ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

RECONFIGURACIÓN DE REDES DE DISTRIBUCIÓN DE ENERGÍA ELÉCTRICA BASADA EN OPTIMIZACIÓN DE MAPEO MEDIA-VARIANZA

TESIS PREVIA A LA OBTENCIÓN DEL GRADO DE MAGÍSTER EN EFICIENCIA ENERGÉTICA

ING. ROSANNA XIMENA LOOR TORO
roxiloort@hotmail.com

DIRECTOR: ING. JOSÉ LUIS RUEDA TORRES, PhD.
J.L.RuedaTorres@tudelft.nl

CO-DIRECTOR: ING. LENIN RODIA UBIDIA GUERRA, MSc.
ubidialenin@yahoo.com

Quito, julio 2014

DECLARACIÓN

Yo Rosanna Ximena Loor Toro, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

ING. ROSANNA LOOR

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por ROSANNA XIMENA LOOR TORO bajo nuestra supervisión.

ING. JOSÉ RUEDA, PhD.
DIRECTOR DE PROYECTO

ING. LENIN UBIDIA, MSc.
CO-DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Agradezco a mi Director de tesis, Ing. José Rueda, por su amistad, competencia y ayuda para resolver las dudas en el desarrollo de la investigación; a mi Co-Director, Ing. Lenin Ubidia por su apoyo y decisión para ser parte de este proyecto, profesionales que bajo su supervisión hicieron posible la culminación de este trabajo.

DEDICATORIA

Para ti mamá,
Te Amo

CONTENIDO

ANEXOS

LISTAD0 DE TABLAS

LISTADO DE FIGURAS

RESUMEN

Esta investigación propone una metodología de solución basada en el algoritmo de optimización de mapeo media – varianza (MVMO), para resolver el problema de reconfiguración de redes radiales de sistemas de distribución de energía eléctrica (SDR), cuyo objetivo es determinar la combinación óptima de apertura-cierre de seccionadores en los ramales de la red que conlleva simultáneamente a un mínimo valor pérdidas de potencia activa, un perfil mejorado de voltaje, y el cumplimiento de las restricciones operativas de voltaje y topología radial. Además, de la revisión de los principios del MVMO, se evalúa su extensión, en base a preceptos de inteligencia de enjambre, a una variante denominada MVMO^S, a fin de mejorar la efectividad del procedimiento de búsqueda del valor óptimo.

La aplicación propuesta involucra la solución de un problema combinatorial con un espacio de búsqueda no-lineal, no convexo, y multimodal. Se implementa el algoritmo de MVMO y su variante MVMO^S en el programa computacional MATLAB, para resolver el problema propuesto en un sistema prototipo ampliamente divulgado en la bibliografía especializada y que ha sido utilizado por varios autores para verificar el desempeño de otros algoritmos metaheurísticos.

Los resultados obtenidos con MVMO y MVMO^S, fueron comparados con otras metodologías presentadas en el estado del arte: Algoritmo Heurístico de Goswami-Basu HA, Algoritmo Genético GA, Algoritmo Genético Redefinido RGA, Búsqueda Tabú Mejorada ITS, Algoritmo Evolutivo EA y Búsqueda de Armonía HSA.

De los resultados obtenidos, el MVMO mostró mejor desempeño frente a las metodologías HA y GA, mientras que con RGA, EA, y MVMO^S se consiguieron valores similares en pérdidas de potencia activa y voltajes, pero en tiempo de procesamiento MVMO presentó mejor rendimiento. El HSA e ITS obtuvieron resultados ligeramente mejores que MVMO en cuanto a pérdidas y tiempos de procesamiento, no así en voltajes. No obstante, las características de convergencia de MVMO y MVMO^S, tuvieron una notable mejor velocidad de convergencia para obtener el valor óptimo frente a las metodologías indicadas, además de una evaluación estadística menor que los algoritmos RGA, GA, ITS y HSA. Para el caso de

estudio propuesto es suficiente aplicar MVMO, puesto que con una sola partícula se alcanzaron valores de pérdidas iguales que MVMO^{S}, el cual, se simuló con varias partículas, lo que ocasionó que tenga un tiempo de procesamiento ligeramente mayor que MVMO, dada la cantidad de cálculos adicionales que debe realizar.

PRESENTACIÓN

OBJETIVOS Y ALCANCE

Objetivo General:

- Reconfigurar redes de distribución de energía eléctrica, por medio del algoritmo de Optimización Mapeo Media – Varianza (MVMO[1] por el acrónimo en inglés).

Objetivos Específicos:

- Reducir las pérdidas totales de potencia activa y mejorar el perfil de voltaje de redes eléctricas de distribución, aplicando el MVMO^S.
- Contrastar y validar los resultados obtenidos vía MVMO^S, con aquellos reportados en publicaciones de revistas internacionales indexadas.
- Verificar cual método presenta mejor desempeño en relación al objetivo específico anterior.

ALCANCE:

- Se definirá una red prototipo y se modelará en Matlab en base a la herramienta MATPOWER[2].
- Se determinarán las configuraciones óptimas para la red prototipo, de tal forma que el estado de seccionadores (cerrado o abierto), sea tal, que se disminuirán las pérdidas totales de potencia activa, mientras se cumplen las restricciones operativas, de: voltaje del sistema, capacidad de corriente de los alimentadores, y estructura radial del sistema de distribución.

[1] Mean-Variance Mapping Optimization (MVMO)
[2] MATLAB POWER SYSTEM SIMULATION PACKAGE

- Se implementará la adaptación y aplicación MVMO en Matlab para resolver la tarea de optimización y se realizará un análisis comparativo con los resultados obtenidos con métodos metaheurísticos, reportados en revistas científicas internacionales.

DESARROLLO DE LA TESIS

Este trabajo de investigación consiste de seis capítulos, que se desarrollan de la siguiente manera:

En el capítulo 1, se realiza una revisión bibliográfica, con breves descripciones de las principales técnicas de optimización para reconfiguración de redes de distribución de energía eléctrica, aplicando métodos heurísticos y metaheurísticos. En el campo de las heurísticas, se explica de manera general los métodos propuestos por Baran y Wu, y Civanlar, los cuales están basados en el intercambio de ramas y también son descritas algunas metaheurísticas que han sido encontradas en la literatura especializada y que están basadas en modelos biológicos como GA, EA y PSO, aplicadas a la reconfiguración de redes.

En el capítulo 2, se realiza una breve introducción de los métodos utilizados para cálculos de Flujos de Potencia FP y se explica la técnica de FP desarrollada para este proyecto. Se escoge una red prototipo y se evalúa el método de FP escogido; los resultados de las simulaciones se muestran, analizan y comparan con otros resultados obtenidos y presentados en artículos científicos. Además, se presentan los resultados obtenidos con la herramienta computacional MATPOWER para el caso de estudio aplicando la metodología Newton Raphson completo NR, donde esta metodología no converge para el sistema propuesto, dada su relación R/X mayor que 1.

En el capítulo 3, se realiza la formulación del problema de reconfiguración de redes para reducir pérdidas, es decir, se especifican la función objetivo, y las restricciones topológicas y operacionales asociadas al problema de optimización.

El capítulo 4, está dedicado a la metaheurística MVMO y su variante $MVMO^S$ utilizada en este trabajo de tesis para resolver el problema de reconfiguración de redes de distribución de energía eléctrica.

En el capítulo 5, se desarrolla la aplicación MVMO y MVMO^S para resolver el problema de reconfiguración de redes en el caso de estudio propuesto; para verificar su eficiencia y efectividad se comparan los resultados obtenidos en tiempo de procesamiento, valores de pérdidas y voltajes con MVMO y MVMO^S, con otros resultados alcanzados en revistas internacionales indexadas para el mismo sistema de distribución radial.

En el capítulo 6, se presentan las conclusiones, recomendaciones y perspectivas de trabajos futuros a realizarse.

Los anexos, poseen los datos de la red prototipo, resumen de típicos valores utilizados para inicializar los parámetros de MVMO y MVMO^S, resultados obtenidos, y códigos **.m** implementados en Matlab de la metodología propuesta.

CAPÍTULO 1

1 INTRODUCCIÓN

1.1 LEVANTAMIENTO BIBLIOGRÁFICO DE TÉCNICAS DE RECONFIGURACIÓN DE REDES DE DISTRIBUCIÓN DE ENERGÍA ELÉCTRICA

En sistemas de distribución primaria, los seccionadores[3] son usados para protección, aislar una falla, y para maniobrar la configuración reconfigurando la red. En la **Figura 1-1**, se muestra un diagrama esquemático de un circuito primario simplificado de un sistema de distribución, con sus seccionadores.

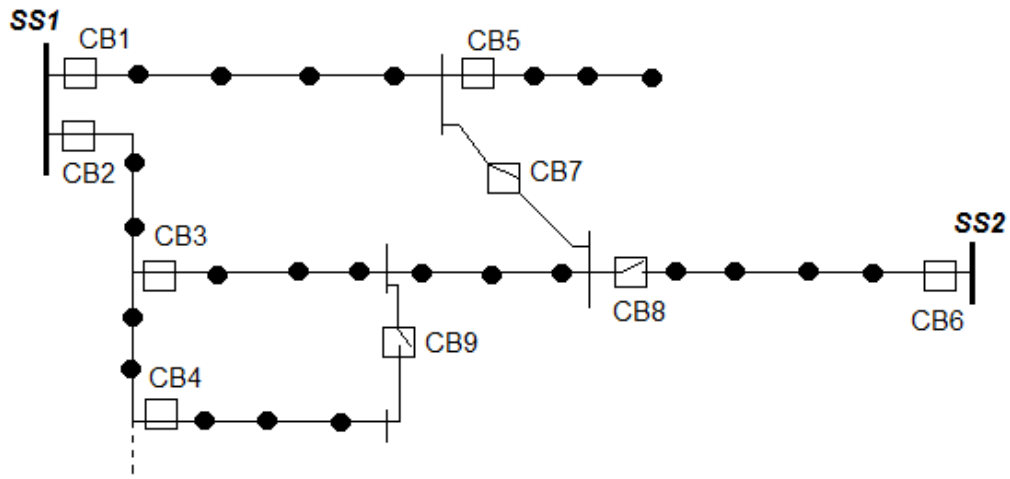


Figura 1-1: Diagrama esquemático de un circuito primario de un sistema de distribución [1]

En la **Figura 1-1**, los puntos de carga, transformadores de distribución desconectados del circuito primario, están marcados por puntos "●". En la figura, también se muestran dos tipos de seccionadores en el sistema: Seccionadores Normalmente Cerrados **SNC** conectando las líneas (CB1 – CB5), y Seccionadores Normalmente Abiertos **SNA** en las ramas de conexión[4] que conectan otros dos alimentadores primarios (CB7), o dos subestaciones (CB8), o laterales tipo-lazo (CB9).

[3] Sectionalizing switches-en idioma inglés.
[4] Tie-lines, en idioma inglés.

Los sistemas de distribución son normalmente operados como redes radiales; sin embargo, la configuración es modificada durante la operación, cambiando el estado de algunos seccionadores. Por ejemplo, en la **Figura 1-1**, los seccionadores CB7 y CB8 pueden ser cerrados y CB3 y CB6 pueden ser abiertos para transferir carga desde un primario a otro.

Un problema de operación importante en el manejo de configuraciones, es la reconfiguración de redes. Como las condiciones de operación cambian, la red es reconfigurada para dos propósitos: (i) reducir las pérdidas de potencia del sistema, (ii) para aliviar las sobrecargas en la red. Otra operación de manejo de configuración envuelve la restauración del servicio a todos los clientes como sea posible durante un estado de restauración después de una falla. Este problema es llamado servicio de restauración y puede ser tratado como un problema especial de balance de cargas. La reconfiguración para reducción de pérdidas puede también ser usada en estudios de planeamiento con una interpretación diferente; es decir, para mejorar la eficiencia en la operación y la calidad del servicio [1].

Minimizar pérdidas en sistemas de distribución ha ganado gran atención debido al alto costo de energía eléctrica y adicionalmente, gran parte de la investigación actual sobre automatización en distribución se ha centrado en el problema de configuración para minimizar pérdidas [2]. Hay algunas alternativas disponibles para reducir pérdidas en el nivel de distribución: reconfiguración, instalación de capacitores [3], balance de cargas, e introducción de niveles de voltajes altos. Esta investigación se centra en la alternativa de reconfiguración.

La reconfiguración de redes es una herramienta de optimización importante en sistemas de automatización de redes de distribución. Con un sistema de predicción de cargas y un sistema de monitoreo remoto, la estructura de la red puede ser reconfigurada en tiempo real, cambiando el estado de cerrado/abierto de algunos seccionadores que operan normalmente en estado abierto o cerrado, para reducir las pérdidas de potencia, equilibrar sobrecargas, mejorar la confiabilidad de suministro de potencia y, hasta cierto punto, incrementar la capacidad de suministro de potencia del sistema de distribución [4].

El problema de reconfiguración de redes es un problema de optimización combinatoria[5] complicado, es difícil obtener una configuración óptima en sistemas de distribución de gran escala en un tiempo de cálculo factible [4]. Este problema tiene una formulación matemática compleja, en que la exigencia de radialidad en la red es una dificultad adicional, y dada su naturaleza combinatoria es de difícil tratamiento [5].

El tema de la optimización combinatoria consiste de un conjunto de problemas que son fundamentales para las disciplinas de la ingeniera y ciencias de la computación. Investigadores en esta área ayudan en el desarrollo de técnicas eficientes para encontrar los valores mínimos o máximos de una función de variables muy independientes. Esta función usualmente llamada función costo o función objetivo, representa una medida cuantitativa de la "bondad" de algunos sistemas complejos. La función costo depende de la configuración detallada de algunas partes del sistema [6].

Varias publicaciones han tratado el tema de reconfiguración de redes de distribución. Los algoritmos meta-heurísticos, heurísticos y la aplicación de métodos de inteligencia artificial han contribuido para la resolución del problema de manera cada vez más eficientes [7].

1.1.1 MÉTODOS HEURÍSTICOS

Para resolver problemas de optimización en una cantidad razonable de tiempo, han sido propuestos métodos alternativos, los cuales no son capaces de determinar perfectamente soluciones exactas, pero si, aproximaciones de buena calidad para soluciones exactas. Estos métodos, llamados heurísticos, inicialmente están fundamentados en *"el conocimiento y la experiencia, y dirigidos para explorar el espacio de búsqueda en un camino particularmente conveniente"* [8].

Básicamente, una heurística es diseñada para proporcionar un mejor desempeño computacional en comparación con las técnicas de optimización convencionales, a costa de una menor precisión. Sin embargo, las "reglas de oro" que subyacen a una heurística suelen ser muy específicas para un problema en cuestión. Además, dado que las heurísticas son técnicas de

[5] Un problema de optimización combinatoria es la búsqueda de la mejor configuración. Fuente: Artículo 22-Optimización combinatoria. Miguel Sánchez García.

solución de problemas basados en la experiencia de las soluciones, utilizan representaciones de dominio específico, generalmente pueden ser exitosamente definidas solo para métodos de solución de problemas fundamentales, tales como los métodos de búsqueda [8].

Los métodos heurísticos fueron aplicados a la reconfiguración de redes a través de dos técnicas, principalmente.

Primero fue el método propuesto por (Merlin; Back, 1975) y posteriormente modificado por (Shirmohammadi; Hong, 1989), el cual consiste en cerrar todos los SNA en las ramas de conexión, obteniendo una red mallada. Se utilizó un flujo de carga óptimo, que comienza abriendo los SNC hasta obtener nuevamente una red radial. A través de este proceso es construido un padrón de flujo óptimo, para luego abrir la rama que presente menor flujo de corriente, obteniendo una mayor reducción de pérdidas. Este algoritmo es mejorado por Goswami y Basu 1992 (Goswami; Basu, 1992). La técnica también utiliza un padrón de flujo óptimo, pero apenas cierra un SNA en una rama de conexión formando un lazo a la vez y abre el mismo seccionador u otro seccionador que pertenece al lazo, de tal forma que la red este configurada de manera radial nuevamente [7].

El segundo método desarrollado por (Civanlar, 1988) [9], propone una metodología heurística para reconfiguración de redes estimando reducir el número de reconfiguraciones candidatas, utilizando como criterio una fórmula interesante y de uso simple que excluye opciones indeseadas de cierre de seccionadores sin la necesidad de efectuarse numerosos cálculos de flujos de potencia, reduciendo significativamente el esfuerzo computacional. Es decir, calcula la variación de pérdidas por la transferencia de un grupo de cargas de un alimentador a otro, aplicando la siguiente fórmula:

$$\Delta P = Re\left\{2\left(\sum_{i\in D} I_i\right)(E_m - E_n)^*\right\} + R_{loop}\left|\sum_{i\in D} I_i\right|^2 \qquad 1\text{-}1$$

Donde:

D Conjunto de nodos que son desconectadas del alimentador 2 y conectadas al alimentador 1;

m	Nodo de interconexión del alimentador 1, al cual, las cargas del alimentador 2 serán conectadas;
n	Nodos de interconexión del alimentador 2 que serán conectados al otro nodo a través de un SNA;
I_i	Corriente compleja de nodo en la barra i;
R_{loop}	Resistencia serie del camino conectando los dos nodos de las subestaciones de alimentador 1 y el alimentador 2, cerrando el seccionador específico.
E_m	Componente de $E = R_{bus} I_{bus}$ correspondiente al nodo m. R_{bus}, es la "*matriz de resistencia de barra*" del alimentador 1 antes de la transferencia de carga que es encontrada usando el nodo de la subestación como referencia. I_{bus}, es el vector de corrientes de nodo para el alimentador 1.
E_n	Análogo a E_m, sin embargo definido para el nodo n del alimentador 2.
Re{.}, *, |.|	Parte real, conjugado complejo y valor absoluto, respectivamente.

Se debe calcular E_m y E_n usando las corrientes I_i de los nodos del caso base antes de la transferencia de carga. ΔP representa una reducción (incremento) en kW cuando este es negativo (positivo).

La reducción de pérdidas puede ser obtenida apenas si existe una diferencia de voltaje entre los nodos de interconexión, esta observación puede ser usada como un criterio interesante para eliminar operaciones de seccionamiento no deseables durante el proceso de eliminación.

Posteriormente, el método descrito anteriormente es modificado por (Baran y Wu, 1989) [1], quienes proponen las *ecuaciones DisFlow simplificadas* para facilitar el cálculo de flujo de potencia y métodos heurísticos para encontrar un intercambio de seccionador resultando en la máxima reducción de pérdidas de potencia en un secionador de rama de interconexión asociado a un lazo de red.

Este método, basado en el intercambio de ramas, puede ser usado para crear árboles expandidos relevantes de un árbol expandido base. En general dado un árbol expandido $\boldsymbol{T_o}$, se asocian a un lazo con una rama abierta en la red considerando como si la rama fuera a

cerrarse. En la **Figura 1-2**, se muestra un lazo asociado con una rama abierta b. El intercambio de ramas crea un nuevo árbol cerrando una rama abierta, (rama b en la figura) y abriendo la rama cerrada en el lazo (rama m en la figura).

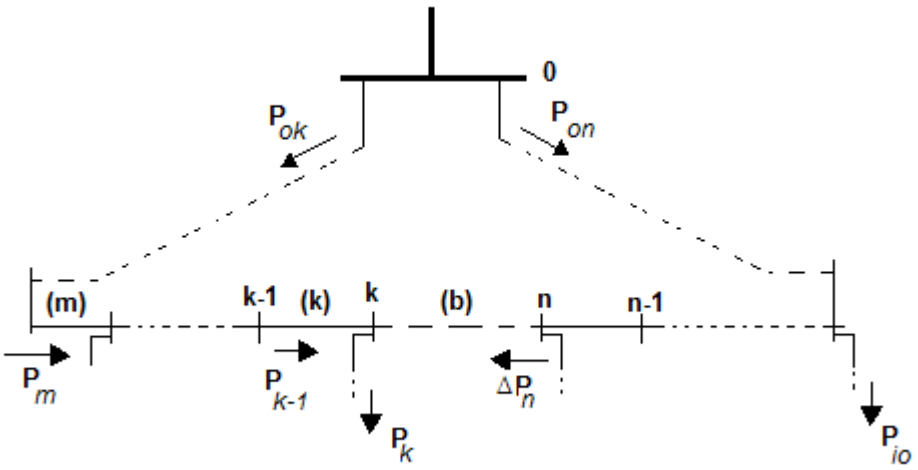


Figura 1-2: Lazo asociado con una rama *b* abierta [1]

La idea básica del esquema de búsqueda usando intercambios de ramas, es arrancar con un árbol (factible) y entonces crear nuevos árboles sucesivamente implementando un intercambio de rama a la vez. En cada nivel, el intercambio de ramas para ser implementado es escogido para ser el "mejor intercambio" (el intercambio que mejora la función objetivo el mejor sin violaciones de restricciones) entre todos los posibles árboles (hijos) que pueden ser generados del árbol (padre) titular expandido por intercambios de ramas.

Paso 1: Dado un árbol factible $\boldsymbol{T_o}$ (padre), correr un flujo de potencia para determinar el punto de operación.

Paso 2: Examinar todos los hijos de los padres, como sigue,
Para cada ramal abierto b

- encontrar un nuevo árbol candidato, T por
  -identificando el lazo
  -decidiendo en la rama, m a ser removido
- para el árbol candidado, T
  -calcular la reducción en el objetivo, $\Delta L\widetilde{P}_{bm}$ (reducción de pérdidas debido al intercambio de ramas)

Paso 3: Clasificar los hijos (examinación de árboles) usando $\boldsymbol{\Delta L\widetilde{P}_{bm}{}'s}$

Paso 4: Encontrar el árbol T* el cual tiene el mayor valor $\boldsymbol{\Delta L\widetilde{P}_{bm}} > 0$ y satisface las restricciones factibles.

Paso 5: Si el proceso es igual a T*, entonces escoge T* como $\boldsymbol{T_o}$ y va al paso 1; entonces termina el proceso.

Los algoritmos heurísticos son algoritmos de búsqueda codiciosos. Ellos son fáciles de implementar y alta eficiencia de búsqueda, pero generalmente no pueden converger a la solución del óptimo global y en sistemas de distribución de gran escala [4].

1.1.2 MÉTODOS METAHEURÍSTICOS

La metaheuristica es un conjunto de conceptos que normalmente utiliza métodos heurísticos aplicables a diversos tipos de problemas.

El término metaheurística fue propuesto por F.Glover (1986) en su trabajo *"Caminos futuros para programación entera y vínculos a la inteligencia artificial"*, para definir una heurística de alto nivel usada para guiar otras heurísticas para una mejor evolución en el espacio de búsqueda.

La mayoría de los algoritmos metaheurísticos son solo algoritmos aproximados, porque ellos no siempre encuentran la solución óptima global. Pero la característica más atractiva de una meta-heurística es que su aplicación no requiere de un conocimiento especial en el problema de optimización a ser resuelto [8].

Las metaheurísticas tienen las siguientes características principales:

- Objetivo: explorar eficientemente el espacio de búsqueda, de forma de encontrar soluciones (sub)óptimas.
- Son estrategias para guiar los procesos de búsqueda.
- Son algoritmos aproximados y no determinísticos.
- Incorporan mecanismos para evitar óptimos locales.
- No son específicas al problema que se intenta resolver, incorporan el conocimiento específico del problema o la experiencia (memoria) *"desviando"* la búsqueda.
- Amplio espectro: desde técnicas sencillas como la búsqueda local a técnicas complejas (procesos de aprendizaje, por ejemplo).

Las aplicaciones y la relevancia de las metaheurísticas han aumentado desde 1986, siendo que se encuentran trabajos usando metaheurísticas realizados en la década del setenta. Dentro de la metaheurísticas que se han utilizado para resolver el problema de reconfiguración de sistemas de distribución se encuentran los algoritmos genéticos (Nara et al., 1992) y (Zhu, 2002), algoritmo de recocido simulado SA[6] (Chiang; Jean-Jumeau,1990a, 1990b), colonia de hormigas, ACO[7] (Chang, 2008) [10].

Se mencionan otros algoritmos que han sido aplicados en los últimos años: búsqueda tabú TS[8] mejorado (Zhang;Fu,2007) [4], procedimiento de búsqueda adaptado aleatoriamente GRASP[9] (Correia Oliveira, 2011) [10], enjambre de partículas PSO[10] mejorado (Abdelaziz; Mekhamer; Mohammed; Badr; 2009) [2], búsqueda de armonía[11] HSA[12] (R.Srinivasa; S.Venkata; M.Ramalinga; A. Srinivasa; 2011) [11] .

A continuación se describen tres técnicas metaheuristicas, algoritmo evolutivo, algoritmos géneticos y enjambre de partícula (métodos basados en modelos biológicos), para resolver el problema de reconfiguración de redes.

Los Algoritmos Evolutivos (EA) [12], son métodos que explotan ideas de la evolución biológica, tal como la reproducción, mutación y recombinación para encontrar la solución. EA se refiere al ambiente biológico, aplicando el principio de supervivencia en un conjunto de soluciones potenciales para producir aproximaciones graduales al óptimo. Ellos consideran la idea de la evolución como una consecuencia de la reproducción, mutación y cruce. Un nuevo conjunto de aproximaciones es creado por selecciones individuales de acuerdo a su función objetivo, llamada ***función de aptitud*** para algoritmos evolucionarios, y creando conjuntamente usando operadores inspirados en los procesos genéticos. Este proceso conduce a la evolución de las populaciones de individuos que se adaptan mejor a su medio ambiente que sus antepasados.

[6] Acrónimo en inglés, Simulated Annealing.
[7] Acrónimo en inglés, Ant Colony Optimization.
[8] Acrónimo en inglés, Tabu Search
[9] Acrónimo en inglés, Greedy Randomized Adaptive Search Procedure.
[10] Acrónimo en inglés, Particle Swarm Optimization.
[11] Armonía musical.
[12] Acrónimo en inglés, Harmony Search Algorithm.

En [5], para la reconfiguración de sistemas de distribución radiales, se presenta un algoritmo de tipo evolutivo que considera un esquema de codificación en base decimal y efectúa los operadores de recombinación y mutación con la finalidad de mantener la radialidad de la red de distribución a través de las configuraciones que se van generando. Para efectuar los cálculos referentes a la función de aptitud y calcular el estado de las diferentes topologías de la red, se implementa un algoritmo de cálculo de flujo de carga rápido y eficiente (usa un método aproximado, semejante al *Forward Update DistFlow* [1]). Como parte de la metodología propuesta, se contempla la obtención del estado de la red para cada configuración usando un algoritmo genético del tipo evolutivo que es un proceso de optimización combinatoria.

Se inicia con la generación de una población, es decir, un conjunto de soluciones (configuraciones) candidatas. Cada configuración es clasificada por el valor de su función objetivo que presenta. Todos los individuos de la población son clasificados por la calidad de su correspondiente función objetivo. Así, cada individuo posee una determinada probabilidad para generar nuevos elementos (configuraciones) en la generación siguiente. Los elementos mejores clasificados en este proceso tienen en el sentido probabilístico, mayor probabilidad de participar en la generación de los elementos de la nueva población. Esa nueva generación es obtenida con los operadores de recombinación y mutación.

La recombinación lleva a una transferencia de materiales genéticos entre las mejores configuraciones. Eventualmente deben aparecer elementos de la nueva generación con mejor material genético de las configuraciones mejores clasificadas, produciendo nuevas configuraciones de excelente calidad.

En la naturaleza, la mutación produce la regeneración de la pérdida de material genético. En los algoritmos evolutivos la mutación produce una modificación aleatoria de algunos genes de un elemento (configuración) de la población. De forma general, los algoritmos evolutivos son técnicas de optimización combinatoria que inician el proceso con una población inicial, haciendo un ordenamiento selectivo de los elementos de la población dando valor a la calidad de la función de aptitud y generan una nueva población en un proceso de tres etapas: Selección, recombinación y mutación.

En la etapa de ***selección***, las configuraciones son escogidas mediante juegos (torneos) y la cantidad de torneos es equivalente al tamaño de la población, tornando esa propuesta significativamente diferente de la selección proporcional. Para cada juego son escogidos aleatoriamente un conjunto de ***k*** configuraciones y la configuración ganadora es aquella con mejor función objetivo. El valor de ***k*** generalmente es pequeño, típicamente $k \in \{2, 3,4, 5\}$. Se Realizan ***n*** juegos.

En el proceso de ***recombinación***, para que las configuraciones seleccionadas sean sometidas a recombinación, se debe generar un número aleatorio $p \in [0,1]$. Si p es menor que una tasa de recombinación (***ρc***) entonces, se debe proceder a la recombinación; en caso contrario las dos configuraciones seleccionadas no son recombinadas. En esta parte se implementa una codificación según las **ramas de conexión,** codificación asociadas con propiedades de cromosomas candidatas.

Y el proceso de ***mutación*** consiste en modificar la estructura genética de los individuos de forma aleatoria. Considerando el cromosoma como una cadena de *bits*, solamente necesita cambiar uno de ellos para obtener un cromosoma con una nueva información genética que no se encuentra en la población base. Este efecto, permite una diversificación de la población, y que las soluciones no puedan converger prematuramente para un óptimo local. La necesidad de la mutación aumenta en la fase final del procedimiento cuanto las populaciones se tornan cada vez mas homogéneas y dominadas por los genes mas eficientes. La mutación introduce mudanzas en torno a las variables, explorando nuevas zonas en el campo de la optimización.

Por último se aplica un criterio de parada, el mismo que compara la solución incumbente, si esta no presenta mejoría durante un número especificado de iteraciones, se considera que el proceso converge.

En [13], se presenta la reconfiguración de redes eléctricas mediante la aplicación de Algoritmos Genéticos AG. La técnica empleada consiste en analizar cada malla de una red de distribución en su conformación inicial representando el estado de los interruptores, lo cual constituye un conjunto de configuraciones iniciales (población inicial del AG). En cada iteración y a partir de la población corriente, usando los operadores de selección,

recombinación y mutación, se obtiene un nuevo conjunto de configuraciones (nueva población). En cada una de ellas se encuentran configuraciones de mejor calidad pretendiéndose hallar una solución (configuración) óptima global. En los AG, la información sobre el objetivo a alcanzar es representada mediante la función de aptitud que se procurará minimizar. En este caso se propone una función que no sea sobreaproximada, la cual conduciría directamente a una solución irrealizable.

La siguente lista contiene diferentes expresiones utilizadas en la genética y su estructura equivalente en AG.

Evolución natural	Algoritmo genético
Genotipo	Código de cadena
Fenotipo	Punto sin codificar
Cromosoma	Cadena
Gen	Posición de cadena
Alelo	Valor en una posición determinada
Función de aptitud o aptitud	Valor de la función objetivo

Tabla 1-1: Expresiones utilizadas en la genética y su estructura equivalente en AG [14]

La optimización por enjambre de partícula[13] [12], es una técnica de optimización estocástica basada en poblaciones, desarrollada por Kennedy y Eberthar en 1995, la cual es motivada por la simulación del comportamiento social de la interacción entre individuos (partículas) de un grupo (enjambre) [15].

Fue desarrollada a partir de las observaciones de parvadas de aves y en el cardumen de peces, en su búsqueda de alimentos en una determinada región. Al observar el comportamiento de estos grupos, se verificó que el comportamiento del grupo era influenciado por la experiencia individual (decisión individual) acumulada por cada individuo, o bien por el resultado de la experiencia acumulada por el grupo (influencia social) [15], donde la mejor solución puede ser representada como un punto o superficie en un espacio n- dimensional. PSO, presenta un sistema que es inicializado con una población de soluciones aleatorias. A diferencia de otros algoritmos, para cada solución potencial (llamada partícula) es también asignada una velocidad aleatoria y entonces esta fluye a través del hiper espacio del problema.

[13] Particle Swarm Optimization (PSO)

PSO ha demostrado que es efectivo para resolver un amplio rango de problemas de ingeniería, este puede manejar variables discretas y continuas, es muy fácil de implementar y resuelve problemas en un tiempo pequeño de simulación. El proceso de PSO, después de inicializar la posición y la velocidad con valores aleatorios, cada posición de la partícula es evaluada para establecer su aptitud o adaptación (condición física). Este es el primer paso del algoritmo, en el cual la posición aptitud es evaluada y si su valor es mejor que el mejor valor de la medida, es decir un conjunto de nuevas mejores posiciones, pbest.

Una vez que todas las partículas de aptitud son evaluadas, el algoritmo se desplaza a un segundo lazo, entre todos los pbest, el mejor valor obtenido hasta el momento por cualquier partícula en la vecindad de pbest se llama gbest. El concepto básico que está detrás de PSO es acelerar cada partícula hacia sus lugares pbest y gbest.

En el algoritmo PSO, el enjambre es inicializado aleatoriamente con una población de soluciones candidatas, siendo cada partícula inicializada con una posición y una velocidad aleatoria.

Un enjambre consiste de un número de partículas "o posibles soluciones" que procede (vuela) a través del espacio de una solución factible para explorar soluciones óptimas. Cada partícula actualiza su posición basada en su propia mejor exploración; la mejor experiencia de todo el emjambre, y su vector de velocidad de acuerdo al siguiente modelo [2]:.

$$v_i^{k+1} = w\, v_i^k + c_1 * rand(\,) * \frac{(pbest_i - s_i^k)}{\Delta t} + c_2 * rand(\,) * \frac{(gbest_i - s_i^k)}{\Delta t} \qquad 1\text{-}2$$

$$s_i^{k+1} = s_i^k + v_i^{k+1} * \Delta t \qquad 1\text{-}3$$

Donde [2]:

v_i^k	componente de velocidad i^{th} en la iteración k
$rand()$	Número aleatorio entre 1 y 0
s_i^k	Posición de la corriente en la dimensión i^{th}
c_1, c_2	Coeficientes de aceleración, que usualmente son seteados a 2.0
$pbest_i$	Mejor posición individual en la dimensión i^{th}
$gbest_i$	Mejor posición global en la dimensión i^{th}
Δt	Intervalo de tiempo
w	Peso inercial

La primera parte en la Ecuación 1-2 es el término de momento de la partícula, siendo una ponderación de inercia (*w*), o el grado de momento de la partícula. La segunda parte corresponde a la parte "cognitiva", que representa el "conocimiento" de la partícula adquirido a lo largo de los desplazmientos. La tercera parte es la parte "social", que representa una colaboración entre las partículas.

La principal ventaja de las técnicas de inteligencia de enjambre es que son impresionantemente resistentes al problema de óptimo local.

En [2], se usa PSO para resolver el problema de reconfiguración de redes, y se realizaron algunas modificaciones, tales como el uso de una masa de inercia que disminuye linealmente durante la simulación. El método tradicional sufre algunas modificaciones para aplicarlo a la reconfiguración de redes, a continuación se describen las ideas generales:

a) ***Ponderación inercial:*** En PSO original la ecuación actualización de velocidad puede ser obtenida seteando *w* = 1. En este trabajo el valor inercial (*w*), se asume que decrese linealmente durante el curso de la simulación desde 0.9 a 0.2.
b) ***Algoritmo de reparación:*** Después de actualizar la posición de cada partícula, se comprueba la posición de la partícula para asegurarse que ninguna de las partículas han "*volado*" fuera del límite del espacio de búsqueda, es decir, no han violado las restricciones. Por lo tanto si una violación es detectada, un algoritmo de reparación es usado para forzar a la partícula a retornar a la región factible.

   La posición de la partícula representa el número de SNA en las ramas de interconexión a ser cerrados. Por lo tanto, los valores de las posiciones de las partículas deben ser enteros, positivos y limitados por el intervalo [1 mp] donde, mp es el número total de SNA en las ramas de interconexión. Tal que la posición (j) debe ser aproximada al número entero más cercano.

La posición escogida para cada partícula representa el número de SNA en las ramas de interconexión, en la configuración $\Omega_{\text{interconexión}}$, a ser cerrada, donde este seccionador se cerrará y creará un lazo en la red. Para restaurar el sistema a la estructura radial, uno de los SNC en el lazo debe ser abierto. El SNC con mínima pérdida ha sido escogido para ser abierto, y se ha encontrado que esta suposición da buenos resultados.

En este artículo, la velocidad de la partícula debe ser limitada por este intervalo [-mv mv], donde mv es la velocidad máxima. No es necesario que la velocidad sea un entero, ya que es usado solo para actualizar la posición de la partícula.

c) ***Diseño de una variable de expresión:*** Un buen diseño de una variable de expresión para establecer la radialidad en la reconfiguración de redes de distribución es muy importante para mejorar la eficiencia del proceso de búsqueda como se indica en el algoritmo propuesto (*literal d*) para chequear que el sistema sea radial.

d) ***Algoritmo para chequear la topología radial del sistema:*** Se introduce un nuevo algoritmo para chequear la radialidad del sistema basado en la matriz de incidencia de barra Â. Una vez que se forma la matriz Â, se evalúa su determinante, si este es igual a 1 ó -1, el sistema es radial, si es igual a 0, significa que el sistema no es radial o el grupo de cargas están desconectadas del servicio.

La aplicación de PSO para resolver el problema de reconfiguración, se resumen en los siguientes pasos [2]:

1. Para cada partícula, los vectores posición y velocidad deben ser aleatoriamente inicializados.
2. Medir la aptitud (pérdida de potencia) por cada partícula (***pbest***) y almacenar la partícula con el mejor valor de aptitud (***gbest***) ejecutando el programa de flujo de carga.
3. Actualizar los vectores posición y velocidad de acuerdo a las ecuaciones 1-2 y 1-3 para cada partícula.
4. Realizar el chequeo de violación. Si la violación es detectada entonces, se aplica el "algoritmo de reparación" explicado en *b)*, entonces finaliza.
5. Disminuir el peso de inercia *(w)* linealmente como es explicado en ***a)***.
6. Repetir los pasos 2-5 hasta que el criterio de finalización es satisfecho.

CAPÍTULO 2

2 CÁLCULO DE FLUJO DE POTENCIA PARA REDES RADIALES DE DISTRIBUCIÓN

2.1 INTRODUCCIÓN

El estudio de flujo de potencia (flujo de carga), en el área de sistemas eléctricos de potencia, es la solución de régimen permanente de la red del sistema. La principal información que se obtiene de este estudio incluye las magnitudes y los ángulos de fase de nodos (barras) de carga, potencia reactiva en los nodos (barras) del generador, flujo real y reactivo de potencias en las líneas (ramas) y otras variables que se especifiquen. Esta información es esencial para el monitoreo continuo del estado actual del sistema y para estudios de planeamiento de la expansión de generación y red ante escenarios futuros de demanda [16].

Es decir, el problema de flujo de potencia consiste en obtener el estado de operación de una red (ángulos y magnitudes de los fasores de voltaje nodal). Una vez obtenido el estado de la red, otros parámetros como: flujo de potencia y corrientes en las ramas, entre otros pueden ser fácilmente determinados. Varios métodos de flujos de potencia han sido propuestos a lo largo de los años y actualmente el método más robusto es el método de Newton y sus versiones desacopladas. La robustez del método de Newton tradicional tiene problemas con la construcción de la inversión de la matriz jacobiana, lo que aumenta el esfuerzo computacional en el algoritmo. En el estudio de reconfiguración de redes de distribución, casi siempre hay la necesidad de realizar el procesamiento repetitivo de numerosos flujos de cargas, por lo tanto, tal método sería inviable [17].

La alta relación R/X y la característica topológica de los sistemas de distribución deterioran la dominancia de la diagonal de la matriz utilizada en el algoritmo Newton-Raphson, volviéndose próxima a una matriz singular.

Por otra parte, un sistema de distribución presenta dos características muy específicas [15]:

1. Operan en forma radial, esto es, no existen lazos o mallas en el sistema.
2. Ciertos sistemas presentan una relación R/X muy elevada en comparación con valores típicos de sistemas de transmisión o subtransmisión, motivo por el cual estos sistemas entran en la categoría de "mal -condicionados".

La primera característica es una ventaja porque simplifica considerablemente la complejidad del problema, entretanto la segunda característica es una desventaja cuando son usadas las versiones desacopladas del método de Newton, pues fueron utilizados eficientemente para resolver sistemas "bien -condicionados", siendo que para sistemas mal condicionados, en la mayoría de veces, presentan problemas de convergencia.

