW]***
1	6,2148	265,291	0,3883	6,1426	111,707	0,3455
2	6,1457	363,765	0,3487	6,1426	70,304	0,3455
3	6,2046	240,685	0,3938	6,1560	47,877	0,3722
4	6,1711	456,836	0,3736	6,1477	66,451	0,3663
5	6,1390	480,678	0,3539	6,1621	85,075	0,3526
6	6,1557	467,191	0,3577	6,2002	65,377	0,3803
7	6,2420	218,818	0,3933	6,1935	101,288	0,384
8	6,1760	567,39	0,3469	6,1710	171,378	0,3871
9	6,2257	318,939	0,3788	6,1942	61,169	0,3765
10	6,1568	296,253	0,3632	6,2130	43,512	0,3489
11	6,2177	247,963	0,3945	6,1554	115,338	0,3616
12	6,2314	313,788	0,3837	6,1599	186,006	0,3551
13	6,2679	277,212	0,3873	6,1448	133,475	0,3518
14	6,3205	262,583	0,3944	6,1836	97,602	0,3739
15	6,2383	362,583	0,4388	6,7111	145,075	0,3665
16	6,1548	257,792	0,3676	6,1381	69,404	0,3567
17	6,1804	317,236	0,372	6,2656	47,954	0,3659
18	6,1804	278,359	0,3736	6,1773	103,577	0,3519
19	6,3717	213,788	0,3886	6,1946	75,365	0,3771
20	6,2116	301,712	0,3852	6,2059	65,066	0,3891

TABLA 4.3.: Registro de 20 ejecuciones utilizando la heurística cúmulo de partículas.

Registro	***N.R con sistema P.U. complejo***			***Flujo radial TRX***		
	C [$]	t_{CPU} ***[s]***	P_{per} ***[MW]***	***C [$]***	t_{CPU} ***[s]***	P_{per} ***[MW]***
1	6,1390	267,189	0,3566	6,1414	45,600	0,3489
2	6,1390	384,404	0,3566	6,1414	53,447	0,3489
3	6,1424	247,737	0,36	6,1381	53,440	0,3455
4	6,1390	228,637	0,3566	6,1381	62,047	0,3455
5	6,2120	299,1	0,3469	6,1381	67,98	0,3455
6	6,1471	270,44	0,3647	6,1461	55,448	0,3536
7	6,1390	269,774	0,3566	6,1414	45,330	0,3489
8	6,1390	240,574	0,3566	6,1381	52,599	0,3455
9	6,1390	198,671	0,3566	6,1461	32,562	0,3536
10	6,1390	148,121	0,3566	6,1461	42,849	0,3536
11	6,1471	210,876	0,3647	6,1381	61,384	0,3455
12	6,1424	179,614	0,36	6,1616	48,763	0,3457
13	6,1471	141,046	0,3647	6,1381	40,723	0,3455
14	6,1390	163,91	0,3566	6,1381	50,257	0,3455
15	6,1390	244,334	0,3566	6,1461	48,169	0,3536
16	6,1390	185,922	0,3566	6,1381	36,867	0,3455
17	6,1390	303,831	0,3566	6,1414	33,703	0,3489
18	6,1390	155,353	0,3566	6,1461	47,502	0,3536
19	6,1424	193,988	0,36	6,1381	51,500	0,3455
20	6,1390	255,352	0,3566	6,1590	27,993	0,3665

TABLA 4.4.: Registro de 20 ejecuciones utilizando la metaheurística del murciélago.

Registro	***N.R con sistema P.U. complejo***			***Flujo radial TRX***		
	C [$]	t_{CPU} ***[s]***	P_{per} ***[MW]***	***C [$]***	t_{CPU} ***[s]***	P_{per} ***[MW]***
1	6,1625	185,141	0,3649	6,1414	52,036	0,3664
2	6,1390	212,532	0,3647	6,1414	49,602	0,3664
3	6,1424	208,798	0,3681	6,1414	49,042	0,3664
4	6,1424	162,408	0,3681	6,1461	45,615	0,3711
5	6,1625	216,694	0,3649	6,1461	56,849	0,3664
6	6,1625	195,504	0,3649	6,1414	46,937	0,3664
7	6,1390	178,271	0,3647	6,1414	50,876	0,3664
8	6,1424	163,527	0,3681	6,2031	44,981	0,3455
9	6,1390	268,882	0,3647	6,1381	48,224	0,363
10	6,1552	131,184	0,3809	6,1414	51,327	0,3664
11	6,2038	145,504	0,3469	6,5933	48,769	0,363
12	6,1390	295,453	0,3647	6,1381	47,947	0,363
13	6,1390	200,927	0,3647	6,1414	42,501	0,3664
14	6,1634	172,75	0,3658	6,1461	39,083	0,3711
15	6,1471	193,015	0,3728	6,1381	55,097	0,363
16	6,1390	366,348	0,3647	6,1616	57,495	0,3632
17	6,1625	236,392	0,3649	6,1616	35,929	0,3628
18	6,1625	355,01	0,3649	6,1461	46,87	0,3711
19	6,1545	204,634	0,3779	6,1381	61,602	0,363
20	6,1424	235,858	0,3681	6,1381	57,783	0,363

TABLA 4.5.: Registro de 20 ejecuciones utilizando la heurística búsqueda arborescente.

Registro	***N.R con sistema P.U. complejo***			***Flujo radial TRX***		
	C [$]	t_{CPU} ***[s]***	P_{per} ***[MW]***	***C [$]***	t_{CPU} ***[s]***	P_{per} ***[MW]***
1	6,1390	30,440	0,3469	6,1381	8,596	0,3455
2	6,1390	30,131	0,3469	6,1381	8,200	0,3455
3	6,1390	30,310	0,3469	6,1381	8,808	0,3455
4	6,1390	30,990	0,3469	6,1381	8,084	0,3455
5	6,1390	31,680	0,3469	6,1381	8,304	0,3455
6	6,1390	31,400	0,3469	6,1381	8,434	0,3455
7	6,1390	31,620	0,3469	6,1381	8,269	0,3455
8	6,1390	33,590	0,3469	6,1381	8,156	0,3455
9	6,1390	34,680	0,3469	6,1381	8,200	0,3455
10	6,1390	34,390	0,3469	6,1381	8,273	0,3455
11	6,1390	34,200	0,3469	6,1381	8,164	0,3455
12	6,1390	31,410	0,3469	6,1381	8,216	0,3455
13	6,1390	31,450	0,3469	6,1381	8,225	0,3455
14	6,1390	30,620	0,3469	6,1381	8,300	0,3455
15	6,1390	30,070	0,3469	6,1381	8,193	0,3455
16	6,1390	29,980	0,3469	6,1381	8,127	0,3455
17	6,1390	29,680	0,3469	6,1381	8,169	0,3455
18	6,1390	29,670	0,3469	6,1381	8,068	0,3455
19	6,1390	32,210	0,3469	6,1381	8,207	0,3455
20	6,1390	34,750	0,3469	6,1381	8,108	0,3455

4.4.1 Análisis de resultados

En la *Tabla* 4.6 se presenta el resumen del estudio estadístico obtenidos para cada una de las heurísticas utilizando el método de Newton Raphson con sistema P.U. complejo. La información de la tabla se da de la siguiente manera: la columna 1 indica la heurística avaluada; la columna 2, 3 y 4 muestra el promedio del consumo, tiempo CPU y perdidas de potencia; la columna 5, 6 y 7 muestra la desviación estándar del consumo, tiempo CPU y perdidas de potencia.

TABLA 4.6.: Promedios y desviación estándar de 20 ejecuciones por cada heurísticas evaluada utilizando el método de Newton Rapshon con sistema P.U. complejo.

Heurísticas	***N.R con sistema P.U. complejo***					
	Promedio			***Desviación estándar***		
	C [$]	$t_{CPU}[s]$	$P_R[MW]$	C [$]	$t_{CPU}[s]$	$P_R[MW]$
Algoritmo genético	6,1445	250,0429	0,3491	0,0088	54,4685	0,0039
Búsqueda aleatoria repetitiva	6,2105	325,4431	0,3792	0,0593	96,8487	0,0209
Cúmulo de partículas	6,1444	229,4437	0,3578	0,0161	61,3959	0,0039
Metaheurística del murciélago	6,1520	216,4416	0,3665	0,0158	62,9319	0,0065
Búsqueda arborescente	6.1390	31,6636	0,3469	0	1.734	0

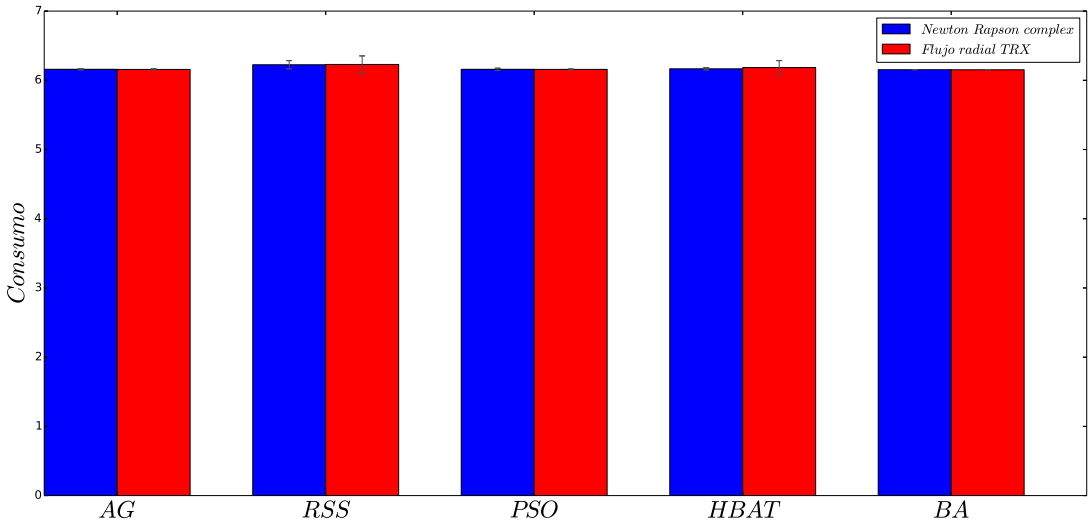
En la *Tabla* 4.7 se presenta el resumen del estudio estadístico obtenidos para cada una de las heurísticas utilizando el método de flujo radial TRX. La información de la tabla se da de la siguiente manera: la columna 1 indica la heurística avaluada; la columna 2, 3 y 4 muestra el promedio del consumo, tiempo CPU y perdidas de potencia; la columna 5, 6 y 7 muestra la desviación estándar del consumo, tiempo CPU y perdidas de potencia.

TABLA 4.7.: Promedios y desviación estándar de 20 ejecuciones por cada heurísticas evaluada utilizando flujo radial TRX

Heurísticas	***Flujo radial TRX***					
	Promedio			***Desviación estándar***		
	C [$]	$t_{CPU}[s]$	$P_R[MW]$	C [$]	$t_{CPU}[s]$	$P_R[MW]$
Algoritmo genético	6,1425	51,1549	0,3476	0,0085	10,6949	0,0030
Búsqueda aleatoria repetitiva	6,2030	93,1500	0,3654	0,1236	40,6621	0,01409
Cúmulo de partículas	6,1430	47,9082	0,3493	0,0067	10,2312	0,0052
Metaheurística del murciélago	6,11692	49,4283	0,3647	0,1009	6,3715	0,0053
Búsqueda arborescente	6.1381	8,2551	0,3455	0	0,1779	0

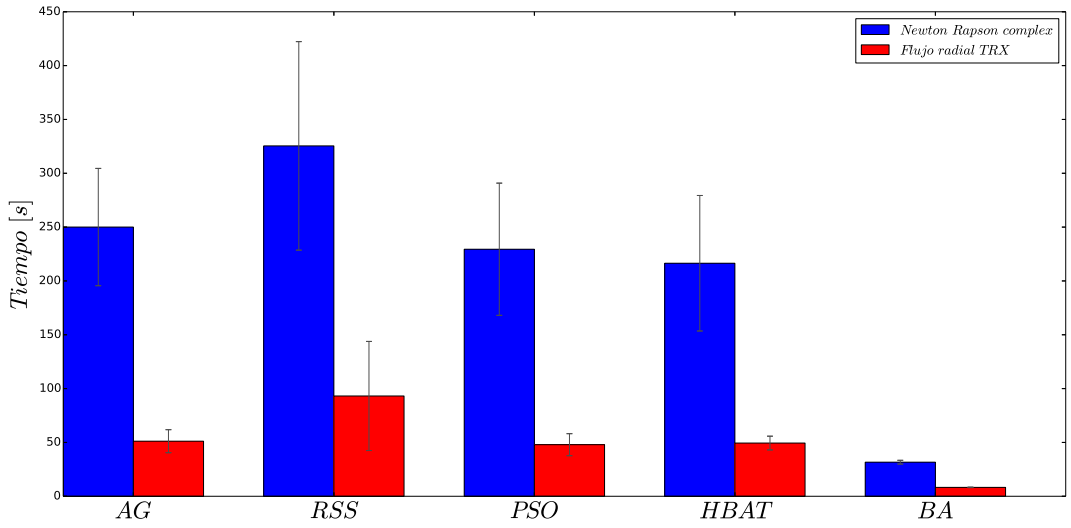
En la *Figura 4.3* se presenta el comparativo de los resultados obtenidos en las *Tablas* 4.6 y 4.7 para el consumo del sistema de distribución de 7 barras. Ahora bien, cabe resaltar que todas las heurísticas llegan a resultados similar con desviaciones estándar muy bajas y que la diferencia del peor resultado (búsqueda aleatoria repetitiva) respecto del mejor resultado (búsqueda arborescente) es del 1,1646 % utilizando NR-C, lo cual indica una alto grado de confiabilidad en los resultados. De igual modo, al realizar el análisis para el flujo radial TRX la diferencia del peor resultado (búsqueda aleatoria repetitiva) respecto del mejor resultado (búsqueda arborescente) es del 1.0571 %. Ahora bien, si se compara el mejor resultado obtenido con NR-C vs el mejor resultado obtenido con flujo radial TRX se tiene un margen de error de 0,0146 %.

Figura 4.3.: Comparación del consumo energético, utilizando el método de Newton Rapshon son sistema P.U. complejo vs Flujo Radial TRX.



En la *Figura* 4.4 se presenta los resultados para del tiempo CPU, del cual se observa una clara diferencia entre los métodos heuristicos y sobre todo en los métodos para la solución del flujo de potencia. Partiendo de los tiempos de solución para las distintas heurísticas empleando NR-C se comprueba que la mejor heurística corresponde a búsqueda arborescente con un tiempo CPU de 31 segundos en promedio y el peor resultado corresponde a búsqueda aleatoria repetitiva con un tiempo CPU de 325 segundos. Igualmente, al emplear el flujo radial TRX se comprueba que búsqueda arborescente tiene el menor tiempo CPU con 8 segundos en promedio y el peor caso con un tiempo CPU de 93 segundos en promedio para búsqueda aleatoria repetitiva.

Figura 4.4.: Comparación del tiempo CPU, utilizando el método de Newton Rapshon cplx vs flujo radial TRX



De acuerdo a estos resultados se establece que la mejor opción para la gestión de recursos energéticos, se obtiene al aplicar búsqueda arborescente y la razón de este es por la adecuada estrategia de búsqueda, pues al partir de un árbol inicial se evita la exploración de zonas no factibles. Otro aspecto a resaltar para búsqueda arborescente se debe a que no necesita de una población inicial pues solo basta con un solo perfil para explorar en forma eficiente el espacio de búsqueda. Por lo cual, el poco tiempo que la toma a esta heurística encontrar la solución es debido a los pocos cálculos que se debe hacer en comparación con sus contendientes evaluadas que necesitan de una población inicial para tratar de cubrir todo el espacio de búsqueda.

En cuanto a los métodos de solución del flujo de potencia, se observa que flujo radial TRX es más rápido que NR-C ya que este utiliza una matriz TRX la cual es constante durante todo el proceso, contrario a NR-C que debe calcular la matriz jacobiana en cada iteración por lo cual este método requiere de un mayor tiempo computacional. Una comparación mas exhaustiva se puede encontrar en (?)

Por otra parte y partiendo de los resultados obtenidos en la *Tabla* 4.6, se determinó que el consumo energético para el sistema de distribución de 7 barras y bajo las condiciones dadas es de 6.1390 [$]. Esta solución indica un ahorro de 1.6403 [$], es decir una reducción de la dependencia energética de la red central en un 21,0864 % y ademas se redujeron las perdidas de potencia en un 9,8726 %.

A continuación se presenta un resumen de la simulación para el caso estudiado.

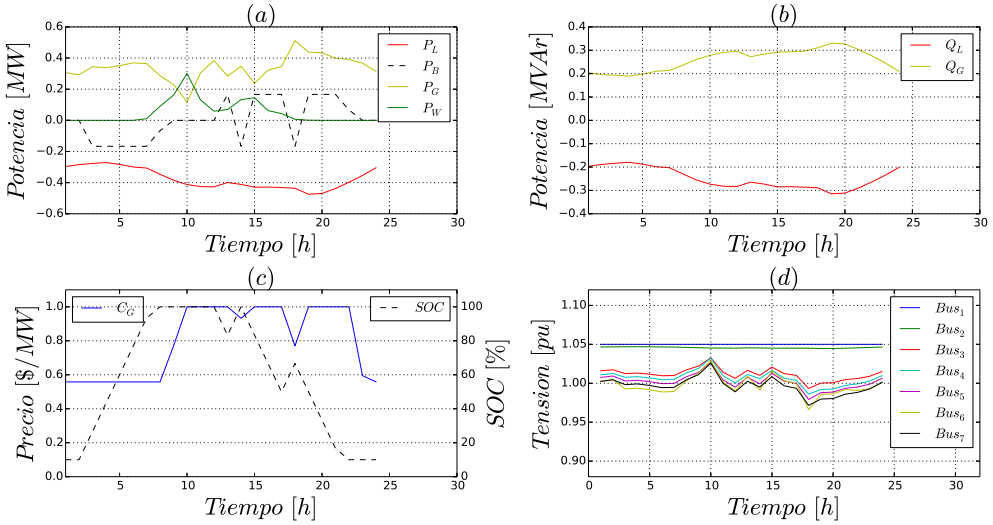
```
1  ================================================
2           TABLA A: CARACTERISTICAS DEL SISTEMA
3  ================================================
4  SISTEMA DE DISTRIBUCION
5  Caso de prueba              case_7
6  Numero de barras            7
7  Numero de lineas            6
8  Numero de generadores       1
9  Potencia base               1 [MVA]
10 Tension base                12.47 [kV]
11 ------------------------------------------------
12 SISTEMA DE ALMACENAMIENTO
13 Ubicacion                   [ 4, 6, 7 ]
14 Capacidad maxima            [1] [MW]
15 Potencia                    0.1667 [MW]
16 ------------------------------------------------
17 FUENTE RENOVABLE
18 Ubicacion                   [ 4.  6.  7.]
19 Potencia                    [ 0.1  0.1  0.1] [MW]
20 ================================================
21
22 ================================================================================================================
23                                TABLA B: FLUJO DE POTENCIA, CONSUMO ENERGETICO
24 ----------------------------------------------------------------------------------------------------------------
25    Hora   P_g [MW]  Q_g [MVAr] P_d [MW] Q_d [MVAr] P_B [MW]   SOC %    P_W [MW]  C_G [$/MW]
   CONSUMO [$]
26 ================================================================================================================
27     0       0.306     0.2041    0.2965    0.1966     0.0       0.1       0.0       0.5579
   0.1707
28     1       0.2932    0.1955    0.2845    0.1886     0.0       0.1       0.0       0.5579
   0.3343
29     2       0.4655    0.2009    0.2769    0.1835   -0.1667    0.2667     0.0       0.5579
   0.594
30     3       0.4591    0.1965    0.271     0.1797   -0.1667    0.4333     0.0       0.5579
   0.8501
31     4       0.4721    0.2053    0.2828    0.1875   -0.1667     0.6       0.0       0.5579
   1.1135
32     5       0.4906    0.2178    0.2995    0.1985   -0.1667    0.7667     0.0       0.5579
   1.3872
33     6       0.4862    0.2218    0.3062    0.203    -0.1667    0.9333    0.0106     0.5579
   1.6585
34     7       0.3338    0.2386    0.3462    0.2295   -0.0667     1.0      0.0905     0.5579
   1.8447
35     8       0.2304    0.261     0.3849    0.2552     0.0       1.0      0.1617     0.7702
   2.0221
36     9       0.1183    0.2782    0.4122    0.2732     0.0       1.0       0.3        1.0
   2.1405
37     10      0.3048    0.2907    0.4253    0.2819     0.0       1.0      0.1314      1.0
   2.4453
38     11      0.3841    0.2957    0.4273    0.2833     0.0       1.0      0.0588      1.0
   2.8294
39     12      0.1694    0.2697    0.3993    0.2647    0.1667    0.8333    0.0694      1.0
   2.9988
40     13      0.4673    0.2902    0.4114    0.2727   -0.1667     1.0      0.1328     0.9325
```

```
    3.4346
41      14      0.1244    0.2902    0.4296    0.2848    0.1667    0.8333    0.1452     1.0
    3.559
42      15      0.207     0.2902    0.4285    0.2841    0.1667    0.6667    0.0625     1.0
    3.766
43      16      0.2293    0.2926    0.4314    0.286     0.1667     0.5      0.0436     1.0
    3.9952
44      17      0.637     0.3216    0.436     0.289    -0.1667    0.6667    0.0069    0.7702
    4.4858
45      18      0.3196    0.3248    0.4746    0.3146    0.1667     0.5      0.0009     1.0
    4.8055
46      19      0.3151    0.3212    0.4695    0.3112    0.1667    0.3333     0.0       1.0
    5.1206
47      20      0.2809    0.2981    0.4374     0.29     0.1667    0.1667     0.0       1.0
    5.4015
48      21      0.3435    0.2736    0.3976    0.2636    0.0667     0.1       0.0       1.0
    5.745
49      22      0.3674    0.2453    0.3537    0.2344     0.0       0.1       0.0      0.595
    5.9636
50      23      0.3145    0.2098    0.3045    0.2019     0.0       0.1       0.0      0.5579
    6.139
51  ================================================================================================
52
53  ============================================================================
54                        TABLA C: RESUMEN FLUJO DE POTENCIA
55  ----------------------------------------------------------------------------
56                         P_g [MW]      Q_g [MVAr]      P_d [MW]       Q_d [MAr]
57  ============================================================================
58  F. renovable             1.2142            0              0              0
59  Red central              8.1195         6.2337            0              0
60  S. almacenamiento        1.2333            0           1.2333            0
61  S. distribucion             0              0           8.9869         5.9572
62  ----------------------------------------------------------------------------
63                           10.567         6.2337         10.2202        5.9572
64  ============================================================================
65
66  ======================================================
67           TABLA D: PARAMETROS DE RENDIEMIENTO
68  ======================================================
69  Flujo de potencia      Newton Raphson (bases complejas)
70  Heuristica             Busqueda Arborescente
71  Ventana de tiempo      24 [horas]
72  Perdidas de potencia   0.3469 [MW]
73  Tiempo CPU             65.832 [s]
74  Consumo                6.139 [$]
75  Perdidas               0.2607 [$]
76  ======================================================
```