El desarrollo de un método de optimización para la reconfiguración óptima necesita de un algoritmo de solución para el flujo de potencia que sea capaz de determinar el flujo de potencia para redes de distribución radiales con miles de ramas (líneas) y de nodos (barras), siendo robusto y eficiente al mismo tiempo. Tal necesidad, se debe al hecho que un problema de optimización de sistema de energía eléctrica, como la optimización de este trabajo, requiere una solución de un elevado número de flujos de carga en cada iteración.

En el siguiente numeral se describen brevemente algunos métodos para resolver flujos de potencia para redes de distribución de energía eléctrica.

2.2 MÉTODOS DE CÁLCULOS DE FLUJO DE POTENCIA PARA REDES DE DISTRIBUCIÓN

Debido a la radialidad de los alimentadores, la alta relación X/R y a la longitud muy variable de las líneas en la distribución, las técnicas iterativas comúnmente usadas en estudios de flujo de potencia en las redes de transmisión no pueden ser adoptadas debido a las pobres características de convergencia que presentan. En las últimas décadas, diferentes procedimientos para flujo de potencia en redes de distribución han sido propuestos. La experiencia muestra que usando el procedimiento backward-forward orientado a ramas, ha obtenido muy buenos resultados en el análisis de las redes de distribución reales de gran tamaño [18].

2.2.1 MÉTODOS DE CÁLCULO DE FLUJO DE POTENCIA EXISTENTES

En esta parte serán analizados distintos métodos de flujo de potencia para redes radiales de distribución de energía eléctrica factibles que han sido estudiados y utilizados en la bibliografía especializada. Primeramente serán comentados los métodos tradicionales para posteriormente analizar los flujos de potencia radiales específicamente construidos para estos sistemas.

2.2.1.1 Métodos tradicionales de cálculo de flujo de potencia [18]

Los métodos tradicionales para resolver flujos de potencia, tales como: Gauss-Seidel, Newton-Raphson y otros, no muestran buenos resultados cuando son aplicados a redes de distribución.

El método Gauss-Seidel indirecto se caracteriza por ser relativamente insensible a los voltajes iniciales estimados, su poco requerimiento de memoria computacional (la Matriz de Admitancia nodal Y es muy porosa debido a la configuración radial del sistema en estudio) y su simple programación. Sin embargo, su lenta convergencia, acentuada en los sistemas radiales, lo hace poco atractivo. Esta lentitud se debe principalmente al no aprovechamiento de la naturaleza porosa de la matriz Y. La característica radial de los sistemas hace que los valores de la diagonal de la matriz Y sean pequeños. Luego, con el perfil de voltajes de la próxima iteración es inversamente proporcional a tales valores. De esa manera, son producidas grandes oscilaciones del voltaje volviéndose lenta la convergencia.

Entre tanto, los métodos Newton-Raphson completo y las versiones desacopladas son ampliamente conocidos por sus excelentes características de convergencia, sobre todo en las versiones desacopladas. La principal desventaja del NR completo consiste en tener que calcular e invertir para cada iteración la matriz Jacobiana, que es aproximadamente dos veces el tamaño de la matriz Y. Como la estructura de la matriz Jacobiana tiene las mismas características de porosidad de la matriz Y, es posible utilizar técnicas de bifactorización en la inversión, reduciendo los tiempos de procesamiento.

Entre tanto, las versiones desacopladas contemplan una serie de aproximaciones que simplifican la matriz Jacobiana, haciendo menor el tiempo de cada iteración. Sin embargo, estas

aproximaciones consideran valores de la razón X/R que no son efectivos en todos los sistemas de distribución. Por lo tanto, estos métodos no son atractivos para ser aplicados en estos sistemas.

El método de Gauss-Seidel directo es más confiable que el método anterior (difícilmente diverge). Además, aunque los voltajes presentan convergencia de oscilaciones mayores comparado con el método indirecto, el proceso global converge mucho más rápido. También, es menos dependiente del tamaño del sistema. Esto sucede ya que la matriz de impedancia de barras (matriz Z) es llena y proporciona un buen acoplamiento matemático entre los voltajes de las barras, esto es, una mejoría en el valor de un voltaje (en media y bajo voltaje) afecta inmediatamente el cálculo de los próximos voltajes. Pero, la principal desventaja es la gran memoria computacional requerida para almacenar explícitamente la matriz Z y el gran tiempo de procesamiento para su obtención. Este problema puede ser solucionado almacenando la matriz Y e invirtiéndola en cada iteración para obtener la matriz Z. Esto reduce significativamente la cantidad de memoria, pero al mismo tiempo, produce un gran aumento en la carga computacional. Por tales motivos, las publicaciones actuales dedicadas a sistemas de distribución apuntan en su mayoría al desarrollo de flujos de potencia para redes radiales, quedando descartados los métodos descritos.

2.2.1.2 Métodos de cálculo de flujo de potencia radial

Los métodos de flujo de potencia radial han sido mejorados y su principal característica es el aprovechamiento de la topología radial de los sistemas de distribución. Los más usados, dentro de los métodos orientados a ramas (*branch-oriented methods*), son:

- Método DistFlow [1], [3]
- Método escalonado (*Ladder Method*) [18]
- Método de Suma de corrientes (*Current Summation Method*) [17]
- Método de Suma de potencias (*Power Summation Method*) [19], [7]

El Método DistFlow fue presentado en [3] y utilizado en [1], basado en una técnica de Newton Raphson, requiere el cálculo de la matriz Jacobiana, una serie de multiplicaciones de la matriz e invertir la matriz al menos una vez [20].

El método escalonado resuelve la red "aguas arriba" (en dirección al nodo fuente), suponiendo previamente un perfil de voltaje, aplicando directamente las leyes de corriente y voltaje de Kirchhoff hasta llegar al nodo fuente. De esta manera es posible calcular un voltaje del nodo fuente. El error obtenido entre este valor y el especificado será sumado al perfil de voltaje previamente supuesto de tal modo que se obtenga un nuevo perfil de voltaje para la próxima iteración. La convergencia es alcanzada cuando el voltaje resultante del nodo fuente es el especificado [18].

Los métodos restantes constan de dos procesos: "aguas arriba" y "aguas abajo". En el proceso de aguas arriba, previamente supuesto un perfil de voltaje, se calculan las corrientes (Suma de corriente) o las potencias nodales (Suma de Potencias), de acuerdo a cada caso. En el proceso de aguas abajo se obtienen nuevos valores para los voltajes, a partir del cálculo anterior. Estos valores de voltaje son los que serán utilizados en la próxima iteración. Finalmente, la convergencia es chequeada con el voltaje o con la potencia especificada. Estos métodos aplicados a sistemas de distribución en general muestran mejores características de convergencia (rapidez y confiabilidad) que los tradicionales.

Por otro lado, el Método Escalonado tiene como principal desventaja el hecho de limitar la profundidad de los subalimentadores (ramas laterales) del sistema, pues cada uno de ellos necesita de subiteracciones. Además, su característica de convergencia no es buena para sistemas cargados. Por estas razones este método no es el más atractivo [18].

También en la literatura especializada se realiza una discusión en el que se compara el método de suma de corriente con el método de suma de potencias. La discusión apunta a que el uso de suma de potencias presenta un error menor (dependiente apenas de las pérdidas del sistema) en el proceso iterativo que la suma de Corrientes (error depende del voltaje inicial). Sin embargo, el método de Suma de corrientes ha sido probado en diferentes escenarios de carga y dimensión de redes, sin presentar problemas de convergencia [18].

A continuación, se describen brevemente algunos métodos para el cálculo de flujos de potencia radiales más destacados en la literatura especializada para sistemas de distribución, en orden cronológico, de acuerdo a lo estudiado en [18].

2.2.1.2.1 M. E. Baran e F. F. Wu, 1989 [3]

El sistema de distribución considerado consiste de un alimentador radial. El diagrama unifilar de tal alimentador con n ramas/nodos (barras) es mostrado en la **Figura 2-1**.

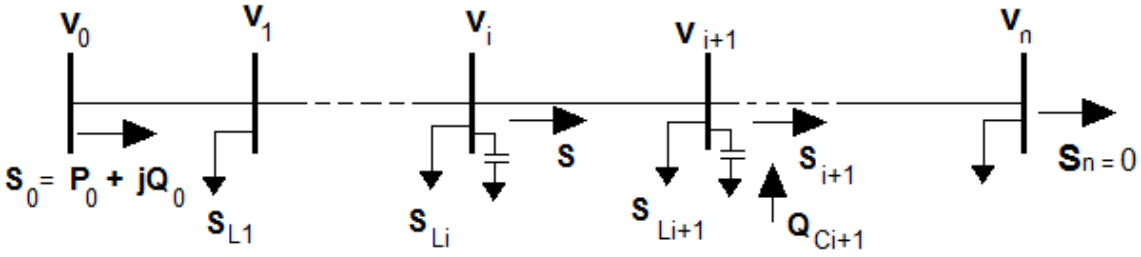


Figura 2-1: Diagrama unifilar del alimentador principal

Si la potencia prevista por la subestación fuera conocida, la potencia y el voltaje del nodo (barra) aguas abajo también lo serán. Así, se obtienen las siguientes fórmulas recursivas para cada rama del alimentador.

$$P_{i+1} = \frac{P_i - r_{i+1}\,(P_i^2 + Q_i^2)}{V_i^2 - P_{Li+1}} \qquad 2\text{-}1$$

$$Q_{i+1} = \frac{Q_i - x_{i+1}\,(P_i^2 + Q_i^2)}{V_i^2 - Q_{Li+1} + Q_{ci+1}} \qquad 2\text{-}2$$

$$V_{i+1}^2 = -\frac{2(r_{i+1}P_i + x_{i+1}Q_i) + (r_{i+1}^2 + x_{i+1}^2)(P_i^2 + Q_i^2)}{V_i^2} \qquad 2\text{-}3$$

Siendo,

r_i	resistencia de la línea "aguas arriba" del nodo *i*;
x_i	reactancia de la línea "aguas arriba" del nodo *i*;
P_i, Q_i	flujos de potencia activa y reactiva en el ramal "aguas arriba" del ramal *i+1* que conecta el nodo *i* con el nodo *i+1* ;
P_{Li}, Q_{Li}	flujos de potencia activa y reactiva de la carga en el nodo *i* ;
V_i	valor del voltaje del nodo *i* ; y
Q_{ci}	inyección de potencia reactiva del capacitor en el nodo *i.*

Las ecuaciones 2-1 , 2-2 y 2-3 son llamadas "ecuaciones de flujo de rama" y tienen la siguiente forma:

$$x_{0i+1} = f_{0i+1}(x_{0i}) \qquad 2\text{-}4$$

Siendo, $x_{0i+1} = [P_i Q_i V_i^2]^T$

Teniéndose las siguientes condiciones terminales:

(i) En la subestación, sea el voltaje de la subestación V^{sp}, luego

$$x_{00_3} = V_0^2 \ = V^{sp2} \qquad 2\text{-}5$$

(ii) Al final del alimentador principal;

$$x_{0n_1} = P_n \ = 0 \ ; \ x_{0n_2} = Q_n \ = 0 \qquad 2\text{-}6$$

Las 3*n* ecuaciones de flujo de ramas (2-1, 2-2 y 2-3) junto con las 2 condiciones limitantes (2-5 y 2-6) constituyen el sistema de ecuaciones y son referenciados como "***Ecuaciones DistFlow***". Son de la forma:

$$G(x_0) = 0 \qquad 2\text{-}7$$

Siendo, $x_0 \ = \ [x_{00}^T \dots x_{0n}^T]^T$ las variables de ramas.

Esta metodología es generalizada para incluir ramas laterales. La solución de las "Ecuaciones *DisFlow*" es realizada usando el método iterativo Newton-Raphson, el cual requiere de la elaboración de una matriz jacobiana que tiene que ser invertida en el proceso. El esfuerzo computacional que precisa este método deja en evidencia el análisis de sistemas de gran tamaño, siendo tres veces mayor en el modelaje trifásico.

2.2.1.2.2 R. Cespedes, 1990 [19]

El método está basado en la equivalencia eléctrica y en la eliminación del ángulo de fase en las ecuaciones a ser desarrolladas, lo que permite obtener una solución exacta trabajando apenas con los módulos de los voltajes. Un valor del ángulo del voltaje no es importante en la mayoría de los estudios relacionados con los niveles de voltaje en la distribución. Además, la diferencia entre los valores de los ángulos de voltaje en el alimentador no excede unos pocos grados. El algoritmo es aplicable en el cálculo de flujo de potencia monofásico y trifásico.

La solución propuesta para el problema de flujo de potencia es revolver para cada ramal la ecuación básica 2-8 que utiliza en la **Figura 2-2**.

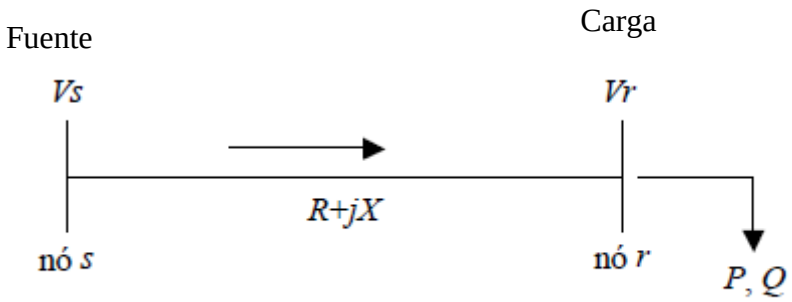


Figura 2-2: Rama de un sistema radial

$$V_r^4 + [2.(PR + QX) - V_S^2].V_r^2 + (P^2 + Q^2).(R^2 + X^2) = 0 \qquad 2\text{-}8$$

Siendo,

S	nodo en la fuente;
r	nodo en la carga;
V_S	módulo de voltaje del nodo fuente;
V_r	módulo de voltaje del nodo de carga;
P,Q	cargas activa y reactiva; y
R,X	resistencia y reactancia de la rama.

La ecuación 2-8 no depende del ángulo de voltaje, lo que simplifica la formulación del problema. En la solución propuesta P y Q son las cargas totales alimentadas por el nodo r, incluyendo la carga del nodo y aquellas alimentadas por el nodo, además de las pérdidas. Las pérdidas de potencias activas y reactivas son calculadas de la siguiente manera:

$$L_p = \frac{R.(P^2 + Q^2)}{V_r^2} \qquad 2\text{-}9$$

$$L_q = \frac{X.(P^2 + Q^2)}{V_r^2} \qquad 2\text{-}10$$

Siendo,

L_p	Pérdidas activas de la rama; y
L_q	Pérdidas reactivas de la rama;

El proceso iterativo comienza con el cálculo de las potencias equivalentes en cada nodo sumando todas las cargas de la red que son alimentadas por cada nodo incluyendo las pérdidas. Esta es llamada de iteración “aguas arriba” desde los nodos finales hasta el nodo fuente. Luego comenzando desde el nodo fuente y usando 2-8, se calcula la tensión V_r para cada nodo. Esta es llamada de iteración “aguas a abajo”, desde el nodo fuente hasta los nodos finales. Posteriormente, con los nuevos voltajes recalcular las pérdidas. Si la variación de pérdida total respecto del valor previamente calculado es mayor que una tolerancia especificada, ir a la iteración “aguas arriba”. Caso contrario, calcular otros parámetros requeridos, como las corrientes por ejemplo.

2.2.1.2.3 C. S. Cheng y D. Shirmohammadi, 1995 [21]

Esta metodología está basada en el cálculo de las corrientes. Fue inicialmente propuesto para redes monofásicas en 1988 y luego adaptado para redes trifásicas. El algoritmo asume un perfil de voltaje, calculando luego las inyecciones de corrientes para tal condición. Posteriormente, son obtenidos los flujos de corrientes en las líneas comenzando por las más distantes de la subestación hasta la más próxima de ella (*bakward sweep*). Usando las corrientes en las líneas, es iniciado el proceso aguas abajo donde son calculados los voltajes en todos los nodos comenzando por la subestación en dirección a los nodos más distantes (*forward sweep*). Estos tres últimos pasos deberán ser repetidos hasta que la convergencia sea alcanzada. La metodología incluye también una propuesta de re-numeración para mejorar el desempeño computacional del algoritmo.

2.2.1.2.4 F. Zhang y C. S. Cheng, 1997 [22]

En este trabajo el método modificado de Newton para sistemas de distribución radiales es derivado de tal manera que la matriz Jacobiana se establezca en la forma UDU_T , siendo U la matriz triangular superior constante dependiente apenas de la topología del sistema y D una matriz diagonal de bloques resultante de la estructura radial y propiedades especiales del sistema de distribución. Con esta formulación los pasos convencionales para la formación de la matriz Jacobiana son sustituidos por barridos "aguas arriba" (*backward*) y "aguas bajo" (*forward*) en los alimentadores radiales con impedancias equivalentes, de esta manera calcular la corrección incremental de las variables de estado.

Las ventajas de esta metodología son:

(1) es un método Newtoniano por lo tanto puede ser extendido para otras aplicaciones tales como estimación de estado;
(2) la matriz Jacobiana en la forma UDU_T no necesita ser explícitamente formada, y los barridos *backward* y *forward* son directamente basados en las ecuaciones lineales de flujo de potencia. Así, posibles "malos acondicionamientos" asociados con la matriz Jacobiana y sus factores LU son completamente evitados; y

(3) los resultados de las pruebas realizadas han mostrado que el método es tan robusto y eficiente como el método de barrido *backward- forward*.

2.3 MÉTODO DE FLUJO DE POTENCIA UTILIZADO PARA LA RED PROTOTIPO

Para este trabajo de investigación se modificaron los códigos encontrados en [23], de tal manera, que solo sea utilizado para correr flujos de potencia sin generación distribuida, puesto que no se pudo aplicar la herramienta MATPOWER, por los problemas que se explican en el desarrollo de este capítulo.

2.3.1 REVISIÓN GENERAL DEL MÉTODO DE FLUJO DE CARGA ([23], [20])

En esta sección, un simple modelo de circuito de una línea de transmisión y ecuaciones de voltaje recursivas asociadas son presentadas. Se asume que el Sistema de Distribución Radial SDR es balanceado y puede ser representado por un diagrama unifilar básico. Los elementos capacitivos a tierra de la línea en el nivel de voltaje de distribución son pequeños y entonces no son considerados. El modelo de la línea es mostrado en la **Figura 2-3**.

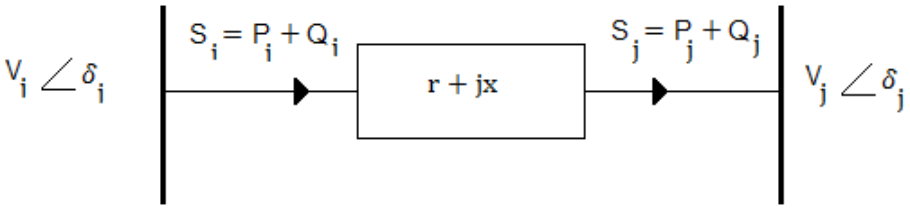


Figura 2-3: Circuito equivalente simple de una línea [20]

La línea conectando esos dos segmentos tiene una impedancia ($Z= r + jx$). El flujo de potencia en el nodo de envío ($S_i = P_i + jQ_i$) es diferente para el flujo de potencia en el nodo de recepción ($S_j = P_j + jQ_j$). Los valores de magnitud de voltaje y ángulo de fase asociados con cada nodo, corresponde a: V_i, δ_i, V_j, δ_j

Un algoritmo de flujo de potencia resuelve las ecuaciones de balance de potencia en todas las barras (nodos) y determina la solución de voltaje correspondiente. En las barras de carga (barras PQ), el algoritmo de flujo de carga será resuelto para la magnitud de voltaje y ángulo de fase de la barra. En **Tabla 2-1**, se muestra el resumen de los valores conocidos y los que tienen que ser resueltos para barras PQ.

Variables	Nodo (Barra) PQ
Variables conocidas	P,Q
Variables no conocidas	V,δ

Tabla 2-1: Valores conocidos y no conocidos para una barra tipo PQ

El método de flujo de carga para este proyecto usa ecuaciones determinísticas y un lazo iterativo para determinar una solución de voltaje para un sistema. Esas ecuaciones fueron basadas en las reportadas en [3]. Cada iteración comienza en los extremos de la cola del SDR a ser resuelto, y trabaja su camino hacia la cima de la estructura SDR. La cima representa la subestación, el punto donde el SDR se interconecta con un sistema de distribución. Para cada solución de barra, valores son necesarios desde los nodos de envío y recepción. Sin embargo, el nodo a ser resuelto es siempre el nodo de recepción.

2.3.2 SOLUCIÓN PARA NODOS PQ

Los pasos usados para tratar nodos PQ que son usados en esta sección son basados en el trabajo reportado en [20]. Estos pasos, son una parte importante del método de flujo de carga de este proyecto.

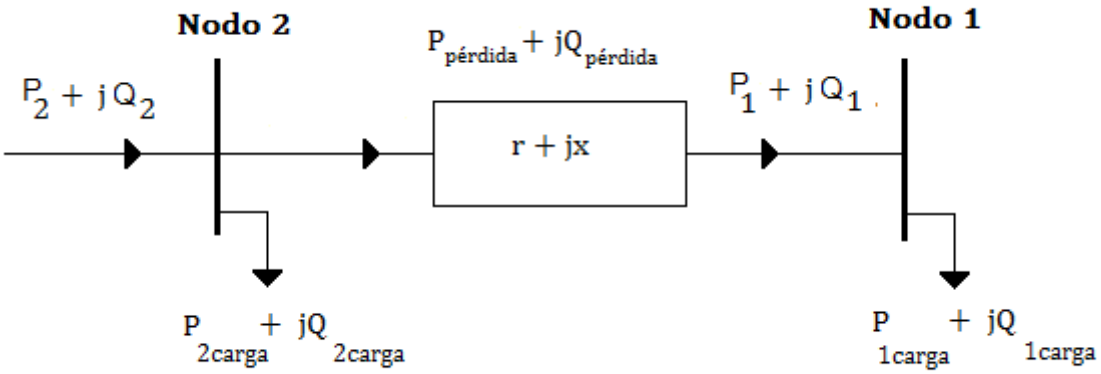
Los parámetros conocidos en el nodo de carga son las potencias activa y reactiva recibidas. Entonces el flujo de carga debe resolver la magnitud de voltaje y el ángulo de fase del nodo. Las ecuaciones 2-11 y 2-12, son usadas para resolver esos valores.

$$V_j^2 = -\left[r\ \ P_j\ + x\ \ Q_j - \frac{V_i^2}{2}\right] + \sqrt{\left[r\ \ P_j\ + x\ \ Q_j - \frac{V_i^2}{2}\right]^2 - \left[r^2 + x^2\ \right]\left[P_j^2 + Q_j^2\right]} \qquad 2\text{-}11$$

$$\delta_j = \delta_i - \cos^{-1}\left[\sqrt{1.0 - \left(\frac{P_j x - Q_j r}{V_i V_j}\right)^2}\right] \quad 2\text{-}12$$

Si los valores de ángulo de fase y magnitud de voltaje son calculados para el nodo de recepción, de acuerdo a las ecuaciones 2-11 y 2-12 (**Figura 2-3**), las únicas variables requeridas son los valores de potencia activa y reactiva del nodo de recepción, los valores de ángulo de fase y magnitud de voltaje del nodo de envío, y el valor de la impedancia de línea conectando los dos nodos. Todos los valores requeridos para los cálculos de la barra PQ son obtenidos fácilmente en la práctica.

Se debe tener cuidado al determinar los valores de P_i y Q_i, las potencias activas y reactivas en el nodo de recepción. P_i y Q_i no deben ser confundidos con los valores de inyección de potencia real y reactiva en esa barra. Por ejemplo, si el nodo de recepción es el último nodo en un rama del SDR, entonces P_i y Q_i podría ser igual a la inyección de la potencia activa y reactiva (demanda de potencia en el nodo, $P_{icarga}\ y\ Q_{icarga}$) en ese nodo. Si el nodo de recepción es además hacia arriba, entonces *Pi* y *Qi* debería ser la suma de todas las potencias activas y reactivas hacia abajo, más $P_{icarga}\ y\ Q_{icarga}$). En la **Figura 2-4**, se ilustra un ejemplo de como P_j y Q_j son calculadas.



$$P_1 + jQ_1 = P_{1carga} + jQ_{1carga}$$
$$P_2 + jQ_2 = P_{2carga} + jQ_{2carga} + P_{pérdida} + jQ_{pérdida} + P_1 + jQ_1$$

Figura 2-4: Diferencia entre potencias activa y reactiva recibida e inyectada (figura modificada de [23])

Las pérdidas de potencia activa y reactiva pueden ser calculados como:

$$P_{pérdida-ij} = r\frac{(P_j^2 + Q_j^2)}{V_j^2} \qquad 2\text{-}13$$

$$Q_{pérdida-ij} = x\frac{(P_j^2 + Q_j^2)}{V_j^2} \qquad 2\text{-}14$$

La potencia reactiva está dada por la siguiente ecuación:

$$Q_j = Q_{carga} + Q_{acc} \qquad 2\text{-}15$$

En la ecuación 2-15, Q_j es la potencia reactiva en el nodo de recepción, Q_{carga} es la carga de potencia reactiva en el nodo de recepción, y Q_{acc} es la potencia reactiva acumulada, pero antes del nodo de recepción. La **Figura 2-5**, ilustra el significado físico de los valores de potencias reactivas.

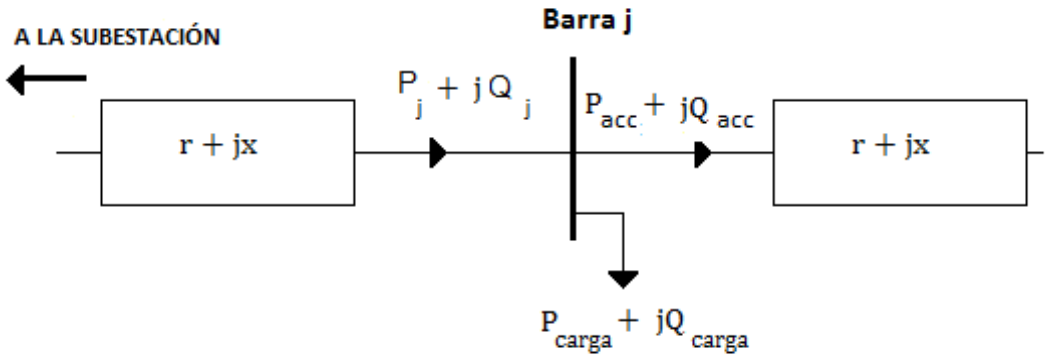


Figura 2-5: Ilustración de valores de potencia reactiva (modificado de [23])

2.3.3 ALGORITMO DE FLUJO DE CARGA

En la **Figura 2-6** se presenta un flujograma que muestra el algoritmo de flujo de carga desarrollado para este proyecto. Esta figura ilustra claramente la naturaleza iterativa del flujo de potencia desarrollado.

El primer paso en el algoritmo de flujo de carga es organizar la información del sistema de distribución radial SDR en una estructura adecuada [20]. Los datos del SDR deben ser organizados de una manera apropiada tal que el programa de flujo de carga pueda empezar en el nodo correcto e iterar a través del resto de las barras SDR en el orden apropiado. El programa de flujo de carga debe arrancar sus cálculos desde el último nodo de una rama terminal de SDR ("nodo de arranque" **Figura 2-6**).

Una vez que el flujo de carga ha empezado desde el "Nodo de Arranque" (un nodo final), puede continuar para resolver hacia arriba (hacia la subestación) hasta el nodo de cabeza de la rama particular que ha sido alcanzado. Cuando el nodo de cabeza de la rama ha sido alcanzado, el programa de flujo de carga debe mirar al nodo padre de la rama y ver si los cálculos de flujos de carga han sido realizados para todos los nodos debajo del nodo padre.

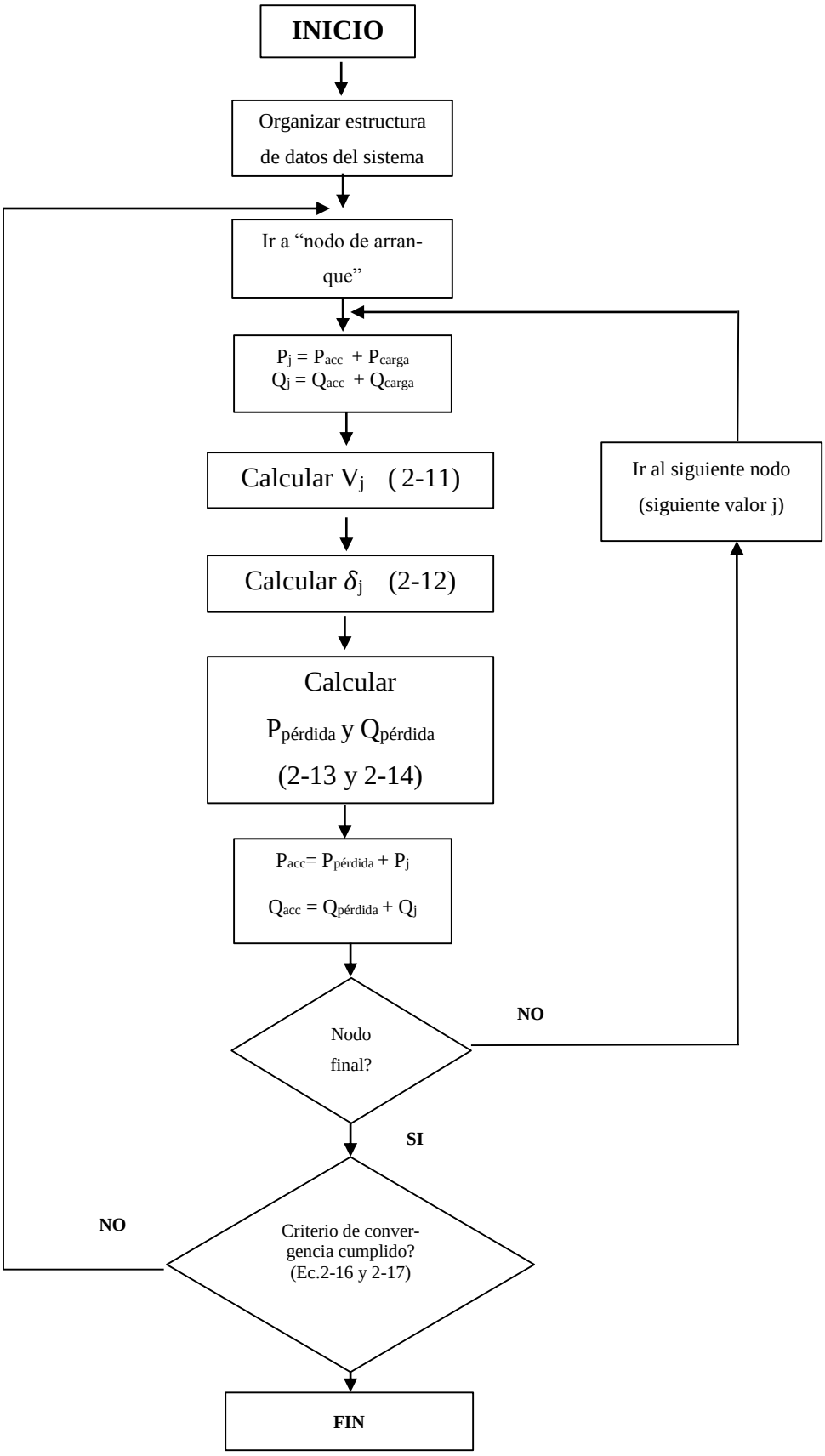


Figura 2-6: Flujograma de SDR con algoritmo de flujo de potencia, (modificado de [23])

En la **Figura 2-7**, se muestra un ejemplo de un SDR de 6 nodos.

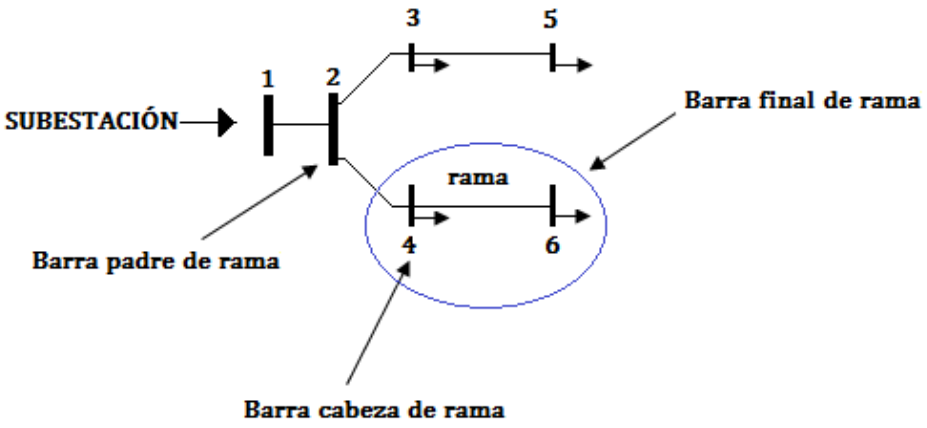


Figura 2-7: SDR de 6 barras, modificado de [24]

Si el flujo de carga arranca en el nodo 6, podría continuar para resolver desde este nodo hasta el último nodo 4. El flujo de carga no será capaz de resolver para el nodo 2 hasta que los cálculos de flujos de carga para todos los otros nodos han sido ejecutados (nodos 3-5). Un posible orden para el programa de flujo de carga para seguir debería realizar los cálculos de flujo de carga para los nodos 3-5 y 4-6 y finalmente el nodo 2.

Organizando los datos de barra del SDR, este puede ser realizado fácilmente y flexiblemente a través del uso de programación orientada a objetos y listas vinculadas. En [20] se propone una estructura de datos que realiza el uso de listas vinculadas para organizar datos dinámicamente del SDR. Este método para organizar los datos del sistema maneja fácilmente alguna reconfiguración radial.

Desafortunamente, el lenguaje de programación MATLAB® utilizado para codificar esta función no facilita naturalmente la programación orientada a objetos como algunos otros lenguajes de programación lo realizan. Así, el código para implementar un vínculo de estructura de datos inicialmente se pensó que era complicado realizar.

Sin embargo, en [23] una función para organizar apropiadamente los datos de un SDR fue codificado satisfactoriamente usando el lenguaje de programación de MATLAB. Esta función de estructura de datos toma los datos del SDR (de acuerdo a su configuración actual), y crea una matriz dentro del cual el SDR es organizado apropiadamente. En la **Tabla 2-2**, se muestra la matriz de resultados de la función de estructura de datos para el sistema que se muestra en la **Figura 2-7**.

OUTPUT =			
0	**2**	**2**	**Nodo Padre**
2	**2**	**2**	**Número de nodos en la rama**
2	**0**	**0**	**Número de ramas derivadas desde la rama**
1	**3**	**4**	**Índices de nodos en ramas**
2	**5**	**6**	
3	**0**	**0**	**Índices de nodo de cabeza de las ramas derivadas**
4	**0**	**0**	

Tabla 2-2: Resultados de la función estructura-datos de MATLAB

Cada columna de la matriz de resultados de la función estructura de datos representa una rama diferente del SDR. El primer, segundo y tercer número de cada columna representa el índice del nodo padre de, el número de nodos en, y el número de ramas derivándose de la rama representada por esa columna, respectivamente. La siguiente serie de números representa los índices de todos los nodos dentro de la rama. La última serie de números son los índices de los nodos de cabeza para cada rama derivada del final de la rama. Algunos ceros representan la ausencia de un cierto tipo de datos, tales como el nodo padre para el nodo de la subestación, y los índices de las ramas derivadas de las ramas terminales.

Por ejemplo, considere la tercera columna de la matriz de la **Tabla 2-2**. El primer número de esa columna es 2, significando que el nodo padre de la rama representada por la tercera columna es nodo 2. Los dos números siguientes, 2 y 0, significan que esta rama contiene dos nodos, y ninguna rama derivada desde el final de esta. Los siguientes números 4 y 6 son los índices de los nodos dentro de la rama. Los dos últimos números 0 y 0 son los nodos de

cabeza de las ramas derivándose desde la rama, que para este caso no contiene ningún nodo derivado de la rama.

Las columnas de la matriz de resultados de la función de estructura de datos son también organizados de una manera específica. El programa de flujo de carga puede iterar apropiadamente a través de las ramas y nodos del SDR seleccionando columnas de la matriz de resultados en una secuencia de izquierda a derecha y seleccionando nodos desde cada uno de las ramas en una secuencia de abajo hacia arriba.

Por ejemplo, considerando la matriz de resultados de la **Tabla 2-2** en la **Figura 2-7**, el programa de flujo de podría seleccionar recibiendo nodos terminales para el cálculo necesario en el siguiente orden: [6, 4, 5,3, 2, 1].

Las siguientes dos ecuaciones son usadas para el programa de flujo de carga para determinar si la convergencia ha sido lograda:

$$\left|\left(\boldsymbol{P_{gen-i}} - P_{carga-i}\right) - \sum_j V_i V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij})\right| \leq \varepsilon \qquad 2\text{-}16$$

$$\left|\left(\boldsymbol{Q_{gen-i}} - Q_{carga-i}\right) - \sum_j V_i V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij})\right| \leq \varepsilon \qquad 2\text{-}17$$

La convergencia es lograda cuando las ecuaciones 2-16 y 2-17, han sido satisfechas dentro de una tolerancia específica, ε, para cada nodo en el sistema. P_{gen-i} y Q_{gen-i} son las potencias reactivas y activas generadas en cada nodo. $P_{carga-i}$ y $Q_{carga-i}$ son las potencias activas y reactivas demandadas en cada nodo. Y_{ij} y θ_{ij} son la magnitud y ángulo de fase, respectivamente, de la correspondiente matriz elemental de admitancia de nodo del SDR. Otras variables son previamente definidas (**Figura 2-3**). Un máximo número de iteraciones pueden ser definidas tal que el programa de flujo de carga terminará en casos de no convergencia.

2.4 DEFINICIÓN DE RED PROTOTIPO Y RESULTADOS DE FLUJOS DE POTENCIA

Se definieron dos sistemas para probar el programa de flujo de carga, el primero SDR de 6 nodos y el segundo de 33 nodos, ambos casos de estudio con relación R/X mayor que 1.

El propósito de definir estos parámetros fue para comprobar que el código converge para el método seleccionado, no así con el método de Newton Raphson NR, para el cual, se simuló el sistema de 33 barras con la aplicación de MATPOWER [25] de acuerdo a lo explicado en [26].

Los datos de carga de potencia activa y reactiva e impedancia de las líneas, de los dos casos se muestran en el **ANEXO 1**.

2.4.1 CASO 1: SISTEMA DE 6 BARRAS

Inicialmente se estudia un sistema pequeño de 6 barras, el mismo que se muestra en la **Figura 2-8** (modificado de [24]). Los resultados muestran que el programa de flujo de carga es capaz de resolver este sistema radial, de alta relación R/X de las ramas y se muestran en la **Tabla 2-4**. El programa de flujo de carga, dio una solución de voltajes y ángulos en 0.68 segundos, en la iteración 5 y el voltaje más bajo en magnitud fue de 0,919 en la barra 5.

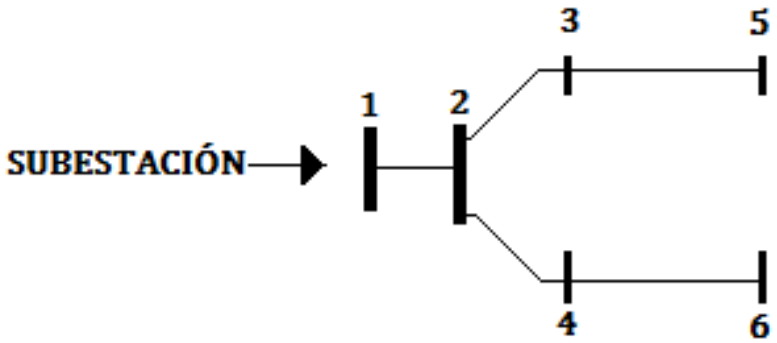


Figura 2-8: Sistema de 6 barras

CASO 1		
S_{base}	1	**MVA**
V_{base}	12,66	**kV**
Z_{base}	160,28	**Ω**
Criterio de convergencia, ε	0,001	**pu**

Tabla 2-3: Configuración predeteminada de la base del sistema de prueba de 6 barras

Caso 1: 6 Barras - Resultados						
Barra No.	**1**	**2**	**3**	**4**	**5**	**6**
Voltaje (V, p.u.)	1,000	0,956	0,928	0,954	**0,919**	0,952
Ángulo de fase (grados)	0,000	0,012	0,017	0,012	0,021	0,013

Tabla 2-4: Resultados de voltaje y ángulo de fase para Caso 1

2.4.2 CASO 2: SISTEMA DE 33 BARRAS – RED PROTOTIPO

Este sistema fue tomado de [23], y con el fin de verificar sus resultados de flujos de potencia, se compararon con los obtenidos en [11]. En la **Figura 2-9**, se muestra el SDR de 33 barras y los valores base de este sistema son iguales a los presentados en la **Tabla 2-3**, es decir, para esta configuración se definió un ε = 0.001 p.u. y S_{base} = 1 MVA.

Los resultados de voltaje para cada barra, se muestran en la **Tabla 2-5**. Basado en la solución de voltaje, las pérdidas totales activa del sistema de potencia fueron calculadas en 211.0 kW, y el mínimo voltaje en magnitud de 0.9038 p.u. en la barra 18.

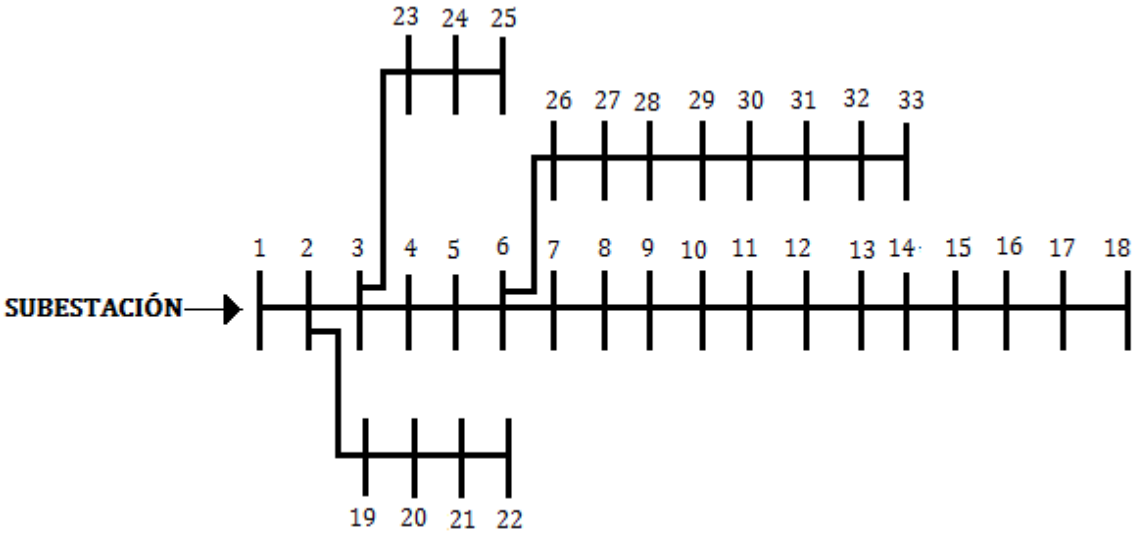


Figura 2-9: Sistema de 33 barras – red prototipo

Caso 2: 33 Barras - Resultados		
Barra No.	**Voltaje (V, p.u.)**	**Ángulo de fase (grados)**
1	1,0000	0,0000
2	0,9970	0,0003
3	0,9829	0,0017
4	0,9754	0,0028
5	0,9680	0,0040
6	0,9495	0,0024
7	0,9460	-0,0017
8	0,9323	-0,0043
9	0,9260	-0,0057
10	0,9201	-0,0068
11	0,9192	-0,0066
12	0,9177	-0,0064
13	0,9115	-0,0081
14	0,9092	-0,0095
15	0,9078	-0,0101
16	0,9064	-0,0105
17	0,9044	-0,0119
18	0,9038	-0,0121
19	0,9965	0,0001

Caso 2: 33 Barras - Resultados		
Barra No.	**Voltaje (V, p.u.)**	**Ángulo de fase (grados)**
20	0,9929	-0,0011
21	0,9922	-0,0014
22	0,9916	-0,0018
23	0,9793	0,0011
24	0,9726	-0,0004
25	0,9693	-0,0012
26	0,9475	0,0031
27	0,9450	0,0040
28	0,9335	0,0055
29	0,9253	0,0069
30	0,9218	0,0087
31	0,9176	0,0072
32	0,9167	0,0068
33	0,9164	0,0067

Tabla 2-5: Solución de voltaje para un sistema de 33 barras

Los resultados obtenidos en magnitud de voltaje, se compararon con los obtenidos en Srinivasa R. [11] y Rost A. [23] y son mostrados en la **Figura 2-10**. Cabe indicar que las pequeñas diferencias obtenidas con los resultados de Srinivasa R., se deben básicamente a que en el ramal 7, este autor utiliza valores diferentes de impedancia.

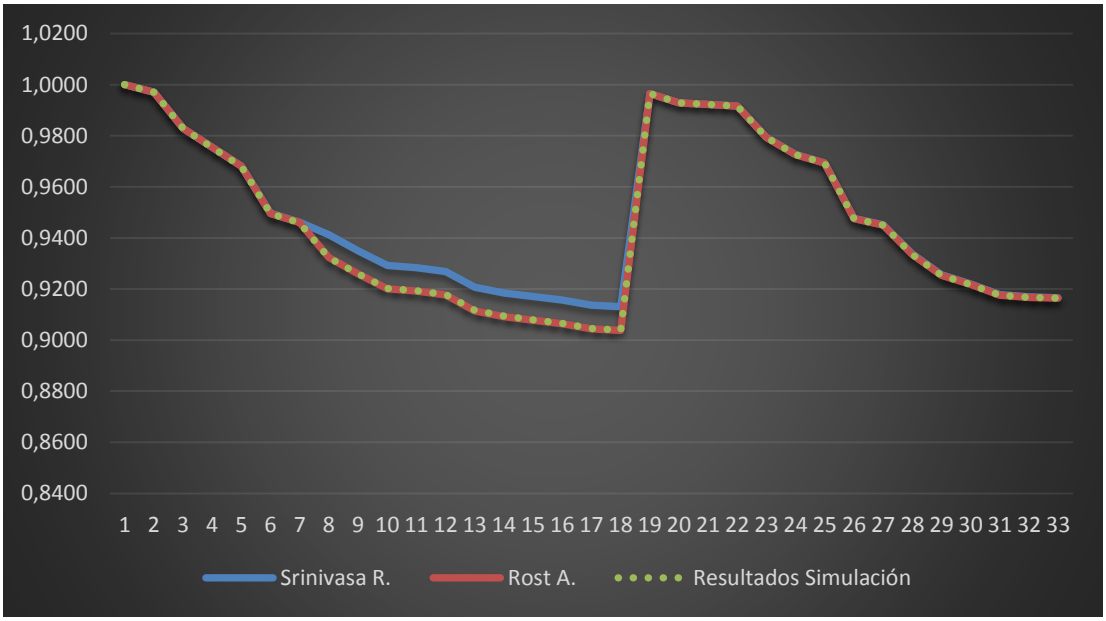


Figura 2-10: Comparación magnitud de voltaje para el sistema de 33 barras

También se simuló con la herramienta computacional MATPOWER, de acuerdo a la metodología explicada en [26] aplicando el método de Newton Raphson donde se verificó que el sistema de 33 barras no converge; el código .m ingresados y sus resultados se muestran en el **ANEXO 2**.

Los datos del SDR de 33 barras ingresados en el programa computacional Matlab y sus códigos, se presentan en el **ANEXO 3**.

CAPÍTULO 3

3 FORMULACIÓN DEL PROBLEMA DE RECONFIGURACIÓN DE REDES DE DISTRIBUCION

3.1 INTRODUCCIÓN

3.1.1 DEFINICIÓN DE UN SISTEMA DE DISTRIBUCIÓN [27]

Un sistema de distribución de energía eléctrica es el conjunto de elementos encargados de conducir la energía desde una subestación de potencia hasta el usuario. Básicamente, la distribución de energía eléctrica comprende los circuitos primarios de distribución, los transformadores de distribución, los circuitos secundarios de distribución y las acometidas y medidores (**Figura 3-1**).

De la definición anterior se puede observar que un sistema de distribución forma parte de un sistema eléctrico, ya que éste comprende la generación, la transmisión y la distribución (**Figura 3-2**). Debido a la complejidad que han alcanzado los sistemas eléctricos de potencia y de distribución, prácticamente en todo el mundo existe una separación en el estudio de ambos. Es decir que por una parte se trata la generación y la transmisión y por otra la distribución. Inclusive en las empresas eléctricas de todo el mundo cada parte es atendida por gerencias diferentes y grupos de ingenieros especializados en cada una de estas ramas de la ingeniería eléctrica. Los dos campos se han especializado de una manera muy profunda, dando como resultado que sea muy difícil tener una persona que domine estas dos áreas del conocimiento a la vez.

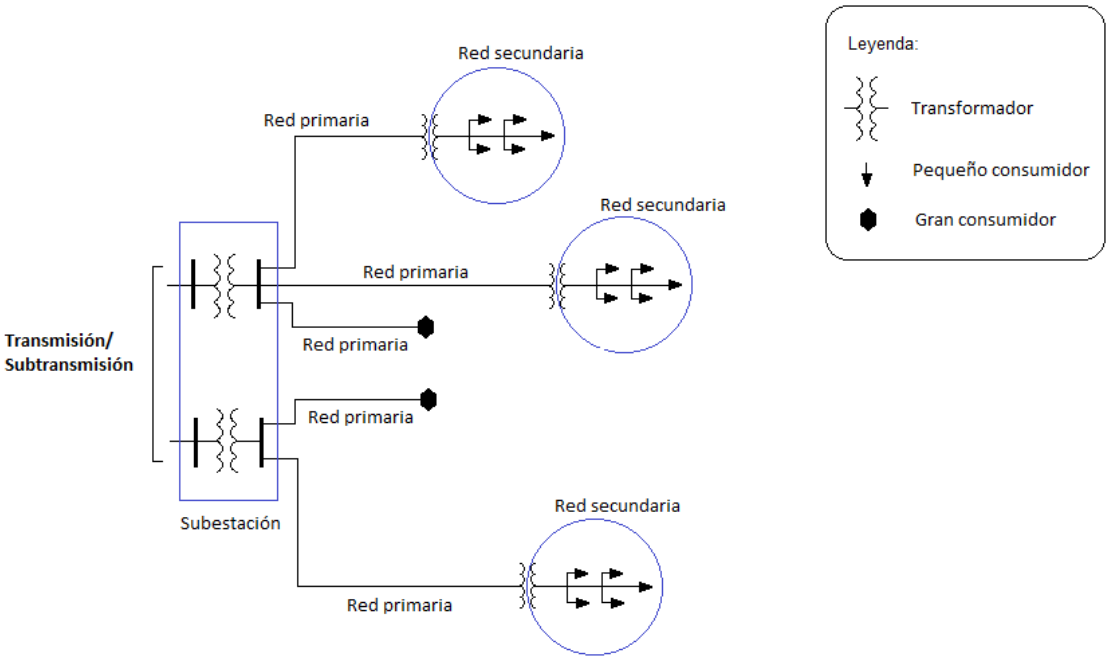


Figura 3-1: Representación de un sistema de distribución de energía eléctrica [28]

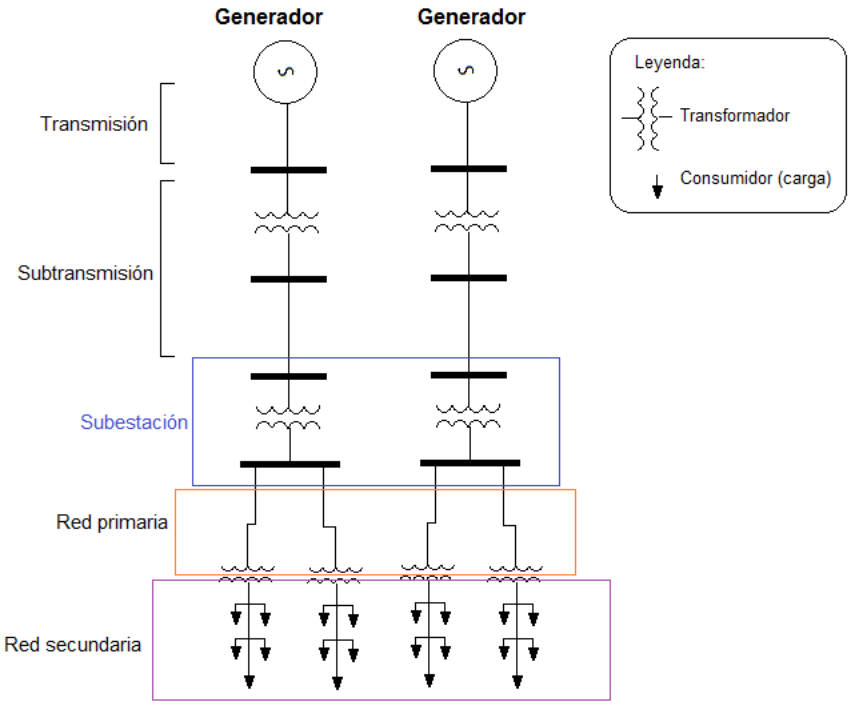


Figura 3-2: Representación de un sistema eléctrico de potencia [28]

3.1.2 OBJETIVO DE LA DISTRIBUCIÓN DE ENERGÍA ELÉCTRICA [27]

La distribución de energía eléctrica debe realizarse de tal manera que el cliente reciba un servicio continuo, sin interrupciones, con un valor de voltaje adecuado que le permita operar sus equipos eléctricos eficientemente, y que la forma de onda senoidal sea pura, es decir que esté libre de armónicas. La distribución de energía eléctrica debe llevarse a cabo con redes bien diseñadas que soporten el crecimiento propio de la carga, y que además sus componentes sean de la mejor calidad para que resistan el efecto del campo eléctrico y los efectos de la intemperie a que se verán sometidas durante su vida útil. Las redes de distribución de energía eléctrica deben ser proyectadas y construidas de manera que tengan la flexibilidad suficiente para ampliarse progresivamente con cambios mínimos en las construcciones existentes, y así asegurar un servicio adecuado y continuo par la carga presente y futura al mínimo costo de operación.

3.1.3 SECCIONADORES PARA RECONFIGURACIÓN REDES DE DISTRIBUCIÓN

Un seccionador es un dispositivo para realizar maniobras de ruptura, o cambio en las conexiones de un circuito eléctrico. Los seccionadores son normalmente divididos en tres clases relativas a la operación: seccionadores de aire, aceite y en vacío [29].

Para el propósito de reconfiguración de redes son utilizados generalmente los seccionadores de aire, los mismos que facilitan el seccionamiento de una línea, el mantenimiento y reparación.

Las características técnicas principales a tomar en cuenta en un seccionador son las siguientes:

Límite de operación.- Este límite está dado por el voltaje, nivel de aislamiento de impulso básico (BIL[14]), corriente nominal, corriente de corta duración, y corriente de interrupción.

[14] Basic-impulse Insulation Level (BIL)

- ***Límite de voltaje***: El límite de voltaje es un valor asignado con el propósito de establecer las características dieléctricas y no debe ser superado en condiciones normales de funcionamiento.
- ***Nivel de aislamiento de impulso básico***: Este es el nivel de aislamiento de referencia expresada como el valor de cresta de impulso de voltaje no disruptiva de una onda de impulso completa especificada.
- ***Límite de corriente continua:*** La capacidad de corriente continua es el límite diseñado de la corriente en amperios que el seccionador llevará continuamente sin exceder un aumento máximo de la temperatura por encima de la temperatura ambiente de 30 °C.
- ***Corriente de interrupción:*** El límite de interrupción indica la máxima corriente de corto circuito que el seccionador puede interrumpir sin un daño sostenible.
- ***Corriente de corta duración:*** Todos los seccionadores tienen una corriente de corta duración, correspondiente a la capacidad del seccionador para llevar la corriente de cortocircuito en la posición cerrada. Este límite incorpora las limitaciones impuestas por los efectos térmicos y electromagnéticos. La corriente de corta duración de un seccionador es la máxima corriente que el interruptor está diseñado para llevar sin daño por los intervalos de corta duración.

En redes de distribución aéreas los seccionadores son montados normalmente en el tope de su estructura de soporte y son operados en conjunto. Ellos son operados por medio de una operación manual o accionados eléctricamente por medio de un mecanismo accionado por un motor. Son usados para ejecutar varias tareas de seccionamiento tales como aislamiento de transformadores y seccionamiento de líneas.

En las siguientes figuras, se aprecian los seccionadores en las redes de distribución de energía eléctrica.

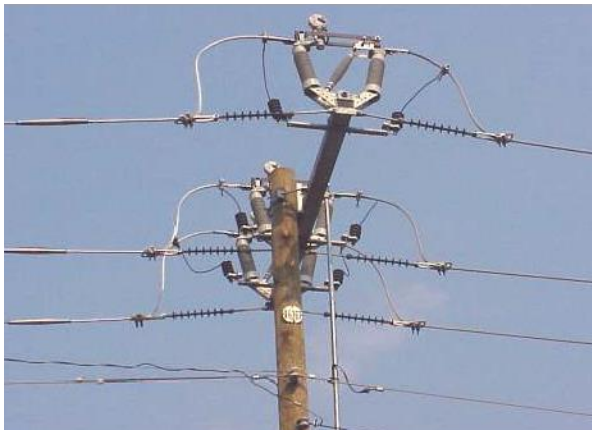


Figura 3-3: Seccionador operado conjuntamente [30]

Figura 3-4: Control manual de un seccionador [30]

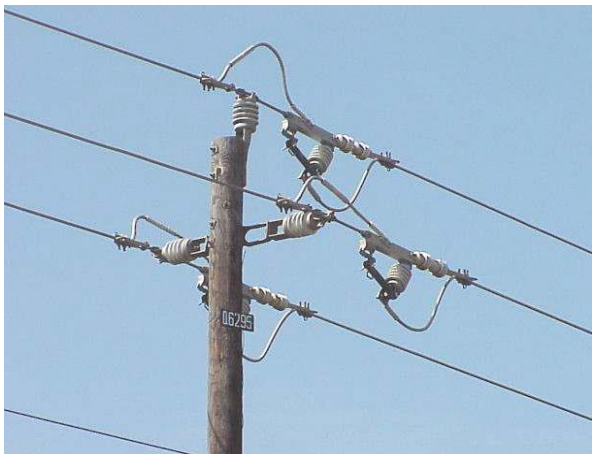


Figura 3-5: Seccionador de rama [30]

Figura 3-6: Seccionador [30]

Algunos seccionadores son equipados con controles de motor y control de supervisión remota, de tal manera que ellos puedan ser operados remotamente. En la siguiente figura se muestra seccionador operado conjuntamente con controles de motor.

Figura 3-7: Seccionador operado conjuntamente con controles de motor [30]

En la actualidad, los seccionadores aislados operados manualmente son reemplazados con seccionadores controlados remotamente en sitios seleccionados.

En el diseño de sistemas de distribución inteligentes, se usan seccionadores electrónicos operados de manera remota, es decir, en tecnologías modernas [31], los componentes electrónicos en sistemas de distribución ofrecen la posibilidad del manejo de la energía en la presencia de recursos renovables distribuidos.

La *"promesa"* de control electrónico en sistemas de distribución, llega por ejemplo con dispositivos con semiconductores de Carburo de Silicio que ofrecen la posibilidad de seccionamiento de alta potencia en altas velocidades [31].

Para hacer que las redes radiales sean más fiables, partes dañadas de los alimentadores pueden ser aislados y fuentes de alimentación alternativos pueden ser conectados por medio de alimentadores medios y seccionadores de líneas instalados en sistemas más avanzados. El funcionamiento de los seccionadores puede ser manual o automático, local o desde un centro de control remoto [32].

Figura 3-8: Control de un seccionador basado en dispositivos electrónicos.

3.2 LA RECONFIGURACIÓN COMO UN PROBLEMA DE OPTIMIZACIÓN – FORMULACIÓN

3.2.1 FUNCIÓN OBJETIVO [5], [23]

El problema de reconfiguración de redes de distribución puede ser formulado como un problema de optimización no-lineal con variables enteras y reales, cuya solución envuelve una selección de entre todas las configuraciones posibles, de aquella que tiene menor pérdida de potencia activa del sistema de distribución de energía eléctrica y que satisface a un conjunto de restricciones. Para el desarrollo de este trabajo, el problema es formulado como:

$$Minimizar\ f(X^k) = \min\left(P_{pérdidas}\right) \quad 3\text{-}1$$

y,

$$P_{pérdidas} = Re\left(V_{ss}\sum\left[\left(V_{ss} - V_j\right)y_{ss,j}\right]^*\right) - \sum P_D \quad 3\text{-}2$$

$$X^k = [\, x_1^k\ ,\ x_2^k\ ,\ x_3^k\ ,\ldots x_p^k\ ] \quad 3\text{-}3$$

Donde [23]:

$f(X)$: Función objetivo de las ramas de la configuración;

V_{ss} : Voltaje en el nodo de la subestación;

V_j : Magnitud de voltaje en algún nodo ligado al nodo de la subestación;

$y_{ss,j}$: Admitancia entre el nodo de la subestación y algún nodo ligado a la subestación;

P_D : Potencia de demanda en cada barra;

X^k : Vector variable de control, configuraciones factibles para una solución kth, X^k=[x_1^k , x_2^k , x_3^k ,. . . x_p^k]. Para un SDR con *p* lazos, x_1^k - x_p^k son las líneas que deben ser abiertas en cada lazo.

Sujeto a a las siguientes restricciones:

- Estructura radial del SDR
- Ecuaciones de flujo de potencia
- Magnitud de voltaje en la barra *i*, $V_{min} \trianglelefteq |V_i| \trianglelefteq V_{max}$

Por lo tanto, el problema de reconfiguración óptima de un sistema de distribución consiste en encontrar aquella configuración radial, del espacio de configuraciones radiales posibles, que produzca pérdidas mínimas de potencia. Este proceso de búsqueda óptima implica analizar implícita o explícita todas las configuraciones radiales posibles.

La modificación de la topología de la red por la reconfiguración, transfiriendo cargas de un alimentador para otro puede mejorar de modo significativo las condiciones de operación de todo el sistema.

3.2.2 RESTRICCIONES

3.2.2.1 ESTRUCTURA RADIAL DEL SDR [5]

Por cuestiones de viabilidad técnica/operacional es común que las redes de distribución (especialmente urbanas) presenten estructuras malladas, siendo la operación del sistema hecha radialmente. Así, son permitidas reconfiguraciones del sistema en caso de

contingencia (por ejemplo, salida de servicio de una línea ó un transformador), en que se busca encontrar una configuración que posea características radiales dependiendo de la localización geográfica y de los criterios de la empresa distribuidora de energía eléctrica, normalmente relacionados con la seguridad del servicio.

La restricción de radialidad es uno de los factores complicados del problema de reconfiguración, pues no es fácil representar esta restricción a través de relaciones algebraicas y, por lo tanto, se torna muy díficil usar algoritmos basados en técnicas tradicionales de optimización. En el problema de reconfiguración se considera que en cada rama existe un seccionador siendo que en las ramas energizadas, los seccionadores están cerrados (seccionamiento) y en las ramas de conexión, los seccionadores están abiertos (interconexión). Así, una operación de maniobra de seccionadores será el que envuelve el intercambio de posiciones de los dos seccionadores, siendo un seccionador que debe ser abierto y otro de interconexión que debe ser cerrado, de tal forma de minimizar las pérdidas de potencia y mantener la radialidad del sistema.

3.2.2.1.1 SECCIONADORES Y DEFINICIÓN DE ESTRUCTURA RADIAL [23]

Para el SDR de 6 barras del capítulo 2, se explica como la restricción de estructura radial es definida, la misma que involucra inicialmente el manejo de seccionadores, como se desarrolla a continuación.

La información del estado de los segmentos de líneas que pueden ser maniobradas está contenido en el vector estado-seccionador (***sws***). Si el valor de un elemento en el vector estado-seccionador representa un segmento de línea que puede ser maniobrado tiene un valor particular de 1, entonces ese segmento de línea está cerrado. De otra manera, un valor de cero en el vector estado-seccionador significa que el segmento de línea correspondiente está abierto.

Al cerrar los seccionadores en un SDR podría crear lazos. Obtener una estructura radial de un SDR puede ser realizado identificando esos lazos y estar seguros que solo un segmento de línea maniobrable está siempre abierto. Considerar el SDR de 6 barras del **ANEXO 1**, donde hay 3 lazos separados dentro del SDR. En la **Tabla 3-1**, se muestran los índices de

los segmentos de línea que pueden ser maniobrados para cada lazo en el SDR de 6 barras y en la **Figura 3-9**, se muestra el diagrama unifilar del SDR de 6 barras con los segmentos de líneas normalmente abiertos y cerrados.

Número de lazos	**Índices de líneas que pueden ser maniobradas**		
1	6	2	3
2	7	4	
3	8	5	

Tabla 3-1: Los índices de los segmentos de línea que pueden maniobrados para cada lazo en el SDR de 6 barras

	Lazo 1			**Lazo 2**		**Lazo 3**	
sws =	0	1	1	0	1	0	1

Tabla 3-2: Vector ***sws*** para SDR de 6 barras

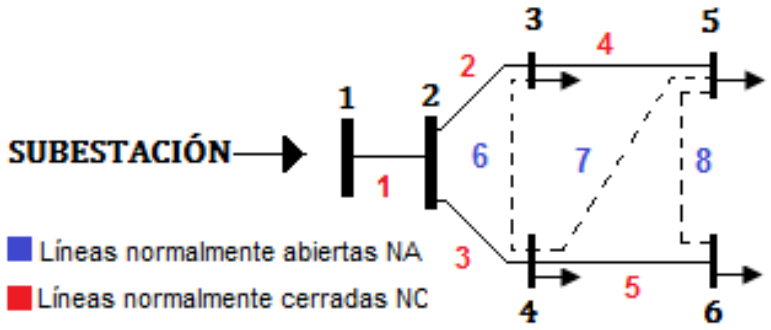


Figura 3-9: SDR 6 barras y segmentos de línea NA y NC

Los seccionadores 6, 7, 8 de los segmentos de líneas normalmente abiertos son escogidos para estar abiertos para forzar la estructura radial en el SDR. Notar que los segmentos de línea normalmente abiertos se ingresan primero para cada lazo en la Tabla 3-1. El resultado del vector ***sws*** se muestra en la Tabla 3-2. El vector ***sws*** solo contiene los valores de 1 y 0. Los primeros 3 elementos del vector ***sws*** resultante corresponden al estado de los seccionadores de los segmentos de líneas que pueden ser maniobrados para el primer lazo. Entonces el valor de 0 para el primer elemento del vector ***sws*** indica que el segmento de línea 6 ha sido abierta (ver Tabla 3-2). Los elementos del vector ***sws*** deben tener solo valores de 1 y 0.

Mantener la estructura radial del SDR, puede ser formulado como una restricción lineal. Por ejemplo, para un lazo particular, *lazo-y*:

$$\sum_{\substack{s=elementos \\ en\ lazo-y}} sws(s) = NSEL_{lazo-y} - 1 \quad 3\text{-}4$$

Donde $NSEL_{lazo-y}$ es un entero que representa el número de segmentos de líneas que pueden ser maniobradas dentro de un lazo específico. Por ejemplo, de acuerdo a los datos de la **Tabla 3-1**, el valor de $NSEL_{lazo-y}$ para el lazo 1 es 3.

De acuerdo a la ecuación 3-4, *lazo- y*, no causará que el SDR tenga una estructura no-radial si la suma de los valores de los segmentos de línea que se pueden maniobrar para el lazo en ***sws*** es igual al total del número de segmentos líneas que se pueden maniobrar en el lazo, menos 1. Entonces la ecuación 3-4 debería dar un valor de 2 para el Lazo 1, para que la configuración del Lazo 1 no cause que el SDR tenga una estructura no-radial. Si la ecuación 3-4 se cumple para todos los lazos dentro de un SDR, la configuración del sistema será radial.

3.2.2.2 Ecuaciones de flujo de potencia [5], [23], [33]

Las restricciones de flujo de potencia analiza directamente el estado de la red, en que las leyes de Kirchhoff ("ley de los nodos" y "ley de las mallas") se deben mantener en equilibrio en todo momento. Con este objetivo se propuso la utilización de un algoritmo para el cálculo de flujo de potencia adecuado, el mismo que fue analizado con más detalle en el Capítulo 2.

Las ecuaciones de balance de carga son restricciones de igualdad, las que se obtienen al imponer una restricción de balance de potencia activa y reactiva en todos los nodos del sistema. En un punto de operación en estado estable, la potencia generada debe ser tal que sea suficiente para cubrir la demanda más las pérdidas en la red.

De otra manera, las ecuaciones de balance de potencia no lineal pueden ser formuladas usando las siguientes ecuaciones:

$$\left(\boldsymbol{P_{gen-i}} - P_{carga-i}\right) - \sum_{j} V_i\, V_j\, Y_{ij} \cos\left(\delta_i - \delta_j - \theta_{ij}\right) = 0 \qquad 3\text{-}5$$

$$\left(\boldsymbol{Q_{gen-i}} - Q_{carga-i}\right) - \sum_{j} V_i\, V_j\, Y_{ij} \cos\left(\delta_i - \delta_j - \theta_{ij}\right) = 0 \qquad 3\text{-}6$$

Donde $\boldsymbol{P_{gen-i}}$ y $\boldsymbol{Q_{gen-i}}$ son la potencia activa y reactiva generada en cada barra; $P_{carga-i}$ y $Q_{carga-i}$ son la potencia activa y reactiva requerida en cada barra; Y_{ij} y θ_{ij} son los valores de admitancia y ángulo de la matriz de admitancia de barra del SDR (YBUS).

3.2.2.3 Magnitud de voltaje

3.2.2.4 Nivel de voltaje [5], [23], [33]

El nivel de voltaje en el consumidor es determinado por el voltaje en la subestación y por las caídas de tensión en las líneas de distribución (puntos de consumo) y transformadores, variando con fluctuaciones en los niveles de consumo.

Normalmente, se dividen las caídas de tensión en dos grupos: fluctuaciones rápidas y fluctuaciones lentas. Las variaciones lentas no pueden ser notadas inmediatamente, siendo causadas, principalmente, por variaciones graduales en los consumos y por fluctuaciones en el voltaje de las barras de las subestaciones. Este tipo de caída tiene un efecto considerable en la eficiencia y tiempo de vida útil de los equipos eléctricos de los consumidores. Normalmente es impuesto un límite máximo de desvío en relación al voltaje nominal (caída de voltaje) en las fluctuaciones de voltaje en las redes de distribución.

Las fluctuaciones de voltaje lentas tienen, entonces, una gran importancia en la calidad de servicio y, consecuentemente, son un factor a considerar en el planeamiento y en la operación de los sistema de distribución. De esta forma, la propuesta de reconfiguración del sistema que tuvieran los voltajes en una barra del sistema o conjunto de barras debajo de un valor predeterminado en relación a el voltaje de la subestación deben ser penalizadas.

$$V_{min} \leq |V_i| \leq V_{max} \qquad 3\text{-}7$$

Donde,

V_i : Voltaje evaluado en la *i*-ésima barra del sistema;

V_{min}, V_{max} : voltajes máximos y mínimos de la barra; (Vmin = 0.95 p.u y Vmax = 1.05 p.u)

3.2.2.4.1 Perfiles de Voltaje

Como se indicó anteriormente el voltaje en los nodos es uno de los criterios de seguridad e índice de calidad de servicio más importante, incluir una restricción que mejore el perfil de voltaje de los nodos de carga del sistema es un aspecto importante a ser considerado en el problema de optimización.

CAPÍTULO 4

4 ALGORITMO DE OPTIMIZACIÓN DE MAPEO MEDIA – VARIANZA

4.1 INTRODUCCIÓN: MVO[15], MVMO CLÁSICO Y MVMO ENJAMBRE[16] MVMO[S]

La optimización con MVMO para resolver el problema de optimización de reconfiguración en redes de energía eléctrica no ha sido aplicada en trabajos anteriores, siendo, este el primer estudio donde se prueba la factibilidad del mismo y la adaptación requerida para resolver este problema.

La Optimización por Mapeo Media Varianza, originalmente propuesto en [34] es una reciente adición a los algoritmos de optimización emergentes con algunas similitudes conceptuales básicas a otros enfoques heurísticos.

En optimización de sistemas de potencia, MVMO ha sido satisfactoriamente aplicado para resolver el despacho óptimo de potencia reactiva de parques eólicos y planeamiento óptimo de la expansión de la transmisión.

Su principio de funcionamiento es basado en una función de mapeo especial aplicada a la mutación del producto de los procesos de reproducción (descendencia, hijos) en base del promedio y la varianza del conjunto comprendido de las n-mejores soluciones obtenidas hasta ese momento y guardadas en un archivo de memoria que se actualiza continuamente entre una iteración y otra. Un rasgo notable de la implementación clásica de MVMO constituye su enfoque en la evolución a una sola partícula dentro de un espacio de búsqueda

[15] A Mean – Variance Optimization Algorithm
[16] SWARM

donde se normalizan todas las variables de decisión/optimización desde sus rangos originales [min,max] a un rango de [0,1], donde la compensación entre los resultados de búsqueda[17] de intensificación[18] y la diversificación[19] resulta en velocidades de progresos rápidos con un riesgo reducido de convergencia prematura.

La contribución novedosa de MVMO es la función de mapeo especial. La forma y localización de la curva de mapeo son ajustadas de acuerdo al proceso de búsqueda. Las entradas (parámetros calculados para cada variable de decisión a partir de la información registrada en el archivo de memoria) y la salida (valor mutado de la variable de decisión) de la función de mapeo están en el rango [0,1]. Esto significa que MVMO puede garantizar la no violación de los límites de la variable durante el proceso de búsqueda. Por otra parte MVMO actualiza la solución candidata alrededor de la mejor solución en cada iteración. Como se indicó anteriormente, gracias al balance bien diseñado entre la búsqueda de intensificación y diversificación, MVMO puede encontrar el óptimo rápidamente con el mínimo de convergencia prematura.

El algoritmo de optimización de mapeo media varianza, opera en una solución única antes que en un conjunto de soluciones como algunos EAs (algoritmos evolucionarios). El espacio de búsqueda interna de todas las variables en MVMO es restringida a [0, 1]. Por lo tanto, los límites max/min reales de las variables tienen que ser normalizadas a 0 y 1. Durante la iteración, no es posible que algunos componentes del vector solución violen los límites correspondientes. Para lograr este objetivo, una función de mapeo especial es desarrollada. Las entradas de esta función son el promedio y la varianza de las mejores soluciones que MVMO ha descubierto hasta ese momento. La propiedad elegante del MVMO es su habilidad para buscar alrededor de la mejor solución encontrada hasta ese momento, con una pequeña posibilidad de quedar atrapado en uno de los óptimos locales. Esta característica se debe a la estrategia de manejar el concepto de varianza cero.

[17] Estrategia de búsqueda: se diseñan balanceando estática o dinámicamente el aprovechamiento de la experiencia acumulada de búsqueda (intensificación) y la exploración del espacio de búsqueda (diversificación) [32].

[18] Mecanismo de intensificación: exploración local [32].
[19] Mecanismo de diversificación : exploración global [32]

Luego de haber realizado una breve introducción del MVMO clásico, a continuación, se describe MVMO Enjambre ($MVMO^S$), el cual, ha sido utilizado para resolver el problema de despacho óptimo de potencia reactiva (ORDP) [35] con resultados satisfactorios. Este nuevo alcance extiende el poder innato de la búsqueda global del original MVMO iniciando la búsqueda con un conjunto de partículas (es decir, enjambre), teniendo cada una su propia memoria (archivo de actualización continuo) y representado por el registro correspondiente y función de mapeo, permitiendo el intercambio de información y reducción dinámica del tamaño del enjambre a través de reglas simples. En $MVMO^S$, también se incorporan un esquema mejorado para la asignación del factor de forma y una penalización dinámica.

4.2 CLÁSICO MVMO [35]

El flujograma de la implementación de MVMO (alcance a una simple partícula) se muestra en la **Figura 4-1**. El procedimiento comienza con una fase de inicialización donde los ajustes de parámetros del algoritmo son definidos y muestras aleatorias para las variables de control del espacio de soluciones posibles son generadas. A continuación, un lazo iterativo es iniciado, en el cual la evaluación de la aptitud (ejemplo, función objetivo más la penalización como una función del grado de cumplimiento de la restricción) es ejecutado, el criterio de finalización es comprobado, la solución almacenada (ejemplo, inclusión o exclusión de soluciones candidatas) es actualizado, la mejor solución global es determinada (ejemplo, asignación de los padres), y nuevas soluciones candidatas son creadas (ejemplo, mutación por proyección de variables seleccionadas sobre la función de mapeo y cruce).

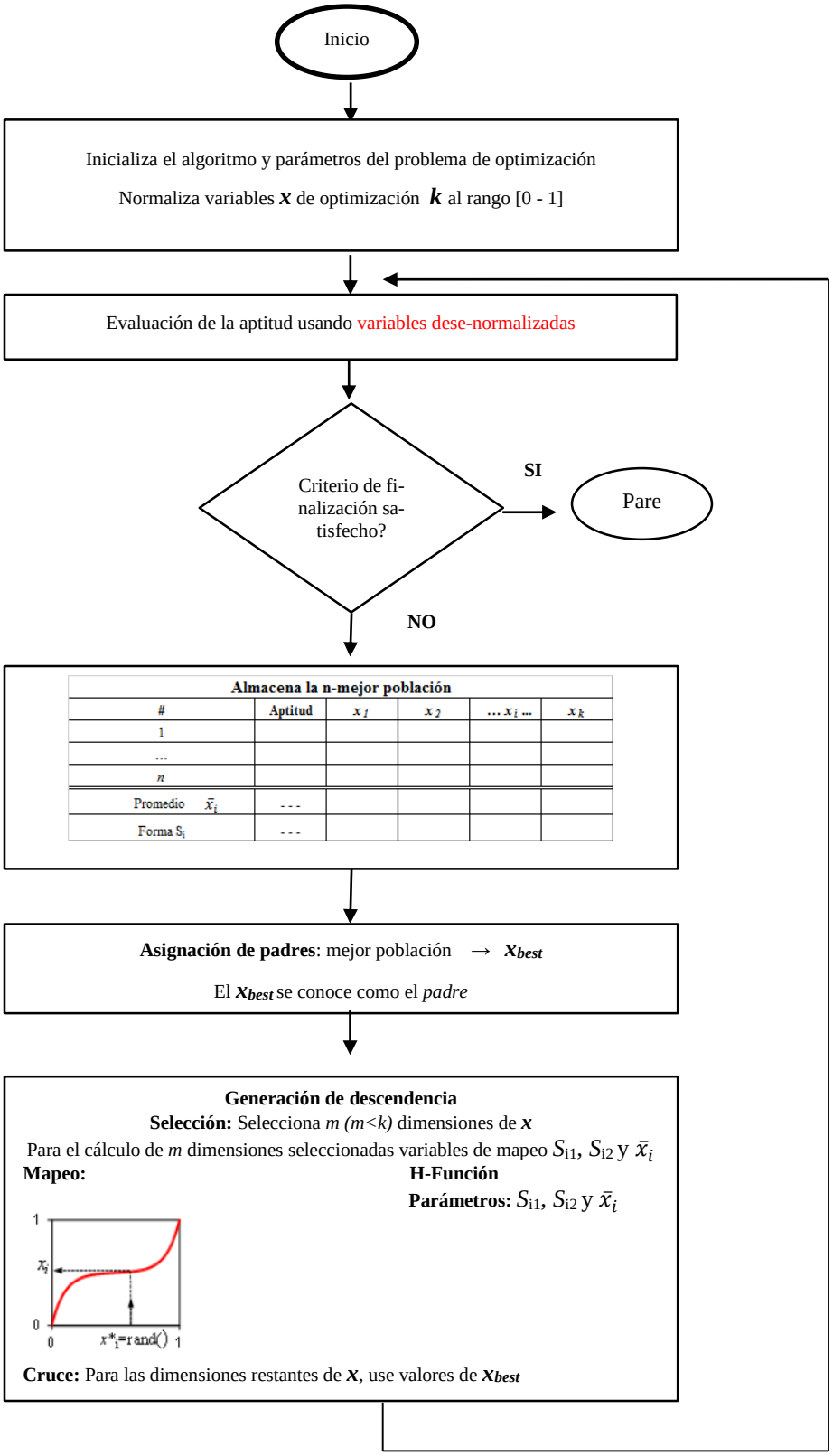


Figura 4-1: MVMO clásico, enfoque a una sola partícula [35]

Las características sobresalientes del algoritmo pueden ser resumidas como sigue:

- Una novedosa función de mapeo es usada para mutar genes en la descendencia (hijos) basándose en la media y la varianza de la solución almacenada.
- Un archivo de solución actualizado dinámicamente y compacto que sirve como la base de conocimiento para guiar la dirección de la búsqueda (ejemplo, memoria adaptiva). Los n-mejores individuos que MVMO ha encontrado hasta el momento son almacenados en el archivo y ordenados en orden descendente de aptitud.
- El concepto de par simple padre-descendencia (hijo) es adoptado. En contraste a otros métodos, donde el término iteración generalmente se refiere al número de evaluaciones de la aptitud (lo cual es proporcional al número total de individuos en el enjambre). MVMO requiere solo una evaluación de la aptitud por iteración independientemente del número de individuos almacenados en el archivo de solución.
- La restricción del rango del espacio de búsqueda para todas las variables de optimización internamente es [0,1]. Esto es una precondición para usar la función de mapeo; en el tope de esto, garantiza que la descendencia generada está siempre dentro los límites de búsqueda. Sin embargo, la evaluación de la aptitud es llevada fuera en la dimensión física original.