Finalmente, en la *Figura* **4.5**.c muestra la curva *SOC* que debe tener el sistema de almacenamiento para obtener el máximo ahorro energético; esta curva (color negra punteada) nos indica que a partir de la hora 3 hasta la hora 8 del día el sistema de almacenamiento debe cargar ya que el precio de la energía eléctrica C_G (curva de color azul continua) es bajo en

comparación con los precios del resto del día. Seguidamente el sistema de almacenamiento se desconecta del sistema por 4 horas, pues este anticipó el alza del precio. Seguidamente se observa que en la hora 13 el sistema descarga parte de su energía anticipando el bajo precio de la energía para cargar nuevamente en la hora 14 y posteriormente descargar desde la hora 15 a la hora 17, donde el precio de la energía vuelve a incrementar. Posteriormente en la hora 18 el precio de la energía baja y el sistema de almacenamiento decide cargar nuevamente para eventualmente descargar toda su energía en las horas restantes del día.

Figura 4.5.: Resumen de resultados para el sistema de distribución de 7 barras.



a, muestra la potencia activa generada por el sistema de almacenamiento (P_B), red central (P_G), fuente de generación renovable (P_W) y la demanda por el sistema de distribución (P_L).

b, muestra la potencia reactiva generada por la red central (Q_G) y la consumida por el sistema de distribución (Q_L).

c, muestra la curva SOC del sistema de almacenamiento y la curva de compra/venta de energía eléctrica (C_G).

d, muestra la tensión de cada una de las barras del sistema de distribución.

4.5 Prueba 2

La segunda prueba consiste en implementar un esquema en el que el almacenamiento estará concentrado en la cabecera del circuito (modelo centralizado) y comparar el desempeño con respecto al caso en el cual el almacenamiento se encuentra conectado en los mismos nodos

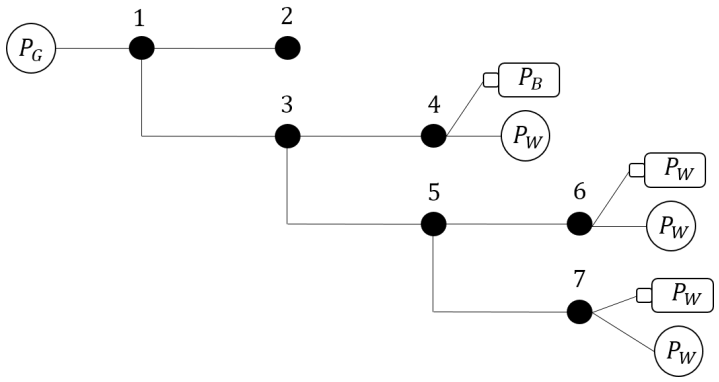
donde se encuentra la generación renovable (modelo distribuido). Este sistema de distribución al igual que en al prueba 1 tendrá como datos de entrada: curva de demanda, precio de compra/venta de la energía eléctrica y perfil de energía fotovoltaica de la base de datos del día cero. Con base a esta información se prosigue a evaluar la heurística búsqueda arborescente para determinar cual de estos dos modelos es el mas adecuado a implementar.

Para la realización de la prueba se empezará con un 50 % de la capacidad máxima de almacenamiento de referencia elegida, seguidamente se registra el consumo energético y perdidas para potencias equivalentes a 1/2, 1/4, 1/6, 1/8, 1/10, 1/12 de la capacidad de almacenamiento calculada. Posteriormente, este proceso se repite para el 75 %, 100 %, 125 %, 150 % , 175 % y 200 % de la capacidad máxima de almacenamiento de referencia.

4.5.1 Modelo con almacenamiento distribuido

En la *Figura 4.6*, se presenta el sistema de distribución de 7 barras implementando el modelo con almacenamiento distribuido y capacidad máxima de almacenamiento equivalente a 0.33 [MW] conectado en las barras número 4, 6 y 7 respectivamente . Adicionalmente, se implementó tres fuentes de generación renovable en las barras número 4, 6 y 7 con máxima potencia de 0.1 [MW] respectivamente.

Figura 4.6.: Modelo con almacenamiento distribuido, en el sistema de distribución de 7 barras.



Resultados: Los resultado de la *Tabla 4.8* se calcularon con la heurística búsqueda arborescente ya que esta obtuvo mejores resultados en comparación con las demás heurísticas evaluadas. La información en esta tabla se presenta de la siguiente manera: la columna uno indica la capacidad máxima de almacenamiento, la columna dos indica la potencia de carga/descarga, la columna tres indica el consumo energético producido por las perdidas del sistema de distribución y la columna cuatro indica el consumo energético total del sistema de distribución.

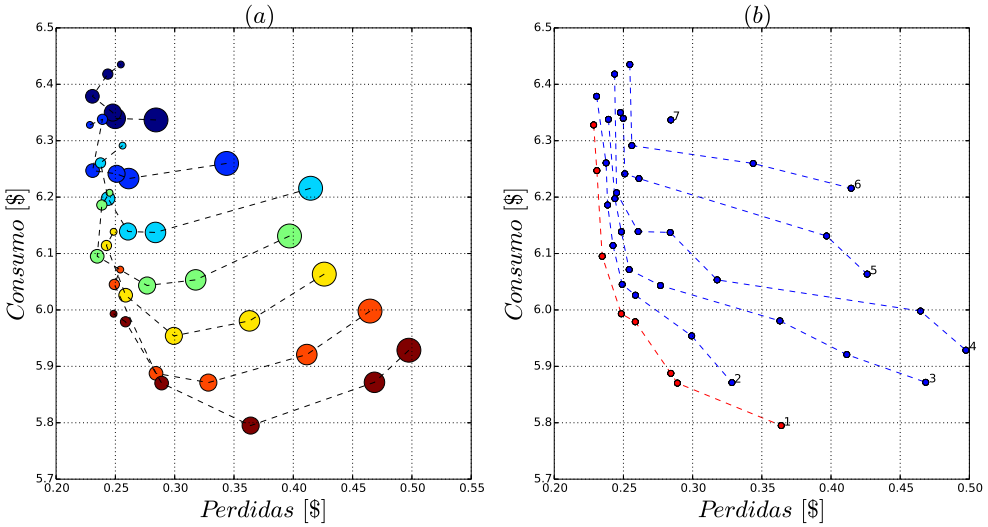
TABLA 4.8.: Consumo energético considerando un modelo con almacenamiento distribuido.

CAPACIDAD [MW]	***POTENCIA [pu]***	***PERDIDAS [$]***	***CONSUMO [$]***
0,5	1/2	0,2842	6,3368
	1/4	0,2498	6,3396
	1/6	0,2478	6,3499
	1/8	0,2306	6,3787
	1/10	0,2436	6,4181
	1/12	0,2545	6,4351
0,75	1/2	0,3437	6,2597
	1/4	0,2612	6,2329
	1/6	0,2509	6,2413
	1/8	0,2307	6,2470
	1/10	0,2391	6,3379
	1/12	0,2284	6,3280
1	1/2	0,4145	6,2157
	1/4	0,2838	6,1374
	1/6	0,2606	6,1390
	1/8	0,2438	6,1974
	1/10	0,2374	6,2606
	1/12	0,2560	6,2913
1,25	1/2	0,3968	6,1311
	1/4	0,3177	6,0533
	1/6	0,2767	6,0433
	1/8	0,2346	6,0951
	1/10	0,2385	6,1860
	1/12	0,2450	6,2077
1,5	1/2	0,4261	6,0635
	1/4	0,3630	5,9806
	1/6	0,2993	5,9540
	1/8	0,2586	6,0261
	1/10	0,2424	6,1142
	1/12	0,2485	6,1385
1,75	1/2	0,4646	5,9980
	1/4	0,4113	5,9209
	1/6	0,3283	5,8712
	1/8	0,2842	5,8876
	1/10	0,2491	6,0452
	1/12	0,2541	6,0715
2	1/2	0,4974	5,9286
	1/4	0,4684	5,8717
	1/6	0,3640	5,7950
	1/8	0,2889	5,8703
	1/10	0,2585	5,9790
	1/12	0,2484	5,9932

Análisis de resultados: Con base en los resultados de la *Tabla* 4.8, en la *Figura* 4.7.a, se presenta el consumo energético del sistema de distribución frente al consumo energético producido por las perdidas de potencia. Por otro lado, se aprecia que para cada capacidad se tiene un comportamiento en «v». Es decir, si partimos de un consumo energético base y se va aumentando la potencia para cada capacidad de almacenamiento dada, se observa que el consumo energético disminuye debido a los aportes de potencia del sistema de almacenamiento, pero a partir de una potencia dada, el consumo energético aumenta ya que las perdidas de potencia debido al transporte son más significativas que el aporte que el sistema de almacenamiento puede dar.

Seguidamente, en la 4.7.b, se presenta la información en frentes de pareto. Para este caso se han obtenido 7 diferentes frentes siendo el primero de mejor calidad y el séptimo de peor calidad. Ahora bien, el frente número 1 resalto en rojo, indica un conjunto de buenas soluciones donde la mayoría corresponden a los casos donde se tiene alta capacidad de almacenamiento y una baja potencia.

Figura 4.7.: Consumo energético considerando el modelo con almacenamiento distribuido.



a, muestra el conjunto de soluciones agrupados por color haciendo referencia a la capacidad de almacenamiento y por área de las circunferencias haciendo referencia a la potencia.

b, muestra el conjunto de soluciones agrupados por frentes de pareto.

Finalmente y de acuerdo a los resultado de las *Tabla* 4.8 y 4.10, se presenta el conjunto de soluciones correspondientes al primer frente de pareto.

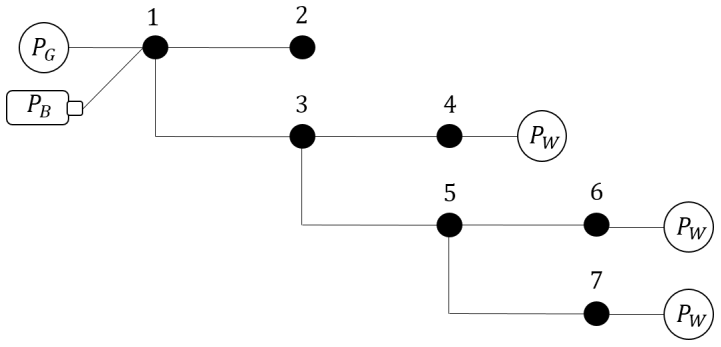
TABLA 4.9.: Conjunto de soluciones considerando el modelo con almacenamiento distribuido.

CAPACIDAD [MW]	*POTENCIA [pu]*	*PERDIDAS [$]*	*CONSUMO [$]*
0,75	1/12	0,2284	6,3280
0,75	1/8	0,2307	6,2470
1,25	1/8	0,2346	6,0951
2	1/12	0,2484	5,9932
2	1/10	0,2585	5,9790
1,75	1/8	0,2842	5,8876
2	1/8	0,2889	5,8703
2	1/6	0,3640	5,7950

4.5.2 Modelo centralizado

En la *Figura 4.8* se muestra un modelo centralizado para el sistema de almacenamiento con capacidad máxima de potencia de 1 [MW] conectado en la barra número 1. Adicionalmente se implementó tres fuentes de generación renovable en las barra número 4, 6 y 7 con potencia máxima de a 0.1 [MW] respectivamente.

Figura 4.8.: Modelo con almacenamiento centralizado, en el sistema de distribución de 7 barras.



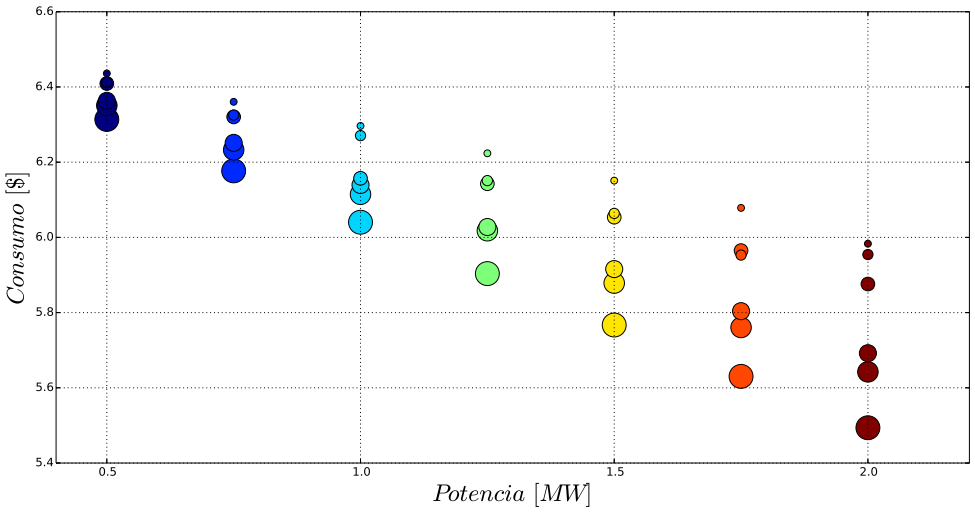
Resultados: Los resultado de la *Tabla 4.10* se calcularon con la heurística búsqueda arborescente ya que esta obtuvo mejores resultados en comparación con las demás heurísticas evaluadas. La información en esta tabla se presenta de la siguiente manera: la columna uno indica la capacidad máxima de potencia del sistema de almacenamiento, la columna dos indica la potencia de carga/descarga del sistema de almacenamiento, la columna tres indica el consumo por perdidas del sistema de distribución y la columna cuatro indica el consumo energético del sistema de distribución.

TABLA 4.10.: Consumo energético considerando un modelo con almacenamiento centralizado.

CAPACIDAD [MW]	***POTENCIA [pu]***	***PERDIDAS [$]***	***CONSUMO [$]***
0,5	1/2	0,2610	6,3136
	1/4	0,2610	6,3507
	1/6	0,2610	6,3631
	1/8	0,2610	6,4091
	1/10	0,2610	6,4125
	1/12	0,2610	6,4359
0,75	1/2	0,2610	6,1769
	1/4	0,2610	6,2327
	1/6	0,2610	6,2513
	1/8	0,2610	6,3203
	1/10	0,2610	6,3253
	1/12	0,2610	6,3605
1	1/2	0,2610	6,0403
	1/4	0,2610	6,1146
	1/6	0,2610	6,1394
	1/8	0,2610	6,1571
	1/10	0,2610	6,2706
	1/12	0,2610	6,2963
1,25	1/2	0,2610	5,9037
	1/4	0,2610	6,0177
	1/6	0,2610	6,0276
	1/8	0,2610	6,1426
	1/10	0,2610	6,1510
	1/12	0,2610	6,2236
1,5	1/2	0,2610	5,7670
	1/4	0,2610	5,8785
	1/6	0,2610	5,9157
	1/8	0,2610	6,0537
	1/10	0,2610	6,0638
	1/12	0,2610	6,1510
1,75	1/2	0,2610	5,6305
	1/4	0,2610	5,7605
	1/6	0,2610	5,8038
	1/8	0,2610	5,9649
	1/10	0,2610	5,9530
	1/12	0,2610	6,0784
2	1/2	0,2610	5,4938
	1/4	0,2610	5,6424
	1/6	0,2610	5,6920
	1/8	0,2610	5,8760
	1/10	0,2610	5,9544
	1/12	0,2610	5,9832

Análisis de resultados: Con base en los resultado presentado en la *Tabla 4.10,* se observa que las perdidas de potencia para todos los casos son constantes, la razón de esto es debido a que el sistema de almacenamiento se instaló en la cabecera, por lo cual la potencia para cargar el sistema de almacenamiento no debe ser transportada hasta la cola evitando así que se genere pérdidas por transporte. Ahora bien, en la *Figura* 4.9 se observa que el consumo energético decrece a medida que se aumente la capacidad de almacenamiento y la potencia, contrario al efecto que se tiene en el modelo con almacenamiento distribuido donde el consumo energético desciende cuando se aumenta la potencia y a partir de un punto dado el consumo energético aumenta.

Figura 4.9.: Consumo energético considerando un modelo con almacenamiento centralizado.



Adicionalmente, se observa que el menor consumo energético se logró con alta capacidad de almacenamiento y con alta potencia, lo cual difiere del caso con almacenamiento distribuido donde el conjunto de soluciones óptimas se logró con altas capacidad de almacenamiento pero con bajas potencias.

Por otro lado, en la *Tabla 4.11*, se presenta el conjunto de soluciones de la *Tabla* 4.9 en conjunto con la mejor solución encontrada en la *Tabla* 4.10. Adicionalmente, en la columna 7 se estableció el nuevo dominio del frente de pareto considerando la solución del modelo centralizado.

TABLA 4.11.: Conjunto de soluciones para el sistema de distribución de 7 barras

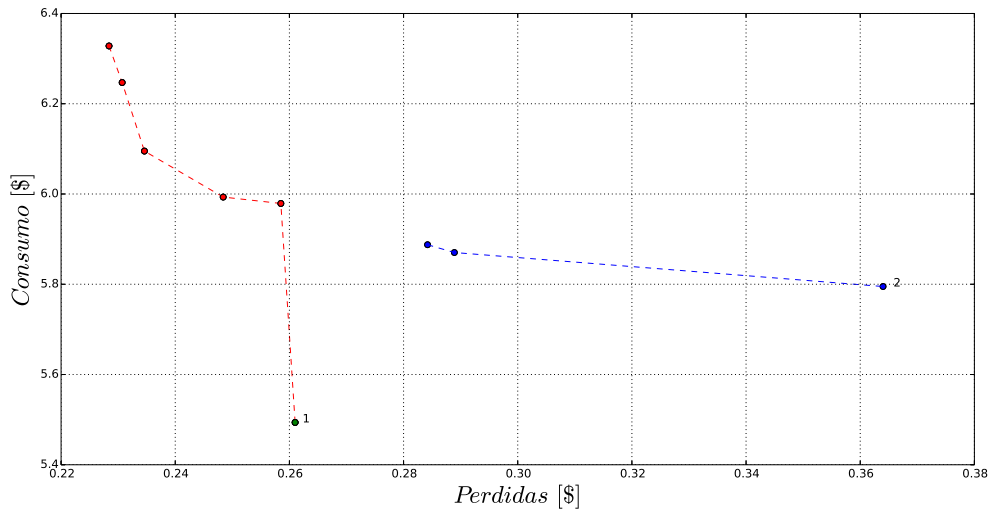
CAP [MW]	***POT [pu]***	***PERDIDAS [$]***	***CONSUMO [$]***	***MODELO***	***FRENTE***
0,75	1/12	0,2284	6,3280	D	1
0,75	1/8	0,2307	6,2470	D	1
1,25	1/8	0,2346	6,0951	D	1
2	1/12	0,2484	5,9932	D	2
2	1/10	0,2585	5,9790	D	1
1,75	1/8	0,2842	5,8876	D	1
2	1/8	0,2889	5,8703	D	2
2	1/6	0,3640	5,7950	D	2
2	1/2	0,2610	5,4938	C	1

C, modelo con almacenamiento centralizado.

D, modelo con almacenamiento distribuido.

Como se observa en *Figura* 4.10 la mejor solución encontrada en el modelo centralizado (punto de color verde, ver) domina sobre tres soluciones del modelo distribuido. Ademas, se observa que 5 soluciones del modelo distribuido mas la solución del modelo centralizado pertenecen el mismo frente de pareto.

Figura 4.10.: Conjunto de soluciones para el sistema de distribución de 7 barras



De acuerdo a los resultados obtenidos, tanto el modelo distribuido como el modelo centralizado presentan buenas soluciones, por lo tanto ambos modelos son aceptados.

5

CONCLUSIONES

- Los algoritmos genéticos son eficientes en la búsqueda de soluciones, pero debido a la gran cantidad de individuos que necesita para cubrir de forma eficiente todo el espacio, es preciso definir cuidadosamente el cromosoma al igual que los parámetros de selección y mutación.

- Las estrategias de optimización basadas en caos para las cuales se utilizó heurísticas fundamentadas en el comportamiento social de ciertas especies, tienen la particularidad de ser algoritmos muy rápidos a pesar de la gran cantidad de individuos que necesita para la exploración del espacio de soluciones..

- La heurística búsqueda arborescente, garantiza un tiempo de solución menor en comparación con los algoritmos genéticos y las estrategias de optimización basadas en caos, ya que permite por medio del árbol inicial de búsqueda evitar la exploración en zonas no factibles que puedan llevar al algoritmo a converger prematuramente hacia óptimos locales.

- El método de *Newton Raphson con sistema P.U. complejo* resulta útil para realizar el seguimiento de la tensión de las barras. Sin embargo, este algoritmo tiene la desventaja que los tiempos de ejecución para encontrar soluciones aumenten en función del número de barras del sistema de distribución.

- El método de *TRX-Power Flow* resulta bastante útil para resolver sistemas de distribución radial en corto de tiempo. Sin embargo, este algoritmo tiene la desventaja de disminuir la precisión de solución para los sistemas de gran tamaño. Por otro lado, este algoritmo presenta limitaciones pues no permite implementar cargar en la barra Slack o tener control sobre el tap's de los transformadores.

- En la prueba 1, se determinó que la mejor heurística para la gestión optima de recursos energéticos locales en un sistema de distribución es *búsqueda arborescente*, debido a la eficiente estrategia de búsqueda ya que permite reducir de manera considerable el espacio de soluciones, teniendo como punto de partida un árbol inicial a diferencia de las demás heurísticas las cuales tienen desventaja al momento de iniciar la exploración con grandes poblaciones, teniendo como consecuencia un exceso en el tiempo de ejecución.

- La curva de precio de compra/venta de la energía eléctrica es dominante respecto a la curva de radiación solar, teniendo como resultado que el SOC del sistema de almacenamiento opte por cargar en los intervalos de tiempo que presenta una disminución del precio y descargar cuando el precio sea elevado.

- Para un modelo con almacenamiento distribuido se determinó que el conjunto de soluciones óptimas (correspondientes al primer frente de pareto) para sistemas de distribución radial o mallado corresponden a casos con alta capacidad de almacenamiento pero con baja potencia.

- Las perdidas de potencia para el modelo con almacenamiento centralizado son constantes debido a que no se transporta potencia adicional a la demanda, evitando así que se genere perdidas por transporte.

- Para un modelo con almacenamiento centralizado se determinó que la solución óptima para sistemas de distribución radial o mallado corresponde al caso con alta capacidad de almacenamiento y con alta potencia.

- No se puede determinar que modelo (distribuido o centralizado) se ajusta mejor al sistema distribución a fin de obtener el menor consumo energético, pues los resultados de la prueba 2 y *Anexo B* indicaron soluciones particulares que no relacionan un sistema de distribución con otro, por lo cual no es posible generalizar una solución.

BIBLIOGRAFÍA

AHUJA, Ravindra K.; MAGNANTI, Thomas L.; ORLIN, James B. Network flows. ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA, 1988.

ARRANZ DE LA PEÑA, Jorge; PARRA TRUYOL, Antonio. Algoritmos Genéticos. Universidad Carlos III, 2007.

BAGLEY, John Daniel. The behavior of adaptive systems which employ genetic and correlation algorithms. 1967.

BESHR, Eman. Comparative study of adding PV/wind energy systems to autonomus micro grid. En Electric Power and Energy Conversion Systems (EPECS), 2013 3rd International Conference on. IEEE, 2013. p. 1-6.

BING, Li; WEISUN, Jiang. Chaos optimization method and its application.Control Theory and Applications, 1997, vol. 14, no 4, p. 613-615.

CAICEDO, Gilberto Carrillo. Explotación óptima de un sistema de distribución de energía eléctrica. Universidad Pontificia de Comillas, 1995.

DASGUPTA, Dipankar; NINO, Fernando. Immunological computation: theory and applications. CRC press, 2008.

DE OLIVEIRA-DE JESUS, P. M.; ALVAREZ, M. A.; YUSTA, J. M. Distribution power flow method based on a real quasi-symmetric matrix. Electric Power Systems Research, 2013, vol. 95, p. 148-159.

ESCOBAR, M. H.; PÉREZ, Z. J. Aplicación de flujo de carga directo a redes de distribución de gran tamaño. Proyecto de grado presentado ante la ilustre Universidad Simón Bolivar para obtener el título de Ingeniero Electricista, 2010.

GARCÍA, Andrea; RESTREPO, Ángela; VELÁSQUEZ, Juan D. Búsqueda aleatoria repetitiva basada en caos. Revista Ingenierías Universidad de Medellín, 2013, vol. 12, no 22, p. 137-146.

GOLDBERG, David E., et al. Genetic algorithms in search optimization and machine learning. Reading Menlo Park: Addison-wesley, 1989.

GRAINGER, John J.; STEVENSON, William D. Análisis de sistemas de potencia. McGraw-Hill, 1996.

HEBER NIETO, J. O. S. É. Olimpiadas matemáticas: el arte de resolver problemas. 2005.

HOLLAND, John H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press, 1975.

ARAUJO, Lourdes; CERVIGÓN, Carlos. Algoritmos evolutivos: un enfoque práctico. Alfaomega, 2009.

NIETO, JM García; TORRES, Enrique Alba. Algoritmos basados en cúmulos de partículas para la resolución de problemas complejos. Septiembre de, 2006.

PEARCE, Joshua M. Photovoltaics—a path to sustainable futures. Futures, 2002, vol. 34, no 7, p. 663-674.

QUINTERO, James Paul Valencia. Generación distribuida: democratización de la energía eléctrica. Criterio Libre, 2008, no 8, p. 105-112.

RAMÍREZ, Samuel. Redes de distribución de energía. Universidad Nacional de Colombia, Sede Manizales, Tercera Edición, Manizales, 2004.

TAVAZOEI, Mohammad Saleh; HAERI, Mohammad. An optimization algorithm based on chaotic behavior and fractal nature. Journal of Computational and Applied Mathematics, 2007, vol. 206, no 2, p. 1070-1081.

TORTELLI, Odilon L., et al. Fast Decoupled Power Flow to Emerging Distribution Systems via Complex pu Normalization. Power Systems, IEEE Transactions on, 2015, vol. 30, no 3, p. 1351-1358.

YANG, Xin-She. A new metaheuristic bat-inspired algorithm. En Nature inspired cooperative strategies for optimization (NICSO 2010). Springer Berlin Heidelberg, 2010. p. 65-74.

ZUNIGA, Guilliermo C., et al. Optimizacion basada en Pronostico para una Micro-red Eléctrica Inteligente. Congreso nacional de control automático, 2013.

ANEXOS

A

ALGORITMOS

A.1 MAESTRO

NOMBRE DEL CAPÍTULO

```
1  import numpy as np
2  from heuristicas import optimizar
3  from pronostico import pronosticos
4  from resultados import graficas, resumen
5
6  ##############################################
7  ############   DATOS DE ENTRADA   ############
8  ##############################################
9
10 case = 'case_7'      # Caso para resolver:
11                      # 'case_7'.
12                      # 'case_13'.
13                      # 'case_33'.
14                      # 'case_30'.
15                      # 'case_34_IEEE'.
16
17 metodo = 'NR'         # Metodo para el flujo de potyencia
18                      # 'NR',      newton raphson.
19                      # 'NR_CPLX', newton raphon bases complejas
```

```
20                      # 'TRX',     flujos radial.
21
22 Heur = 'PSO'      # Metodo heuristico a empleas:
23                   # 'A_G',     algoritmo genetico.
24                   # 'RSS',     busqueda aleatoria repetitiva.
25                   # 'PSO',     cumulo de particulas.
26                   # 'H_BAT',   Metaheuristica del murcielago.
27                   # 'B_A',     busqueda arborescente.
28
29 h = 24               # Numero de horas a optimizar (1 <= h <= 24).
30 dia_C_G = 0          # Dia para el precio de compra/venta de la energia electric
31 dia_P_W = 0          # Dia para la curva de radiacion solar.
32 dia_P_L = 0          # Dia para la curva de demanda.
33
34 curvas = pronosticos(h, dia_C_G, dia_P_W, dia_P_L)   # Funcion para traer:
   # C_G, precio de compra/venta de la energia normalizada.
   # P_w, curva de la radiacion solar normalizada.
   # P_l, curva de la demanda electrica normalizada.
35
36 K = 0.3        # Potencia base para la fuente de generacion renovable [MW].
37 Cap_B = 1     # Potencia base para el sistema de almacenamiento [MW].
38 P_B_base = 1 / 6.  # Capacidad de carga/descarga maxima para el sistema de alm
39
40 Bus_bateria = np.array([[7, 5, 4],      # Numero del bus para conectar la bater
   [0.33, 0.33, 0.33]])    # Porcentaje de la capcidad total de la bateria.
41 Bus_renovable = np.array([[7, 5, 4],   # Numero del bus para conectar la fuent
   [0.33, 0.33, 0.33]])  # Porcentaje de la capcidad total de la fuente renovable
42
43 in_data = np.array([curvas, case, K, Bus_renovable, Cap_B, P_B_base, Bus_bater
44
45 ################################################
46 ############     OPTIMIZACION     ############
47 ################################################
48
49 r = optimizar(in_data, Heur)
50
51 ################################################
52 ############      RESULTADOS      ############
```

```
53 ##############################################
54
55 graficas(r) # Muestra las graficas de potencia activa, reactiva, perfiles de t
56 resumen(r, in_data) # Muestra un resumen de los resultados.
```

A.2 CARGAR CASO

```
1  def loadcase(casedata) :
2
3      if casedata == 'case_4gs' :
4          mpc = case_4gs()
5      if casedata == 'case_7' :
6          mpc = case_7()
7      if casedata == 'case_13' :
8          mpc = case_13()
9      if casedata == 'case_33' :
10         mpc = case_33()
11     if casedata == 'case_30' :
12         mpc = case_30()
13
14     return mpc
```

A.3 Case 4gs

```
1  import numpy as np
2
3  def loadcase(casedata):
4
5      if casedata == 'case_4gs' :
6          baseMVA = 100
7          basekV = 230
8          # Tipo - P_g - Q_g - P_d - Q_shunt - V_pu - ang - V_base
9          bus = np.array([
10                         [0,   0, 0, 50,   30.99,   0,     1,  0,
   basekV],
11                         [1,   0, 0, 170, 105.35,   0,     1,  0,
   basekV],
12                         [1,   0, 0, 200, 123.94,   0,     1,  0,
   basekV],
13                         [2, 318, 0,  80,  49.58,   0,  1.02,  0,
   basekV]])
14
15         bus[:, 1:5] = bus[:, 1:5] / baseMVA
16         #bus[:, 3:5] = bus[:, 3:5] / baseMVA
17
18         # barra_1 - barra_2 - R - X - B/2
19         branch = np.array([
20                         [1, 2, 0.01008, 0.0504, 0.1025 / 2, 1],
21                         [1, 3, 0.00744, 0.0372, 0.0775 / 2, 1],
22                         [2, 4, 0.00744, 0.0372, 0.0775 / 2, 1],
23                         [3, 4, 0.01272, 0.0636, 0.1275 / 2, 1]])
24
25         N_bus = len(bus[:, 0])
26         N_branch = len(branch[:, 0])
27         N_gen = 1
28
29         mpc = [baseMVA, bus, branch, N_bus, N_branch, basekV, N_gen]
30         return mpc
```

A.4 CASE 7

```
1  import numpy as np
2
3  if casedata == 'case_7' :
4
5      baseMVA = 1
6      basekV = 12.47
7
8      # Tipo - P_g - Q_g - P_d - Q_shunt - V_pu - ang - V_base
9      bus = np.array([
10                     [0,   0,  0,   0.0000,  0.0000,   0,   1.05,
   0,   basekV],
11                     [1,   0,  0,   0.1017,  0.0635,   0,   1,
   0,   basekV] ,
12                     [1,   0,  0,   0.0547,  0.0342,   0,   1,
   0,   basekV] ,
13                     [1,   0,  0,   0.0809,  0.0596,   0,   1,
   0,   basekV] ,
14                     [1,   0,  0,   0.1017,  0.0635,   0,   1,
   0,   basekV] ,
15                     [1,   0,  0,   0.0547,  0.0342,   0,   1,
   0,   basekV] ,
16                     [1,   0,  0,   0.0809,  0.0596,   0,   1,
   0,   basekV]])
17
18     # barra_1 - barra_2 - R - X - B/2
19     branch = np.array([
20                     [1,  2,  0.0265,  0.0462,  0,  1] ,
21                     [1,  3,  0.1005,  0.0693,  0,  1] ,
22                     [3,  4,  0.0670,  0.0462,  0,  1] ,
23                     [3,  5,  0.0265,  0.0462,  0,  1] ,
24                     [5,  6,  0.1005,  0.0693,  0,  1] ,
25                     [5,  7,  0.0670,  0.0462,  0,  1]])
26
27     N_bus = len(bus[:, 0])
28     N_branch = len(branch[:, 0])
```

```
29      N_gen = 1
30
31      mpc = [baseMVA, bus, branch, N_bus, N_branch, basekV, N_gen]
32      return mpc
```

A.5 CASE 13

```
1  import numpy as np
2
3  if casedata == 'case_13' :
4
5      baseMVA = 1
6      basekV = 10
7
8      # Tipo - P_g - Q_g - P_d - Q_shunt - V_pu - ang - V_base
9      bus = np.array([
10                     [0,   0,  0,   0.0000,  0.000,    0,   1.05,
   0,   basekV],
11                     [1,   0,  0,   0.1840,  0.046,    0,   1,   0,
   basekV],
12                     [1,   0,  0,   0.0980,  0.034,    0,   1,   0,
   basekV],
13                     [1,   0,  0,   0.1790,  0.045,    0,   1,   0,
   basekV],
14                     [1,   0,  0,   0.1600,  0.184,    0,   1,   0,
   basekV],
15                     [1,   0,  0,   0.1610,  0.060,    0,   1,   0,
   basekV],
16                     [1,   0,  0,   0.0780,  0.011,    0,   1,   0,
   basekV],
17                     [1,   0,  0,   0.1150,  0.006,    0,   1,   0,
   basekV],
18                     [1,   0,  0,   0.0980,  0.013,    0,   1,   0,
   basekV],
19                     [1,   0,  0,   0.1640,  0.020,    0,   1,   0,
   basekV],
20                     [1,   0,  0,   0.1610,  0.060,    0,   1,   0,
   basekV],
21                     [1,   0,  0,   0.0115,  0.006,    0,   1,   0,
   basekV],
22                     [1,   0,  0,   0.0980,  0.034,    0,   1,   0,
   basekV]])
```

```
23
24      # barra_1 - barra_2 - R - X - B/2
25      branch = np.array([
26                      [1,  2,  0.00230, 0.00780, 0, 1],
27                      [2,  3,  0.00030, 0.01140, 0, 1],
28                      [3,  4,  0.01410, 0.02280, 0, 1],
29                      [4,  5,  0.01320, 0.01150, 0, 1],
30                      [5,  6,  0.03750, 0.03270, 0, 1],
31                      [6,  7,  0.01710, 0.01490, 0, 1],
32                      [7,  8,  0.03890, 0.02200, 0, 1],
33                      [8,  9,  0.09060, 0.05130, 0, 1],
34                      [4,  10, 0.10100, 0.05720, 0, 1],
35                      [10, 11, 0.01410, 0.02280, 0, 1],
36                      [11, 12, 0.03750, 0.03270, 0, 1],
37                      [12, 13, 0.01710, 0.01490, 0, 1]])
38
39      N_bus = len(bus[:, 0])
40      N_branch = len(branch[:, 0])
41      N_gen = 1
42
43      mpc = [baseMVA, bus, branch, N_bus, N_branch, basekV, N_gen]
44      return mpc
```