4.2.1 MEJORAS EN EL MVMO

a. Mapeo mejorado

El individuo con la mejor aptitud hasta el momento en el archivo (primera posición) es usado en cada iteración para generar un nuevo descendiente (ejemplo, padre asignado). Básicamente, *m* fuera de las dimensiones de *k* del problema de optimización son estratégicamente seleccionados para la operación de mutación mediante la función de mapeo mientras las dimensiones restantes heredan los valores correspondientes del padre. Los métodos de selección alternativos se muestran en la **Figura 4-2**, y se explican con más detalle en el siguiente numeral.

Para seleccionar la variable, MVMO realiza la búsqueda alrededor de la media almacenada en el archivo para la mejor solución solamente en las direcciones seleccionadas de *m*. Esto

significa que solo esas dimensiones seleccionadas *m* de la descendencia serán actualizadas mientras las dimensiones restantes *D-m* toman los valores correspondientes de $\mathbf{x}_{best}$. Cuatros estrategias para seleccionar las variables fueron implementadas en MVMO (**Figura 4-2**). De la experiencia en otros trabajos, las estrategias 2-4 generalmente se desempeñan mejor [36].

Como padre de la nueva población se utiliza la mejor población guardada en el archivo (primera posición). Luego se seleccionan unas pocas variables y son reflejadas en la función de mapeo.

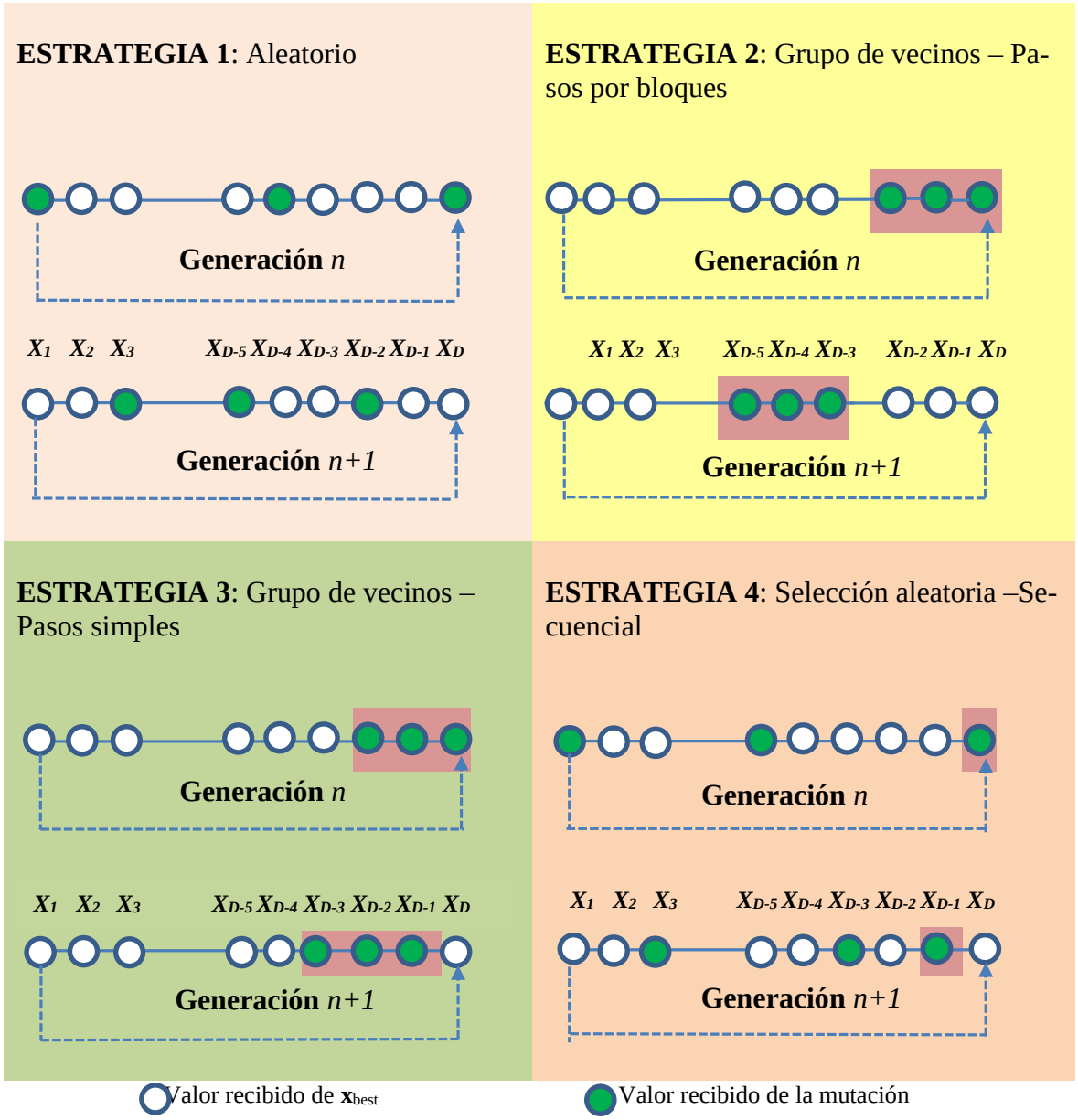


Figura 4-2: Selección de Estrategias [36], [37]

Igual que en MVO, el nuevo valor de cada dimensión seleccionada x_i es determinada por la ecuación 4-11, donde x_i^* es una variable modificada aleatoriamente con distribución uniforme entre 0 y 1 y el término h se refiere a la transformación de la función de mapeo, definido en 4-10. Las variables h_x, h_1 *y* h_0 (4-12) son las salidas de la función de mapeo y S_i es la *variable de forma* (4-9), variables que son definidas en el numeral 4.3

Al inicio, el promedio $\overline{x}$ corresponde con el valor inicializado de x_i y la varianza v_i (asociado a S_i) y es definido a uno. En el progreso de la optimización, los parámetros son recalculados después de cada actualización del archivo para cada variable de optimización seleccionada. La entrada y la salida de la función de mapeo cubre el rango [0,1]. Notar que la forma de la función de mapeo es influenciada por $\overline{x}$ y los factores de forma S_1 y S_2. El efecto de esos parámetros se muestran en las Figura 4-5, Figura 4-6 y Figura 4-7, de donde se deduce fácilmente que la diversidad de la búsqueda puede ser mejorada a través de asignar apropiadamente las variables de forma. Notar también que, incrementando el valor del parámetro de forma, la curva de mapeo llega a ser más plana tal que el espacio a ser buscado se centra en la región cercana al valor medio.

En la ecuación 4-9, es evidente que el factor f_S puede ser usado para cambiar la forma de la función. Un valor pequeño (ejemplo, entre 0.5 y 1.0) permite que la pendiente de la curva de mapeo incremente y así habilita una mejor exploración, mientras los valores sobre 1.0 resultarán en una curva plana y así conduce a mejorar la explotación. Generalmente, es recomendado comenzar el proceso de búsqueda con un pequeño f_S y entonces ir incrementándolo conforme la optimización avanza. En varias aplicaciones la variación aleatoria de f_S de acuerdo a 4-1 ha resultado en mejoras significativas.

$$f_S = f_S^* . (1 + \boldsymbol{rand}) \qquad 4\text{-}1$$

Donde f_S^* es el valor más pequeño de f_S y $\boldsymbol{rand}$ es un número aleatorio en el rango de [0,1]. Cuando la exactitud de la optimización necesita ser mejorada, la siguiente extensión puede ser añadida para permitir un incremento progresivo de f_S^* :

$$f_S^* = f_{S\_ini}^* + \left(\frac{i}{i_{final}}\right)^2 \left(f_{S\_final}^* - f_{S\_ini}^*\right) \quad 4\text{-}2$$

La variable *i* representa el número de iteración. Si $f_{S\_final}^* = f_{S\_ini}^*$, el factor f_S es fijado a lo largo de la optimización. Es recomendado definir $f_{S\_ini}^*$= 0.9.….1.0 y $f_{S\_final}^*$ = 1.0 ….3.0.

A diferencia de MVO, los factores de forma S_1 y S_2 de la variable x_i no son calculadas directamente de (4-9), en MVMO se usa en el siguiente procedimiento:

$S_{i1} = S_{i2} = S_i$
if $S_i > 0$ then

$\Delta d = (1 + \Delta d_0) + 2 \,.\, \Delta d_0 \,.\, (rand - 0.5)$
if $S_i > d_i$

$d_i = d_i \,.\, \Delta d$

else

$d_i = d_i \,/\, \Delta d$

end if 4-3

if *rand* $\geq$ 0.5 then

$S_{i1} = S_i \;;\; S_{i2} = d_i$

else

$S_{i1} = d_i \;;\; S_{i2} = S_i$

end if

end if

Los valores iniciales de d_i son definidos para todas las variables al inicio de la optimización. Las experiencias han demostrado que los valores alrededor de 1 - 5 garantizan un buen desempeño inicial. En cada iteración, cada d_i es escalado hacia arriba o abajo con el factor Δd. Si $d_i > S_i$, el valor actual d_i es dividido para Δd el cual es siempre mayor que 1.0 y por

lo tanto conduce al valor reducido de d_i. En caso de que $d_i < S_i$, d_i será multiplicado por Δd resultando en d_i incrementado. Por lo tanto, d_i siempre oscilará alrededor del factor de forma actual S_i . Por otra parte, Δd es variado aleatoriamente alrededor del valor $1 + \Delta d_0$ con la amplitud de Δd_0 ajustado de acuerdo a 4-4.

$$\Delta d_0 = \Delta d_0^{ini} + \left(\frac{i}{i_{final}}\right)^2 \left(\Delta d_0^{final} - \Delta d_0^{ini}\right) \qquad 4\text{-}4$$

Experiencias hasta ahora han mostrado un excelente desempeño permitiendo el ancho de banda del factor Δd_0 decrecer cuadráticamente en el rango $0.40 > \Delta d_0 > 0.01$. Un alto Δd_0 implica una diversificación de búsqueda global más amplia sobre todo el espacio mientras una más pequeña podría conducir al espacio de búsqueda concentrado ayudando en una mejora de exactitud. Utilizando el procedimiento descrito la característica asimétrica de la función de mapeo es también explotada completamente usando diferentes valores para S_{i1} y S_{i2} conduciendo al mejoramiento del desempeño de búsqueda (ejemplo, robustez) y manejo de varianza cero. Varianza cero puede ocurrir cuando todos los valores de x_i en el archivo son idénticos. En este caso el valor previo de no-cero además puede ser usado. Sin embargo, este valor puede resultar, bajo circunstancias, en el estancamiento del comportamiento de convergencia. El procedimiento de acuerdo a (4-3) y (4-4) supera este problema muy bien. Por la misma razón el promedio y la varianza son calculados para valores diferentes de x_i almacenado solamente en el archivo.

El promedio y la varianza no son calculados antes de que un cierto número de soluciones estén disponibles en el archivo. Usualmente, terminan siendo calculadas inmediatamente después de que dos soluciones han sido almacenadas. Sin embargo, podría también decidirse hacer esto una vez que el archivo esté lleno completamente, lo cual resultará en más soluciones iniciales robustas. En esta etapa, la búsqueda es ejecutada con $S_{i1} = S_{i2} = 0$ que corresponde con una línea recta entre cero y uno como la función de mapeo. El valor promedio en este caso no tiene ningún efecto en la función de mapeo. La **Figura 4-3** ilustra el efecto del procedimiento descrito en la forma de mapeo.

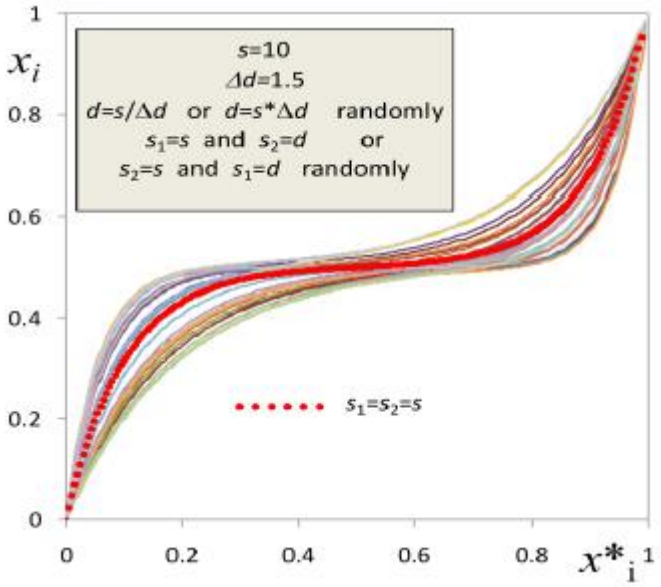


Figura 4-3: Variación de la forma de la función de mapeo con el procedimiento de asignación de factor de forma propuesto

b. Manejo de restricciones

Cada solución potencial es evaluada a la luz de su medición de aptitud. Esto envuelve el cálculo de flujo de potencia para calcular la función objetivo así como determinar que se cumplan las restricciones, lo cual es ejecutado usando los valores actuales en el espacio del problema. Un esquema de penalización dinámica es definido para considerar apropiadamente el grado de cumplimiento junto con las dinámicas del proceso de búsqueda. La aptitud es calculada de la siguiente manera:

$$f' = f + \rho \,.\, n_{cv} \,.\, \sum_{i=1}^{n} max[0, \mathrm{g}_i]^2 \qquad 4\text{-}5$$

Donde f es el valor de la función objetivo, n_{cv} es el número de violaciones de las restricciones, g_i expresa la restricción i-th, y ρ es un factor de penalización adaptado determinado por:

$$\rho \begin{cases} = \rho_{ini} \ \ siempre\ y\ cuando\ ninguna\ solución\ factible\ para\ \ \mathbf{x}\ es\ obtenida, \\ = \rho_{ini} + \left(\dfrac{i - i_{first}}{i_{final} - i_{first}}\right)^2 (\rho_{final} - \rho_{ini}) \ \ i > i_{first} \end{cases}$$

i_{first} = conteo de iteraciones cuando la primera solución factible es encontrada,

$i_{first} = i_{max}/2$ si la solución no es factible alcanzando hasta $i \trianglelefteq i_{max}/2$

ρ_{ini} podría ser asignado a un valor bajo (ejemplo, 1e3) para estáticamente imponer bajas penalizaciones para las soluciones no factibles durante la etapa de búsqueda, pero una vez que son encontradas soluciones factibles, el factor de penalización aumenta cuadráticamente con la iteración número *i* desde ρ_{ini} a ρ_{final} (un valor alto pre-especificado, ejemplo 1e6) para incrementar la presión con el avance de la búsqueda para conducir los descendientes de las mejores soluciones factibles hacia el óptimo.

4.3 ALGORITMO MVO [34]

Los pasos básicos a seguirse en el algoritmo MVO son descritos a continuación:

a. Inicialización del Algoritmo MVO y Normalización de Variables

El algoritmo MVO y parámetros del problema de optimización tienen que ser inicializados. Los pocos parámetros del algoritmo MVO tienen que ser inicializados, los cuales incluyen:

- Tamaño de población dinámica, *n*
- Número de dimensiones (variables del problema), *m*, a ser seleccionados para la mutación.
- Selección del método usado

Además parámetros opcionales usados en la transformación

- Factor de escalamiento de forma, f_s
- Factor de Asimetría *AF*

- Valor inicial del factor de forma S_d

El número de variables del problema, *k*, a ser optimizadas tienen que ser inicializadas dentro de los límites permitidos. Esto puede ser realizado por el usuario; de otra manera las variables son inicializadas de manera aleatoria.

El rango del espacio de búsqueda para las variables de optimización dentro del algoritmo MVO es [0,1]. Sin embargo, la evaluación de la aptitud es llevada fuera usando los valores actuales en el espacio del problema (espacio de la evaluación aptitud). Aunque, durante la optimización, la desnormalización es llevada fuera en cada simple iteración.

b. Criterio de finalización

El proceso de búsqueda MVO se puede terminar sobre la base de una realización de un número especificado de iteraciones (evaluaciones de la aptitud), la solución $\boldsymbol{x}$, alcanzando una aptitud deseada o ninguna mejora en la aptitud de las últimas iteraciones. El criterio de finalización es determinado por el usuario para un determinado problema de optimización. En este contexto, el algoritmo MVO no difiere de algunos algoritmos de optimización heurísticos. Debe notarse que el número de iteraciones en el algoritmo MVO es equivalente al número de evaluaciones de descendencia de la aptitud.

c. Población dinámica

El MVO utiliza un simple concepto de la pareja padre-hijo (descendencia) pero incorpora información de desempeño de los *n* mejores individuos almacenado en el archivo de memoria a través de la media y la varianza. La dimensión *n* de la población, se considera para un número mínimo de soluciones a almacenar en el archivo de memoria de dos, debido a que con una no es posible calcular la media y varianza de cada variable de decisión. Para factorizar el desempeño de la población en las operaciones del MVO (descritas abajo), un mínimo de dos soluciones aleatorias son necesarias al inicio. Si *n* es más grande que dos la tabla de los mejores individuos es llenada progresivamente en un orden descendiente de la aptitud. Cuando la tabla es llenada con *n* miembros una actualización es ejecutada solo si la

aptitud de la nueva población es mejor que esos en la tabla. La aptitud mejora con las iteraciones, los miembros de la población van cambiando, es decir, es un proceso dinámico.

La media $\bar{x}_i$ y la varianza, v_i, son calculados después de cada actualización del archivo para cada dimensión usando 4-6 y 4-7, respectivamente.

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_i(j) \qquad 4\text{-}6$$

$$v_i = \frac{1}{n}\sum_{j=1}^{n} (x_i(j) - \bar{x}_i)^2 \qquad 4\text{-}7$$

Donde, *j* va desde 1 a *n* (tamaño de la población). Al comienzo $\bar{x}_i$ corresponde con el valor inicializado de x_i y la varianza es definida para $v_i = 0$.

Un pequeño tamaño de la población dará lugar a centrarse en la exploración, mientras que un tamaño grande conduce a una mejor explotación.

d. Asignación de padres

El individuo con la mejor aptitud (primera posición en la tabla), y sus valores de optimización correspondientes, $\boldsymbol{x}_{best}$, son almacenados en la memoria como el *"padre"* de la población para esa iteración. Generalmente, es posible usar cualquier individuo almacenado en la tabla o sus combinaciones como padre.

e. Creación de la descendencia

La creación de una descendencia, de *k* dimensiones envuelve tres operaciones comunes de los algoritmos evolucionarios, llamados: selección, mutación y cruce.

Selección – m de *k* dimensiones del problema de optimización son seleccionados para la operación de mutación. Las siguientes estrategias de selección fueron estudiadas inicialmente:

1) Selección aleatoria
2) Selección del grupo vecino
   a) moviendo el grupo hacia adelante en múltiples pasos
   b) moviendo el grupo haciendo adelante en pasos individuales
3) Selección secuencial de la primer variable y el resto, aleatoriamente.

Diferentes estrategias pueden ser aplicadas para la selección y la estrategia correcta depende del problema de optimización.

Mutación – En primer lugar los valores de las dimensiones *m* seleccionadas son aleatorias de una distribución uniforme en el rango [0,1].

$$x_i^{'} = random() \qquad 4\text{-}8$$

Los valores aleatorios generados son transformados basados en la media y la varianza de la mejor población *m* a lmacenada en la tabla del archivo. La transformación y la función correspondiente son las características claves del algoritmo MVO.

La función de transformación, *h*, es introducida basándose en el factor de forma y la media, dada en (4-9).

$$S_i = -\ln(v_i) \cdot f_S \qquad 4\text{-}9$$

El factor de escalamiento f_S permite controlar el proceso de búsqueda. Un valor pequeño de f_S (entre 0.9 y 1.0) permite la pendiente de la curva de mapeo incrementarse y por lo tanto una mejor exploración. Valores de f_S sobre 1.0 (hasta 10) resultarán en una curva plana y

por lo tanto una explotación mejorada. El parámetro f_S permite también definir varias pendientes S_{i1}, S_{i2} para enfocarse en el espacio a ser buscado por debajo o aún por encima del valor medio.

Dimensiones *m* de las variables generadas aleatoriamente son transformados usando (4-10) y (4-11).

$$h(\bar{x}_i\ S_{i1},S_{i2},\ u_i) = \bar{x}_i\ .(1 - e^{-u_i S_{i1}}) + (1 - \bar{x}_i).e^{(1-u_i).S_{i2}} \quad 4\text{-}10$$

$$x_i = h_x + (1 - h_1 + h_0)\,.x_i^{'} - \ h_0 \quad 4\text{-}11$$

Donde,

$$h_x = h(u_i{=}\ x_i^{'}),\ h_0 = h(u_i{=}\ 0),\ h_1 = h(u_i{=}\ 1) \quad 4\text{-}12$$

La función de transformación es mostrada en la **Figura 4-4**.

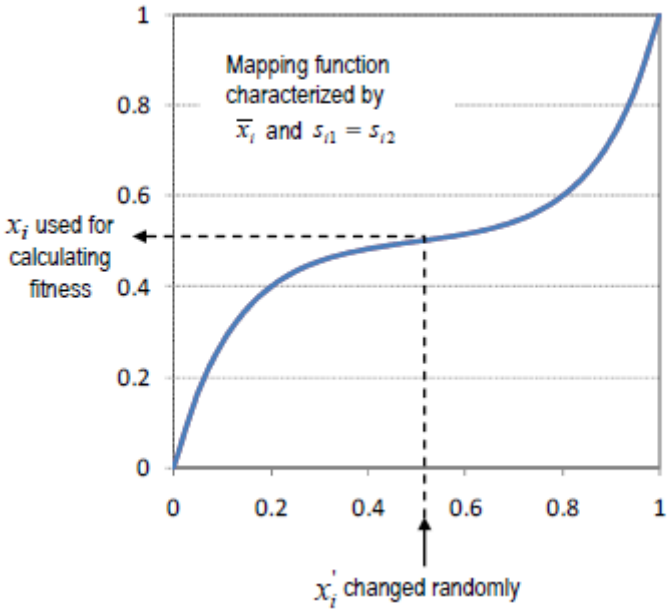


Figura 4-4: Figura de función de mapeo y transformación.

Extendiendo la función *h(*)* por h_1 y h_0 de acuerdo a 4-11 el rango de salida de la función llega a ser exactamente [0,1] como el caso para la entrada. Los efectos de la media y el factor de forma en la función de transformación son mostrados en las Figura 4-5 y Figura 4-6.

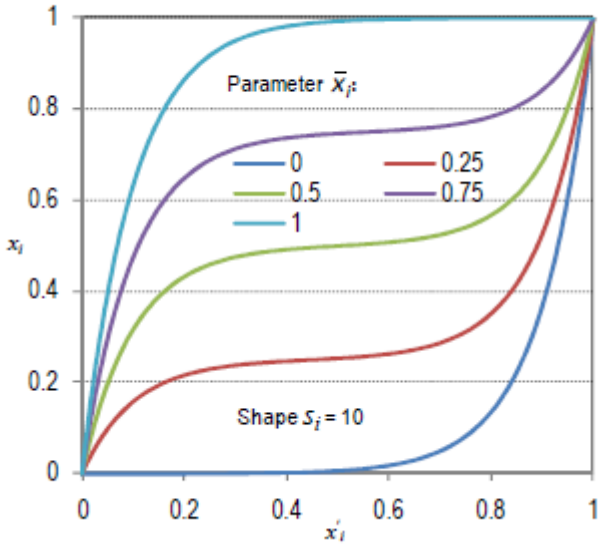


Figura 4-5: Efectos de la media de la población dinámica en la función de transformación *h*.

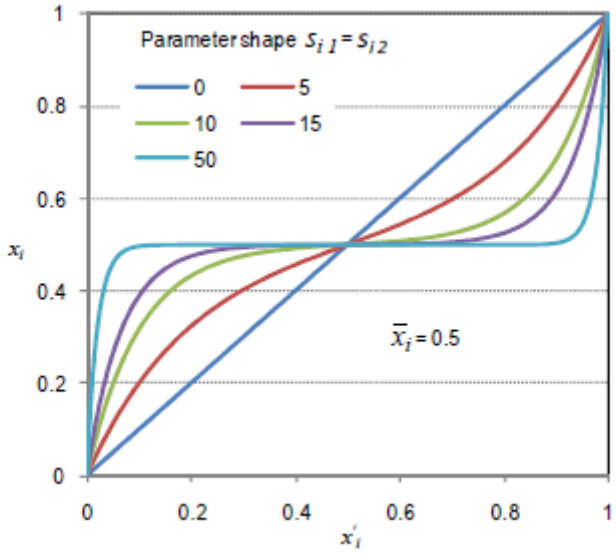


Figura 4-6: Efectos del factor de escalamiento de forma en la función de transformación *h*.

Como se puede observar, al decrecer la varianza entonces se incrementa el factor de forma, la curva correspondiente es cada vez más plana tal que el espacio a ser buscado es centralizado en la región cercano al valor medio. Sin embargo, todas las curvas arrancan y terminan en cero y uno, respectivamente. Por lo tanto, la búsqueda aún cubre el espacio completo entre los límites min/max, a pesar de que la probabilidad de ser seleccionado cerca del borde es menor.

El efecto de diferentes factores de forma $S_{i1} \neq S_{i2}$ es demostrado en Figura 4-7. Obviamente, escogiendo diferentes factores de forma el espacio a ser buscado puede ser controlado por el cambio de énfasis al lado preferido de la curva alrededor del valor medio. El algoritmo implementado por los autores para utilizar esta alternativa es de la siguiente manera:

$$Si\ x_i^{best} < \bar{x}_i \rightarrow S_{i2} = S_i\,.AF\ ;\ S_{i1} = S_i$$
$$else\ if\ \ x_i^{best} > \bar{x}_i \rightarrow S_{i1} = S_i\,.AF\ ;\ S_{i2} = S_i$$

Donde *AF* representa el factor de asimetría en el rango de [1-10].

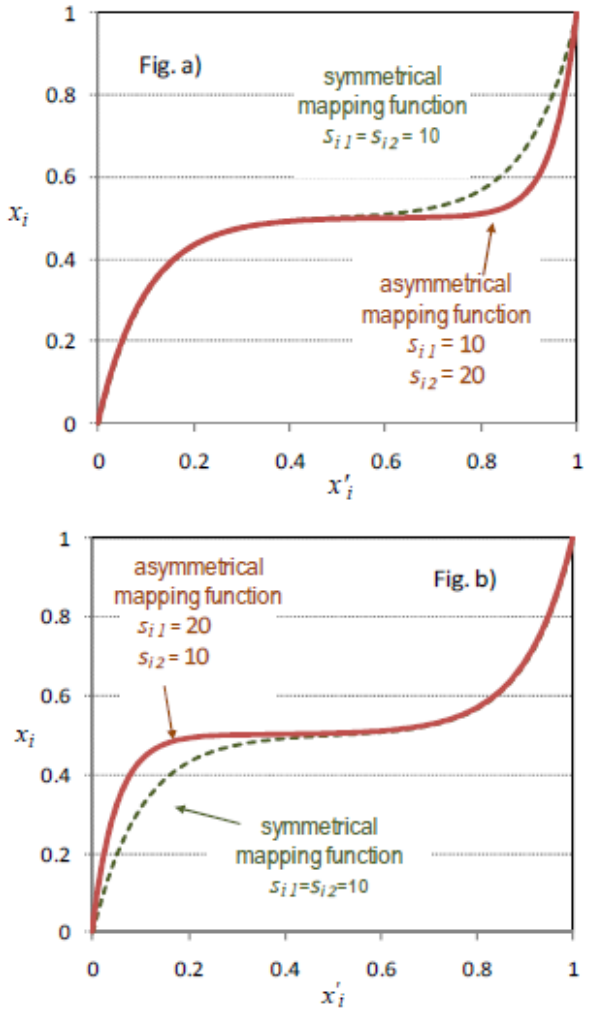


Figura 4-7: Efecto de diferentes factores de forma $S_{i1} \neq S_{i2}$

Cruce – Para las dimensiones no-mutadas restantes los genes del padre, $\boldsymbol{x_{best}}$, son heredadas. En otras palabras, los valores de esas dimensiones no mutadas son clones del padre. Aquí, el cruce es por clonación directa de ciertos genes. En este camino la descendencia es creada combinando el vector, $\boldsymbol{x_{best}}$, y el vector de dimensiones mutado *m*.

Tratamiento de varianza cero – Varianza cero puede ocurrir cuando todos los individuos almacenados en el archivo no difieren con respecto a las variables de optimización particulares. En consecuencia, el factor de forma de la curva de mapeo tiende a cero y la optimización no progresa. La probabilidad de varianza cero aumenta a medida que el número de variables mutadas *m* es pequeña.

Uno de los enfoques para resolver este problema es usar además la última varianza no-cero.

4.4 TEORÍA DE INTELIGENCIA DE ENJAMBRE MVMO^S [35]

La estructura general de MVMO^S es mostrada en la **Figura 4-8**. Comparado con el clásico MVMO, la variante enjambre explora el espacio de solución más agresivamente. El proceso de búsqueda es iniciado con un conjunto de n_p partículas, cada una teniendo su propia memoria definida en términos del archivo correspondiente y función de mapeo. Inicialmente, cada partícula ejecuta m pasos independientes para recolectar un conjunto robusto de soluciones individuales. Entonces, las partículas inician para comunicar e intercambiar información.

Sin embargo, no vale la pena seguir partículas que están muy cerca una de la otra, ya que esto implicaría redundancia. Por lo tanto, en la implementación de inteligencia de enjambre, la distancia normalizada de cada mejor solución local de la partícula $\mathbf{x}^{lbest,i}$ a la mejor solución global $\mathbf{x}^{gbest,i}$ es calculada por:

$$D_i = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(x_k^{gbest} - x_k^{lbest,i}\right)^2} \qquad 4\text{-}13$$

$$D_i < D_{min}$$

Donde *n* expresa el número de variables de optimización. La partícula *i-th* es descartada del proceso de optimización si la distancia D_i es inferior a un umbral definido por el usuario D_{min}. Un umbral cero significa que todas las partículas son consideradas a lo largo de todo el proceso mientras el umbral de la unidad resultará en la caída de todas las partículas excepto el mejor global. En este caso después de las evaluaciones de aptitud (m* $NP + NP$) solo una partícula, la gbest, permanece. Valores de umbrales intermedios implican una mejor adaptación a cualquier problema de optimización.

Es decir, la distancia D_i puede variar en el rango [0,1], $D_i = 0$ implica que las partículas nunca morirán, por lo que ellas permanecen con vida hasta el final de repetitivas evaluaciones de la función (proceso iterativo). Por otra parte, $D_i = 1$ morirán todas las partículas excepto el mejor global [38].

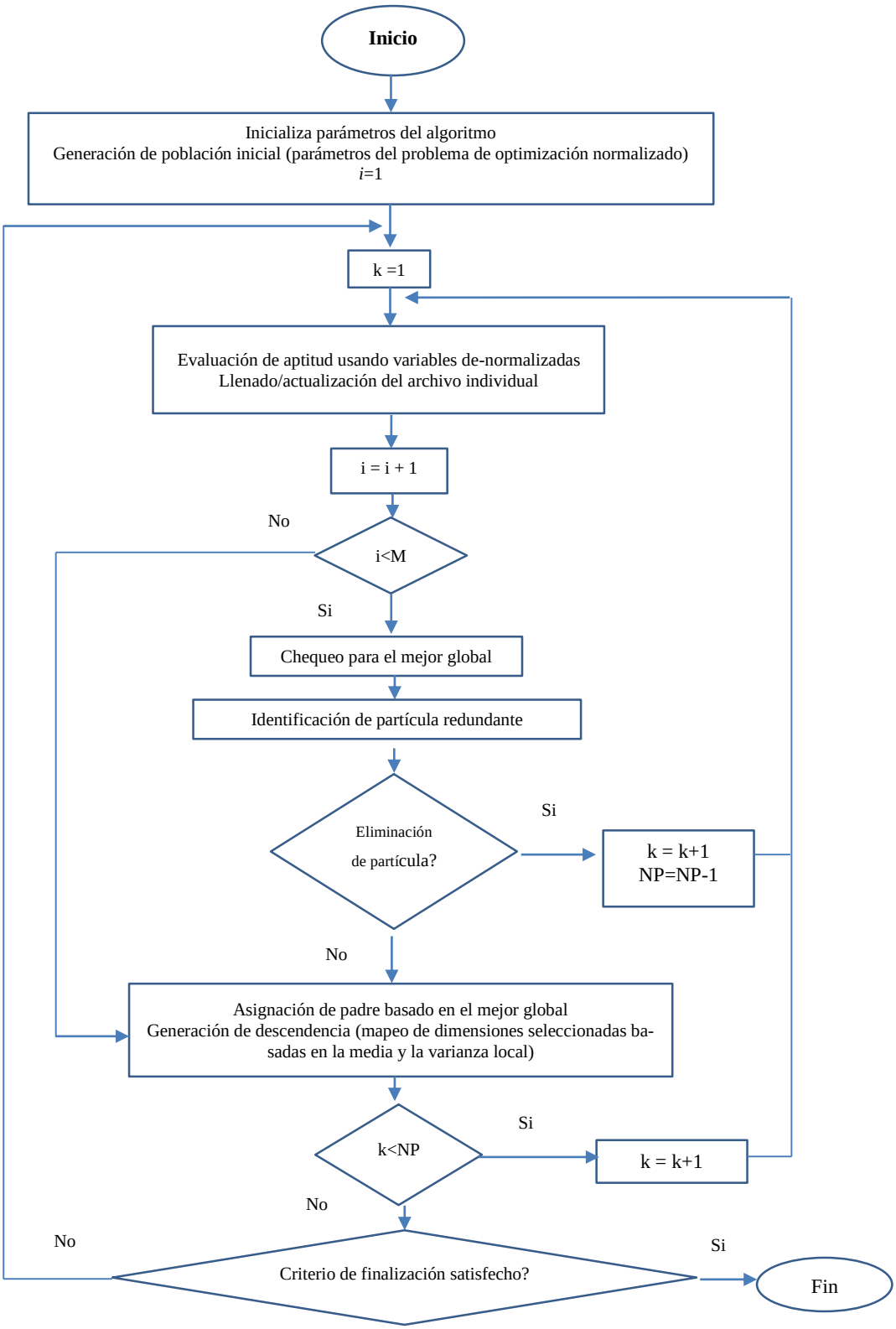


Figura 4-8: Flujograma de MVMO^S (La evaluación de la función y contadores de partículas son indicados por i y k, mientras M y NP representan el número máximo de ejecuciones independientes y el número total de partículas, respectivamente [35].

Después de una evaluación independiente, y si además la partícula es considerada, la búsqueda será dirigida hacia la solución gbest asignando $\mathbf{x}^{gbest}$ en lugar de $\mathbf{x}^{lbest,i}$, como padre para el descendiente de la partícula. Los pasos restantes son idénticos con el clásico MVMO (simple partícula): Un subconjunto de dimensiones en el vector padre son directamente heredados mientras las dimensiones restantes son seleccionadas y mutadas basadas en la evaluación estadística (promedio y varianza) de la partícula usando su propia función de mapeo. Esta etapa es mostrada en la **Figura 4-9**.

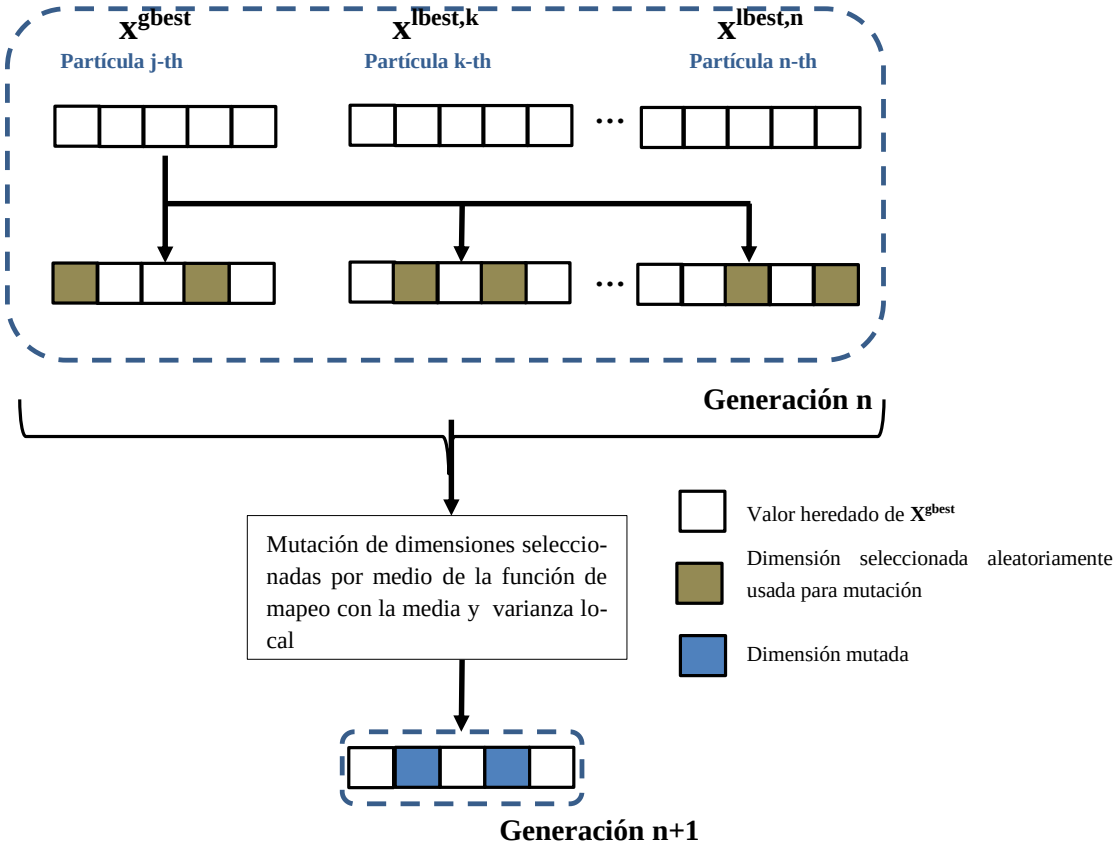


Figura 4-9: Creación de descendencia en MVMO enjambre [38]

Es importante señalar que, a diferencia de otras técnicas de optimización basadas en teoría de inteligencia (enjambre) tales como PSO, MVMO^S no requiere estrictamente varias partículas para proceder. No obstante, el uso de más de una partícula en MVMO^S, la capacidad de búsqueda global puede ampliarse considerablemente. Para problemas de optimización menos desafiantes, el enfoque de una sola partícula puede ser suficiente. Similar al MVMO clásico, el número de iteraciones en MVMO^S es equivalente al número de evaluaciones de

aptitud de descendientes lo que en aplicaciones prácticas usualmente consumen más tiempo que el algoritmo de optimización.