A.6 CASE 33

```
import numpy as np

if casedata == 'case_33' :

    baseMVA = 100
    basekV = 12.66

    # Tipo - P_g - Q_g - P_d - Q_shunt - V_pu - ang - V_base
    bus = np.array([
                    [0,   0,  0,     0,      0,    0,   1.05,   0,
basekV],
                    [1,   0,  0,   100,     60,    0,   1,   0,
basekV],
                    [1,   0,  0,    90,     40,    0,   1,   0,
basekV],
                    [1,   0,  0,   120,     80,    0,   1,   0,
basekV],
                    [1,   0,  0,    60,     30,    0,   1,   0,
basekV],
                    [1,   0,  0,    60,     20,    0,   1,   0,
basekV],
                    [1,   0,  0,   200,    100,    0,   1,   0,
basekV],
                    [1,   0,  0,   200,    100,    0,   1,   0,
basekV],
                    [1,   0,  0,    60,     20,    0,   1,   0,
basekV],
                    [1,   0,  0,    60,     20,    0,   1,   0,
basekV],
                    [1,   0,  0,    45,     30,    0,   1,   0,
basekV],
                    [1,   0,  0,    60,     35,    0,   1,   0,
basekV],
                    [1,   0,  0,    60,     35,    0,   1,   0,
basekV],
```

```
23                      [1,   0,  0,   120,    80,    0,   1,   0,
    basekV],
24                      [1,   0,  0,    60,    10,    0,   1,   0,
    basekV],
25                      [1,   0,  0,    60,    20,    0,   1,   0,
    basekV],
26                      [1,   0,  0,    60,    20,    0,   1,   0,
    basekV],
27                      [1,   0,  0,    90,    40,    0,   1,   0,
    basekV],
28                      [1,   0,  0,    90,    40,    0,   1,   0,
    basekV],
29                      [1,   0,  0,    90,    40,    0,   1,   0,
    basekV],
30                      [1,   0,  0,    90,    40,    0,   1,   0,
    basekV],
31                      [1,   0,  0,    90,    40,    0,   1,   0,
    basekV],
32                      [1,   0,  0,    90,    50,    0,   1,   0,
    basekV],
33                      [1,   0,  0,   420,   200,    0,   1,   0,
    basekV],
34                      [1,   0,  0,   420,   200,    0,   1,   0,
    basekV],
35                      [1,   0,  0,    60,    25,    0,   1,   0,
    basekV],
36                      [1,   0,  0,    60,    25,    0,   1,   0,
    basekV],
37                      [1,   0,  0,    60,    20,    0,   1,   0,
    basekV],
38                      [1,   0,  0,   120,    70,    0,   1,   0,
    basekV],
39                      [1,   0,  0,   200,   600,    0,   1,   0,
    basekV],
40                      [1,   0,  0,   150,    70,    0,   1,   0,
    basekV],
41                      [1,   0,  0,   210,   100,    0,   1,   0,
    basekV],
```

```
42                      [1,   0,  0,    60,     40,    0,   1,   0,
    basekV]])
43
44      bus[:, 3] = bus[:, 3] / (baseMVA * 1e3)
45      bus[:, 4] = bus[:, 4] / (baseMVA * 1e3)
46
47      # barra_1 - barra_2 - R - X - B/2
48      branch = np.array([
49                      [1,   2,      0.0575, 0.2930, 0, 1],
50                      [2,   3,      0.3076, 0.1567, 0, 1],
51                      [3,   4,      0.2284, 0.1163, 0, 1],
52                      [4,   5,      0.2378, 0.1211, 0, 1],
53                      [5,   6,      0.5110, 0.4411, 0, 1],
54                      [6,   7,      0.1160, 0.3861, 0, 1],
55                      [7,   8,      0.4439, 0.1467, 0, 1],
56                      [8,   9,      0.6426, 0.4617, 0, 1],
57                      [9,   10,    0.6514, 0.4617, 0, 1],
58                      [10,    11,  0.1227, 0.0406, 0, 1],
59                      [11,    12,  0.2336, 0.0772, 0, 1],
60                      [12,    13,  0.2336, 0.7206, 0, 1],
61                      [13,    14,  0.3379, 0.4448, 0, 1],
62                      [14,    15,  0.3687, 0.3282, 0, 1],
63                      [15,    16,  0.4656, 0.3400, 0, 1],
64                      [16,    17,  0.8042, 1.0738, 0, 1],
65                      [17,    18,  0.4567, 0.3581, 0, 1],
66                      [ 2,    19,  0.1023, 0.0976, 0, 1],
67                      [19,    20,  0.9385, 0.8457, 0, 1],
68                      [20,    21,  0.2555, 0.2985, 0, 1],
69                      [21,    22,  0.4423, 0.5848, 0, 1],
70                      [ 3,    23,  0.2815, 0.1924, 0, 1],
71                      [23,    24,  0.5603, 0.4424, 0, 1],
72                      [24,    25,  0.5590, 0.4374, 0, 1],
73                      [ 6,    26,  0.1267, 0.0645, 0, 1],
74                      [26,    27,  0.1773, 0.0903, 0, 1],
75                      [27,    28,  0.6607, 0.5826, 0, 1],
76                      [28,    29,  0.5018, 0.4371, 0, 1],
77                      [29,    30,  0.3166, 0.1613, 0, 1],
78                      [30,    31,  0.6080, 0.6008, 0, 1],
```

```
79                      [31,    32,  0.1937, 0.2258, 0, 1],
80                      [32,    33,  0.2128, 0.3308, 0, 1]])
81
82      N_bus = len(bus[:, 0])
83      N_branch = len(branch[:, 0])
84      N_gen = 1
85
86      mpc = [baseMVA, bus, branch, N_bus, N_branch, basekV, N_gen]
87      return mpc
```

A.7 Case 30

```
import numpy as np

def case_30() :

    baseMVA = 100
    basekV = 135

    # Tipo - P_g - Q_g - P_d - Q_shunt - V_pu - ang - V_base
    bus = np.array([
                    [0,   0,        0,      0,       0,    0,   1.05,
0,    basekV],
                    [2,   60.97,    0,   21.7,    12.7,    0,   1,
0,   basekV],
                    [1,   0,        0,    2.4,     1.2,    0,   1,
0,   basekV],
                    [1,   0,        0,    7.6,     1.6,    0,   1,
0,   basekV],
                    [1,   0,        0,      0,       0,    0,   1,
0,   basekV],
                    [1,   0,        0,      0,       0,    0,   1,
0,   basekV],
                    [1,   0,        0,   22.8,    10.9,    0,   1,
0,   basekV],
                    [1,   0,        0,     30,      30,    0,   1,
0,   basekV],
                    [1,   0,        0,      0,       0,    0,   1,
0,   basekV],
                    [1,   0,        0,    5.8,       2,    0,   1,
0,   basekV],
                    [1,   0,        0,      0,       0,    0,   1,
0,   basekV],
                    [1,   0,        0,   11.2,     7.5,    0,   1,
0,   basekV],
                    [2,   37,       0,      0,       0,    0,   1,
0,   basekV],
```

```
23                      [1,   0,        0,    6.2,     1.6,    0,   1,
    0,   basekV],
24                      [1,   0,        0,    8.2,     2.5,    0,   1,
    0,   basekV],
25                      [1,   0,        0,    3.5,     1.8,    0,   1,
    0,   basekV],
26                      [1,   0,        0,      9,     5.8,    0,   1,
    0,   basekV],
27                      [1,   0,        0,    3.2,     0.9,    0,   1,
    0,   basekV],
28                      [1,   0,        0,    9.5,     3.4,    0,   1,
    0,   basekV],
29                      [1,   0,        0,    2.2,     0.7,    0,   1,
    0,   basekV],
30                      [2,   21.59,    0,   17.5,    11.2,    0,   1,
    0,   basekV],
31                      [2,   19.20,    0,      0,       0,    0,   1,
    0,   basekV],
32                      [1,   0,        0,    3.2,     1.6,    0,   1,
    0,   basekV],
33                      [1,   0,        0,    8.7,     6.7,    0,   1,
    0,   basekV],
34                      [1,   0,        0,      0,       0,    0,   1,
    0,   basekV],
35                      [1,   0,        0,    3.5,     2.3,    0,   1,
    0,   basekV],
36                      [2,   26.91,    0,      0,       0,    0,   1,
    0,   basekV],
37                      [1,   0,        0,      0,       0,    0,   1,
    0,   basekV],
38                      [1,   0,        0,    2.4,     0.9,    0,   1,
    0,   basekV],
39                      [1,   0,        0,   10.6,     1.9,    0,   1,
    0,   basekV]])
40
41      bus[:, 1] = bus[:, 1] / baseMVA
42      bus[:, 3] = bus[:, 3] / baseMVA
43      bus[:, 4] = bus[:, 4] / baseMVA
```

```
44
45      # barra_1 - barra_2 - R - X - B/2
46      branch = np.array([
47                        [1,   2,     0.02, 0.06, 0.03/2., 1],
48                        [1,   3,     0.05, 0.19, 0.02/2., 1],
49                        [2,   4,     0.06, 0.17, 0.02/2., 1],
50                        [3,   4,     0.01, 0.04, 0.00/2., 1],
51                        [2,   5,     0.05, 0.20, 0.02/2., 1],
52                        [2,   6,     0.06, 0.18, 0.02/2., 1],
53                        [4,   6,     0.01, 0.04, 0.00/2., 1],
54                        [5,   7,     0.05, 0.12, 0.01/2., 1],
55                        [6,   7,     0.03, 0.08, 0.01/2., 1],
56                        [6,   8,     0.01, 0.04, 0,       1],
57                        [6,   9,     0.00, 0.21, 0,       1],
58                        [6,  10,     0.00, 0.56, 0,       1],
59                        [9,  11,     0.00, 0.21, 0,       1],
60                        [9,  10,     0.00, 0.11, 0,       1],
61                        [4,  12,     0.00, 0.26, 0,       1],
62                        [12, 13,     0.00, 0.14, 0,       1],
63                        [12, 14,     0.12, 0.26, 0,       1],
64                        [12, 15,     0.07, 0.13, 0,       1],
65                        [12, 16,     0.09, 0.20, 0,       1],
66                        [14, 15,     0.22, 0.20, 0,       1],
67                        [16, 17,     0.08, 0.19, 0,       1],
68                        [15, 18,     0.11, 0.22, 0,       1],
69                        [18, 19,     0.06, 0.13, 0,       1],
70                        [19, 20,     0.03, 0.07, 0,       1],
71                        [10, 20,     0.09, 0.21, 0,       1],
72                        [10, 17,     0.03, 0.08, 0,       1],
73                        [10, 21,     0.03, 0.07, 0,       1],
74                        [10, 22,     0.07, 0.15, 0,       1],
75                        [21, 22,     0.01, 0.02, 0,       1],
76                        [15, 23,     0.10, 0.20, 0,       1],
77                        [22, 24,     0.12, 0.18, 0,       1],
78                        [23, 24,     0.13, 0.27, 0,       1],
79                        [24, 25,     0.19, 0.33, 0,       1],
80                        [25, 26,     0.25, 0.38, 0,       1],
81                        [25, 27,     0.11, 0.21, 0,       1],
```

```
82                      [28, 27,     0.00, 0.40, 0,     1],
83                      [27, 29,     0.22, 0.42, 0,     1],
84                      [27, 30,     0.32, 0.60, 0,     1],
85                      [29, 30,     0.24, 0.45, 0,      1],
86                      [8,  28,     0.06, 0.20, 0.02/2., 1],
87                      [6,  28,     0.02, 0.06, 0.01/2., 1]])
88
89      N_bus = len(bus[:, 0])
90      N_branch = len(branch[:, 0])
91      N_gen = 6
92      mpc = [baseMVA, bus, branch, N_bus, N_branch, basekV, N_gen]
93
94      return mpc
```

A.8 CURVAS DE RADIACIÓN SOLAR, DEMANDA Y PRECIO DE LA ENERGÍA ELÉCTRICA

```
1  import numpy as np
2
3  def pronosticos(horas, dia_C_G, dia_P_W, dia_P_L) :
4
5      Precios = np.genfromtxt('PB0109201130042012.csv', delimiter = ',')
6      Solar   = np.genfromtxt('RS0109201130042012.csv', delimiter = ',')
7      Demanda = np.genfromtxt('DL0109201130042012'    , delimiter = ',')
8      C_G = np.array(Precios[dia_C_G, : 24])
9      P_W = np.array(Solar[dia_P_W, : 24])
10     P_L = np.array(Demanda[dia_P_L * 24 : 24 * (dia_P_L + 1)])
11     C_G = C_G / max(C_G)
12     P_W = P_W / max(P_W)
13     P_L = P_L / max(P_L)
14
15     out = [C_G[:horas], P_W[:horas], P_L[:horas]]
16     return out
```

A.9 Optimizar

```
1  from time import time
2
3  def optimizar(in_data, heuristica) :
4
5      start_time = time()                # Inicializa el tiempo del CPU.
6
7      if heuristica == 'A_G' :
8          out = A_G(in_data)
9      if heuristica == 'RSS' :
10         out = RSS(in_data)
11     if heuristica == 'PSO' :
12         out = PSO(in_data)
13     if heuristica == 'H_BAT' :
14         out = H_BAT(in_data)
15     if heuristica == 'B_A' :
16         out = B_A(in_data)
17
18     end_time = time() - start_time     # Calcula el tiempo transcurrido en la o
19
20     r = [out, end_time]
21     return r
```

A.10 ALGORITMO GENÉTICO

```
1  import numpy as np
2  from funcion_objetivo import funcion_evaluacion
3
4  def algo_gene(in_data) :
5
6      def maximo( lista ) :
7          h = 0
8          while lista[h] != max(lista) :
9          h = h + 1
10         return h
11
12     bit = in_data[7]
13     PI = 50                              # POBLACION TOTAL
14     TS = 0.75                            # TASA DE SELECCION [0 : 1]
15     TM = 0.01                            # TASA DE MUTACION  [0 : 1]
16     caso = 'minimo'                      # CASO MAXIMIZAR O MINIMIZAR
17     stop = 0                             # CRITERO DE TRUNCAMIENTO
18     generaciones = 10                    # LIMITE DE TRUNCAMIENTO
19     generacion = 0                       # CONTADOR DE GENERACIONES
20     optimo_global = np.zeros(300)        # HISTORIA DEL INIVIDUO MEJOR ADAPTADO
21     pob_total = np.zeros((2 * PI, bit))
22     pob_hijos = np.zeros((PI, bit))      # POBLACION DE LOS HIJOS
23     M_padres = np.zeros((PI, 2))
24     M_hijos = np.zeros((PI, 2))
25     M_total = np.zeros((2 * PI, 2))
26
27     #  POBLACION DE PADRES
28     pob_padres = np.round(np.random.rand(PI, bit))
29     # FUNCIONDE OBJETIVO (padres)
30     for ind in range(PI)  :
31         M_padres[ind, 0], pob_padres[ind, :], out = funcion_evaluacion(pob_padres[ind, :], in
32     # FUNCION DE ADAPTACION (padres)
33     if caso == 'minimo' :
34         M_padres = - np.array(M_padres)
35         f = M_padres[:, 0] - min(M_padres[:, 0])
36         M_padres[:, 1]= f / max(f)
37
38     while stop <= generaciones :
39
40         for ind in range(PI / 2) :
41             padre = 1
42             madre = 1
43             adap_p = 0.1
44
```

```
45              # SELECCION  DE LOS MEJORES PADRES
46              while adap_p <= TS or padre == madre :
47                  padre = np.random.randint(0, PI)
48                  adap_p = M_padres[padre, 1]
49                  madre = np.random.randint(0, PI)
50
51              # CRUCE
52              cruce = np.random.randint(1, bit - 1)
53              pob_hijos[2 * ind, : cruce] = np.array(pob_padres[padre, : cruce])
54              pob_hijos[2 * ind, cruce :] = np.array(pob_padres[madre, cruce :])
55              pob_hijos[2 * ind + 1, : cruce] = np.array(pob_padres[madre, : cruce])
56              pob_hijos[2 * ind + 1, cruce :] = np.array(pob_padres[padre, cruce :])
57
58          # MUTACION DE LOS HIJOS
59          for ind in range(int(np.round(TM * PI * bit))) :
60          pos_x = np.random.randint(0, bit)
61          pos_y = np.random.randint(0, PI)
62          pob_hijos[pos_y, pos_x] = np.round(np.random.rand(1))
63
64          # FUNCIONDE OBJETIVO (hijos)
65          for ind in range(PI)  :
66              M_hijos[ind, 0], pob_hijos[ind, :], out = funcion_evaluacion(pob_hijos[ind, :], i
67
68          # FUNCION DE ADAPTACION (hijos)
69          if caso == 'minimo' :
70              M_hijos = - np.array(M_hijos)
71              M_total[: PI,:] = np.array(M_padres)
72              M_total[PI :,:] = np.array(M_hijos)
73              f = M_total[:, 0] - min(M_total[:, 0])
74              M_total[:, 1]= f / max(f)
75
76          # SUPERVIVIENTES
77          pob_total[: PI,:] = np.array(pob_padres)
78          pob_total[PI :,:] = np.array(pob_hijos)
79          pob_aux = np.array(pob_total)
80          fund_adap = np.array(M_total[:, 1])
81          M_aux = np.array(M_total)
82
83          for ind in range(PI) :
84              posicion = maximo(fund_adap)
85              fund_adap[posicion] = 0
86              pob_padres[ind, :] = np.array(pob_aux[posicion, :])
87              M_padres[ind, :] = np.array(M_total[posicion, :])
88
89          # CONDICION DE TRUNCAMIENTO
90          fund_adap = np.array(M_total[:, 1])
```

```
91          if generacion == 0 :
92              posicion = maximo(fund_adap)
93              mejor = np.array(M_total[posicion, 0])
94              posicion = maximo(fund_adap)
95              optimo = np.array(M_total[posicion, 0])
96          if mejor == optimo :
97              stop = stop + 1
98          else:
99              stop = 0
100             mejor = np.array(optimo)
101
102         # HISTORIA DEL INDIVIDUO MEJOR ADAPTADO
103         if caso == 'minimo' :
104             optimo_global[generacion]  = - optimo
105         else:
106             optimo_global[generacion]  =  optimo
107
108         generacion = generacion + 1
109
110     f, X_optimo, out = funcion_evaluacion(pob_total[posicion, :], in_data, 'reparar')
111
112     return out
```

A.11 BÚSQUEDA ALEATORIA REPETITIVA BASADA EN CAOS

```
1  import numpy as np
2  from funcion_objetivo import funcion_evaluacion
3
4  def RRS(in_data) :
5
6      def minimo(lista) :
7          h = 0
8          while lista[h] != min(lista) :
9              h = h + 1
10         return h
11
12     n = in_data[7]
13     PI = 80
14     limite = 10
15     funcion = 5
16     optimo = np.zeros(300)
17     f_p = np.zeros(PI)
18
19     # POBLACION INICIAL
20     X = np.round(np.random.rand(PI, n))
21     X[0, :] = generar_estado(n, C_G)
22     dX = np.zeros(n)
23     Xo = np.zeros(n)
24
25     # FUNCIONDE OBJETIVO (pob incial)
26     for ind in range(PI)  :
27         f_p[ind], X[ind, :], out = funcion_evaluacion(X[ind, :], in_data, 'off')
28
29     Z = np.array(X)
30     lamb = 5 * np.random.rand(PI)
31     betha = np.random.rand(1)
32     pitha = np.random.rand(1)
33
34     t = 0
35     stop = 0
36
37     while stop <= limite :
38         for ind in range(PI) :
39             optimo_G = minimo(f_p)
40             optimo[t] = f_p[optimo_G]
```

```
41              u = np.random.randint(0, 2, n)
42
43              betha = np.random.randint(0, n, 1)
44              if np.random.rand(1) > 0.5 :
45                  dX[: betha] = np.array(u[: betha])
46                  dX[betha :] = np.array(Z[ind, betha :])
47              else :
48                  dX[: betha] = np.array(Z[ind,: betha])
49                  dX[betha :] = np.array(u[betha :])
50
51              lamb = np.random.randint(0, n, 1)
52              if np.random.rand(1) > 0.5 :
53                  Xo[: lamb] = np.array(dX[: lamb])
54                  Xo[lamb :] = np.array(X[ind, lamb :])
55              else :
56                  Xo[: lamb] = np.array(X[ind,: lamb])
57                  Xo[lamb :] = np.array(dX[lamb :])
58
59              f_f, Xo, out = funcion_evaluacion(Xo, in_data, 'off')
60
61              if f_f < f_p[ind]  :
62                  f_p[ind] = f_f ;
63                  X[ind, :] = np.array(Xo)
64
65                  pitha = np.random.randint(0, n, 1)
66                  if np.random.rand(1) > 0.5 :
67                      Z[ind, : pitha] = np.array(dX[: pitha])
68                  else :
69                      Z[ind, pitha :] = np.array(dX[pitha :])
70
71          if t > 0 :
72              if abs( optimo[t] - optimo[t - 1] ) <= 0.00001 :
73                  stop = stop + 1 ;
74              else :
75                  stop = 0
76
77          t = t + 1
78
79      f, X_optimo, out = funcion_evaluacion(X[optimo_G], in_data, 'reparar')
80
81      return out
```

A.12 CÚMULO DE PARTÍCULAS

```
1  import numpy as np
2  from funcion_objetivo import funcion_evaluacion
3
4  def PSO(in_data) :
5
6      def minimo(lista) :
7          h = 0
8          while lista[h] != min(lista) :
9              h = h + 1
10             return h
11
12     n = in_data[7]
13     Pl = 50
14     limite = 10
15     rango = 1
16     stop = 0
17     t = 0
18     optimo = np.zeros(300)
19     f_p = np.zeros(Pl)
20     comp = n
21
22     # POBLACION INICIAL
23     X = np.round(np.random.rand(Pl, n))
24     X[0, :] = generar_estado(n, C_G)
25     V = np.zeros(n)
26
27     # FUNCIONDE OBJETIVO (pob inicial)
28     for ind in range(Pl) :
29         f_p[ind], X[ind, :], out = funcion_evaluacion(X[ind, :], in_data, 'off')
30
31     optimo_P = np.array(X)
32     f_aux = np.array(f_p)
33
34     while stop < limite :
35
36         comp = comp - 1
37         if comp < 1 :
38             comp = 0
39
40         for ind in range(Pl) :
41
42             if f_p[ind] < f_aux[ind] :
43                 optimo_P[ind, :] = X[ind, :]
44                 f_aux[ind] = f_p[ind]
```

```
45
46          optimo_C = minimo(f_p)
47          optimo[t] = f_p[optimo_C]
48
49          # VELOCIDAD DEL CUMULO
50          V_cumulo = np.array(X[optimo_C, :])
51
52          # VELOCIDAD PARTICULAR
53          V_particular = np.array(optimo_P[ind, :])
54
55          # SUMA DE LA VELOCIDADES
56          if np.random.rand(1) > 0.5 :
57              V[: comp] = np.array(V_particular[: comp])
58              V[comp :] = np.array(V_cumulo[comp :])
59          else :
60              V[: comp] = np.array(V_cumulo[: comp])
61              V[comp :] = np.array(V_particular[comp :])
62
63          # POSICION ACTUAL DE LA PARTICULA
64          corte = np.random.randint(0, n, 1)
65          if np.random.rand(1) > 0.5 :
66              X[ind, : corte] = np.array(V[: corte])
67          else :
68              X[ind, corte :] = np.array(V[corte :])
69
70          # FACTOR DE INERCIA
71          if np.random.rand(1) > 0.3 :
72              punto = np.random.randint(0, n, 1)
73              X[ind, punto] = np.array(np.round(np.random.rand(1)))
74
75          # EVALUACION DE LA NUEVA PARTICULA
76          f_p[ind], X[ind, :], out = funcion_evaluacion(X[ind, :], in_data, 'off')
77
78      if t > 0 :
79          if abs(optimo[t] - optimo[t - 1]) <= 0.0001 :
80              stop = stop + 1
81          else:
82              stop = 0
83      t = t + 1
84
85      f, X_optimo, out = funcion_evaluacion(X[optimo_C], in_data, 'reparar')
86
87      return out
```

A.13 Metaheurística del murciélago

```
1  import numpy as np
2  from funcion_objetivo import funcion_evaluacion
3
4  def H_BAT(in_data) :
5
6      def minimo(lista) :
7          h = 0
8          while lista[h] != min(lista) :
9          h = h + 1
10         return h
11
12     n = in_data[7]
13     Pl = 50
14     limite = 10
15     rango = 1
16     stop = 0
17     t = 0
18     optimo = np.zeros(300)
19     f_p = np.zeros(Pl)
20
21     # PARAMETROS DE SONIDO Y PULSOS
22     alpha = 0.8
23     pitha = 1 / 50.
24     ro = np.random.rand(Pl)
25     r = np.array(ro)
26     A = np.random.rand(Pl)
27
28     # VECTORES TEMPORALES
29     Xo = np.zeros(n)
30     Vo = np.zeros(n)
31
32     # POSICIONES Y VELOCIDADES INICIALES
33     X = np.round(np.random.rand(Pl, n))
34     V = np.round(np.random.rand(Pl, n))
35
36     # FUNCIONDE OBJETIVO
37     for ind in range(Pl)  :
38         f_p[ind], X[ind, :], out = funcion_evaluacion(X[ind, :], in_data, 'off')
39
40     while stop < limite :
41
42         for ind in range(Pl) :
43
44             # VELOCIDAD GLOBAL
```

```
45              optimo_G = minimo(f_p)
46              optimo[t] = f_p[optimo_G]
47              dV = np.array(X[optimo_G, :])
48
49              # VELOCIDAD LOCAL
50              if np.random.rand(1) > r[ind] :
51                  punto = np.random.randint(0, n, 1)
52                  dV[punto] = np.array(np.round(np.random.rand(1)))
53
54              # SUMA DE LAS VELOCIDADES
55              corte = np.random.randint(0, n / 2, 1)
56              Vo[: corte] = np.array(V[ind, : corte])
57              Vo[corte :] = np.array(dV[corte :])
58
59              # POSICION ACTUAL DE LA PARTICULA
60              corte = np.random.randint(0, n, 1)
61              if np.random.rand(1) > 0.5 :
62                  Xo[: corte] = np.array(X[ind, : corte])
63                  Xo[corte :] = np.array(Vo[corte :])
64              else :
65                  Xo[: corte] = np.array(Vo[: corte])
66                  Xo[corte :] = np.array(X[ind, corte :])
67
68              # EVALUACION DE LA NUEVA PARTICULA
69              f_f, Xo, out = funcion_evaluacion(Xo, in_data, 'off')
70
71              if f_f < f_p[ind] and np.random.rand(1) > A[ind]:
72                  X[ind, :] = np.array(Xo)
73                  V[ind, :] = np.array(Vo)
74                  f_p[ind] = np.array(f_f)
75                  A[ind] = np.array(alpha * A[ind])
76                  r[ind] = np.array(ro[ind] * (1 - np.exp(- pitha * t)))
77
78          if t > 0 :
79              if abs(optimo[t] - optimo[t - 1]) <= 0.00001 :
80                  stop = stop + 1
81              else:
82                  stop = 0
83
84          t = t + 1
85
86      f, X_optimo, out = funcion_evaluacion(X[optimo_G], in_data, 'reparar')
87
88      return out
```

A.14 Búsqueda arborescente

```
import numpy as np
from funcion_objetivo import funcion_evaluacion

def B_A(in_data) :

    def arbol_inicial(in_data) :

        C_G = in_data[0][0]       # Precio de compra/venta de la energia electrica
        h = in_data[7]            # Numero de horas a optimizar
        estado = np.zeros(h)

        for t in range(1, h) :
            if C_G[t] < C_G[t - 1] :
                estado[t] = 0
            else :
                estado[t] = 1

        return estado

    n = in_data[7]           # Numero de horas a optimizar
    arbol = arbol_inicial(in_data)
    f_t = np.zeros(100)
    cont = 0
    stop = 0

    while stop == 0 :

        f_1, arbol, out = funcion_evaluacion(arbol, in_data, 'off')
        f_t[cont] = f_1

        for t in range(1, n) :

            arbol_aux = np.array(arbol)

            if arbol[t] == 0 :
                arbol_aux[t] = 1
            else :
                arbol_aux[t] = 0

            f_2, arbol_aux, out = funcion_evaluacion(arbol_aux, in_data, 'off')

            if f_1 > f_2 :
                arbol = np.array(arbol_aux)
                if np.random.rand(1) < 0.01 :
```

```
45                      b_g = np.random.randint(1, n - 1)
46                      if arbol[b_g] == 0 :
47                          arbol[b_g] == 1
48                      else :
49                          arbol[b_g] == 0
50          if cont > 0 :
51              if f_t[cont] == f_t[cont - 1] :
52              stop = 1
53
54          cont = cont + 1
55
56      f, arbol, out = funcion_evaluacion(arbol, in_data, 'reparar')
57
58      return out
```

A.15 DINÁMICA DE LA BATERÍA

```
1  import numpy as np
2  from case import loadcase
3
4  def dinamica_bateria(estado, in_data, reparar) :
5      h = in_data[7]
6      P_B = np.zeros(h)        # Potencia de la bateria
7      SOC = np.zeros(h)        # Nivel de carga de la barteria
8      SOC_max = 1
9      SOC_min = 0.1
10     SOC[0] = SOC_min
11     P_B_base = in_data[4]
12
13     # start informacio del flujo
14     curva = in_data[0]
15     P_l = curva[2]           # Demanda normalizada
16     P_W = curva[1] * in_data[2]
17     P_g = np.zeros(h)        # Activa generada
18     Q_g = np.zeros(h)        # Reactiva generada
19     P_g_t = np.zeros(h)        # Activa generada total
20     Q_g_t = np.zeros(h)        # Reactiva generada total
21     P_d = np.zeros(h)        # Activa demanda
22     Q_d = np.zeros(h)        # Reactiva demanda
23
24     mpc = loadcase(in_data[1])
25     N_bus = mpc[3]           # Numero de buses
26     buses = np.zeros((N_bus, h))
27     # end informacion del flujo
28
29     B_b = in_data[6]      # Posicion de la bateria
30     B_r = in_data[3]    # Posicion de la fuente renovable
31
32     for t in range(h) :
33         delta_SOC = np.array(in_data[5])
34         if t == 0 :
35             P_g[t], P_d[t], P_W[t], buses[:, t], Q_g[t], Q_d[t], P_g_t[t], Q_g_t[t] = ...
36             power_flow(in_data, P_B[t], P_l[t], P_W[t], t)
37             P_B[t] = 0
38         else :
39             # Descarga bateria (inyecta potencia)
40             if estado[t] == 0 :
41                 cond = 2
42                 while cond > 1.1 :
43                     SOC[t] = SOC[t - 1] - delta_SOC
44                     if SOC[t] < SOC_min :
45                         SOC[t] = SOC_min
46                         if SOC[t] == SOC[t - 1] and reparar == 'reparar' :
47                             estado[t] = 2
48                     # Potencia de la bateria
49                     P_B[t] = (SOC[t - 1] - SOC[t]) * P_B_base
50                     P_g[t], P_d[t], P_W[t], buses[:, t], Q_g[t], Q_d[t], P_g_t[t], Q_g_t[t] = ...
51                     power_flow(in_data, P_B[t], P_l[t], P_W[t], t)
```

```
52                      cond = max(buses[:, t])
53                      delta_SOC = 0.9 * delta_SOC
54
55              # Carga bateria (Extrae potencia)
56              if estado[t] == 1:
57                  cond = 0.5
58                  while cond < 0.9 :
59                      SOC[t] = SOC[t - 1] + delta_SOC
60                      if SOC[t] > SOC_max :
61                          SOC[t] = SOC_max
62                              if SOC[t] == SOC[t - 1] and reparar == 'reparar' :
63                                  estado[t] = 2
64                      # Potencia de la bateria
65                      P_B[t] = (SOC[t - 1] - SOC[t]) * P_B_base
66                      P_g[t], P_d[t], P_W[t], buses[:, t], Q_g[t], Q_d[t], P_g_t[t], Q_g_t[t] = ...
67                      power_flow(in_data, P_B[t], P_l[t], P_W[t], t)
68                      cond = min(buses[:, t])
69                      delta_SOC = 0.9 * delta_SOC
70
71              # Desconexion
72              if estado[t] == 2 and reparar == 'reparar' :
73                  SOC[t] = SOC[t - 1]
74
75      return estado, SOC, P_B, P_g, P_d, P_W, buses, Q_g, Q_d, P_g_t, Q_g_t
```

A.16 POWER FLOW

```
1  import numpy as np
2  from flujo_radial import runTRX
3  from newton_raphson import runpf, cplx_base, cplx_real, pu_cplx
4  from case import loadcase
5
6  def power_flow(in_data, P_B, P_l, P_W, t):
7
8      B_b = in_data[6]      # Posicion de la bateria
9      B_r = in_data[3]      # Posicion de la fuente renovable
10
11     # ADQUISICION DEL CASO A RESOLVER
12     case = in_data[1]
13     mpc = loadcase(case)
14     bus = mpc[1]
15
16     # INGRESA LA DEMANDA AL SISTEMA
17     bus[:, 1] = bus[:, 1] * P_l
18     bus[:, 3] = bus[:, 3] * P_l
19     bus[:, 4] = bus[:, 4] * P_l
20     P_d = sum(bus[:, 3])
21     Q_d = sum(bus[:, 4])
22
23     # INGRESA LA POTENCIA DE LA BATERIA Y LA FOTOVOLTAICA
24     for k in range(len(B_r[0, :])) :
25         # Ingresa la potencia de fuente renovable
26         bus[B_r[0, k] - 1, 3] = bus[B_r[0, k] - 1, 3] - P_W * B_r[1, k] / abs(mpc[0])
27     for k in range(len(B_r[0, :])) :
28         # Ingresa la potencia de la bateria
29         bus[B_b[0, k] - 1, 3] = bus[B_b[0, k] - 1, 3] - P_B * B_b[1, k] / abs(mpc[0])
30
31     # FLUJO DE POTENCIA
32     if in_data[8] == 'NR_CPLX' :
33         base_compleja = cplx_base(mpc)
34         mpc_cplx = pu_cplx(mpc, base_compleja)
35         r_cplx = runpf(mpc_cplx)
36         r = cplx_real(r_cplx)
37     if in_data[8] == 'NR' :
38         r = runpf(mpc)
39     if in_data[8] == 'TRX' :
40         r = runTRX(mpc)
```

```
41      bus = np.array(r[1])
42      P_g = bus[0, 1]
43      Q_g = bus[0, 2]
44      P_g_t = sum(bus[:, 1])
45      Q_g_t = sum(bus[:, 2])
46      buses = bus[:, 6]
47
48      return P_g, P_d, P_W, buses, Q_g, Q_d, , P_g_t, Q_g_t
```

A.17 Función de evaluación

```
1  import numpy as np
2  from bateria import dinamica_bateria
3
4  def funcion_evaluacion(perfil, in_data, reparar) :
5
6      # Dinamica de la bateria
7      perfil, SOC, P_B, P_G, P_L, P_W, buses, Q_g, Q_d, P_g_t, Q_g_t = ...
8      dinamica_bateria(perfil, in_data, reparar)
9
10     C_G = in_data[0][0]          # Precio de la energia electrica
11     f = sum(C_G * (P_G - P_B))   # Funcion evaluacion
12
13     out = [perfil, C_G, P_L, P_W, SOC, P_B, P_G, buses, Q_g, Q_d, P_g_t, Q_g_t]
14
15     return f, perfil, out
```

A.18 Newton Raphson

```
import numpy as np
import math as mt

#########################################################################
##### METODO PARA RESOLVER LOS SIST. DE DISTRIBUCION #######
#########################################################################

def cplx_bases(mpc) :
    bus = mpc[1]
    branch = mpc[2]
    N_bus = mpc[3]
    N_branch = mpc[4]
    base_MVA = mpc[0]
    base_KV = mpc[5]
    R_X = np.zeros(N_branch)

    for k in range(N_branch) :
        if branch[k, 2] == 0 :
            R_X[k] = np.pi / 2
        else :
            R_X[k] = mt.atan(branch[k, 3] / branch[k, 2])

    alpha =  sum(R_X) / N_branch    gamma = np.mean([max(R_X), min(R_X)])
    P_Q = np.zeros(N_bus)
    c = 0

    for k in range(N_bus) :
        if bus[k, 3] != 0 :
            P_Q[c] = np.cos(mt.atan(bus[k, 4]) / bus[k, 3])
            c = c + 1
        if bus[k, 3] == 0 and bus[k, 4] != 0:
            P_Q[c] = 1
            c = c + 1

    e = 1 - sum(P_Q) / c
```

```
36      phi_base = (0.5 * np.pi  - (alpha + gamma) / 2) * (1 + e)
37
38      Scplx_base = base_MVA * np.exp(- phi_base * 1j)
39      Zcplx_base = base_KV ** 2 / np.conj(Scplx_base)
40      Icplx_base = base_KV / Zcplx_base
41
42      base_compleja = np.array([Scplx_base, Zcplx_base, Icplx_base,])
43      return base_compleja
44
45
46  def cpu_normalization(mpc, base_compleja) :
47      base_MVA = mpc[0]
48      base_KV = mpc[5]
49      base_Z = base_KV ** 2 / base_MVA
50
51      Sg_cplx = base_MVA * (mpc[1][:, 1] + mpc[1][:, 2] * 1j) / base_compleja[0]
52      mpc[1][:, 1] = Sg_cplx.real
53      mpc[1][:, 2] = Sg_cplx.imag
54
55      Sd_cplx = base_MVA * (mpc[1][:, 3] + mpc[1][:, 4] * 1j) / base_compleja[0]
56      mpc[1][:, 3] = Sd_cplx.real
57      mpc[1][:, 4] = Sd_cplx.imag
58
59      Z_cplx = base_Z * (mpc[2][:, 2] + mpc[2][:, 3] * 1j) / base_compleja[1]
60      mpc[2][:, 2] = Z_cplx.real
61      mpc[2][:, 3] = Z_cplx.imag
62
63      mpc[0] = base_compleja[0]
64      return mpc
65
66  def reverse_cpu_normalization(r_plx) :
67      S_cplx = r_plx[0]
68      base_KV = r_plx[5]
69      Z_cplx = base_KV ** 2 / np.conj(S_cplx)
70
71      Sg = (r_plx[1][:, 1] + r_plx[1][:, 2] * 1j) * S_cplx
72      r_plx[1][:, 1] = Sg.real
73      r_plx[1][:, 2] = Sg.imag
```

```
74
75      Sd = (r_plx[1][:, 3] + r_plx[1][:, 4] * 1j) * S_cplx
76      r_plx[1][:, 3] = Sd.real
77      r_plx[1][:, 4] = Sd.imag
78
79      Z = (r_plx[2][:, 2] + r_plx[2][:, 3] * 1j) * Z_cplx
80      r_plx[2][:, 2] = Z.real
81      r_plx[2][:, 3] = Z.imag
82
83      return r_plx
84
85  def runpf(mpc) :
86      ##########################################################
87      ########       METODO DE NEWTON-RAPHSON       #########
88      ########   PARA RESOLVER FLUJOS DE POTENCIA    ########
89      ##########################################################
90
91      P_base = mpc[0]
92      bus = mpc[1]
93      branch = mpc[2]
94      N_barras = mpc[3]
95      N_lineas = mpc[4]
96      N_Slack = 0
97      N_PQ = 0
98      N_PV = 0
99
100     for i in range(N_barras) :
101         if bus[i, 0] == 0 :
102             N_Slack = N_Slack + 1
103         if bus[i, 0] == 1 :
104             N_PQ = N_PQ + 1
105         if bus[i, 0] == 2 :
106             N_PV = N_PV + 1
107
108     Z_line = np.array(branch[:, 2] + branch[:, 3] * 1j)
109     lineas  = np.array(branch[:, : 2]) - 1
110     slack = 0
111     PQ = 1
```

```
112     PV = 2
113     V = bus[:, 6]
114     delta = bus[:, 7]
115     P_g = bus[:, 1]
116     Q_g = bus[:, 2]
117     P_d = bus[:, 3]
118     Q_d = bus[:, 4]
119     t_barra = bus[:, 0]
120
121     ################################################
122     ##########   MATRIZ DE Y_BARRA   ############
123     ################################################
124
125     Y_bus = np.zeros((N_barras, N_barras)) + 0j
126     # Elementos fuera de la diagonal
127     for k in range(N_lineas) :
128         i = np.array(lineas[k, 0])
129         j = np.array(lineas[k, 1])
130         Y_bus[i, j] =  np.array(- 1 / Z_line[k])
131         Y_bus[j, i] =  np.array(- 1 / Z_line[k])
132
133     # Elementos de la diagonal
134     for i in range(N_barras) :
135         for j in range(N_lineas) :
136             if i == lineas[j, 0] :
137                 Y_bus[i, i] = Y_bus[i, i] + 1 / Z_line[j] + branch[j, 4] * 1j
138             if i == lineas[j, 1] :
139                 Y_bus[i, i] = Y_bus[i, i] + 1 / Z_line[j] + branch[j, 4] * 1j
140
141     error = 1
142     Iteraciones = 0
143     while  error > 1e-5 and Iteraciones <= 15 :
144         Iteraciones = Iteraciones + 1
145         ################################################
146         ##########   VECTOR DE ERRORES   ############
147         ################################################
148
149         # Matriz diagonal de tensiones
```

```
150         M_V = np.zeros((N_barras, N_barras)) +0j
151         for k in range(N_barras) :
152             M_V[k, k] = np.array(V[k] * np.exp(delta[k] * 1j))
153             M_V = np.matrix(M_V)
154             Y_bus = np.matrix(Y_bus)
155 
156         # Matriz compleja
157         M_S = np.array(M_V * np.conj(Y_bus) * np.conj(M_V))
158         P = np.real(M_S.T)
159         Q = np.imag(M_S.T)
160         P_cal = np.array(sum(P))
161         Q_cal = np.array(sum(Q))
162         P_prog = P_g - P_d
163         Q_prog = Q_g - Q_d + Q_shunt
164         dP = P_prog - P_cal
165         dQ = Q_prog - Q_cal
166 
167         # Vector de errores
168         df = np.zeros(2 * N_barras - 2 * N_Slack - N_PV)
169         c = 0
170         for i in range(N_barras) :
171             if t_barra[i] != slack :
172                 df[c] = np.array(dP[i])
173                 c = c + 1
174         for i in range(N_barras) :
175             if t_barra[i] != slack and t_barra[i] != PV :
176                 df[c] = np.array(dQ[i])
177                 c = c + 1
178 
179         ##############################################
180         ##########   MATRIZ JACOBIANA    ############
181         ##############################################
182         J_11 = np.array(Q)
183         J_21 = np.array(- P)
184         J_12 = np.array(P)
185         J_22 = np.array(Q)
186 
187         # Correccion de la diagonal
```

```
188         for i in range(N_barras) :
189         J_11[i, i] = J_11[i, i] - Q_cal[i]
190         J_21[i, i] = J_21[i, i] + P_cal[i]
191         J_12[i, i] = J_12[i, i] + P_cal[i]
192         J_22[i, i] = J_22[i, i] + Q_cal[i]
193
194         # Jacobiano J11
195         Ja = np.zeros((N_barras - N_Slack, N_barras - N_Slack))
196         ci = - 1
197         for i in range(N_barras) :
198             if t_barra[i] != slack :
199                 ci = ci + 1
200                 cj = - 1
201         for j in range(N_barras) :
202             if t_barra[j] != slack :
203                 cj = cj + 1
204                 Ja[ci, cj] = J_11[i, j]
205         # Jacobiano J21
206         Jb = np.zeros((N_barras - N_Slack -N_PV, N_barras - N_Slack))
207         ci = - 1
208         for i in range(N_barras) :
209             if t_barra[i] != slack and t_barra[i] != PV :
210                 ci = ci + 1
211                 cj = - 1
212         for j in range(N_barras) :
213             if t_barra[j] != slack :
214                 cj = cj + 1
215                 Jb[ci, cj] = J_21[i, j]
216
217         # Jacobiano J12
218         Jc = np.zeros((N_barras - N_Slack, N_barras - N_Slack - N_PV))
219         ci = - 1
220         for i in range(N_barras) :
221             if t_barra[i] != slack  :
222                 ci = ci + 1
223                 cj = - 1
224                 for j in range(N_barras) :
225                     if t_barra[j] != slack and t_barra[j] != PV :
```

```
226                         cj = cj + 1
227                         Jc[ci, cj] = J_12[i, j]
228 
229         # Jacobiano J22
230         Jd = np.zeros((N_barras - N_Slack - N_PV, N_barras - N_Slack - N_PV))
231         ci = - 1
232         for i in range(N_barras) :
233             if t_barra[i] != slack and t_barra[i] != PV :
234                 ci = ci + 1
235                 cj = - 1
236                 for j in range(N_barras) :
237                     if t_barra[j] != slack and t_barra[j] != PV :
238                         cj = cj + 1
239                         Jd[ci, cj] = J_22[i, j]
240 
241         x_1 = len(Ja[0, :])
242         y_1 = len(Ja[:, 0])
243         x_2 = len(Jd[0, :])
244         y_2 = len(Jd[:, 0])
245 
246         # Matriz jacobiana
247         J = np.zeros((y_1 + y_2, x_1 + x_2))
248         J[: y_1, : x_1] = np.array(Ja)
249         J[y_1 :, : x_1] = np.array(Jb)
250         J[: y_1, x_1 :] = np.array(Jc)
251         J[y_1 :, x_1 :] = np.array(Jd)
252 
253         ##############################################
254         ########## VECTOR DE CORRECIONES ############
255         ##############################################
256         J_aux = np.matrix(J)
257         df_aux = np.matrix(df)
258         dX = J_aux.I * df_aux.T
259         dX = np.array(dX.T)
260 
261         # Vector de actualizacion
262         c = 0
263         for i in range(N_barras) :
```

```
264             if t_barra[i] != slack :
265                 delta[i] = np.array(delta[i] + dX[0, c])
266                 c = c + 1
267         for i in range(N_barras) :
268             if t_barra[i] != slack and t_barra[i] != PV :
269             V[i] = np.array(V[i] * (1 + dX[0, c]))
270             #V[i] = np.array(V[i] + dX[0, c])
271             c = c + 1
272         error = abs(max(df))
273 
274     ################################################
275     #########   CALCULO DE POTENCIA  #############
276     ################################################
277     if P_base.imag != 0:
278         bus[:, 1] = P_cal + P_d
279         bus[:, 2] = Q_cal + Q_d
280     else:
281         bus[:, 1] = (P_cal + P_d) * P_base
282         bus[:, 2] = (Q_cal + Q_d) * P_base
283         bus[:, 3] = bus[:, 3] * P_base
284         bus[:, 4] = bus[:, 4] * P_base
285 
286     ################################################
287     ######### MOSTRAR LOS RESULTADOS #############
288     ################################################
289     bus[:, 7] = bus[:, 7] * 180 / np.pi
290     if Iteraciones < 20 :
291         return mpc
292     else :
293         print '============================='
294         print '␣␣␣␣NO␣SE␣PUEDE␣RESOLVER'
295         print '============================='
```