$MVMO^S$ requiere solo tres parámetros adicionales:

1) NP: número de partículas iniciales (si es escogido uno, $MVMO^S$ se ejecuta como el clásico MVMO)
2) M: número de ejecuciones independientes de las partículas
3) D_{min}: umbral de distancia mínimo para la solución del mejor global.

CAPÍTULO 5

5 RECONFIGURACIÓN DE SISTEMAS DE DISTRIBUCIÓN DE ENERGÍA A TRAVÉS DE MVMO CLÁSICO Y MVMO ENJAMBRE

En este capítulo se desarrolla la metodología para resolver el problema de reconfiguración de redes para minimizar pérdidas de potencia activa, aplicando MVMO clásico y MVMO^{S}. Se simula el Caso 2[20], vía MVMO y MVMO^{S}, y sus resultados son comparados con otros métodos de optimización reportados en publicaciones de revistas internacionales indexadas, con el fin de contrastar, validar los resultados obtenidos, y verificar el desempeño de los algoritmos propuestos.

El objetivo para el problema antes indicado, es determinar la configuración "***óptima***" del estado de los seccionadores conduciendo al mínimo de pérdidas de potencia activa mientras se mantienen las restricciones de voltaje en las barras, sistema radial y ecuaciones de flujo de potencia.

La solución exacta para el problema de reconfiguración envuelve una selección, de entre todas las configuraciones posibles, de aquella que tenga menor pérdida. La búsqueda, que examina todas las configuraciones posibles, encontrará la solución exacta del problema. Pero eso es imposible, pues el número de configuraciones factibles generales por el seccionamiento crece exponencialmente con el número y la disposición de seccionadores en el sistema, volviendo al proceso de búsqueda molestoso para sistemas reales. El hecho del crecimiento exponencial del número de posibilidades es denominado explosión de combinaciones [39] .

Por la causa de la razón descrita, se ha propuesto para resolver este problema, el algoritmo meta heurístico MVMO y su variante MVMO^{S}, el mismo que presenta ciertas similitudes con otros algoritmos meta-heurísticos [40], que en si no garantiza el óptimo de la solución

[20] Para este capítulo se usan los datos de impedancia y, potencias activa y reactiva, encontrados en [1].

encontrada, pero sí que la solución sea buena o casi óptima. Este algoritmo está encaminado a establecer criterios que puedan ser usados para eliminar opciones indeseables de seccionamiento en el sentido de disminuir la dimensión del problema.

La metodología propuesta, está dada por un algoritmo estratégico de mutación, el cual se basa en la media y la varianza de la población mejor situado o evaluada. Este modelo converge a un "*óptimo*" cuando la varianza de las variables de control tienda a cero [40].

5.1 PROCEDIMIENTO DE APLICACIÓN MVMO Y MVMO^S, RECONFIGURACIÓN PARA MINIMIZAR PÉRDIDAS DE POTENCIA ACTIVA (RMPPA)

A continuación se presenta la formulación matemática del problema a resolver, el mismo que fue explicado con más detalle en el numeral 3.2. Además, se desarrolla el procedimiento de aplicación de MVMO y MVMO^S.

El problema general es formulado matemáticamente como:

$$\boldsymbol{P_{pérdidas} = f(X, u, d)}$$

Minimizar[21]

$$P_{pérdidas} = Re\left(V_{ss}\sum\left[\left(V_{ss} - V_j\right)y_{ss,j}\right]^*\right) - \sum P_D$$

Sujeto a[22]***:***

a) Configuración radial del sistema
b) Flujos de potencia
c) Magnitud de voltaje en la barra i, $0.95 \trianglelefteq |V| \trianglelefteq 1.05$ i=1,2,…, n_{PQ}

[21] En el numeral 3.2.1, fue explicado con detalle el significado de las variables.
[22] Cada restricción fue explicada con detalle en el numeral 3.2.2.

Donde $\boldsymbol{P}_{pérdidas}$ son las pérdidas de potencia activa de la red de distribución. El vector variables de control es $\boldsymbol{X} = [x_1^k , x_2^k , x_3^k , \ldots x_p^k]^T$, el mismo que contiene los segmentos de líneas (ramales de interconexión) en estado abierto para cada lazo p del SDR. El vector de variables de estado (dependiente) es $\mathbf{u} = [V]^T$, sus límites están dados en c), y n_{PQ}, corresponde al número de barras de carga (PQ). El vector $\mathbf{d}= [P_D \; Q_D]^T$, contienen la potencia activa y reactiva demandada en las barras de carga.

En cada actualización del vector de control $\boldsymbol{X}$, el vector de estado **u**; la magnitud de voltaje de las barra de carga y ángulo son calculados por las ecuaciones de flujo de potencia (definidas en 3-5 y 3-6). Es decir, que una vez determinados los valores de las variables de control, para cada partícula o cada una de las partículas (***posibles soluciones***), se ejecuta una rutina de flujos de potencia que permite determinar los flujos de potencia activa y reactiva a través de las ramas, los voltajes y ángulos de fase en cada barra del SDR. Además, se obtienen las pérdidas totales de potencia activa en el SDR.

En el espacio de búsqueda de un problema de optimización con restricciones hay dos clases de partículas: factibles y no factibles. Las soluciones factibles satisfacen todas las restricciones, mientras que las no factibles no cumplen con al menos una de ellas [33].

La forma común para tratar los problemas de optimización con restricciones es usar funciones de penalización. El problema con restricciones se transforma en uno sin restricciones a través de la penalización de la función objetivo. Cuando se usa una función de penalización, la magnitud de la desviación de la restricción se usa para "castigar" o "penalizar" una solución que no es factible, de manera que las soluciones factibles se vean favorecidas [33]. Las funciones de penalización pueden clasificarse en dos categorías: estacionarias y no estacionarias, las primeras son valores de penalización fija a lo largo del algoritmo de optimización mientras que en las segundas, los valores de penalización son dinámicamente modificados.

La función constituida por la función objetivo más las funciones de penalización se denominan función de aptitud [33]. Para definir la función aptitud para este trabajo de tesis, se tomaron como referencias [23] y [41] , y se formula de la siguiente manera:

$$J(X) = \begin{cases} K = 1e100, configuración\ no\ factible \\ P_{pérdidas} + k.vd \end{cases} \quad 5\text{-}1$$

Donde, *K* es un número grande asignado a la función aptitud si el conjunto de configuraciones de seccionadores no es factible para el SDR, es decir, no es una configuración radial. Mientras, $P_{pérdidas}$ son las pérdidas de potencia en el sistema de distribución, a la cual, se le adiciona un nuevo término de penalización $k.vd$, que sirve para mejorar los perfiles de voltaje; *k* es un valor entre 5 y 10, y *vd* está definido de la siguiente manera:

Primera variante, se incorpora una restricción de tal manera de fijar el voltaje en 1.0 p.u:

$$Vdev = |V - 1| \quad 5\text{-}2$$

Si $Vdev \trianglelefteq 0.07 \rightarrow Vdev = 0$ 5-3

y, $vd = \sum Vdev$ 5-4

Segunda variante, se considera la banda de voltaje permitido para el límite inferior (-5%):

$$Vdev = |V - 0.95| \quad 5\text{-}5$$

Si $Vdev <$ 1exp-7 $\rightarrow Vdev = 0$ 5-6

y, $vd = \sum Vdev$ 5-7

El valor *V* es la magnitud de voltaje en la barra *i*; y vd es la sumatoria de los voltajes $Vdev$, luego de que se han calculado sus valores para todas las barras del SDR con 5-2 y 5-5 y verificado sus restricciones con 5-6 y 5-3, de esta manera se asegura que *V* no sobrepase los límites definidos y se cumplan las restricciones para el voltaje.

Como se puede verificar en la ecuación 5-1, el objetivo consiste de minimizar las pérdidas de potencia activa, mientras se cumplen las restricciones de radialidad del sistema y límites de voltaje. La inclusión de *vd* ayuda asegurar que los voltajes de barra sean mantenidos dentro de un rango aceptable. El factor *k* es un peso sobre la contribución de *vd* al valor objetivo.

La solución de voltaje debe ser obtenida para una solución particular de reconfiguración del SDR antes que las pérdidas de potencia asociadas con la solución puedan ser calculadas. El método de flujo de carga usado para calcular las soluciones de voltaje fue descrito en el Capítulo 2.

5.1.1 APLICACIÓN MVMO Y MVMO^S

En MVMO una solución candidata del problema de optimización se caracteriza por un vector en el espacio real de dimensión *D*. El proceso de búsqueda del modelo MVMO comienza desde un punto inicial ***X*** **(vector de configuración inicial)** generado aleatoriamente o especificado por el usuario. La escala interna de todas las variables está restringida entre [0,1]. Eso significa que los límites reales tienen que ser normalizados entre [0,1]. El vector respuesta o de aptitud es des-normalizado a los rangos reales sólo cuando la función objetivo o de aptitud es evaluada [40].

Los vectores respuestas con mejor aptitud son almacenados en un archivo de soluciones, el cual sirve como base del proceso de conocimiento y aprendizaje del algoritmo. Cabe señalar que MVMO es un algoritmo de búsqueda de agente único, ya que únicamente genera o muta un descendiente en cada iteración. Por lo tanto, el número de evaluaciones físicas es idéntico al número de iteraciones [40].

Mientras el algoritmo MVMO, tiene búsqueda de agente único, MVMO^S, está enfocado a varios agentes o conjunto de partículas; cada partícula tiene su propia memoria representada por su correspondiente archivo de soluciones y funciones de mapeo que tienen su propia memoria y su propio archivo de soluciones. Al inicio cada partícula ejecuta **M** pasos independientes para recolectar un conjunto de soluciones individuales, luego éstas se comunican e intercambian información [37].

De esta manera, el procedimiento de MVMO para resolver el problema de RMPPA, puede ser resumido como:

1. Lee los datos del SDR y establece los parámetros iniciales de MVMO, los cuales se indican en el ANEXO 4.
2. **Inicialización:** Inicializa el vector ***X*** entre [0,1].
3. **Evaluación de la función objetivo o de aptitud *J(X)***: Desenormaliza ***X,*** verifica restricción de radialidad, corre el flujo de potencia, y verifica restricciones de voltajes.

   El vector solución, ***X***, es un vector que contiene el número del segmento de línea para cada lazo del SDR que debe ser abierto. En este camino, la estructura radial del SDR es garantizado para cada solución.
4. **Finalización**: chequea el criterio de finalización del proceso. Si cumple el número de iteraciones, termina MVMO, caso contrario continúa al paso 5.
5. **Archivo de soluciones**: Almacena los mejores vectores solución (soluciones con mejor aptitud).
6. Basado en el archivo de soluciones, computa la media $\bar{x}_i$ y la varianza v_i para cada solución i almacenada.
7. **Asignación del mejor candidato**: Asigna la mejor solución archivada $\mathbf{x_{best}}$ como el padre.
8. **Selección de variable**: selecciona una dimensión m < D del vector solución ***X***.
9. **Mutación**: Aplica la función de mapeo a las mejores soluciones almacenadas en el vector de dimensión m.
10. **Cruce** (interacción, acople): Fija la dimensión restante de ***X*** a los valores de $\mathbf{x_{best}}$.
11. Ir al paso 3.

La metodología MVMO basada en programación evolucionaria, para resolver el problema de RMPPA trabaja inicialmente generando soluciones aleatorias y evaluándolas a ellas. Una vez las soluciones han sido evaluadas, un cierto número de ellas son escogidas como "mejores soluciones". Este archivo de "mejores soluciones" entonces es usado para generar más soluciones, a través de la función de mapeo. Esas nuevas soluciones, junto con las "mejores soluciones" son evaluadas otra vez. Un nuevo archivo de "mejores soluciones" es entonces

escogido, a partir de la cual nuevas soluciones son otra vez generadas. Este ciclo se repte a sí mismo para un específico número de veces. Es esperado que luego de algunas iteraciones, la solución del óptimo global ha sido encontrada (aunque esto no es garantizado). Este tipo de método siempre dará una solución la cual no depende del estado inicial del sistema a ser resuelto.

Es decir, para una cierta solución kth, X^k=[x_1^k , x_2^k , x_3^k ,. . . x_p^k]. Para un SDR con *p* lazos, x_1^k - x_p^k son las líneas que deben ser abiertas en cada lazo. La solución para X^{k+n} = [x_1^{k+n}, x_2^{k+n}, x_3^{k+n} ,. . . x_p^{k+n}], debería evolucionar de la solución X^k .

Por ejemplo para la etapa de mutación considerando la estrategia 1, para crear una solución candidata nueva (descendencia), MVMO elige la mejor solución (ranking en el archivo de solución), luego de esa solución. Suponiendo que se tiene D variables de optimización, se eligen *m* dimensiones aleatoriamente, y se les aplica la función de mapeo al resto, es decir D-m variables/dimensiones quedan tal cual con los valores del padre, por lo que una solución nueva tiene rasgos de la mejor solución, y otros nuevos son elegidos aleatoriamente y mutados con la función de mapeo.

Para el problema propuesto, D variables de optimización son los seccionadores a ser abiertos o cerrados, para cada lazo, por ejemplo si se tienen 10 seccionadores que son equivalentes a tener 10 variables de optimización o dimensiones del problema, cada variable puede tomar un valor cero (abierto) o 1(cerrado), una solución es un vector de 10 variables y una solución nueva (descendencia) tiene también 10 variables, solo que de las 10 se podrían seleccionar 4 aleatoriamente para obtener un valor nuevo (variables discretas de 0 y 1), el resto quedan con los valores (cero o 1) de la mejor solución.

Para MVMO^S, la metodología es parecida a la aplicada en MVMO, con la excepción que ya no inicializa una sola partícula como MVMO, sino un número NP de partículas. A continuación se describen los pasos a seguir:

1. Lee los datos del SDR y establece los parámetros iniciales de MVMOS, los cuales se indican en el ANEXO 4.

2. **Inicialización:** Genera la población inicial compuesta por los vectores $\boldsymbol{X_1, X_2,\ldots.X_{NP}}$ *(soluciones posibles, denominadas números de partículas)* entre [0,1].
3. **Evaluación de la función objetivo o de aptitud**: Desenormaliza los vectores $\boldsymbol{X_1, X_2,\ldots.X_{NP},}$ verifica restricción de radialidad, corre el flujo de potencia, y verifica restricción de voltajes, para cada uno de los vectores o partículas.

   Los vectores solución $\boldsymbol{X}$ para cada partícula son vectores que contienen el número del segmento de línea para cada lazo del SDR que debe ser abierto. En este camino, la estructura radial del SDR es garantizado para cada solución.
4. **Archivo de soluciones**: Almacena los mejores vectores solución (soluciones con mejor aptitud), para cada partícula. Cabe indicar que cada partícula tiene su propio archivo de memoria.
5. Chequea las partículas que tienen el "mejor global"
6. Identifica si existen partículas redundantes o soluciones que se repiten varias veces.
7. Procede a la eliminación de partículas (muerte de partículas) redundantes en el caso de que las existiera.
8. **Asignación del mejor candidato**: Asigna la mejor solución archivada $\mathbf{x_{gbest}}$ como el padre, para cada partícula. Basándose en los archivos de soluciones, computa la media $\bar{x}_i$ y la varianza v_i para cada solución i almacenada, de cada partícula.

   Aplica la función de mapeo a las mejores soluciones almacenadas en el vector de dimensión m, para cada partícula.
9. **Cruce** (interacción, acople): Fija la dimensión restante de $\boldsymbol{X}$ para cada partícula a los valores de $\mathbf{x_{best}}$.
10. **Finalización**: chequea el criterio de finalización del proceso. Si cumple el número de iteraciones, termina MVMO^S, caso contrario continúa al paso 3.

5.2 RESULTADOS DE MVMO Y MVMO^S, SDR 33 BARRAS

Los métodos de solución propuestos basados en MVMO y MVMO^S, son probados en el SDR de 33 barras, definido en [1]. Para el SDR -33 barras, el voltaje de la subestación es considerado como 1 p.u., y los ramales de interconexión y ramales de seccionamiento de la **Tabla 5-1** , son considerados como seccionadores candidatos para el problema de reconfiguración. El algoritmo fue desarrollado en MATLAB, y las simulaciones fueron realizadas

en un computador personal con procesador Intel(R) Core(TM) i5, 2.53 GHz, RAM (2,93 GB), sistema operativo 32 bits.

5.2.1 CASO DE PRUEBA: SISTEMA DE 33 BARRAS

El SDR – 33 barras, 12.66 kV, se muestra en la **Figura 5-1**, que consiste de 5 ramales de interconexión 33, 34, 35, 36 y 37 y ramales de seccionamiento numerados del 1 al 32. Los datos de carga e impedancias de las líneas del SDR son obtenidos de [1], así como las potencias activa y reactiva de las cargas corresponden a un total 3,715 kW y 2,300 kVAr, respectivamente. Las pérdidas de potencia activa iniciales de este sistema son de 202.67 kW. El valor de voltaje más bajo es de 0,9131 p.u., en la barra 18. Los resultados indicados son el resultado de la simulación de flujo de carga para el SDR – 33 barras.

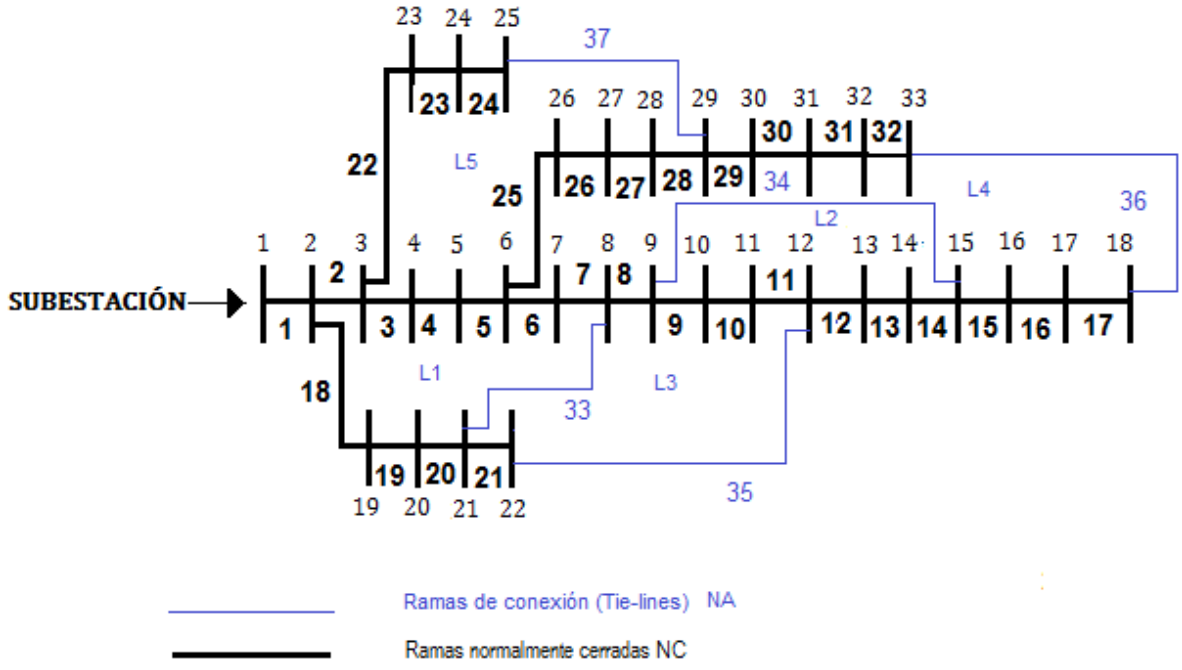


Figura 5-1: Configuración inicial SDR 33 barras

Número de lazos	Índices de líneas que pueden ser maniobradas								
L1	33	18	19	20	06	07	03	04	05
L2	34	12	13	14					
L3	35	08	11	10	09	21			
L4	36	32	17	16	15	29	30	31	
L5	37	24	23	22	28	25	26	27	

Tabla 5-1: Índices de los segmentos de línea que pueden maniobrados para cada lazo en el SDR de 33 barras

	Lazo 1								Lazo 2				Lazo 3						Lazo 4								Lazo 5							
s	0	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

Tabla 5-2: Vector *sws* para SDR 33 barras

5.2.1.1 Resultados con MVMO

Los parámetros iniciales de MVMO usados en varias simulaciones del SDR-33 barras son mostrados en la **Tabla 5-3** y la configuración óptima para el algoritmo propuesto es [7 14 9 32 37], con pérdidas de potencia activa de 139,55121 kW, lo que significa una reducción del 31.14 % en pérdidas de potencia activa. El voltaje mínimo de barra del sistema es mejorado a 0.9378 p.u. (barra 32) después de la reconfiguración.

MVMO	**Simulación 1**	**Simulación 2**	**Simulación 3**	**Simulación 4**
PÁRAMETROS INICIALES				
Población	*5*	*5*	*5*	*5*
Estrategia	*4*	*4*	*4*	*4*
$f^*_{s\_ini}$	*0.9*	*1*	*1*	*0.9*
$f^*_{s\ final}$	*2*	*3*	*3*	*1.5*
Δd_{0ini}	*0.05*	*0.4*	*0.35*	*0.05*
Δd_{0final}	*0.35*	*0.4*	*0.35*	*0.35*
d_i	*1*	*2*	*2*	*1*
Corridas (repeticiones)	***200***	***200***	***200***	***200***
Número de evaluaciones por cada corrida.	***250***	***250***	***250***	***250***
RESULTADOS				
Repetición con mejor resultado	***46***	***85***	***104***	***38***
Iteración con la que alcanzar el mejor resultado	***3***	***107***	***208***	***14***
X_f	*[7 14 9 32 37]*	*[7 14 9 32 37]*	*[7 14 9 32 28]*	*[7 14 9 32 28]*
$P_{pérdidas}$ ***(kW)***	*139.55121*	*139.55122*	*139.9781*	*139.9781*
V_{min} ***(p.u)***	*0.9378*	*0.9316*	*0.9382*	*0.9369*
Media	*146.3140*	*145.7837*	*146.4575*	*146.4350*
Std	*3.4341*	*3.4663*	*3.1223*	*3.3205*
CPU Time (s)	*9.79*	*10.07*	*10.28*	*10.43*

Tabla 5-3: Resumen resultados MVMO con varios parámetros iniciales

De los resultados obtenidos y mostrados en la **Tabla 5-3**, se verifica que la simulación 1, que considera una población de 5, estrategia de mutación 4, factores de escalamiento inicial y final, 0.9 y 2 respectivamente, con un incremento del factor de forma Δd_0 inicial y final de 0.05 y 0.35, factor inicial de forma $\boldsymbol{d_i}$ igual 1; MVMO presenta mejor desempeño, dado que de 200 repeticiones de optimización realizadas considerando 250 evaluaciones o iteraciones para cada repetición, se obtiene un valor mínimo de pérdida de potencia en la repetición número 46 desde su tercera evaluación, en un tiempo de 9.79 segundos. El esquema unifilar de la configuración radial final se muestra en **Figura 5-2**

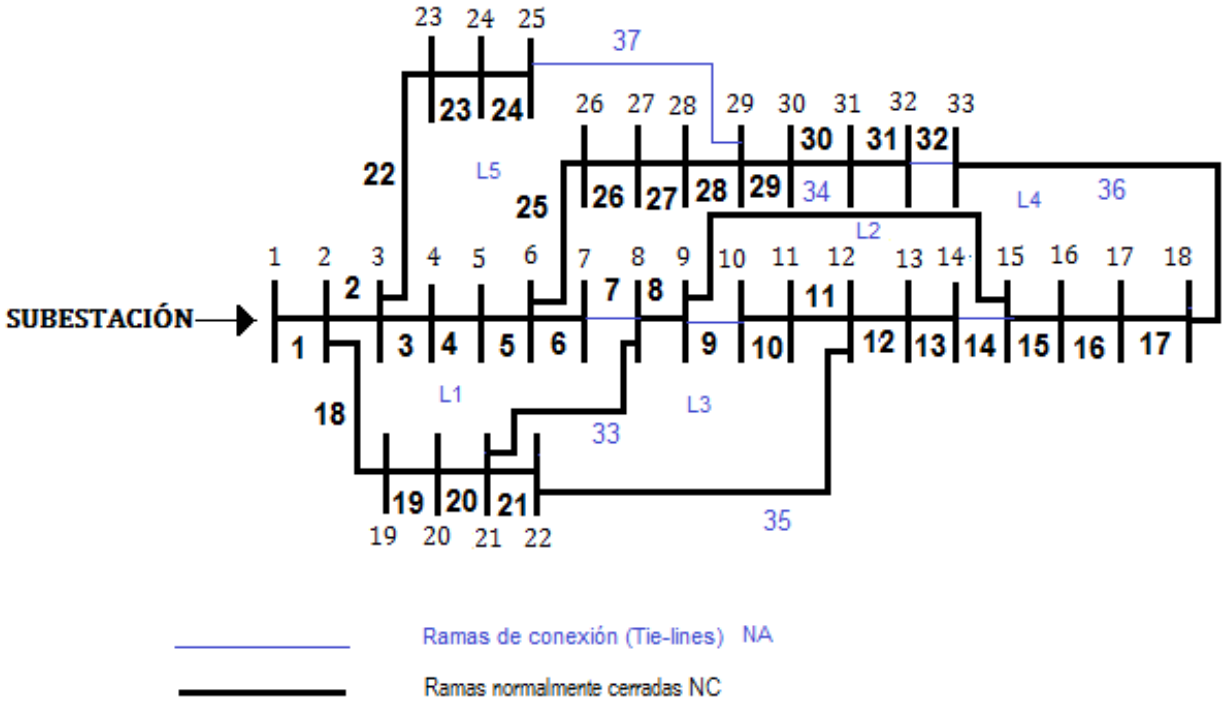


Figura 5-2: Configuración final SDR 33 barras

El perfil de voltaje de barra antes y después de la reconfiguración se muestra en la **Figura 5-3**. El voltaje mínimo en el sistema después de la reconfiguración es mejorado en 2,63 %.

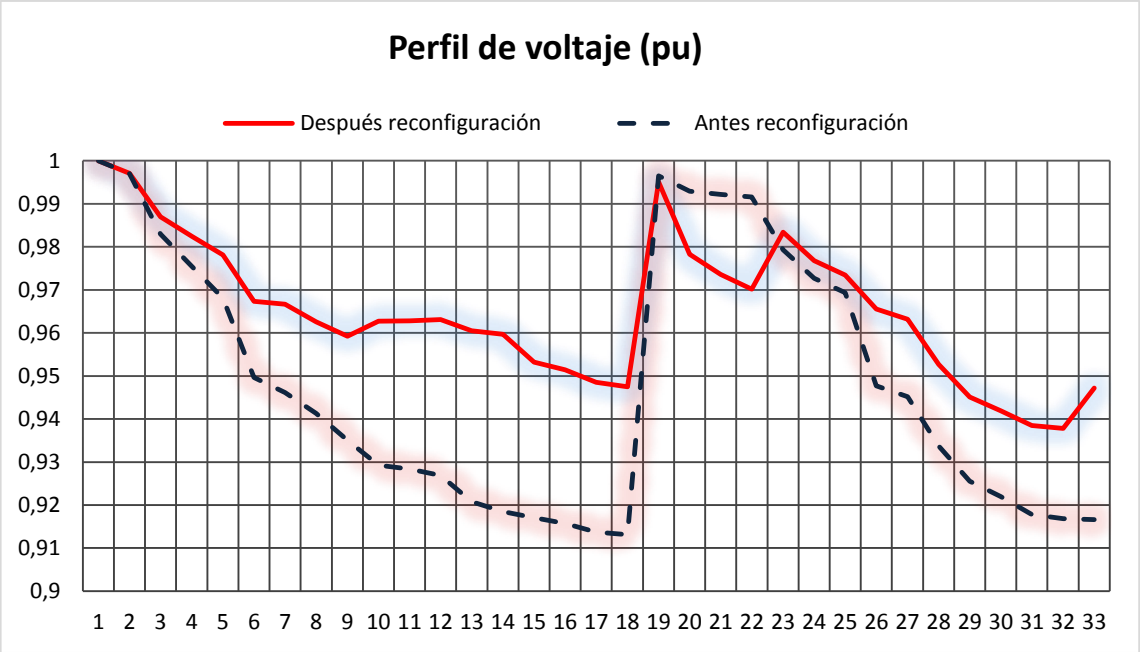


Figura 5-3: Perfil de voltaje antes y después de la reconfiguración con MVMO SDR 33 barras (Voltaje vs Barra del sistema)

La característica de convergencia de MVMO para la simulación 1, se muestra en la **Figura 5-4**, y el reporte de resultados con los valores de evaluación de la función aptitud ($P_{pérdidas}$) para las 250 evaluaciones impresas cada 25 pasos para la repetición No. 46, se presenta en el **ANEXO 5**. MVMO obtiene un valor mínimo de pérdidas de 139,55 kW, desde la iteración 3.

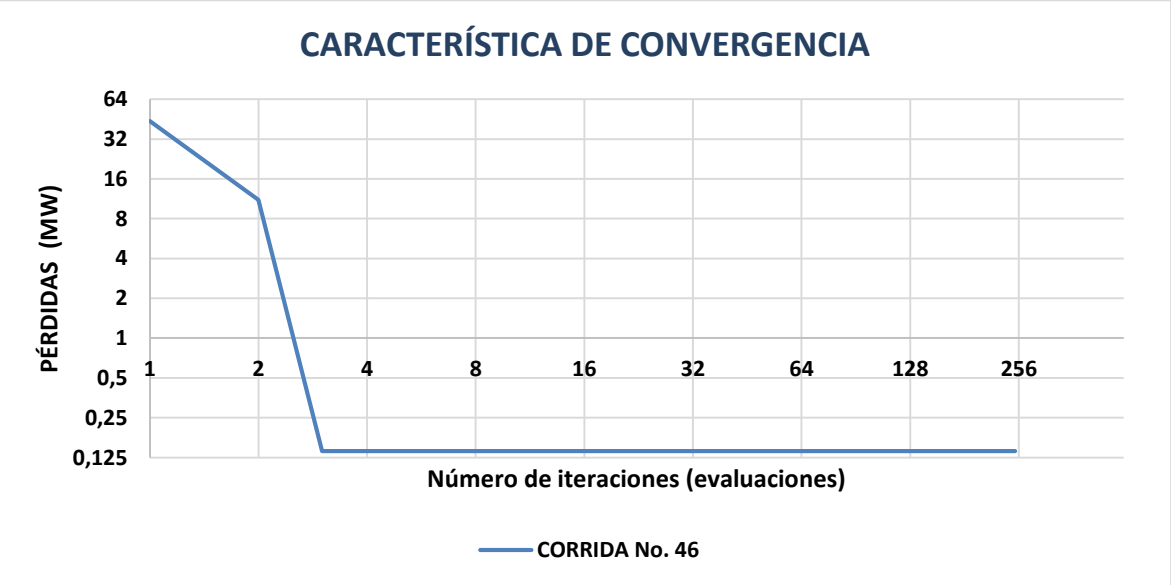


Figura 5-4: Característica de convergencia para MVMO, SDR 33 barras

5.2.1.2 Resultados con MVMO^S

Los parámetros iniciales de MVMO^S usados en las simulaciones del SDR-33 barras, son mostrados en la **Tabla 5-4** . MVMO^S tiene 3 parámetros iniciales adicionales que se describen a continuación: número de partículas **NP**, distancia entre partículas $\boldsymbol{d_{max}}$ y por último el número de simulaciones independiente para cada partícula **M**.

La configuración óptima para MVMO^S es [7 14 9 32 37] y pérdida de potencia activa de 139,55 kW, correspondiente a la simulación 1, por lo que, el valor de pérdida y ramales que deben ser abiertos para cada lazo, son iguales a los obtenidos para MVMO, pero con la diferencia que al tener 2 partículas trabajando independientemente ejecutando 10 simulaciones independientes cada una de ella, para las 200 corridas y 250 evaluaciones de la función objetiva, para encontrar la mejor solución, el tiempo de procesamiento es levemente mayor (de 9.79 segundos se incrementa a 10.03 segundos) para MVMO^S, es decir, se necesitaron 0.24 segundos adicionales, a pesar de la cantidad de cálculos adicionales que realiza esta variante de MVMO clásico.

Igual que MVMO, su variante MVMO^S, fue corrido para 200 repeticiones y para cada una de ellas se realizaron 250 evaluaciones. En la simulación 1, MVMO^S encontró la mejor solución en la corrida 164, desde su evaluación número 95, la partícula que presentó mejor aptitud fue la número 1 de las 2 partículas, y la distancia entre partículas fue 0. En el ANEXO 5, se muestra la tabla de resultados de la simulación 1.

Es importante resaltar que al aumentar el número de partículas y distancia entre ellas, no se obtienen resultados mejores que los obtenidos, como se muestra en las simulaciones 2, 3, 4, y 5, lo que indica que con estos valores MVMO^S está eliminando partículas más rápidamente, perjudicando la diversidad de la búsqueda prematuramente, por lo que se reduce la posibilidad de obtener mejores resultados.

MVMO^S	Simulación 1	Simulación 2	Simulación 3[23]	Simulación 4	Simulación 5
PÁRAMETROS INICIALES					
Población	*5*	*5*	*5*	*5*	*5*
Estrategia	*4*	*4*	*4*	*4*	*4*
NP	*2*	*2*	*10*	*5*	*3*
d_{max}	*0*	*0.5*	*0.5*	*1*	*0.8*
M	*10*	*10*	*10*	*10*	*1*
$f^*_{S\_ini}$	*1*	*1*	*1*	*1*	*1*
$f^*_{S\ final}$	*2*	*2*	*2*	*2*	*3*
Δd0ini	*0.35*	*0.35*	*0.35*	*0.35*	*0.35*
Δd0final	*0.35*	*0.35*	*0.35*	*0.35*	*0.35*
d_i	*1*	*1*	*1*	*1*	*1*
Corridas (repeticiones)	***200***	***200***	***200***	***200***	***200***
Número de evaluaciones por cada corrida.	***250***	***250***	***250***	***250***	***250***
RESULTADOS					
Repetición con mejor resultado	*164*	*47*	*1*	*63*	*62*
Iteración con la que alcanzar el mejor resultado	*95*	*29*	*238*	*189*	*225*
Partícula con mejor aptitud	*1*	*1*	*3*	*1*	*3*
X_f	*[7 14 09 32 37]*	*[7 14 09 32 37]*	*[7 14 09 32 37]*	*[7 14 09 32 37]*	*[7 14 09 32 28]*
$P_{pérdidas}$ ***(kW)***	*139.551*	*139.551*	*139.551*	*139.551*	*139.978*
V_{min} ***(p.u)***	*0.9387*	*0.9356*	*0.9335*	*0.9319*	*0.9372*
Media	*145.8262*	*146.1978*	*0*	*145.7026*	*146.3546*
Std	*3.1888*	*3.4745*	*0*	*3.4544*	*3.0*
CPU Time (s)	*10.03*	*10.10*	*9,89*	*10.14*	*9.73*

Tabla 5-4: Resumen resultados MVMO^S

En relación al voltaje mínimo obtenido con MVMO^S, el resultado fue ligeramente mejor que MVMO clásico, de 0.9378 se incrementó a 0.9387.

[23] No completó las 200 corridas, en la corrida 16 finalizó la simulación, por lo que estos resultados no se consideran en el análisis de mejores resultados obtenidos.

5.2.1.3 Comparación con otras metodologías

Para el propósito de evaluación de los resultados obtenidos con MVMO y MVMO^S, son comparados con los alcanzados en [11] y [42].

En [11], se resuelve el problema de reconfiguración de redes para el SDR 33 barras aplicando la metodología del algoritmo de optimización denominado Búsqueda de Armonía HSA y con el fin de probar su potencial, sus resultados son comparados con Algoritmos Genéticos GA, Algoritmos Genéticos Refinados RGA, y Búsqueda Tabú Mejorado ITS.

Por otra parte en [23] para resolver el problema de reconfiguración del SDR-33 barras con generación distribuida, se aplica un algoritmo evolucionario EA, y para fines de comparación con MVMO y MVMO^S, se implementan los códigos encontrados en [23], pero redefiniéndolos, con el objetivo que no tengan la característica de generación distribuida.

Los algoritmos GA y RGA, fueron simulados para una magnitud o tamaño de población de 85, mientras que para HSA se consideró un tamaño de memoria de armonía[24] de 10 [11] . Para MVMO y MVMO^S, las simulaciones fueron realizadas para un tamaño del archivo (población) de 5.

En la **Tabla 5-5**, se muestran los resultados para cada metodología aplicada al SDR de 33 barras [1], sistema que fue escogido para verificar el desempeño de MVMO y MVMO^S.

[24] Número de notas óptimas del instrumento *i*

Item		Configuración inicial		Configuración Final						
			HA[25] [42]	GA [11]	RGA [11]	ITS [11]	HSA [11]	EA[26] [23]	Metodología propuesta	
									MVMO	MVMO^S
Corridas				200	200	200	200	200	200	200
Máximo número de iteraciones				250	250	250	250	250	250	250
Ramales de interconexión (abiertos)		33,34,35,36,37	37,9,32	7,9,14,37,32	7,9,14,37,32	7,9,14,37,36	7,10,14,37,36	7,9,14,32,37	7,9,14,32,37	7,9,14,32,37
Pérdidas (kW)	**Mejor**	202.67	140.8	141.6	139.5	139.28	138.06	139.55	139.55	139.55
	Media			166.2	164.9	163.5	152.33	144.74	146.3140	145.82
	Std			14.53	13.34	12.11	11.28	2.50	3.43	3.19
Voltaje Mínimo p.u.			0.9376	0.9290	0.9315	0.9210	0.9342	0.9377	0.9378	0.9387
Tiempo CPU(s)				19.1	13.8	8.1	7.2	167.50	9.79	10.03

Tabla 5-5: Resultados con diferentes métodos para el SDR – 33 barras

Para verificar el desempeño de los algoritmos, en [11] HSA fue corrido considerando 200 repeticiones y para cada repetición se realizaron 250 evaluaciones de la función aptitud (iteraciones o cálculos de flujos de potencia), de igual manera, se realizó para GA, RTS e ITS, y para propósito de comparación de EA, con MVMO y MVMO^S, se evaluaron con los mismos valores de repeticiones y evaluaciones, a excepción de HA propuesta en [42] que es una metodología que utiliza el concepto de Patrón de Flujo Óptimo, en un único lazo de la red, de esta manera en vez de cerrar todos los ramales de interconexión de la red, para formar una red con lazos y abrir los seccionadores uno tras otro, se cierra un solo seccionador a la vez para introducir un lazo en el sistema, para luego abrir un seccionador del lazo, volviendo a ser nuevamente una red radial [7], este proceso es repetido hasta que la configuración de pérdidas mínimas es obtenida.

Las pérdidas de potencia activa conseguidas con MVMO y MVMO^S, presentan un valor de 139.55 kW similar a las alcanzadas con RGA y EA, no obstante, el mejor resultado de pérdidas mínimas, es para HSA con un valor de 138.06 kW, lo que corresponde a un porcentaje de 1.07 % ligeramente superior a las metodologías propuestas.

[25] Algoritmo Heurístico de Goswami-Basu, 1992, pérdidas iniciales de 205.81 kW
[26] Se implementó el EA en la herramienta computacional Matlab, de acuerdo a los códigos de [19].

MVMO^S, presentó un mejor voltaje, seguido de MVMO, EA y HA, con porcentajes de mejora de 2.73%, 2.63%, 2.63% y 2.62%, respectivamente. Los perfiles de voltaje para las configuraciones finales de los algoritmos propuestos para fines de comparación, se muestran en la **Figura 5-5**.