A.19 FLUJO RADIAL TRX

```
1  import numpy as np
2  from case import loadcase
3
4  def runTRX(mpc) :
5
6      baseMVA = mpc[0]
7      bus = mpc[1]
8      branch = mpc[2]
9      branch[:, 0: 2] = branch[:, 0: 2] - 1
10     N_branch = len(branch[:, 0])
11     N_bus = len(bus[:, 0])
12     T = np.zeros((N_branch, N_branch))
13
14     for i in range(N_branch) :
15         if branch[i, 0] == 0 :
16             T[branch[i, 1] - 1, branch[i, 1] - 1] = 1
17         else :
18             T[:, branch[i, 1] - 1] = T[:, branch[i, 0] - 1]
19             T[branch[i, 1] - 1, branch[i, 1] - 1] = 1
20
21     T = np.matrix(T)
22     S = bus[:, 3] + bus[:, 4] * 1j
23     Z = np.matrix(np.diag(branch[:, 2] + branch[:, 3] * 1j))
24     Vo = np.zeros(N_bus) + bus[0, 6] + 0j
25     S = np.matrix(S[1:])
26     Vo = np.matrix(Vo[1:])
27     V = np.matrix(Vo)
28     TRX = T.T * Z * T
29     iteracion = 1
30     e = 1
31     max_iter = 10
32
33     while e > 0.001 and iteracion <= max_iter  :
34         iteracion = iteracion + 1
35         I = np.conj(S.T / V.T)
```

```
36          DV = TRX * I
37          V_aux = Vo - DV.T
38          e = max(abs(np.array(V_aux.T - V.T)))
39          V = V_aux
40
41      I_branch = T * I
42
43      if iteracion <= max_iter :
44          V = np.array(V.T)
45          for k in range(N_bus - 1) :
46              bus[k + 1, 6] = abs(np.array(V[k]))
47              bus[k + 1, 7] = np.angle(np.array(V[k])) * 180 / np.pi
48
49          S_branch = 0
50          for k in range(N_branch) :
51              if branch[k, 0] == 0 :
52                  S_branch = S_branch + np.conj(np.array(I_branch[k])) * bus[0,
53
54          bus[0, 1] = S_branch.real * baseMVA
55          bus[0, 2] = S_branch.imag * baseMVA
56          bus[:, 3] = bus[:, 3] * baseMVA
57          bus[:, 4] = bus[:, 4] * baseMVA
58
59          return mpc
60      else:
61          print '============================='
62          print '    NO SE PUEDE RESOLVER'
63          print '============================='
```

A.20 GRÁFICAS

```
1  import matplotlib.pyplot as pl
2
3  def graficas(r) :
4
5      mpc = loadcase(in_data[1])
6      out = r[0]
7      C_G = out[1]
8      P_L = out[2] * mpc[0]
9      P_W = out[3]
10     SOC = out[4]
11     P_B = out[5]
12     P_G = out[6]
13     buses = out[7]
14     Q_G = out[8]
15     Q_L = out[9] * mpc[0]
16     P_G_t = out[10]
17     Q_G_t = out[11]
18     tiempo = np.array(range(len(C_G))) + 1
19
20     pl.subplot(2, 2, 1)
21     pl.plot(tiempo, - P_L, 'r')
22     pl.plot(tiempo, P_B, '--k')
23     pl.plot(tiempo, P_G_t, 'y')
24     pl.plot(tiempo, P_W, 'g')
25     pl.ylabel('$Potencia$␣$[MW]$', fontsize = 25)
26     pl.xlabel('$Tiempo$␣$[h]$', fontsize = 25)
27     pl.legend(['$P_L$', '$P_B$', '$P_G$', '$P_W$'])
28     pl.grid(True)
29
30     pl.subplot(2, 2, 2)
31     pl.plot(tiempo, - Q_L, 'r')
32     pl.plot(tiempo, Q_G_t, 'y')
33     pl.ylabel('$Potencia$␣$[MVAr]$', fontsize = 25)
34     pl.xlabel('$Tiempo$␣$[h]$', fontsize = 25)
35     pl.legend(['$Q_L$', '$Q_G$'])
```

```
36      pl.grid(True)
37
38      pl.subplot(2, 2, 3)
39      pl.plot(tiempo, C_G)
40      pl.plot(tiempo, SOC, '--k')
41      pl.xlabel('$Tiempo$␣$[h]$', fontsize = 25)
42      pl.legend(['$C_G$', '$SOC$'])
43      pl.axis([0, 25, 0, 1.1])
44      pl.grid(True)
45
46      pl.subplot(2, 2, 4)
47      N_bus = len(buses[:, 0])
48      for k in range(N_bus) :
49          pl.plot(tiempo, buses[k, :])
50      pl.ylabel('$Tension$␣$[pu]$', fontsize = 25)
51      pl.xlabel('$Tiempo$␣$[h]$', fontsize = 25)
52      pl.axis([0, 25, 0.88, 1.12])
53      pl.grid(True)
54
55      pl.show()
```

A.21 Resumen

```
def resumen(r, in_data) :

    mpc = loadcase(in_data[1])
    out = r[0]
    end_time = r[1]

    print '======================================'
    print 'CARACTERISTICAS␣DEL␣SISTEMA␣␣␣␣␣␣␣␣␣␣␣␣'
    print '======================================'
    print 'SISTEMA␣DE␣DISTRIBUCION␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'
    print 'Caso␣de␣prueba␣␣␣␣␣␣␣␣␣␣␣␣␣', in_data[1]
    print 'Numero␣de␣barras␣␣␣␣␣␣␣␣␣␣␣', mpc[3]
    print 'Numero␣de␣lineas␣␣␣␣␣␣␣␣␣␣␣', mpc[4]
    print 'Numero␣de␣generadores␣␣␣␣␣␣␣', mpc[6]
    print 'Potencia␣base␣␣␣␣␣␣␣␣␣␣␣␣␣␣', mpc[0], '[MVA]'
    print 'Tension␣base␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣', mpc[5], '[kV]'
    print '--------------------------------------'
    print 'SISTEMA␣DE␣ALMACENAMIENTO␣␣␣␣␣␣␣␣␣␣␣␣␣'
    print 'Ubicacion␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣', in_data[6][0, :]
    print 'Potencia␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣', np.round(in_data[4] * in_data[6][1, :], 4) , '[MW]'
    print 'Capacidad␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣', np.round(in_data[5], 4)
    print '--------------------------------------'
    print 'FUENTE␣RENOVABLE␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'
    print 'Ubicacion␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣', in_data[3][0, :]
    print 'Potencia␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣', np.round(in_data[2] * in_data[3][1, :], 4), '[MW]'
    print '======================================'
    print ''
    print ''

    if in_data[9] == 'A_G' :
        Heuristica = 'Algoritmo␣Genetico'
    if in_data[9] == 'RSS' :
        Heuristica = 'Busqueda␣Aleatoria␣Repetitiva'
    if in_data[9] == 'PSO' :
        Heuristica = 'Cumulo␣de␣Partculas'
    if in_data[9] == 'H_BAT' :
        Heuristica = 'Metaheuristica␣del␣Murcielago'
    if in_data[9] == 'B_A' :
        Heuristica = 'Busqueda␣Arborescente'

    if in_data[8] == 'TRX' :
        metodo = 'Flujo␣Radial␣TRX'
    if in_data[8] == 'NR' :
        metodo = 'Newton␣Raphson'
    if in_data[8] == 'NR_CPLX' :
        metodo = 'Newton␣Raphson␣(bases␣complejas)'

    C_G = out[1]
    P_L = out[2] * mpc[0]
    P_W = out[3]
    SOC = out[4]
```

```
    P_B = out[5]
    P_G = out[6]
    Q_G = out[8]
    Q_L = out[9] * mpc[0]
    P_G_t = out[10]
    Q_G_t = out[11]

    print'=========================================================================================
    print'FLUJO␣DE␣POTENCIA␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    print'-----------------------------------------------------------------------------------------
    print '{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}'.format('Hora', 'P_g␣[MW]', 'Q_
␣␣␣␣[MVAr]', 'P_B␣[MW]', 'SOC␣%', 'P_W␣[MW]', 'C_G␣[$]', 'CONSUMO␣[$]')
    print '=========================================================================================
    for k in range(in_data[7]) :
        print '{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}{:^10}'.format(k, np.round(P_G_t[k
        np.round(P_L[k], 4), np.round(Q_L[k], 4), np.round(P_B[k], 4), np.round(SOC[k], 4), np.round(P_W
        np.round(sum(P_G[:k + 1] * C_G[:k + 1]), 4))
    print '=========================================================================================
    print ''
    print ''

    Perdidas_W = np.round(sum(P_G_t + P_B - P_L + P_W), 4)
    Perdidas = sum((P_G_t + P_B - P_L + P_W) * C_G)
    P_G_g = 0   P_G_d = 0

    for k in range(len(C_G)) :
        if P_G_t[k] >= 0 :
            P_G_g = P_G_g + P_G_t[k]
        else :
            P_G_d = P_G_d - P_G_t[k]

    P_B_g = 0
    P_B_d = 0
    for k in range(len(C_G)) :
        if P_B[k] >= 0 :
            P_B_g = P_B_g + P_B[k]
        else :
            P_B_d = P_B_d - P_B[k]

    P_G_g = np.round(P_G_g, 4)
    P_G_d = np.round(P_G_d, 4)
    Q_G = np.round(sum(Q_G), 4)
    P_B_g = np.round(P_B_g, 4)
    P_B_d = np.round(P_B_d, 4)
    P_W = np.round(sum(P_W), 4)
    P_L = np.round(sum(P_L), 4)
    Q_L = np.round(sum(Q_L), 4)

    print '==============================================================================='
    print 'RESUMEN␣FLUJO␣DE␣POTENCIA␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'
    print '-------------------------------------------------------------------------------'
    print '{:^20}{:^15}{:^15}{:^15}{:^15}'.format('␣', 'P_g␣[MW]', 'Q_g␣[MVAr]', 'P_d␣[MW]', 'Q_d␣[MAr]
    print '==============================================================================='
    print '{:^20}{:^15}{:^15}{:^15}{:^15}'.format('F.␣renovable',     P_W,      0,
```

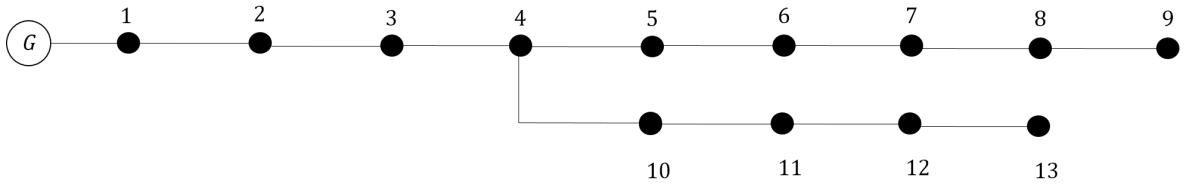
```
    0,          0)
106     print '{:^20}{:^15}{:^15}{:^15}{:^15}'. format('Red central',        P_G_g,      Q_G,
    P_G_d,      0)
107     print '{:^20}{:^15}{:^15}{:^15}{:^15}'. format('S. almacenamiento', P_B_g,      0,
    P_B_d,      0)
108     print '{:^20}{:^15}{:^15}{:^15}{:^15}'. format('S. distribucion',    0,          0,
    P_L,        Q_L)
109     print '--------------------------------------------------------------------------------'
110     print '{:^20}{:^15}{:^15}{:^15}{:^15}'. format(' ', P_W + P_G_g + P_B_g, Q_G, P_G_d + P_B_d + P_L, Q
111     print '================================================================================'
112     print ''
113     print ''
114
115     print '================================================'
116     print 'PARAMETROS DE RENDIEMIENTO                      '
117     print '================================================'
118     print 'Flujo de potencia    ', metodo
119     print 'Heuristica           ', Heuristica
120     print 'Ventana de tiempo    ', in_data[7]
121     print 'Perdidas de potencia ', Perdidas_W, '[MW]'
122     print 'Tiempo CPU           ', np.round(end_time, 4), '[s]'
123     print 'Consumo              ', np.round(sum(P_G * C_G), 4), '[$]'
124     print 'Perdidas             ', np.round(Perdidas, 4), '[$]'
125     print '================================================'
126     print ''
```

B

Casos de prueba

B.1 CASE 13

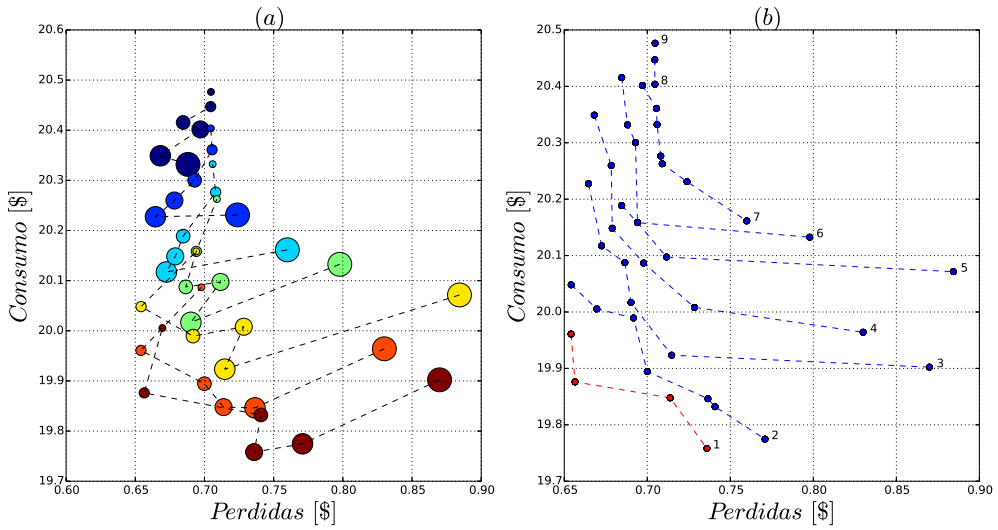
Figura B.1.: Diagrama unifilar para el sistema de distribución de 13 barras.




```
1  ======================================
2  CARACTERISTICAS DEL SISTEMA
3  ======================================
4  SISTEMA DE DISTRIBUCION
5  Caso de prueba              case_13
6  Numero de barras            13
7  Numero de lineas            12
8  Numero de generadores       1
9  Potencia base               1 [MVA]
10 Tension base                10 [kV]
11 --------------------------------------
12 SISTEMA DE ALMACENAMIENTO
13 Ubicacion                   [ 9, 13 ]
14 --------------------------------------
15 FUENTE RENOVABLE
16 Ubicacion                   [ 9, 13]
17 Potencia                    [ 0.5  0.5 ] [MW]
18 ======================================
19
20 ======================================================
21                 MODELO DISTRIBUIDO
22 ------------------------------------------------------
23 CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
24 ======================================================
25     0.5          0.5         0.6881          20.3318
26     0.5          0.25        0.6681          20.349
27     0.5         0.1667       0.6971          20.4016
28     0.5         0.125        0.6846          20.4155
29     0.5          0.1         0.7045          20.4471
30     0.5         0.0833       0.7047          20.4764
31     0.75          0.5        0.7239          20.2311
32     0.75          0.25       0.6646          20.2274
33     0.75         0.1667      0.6783          20.2598
34     0.75         0.125       0.6929          20.3002
35     0.75          0.1        0.7055          20.361
36     0.75         0.0833      0.7046          20.4037
37     1.0          0.5         0.7599          20.1615
38     1.0          0.25        0.6725          20.1173
```

```
39      1.0         0.1667       0.6789          20.1484
40      1.0         0.125        0.6846          20.1887
41      1.0          0.1         0.7081          20.2764
42      1.0         0.0833       0.7059          20.3323
43      1.25          0.5        0.7978          20.1326
44      1.25          0.25       0.6902          20.0169
45      1.25         0.1667      0.7116          20.0974
46      1.25         0.125       0.6865          20.0875
47      1.25          0.1        0.6941          20.1584
48      1.25         0.0833      0.709           20.2628
49      1.5          0.5         0.8843          20.0715
50      1.5          0.25        0.7147          19.9234
51      1.5         0.1667       0.7284          20.0081
52      1.5         0.125        0.6917          19.9894
53      1.5          0.1         0.6542          20.0482
54      1.5         0.0833       0.6941          20.1584
55      1.75          0.5        0.83            19.9641
56      1.75          0.25       0.7365          19.8463
57      1.75         0.1667      0.7138          19.8478
58      1.75         0.125       0.7             19.8945
59      1.75          0.1        0.6541          19.9609
60      1.75         0.0833      0.6978          20.0867
61      2.0          0.5         0.8699          19.9022
62      2.0          0.25        0.7709          19.7746
63      2.0         0.1667       0.7359          19.7581
64      2.0         0.125        0.7408          19.8321
65      2.0          0.1         0.6565          19.8761
66      2.0         0.0833       0.6696          20.0055
67  ======================================================
68
69  ======================================================
70            CONJUNTO OPTIMO DE SOLUCIONES
71  ------------------------------------------------------
72  CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
73  ======================================================
74      1.75         0.1667      0.7138          19.8478
75      1.75         0.1         0.6541          19.9609
76      2.0          0.1667      0.7359          19.7581
77      2.0          0.1         0.6565          19.8761
78  ======================================================
```

Figura B.2.: Consumo energético considerando el modelo con almacenamiento distribuido para el sistema de distribución de 13 barras.



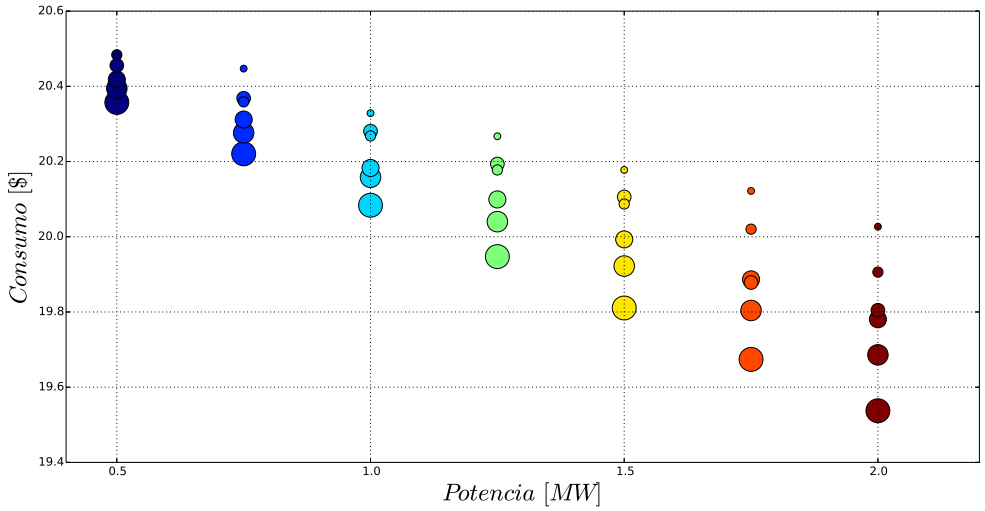
a, muestra el conjunto de soluciones agrupados por color haciendo referencia a la capacidad de almacenamiento y por área de las circunferencias haciendo referencia a la potencia.

b, muestra el conjunto de soluciones agrupados por frentes de pareto.

```
1  ======================================
2  CARACTERISTICAS DEL SISTEMA
3  ======================================
4  SISTEMA DE DISTRIBUCION
5  Caso de prueba              case_13
6  Numero de barras            13
7  Numero de lineas            12
8  Numero de generadores       1
9  Potencia base               1 [MVA]
10 Tension base                10 [kV]
11 --------------------------------------
12 SISTEMA DE ALMACENAMIENTO
13 Ubicacion                   [ 1 ]
14 --------------------------------------
15 FUENTE RENOVABLE
16 Ubicacion                   [ 9, 13]
17 Potencia                    [ 0.5  0.5 ] [MW]
18 ======================================
19
20 ======================================================
21                 MODELO CENTRALIZADO
22 ------------------------------------------------------
23 CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
24 ======================================================
25     0.5          0.5         0.7134          20.3571
26     0.5          0.25        0.7134          20.3943
27     0.5         0.1667       0.7134          20.4179
28     0.5         0.125        0.7134          20.4555
29     0.5          0.1         0.7134          20.4838
30     0.5         0.0833       0.7134          20.4795
31     0.75           0.5        0.7134          20.2205
32     0.75           0.25       0.7134          20.2762
33     0.75         0.1667       0.7134          20.3117
34     0.75         0.125        0.7134          20.3681
35     0.75           0.1        0.7134          20.3587
36     0.75         0.0833       0.7134          20.4471
37     1.0          0.5          0.7134          20.0838
38     1.0          0.25         0.7134          20.1582
39     1.0         0.1667        0.7134          20.1829
40     1.0         0.125         0.7134          20.2807
41     1.0          0.1          0.7134          20.2682
42     1.0         0.0833        0.7134          20.3286
43     1.25           0.5        0.7134          19.9472
44     1.25           0.25       0.7134          20.0401
45     1.25         0.1667       0.7134          20.0992
46     1.25         0.125        0.7134          20.1933
47     1.25           0.1        0.7134          20.1777
48     1.25         0.0833       0.7134          20.2672
49     1.5          0.5          0.7134          19.8106
50     1.5          0.25         0.7134          19.9221
51     1.5         0.1667        0.7134          19.993
52     1.5         0.125         0.7134          20.1059
53     1.5          0.1          0.7134          20.0871
54     1.5         0.0833        0.7134          20.1777
```

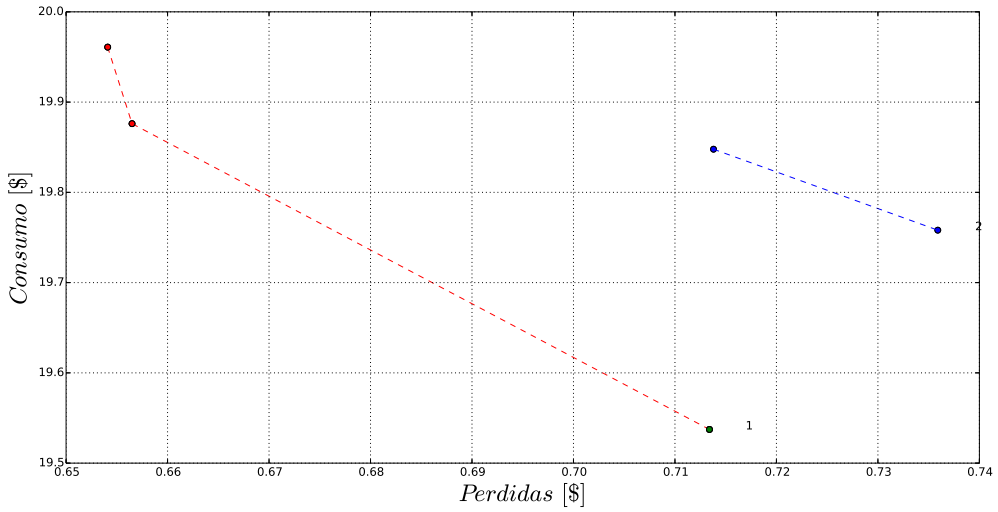
```
55      1.75          0.5         0.7134          19.6739
56      1.75          0.25        0.7134          19.804
57      1.75          0.1667      0.7134          19.8867
58      1.75          0.125       0.7134          19.8783
59      1.75          0.1         0.7134          20.0202
60      1.75          0.0833      0.7134          20.1219
61      2.0           0.5         0.7134          19.5373
62      2.0           0.25        0.7134          19.686
63      2.0           0.1667      0.7134          19.7805
64      2.0           0.125       0.7134          19.8046
65      2.0           0.1         0.7134          19.906
66      2.0           0.0833      0.7134          20.0268
67  ======================================================
68
69  ======================================================
70            CONJUNTO OPTIMO DE SOLUCIONES
71  ------------------------------------------------------
72  CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
73  ======================================================
74      2.0           0.5         0.7134          19.5373
75  ======================================================
```

Figura B.3.: Consumo energético considerando el modelo con almacenamiento centralizado, para el sistema de distribución de13 barras.



```
1  ======================================================
2          CONDICIONES INCIALES DEL SISTEMA
3  ======================================================
4  Perdidas              0.9852 [$]
5  Consumo              24.6417 [$]
6  ======================================================
7
8  ========================================================================
9              CONJUNTO TOTAL DE SOLUCIONES
10 ------------------------------------------------------------------------
11 CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]  MODELO  FRENTE
12 ========================================================================
13     1.75         0.1667      0.7138          19.8478         D        2
14     1.75         0.1         0.6541          19.9609         D        1
15     2.0          0.1667      0.7359          19.7581         D        2
16     2.0          0.1         0.6565          19.8761        D       1
17     2.0          0.5         0.7134          19.5373         C        1
18 ========================================================================
19
20 D,  modelo con almacenamiento distribuido.
21 C,  modelo con almacenamiento centralizado.
```

Figura B.4.: Conjunto total de soluciones para el sistema de distribución de 13 barras.



De acuerdo con la *Figura B.4*, dos soluciones del modelo con almacenamiento distribuido se encuentran en el frente número uno, en conjunto con la solución del modelo con almacenamiento centralizado. Por otro lado, la solución del modelo con almacenamiento centralizado domina sobre dos soluciones del modelo con almacenamiento distribuido.

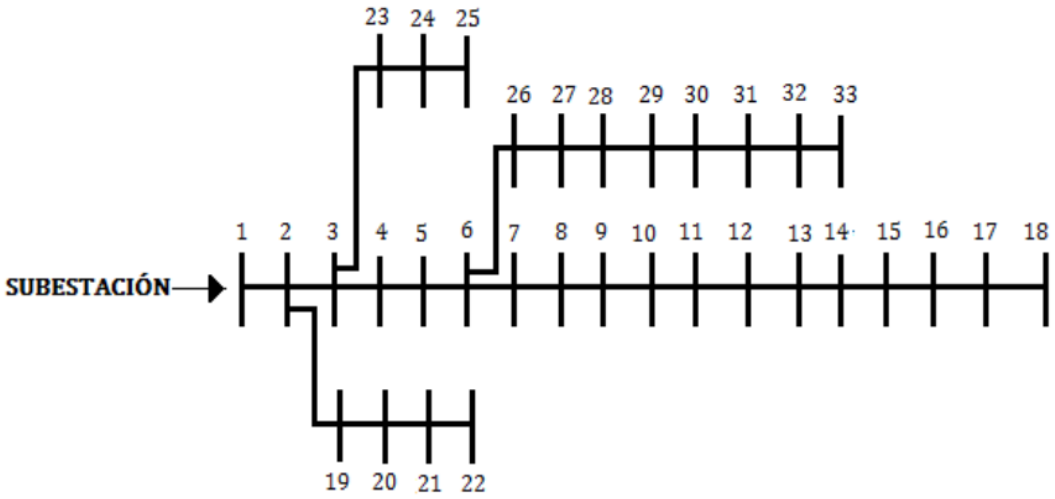
Adicionalmente, se observó que el modelo con almacenamiento centralizado logró un menor consumo energético con alta capacidad de almacenamiento y con alta potencia. De igual forma, el conjunto de soluciones óptimas para el modelo con almacenamiento distribuido se logró con

alta capacidad de almacenamiento pero con bajas potencias.

Finalmente, se establece que el modelo distribuido como el modelo centralizado de almacenamiento, presentan buenas soluciones, por lo tanto ambos modelos son aceptados.

B.2 Case 33

Figura B.5.: Diagrama unifilar para el sistema de distribución de 33 barras.

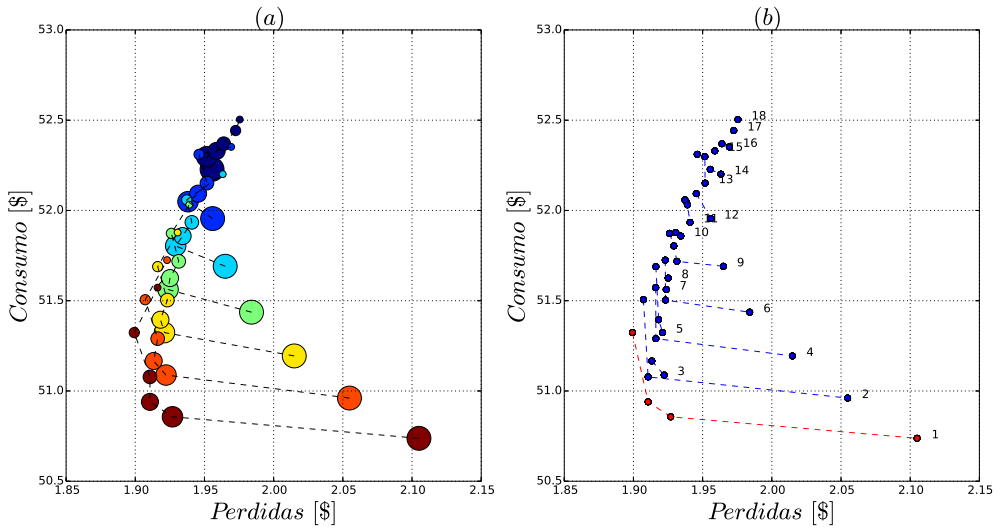


Tomado de: http://bibdigital.epn.edu.ec/bitstream/15000/8176/3/CD-5714.pdf

```
1  ======================================
2  CARACTERISTICAS DEL SISTEMA
3  ======================================
4  SISTEMA DE DISTRIBUCION
5  Caso de prueba              case_33
6  Numero de barras            33
7  Numero de lineas            32
8  Numero de generadores       1
9  Potencia base               100 [MVA]
10 Tension base                12.66 [kV]
11 --------------------------------------
12 SISTEMA DE ALMACENAMIENTO
13 Ubicacion                   [ 18, 22, 25, 33 ]
14 --------------------------------------
15 FUENTE RENOVABLE
16 Ubicacion                   [ 18, 22, 25, 33]
17 Potencia                    [ 0.25, 0.25, 0.25, 0.25 ] [MW]
18 ======================================
19
20 ====================================================
21                 MODELO DISTRIBUIDO
22 ----------------------------------------------------
23 CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
24 ====================================================
25     0.5          0.5         1.9555          52.2278
```

```
26      0.5          0.25        1.9516          52.2981
27      0.5         0.1667       1.9588          52.3302
28      0.5         0.125        1.9639          52.3698
29      0.5          0.1         1.9725          52.4426
30      0.5         0.0833       1.9755          52.5037
31      0.75         0.5         1.9559          51.9549
32      0.75         0.25        1.9381          52.0486
33      0.75        0.1667       1.9454          52.0931
34      0.75        0.125        1.9518          52.1513
35      0.75         0.1         1.9461          52.3109
36      0.75        0.0833       1.9693          52.3522
37      1.0          0.5         1.965           51.6907
38      1.0          0.25        1.9292          51.8036
39      1.0         0.1667       1.9342          51.8582
40      1.0         0.125        1.9409          51.934
41      1.0          0.1         1.9372          52.0586
42      1.0         0.0833       1.9632          52.2009
43      1.25         0.5         1.984           51.4364
44      1.25         0.25        1.9237          51.562
45      1.25        0.1667       1.9251          51.6253
46      1.25        0.125        1.9314          51.7181
47      1.25         0.1         1.9261          51.8732
48      1.25        0.0833       1.939           52.0314
49      1.5          0.5         2.0149          51.1941
50      1.5          0.25        1.9211          51.3233
51      1.5         0.1667       1.9182          51.3947
52      1.5         0.125        1.9232          51.5034
53      1.5          0.1         1.9162          51.6889
54      1.5         0.0833       1.9306          51.8777
55      1.75         0.5         2.0548          50.9607
56      1.75         0.25        1.9222          51.0882
57      1.75        0.1667       1.9133          51.1661
58      1.75        0.125        1.9162           51.29
59      1.75         0.1         1.9072          51.5056
60      1.75        0.0833       1.923           51.7248
61      2.0          0.5         2.105           50.7377
62      2.0          0.25        1.9269          50.8569
63      2.0         0.1667       1.9106          50.9396
64      2.0         0.125        1.9106          51.0779
65      2.0          0.1         1.8993          51.3234
66      2.0         0.0833       1.9161          51.5726
67  ======================================================
68
69  ======================================================
70            CONJUNTO OPTIMO DE SOLUCIONES
71  ------------------------------------------------------
72  CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
73  ======================================================
74      2.0          0.1         1.8993          51.3234
75      2.0          0.125       1.9106          51.0779
76      2.0          0.25        1.9269          50.8569
77      2.0          0.5         2.105           50.7377
78  ======================================================
```

Figura B.6.: Consumo energético considerando el modelo con almacenamiento distribuido, para el sistema de distribución de 33 barras.



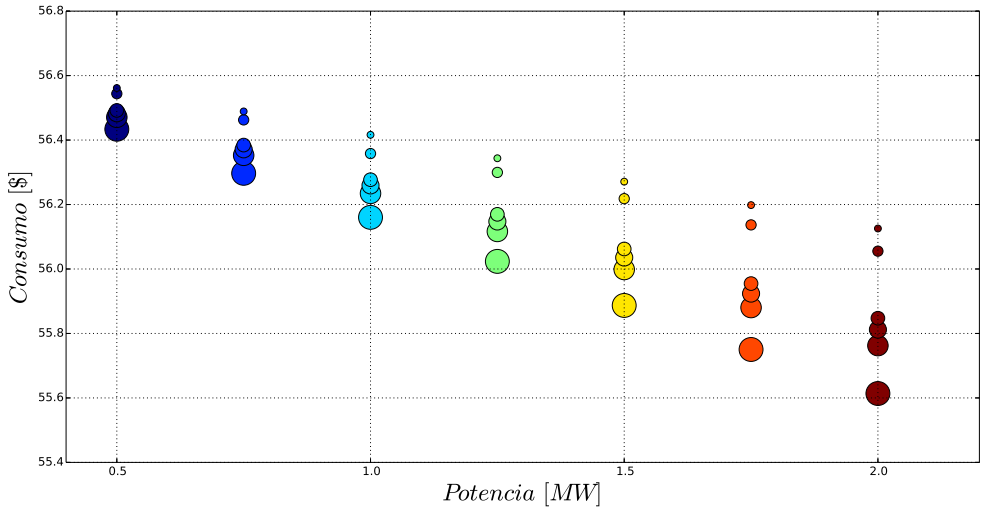
a, muestra el conjunto de soluciones agrupados por color haciendo referencia a la capacidad de almacenamiento y por área de las circunferencias haciendo referencia a la potencia.

b, muestra el conjunto de soluciones agrupados por frentes de pareto.

```
=====================================
CARACTERISTICAS DEL SISTEMA
=====================================
SISTEMA DE DISTRIBUCION
Caso de prueba              case_33
Numero de barras            33
Numero de lineas            32
Numero de generadores       1
Potencia base               100 [MVA]
Tension base                12.66 [kV]
-------------------------------------
SISTEMA DE ALMACENAMIENTO
Ubicacion                   [ 1 ]
-------------------------------------
FUENTE RENOVABLE
Ubicacion                   [ 18, 22, 25, 33]
Potencia                    [ 0.25, 0.25, 0.25, 0.25 ] [MW]
=====================================

=====================================================
                MODELO CENTRALIZADO
-----------------------------------------------------
CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
=====================================================
    0.5          0.5        2.1484         56.4335
    0.5          0.25       2.1484         56.4706
    0.5         0.1667      2.1484         56.483
    0.5         0.125       2.1484         56.4919
    0.5          0.1        2.1484         56.5439
    0.5         0.0833      2.1484         56.5614
    0.75          0.5       2.1484         56.2968
    0.75          0.25      2.1484         56.3526
    0.75         0.1667     2.1484         56.3712
    0.75         0.125      2.1484         56.3844
    0.75          0.1       2.1484         56.4625
    0.75         0.0833     2.1484         56.4888
    1.0          0.5        2.1484         56.1602
    1.0          0.25       2.1484         56.2345
    1.0         0.1667      2.1484         56.2593
    1.0         0.125       2.1484         56.277
    1.0          0.1        2.1484         56.3581
    1.0         0.0833      2.1484         56.4162
    1.25          0.5       2.1484         56.0236
    1.25          0.25      2.1484         56.1165
    1.25         0.1667     2.1484         56.1474
    1.25         0.125      2.1484         56.1696
    1.25          0.1       2.1484         56.2996
    1.25         0.0833     2.1484         56.3435
    1.5          0.5        2.1484         55.8869
    1.5          0.25       2.1484         55.9984
    1.5         0.1667      2.1484         56.0356
    1.5         0.125       2.1484         56.0621
    1.5          0.1        2.1484         56.2182
    1.5         0.0833      2.1484         56.2709
```

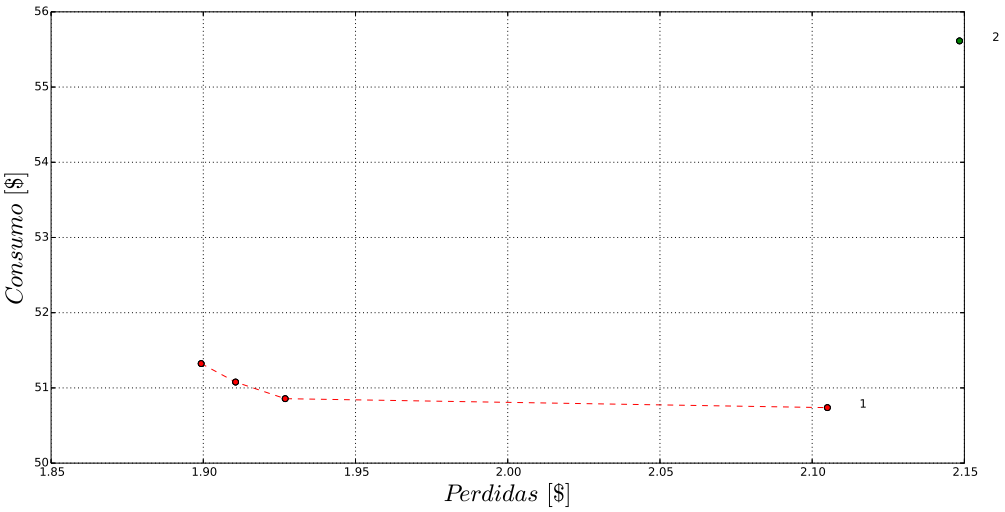
```
55      1.75          0.5         2.1484          55.7503
56      1.75          0.25        2.1484          55.8804
57      1.75          0.1667      2.1484          55.9237
58      1.75          0.125       2.1484          55.9547
59      1.75          0.1         2.1484          56.1368
60      1.75          0.0833      2.1484          56.1982
61      2.0           0.5         2.1484          55.6137
62      2.0           0.25        2.1484          55.7623
63      2.0           0.1667      2.1484          55.8119
64      2.0           0.125       2.1484          55.8473
65      2.0           0.1         2.1484          56.0553
66      2.0           0.0833      2.1484          56.1256
67  ======================================================
68
69  ======================================================
70            CONJUNTO OPTIMO DE SOLUCIONES
71  ------------------------------------------------------
72  CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
73  ======================================================
74      2.0           0.5         2.1484          55.6137
75  ======================================================
```

Figura B.7.: Consumo energético considerando el modelo con almacenamiento centralizado, para el sistema de distribución de 33 barras.



```
1  ======================================================
2          CONDICIONES INCIALES DEL SISTEMA
3  ======================================================
4  Perdidas               2.3524 [$]
5  Consumo               60.6502 [$]
6  ======================================================
7
8  ======================================================================
9              CONJUNTO TOTAL DE SOLUCIONES
10 ----------------------------------------------------------------------
11 CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]  MODELO  FRENTE
12 ======================================================================
13     2.0          0.1         1.8993          51.3234         D         1
14     2.0        0.125         1.9106          51.0779        D        1
15     2.0         0.25         1.9269          50.8569        D        1
16     2.0          0.5         2.105           50.7377        D        1
17     2.0          0.5         2.1484          55.6137        C        2
18 ======================================================================
19
20 D,  modelo con almacenamiento distribuido.
21 C,  modelo con almacenamiento centralizado.
```

Figura B.8.: Conjunto total de soluciones para el sistema de distribución de 33 barras.