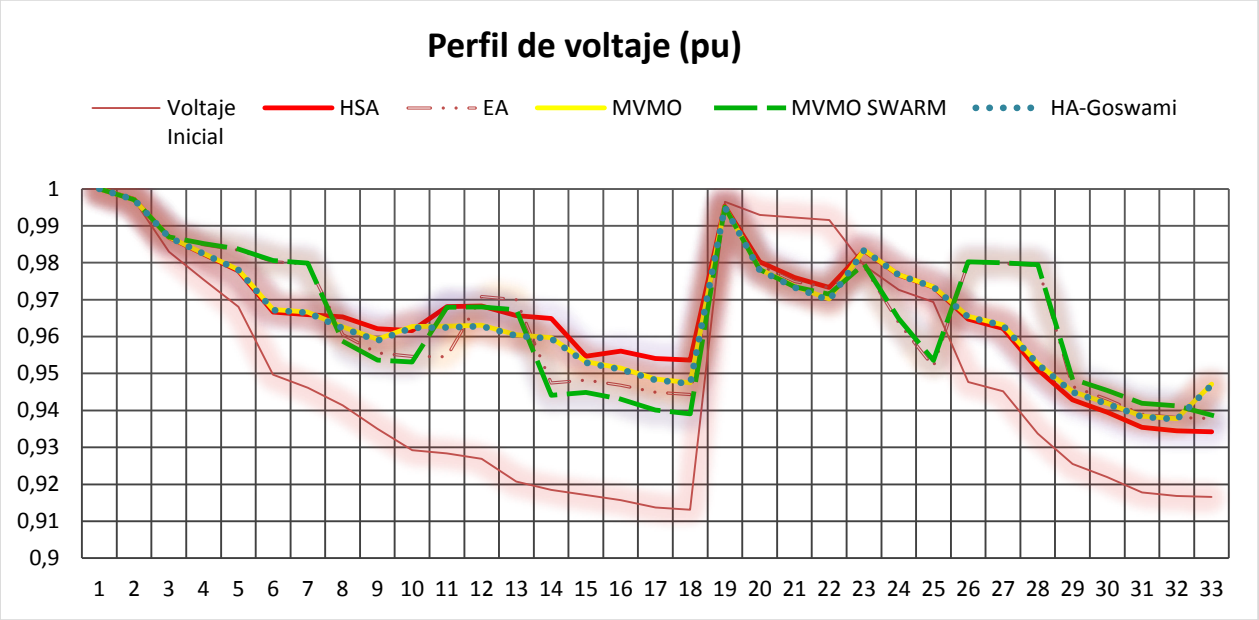


Figura 5-5: V(pu) vs Número Barra del sistema (antes y después de la reconfiguración con varios métodos de optimización)

El menor tiempo de procesamiento fue para HSA que también obtuvo el menor valor de pérdidas activas. Mientras, el menor tiempo de procesamiento entre los algoritmos que obtuvieron pérdidas de alrededor de 135.55 kW, fuera para MVMO (9.79 s), seguido de MVMO^S (10.03 s), RGA (13.8 s) y en último lugar EA (167.50 s), el cual, obtiene menores pérdidas en un tiempo muy superior a los otros algoritmos.

De la evaluación estadística, se observa que GA, RGA, ITS, HSA, obtuvieron convergencia prematura, dado que sus valores de desviación estándar son superiores a EA, MVMO, MVMO^S. Por otra parte, los valores pequeños de Std implican que la mayor parte de las mejores soluciones están muy próximas al promedio, lo cual, se puede verificar en los resultados obtenidos para las metodologías propuestas, donde su valor medio es más cercano (alrededor de 145 kW), al mejor valor de potencia activa encontrado.

Los códigos .m para MVMO, MVMO^S y EA, fueron implementados en Matlab, de esta manera se pudo comparar sus características de convergencia. Dado que este trabajo de tesis fue desarrollado para MVMO, sus códigos son presentados en el ANEXO 6. Además se presentan los códigos para las subrutinas de flujos de potencia disponibles en [23], en el ANEXO 6.

En la **Figura 5-6**, se muestran las características de convergencia para MVMO, MVMO^S y EA. De estos resultados, se puede verificar que MVMO presenta mejor desempeño que MVMO^S y EA, dado que encuentra su valor mínimo de potencia antes que los otros dos métodos, mientras los otros realizan una búsqueda más exhaustiva y toman más tiempo, pero no logran encontrar un mejor valor que el encontrado por MVMO.

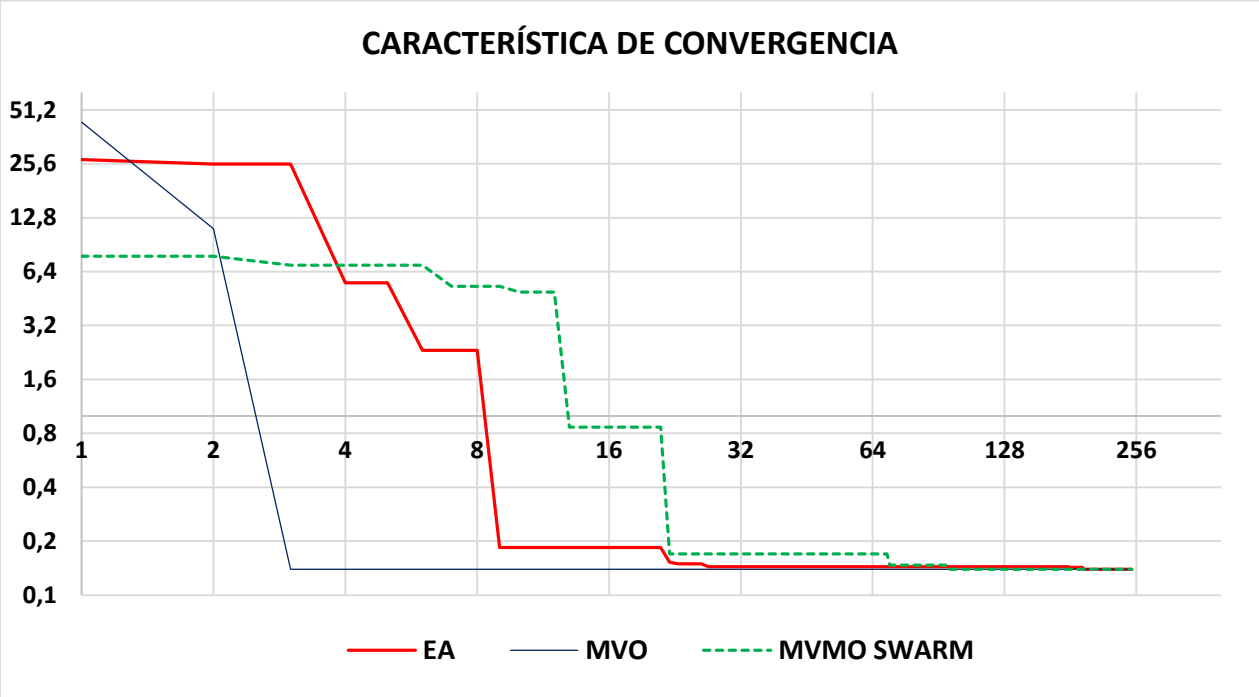


Figura 5-6: Característica de convergencia para EA, MVMO y MVMO^S para el SDR 33 barras. Pérdidas potencia activa (MW) *vs* No. Evaluaciones función aptitud

A continuación, en la **Figura 5-7** se muestra el detalle de la característica de convergencia en una escala donde se puede visualizar mejor los resultados obtenidos para las metodologías indicadas, sobre todo desde la evaluación 95 de las 250 realizadas, para EA y MVMO^S. En la figura se indica que EA encuentra su valor mínimo de potencia desde la evaluación 194 en la corrida (ó repetición) No. 39, mientras MVMO lo hace desde su evaluación 3 en la

repetición No.46 y MVMO^S, desde la evaluación No. 95 en su corrida No. 164, mostrando así la efectividad de MVMO para encontrar el valor mínimo de manera más rápida que los otros métodos.

MVMO^S, obtiene su valor mínimo de pérdidas con una sola partícula, puesto que se simuló con 2 partículas, de las cuales murió la segunda y permaneció viva la que mostró mejor aptitud (partícula 1).

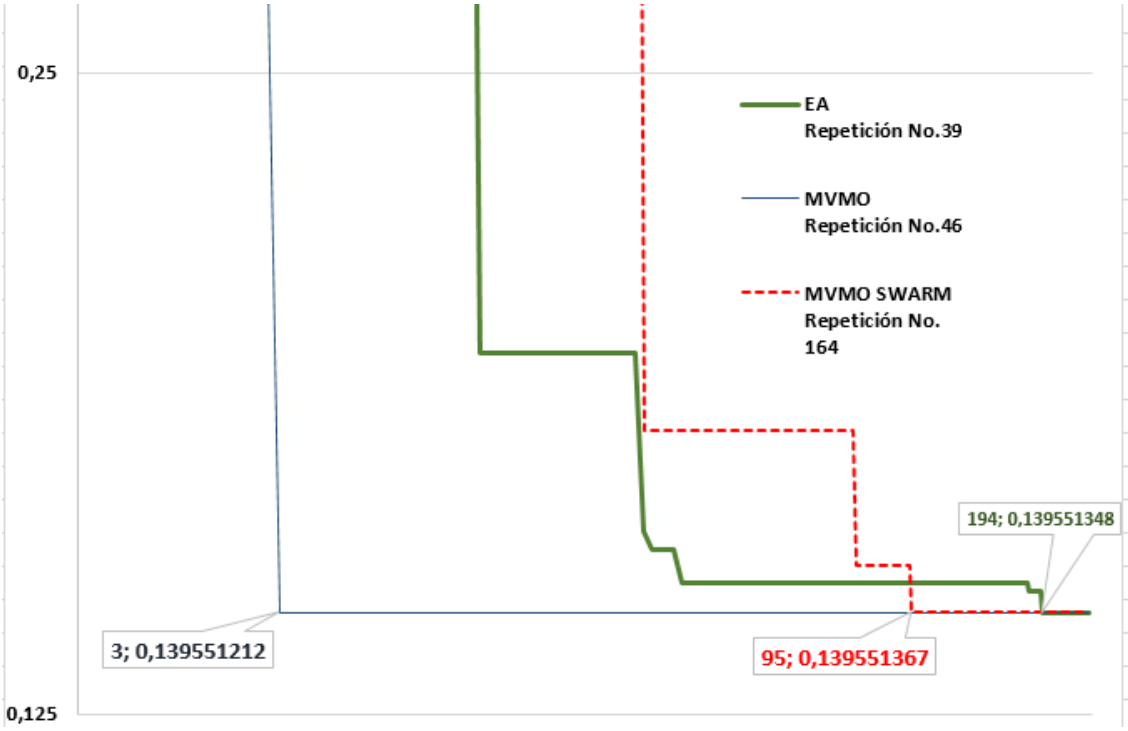


Figura 5-7: Característica de convergencia - Valores de número de iteración (evaluación) y valor de pérdidas de potencia activa (MW) para EA, MVMO y MVMO^S

En la siguiente figura, se muestran las características de convergencia para las metodologías RGA, GA, ITS, HSA (reportadas en [11]), de lo cual, se puede verificar que para el SDR - 33 barras, HSA obtiene su solución óptima desde la iteración número 160 [11], mientras que RGA, GA, ITS, convergen a la mejor solución más lentamente que HSA. Pero si se verifican las características de convergencia de MVMO y MVMO^S, estos poseen una notable mejor velocidad de convergencia, además de una evaluación estadística menor que los algoritmos RGA, GA, ITS, HSA (ver Tabla 5-5).

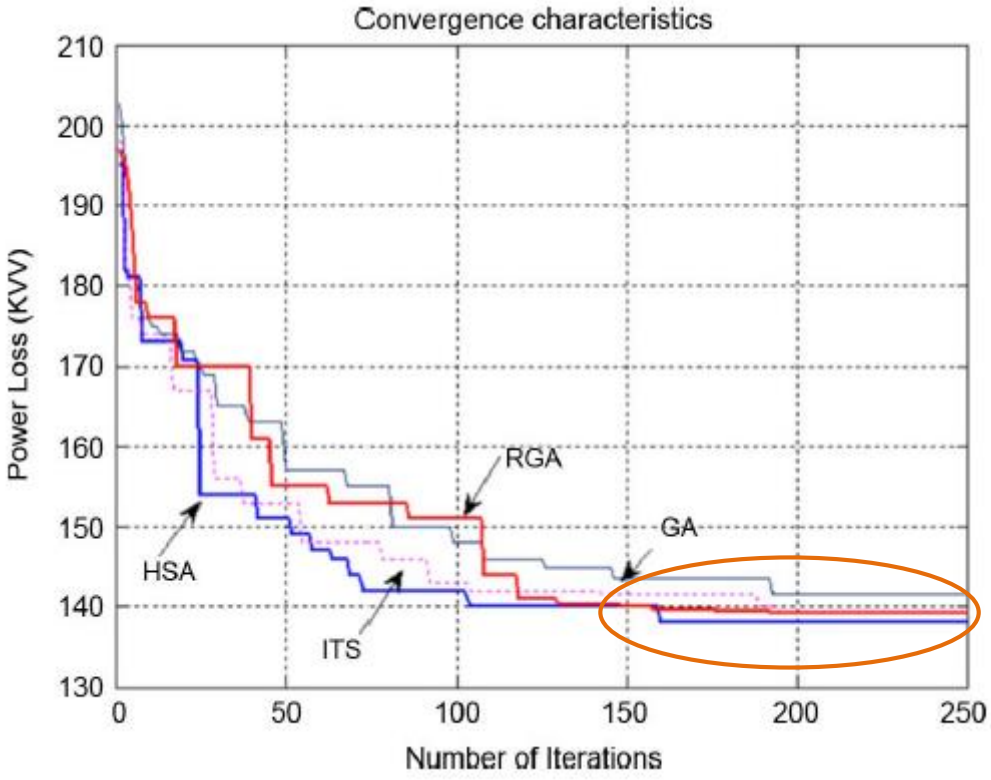


Figura 5-8: Característica de convergencia para SDR 33 barras – RGA, GA, ITS, HSA [11]

CAPÍTULO 6

6 CONCLUSIONES Y RECOMENDACIONES

- Los resultados obtenidos con MVMO y MVMO^S, fueron presentados para un sistema hipotético ampliamente divulgado en la literatura especializada. En consecuencia del tratamiento meta-heurístico empleado para la solución del problema, no se puede garantizar que las propuestas de solución son óptimas, una vez que no es realizada una búsqueda exhaustiva verificándose todas las opciones posibles de seccionamiento. Entretanto, para el sistema probado se obtuvieron resultados similares presentados en las referencias bibliográficas revisadas, mostrando que el algoritmo posee un buen desempeño.

- Las simulaciones fueron llevados para un sistema radial de 33 barras y los resultados obtenidos con MVMO y MVMO^S fueron comparados con otras metodologías, que se indican a continuación: Algoritmo Heurístico de Goswami-Basu HA, Algoritmo Genético GA, Algoritmo Genético Redefinido RGA, Búsqueda Tabú Mejorada ITS, Algoritmo Evolutivo EA y Búsqueda de Armonía HSA, ITS disponibles en la literatura especializada. Los resultados muestran que el algoritmo propuesto puede converger a la solución óptima más rápidamente con valores similares de pérdidas y voltajes a los otros métodos indicados

- Se simularon las metodologías propuestas con diferentes valores de los parámetros de inicialización de MVMO y MVMO^S, que se describen a continuación: Factor de escalamiento de forma f_s, Incremento del factor de forma Δd, Factor inicial de forma d_i, Máximo número de partículas iniciales NP, Máxima distancia Euclidiana d_{max} (distancias entre partículas), Número de ejecuciones independientes de las partículas M, con el fin de determinar con que valores se obtenían mejores resultados, con lo que se demostró que MVMO es efectivo para reducir pérdidas para varios parámetros de inicialización demostrando así su buen desempeño de la adaptación propuesta, no así MVMO^S que a mayor número de partículas y distancia entre ellas, mayor corridas independientes de

las partículas, comienza a eliminar partículas más rápidamente, perjudicando la diversidad de la búsqueda prematuramente, reduciendo la posibilidad de obtener mejores resultados. Es así, que MVMO obtuvo el menor valor de pérdidas para factores de escalamiento inicial y final, 0.9 y 2 respectivamente, con un incremento del factor de forma Δd_0 inicial y final de 0.05 y 0.35, factor inicial de forma $\boldsymbol{d_i}$ igual 1; mientras que MVMO^S lo logró con valores de factores de escalamiento inicial y final, 1 y 2 respectivamente, con un incremento del factor de forma Δd_0 inicial y final de 0.35 y 0.35, factor inicial de forma $\boldsymbol{d_i}$ igual a 1, número de partículas igual a 2, distancia entre partículas igual a 0 y corridas independientes para cada partícula igual a 10.

- Basándose en otros trabajos desarrollados con MVMO y MVMO^S para optimización y en la experiencia de sus resultados, se realizaron las simulaciones para valores de tamaño de población igual a 5, Estrategia 4 (selección aleatoria- secuencial, para la operación de mutación), y con fines de comparación con los resultados del SDR -33 barras reportados en revistas científicas internacionales para HA, GA, RGA, ITS, HSA y EA, se realizaron las simulaciones para 200 corridas y cada repetición de ellas se evaluaron 250 veces, de donde se obtuvieron los siguiente resultados:

  - ✓ La curva de convergencia confirma que el método MVMO puede ser más eficiente buscando la solución óptima o cercana al óptimo para el problema de reconfiguración propuesto
  - ✓ MVMO y MVMO^S, tienen medidas estadísticas más bajas que los algoritmos GA, RGA, ITS.
  - ✓ Con respecto a EA, sus resultados no difieren de otras metodologías como MVMO, MVMO^S y RGA, en cuanto a voltaje y pérdidas, pero el tiempo de procesamiento que precisa es muy alto en comparación con las otras metodologías, haciéndolo menos eficiente que los otros algoritmos en cuanto al tiempo utilizado.
  - ✓ Tanto MVMO, MVMO^S y EA, fueron corridos con el mismo método de flujo de potencia, de lo que se deduce que MVMO no tuvo ninguna ventaja en tiempo de procesamiento para el cálculo de flujo de potencia, mostrando así su mejor desempeño en obtener buenos resultados en el menor tiempo.

- ✓ MVMO se mostró superior a la metodología heurística tradicional GA, dado que presentó mejores resultados de pérdida, voltaje, tiempo de procesamiento y evaluación estadística. Con respecto a HA, MVMO tuvo un mejor resultado en pérdidas de potencia activa, mientras que en voltaje tuvieron un comportamiento similar.
- ✓ HSA e ITS obtuvieron resultados ligeramente mejores que MVMO en cuanto a pérdidas y tiempos de procesamiento, pero la metodología propuesta fue mejor en voltajes y en característica de convergencia.

- Para el SDR-33 barras, es suficiente simularlo con MVMO clásico, cuyo enfoque es a una sola partícula, dado que se obtuvieron buenos resultados con esta metodología. Con respecto a $MVMO^S$, si más bien sus resultados también son buenos en cuanto a pérdidas y voltajes, esta metodología precisa de un tiempo ligeramente mayor que MVMO para obtener un valor mínimo de la función objetivo, dado su algoritmo necesita ejecutar más cálculos por la naturaleza del mismo, además, si se aumentan partículas y distancia entre ellas, y asociado a ellos se incrementan más corridas independientes de las partículas, esté elimina partículas más rápidamente, perjudicando la diversidad de la búsqueda prematuramente, por lo que se reduce la posibilidad de obtener mejores resultados.

- El desarrollo de esta investigación, no tenía como objetivo la discusión genérica del problema de flujo de carga en redes radiales, pero si formaba parte de la metodología determinar el estado de la red para cada configuración radial, necesitando de un programa de flujo de carga radial que se mostrará eficiente tanto en tiempo como en resultados obtenidos, dada la cantidad elevada de número de veces a ser ejecutado durante el proceso de solución, de esta manera se seleccionó un método encontrado en la bibliografía, el mismo que dio buenos resultados para el sistema de prueba cuando fue comparado con otros reportados en publicaciones de revistas internacionales indexadas para el SDR de 33 barras. No obstante, también se simuló el sistema con la aplicación MATPOWER, utilizando la metodología Newton Raphson completo, el cual no convergió para el sistema propuesto, verificándose de esta manera que para SDRs con relación X/R mayor que 1, estos métodos no son los más convenientes para ser aplicados.

- Si más bien el método de flujo de carga en redes radiales dio buenos resultados, se recomienda para trabajos futuros verificar el desempeño de MVMO aplicando el mismo método de flujo de potencia y contrastar los resultados a obtenerse, frente a las metodologías ITS y HSA, que mostraron valores de pérdidas ligeramente mejores (menores en 0.27 kW y 1.49 kW, para ITS y HSA respectivamente) y tiempos de procesamiento inferiores en 1.69 a 2.59 segundos frente a MVMO. En aplicaciones como Smart Grid, los tiempos de respuesta son muy importantes, puesto que para operación en tiempo real en centros de control, 1 segundo es muy significativo sobre todo cuando se requiere reincorporar el sistema cuando existen eventos de salida de elementos de la red.

- Por los resultados obtenidos, se podría asumir que el algoritmo puede resolver el problema de reconfiguración en sistemas reales, por lo que se recomienda para trabajos futuros aplicar MVMO y MVMO^{S} a una red real y de mayor tamaño.

BIBLIOGRAFÍA

[1] F. F. Mesut E.Baran, «Network reconfiguration in distribution systems for loss reduction and load balancing,» *IEEE Transactions on Power Delivery,* vol. 4, nº 2, pp. 1401-1407, April 1989.

[2] S. F. M. A.Y.Abdelaziz, «A Modified Particle Swarn Technique for Distribution System Reconfiguration,» *The Online Journal on Electronics and Electrical Engineering (OJEEE),* 2009.

[3] F. F. Mesut E. Baran, «Optimizal sizing of capacitors placed on a radial distribution system,» *IEEE Transactions on Power Delivery,* vol. 4, nº 1, pp. 735-743, January 1989.

[4] Z. F. L. Z. Dong Zhang, «An improved TS algorithm for loss-minimum reconfiguration,» *ELSEVIER,* 2007.

[5] J. C. C. Amasifen, *Algoritmo evolutivo dedicado à solução do problema de reconfiguração de sistemas de distribuição radiais,* Dissertação de Pós-Graduação - UNESP, Abril 2003.

[6] C. D. G. J. M. P. V. S. Kirkpatrick, «Optimization by Simulated Annealing,» *SCIENCE,* vol. 220, nº 4598, 13 May 1983.

[7] W. G. Zvietcovich, *Reconfiguração de Sistemas de Distribuição de Energia Elétrica Utilizando a Metaheurística Busca en Vizinhança Variável,* Dissertação de Pós-Graduação - UNESP, 2006.

[8] M. Gavrilas, «Heuristic and Metaheuristic Optimization Techniques with Application to Power Systems,» *Power System Department-"Gheorghe Asachi" Technical University of Iasi Romania,* 2010.

[9] J. H. S. S.Civanlar, «Distribution feeder reconfiguration for loss reduction,» *IEEE Transactions on Power Delivery,* vol. 3, nº 3, pp. 1217-1223, July 1988.

[10] M. B. C. d. Oliveira, *Reconfiguração de alimentadores em sistemas de distribuição usando a metaheurística Grasp,* Dissertação de Mestrado - UNESP, Maio 2011.

[11] S. V. L. N. M. R. R. A. S. R. Rayapudi Srinivasa Rao, «Optimal Network Reconfiguration of Large – Scale Distribution System Using Harmony Search Algorithm,» *IEEE Transactions on Power System,* vol. 26, nº 3, pp. 1080-1088, Agosto 2011.

[12] O. G.-B. A. S.-A. Paola Pezzinia, «Optimization techniques to improve energy efficiency in power systems,» *ELSEVIER-Renewable and Sustainable Energy Reviews,* 2011.

[13] G. F. d. M. G. M. J. A. S. Daniel O. Anaut, «Optimización de redes eléctricas mediante la aplicación de algoritmos genéticos,» *SCIELO,* vol. 20, nº 4, pp. 137-148, 2009.

[14] P. P. Cruz, Inteligencia artificial con aplicaciones a la ingeniería, México: Alfaomega, 2010.

[15] P. A. F. Brandini, *Metaheurística Particle Swarm Utilizada para Alocação Ótima de Bancos de Capacitores em Sistemas de Distribuição Radial,* Dissertação Mestre - UNESP, Março de 2007.

[16] I. D.P.Kothari, Sistemas Eléctricos de Potencia, México: McGraw-Hill Interamericana, 2008.

[17] M. A. d. N. Guimarães, *Plataforma Integrada para o Planejamento de Sistemas de Distribuição de Energia Elétrica Utilizando Metaheurísticas,* Dissertação Doutorado - UNICAMP, Setembro de 2009.

[18] L. F. O. Pizzali, *Cálculo de fluxo de potência em redes de distribuição com modelagem a quatro fios,* Universidade Estadual Paulista - Dissertação Mestre, Maio de 2003.

[19] R. Cespedes, «New Method For The Analysis of Distribution Networks,» *IEEE Transactions on Power Delivery,* vol. 5, nº 1, pp. 391-396, 1990.

[20] R. R. B. Venkatesh, «Data structure for radial distribution load flow analysis,» *IEE Proceeding in Generation, Transmission and Distribution,* vol. 150, nº 1, pp. 101-106, January 2003.

[21] D. S. Carol S.Cheng, «A Three-phase Power Flow Method for Real-Time Distribution System Analysis,» *IEEE Transactions on Power Systems,* vol. 10, nº 2, pp. 671-679, 1995.

[22] C. S. C. Fan Zhang, «A Modified Newton for Radial Distribution System Power Flow Analysis,» *IEEE Transactions on Power Systems,* vol. 12, nº 1, pp. 389-397, 1997.

[23] A. I. Rost, *Distribution systems with distributed generation: analysis and operation,* Thesis of Masters of Science in Engineering. The University of Western Ontario., 2006.

[24] K. V. G. M. d. C. F.S. Pereira, *Distribution System Reconfiguration for Loss Reduction Based on Ant Colony Behavior,* Venezuela: IEEE PES Transmission and Distribution Conference and Exposition Latin America, 2006.

[25] MATPOWER, «A MATLAB Power System Simulation - http://www.pserc.cornell.edu/matpower/,» [En línea].

[26] H. H. Mudathir F. Akorede, «Teaching Power System Analysis Courses using MATPOWER,» *IEEE Xplore,* pp. 45-51, 2009.

[27] J. A. Y. Morón, Sistemas Eléctricos de Distribución, México: Reverté, 2009.

[28] V. J. Garcia, *Metaheurísticas multiobjetivo para o problema de restauração do serviço em redes de distribuição de energia elétrica,* Tese (doutorado), UNICAMP, 2005.

[29] N. F. E. Command, «http://www.wbdg.org/,» April 1990. [En línea]. Available: http://www.wbdg.org/ccb/NAVFAC/OPER/mo201.pdf.

[30] ieee, «http://www.ieee.li/,» [En línea]. Available: http://www.ieee.li/pdf/viewgraphs/automating_power_distribution_system.pdf.

[31] G. T. H. D. Haughton, «Smart Distribution System Design: Automatic Reconfiguration for Improved Reliability,» de *Power and Energy Society General Meeting, 2010 IEEE*, Tempe, AZ, USA , July 2010.

[32] X. R. R. S. J. M. F. L. V.Spitsa, «On the Transient Behavior of Large-Scale Distribution Networks During Automatic Feeder Reconfiguration,» *IEEE Transactions on Smart Grid,* vol. 3, nº 2, June 2012.

[33] P. E. O. Yumbla, *Solución del problema de flujos de potencia óptimo con restricciones de seguridad por un optimizador de partículas modificado,* Tesis de grado Doctor , Centro de Investigación y de Estudios Avanzados del I.P.N., 2008.

[34] G. K. V. N. W. István Erlich, «A Mean-Variance Optimization Algorithm,» *IEEE ,* pp. 344-349, 2010.

[35] I. E. José L. Rueda, *Optimal Dispatch of Reactive Power Sources by Using MVMO SWARM Optimization,* University Duisburg-Essen, 2013.

[36] I. E. J. L. R. Worawat Nakawiro, «A Novel Optimization Algorithm for Optimal Reactive Power Dispatch: A Comparative Study.,» *IEEE,* 2011.

[37] P. I. E. University Duisburg-Essen, «https://www.uni-due.de/mvmo/,» Diciembre 2013. [En línea].

[38] P. I. E. University Duisburg-Essen, «https://www.uni-due.de/mvmo/,» Mayo 2013. [En línea].

[39] F. C. R. A. José R.S. Mantovani, «Reconfiguração de sistemas de distribuição radiais utilizando o critério de queda de tensão,» *SBA Controle&Automação ,* vol. 11, nº 03, pp. 150-159, 2000.

[40] M. O.Añó, «Economic Dispatch of Energy and Reserve in Competitive Markets Using Meta-heuristic Algorithms,» *IEEE Latin America Transactions,* vol. 11, nº 1, pp. 473-478, 2013.

[41] S. C. S. S. B. Amanulla, «Reconfiguration of Power Distribution Systems Considering Reliability and Power Loss,» *IEEE TRANSACTIONS ON POWER DELIVERY,* vol. 27, nº 2, pp. 918-926, 2012.

[42] S. S.K.Goswami, «A New Algorithm for the Reconfiguration of Distribution Feeders,» *IEEE Transaction on Power Delivery,* vol. 7, nº 3, pp. 1484-1490, 1992.

[43] X. L. F. L. F. Z. P. P. N. M. H.Qi, «A Resilient Real-Time System Design for a Secure and Reconfigurable Power Grid,» *IEEE Transactions on Smart Grid,* vol. 2, nº 4, December 2011.

[44] «[PPT] METAHEURISTICAS,» [En línea]. Available: www.fing.edu.uy/inco/grupos/invop/mh/material/pres4.ppt.

[45] R. R. H. G. B.Venkatesh, «Optimal reconfiguration of radial distribution systems to maximize loadability,» *IEEE Transactions on Power Systems,* vol. 19, nº 1, pp. 260-266, February 2004.

ANEXO 1: DATOS CASOS DE ESTUDIO

CASO 6 BARRAS

Line	From	To	R	X	R	X	Load Receiving End Bus	
No	Bus	Bus	Ω	Ω	pu	pu	Real Power Load (kW)	Reactive Power Load (kVAR)
1	1	2	1,0578	0,5289	0,0066	0,0033	280	1280
2	2	3	1,0578	0,5289	0,0066	0,0033	1280	1280
3	2	4	0,2564	0,0962	0,0016	0,0006	320	160
4	3	5	0,8174	0,0801	0,0051	0,0005	1600	800
5	4	6	0,4327	0,1923	0,0027	0,0012	740	370
6	3	4	0,0481	0,0321	0,0003	0,0002	TIE LINES	
7	4	5	0,0801	0,0801	0,0005	0,0005	TIE LINES	
8	5	6	0,5289	0,2404	0,0033	0,0015	TIE LINES	
TOTAL							4220	3890

CASO 33 BARRAS

Line	From	To	R	X	R	X	Load Receiving End Bus	
No	Bus	Bus	Ω	Ω	pu	pu	Real Power Load (kW)	Reactive Power Load (kVAR)
1	1	2	0,0922	0,0470	0,0006	0,0003	100	60
2	2	3	0,4930	0,2511	0,0031	0,0016	90	40
3	3	4	0,366	0,1864	0,0023	0,0012	120	80
4	4	5	0,3811	0,1941	0,0024	0,0012	60	30
5	5	6	0,8190	0,7070	0,0051	0,0044	60	20
6	6	7	0,1872	0,6188	0,0012	0,0039	200	100
7	7	8	**1,7114**	**1,2351**	**0,0044**	0,0015	200	100
8	8	9	1,0300	0,7400	0,0064	0,0046	60	20
9	9	10	1,0440	0,7400	0,0065	0,0046	60	20
10	10	11	0,1966	0,0650	0,0012	0,00041	45	30
11	11	12	0,3744	0,1238	0,0023	0,0008	60	35
12	12	13	1,4680	1,1550	0,0092	0,0072	60	35
13	13	14	0,5416	0,7129	0,0034	0,0044	120	80
14	14	15	0,5910	0,5260	0,0037	0,0033	60	10
15	15	16	0,7463	0,5450	0,0047	0,0034	60	20

Line No	From Bus	To Bus	R Ω	X Ω	R pu	X pu	Load Receiving End Bus	
							Real Power Load (kW)	Reactive Po-wer Load (kVAR)
16	16	17	1,2890	1,7210	0,0080	0,0107	60	20
17	17	18	0,7320	0,5740	0,0046	0,0036	90	40
18	18	19	0,1640	0,1565	0,0010	0,0010	90	40
19	19	20	1,5042	1,3554	0,0094	0,0085	90	40
20	20	21	0,4095	0,4784	0,0026	0,0030	90	40
21	21	22	0,7089	0,9373	0,0044	0,0058	90	40
22	22	23	0,4512	0,3083	0,0028	0,0019	90	50
23	23	24	0,8980	0,7091	0,0056	0,0044	420	200
24	24	25	0,8960	0,7011	0,0056	0,0044	420	200
25	25	26	0,2030	0,1034	0,0013	0,0006	60	25
26	26	27	0,2842	0,1447	0,0018	0,0009	60	25
27	27	28	1,0590	0,9337	0,0066	0,0058	60	20
28	28	29	0,8042	0,7006	0,0050	0,0044	120	70
29	29	30	0,5075	0,2585	0,0032	0,0016	200	600
30	30	31	0,9744	0,9630	0,0061	0,0060	150	70
31	31	32	0,3105	0,3619	0,0019	0,0023	210	100
32	32	33	0,3410	0,5302	0,0021	0,0033	60	40
33*	**21**	**8**	**2,0000**	**2,0000**	**0,0125**	**0,0125**	**TIE LINES**	
34*	**9**	**15**	**2,0000**	**2,0000**	**0,0125**	**0,0125**	**TIE LINES**	
35*	**12**	**22**	**2,0000**	**2,0000**	**0,0125**	**0,0125**	**TIE LINES**	
36*	**18**	**33**	**0,5000**	**0,5000**	**0,0031**	**0,0031**	**TIE LINES**	
37*	**25**	**19**	**0,5000**	**0,5000**	**0,0031**	**0,0031**	**TIE LINES**	
TOTAL							**3715**	**2300**

ANEXO 2: SISTEMA DE 33 BARRAS - SIMULACIÓN UTILIZANDO MATPOWER Y RESULTADOS

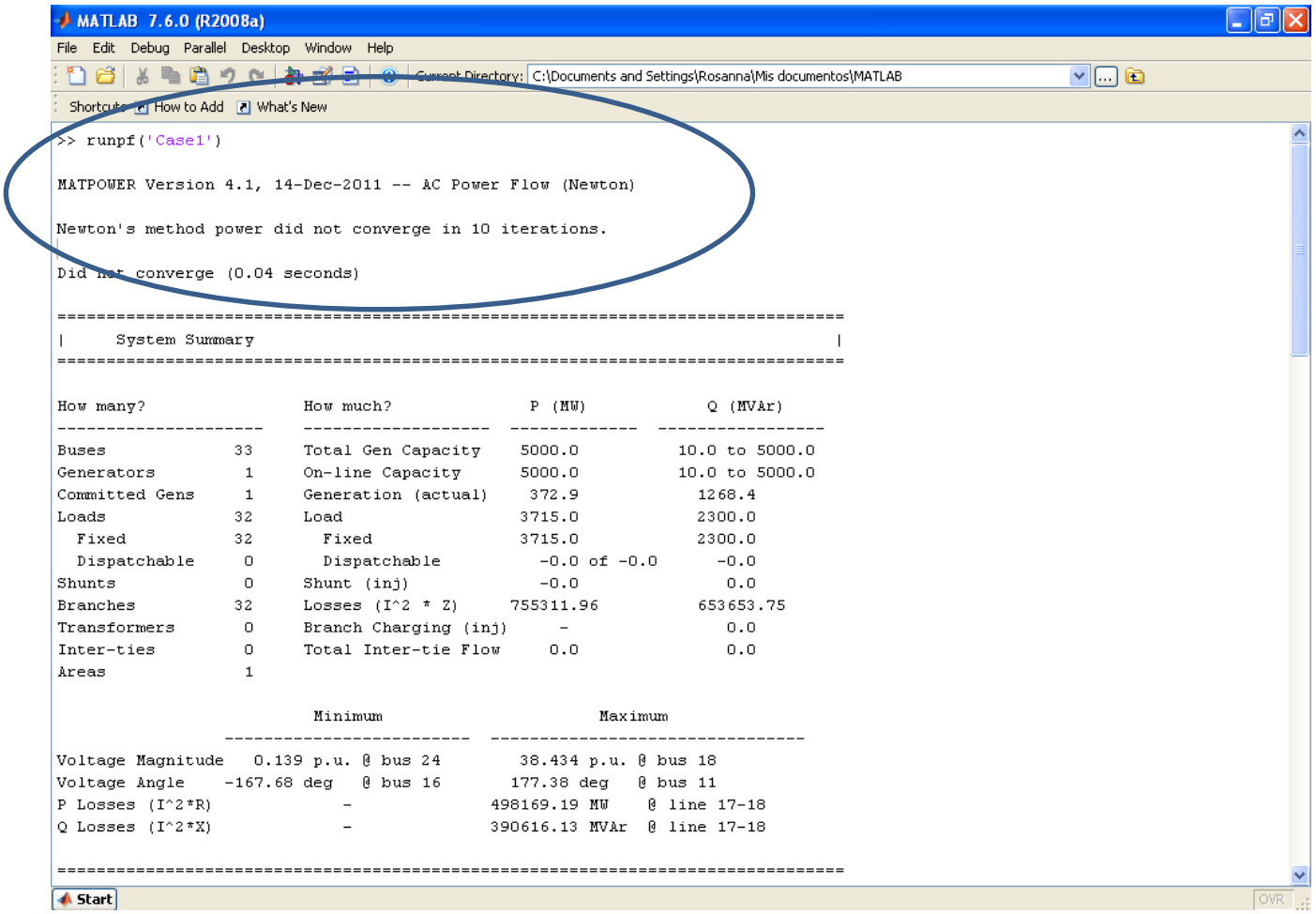
Case1.m

```
function mpc = Case1


mpc.baseMVA=100;

          %  bus type P    Q  G  B A  Vm  Va basekV  zone  Vmax  Vmin
  mpc.bus=[  1   3    0    0  0  0 1   1   0  12.66     1  1.06  0.70
             2   1  100   60  0  0 1   1   0  12.66     1  1.06  0.70
             3   1   90   40  0  0 1   1   0  12.66     1  1.06  0.70
             4   1  120   80  0  0 1   1   0  12.66     1  1.06  0.70
             5   1   60   30  0  0 1   1   0  12.66     1  1.06  0.70
             6   1   60   20  0  0 1   1   0  12.66     1  1.06  0.70
             7   1  200  100  0  0 1   1   0  12.66     1  1.06  0.70
             8   1  200  100  0  0 1   1   0  12.66     1  1.06  0.70
             9   1   60   20  0  0 1   1   0  12.66     1  1.06  0.70
             10  1   60   20  0  0 1   1   0  12.66     1  1.06  0.70
             11  1   45   30  0  0 1   1   0  12.66     1  1.06  0.70
             12  1   60   35  0  0 1   1   0  12.66     1  1.06  0.70
             13  1   60   35  0  0 1   1   0  12.66     1  1.06  0.70
             14  1  120   80  0  0 1   1   0  12.66     1  1.06  0.70
             15  1   60   10  0  0 1   1   0  12.66     1  1.06  0.70
             16  1   60   20  0  0 1   1   0  12.66     1  1.06  0.70
             17  1   60   20  0  0 1   1   0  12.66     1  1.06  0.70
             18  1   90   40  0  0 1   1   0  12.66     1  1.06  0.70
             19  1   90   40  0  0 1   1   0  12.66     1  1.06  0.70
             20  1   90   40  0  0 1   1   0  12.66     1  1.06  0.70
             21  1   90   40  0  0 1   1   0  12.66     1  1.06  0.70
             22  1   90   40  0  0 1   1   0  12.66     1  1.06  0.70
             23  1   90   50  0  0 1   1   0  12.66     1  1.06  0.70
             24  1  420  200  0  0 1   1   0  12.66     1  1.06  0.70
             25  1  420  200  0  0 1   1   0  12.66     1  1.06  0.70
             26  1   60   25  0  0 1   1   0  12.66     1  1.06  0.70
             27  1   60   25  0  0 1   1   0  12.66     1  1.06  0.70
             28  1   60   20  0  0 1   1   0  12.66     1  1.06  0.70
             29  1  120   70  0  0 1   1   0  12.66     1  1.06  0.70
             30  1  200  600  0  0 1   1   0  12.66     1  1.06  0.70
             31  1  150   70  0  0 1   1   0  12.66     1  1.06  0.70
             32  1  210  100  0  0 1   1   0  12.66     1  1.06  0.70
             33  1   60   40  0  0 1   1   0  12.66     1  1.06  0.70];
```

```
        % bus Pg  Qg  Qmax  Qmin    Vg   mBase  status  Pmax  Pmin
mpc.gen=[ 1  0   0   5000   10   1.05   100      1     5000   10 ];
```

```
           %fbus tbus    r        x      b  rateA rateB rateC ratio angle
status
mpc.branch=[ 1    2     0.0575 0.0293   0    150    0      0      0      0
1
             2    3     0.3076 0.1567   0    150    0      0      0      0
1
             3    4     0.2284 0.1163   0    150    0      0      0      0
1
             4    5     0.2378 0.1211   0    150    0      0      0      0
1
             5    6     0.5110 0.4411   0    150    0      0      0      0
1
             6    7     0.116  0.3861   0    150    0      0      0      0
1
             7    8     0.4439 0.1467   0    150    0      0      0      0
1
             8    9     0.6426 0.4617   0    150    0      0      0      0
1
             9   10     0.6514 0.4617   0    150    0      0      0      0
1
            10   11     0.1227 0.0406   0    150    0      0      0      0
1
            11   12     0.2336 0.0772   0    150    0      0      0      0
1
            12   13     0.9159 0.7206   0    150    0      0      0      0
1
            13   14     0.3379 0.4448   0    150    0      0      0      0
1
            14   15     0.3687 0.3282   0    150    0      0      0      0
1
            15   16     0.4656 0.3400   0    150    0      0      0      0
1
            16   17     0.8042 1.0738   0    150    0      0      0      0
1
            17   18     0.4567 0.3581   0    150    0      0      0      0
1
             2   19     0.1023 0.0976   0    150    0      0      0      0
1
            19   20     0.9385 0.8457   0    150    0      0      0      0
1
            20   21     0.2555 0.2985   0    150    0      0      0      0
1
            21   22     0.4423 0.5848   0    150    0      0      0      0
1
             3   23     0.2815 0.1924   0    150    0      0      0      0
1
            23   24     0.5603 0.4424   0    150    0      0      0      0
1
            24   25     0.5590 0.4374   0    150    0      0      0      0
1
             6   26     0.1267 0.0645   0    150    0      0      0      0
1
```

```
                26    27      0.1773 0.0903    0     150     0       0       0       0
  1
                27    28      0.6607 0.5826    0     150     0       0       0       0
  1
                28    29      0.5018 0.4371    0     150     0       0       0       0
  1
                29    30      0.3166 0.1613    0     150     0       0       0       0
  1
                30    31      0.6080 0.6008    0     150     0       0       0       0
  1
                31    32      0.1937 0.2258    0     150     0       0       0       0
  1
                32    33      0.2128 0.3308    0     150     0       0       0       0
  1 ];
```



ANEXO 3: DATOS PARA SIMULACIÓN DE FLUJOS DE POTENCIA PARA SISTEMA DE 33 BARRAS Y CÓDIGOS .m (ARCHIVO .TXT)

Contiene los códigos en MATLAB para el cálculo de flujo de potencia para el SDR 33 barras. El código principal para correr esta aplicación se denomina **branch_lf.m**

37i.txt:

```
12.66-volts-radial-33-node-system
2013
PEAK_LOAD
 001
 33  0033  0001 1     0    0 32 0 37 0 0 0 1
 1   0.000 1   1.050 0.95 100
MAIN
1   1 1  BUS1    0.0  0.0  0.000 100.00 -100.000 1.0000
1   1 2  BUS2  100.0 60.0
2   1 3  BUS3   90.0 40.0
3   1 4  BUS4  120.0 80.0
4   1 5  BUS5   60.0 30.0
5   1 6  BUS6   60.0 20.0
6   1 7  BUS7  200.0 100.0
7   1 8  BUS8  200.0 100.0
8   1 9  BUS9   60.0 20.0
9   1 10 BUS10  60.0 20.0
10  1 11 BUS11  45.0 30.0
11  1 12 BUS12  60.0 35.0
12  1 13 BUS13  60.0 35.0
13  1 14 BUS14 120.0 80.0
14  1 15 BUS15  60.0 10.0
15  1 16 BUS16  60.0 20.0
16  1 17 BUS17  60.0 20.0
17  1 18 BUS18  90.0 40.0
18  1 19 BUS19  90.0 40.0
19  1 20 BUS20  90.0 40.0
20  1 21 BUS21  90.0 40.0
21  1 22 BUS22  90.0 40.0
22  1 23 BUS23  90.0 50.0
23  1 24 BUS24 420.0 200.0
24  1 25 BUS25 420.0 200.0
25  1 26 BUS26  60.0 25.0
26  1 27 BUS27  60.0 25.0
27  1 28 BUS28  60.0 20.0
28  1 29 BUS29 120.0 70.0
29  1 30 BUS30 200.0 600.0
30  1 31 BUS31 150.0 70.0
```

```
31  1 32 BUS32 210.0 100.0
32  1 33 BUS33  60.0 40.0
1   1 2  1 0.0922 0.0470 0.00000 00.000 1.000  12.66
2   2 3  1 0.4930 0.2511 0.00000 00.000 1.000  12.66
3   3 4  1 0.3660 0.1864 0.00000 00.000 1.000  12.66
4   4 5  1 0.3811 0.1941 0.00000 00.000 1.000  12.66
5   5 6  1 0.8190 0.7070 0.00000 00.000 1.000  12.66
6   6 7  1 0.1872 0.6188 0.00000 00.000 1.000  12.66
7   7 8  1 1.7114 1.2351 0.00000 00.000 1.000  12.66
8   8 9  1 1.0300 0.7400 0.00000 00.000 1.000  12.66
9   9 10 1 1.0440 0.7400 0.00000 00.000 1.000  12.66
10 10 11 1 0.1966 0.0650 0.00000 00.000 1.000  12.66
11 11 12 1 0.3744 0.1238 0.00000 00.000 1.000  12.66
12 12 13 1 1.4680 1.1550 0.00000 00.000 1.000  12.66
13 13 14 1 0.5416 0.7129 0.00000 00.000 1.000  12.66
14 14 15 1 0.5910 0.5260 0.00000 00.000 1.000  12.66
15 15 16 1 0.7463 0.5450 0.00000 00.000 1.000  12.66
16 16 17 1 1.2890 1.7210 0.00000 00.000 1.000  12.66
17 17 18 1 0.7320 0.5740 0.00000 00.000 1.000  12.66
18 2  19 1 0.1640 0.1565 0.00000 00.000 1.000  12.66
19 19 20 1 1.5042 1.3554 0.00000 00.000 1.000  12.66
20 20 21 1 0.4095 0.4784 0.00000 00.000 1.000  12.66
21 21 22 1 0.7089 0.9373 0.00000 00.000 1.000  12.66
22 3  23 1 0.4512 0.3083 0.00000 00.000 1.000  12.66
23 23 24 1 0.8980 0.7091 0.00000 00.000 1.000  12.66
24 24 25 1 0.8960 0.7011 0.00000 00.000 1.000  12.66
25 6  26 1 0.2030 0.1034 0.00000 00.000 1.000  12.66
26 26 27 1 0.2842 0.1447 0.00000 00.000 1.000  12.66
27 27 28 1 1.0590 0.9337 0.00000 00.000 1.000  12.66
28 28 29 1 0.8042 0.7006 0.00000 00.000 1.000  12.66
29 29 30 1 0.5075 0.2585 0.00000 00.000 1.000  12.66
30 30 31 1 0.9744 0.9630 0.00000 00.000 1.000  12.66
31 31 32 1 0.3105 0.3619 0.00000 00.000 1.000  12.66
32 32 33 1 0.3410 0.5302 0.00000 00.000 1.000  12.66
33 08 21 1 2.0000 2.0000 0.00000 00.000 1.000  12.66
34 09 15 1 2.0000 2.0000 0.00000 00.000 1.000  12.66
35 12 22 1 2.0000 2.0000 0.00000 00.000 1.000  12.66
36 18 33 1 0.5000 0.5000 0.00000 00.000 1.000  12.66
37 25 29 1 0.5000 0.5000 0.00000 00.000 1.000  12.66

5
9 33 18 19 20 06 07 03 04 05
4 34 12 13 14
6 35 08 11 10 09 21
8 36 32 17 16 15 29 30 31
8 37 24 23 22 28 25 26 27
```

branch_lf.m

```
% This is the main m-file for the RDS with DG load flow.
% Authored by Alex Rost.

% MODIFICADO RLOOR: This is the main m-file for the RDS withOUT DG load
flow.
% SISTEMA DE 33 BARRAS

global NB
global PG QG PD QD V DEL QGMAX QGMIN VSH
global FB TB ZL

clear all;
close all;
clc;

tic
data;
YBUS;
%line_change;
FB_TB_shorten;
[output] = system_array(1,0);
output = fliplr(output);
%G_modify; MODIFICADO RLOOR

iter = 0;
max_iter = 1000;
min_err = 0.001;
err = 100;

 while err > min_err && iter < max_iter
     VA_acc = zeros(2,NB);
     q_generator = zeros(NB,1);
     iter = iter+1;
        for br = 1:size(output,2)
             this_branch = output(:,br)
             parent_bus = this_branch(1)
             num_buses = this_branch(2)
             num_branches = this_branch(3)
             branch_segment = [parent_bus  this_branch(4:3+num_buses)']
             branch_segment = fliplr(branch_segment)

                 for i=1:num_buses
                     send_bus = branch_segment(i+1);
                        if send_bus > 0
                           receive_bus = branch_segment(i);
                           v_send = V(send_bus);
                           delta_send = DEL(send_bus);
                           temp_1 = FB - min([send_bus receive_bus]);
                           temp_2 = TB - max([send_bus receive_bus]);
                           r_x_ind = find(or(temp_1,temp_2)==0);
                           r = real(ZL(r_x_ind));
                           x = imag(ZL(r_x_ind));
                           p_inj = VA_acc(1,receive_bus) + PD(re-
ceive_bus) - PG(receive_bus);

                                 if PG(receive_bus) == 0
```

```
                                    q_inj = VA_acc(2,receive_bus) + QD(re-
ceive_bus) - QG(receive_bus);
                                    [v_receive] =
v_calc(r,x,p_inj,q_inj,v_send);
                                else
                                    q_violation_flag = 0;
                                    v_receive = VSH(receive_bus);
                                    [q_inj] = q_calc(r, x, p_inj , v_send,
v_receive);
                                    q_gen = q_inj - QD(receive_bus) -
VA_acc(2,receive_bus);

                                    if q_gen > QGMAX(receive_bus)
                                       q_gen = QGMAX(receive_bus);
                                       q_violation_flag = 1;
                                      elseif q_gen < QGMIN(receive_bus)
                                        q_gen = QGMIN(receive_bus);
                                        q_violation_flag = 1;
                                     end
                                         q_generator(receive_bus) = q_gen;
                                         if q_violation_flag == 1
                                             q_inj = q_gen + QD(re-
ceive_bus) + VA_acc(2,receive_bus);
                                             [v_receive] =
v_calc(r,x,p_inj,q_inj,v_send);
                                          end
                                end
                                 [delta_receive,p_loss,q_loss] =
delta_ploss_qloss(r,x,p_inj,q_inj,v_send,v_receive,delta_send);
%ploss,qloss
                                V(receive_bus) = v_receive;  %magnitud de
voltaje en p.u
                                DEL(receive_bus) = delta_receive; % angulo
en radianes de los voltajes nodales
                                VA_acc(1,send_bus) = VA_acc(1,send_bus) +
p_inj + p_loss;
                                VA_acc(2,send_bus) = VA_acc(2,send_bus) +
q_inj + q_loss;
                        end
                end
        end
       [err] = s_balance_YB(q_generator);

 end



iter
err
toc
```

line_change.m

```
global SWSTI SWST

% This file allows for a change in the line switching so that the load
flow
% can be tested for different system configurations.
% authored by alex rost

switch_out = 18;
switch_in = 33;
SWST(find(SWSTI == switch_out)) = 0;
SWST(find(SWSTI == switch_in)) = 1
```

En el ANEXO 6, se presentan los códigos branch_array.m, data.m, delta_ploss_qloss, FB_TB_shorten.m, s_balance_YB.m, SWMAT.m, system_array, v_calc.m, YBUS.m obtenidos de [23], para simular el SDR 33 barras.

ANEXO 4: PARÁMETROS DE INICIALIZACIÓN - MVMO y MVMO^S

MVMO			
Parámetros de inicialización		**Valores típicos**	**Observaciones**
1	Tamaño del archivo (población) ***n***	2-5	*n=5, es suficiente para el tamaño del archivo* **[38]**
2	Número de variables seleccionadas simultáneamente $\mathbf{x_i}$	-	*Depende del problema de optimización Número de parámetros a ser optimizados, i.e. la dimensión del problema del problema*
3	Método de selección – Estrategias de mutación. *(Estrategia de la selección de variable para la creación de descendencia)*	1-4	*Estrategias 2- 4 generalmente se desempeñan mejor que las estrategia 1* [36].
4	Factor de escalamiento de forma, f_s $f^*_{S\_ini}$ $f^*_{S\_final}$	$f^*_{S\_ini}$*=0.9……1.0* $f^*_{S\_final}$*=1….3*	*Cuando la exactitud necesita ser mejorada* $f_s>1$ [35] *Cuando se desea aumentar la búsqueda global* $f_s<1$ [35] *0.5 -1 mejor exploración* [35] *>1 mejor explotación* [35]
5	Incremento del factor de forma Δd *Δd es variado aleatoriamente alrededor de* $1+\Delta d_0$	$0.40>\Delta d_0>0.01$	Δd_0*: Un valor alto define diverficación de búsqueda global más amplio, un valor menor conduce a concentrarse en la búsqueda local ayudando a mejorar la exactitud* [35].
6	Factor inicial de forma d_i	*1-5*	*Son definidos para todas las variables en el inicio de la optimización.* [35], [38]
7	Máximo número de iteraciones (evaluaciones)		*Depende de la dimensión del problema de optimización*
8	Máximo número de corridas (repeticiones)		*Depende de la dimensión del problema de optimización*
MVMO^S			
Parámetros iniciales		**Valores típicos**	**Observaciones**
1	Máximo número de partículas iniciales **NP**	-	*Depende de la dimensión del problema de optimización*
2	Máxima distancia Euclidiana $\boldsymbol{d_{max}}$ (distancias entre partículas), para la solución del mejor global.	0-1	*0 implica que las partículas nunca morirán.* *1 resultará en la muerte de todas excepto el mejor global.* [38]
3	Número de ejecuciones independientes de las partículas **M**	-	*Depende de la dimensión del problema de optimización*

ANEXO 5: RESULTADOS MVMO Y MVMO^S – SDR 33 BARRAS

Resultados obtenidos con MVMO

==========

Mean-Variance Mapping Optimization Results

==========

Nomenclature:

FE-No. => Objective function evaluation number

NAP => Number of active particles

AGBP => Actual global best particle

Fitn => Global best fitness: Objective function plus penalty due to constraint violation

Run => Actual run

FE-No.	NAP	AGBP	Fitn	Run
1	1	1	43,7986	46
25	1	1	0,1395512	46
50	1	1	0,1395512	46
75	1	1	0,1395512	46
100	1	1	0,1395512	46
125	1	1	0,1395512	46
150	1	1	0,1395512	46
175	1	1	0,1395512	46
200	1	1	0,1395512	46
225	1	1	0,1395512	46
250	1	1	0,1395512	46

==========

Resultados obtenidos con MVMO^S

```
================================================================================
            Mean-Variance Mapping Optimization Results
================================================================================
Nomenclature:
FE-No. => Objective function evaluation number
NAP    => Number of active particles
AGBP   => Actual global best particle
Fitn   => Global best fitness: Objective function plus penalty due to constraint violation
Run    => Actual run

FE-No.     NAP       AGBP         Fitn          Run
------    ------   ---------   ---------------   --------
   1       2        1          7.8073043         164
  25       2        2           0.1698319        164
  50       2        2           0.1698319        164
  75       2        2           0.1468387        164
 100       2        1           0.1395514        164
 125       2        1           0.1395514        164
 150       2        1           0.1395514        164
 175       2        1           0.1395514        164
 200       2        1           0.1395514        164
 225       2        1           0.1395514        164
 250       2        1           0.1395514        164
================================================================================
```

ANEXO 6: CÓDIGOS EN MATLAB PARA OPTIMIZACIÓN MEDIANTE MVMO (y variante MVMO^S)

Contiene los códigos en MATLAB para la metodología propuesta para el SDR 33 barras. El código principal para correr esta aplicación se denomina MVMOS_RRPLM.m (Ver CD-ANEXO 6)

Datos de Ingreso (modificado de [23])

37i.txt

12.66-volts-radial-33-node-system

2013

PEAK_LOAD

001

33 0033 0001 1 0 0 32 0 37 0 0 0 1

1 0.000 1 1.050 0.95 100

MAIN

1 1 1 BUS1 0.0 0.0 0.000 100.00 -100.000 1.0000

1 1 2 BUS2 100.0 60.0

2 1 3 BUS3 90.0 40.0

3 1 4 BUS4 120.0 80.0

4 1 5 BUS5 60.0 30.0

5 1 6 BUS6 60.0 20.0

6 1 7 BUS7 200.0 100.0

7 1 8 BUS8 200.0 100.0

8 1 9 BUS9 60.0 20.0

9 1 10 BUS10 60.0 20.0

10 1 11 BUS11 45.0 30.0

11 1 12 BUS12 60.0 35.0

12 1 13 BUS13 60.0 35.0

13 1 14 BUS14 120.0 80.0

14 1 15 BUS15 60.0 10.0

15 1 16 BUS16 60.0 20.0

16 1 17 BUS17 60.0 20.0

17 1 18 BUS18 90.0 40.0

18 1 19 BUS19 90.0 40.0

19 1 20 BUS20 90.0 40.0

20 1 21 BUS21 90.0 40.0

21 1 22 BUS22 90.0 40.0

22 1 23 BUS23 90.0 50.0

23 1 24 BUS24 420.0 200.0

24 1 25 BUS25 420.0 200.0

25 1 26 BUS26 60.0 25.0

26 1 27 BUS27 60.0 25.0

27 1 28 BUS28 60.0 20.0

28 1 29 BUS29 120.0 70.0

29 1 30 BUS30 200.0 600.0

30 1 31 BUS31 150.0 70.0

31 1 32 BUS32 210.0 100.0

32 1 33 BUS33 60.0 40.0

1 1 2 1 0.0922 0.0470 0.00000 00.000 1.000 12.66

2 2 3 1 0.4930 0.2511 0.00000 00.000 1.000 12.66

3 3 4 1 0.3660 0.1864 0.00000 00.000 1.000 12.66

4 4 5 1 0.3811 0.1941 0.00000 00.000 1.000 12.66

5 5 6 1 0.8190 0.7070 0.00000 00.000 1.000 12.66

6 6 7 1 0.1872 0.6188 0.00000 00.000 1.000 12.66

7 7 8 1 0.7114 0.2351 0.00000 00.000 1.000 12.66

8 8 9 1 1.0300 0.7400 0.00000 00.000 1.000 12.66

9 9 10 1 1.0440 0.7400 0.00000 00.000 1.000 12.66

10 10 11 1 0.1966 0.0650 0.00000 00.000 1.000 12.66

11 11 12 1 0.3744 0.1238 0.00000 00.000 1.000 12.66

12 12 13 1 1.4680 1.1550 0.00000 00.000 1.000 12.66

13 13 14 1 0.5416 0.7129 0.00000 00.000 1.000 12.66

14 14 15 1 0.5910 0.5260 0.00000 00.000 1.000 12.66

15 15 16 1 0.7463 0.5450 0.00000 00.000 1.000 12.66

16 16 17 1 1.2890 1.7210 0.00000 00.000 1.000 12.66

17 17 18 1 0.7320 0.5740 0.00000 00.000 1.000 12.66

18 2 19 1 0.1640 0.1565 0.00000 00.000 1.000 12.66
19 19 20 1 1.5042 1.3554 0.00000 00.000 1.000 12.66
20 20 21 1 0.4095 0.4784 0.00000 00.000 1.000 12.66
21 21 22 1 0.7089 0.9373 0.00000 00.000 1.000 12.66
22 3 23 1 0.4512 0.3083 0.00000 00.000 1.000 12.66
23 23 24 1 0.8980 0.7091 0.00000 00.000 1.000 12.66
24 24 25 1 0.8960 0.7011 0.00000 00.000 1.000 12.66
25 6 26 1 0.2030 0.1034 0.00000 00.000 1.000 12.66
26 26 27 1 0.2842 0.1447 0.00000 00.000 1.000 12.66
27 27 28 1 1.0590 0.9337 0.00000 00.000 1.000 12.66
28 28 29 1 0.8042 0.7006 0.00000 00.000 1.000 12.66
29 29 30 1 0.5075 0.2585 0.00000 00.000 1.000 12.66
30 30 31 1 0.9744 0.9630 0.00000 00.000 1.000 12.66
31 31 32 1 0.3105 0.3619 0.00000 00.000 1.000 12.66
32 32 33 1 0.3410 0.5302 0.00000 00.000 1.000 12.66
33 08 21 1 2.0000 2.0000 0.00000 00.000 1.000 12.66
34 09 15 1 2.0000 2.0000 0.00000 00.000 1.000 12.66
35 12 22 1 2.0000 2.0000 0.00000 00.000 1.000 12.66
36 18 33 1 0.5000 0.5000 0.00000 00.000 1.000 12.66
37 25 29 1 0.5000 0.5000 0.00000 00.000 1.000 12.66

5
9 33 18 19 20 06 07 03 04 05
4 34 12 13 14
6 35 08 11 10 09 21
8 36 32 17 16 15 29 30 31
8 37 24 23 22 28 25 26 27

MVMOS_RRPLM.m

```
%**************************************************************************
**
%                            MVMOS-based RRPLM
%**************************************************************************
**
%By: Dr. José L. Rueda
%Date: 17.02.2012

close all
clear all
clc

%
========================================================================
%                        Setting global variables and Flags
%
========================================================================
global parameter
global printff sn
global SWSTI SWST
global NLOOPS NLEL

data;
YBUS;
%G_modify;  %MODIFICADO Rosanna Loor

[err,iter,no_soln] = branch_lf;
original_switched_out = SWSTI(find(not(SWST)));
[original_sys_p_loss,original_vd] = system_p_loss;
original_sys_p_loss = original_sys_p_loss*1000;
% original_vd




%
========================================================================
%                              Define MVO parameters
%
========================================================================

%-------MVMO---------------
% % printff= 4; % print  the results at every printff function evaluation
on the screen
% % parameter.MaxEval = 20; % max no. of function evaluations
% % parameter.mode =4; %Selection strategy for offspring creation
% % parameter.n_tosave = 4; %Population to be stored in the table
% % parameter.n_random = 10; %Dimensions/control variables selected for
mutation operation
% % parameter.n_random_end = 10; %Final dimension change
% % parameter.n_randomly_freq = 1000; %FE number after which the selected
variables are changed
% % parameter.fs_factor_start = 1; %Initial scaling factor
% % parameter.fs_factor_end = 1; %Final scaling factor
```

```
% % parameter.dddd = 1;  %Initial value of the shape factor
% % parameter.delta_dddd = 0.05; %Now, normally between 0 - 1
% % parameter.n_var = 40; % Total no. of control variables
% % parameter.ncon_var =  40; % No. of continuous variables
% % parameter.ndis_var = parameter.n_var-parameter.ncon_var; % no. of
discrete variables
% % parameter.n_constr = 0; % no. of constraints


printff= 25; % print  the results at every printff FEs on the screen
parameter.MaxEval = 250; %15*201
sn = true; % logic control for screen printing
parameter.n_par = 2;
parameter.mode = 4;
parameter.n_tosave = 5;
parameter.n_random =NLOOPS-0;
parameter.n_random_min = NLOOPS-0;
parameter.n_randomly_freq = 2000; round(parameter.MaxEval/2);
parameter.fs_factor_start = 1;
parameter.fs_factor_end = 2;
parameter.pnlty_factor_start = 1e5;%100;   % penalización ini
parameter.pnlty_factor_end = 1e5;%50000;   % penalización final
parameter.dddd = 1;
parameter.delta_dddd_start=0.35;
parameter.delta_dddd_end=0.35;
parameter.n_var = NLOOPS;
parameter.Norm_limit = 0;  %80;rloor Maximum euclidean distance MVMOSwarm
parameter.Indep_run = 4;%000;


% Min and max limits of control variables
parameter.x_min = ones(1,NLOOPS)
parameter.x_max = NLEL'
parameter.scaling = (parameter.x_max-parameter.x_min)
%
================================================================================




%================================================================================
==
%                       Performing MVMO-based optimization
%================================================================================
==
runs=200;


of_serie=zeros(parameter.MaxEval,runs);
best_final=zeros(runs,parameter.n_var);
losses=zeros(runs,1);
switched_out=zeros(parameter.n_var,runs);
time_meas=zeros(runs,1);

for iopt=1:runs
    time=cputime;
    [ofcn,best] = MVMOS_new3m('test_func_RRPLM',iopt);
    time=cputime-time;
```

```
    best_sol = best(end,:);
    switching = ones(1,sum(NLEL));
    ind = 0;
    for j =1:NLOOPS
        temp_1 = NLEL(j);
        temp_2 = best_sol(j);
        switching(ind+temp_2) = 0;
        ind = ind + temp_1;
    end
    SWST= switching';
    final_switched_out = SWSTI(find(not(switching)));
    [~, ~, no_soln] = branch_lf;
    [final_sys_p_loss,final_vd] = system_p_loss;
    final_sys_p_loss = final_sys_p_loss*1000;
%     final_vd


    of_serie(:,iopt) = ofcn;
    best_final(iopt,:) = best_sol;
    losses(iopt,:) = final_sys_p_loss;
    switched_out(:,iopt) = final_switched_out;
    time_meas(iopt,:) = time;

end


%=========================================================================
==


% switching = ones(1,sum(NLEL));
% ind = 0;
% for j =1:NLOOPS
%     temp_1 = NLEL(j);
%     temp_2 = best_sol(j);
%     switching(ind+temp_2) = 0;
%     ind = ind + temp_1;
% end
% SWST= switching';
% final_switched_out = SWSTI(find(not(switching)));
% [err, iter, no_soln] = branch_lf;
% [final_sys_p_loss,final_vd] = system_p_loss;
% final_sys_p_loss = final_sys_p_loss*1000;
% final_vd

% save ecucol_ident_mvmos1t_1Pa ecucol_ident_mvmos1t_1P

fbest_stats=[min(of_serie(end,:)),max(of_serie(end,:)),me-
dian(of_serie(end,:)),mean(of_serie(end,:)),std(of_serie(end,:))];

plot(of_serie);
xlabel('No. of objective function evaluations');
ylabel('Objective function');
```

test_func_RRPLM.m

```
function [f,g,xn_out] = test_func_RRPLM(xx_yy)

global parameter
global NLOOPS NLEL SWST


% ----------------  Denormalize to the original range ------------------
--
 xx_yy1 = parameter.x_min+parameter.scaling.* xx_yy;
 xx_yy2 = round(xx_yy1);
%-------------------------------------------------------------------------
---



switching = ones(1,sum(NLEL));
ind = 0;
for j =1:NLOOPS
    temp_1 = NLEL(j);
    temp_2 = xx_yy2(j);
    switching(ind+temp_2) = 0;
    ind = ind + temp_1;
end

SWST= switching';
[~, ~, no_soln] = branch_lf; % ver
if no_soln == 1
    f = 1e100;
else
    f = ob_calc;
end


g=0;


% ------------------------  Normalize again ----------------------------
--
xn_out = (xx_yy2-parameter.x_min)./parameter.scaling;
% -------------------------------------------------------------------------
--

end
```

data.m

```
% Function data, which takes the input data and creates corresponding
system
```

```
% vectors .
% authored by dr. b. venkatesh,

% clear all;

global NB NBB NS NG NLB NTR NTRL NT NSHC NSVS NSHR NSH NREG
global VSLACK TOLER PBASE VLMAX VLMIN ITMAX
global BIND BSN BNAM PG QG PD QD V DEL QGMAX QGMIN VSH
global FB TB NCKT YL ZL BL TAP RAT KV LEN TAPMAX TAPMIN TAPSTP
global SNO SUS SUSMAX SUSMIN SUSSTP
global YB
global LAM
global PGMAX PGMIN VGMAX VGMIN ap bp cp
global ofp ifp % output and input file pointers
global NLOOPS NLEL SWSTI SWST FBTEMP TBTEMP

ofp = fopen('37out.txt', 'w');
ifp = fopen('37i.txt ', 'r');

fprintf(ofp,'INPUT FILENAME: 37i.txt \n');
fprintf(ofp,'OUTPUT FILE NAME: 37out.txt \n');

temp= fscanf(ifp,'%s', [1]);
temp= fscanf(ifp,'%s', [1]);
temp= fscanf(ifp,'%s', [1]);
temp= fscanf(ifp,'%s', [1]);

fprintf(ofp,'SYSTEM %s\n: %s\n', temp);
fprintf(ofp,'YEAR %s\n  : %s\n', temp);
fprintf(ofp,'CASE %s\n  : %s\n', temp);
fprintf(ofp,'NUMBER %s\n: %s\n', temp);

ttt = fscanf(ifp,'%d %d %d %d %d %d %d %d %d %d %d %d %d', [1,13]);

NB = ttt(1);
NBB = ttt(2);
NS = ttt(3);
NG = ttt(4);
temp = ttt(5);
temp = ttt(6);
NLB= ttt(7);
NTR = ttt(8);
NTRL = ttt(9);
NSHC = ttt(10);
NSVS = ttt(11);
NSHR = ttt(12);
NREG = ttt(13);

NT = NTR + NTRL;
NSH = NSHC + NSVS + NSHR;
ttt = fscanf(ifp,'%f %f %f %f %f %d',[1,6]);
VSLACK = ttt(1);
TOLER = ttt(2);
PBASE = ttt(3);
VLMAX = ttt(4);
VLMIN = ttt(5);
ITMAX = ttt(6);
```

```
fprintf(ofp,'NUMBER OF BUSES : %d\n', NB);
fprintf(ofp,'SLACK BUS NUMBER : %d\n', NS);
fprintf(ofp,'NUMBER OF GENERATORS : %d\n', NG);
fprintf(ofp,'NUMBER OF LOAD BUSES : %d\n', NLB);
fprintf(ofp,'NUMBER OF TRANSFORMERS : %d\n', NTR);
fprintf(ofp,'NUMBER OF TRANSMISSION LINES : %d\n', NTRL);
fprintf(ofp,'NUMBER OF SHUNT CAPACITORS : %d\n', NSHC);
fprintf(ofp, 'NUMBER OF SWITCHABLE CAPACITORS :%d\n',NSVS);
fprintf(ofp,'NUMBER OF SHUNT REACTORS : %d\n', NSHR);
fprintf(ofp,'SLACK BUS VOLATGE : %8.4f\n', VSLACK);
fprintf(ofp, 'TOLERANCE (MW) : %8.4f\n', TOLER*PBASE); %convierte a MW
los valores de potencia en kW
fprintf(ofp,'BASE MVA \t60 : %8.4f\n', PBASE);
fprintf(ofp,'MINIMUM LOAD BUS VOLTAGE : %8.4f\n', VLMIN);
fprintf(ofp,'MAXIMUM LOAD BUS VOLTAGE : %8.4f\n', VLMAX);
fprintf(ofp,'MAXIMUM NUMBER OF ITERATIONS : %d\n', ITMAX);

ttt = fscanf(ifp,'%s',[1]); % region

BSN = zeros(NB,1);
BNAM = cell(NB);
PD = zeros(NB,1);
QD= zeros(NB,1);
PG = zeros(NB,1);
QG= zeros(NB,1);
QGMAX = zeros(NB,1);
QGMIN = zeros(NB,1);
VSH = zeros(NB,1);

V = ones(NB,1);
DEL = zeros(NB,1);
fprintf(ofp, '   \n');
fprintf(ofp, '   \n');
fprintf(ofp,'*****DETAILED OUTPUT IN PER UNIT***** \n');
fprintf(ofp, '\n');
fprintf(ofp, '\n');
fprintf(ofp, '\t\t  GENERATOR BUS DATA\n') ;
fprintf(ofp, 'SNO R.NO B.NO BUS N A M E ---PG--------PD--- ---- QD--- —
QGMX--— QGMN— — V SH— \n');
BSNMAX=0;


for k = 1:NG
    ttt = fscanf(ifp, '%d %d %d',[1,3]);
    BSN(k) = ttt(3);
    ttt = fscanf(ifp, '%s ',[1]);
    BNAM(k)= cellstr(ttt);
    ttt = fscanf(ifp,'%f %f %f %f %f %f',[1,6]);
    PD(k) = ttt(1);
    QD(k) = ttt(2);
    PG(k) = ttt(3);
    QG(k) = 0;
    QGMAX(k) = ttt(4);
    QGMIN(k) = ttt(5);
    VSH(k) = ttt(6);
    V(k) = VSH(k);

    if (BSNMAX < BSN(k))
         BSNMAX = BSN(k);
```

```
    end

fprintf(ofp,'%3d %4d %4d %8s %8.4f %8.4f %8.4f %8.4f %8.4f
%8.4f\n',k,k,BSN(k), char(BNAM(k)),PG(k),PD(k),QD(k),QGMAX(k),QGMIN(k),
VSH(k));

PD(k) = PD(k)/PBASE;
QD(k) = QD(k)/PBASE;
PG(k) = PG(k)/PBASE;
QG(k) = QG(k)/PBASE;
QGMAX(k) = QGMAX(k)/PBASE;
QGMIN(k) = QGMIN(k)/PBASE;

end

fprintf(ofp, '\n');

fprintf(ofp,'\t\t LOAD BUS DATA\n');
fprintf(ofp,'BUS# BUS NAME ---PD--- ---- QD---\n');

for k = NG+1:NB
    ttt = fscanf(ifp, '%d %d %d',[1,3]);
    BSN(k)=ttt(3);
    temp_a = ttt(1);
    temp_b = ttt(2);
    ttt = fscanf(ifp,'%s',[1]);
    BNAM(k)=cellstr(ttt);
    ttt = fscanf(ifp, '%f %f', [1,2]);
    PD(k)=ttt(1);
    QD(k)=ttt(2);

if (BSNMAX < BSN(k))
    BSNMAX = BSN(k);
end

fprintf(ofp,'%4d %8s %8.4f %8.4f\n', BSN(k), char(BNAM(k)),PD(k),QD(k));
PD(k) = PD(k)/PBASE;
QD(k) = QD(k)/PBASE;
end

BIND = zeros(BSNMAX,1);

for k = 1:NB

    BIND(BSN(k)) = k;

end

ZL = (0+j*0) * zeros(NT,1);
YL = (0+j*0) * zeros(NT,1);
BL = (0+j*0) * zeros(NT,1);

FB = zeros(NT,1);
TB = zeros(NT,1);
NCKT=zeros(NT,1);
TAP= ones(NT,1);
RAT= zeros(NT,1);
```

```
KV = zeros(NT,1);
LEN= zeros(NT,1);
TAPMAX= ones(NT,1);
TAPMIN= ones(NT,1);
TAPSTP= ones(NT,1);
fprintf(ofp, '\n');


fprintf(ofp,'\t\t    TRANSFORMER DATA FOR TOTAL NOS OF CIRCUITS\n');
fprintf(ofp,'SNO F.NO T.NO NCKT — R PU— — X PU— ---AN---- RAT-\n') ;

for k=1:NTR
    ttt = fscanf(ifp, '%d %d %d %d %f %f %f % f', [1,8]);
    FB(k)=ttt(2);
    TB(k)=ttt(3);
    NCKT(k)= ttt(4);
    ZL(k) = complex(ttt(5),ttt(6));
    TAP(k) = ttt(7);
    BL(k) = complex(0,0);
    RAT(k) = ttt(8)/PBASE;
    LEN(k) = 0;
    KV(k) = 0;

    BL(k)= BL(k) * NCKT(k);    %X by NCKT
    ZL(k)= ZL(k) / NCKT(k);    %/ NCKT
    YL(k) = 1/ZL(k);
    ttt = fscanf(ifp,'%f %f %f',[1,3]);
    TAPMAX(k) = ttt(1);
    TAPMIN(k) = ttt(2);
    TAPSTP(k) = ttt(3);

fprintf(ofp,'%3d %4d %4d %4d %8.4f %8.4f %8.4f %8.4f \n', k,
FB(k),TB(k),NCKT(k),real(ZL(k)),imag(ZL(k)),TAP(k),RAT(k));

end
fprintf(ofp,' \n');
fprintf(ofp,'\t\t   LINE DATA FOR TOTAL NOS OF CIRCUITS\n');
fprintf(ofp,'SNO F.NO T.NO NCKT — R PU— — X PU— — HLC---- RAT—AN--- \n');

for k=NTR+1:NT
    ttt = fscanf(ifp, '%d %d %d %d %f %f %f %f %f %f ', [1,10]);
    FB(k) =ttt(2);
    TB(k) =ttt(3);
    NCKT(k)= ttt(4);
    ZL(k) = complex(ttt(5),ttt(6));
    BL(k) = complex(0,ttt(7));
    RAT(k) = ttt(8)/PBASE;
    LEN(k) = ttt(9);
    KV(k) = ttt(10);

    BL(k) = BL(k)* NCKT(k);  % X by NCKT
    ZL(k)=  ZL(k)/NCKT(k);   % / NCKT


if LEN(k)>0 && KV(k)>0          %Line length & KV
                                 %KV*KV/PBASE
    ZBASE = KV(k) * KV(k) / PBASE;
    ZL(k) = ZL(k) * LEN(k) / ZBASE;
```

```
    BL(k) = BL(k) * LEN(k) * ZBASE;
end

YL(k) = 1/ZL(k);
fprintf(ofp,'%3d %4d %4d %4d %8.4f %8.4f %8.4f %8.4f % 8.4f\n', k,FB(k),
TB(k), NCKT(k),real(ZL(k)),imag(ZL(k)), BL(k),RAT(k), TAP(k));
end

%%%%
SNO = zeros(NSH,1);
SUS = zeros(NSH,1);
SUSMAX = zeros(NSH,1);
SUSMIN = zeros(NSH,1);
SUSSTP = zeros(NSH,1);
fprintf(ofp,'  \n') ;
fprintf(ofp,'\t\tSHUNT CAPACITOR DATA\n') ;
fprintf(ofp,'SNO B.NO -MVAR-pu-\n');

for k = 1:NSHC % Fixed capacitors
    ttt = fscanf(ifp,'%d %d %f',[1,3]);
    SNO(k) = ttt(2);
    SUS(k) = ttt(3) / PBASE;
    fprintf(ofp,'%3d %4d % 8 .4f\n',k,SNO(k), SUS(k)) ;
end

fprintf(ofp, '\n') ;
fprintf(ofp, '\t\tSWITCHABLE CAPACITOR DATA\n');
fprintf(ofp, 'SNO B.NO — MAX------ MIN---- STEP----- ACTUAL-\n');
fprintf(ofp, ' --MVAR— — MVAR— --MVAR-- --MVAR— \n');

for k = NSHC+1 : NSHC+NSVS
    ttt = fscanf(ifp,'%d %d %f %f %f %f',[1,6]);
    SNO(k) = ttt(2);
    SUSMAX(k) = ttt(3) / PBASE;
    SUSMIN(k) = ttt(4) / PBASE;
    SUSSTP(k) = ttt(5) / PBASE;
    SUS(k) = ttt(6) / PBASE;
    fprintf(ofp,'%3d %4d %8.4f %8.4f %8.4f % 8
.4f\n',k,SNO(k),SUSMAX(k),SUSMIN(k),SUSSTP(k),SUS(k));
end

fprintf(ofp, '  \n') ;
fprintf(ofp, '\t\tSHUNT REACTOR DATA\n') ;
fprintf(ofp,'SNO B.NO — MVAR— \n');

for k = NSHC+NSVS+1:NSH
    ttt = fscanf(ifp,'%d %d %f',[1,3]);
    SNO(k) = ttt(2);
    SUS(k) = -ttt(3) / PBASE;
    fprintf(ofp, '%3d %4d % 8 .4f\n',k,SNO(k), SUS(k));
end

PGMAX = zeros(NG,1);
PGMIN = zeros(NG,1);
ap = zeros(NG,1);
bp = zeros(NG,1);
cp = zeros(NG,1);
%loops and loop data
```

```
fprintf(ofp, '  \n') ;
fprintf(ofp, '\t\t LOOP DATA\n');
ttt = fscanf(ifp, '%d', [1,1]);
NLOOPS = ttt;
fprintf(ofp,'NUMBER OF LOOPS : %d\n', NLOOPS);
fprintf(ofp,'NO. SWITCHES SWITCH PLACE (LINE NO.) \n');

for k = 1:NLOOPS
    ttt = fscanf(ifp, '%d', [1,1]);
    NLEL = [NLEL;ttt];
    fprintf(ofp,'%5d\t\t\t',ttt);
    ttt = fscanf(ifp, '%d', [1,NLEL(k)]) ;
    fprintf(ofp,'%3d',ttt);
    fprintf(ofp, '\n');
    SWSTI=[SWSTI;ttt'];

end

SWST = ones(length(SWSTI),1);
SWST(1) = 0;

for k = 1:length(NLEL)-1
SWST(sum(NLEL(1:k))+1) = 0;
end
PD = PD./1000;
QD = QD./1000;
```