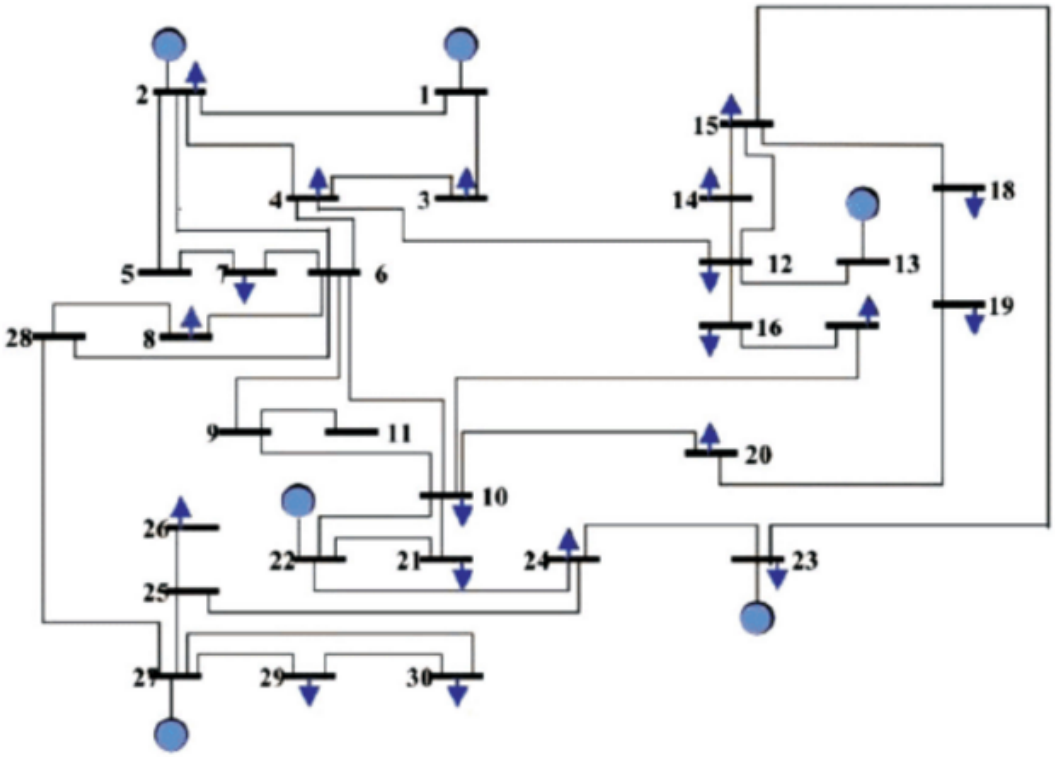
De acuerdo con la *Figura* B.8, el conjunto de soluciones del modelo con almacenamiento distribuido domina por completo la mejor solución del modelo con almacenamiento centralizado.

Adicionalmente, se observó que el modelo con almacenamiento centralizado logró el menor consumo energético con alta capacidad de almacenamiento y con alta potencia. De igual forma, el conjunto de soluciones óptimas para el modelo con almacenamiento distribuido se logró con alta capacidad de almacenamiento pero con bajas potencias.

Finalmente, se establece que el modelo con almacenamiento distribuido presenta mejores soluciones que el modelo con almacenamiento centralizado.

B.3 Case 30

Figura B.9.: Diagrama unifilar para el sistema de distribución de 30 barras.



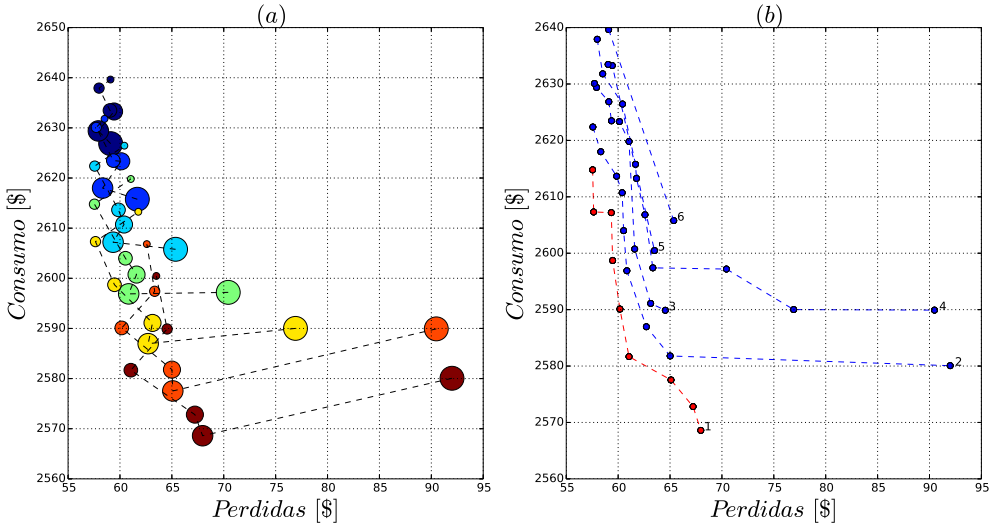
Tomado de: http://www.scielo.org.ve/scielo.php?script=sci_arttext&pid=S0798-40652010000100007

```
1  ======================================
2  CARACTERISTICAS DEL SISTEMA
3  ======================================
4  SISTEMA DE DISTRIBUCION
5  Caso de prueba              case_30
6  Numero de barras            30
7  Numero de lineas            41
8  Numero de generadores       6
9  Potencia base               100 [MVA]
10 Tension base                135 [kV]
11 ---------------------------------------
12 SISTEMA DE ALMACENAMIENTO
13 Ubicacion                   [ 18.  19.  20.  23.  26.  30.]
14 ---------------------------------------
15 FUENTE RENOVABLE
16 Ubicacion                   [ 18.  19.  20.  23.  26.  30.]
17 Potencia                    [ 16.6667  16.6667  16.6667  16.6667  16.6667  16.6667] [MW]
```

```
18 ======================================
19
20 ======================================================
21                 MODELO DISTRIBUIDO
22 ------------------------------------------------------
23 CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
24 ======================================================
25     50          0.5         59.1018        2626.854
26     50          0.25        57.9185        2629.3871
27     50         0.1667       59.4323        2633.2647
28     50         0.125        59.0359        2633.4718
29     50          0.1         57.9844        2637.9279
30     50         0.0833       59.087         2639.6379
31     75          0.5         61.673         2615.7618
32     75          0.25        58.3271        2617.9905
33     75         0.1667       60.1247        2623.3337
34     75         0.125        59.3592        2623.4735
35     75          0.1         57.7158        2630.0914
36     75         0.0833       58.5134        2631.8002
37     100         0.5         65.3574        2605.7828
38     100         0.25        59.3382        2607.1964
39     100        0.1667       60.3886        2610.7244
40     100        0.125        59.8568        2613.6495
41     100         0.1         57.5666        2622.3744
42     100        0.0833       60.423         2626.4457
43     125         0.5         70.4327        2597.1946
44     125         0.25        60.8402        2596.8932
45     125        0.1667       61.5947        2600.7447
46     125        0.125        60.5282        2603.9993
47     125         0.1         57.5419        2614.7819
48     125        0.0833       61.0462        2619.8048
49     150         0.5         76.9113        2590.0098
50     150         0.25        62.7171        2586.9649
51     150        0.1667       63.1351        2591.0993
52     150        0.125        59.4715        2598.7081
53     150         0.1         57.6414        2607.3136
54     150        0.0833       61.7672        2613.2617
55     175         0.5         90.4791        2589.9142
56     175         0.25        65.0909        2577.5334
57     175        0.1667       65.0095        2581.7878
58     175        0.125        60.1728        2590.1024
59     175         0.1         63.3467        2597.4073
60     175        0.0833       62.586         2606.8164
61     200         0.5         91.9925        2580.0623
62     200         0.25        67.9628        2568.6001
63     200        0.1667       67.219         2572.8115
64     200        0.125        61.0473        2581.6698
65     200         0.1         64.5444        2589.8881
66     200        0.0833       63.503         2600.4693
67 ======================================================
68
69 ======================================================
70           CONJUNTO OPTIMO DE SOLUCIONES
71 ------------------------------------------------------
```

```
CAPACIDAD[MW] POTENCIA [pu] PERDIDAS [$]  CONSUMO [$]
=====================================================
     125          0.1         57.5419        2614.7819
     150          0.1         57.6414        2607.3136
     100          0.25        59.3382        2607.1964
     150          0.125       59.4715        2598.7081
     175          0.125       60.1728        2590.1024
     200          0.125       61.0473        2581.6698
     175          0.25        65.0909        2577.5334
     200          0.1667      67.219         2572.8115
     200          0.25        67.9628        2568.6001
=====================================================
```

Figura B.10.: Consumo energético considerando el modelo con almacenamiento distribuido, para el sistema de distribución de 30 barras.



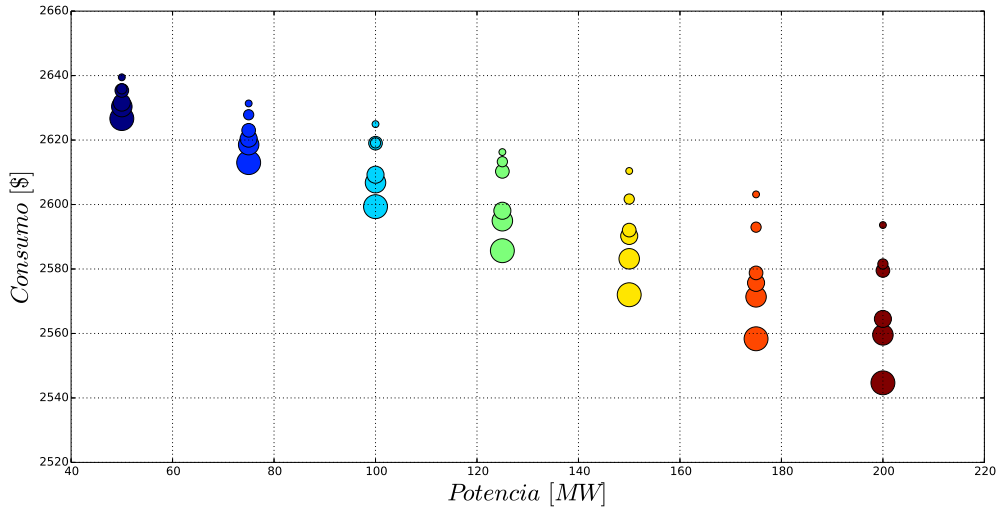
a, muestra el conjunto de soluciones agrupados por color haciendo referencia a la capacidad de almacenamiento y por área de las circunferencias haciendo referencia a la potencia.

b, muestra el conjunto de soluciones agrupados por frentes de pareto.

```
=====================================
CARACTERISTICAS DEL SISTEMA
=====================================
SISTEMA DE DISTRIBUCION
Caso de prueba              case_30
Numero de barras            30
Numero de lineas            41
Numero de generadores       6
Potencia base               100 [MVA]
Tension base                135 [kV]
-------------------------------------
SISTEMA DE ALMACENAMIENTO
Ubicacion                   [ 1]
-------------------------------------
FUENTE RENOVABLE
Ubicacion                   [ 18.  19.  20.  23.  26.  30.]
Potencia                    [ 16.6667  16.6667  16.6667  16.6667  16.6667  16.6667] [MW]
=====================================

=====================================================
                MODELO CENTRALIZADO
=====================================================
CAPACIDAD[MW] POTENCIA [%] PERDIDAS [$]  CONSUMO [$]
=====================================================
    50          0.5         58.9144        2626.6666
    50          0.25        58.9144        2630.383
    50         0.1667       58.9144        2631.6219
    50         0.125        58.9144        2635.3793
    50          0.1         58.9144        2635.8847
    50         0.0833       58.9144        2639.4653
    75          0.5         58.9144        2613.0032
    75          0.25        58.9144        2618.5778
    75         0.1667       58.9144        2620.436
    75         0.125        58.9144        2623.0287
    75          0.1         58.9144        2627.8427
    75         0.0833       58.9144        2631.3575
    100         0.5         58.9144        2599.3398
    100         0.25        58.9144        2606.7726
    100        0.1667       58.9144        2609.2502
    100        0.125        58.9144        2619.0271
    100         0.1         58.9144        2619.1258
    100        0.0833       58.9144        2624.9371
    125         0.5         58.9144        2585.6763
    125         0.25        58.9144        2594.9674
    125        0.1667       58.9144        2598.0644
    125        0.125        58.9144        2610.2855
    125         0.1         58.9144        2613.2816
    125        0.0833       58.9144        2616.2668
    150         0.5         58.9144        2572.0129
    150         0.25        58.9144        2583.1621
    150         .1667       58.9144        2590.2533
    150         .125        58.9144        2592.0638
    150         0.1         58.9144        2601.6919
    150         .0833       58.9144        2610.4089
```

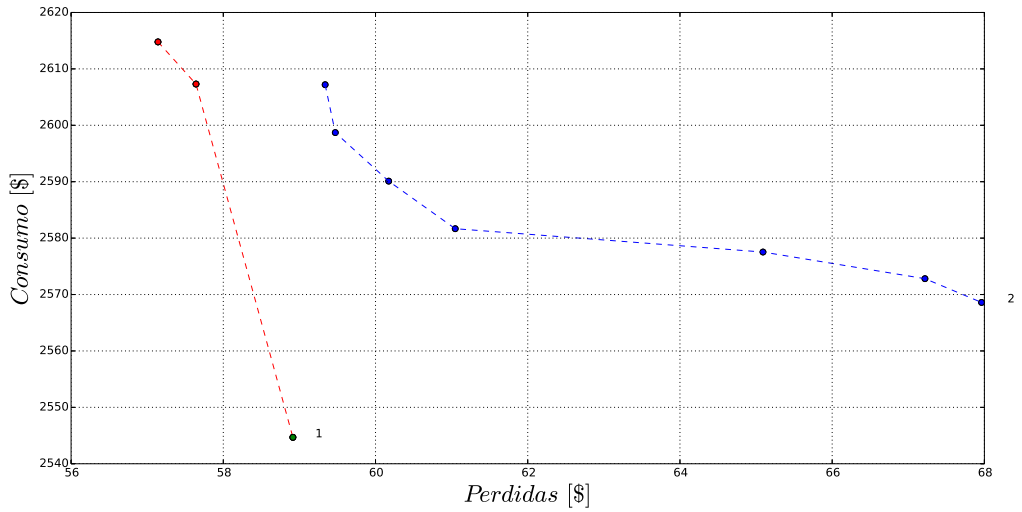
```
     175          0.5          58.9144         2558.3494
     175          0.25         58.9144         2571.3569
     175         0.1667        58.9144         2575.6927
     175         0.125         58.9144         2578.7893
     175          0.1          58.9144         2592.975
     175         0.0833        58.9144         2603.1448
     200          0.5          58.9144         2544.686
     200          0.25         58.9144         2559.5517
     200         0.1667        58.9144         2564.5069
     200         0.125         58.9144         2579.5368
     200          0.1          58.9144         2581.5583
     200         0.0833        58.9144         2593.6308
=========================================================

=========================================================
           CONJUNTO OPTIMO DE SOLUCIONES
---------------------------------------------------------
CAPACIDAD[MW] POTENCIA [%] PERDIDAS [$]  CONSUMO [$]
=========================================================
     200           0.5          58.9144         2544.686
=========================================================
```

Figura B.11.: Consumo energético considerando el modelo con almacenamiento centralizado, para el sistema de distribución de 30 barras



```
1  ======================================================
2          CONDICIONES INCIALES DEL SISTEMA
3  ======================================================
4  Perdidas              59.3666 [$]
5  Consumo             3028.3953 [$]
6  ======================================================
7
8  ========================================================================
9              CONJUNTO TOTAL DE SOLUCIONES
10 ------------------------------------------------------------------------
11 CAPACIDAD[MW] POTENCIA [%] PERDIDAS [$]  CONSUMO [$]  MODELO  FRENTE
12 ========================================================================
13     125         0.1         57.1419        2614.7819         D       1
14     150         0.1         57.6414        2607.3136        D       1
15     100         0.25        59.3382        2607.1964        D      2
16     150         0.125       59.4715        2598.7081        D      2
17     175         0.125       60.1728        2590.1024        D      2
18     200         0.125       61.0473        2581.6698        D      2
19     175         0.25        65.0909        2577.5334        D      2
20     200         0.1667      67.219         2572.8115        D      2
21     200         0.25        67.9628        2568.6001        D      2
22     200         0.5         58.9144        2544.686         C       1
23 ========================================================================
24
25 D,  modelo con almacenamiento distribuido.
26 C,  modelo con almacenamiento centralizado.
```

Figura B.12.: Consumo energético considerando el modelo con almacenamiento centralizado, para el sistema de distribución de 30 barras.



De acuerdo con la *Figura* B.12, dos soluciones del modelo con almacenamiento distribuido se encuentra en el frente numero uno, en conjunto con la solución del modelo con almacenamiento centralizado. Por otro lado, la solución del modelo con almacenamiento centralizado domina

sobre siete soluciones del modelo con almacenamiento distribuido.

Adicionalmente, se observó que el modelo con almacenamiento centralizado logró un menor consumo energético con alta capacidad de almacenamiento y con alta potencia. De igual forma, el conjunto de soluciones óptimas para el modelo con almacenamiento distribuido se logró con alta capacidad de almacenamiento pero con bajas potencias.

Finalmente, se establece que el modelo distribuido como el modelo centralizado de almacenamiento, presentan buenas soluciones, por lo tanto ambos modelos son aceptados.