branch_array.m

```
function [output] = branch_array(current_bus,parent_bus)

% Authored by Alex Rost.
% This function uses the radial system configuration to build a vector
% that contains the following information for a branch segment of the
% system:
% 1. parent bus of the branch
% 2. number of buses in the branch
% 3. number of branches emanating from the end of the branch
% 4. the list of the buses in the branch
% 5. the list of the branch buses

global FBTEMP TBTEMP

stop_flag = 0;
num_buses = 0;
num_branches = 0;
bus_array = [];
branch_buses = [];

while stop_flag == 0
      num_buses = num_buses + 1;
      bus_array = [bus_array  current_bus];
      FB_ind = find(FBTEMP == current_bus);
      TB_ind = find(TBTEMP == current_bus);
      if length(FB_ind) + length(TB_ind) == 0
          end_flag = 1;
```

```
            stop_flag = 1;
        elseif length(FB_ind) + length(TB_ind) == 1
            if length (FB_ind) == 1
               current_bus = TBTEMP(FB_ind);
               TBTEMP(FB_ind) = 0;
               FBTEMP(FB_ind) = 0;
            else
               current_bus = FBTEMP(TB_ind);
               TBTEMP(TB_ind) = 0;
               FBTEMP(TB_ind) = 0;
            end
            FBTEMP = FBTEMP(find(FBTEMP));
            TBTEMP = TBTEMP(find(TBTEMP));
        else
           if length(FB_ind) == 0
              branch_buses = FBTEMP(TB_ind)';
              TBTEMP(TB_ind) = 0;
              FBTEMP(TB_ind) = 0;

              elseif length(TB_ind) == 0
                      branch_buses = TBTEMP(FB_ind)';
                      TBTEMP(FB_ind) = 0;
                      FBTEMP(FB_ind) = 0;
              else
                  branch_buses = [FBTEMP(TB_ind)'  TBTEMP(FB_ind)'];
                  TBTEMP(TB_ind) = 0;
                  FBTEMP(TB_ind) = 0;
                  TBTEMP(FB_ind) = 0;
                  FBTEMP(FB_ind) = 0;
           end
           num_branches = length(branch_buses);
           FBTEMP = FBTEMP(find(FBTEMP));
           TBTEMP = TBTEMP(find(TBTEMP));
           stop_flag = 1;
    end
end
output = [parent_bus  num_buses  num_branches  bus_array  branch_buses];
```

branch_lf.m

```
function [err,iter,no_soln] = branch_lf

% this is a modified version of the RDS LF program in
% appendix D of my MScE thesis.
% authored by alex rost

global NB
global PG QG PD QD V DEL QGMAX QGMIN VSH
global FB TB ZL

FB_TB_shorten;
[output] = system_array(1,0);
output = fliplr(output);

iter = 0;
```

```
max_iter = 1000;
min_err = 0.001;
err = 100;

while err > min_err && iter < max_iter
      VA_acc = zeros(2,NB);
      q_generator = zeros(NB,1);
      iter = iter + 1;
      for br = 1:size(output,2)
          this_branch = output(:,br);
          parent_bus = this_branch(1);
          num_buses = this_branch(2);
          num_branches = this_branch(3);
          branch_segment = [parent_bus this_branch(4:3+num_buses)'];
          branch_segment = fliplr(branch_segment);

          for i = 1:num_buses
              send_bus = branch_segment(i+1);
              if send_bus > 0
              receive_bus = branch_segment(i);
              v_send = V(send_bus);
              delta_send = DEL(send_bus);
              temp_1 = FB - min([send_bus receive_bus]);
              temp_2 = TB - max([send_bus receive_bus]);
              r_x_ind = find(or(temp_1,temp_2)==0);
              r = real(ZL(r_x_ind));
              x = imag(ZL(r_x_ind));
              p_inj = VA_acc(1,receive_bus) + PD(receive_bus) - PG(re-
ceive_bus);
              if PG(receive_bus) == 0
                 q_inj = VA_acc(2,receive_bus) + QD(receive_bus) - QG(re-
ceive_bus);
                 [v_receive] = v_calc(r,x,p_inj,q_inj,v_send);
              else
                  q_violation_flag = 0;
                  v_receive = VSH(receive_bus);
                  [q_inj] = q_calc(r,x,p_inj,v_send,v_receive);
                  q_gen = q_inj - QD(receive_bus) - VA_acc(2,re-
ceive_bus);

                   if q_gen > QGMAX(receive_bus)
                      q_gen = QGMAX(receive_bus);
                      q_violation_flag = 1;
                    elseif q_gen < QGMIN(receive_bus)
                        q_gen = QGMIN(receive_bus);
                        q_violation_flag = 1;
                   end
                   q_generator(receive_bus) = q_gen;
                   if q_violation_flag == 1
                       q_inj = q_gen + QD(receive_bus) + VA_acc(2,re-
ceive_bus);
                       [v_receive] = v_calc(r,x,p_inj,q_inj,v_send);
                   end
              end
              [delta_receive,p_loss,q_loss] =
delta_ploss_qloss(r,x,p_inj,q_inj,v_send,v_receive,delta_send);
              V(receive_bus) = v_receive;
              DEL(receive_bus) = delta_receive;
              VA_acc(1,send_bus) = VA_acc(1,send_bus) + p_inj + p_loss;
```

```
                VA_acc(2,send_bus) = VA_acc(2,send_bus) + q_inj + q_loss;
                end
            end
        end
        [err] = s_balance_YB(q_generator);
end
QG = -q_generator ;
no_soln = 0;
if iter == max_iter
    no_soln = 1;
end
```

delta_ploss_qloss.m

```
function [delta_j,p_loss,q_loss] =
delta_ploss_qloss(r_ij,x_ij,p_ij,q_ij,v_i,v_j,delta_i)

% This function calculates the receiving end bus angle and the real and
% reactive power line losses between the sending end and receiving end
% buses. Parameters needed for this function are the line resistance,
% reactance, receiving end real and reactive powers, receiving and
% sending end bus voltages and the sending end bus angle.
% authored by alex rost.

delta_j = delta_i-asin((p_ij*x_ij-q_ij*r_ij)/(v_i*v_j));
p_loss = r_ij*((p_ij^2+q_ij^2)/v_j^2);
q_loss = x_ij *((p_ij^2+q_ij^2)/v_j^2);
```

FB_TB_shorten.m

```
global FB TB
global SWSTI SWST FBTEMP TBTEMP

% This function creates two vectors, FBTEMP and TBTEMP, by cutting out
the
% entries of FB and TB for lines that are switched out. This is used for
% constructing the system matrix of the current system configuration.
% authored by alex rost

temp1 = SWSTI(find(SWST == 0));
temp2 = FB;
temp3 = TB;
temp2(temp1) = 0;
temp3(temp1) = 0;

FBTEMP = temp2(find(temp2));
TBTEMP = temp3(find(temp2));
```

ob_calc.m

```
function [ob_val] = ob_calc
```

```
% authored by alex rost

global PG PD V DEL
global YL

p_d = sum(PD);
p_g = sum(PG);

v_sub = V(1)*(cos(DEL(1)) + j*sin(DEL(1)));
v_2 = V(2)*(cos(DEL(2)) + j*sin(DEL(2)));
y = YL(1);
k = 10;

vdev=abs(V-1);
vdev(vdev<=0.07)=0;
vd = sum(vdev);
% vd=sqrt(sum(V-1)^2/length(V));

ob_val = real(v_sub*conj((v_sub - v_2)*y)) - p_d + p_g + k*vd;

%vdev=abs(V-0.95); %V=PERFIL DE VOLTAJE
%vdev(vdev<1.0d-7)=0;  % RLOOR, SE CAMBIÓ EL CÁLCULO DE VD, %vd =
sum(vdev);
```

s_balance_YB.m

```
function [err] = s_balance_YB(q_generator)

% This function calculates the system power balance and returns the
% greatest imbalance error. This function uses a modified version of the
% system YBUS that takes into account which lines are switched out
(SWMAT).
% authored by alex rost

global NB
global PG PD QD V DEL
global SWST

X = [SWST' V' DEL'];
[sw_mat] = SWMAT(X) ;
P_balance = zeros(1,NB);
Q_balance = zeros(1,NB);

for i = 2:NB
    temp_1 = sw_mat(i,:);
    temp_2 = find(temp_1);
    p = 0;
    q = 0;

for j = 1:length(temp_2)
    bus_num = temp_2(j);
    y_ij = abs(sw_mat(i,bus_num));
    theta_ij = angle(sw_mat(i,bus_num));
    p = p + V(i)*V(bus_num)*y_ij*cos(DEL(i) - DEL(bus_num) - theta_ij);
    q = q + V(i)*V(bus_num)*y_ij*sin(DEL(i) - DEL(bus_num) - theta_ij);
```

```
end

P_balance(i) = p;
Q_balance(i) = q;

end

P_err = P_balance + PD' - PG';
Q_err = Q_balance + QD' + q_generator';
err_vector = max(abs(P_err),abs(Q_err));
err = max(err_vector);
```

SWMAT.m

```
function [sw_mat] = SWMAT(X)

%Creates a version of ybus that depends on the state of the switches in
%the system. Penalty functions dependant on X can be used.
%authored by alex rost.

global NB
global FB TB YL
global YB
global NLOOPS NLEL SWSTI

exp_num = 10;
sw_mat = YB;
tempfrom = [];
tempto = [];
tempXind = [];
tempYLind = [];

%this first section finds which switch is open for each loop (the first
%part of the X vector) and then subtracts the corresponding admittance
%for the proper bus in the ybus matrix, this is for all off diagonal
%elements only,

for k = 1:NLOOPS,
    if k == 1,
        temploop = X(1:NLEL(1));
        tempind = find(temploop == min(temploop));
    else
        tempstart = sum(NLEL(1:k-1));
        temploop = X((tempstart+1):(tempstart+NLEL(k)));
        tempind = tempstart + find(temploop == min(temploop));
    end

    tempXind = [tempXind  tempind];
    tempind = SWSTI(tempind);
    tempYLind = [tempYLind  tempind];
    tempfrom = [tempfrom  FB(tempind)];
    tempto = [tempto  TB(tempind)];

end

for i = 1 : length(tempfrom)
```

```
    sw_mat(tempfrom(i),tempto(i)) = X(tempXind(i))^exp_num*YB(temp-
from(i),tempto(i)) ;
    sw_mat(tempto(i),tempfrom(i)) =
X(tempXind(i))^exp_num*YB(tempto(i),tempfrom(i ));
end

%this second uses the open switch information obtained above and modifies
%the diagonal elements of YBUS accordingly,

for i = 1:NB
    tempind = find(tempfrom == i);

     for j = 1:length(tempind)
         sw_mat(i,i) = sw_mat(i,i )-YL(tempYLind(tempind(j)));
     end

tempind = find(tempto == i);

for j = 1:length(tempind)
    sw_mat(i,i) = sw_mat(i,i)-YL(tempYLind(tempind(j)));
 end
end
```

system_array.m

```
function [output] = system_array(current,parent)

% This file uses recursion and the function branch_array to construct
% a matrix ordered such that each column has the following information
% for each branch:
% 1. parent bus of the branch
% 2. number of buses in the branch
% 3. number of branches emanating from the end of the branch
% 4 . the list of the buses in the branch
% 5. the list of the branch buses
% This matrix is ordered in such a way to facilitate the radial
% distribution system load flow.
% authored by alex rost.

[output_1] = branch_array(current,parent);
output = output_1';
num_branches = output_1(3);

if num_branches > 1
   num_buses = output_1(2);
   branch_buses = output_1(4+num_buses:length (output_1));
   branch_buses = sort(branch_buses);
   parent = output_1(length(output_1)-num_branches);

   for i = 1 :num_branches
   current = branch_buses(i);
   [output_2] = system_array(current,parent);
   length_1 = size(output,1);
   length_2 = size(output_2,1);
      if length_1 > length_2
          temp_1 = eye(length_2) ;
```

```
          temp_2 = zeros(length_1 - length_2,length_2);
          temp = [temp_1; temp_2];
          output_2 = temp*output_2;
               elseif length_2 > length_1
                    temp_1 = eye (length_1) ;
                    temp_2 = zeros(length_2-length_1, length_1) ;
                    temp = [temp_1; temp_2];
                    output = temp*output;
      end

       output = [output output_2];

    end
end
```

system_p_loss.m

```
function [p_loss,vd] = system_p_loss
% authored by alex rost

global PG PD V DEL
global YL

p_d = sum(PD);
p_g = sum(PG);

v_sub = V(1)*(cos(DEL(1)) + j*sin(DEL(1)));
v_2 = V(2)*(cos(DEL(2)) + j*sin(DEL(2)));
y = YL(1);

p_loss = real(v_sub*conj((v_sub - v_2)*y)) - p_d + p_g;

vdev=abs(V-0.95); %V=PERFIL DE VOLTAJE
vdev(vdev<1.0d-7)=0; % RLOOR, SE CAMBIÓ EL CÁLCULO DE VD,
vd = sum(vdev);

%-----
% vd=sqrt(sum(V-1)^2/length(V));
% vd = sqrt(sum(V-1)^2/length(V));
```

v_calc.m

```
function [v_j] = v_calc(r_ij,x_ij,p_ij,q_ij,v_i)

% This function calculates the receiving end bus voltage.
% Parameters needed for this function are the line resistance and
% reactance, receiving end real and reactive powers and the and the
% sending end bus voltage.
% authored by alex rost

a = r_ij*p_ij+x_ij*q_ij-v_i^2/2;
b = sqrt(a^2-(r_ij^2+x_ij^2)* (p_ij^2+q_ij^2)) ;
v_j = sqrt(-a+b);
```

YBUS.m

```
% function YBUS. This function calculates the YBUS of the total radial
% distribution system.
% authored by dr. b. venkatesh.

global NB NT NSH
global BIND
global FB TB YL BL TAP
global SNO SUS
global YB

YB = (0+j*0) * zeros(NB,NB); %% Creating an empty 6x6 matrix

for k=1:NT
    k1 = BIND(FB(k));
    k2 = BIND(TB(k));

    YB(k1,k2) = YB(k1,k2)- YL(k)/TAP(k);  % Off Diagonal Elements
    YB(k2,k1) = YB(k2,k1)- YL(k)/TAP(k);
    YB(k1,k1) = YB(k1,k1) + BL(k) + YL(k)/(TAP(k)^2); % Diagonal Elements
    YB(k2,k2) = YB(k2,k2) + BL(k) + YL(k);
end

for k = 1:NSH
    k1 = BIND(SNO(k));
    YB(k1,k1) = YB(k1,k1) + SUS(k);
end
```

MVMOS_new3m.m

```
%By Dr. Jose L. Rueda (jose.rueda@uni-duisburg-essen.de)
    % Standard Mean-Variance Mapping Optimization (MVMO) - Version 2012 ©
    %              ------------------*------------------
    % MVMO is a novel optimization algorithm. The basic concept of MVMO
shares
    % certain similarities to other heuristic approaches. However, the
novel
    % contribution in MVMO is the special mapping function. The shape and
    % location of the underlying mapping curve are adjusted according to
the
    % searching process. The output of this mapping function is always
inside
    % [0,1]. This means that MVMO can guarantee no violation of the vari-
ables'
    % limits during the searching process. Moreover, MVMO updates the
candidate
    % solution around the best solution in every iteration. Thanks to the
    % well-designed % balance between search diversification and intensi-
fication,
```

```
    % MVMO can find the optimum quickly with minimum risk to premature
convergence.
    % This version of MVMO is implemented according to [1], but including
the
    % modifications given in [2], which improve the zero-variance han-
dling and
    % robustness of the algorithm. The static penalty scheme is used to
handle
    % constraints. Nevertheless,other constraint handling techniques are
also
    % applicable to MVMO.
    % Key references:
    %[1] I. Erlich, G. K. Venayagamoorthy, and W. Nakawiro, "A mean-vari-
ance
    % optimization algorithm," in Proc. 2010 IEEE World Congress on Com-
putational
    % Intelligence, Barcelona, Spain.
    %[2] I. Erlich, F. Shewarega, C. Feltes, F. Koch and J. Fortmann,
"Determination
    % of Dynamic Wind Farm Equivalents using Heuristic Optimization,"
2012 IEEE
    % Power & Energy Society General Meeting, San Diego, USA.
    %              --------------------*--------------------
    %============================== Nomenclature
==============================
    % max_eval :=  Maximum number of objective function evaluations
    % n_var := Number of parameters to be optimized, i.e. the dimension-
ality of the problem.
    % n_par := Number of particles
    % mode := Variable selection strategy for offspring creation
    % dddd := Initial shape factor
    % delta_dddd := Shape factor increment
    % n_randomly := Initial number of variables selected for mutation
    % n_randomly_min := Minimum number of variables selected for mutation
    % n_randomly_freq := Function evaluation no. after which the number
of selected variables is adjusted
    % n_to_save :=  n-best individuals to be stored in the table
    % fs_factor_start := Initial shaping scaling factor
    % fs_factor_end := Final shaping scaling factor
    %================================= Syntax
=================================
    %  [ofcn, feas, best] = MVMO(fitfcn,x_normalized,varargin)
    %  Starts at x_normalized and finds a minimum x to the objective
function
    %  fitfcn, subject to inequality constraints represented by g(x)<=0.
fitfcn
    %  is a function handle, which accepts input x_normalized and returns
scalar
    %  values of the objective function and the constraints evaluated at
x.
    %  ofcn and feas are the values of the objective function fitfcn at
the best-so-far
    %  solution 'best'and its feasibility. Possible values of feas are
listed below.
    %  - 0 constraint violation
    %  - 1 optimality conditions satisfied (i.e. no constraint violation)

%=========================================================================
==
```

```
    global parameter;
    global x_normalized_best
    global considered changed fs_factor
    global printff sn
    global shape
    global meann
    global table
    global variance
    global l_vari
    global izm izz

    xx=zeros(parameter.n_par,parameter.n_var);
    x_norm=xx;
    %Start initialization: x_normalized
    for iijj=1:parameter.n_par
        for jjkk=1:parameter.n_var
            xx(iijj,jjkk)=parameter.x_min (jjkk) + rand*(parame-
ter.x_max(jjkk)-parameter.x_min(jjkk));
        end
        x_norm(iijj,:)=(xx(iijj,:)-parameter.x_min)./parameter.scaling;
    end% End initialization
    x_normalized=x_norm;


    %% -------------------- Intialize control parameters ---------------
------
    n_var = parameter.n_var;
    max_eval = parameter.MaxEval;
    n_par = parameter.n_par;
    mode = parameter.mode;
    dddd = ones(n_par,n_var)*parameter.dddd;
    delta_dddd_start = parameter.delta_dddd_start;
    delta_dddd_end = parameter.delta_dddd_end;
    n_randomly = parameter.n_random;
    n_randomly_min = parameter.n_random_min;
    n_randomly_freq = parameter.n_randomly_freq;
    n_to_save = parameter.n_tosave;
    fs_factor_start = parameter.fs_factor_start;
    fs_factor_end = parameter.fs_factor_end;
    pnlty_factor_start  = parameter.pnlty_factor_start;
    pnlty_factor_end = parameter.pnlty_factor_end;

    %% --------------------- Data structure for the table ---------------
------
    table.bests = zeros(parameter.n_tosave,parameter.n_var,n_par);
    table.objective = 1e10*ones(parameter.n_tosave,1,n_par);
    table.fitness = 1e10*ones(parameter.n_tosave,1,n_par);
    table.feasibility = zeros(parameter.n_tosave,1,n_par);
  %  table.indpendent_run_p=zeros(n_par);

    %% ---------------------------- Outputs ----------------------------
------
    ofcn = zeros(max_eval,1);
    fitn = zeros(max_eval,1);
    feas = zeros(max_eval,1);
    best = zeros(max_eval,n_var);
    best_norm = zeros(max_eval,n_var);
```

```
    %% ---------------------------- Mapping ----------------------------
------
    shape = zeros(n_par,n_var);
    x_normalized_best = x_normalized;
    meann = x_normalized;

%     solution_feasible=zeros(1,n_par);
    iiem=zeros(1,n_par);
    pnlty_factor0=ones(1,n_par);

    %% ------------------------ Variable selection ----------------------
------
    izm = zeros(1,n_par);
    izz = zeros(1,n_par);
    considered = true(n_par,n_var);
    variance =ones(n_par,n_var);
%     vsel = repmat(1:n_randomly,n_par,1); vv =repmat(1:n_ran-
domly,n_par,1);

    if (n_randomly<=0)
        n_randomly=1;
    end

    if (n_randomly>n_var)
        n_randomly=n_var;
    end

    if (n_randomly_min<1)
        n_randomly_min=1;
    end

    if (n_randomly_min>n_var)
        n_randomly_min=n_var;
    end

    if (mode<1)
        mode=4;
    end

    if (mode>4)
        mode=4;
    end

    if (max_eval<=0)
        max_eval=100000;
    end

    if ((n_randomly_freq<=0) || (n_randomly_freq>max_eval))
        n_randomly_freq=max_eval/20;
    end

    if (fs_factor_start<=0)
        fs_factor_start=1;
    end

    if (fs_factor_end<=0)
        fs_factor_end=1;
    end
```

```
    fs_factor0=fs_factor_start;
    pnlty_factor0=pnlty_factor_start*pnlty_factor0;

    if (n_to_save<=1)
        n_to_save=2;
    end

%     if (n_to_save>n_var)
%         n_to_save=n_var;
%     end

    if (delta_dddd_start<=0)
        delta_dddd_start=0.2;
    end

    if (delta_dddd_end<=0)
        delta_dddd_end=0.2;
    end

    delta_dddd0=delta_dddd_start;

    yes_fs_factor=true;
    if (fs_factor_start==fs_factor_end)
        yes_fs_factor=false;
    end

    yes_pnlty_factor=true;
    if (pnlty_factor_start==pnlty_factor_end)
        yes_pnlty_factor=false;
    end



    yes_delta_dddd=true;
    if (delta_dddd_start==delta_dddd_end)
        yes_delta_dddd=false;
    end

    yes_n_randomly=true;
    if (n_randomly==n_randomly_min)
        yes_n_randomly=false;
    end


    %% ---------------------------- Counters ---------------------------
------
    ip_first0=1;
    ip_last0=n_par;
    no_in = zeros(1,n_par);
    no_inin = zeros(1,n_par);
    indpendent_run_p=zeros(1,n_par);
 %   iiem=0;
    solution_feasible = false (1,n_par);

    l_vari=2    ;  %n_to_save/2;
    if (l_vari<2)
        l_vari=2;
```

```
    end

    Consider_particle = true(1,n_par);
    ip_Globalbest=0;
    na_pg=n_par;

    Norm_limit=(parameter.Norm_limit)*(parameter.Norm_limit)*real(n_var);
    Indep_run=parameter.Indep_run;


    kk=0;
    iie=1;
    fe_count1=0;
    ip_Globalbest_not_exist=1;

    %% --------------------------------- MVMO ----------------------------
------

fprintf('=================================================================
==============\n');
    fprintf('                    Mean-Variance Mapping Optimization Re-
sults                  \n');

fprintf('=================================================================
==============\n');
    fprintf('Nomenclature:
\n');
    fprintf('FE-No. => Objective function evaluation number
\n');
    fprintf('NAP    => Number of active particles
\n');
    fprintf('AGBP   => Actual global best particle
\n');
    fprintf('Fitn   => Global best fitness: Objective function plus pen-
alty due to constraint violation    \n');
    fprintf('Run    => Actual run
\n');
    fprintf('\n');
    fprintf('FE-No.      NAP        AGBP           Fitn             Run
\n');
    fprintf('------     ------   ---------    ---------------     -------
-  \n');

%    n_randomly0=n_randomly ;
    tol_flag=0;

    while (tol_flag==0)  %(iie<=max_eval)

        if yes_n_randomly
            kk=kk+1;
            if (kk>=n_randomly_freq)
                n_randomly=n_randomly-1;
                if (n_randomly < n_randomly_min)
                    n_randomly=n_randomly_min;
                    yes_n_randomly=false;
                end
                kk=1;
            end
```

```
        end

        if yes_fs_factor
            ff=real(iie)/real(max_eval);
            fs_factor0=fs_factor_start+ff*ff*(fs_factor_end-fs_fac-
tor_start);
        end



        if yes_delta_dddd
            ff1=real(iie)/real(max_eval);
            delta_dddd0=delta_dddd_start+ff1*ff1*(delta_dddd_end-
delta_dddd_start);
        end

        ip_fisrt=ip_first0;
        ip_last=ip_last0;
        ip_first0=0;
        ip_last0=0;
        for ipp=ip_fisrt:ip_last  %for n parrrrrr
            if ~Consider_particle(ipp)
                continue
            end

            if yes_pnlty_factor
                ff=real(iie-iiem(ipp))/real(max_eval-iiem(ipp));
                %  if xt.feasibility  solution_feasible=true; end
                if  ((solution_feasible(ipp)) || (iie > max_eval*2/5))
                    solution_feasible(ipp)=true;
                    pnlty_factor0(ipp)=pnlty_fac-
tor_start+ff*ff*(pnlty_factor_end-pnlty_factor_start);
                else
                    pnlty_factor0(ipp)=pnlty_factor_start;
                    iiem(ipp)=iie;
                end
            end


            if (ip_first0==0)
                ip_first0=ipp;
            end
            ip_last0=ipp;


%             tic;

            [ffx,ggx,x_normalized(ipp,:)] = feval(fitfcn, x_normal-
ized(ipp,:)); %%FUNCION OBJETIVO


            iie=iie+1;
            fe_count1=fe_count1+1;
            tol_flag=fe_count1>=max_eval;

%             ggx = 0;
```

```
            xt.objective = ffx;
            xt.fitness = static_penalty(ffx,ggx);

            if xt.fitness==xt.objective
                xt.feasibility= true;
                solution_feasible(ipp)=true;
            else
                xt.feasibility= false;
            end

            % Store the n-best solutions to the archive
            save_best_feasible();

            cc_idx=find(Consider_particle==1);
            [~,aa_yy]=min(table.fitness(1,:,cc_idx));
            ip_min=cc_idx(aa_yy);


            if ((fe_count1 == 1) || (mod(fe_count1,printff) == 0)) && sn
                if sn
                    fprintf('%5d    %5d     %5d     %17.7f       %5d
\n',...
                        fe_count1,na_pg,ip_min,table.fit-
ness(1,:,ip_min),iopt);
                end
            end
 %            n_randomly=n_randomly0;


             if (na_pg==1) % only one particle
                 ip_Globalbest=ipp;
                 x_normalized(ipp,:)=x_normalized_best(ipp,:);

             else %several particles
                 indpendent_run_p(ipp)=indpendent_run_p(ipp)+1;
                 if (indpendent_run_p(ipp)> Indep_run )     % not inde-
pendent run

                     if ip_Globalbest_not_exist
                         ip_Globalbest=1;
                         ip_Globalbest_not_exist=0;
                         for do_ipx=2:n_par
                             if ~Consider_particle(do_ipx)
                                 continue
                             end
                             if (table.fitness(1,:,do_ipx)<table.fit-
ness(1,:,ip_Globalbest))
                                 ip_Globalbest=do_ipx;
                             end
                         end
                     else
                         if(table.fitness(1,:,ipp)<table.fit-
ness(1,:,ip_Globalbest))
                             ip_Globalbest=ipp;
                         end
                     end
                 %    ww=table.fitness(1,:,ip_Globalbest)/table.fit-
ness(1,:,ipp);
```

```
                            E_Norm = 1.d200;
                            if (ipp~=ip_Globalbest)
                                E_Norm=0;
                                E_Norm1=table.bests(1,:,ip_Globalbest)-ta-
ble.bests(1,:,ipp);
                                E_Norm=E_Norm+sum(E_Norm1.^2);
                            end

                            if (E_Norm<Norm_limit)
                                Consider_particle(ipp)=false;
                                na_pg=na_pg-1;
%                                 if (na_pg==1)
%                                     na_pg_ende=fe_count1;
%                                 end

                                ofcn(fe_count1,:)=table.objec-
tive(1,:,ip_min);
                                fitn(fe_count1,:)=table.fitness(1,:,ip_min);
                                feas(fe_count1,:)=table.feasibil-
ity(1,:,ip_min); %not necessary (s.a.)
                                best(fe_count1,:)=parameter.x_min+(parame-
ter.x_max-parameter.x_min).*table.bests(1,:,ip_min);
                                best_norm(fe_count1,:)=ta-
ble.bests(1,:,ip_min);
                                continue
                            end

                            ww=0.0d0 ;
                            for jl=1:n_var

                                    x_normalized(ipp,jl) =  ww*x_normal-
ized_best(ipp,jl)+(1.d0-ww)*x_normalized_best(ip_Globalbest,jl) ;
                            end

                    else    % (indpendent_run_p(ipp)<= Indep_run) still in-
dependent run
                            ip_Globalbest=ipp;
                            x_normalized(ipp,:)=x_normalized_best(ipp,:);
                    end
                end

            % change the corresponding variable(s)
 %           if (n_randomly < n_var)
              considered(ipp,1:n_var) = false;
              VariableSelect1(); % Call random variable selection strategy
 %           else
 %              considered(ipp,1:n_var) = true;
 %           end

            for jl=1:n_var
                 if considered(ipp,jl)
                      x_normalized(ipp,jl) = rand;
                 end
            end
%               vrand = rand(1,n_var);
%               x_normalized(ipp,considered(ipp,:)) = vrand(consid-
ered(ipp,:));
```

```
          if (n_randomly<n_var)
      %         n_randomly
      %     end
             for ivar=1:n_var
    %             if ((shape(ipp,ivar)>1.d-50)) %considered(ipp,ivar)
                  if ((considered(ipp,ivar))&&(shape(ipp,ivar)>1.d-50))
%considered(ipp,ivar)
                      fs_factor=fs_factor0*(1.d0+0.9*rand);
                      sss1 = shape(ipp,ivar);
                      sss2 = sss1;
                      arand = rand;
                      delta_ddd_x=delta_dddd0*(arand-
0.5d0)*2d0+(1.d0+delta_dddd0);
                       isteur = round(rand);
                      if shape(ipp,ivar)>1.d-70
                          if (shape(ipp,ivar)>dddd(ipp,ivar))
                              dddd(ipp,ivar) = dddd(ipp,ivar)*delta_ddd_x;
                          else
                              dddd(ipp,ivar) = dddd(ipp,ivar)/delta_ddd_x;
                          end
                          if isteur
                              sss1=dddd(ipp,ivar);
                          else
                              sss2=dddd(ipp,ivar);
                          end
                      end
                      sss1=sss1*fs_factor;
                      sss2=sss2*fs_factor;

                      x_normalized(ipp,ivar) = ...
                       h_function(meann(ipp,ivar),sss1,sss2,x_normal-
ized(ipp,ivar));

                      if isteur
                          if (meann(ipp,ivar)>=0.98 && x_normal-
ized(ipp,ivar)>meann(ipp,ivar))
                              x_normalized(ipp,ivar)=1;
                          elseif (meann(ipp,ivar)<=0.02 && x_normal-
ized(ipp,ivar)<meann(ipp,ivar))
                              x_normalized(ipp,ivar)=0;
                          end
                      end
                  end
              end
             end

              ofcn(fe_count1,:)=table.objective(1,:,ip_min);
              fitn(fe_count1,:)=table.fitness(1,:,ip_min);
              feas(fe_count1,:)=table.feasibility(1,:,ip_min); %not neces-
sary (s.a.)
              best(fe_count1,:)=parameter.x_min+(parameter.x_max-parame-
ter.x_min).*table.bests(1,:,ip_min);
              best_norm(fe_count1,:)=table.bests(1,:,ip_min);

          end %for n parrrrrr
      end
```

```
    %% ---------------------- Complementary functions ------------------
------
        function save_best_feasible()
        no_in(ipp) = no_in(ipp)+1;
        changed = false;

        if no_in(ipp) ==1 % the solution coming to the table for the
first time
            table.bests(1,:,ipp) = x_normalized(ipp,:);
            table.objective(1,:,ipp) = xt.objective;
            table.fitness(1,:,ipp) = xt.fitness;
            table.feasibility(1,:,ipp) =xt.feasibility;

            x_normalized_best(ipp,:) = table.bests(1,:,ipp); % set the
best solution to the one of the first rank
            no_inin(ipp)=no_inin(ipp)+1;
        else % not for the first time and check for the update
           i_position =0;
           % check if the new coming solution is less than any in the ta-
ble
% % % %            if (xt.fitness<table.fitness(n_to_save,:,ipp))
               for i=1:n_to_save
                   if changed; break; end
                   ixbetter = elitism(xt,table,ipp,i);
% % % %                    if (xt.fitness<table.fitness(i,:,ipp))
%ixbetter
                   if ixbetter
                       i_position = i;
                       changed =true;
                       if (i<n_to_save)
                           no_inin(ipp) = no_inin(ipp)+1; % how many
times good solutions were found
                       end
% % % %                            break;
                   end
               end
% % % %            end
        end

        if changed   % if the new individual is better than any archived
individual.
                     % Move the individuals and corresponding fitness
values
                     % downward so that the individuals are sorted based
on the
                     % fitness value in a descending order

            nnnnnn = n_to_save;
            if (no_inin(ipp) < n_to_save); nnnnnn = no_inin(ipp); end
            for i=nnnnnn:-1:i_position+1
                table.objective(i,:,ipp) = table.objective(i-1,:,ipp);
                table.bests(i,:,ipp) = table.bests(i-1,:,ipp);
                table.fitness(i,:,ipp)= table.fitness(i-1,:,ipp);
                table.feasibility(i,:,ipp)= table.feasibility(i-1,:,ipp);
            end

            % save the new best
            table.bests(i_position,:,ipp) = x_normalized(ipp,:);
            table.objective(i_position,:,ipp) = xt.objective(1);
```

```
            table.fitness(i_position,:,ipp) = xt.fitness;
            table.feasibility(i_position,:,ipp) =xt.feasibility;

            % calculation of mean and variance
            if ((no_inin(ipp)>l_vari))
                for ivvar = 1:n_var
                    [meann(ipp,ivvar),variance(ipp,ivvar)] =
mv_noneq(nnnnnn,table.bests(1:nnnnnn,ivvar,ipp));
                end
                id_nonzero = (variance(ipp,:) > 1.1d-100 );%1.d-100);
                shape(ipp,id_nonzero) = -log(variance(ipp,id_nonzero));
            end

            % Set new global best
            x_normalized_best(ipp,:) = table.bests(1,:,ipp);

        end
        end

        function VariableSelect1()
            if (mode == 1)
                for ii=1:n_randomly
                    isrepeat = false;
                    while ~isrepeat
                        inn=round(rand*(n_var-1))+1;
                        if (~considered(ipp,inn))
                            isrepeat = true;
                        end
                    end
                    considered(ipp,inn)=true;
                end
            elseif (mode == 2)
                in_randomly=0;
                isrepeat = false;
                izz(ipp)=round(rand*(n_var-1))+1; %NEWWWWW
                while ~isrepeat
                    in_randomly=in_randomly+1;
                    if (izz(ipp)< 1)
                        izz(ipp)=n_var;
                    end
                    considered(ipp,izz(ipp))=true;
                    izz(ipp)=izz(ipp)-1;
                    if (~(in_randomly<n_randomly))
                        isrepeat = true;
                    end
                end
            elseif (mode == 3)
                in_randomly=0;
                izm(ipp)=izm(ipp)-1;
                if (izm(ipp)< 1)
                    izm(ipp)=n_var;
                end
                izz(ipp)=izm(ipp);
                isrepeat = false;
                while ~isrepeat
                    in_randomly=in_randomly+1;
                    if (izz(ipp)< 1)
                        izz(ipp)=n_var;
                    end
```

```
                    considered(ipp,izz(ipp))=true;
                    izz(ipp)=izz(ipp)-1;
                    if (~(in_randomly<n_randomly))
                        isrepeat = true;
                    end
                end
            elseif (mode == 4)
                izm(ipp)=izm(ipp)-1;
                if (izm(ipp)< 1)
                    izm(ipp)=n_var;
                end
                considered(ipp,izm(ipp))=true;
                if (n_randomly>1)
                    for ii=1:n_randomly-1
                        isrepeat = false;
                        while ~isrepeat
                            inn=round(rand*(n_var-1))+1;
                            if (~considered(ipp,inn))
                                isrepeat = true;
                            end
                        end
                        considered(ipp,inn)=true;
                    end
                end
            end
        end

     function f_fit = static_penalty(fxin,gxin)
            ggxin = gxin;
            ggxin(gxin<1.0d-6) = 0.0 ;
            N =(ggxin > 0.0) ;
            NN=sum(N) ;
           if NN < 1
               NN=1;
           end
           f_fit = fxin+ pnlty_factor0(ipp)*NN*sum(ggxin.^2) ;
    %        f_fit = fxin+ 1.d5*sum(abs(ggxin)) ;
     end

       function isbetter= elitism(xnew,xtable,ipp,i)
            isbetter = false;
            if (xnew.feasibility && xtable.feasibil-
ity(i,:,ipp))&&(xnew.objective < xtable.objective(i,:,ipp))
                isbetter = true;
            elseif (~xnew.feasibility && ~xtable.feasibil-
ity(i,:,ipp))&&(xnew.fitness <xtable.fitness(i,:,ipp))
                isbetter = true;
            elseif ( xnew.feasibility && ~xtable.feasibility(i,:,ipp))
                isbetter = true;
            end
        end

        function [vmean,vvariance] = mv_noneq(n_to_save,values)
            values_noneq = zeros(n_to_save,1);
            iz =1;
            values_noneq(iz)=values(1);
            for ii_jj=2:n_to_save
                izz = iz;
                gleich = false;
```

```
                for kk_ii=1:izz
                    if (values_noneq(kk_ii) == values(ii_jj));
                        gleich = true;
                        break;
                    end
                end
                if ~gleich;
                    iz = iz+1;
                    values_noneq(iz)=values(ii_jj);
                end
            end
            vmean = values_noneq(1);
            if (iz>1)
                for kk_ii=2:iz
                    vmean = vmean+values_noneq(kk_ii);
                end
                vmean = vmean/iz;
            end
            vvariance = 0;
            if (iz>1)
                for kk_ii=1:iz
                    vvariance =vvariance+(values_noneq(kk_ii)-
vmean)*(values_noneq(kk_ii)-vmean);
                end
                vvariance = vvariance/iz;
            else
                vvariance=1.0d-100;%1.1d-200;
            end
        end

        %% Evacuated h-function
     function x = h_function(x_bar,s1,s2,x_p)
         H = x_bar .* (1.d0 - exp(-x_p.*s1)) + ...
             (1.0d0 - x_bar) .* exp(-(1.d0-x_p).*s2);
         H0 = (1.d0 - x_bar) .* exp(-s2);
         H1 = x_bar .* exp(-s1);
         x = H + H1 .* x_p + H0 .* (x_p - 1.d0);
     end

    end
```