UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIAS
MAESTRÍA EN CIENCIAS MENCIÓN OPTIMIZACIÓN

APLICACIÓN DEL ALGORITMO EVOLUTIVO NSGAII EN LA UBICACIÓN DE CENTRALES TELEFÓNICAS EN UN ÁREA GEOGRÁFICA DETERMINADA

Autor: Lcda. Clavel Quintana
Tutor: Dra. Flor Montes de Oca

Barquisimeto, julio de 2009

UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIAS
MAESTRÍA EN CIENCIAS MENCIÓN OPTIMIZACIÓN

APLICACIÓN DEL ALGORITMO EVOLUTIVO NSGAII EN LA UBICACIÓN DE CENTRALES TELEFÓNICAS EN UN ÁREA GEOGRÁFICA DETERMINADA

Trabajo presentado para optar al Grado de
Magister Scientiarum. Mención Optimización

Autor: Lcda. Clavel Quintana
Tutor: Dra. Flor Montes de Oca

Barquisimeto, julio de 2009

Resumen

Almeida, Amarilla y Barán [2] estudiaron el problema de minimización multiobjetivo de costo a corto, mediano y largo plazo para la ubicación de centrales telefónicas en la ciudad de la Asunción, Paraguay.

Ellos determinaron el número de centrales y la ubicación óptima de estas centrales usando el algoritmo evolutivo SPEAII. En este trabajo aplicamos el algoritmo evolutivo NSGAII al problema de ubicar centrales telefónicas en Cabudare, Estado Lara, Venezuela; mas aún se realizarón pruebas númericas del NSGAII con los operadores de cruce SBX y BLX-α y operadores de mutación gaussiana, uniforme y polinomial demostrando que constituye una opción valida para abordar el problema multiobjetivo.

Agradecimientos

A Dios por guiarme durante toda mi vida universitaria y por iluminar mi camino en aquellos momentos en los que me pareció muy díficil de alcanzar esta meta.

A mis padres Loida Carlone y German Quintana quienes con sus palabras de alientos, consejos y su apoyo me animaron a continuar.

A mis hermanas y grandes amigas Rosana, Raquel y Claravel.

A mi tutora Flor Montes de Oca que me ayudo con sus conocimientos y su tiempo para guiarme en la elaboración de este trabajo.

A mis profesores de siempre Eibar Hernandez y Romulo Castillo.

A mis cuñados Humberto Bacino y Dany Peña, gracias por su amistad.

A mis amigos Marbelis Rodriguez, Marlyn Cuadrado y Adrian Rojas, gracias por su ayuda.

A Juan Carlos Juárez mi amigo, siempre estaré agradecida por tu ayuda y correcciones necesarias.

A Dios y a mis padres

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Algoritmos genéticos

Los algoritmos genéticos, integran una serie de técnicas inspiradas en los principios de la teoría Neo-Darwiniana de la evolución natural, desarrollados por John Holland a principios de 1960 y motivados en resolver problemas de Inteligencia Artificial. La simulación de procesos de evolución natural de las especies, da como resultado una técnica de optimización estocástica, que posteriormente fue llamado **algoritmos evolutivos**, y que están enmarcadas dentro de las técnicas no convencionales de optimización para problemas del mundo real.

Los algoritmos genéticos basan su funcionamiento en la simulación del proceso de evolución natural, reproducción y supervivencia del más apto; según Goldberg (véase [10]):

> *Los algoritmos genéticos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural, combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado aunque aleatorio, para construir así un algoritmo de búsqueda que tengan algo de la genialidad humana.*

1.1. Elementos básicos que conforman la aplicación de un algoritmo genético

Codificación del problema

En los algoritmos genéticos es necesaria una buena representación o codificación para describir cada individuo en la población de interés, mas aún la representación determina los operadores genéticos que se utilizarán.

Las soluciones del problema se codifican mediante cadenas de caracteres denominados **cromosomas**. Cada individuo o cromosoma está formado por una secuencia de genes de un determinado alfabeto binario, números enteros, símbolos reales entre otros.

Una buena codificación debe representar todo el espacio de soluciones, no debe generar individuos incorrectos al aplicar los operadores y debe cubrir todo el espacio de búsqueda de una manera continua.

Desde los primeros trabajos de John Holland, la codificación suele hacerse mediante valores binarios. Desde entonces, el problema de la representación ha sido objeto de muchas investigaciones debido a que la representación binaria tiene varias desventajas cuando el algoritmo genético se usa para resolver ciertos problemas del mundo real (véase [7], [9], [10] y [6]).

En la codificación binaria, se asigna un determinado número de bits a cada parámetro (genes) y se realiza una discretización de la variable representada para cada gen. El número de bits asignado dependerá del grado de ajuste que se desee alcanzar. Cada uno de los bits pertenecientes al gen reciben el nombre de **alelo**.

El conjunto de parámetros representados por un cromosoma particular recibe el nombre de **genotipo**. El genotipo contiene la información necesaria para la construcción del organismo, y la solución real al problema se denominada **fenotipo**.

Ejemplo 1.1.1 *La representación tradicional es la representacion binaria, y un cromosoma puede representarse como:*

$$cromosoma \rightarrow \boxed{\begin{array}{c}0\\1\\1\\0\\1\end{array}}$$

A la cadena binaria, se la llama **cromosoma**. *A cada posición de la cadena se le denomina* **gen** *y el valor dentro de esta posición se le llama* **alelo**.

La decodificación para este cromosoma sería:

$$0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 13$$

Donde 13 *representa el fenotipo para este cromosoma. En el espacio real:*

$$cromosoma \rightarrow \boxed{\begin{array}{c}2{,}15\\1{,}89\\0{,}43\\3{,}14\\5{,}11\end{array}}$$

En el enfoque tradicional, se usa un número binario para representar un número real, que define limite inferior y superior para cada variable, así como la precisión deseada. Por ejemplo,

si queremos codificar una variable que va de 0,35 a 1,40, usando una precisión de 2 decimales, necesitaremos $\log(140 - 35) \approx 7$ bits para representar cualquier número real dentro de ese rango. Uno de los principales problemas de esta representación es que no mapea adecuadamente el espacio de busqueda con el espacio de representación.

Otra dificultad es el risco de Hamming (código predictor de errores) asociado con ciertos strings (tales como 01111 y 10000 equivalentes a 15 y 16 respectivamente) de los cuales una transición de soluciones vecinas (en el espacio real) requiere la alteración de muchos bits.

Existe otro problema más importante, cuando tratamos de desarrollar aplicaciones del mundo real: **alta dimensionalidad**.

Si tenemos demasiadas variables, y queremos una buena precisión para cada una de ellas, entonces las cadenas binarias que se produzcan se volverán extremadamente largas y el algoritmo genético tendrá un desempeño pobre.

Con una representación real se puede tratar las molestias ocasionadas por los riscos de Hamming, producidos cuando las variables utilizadas son números reales, porque un cambio pequeño en la representación es mapeado como un cambio pequeño en el espacio de búsqueda y aunque esta representación tiene una cardinalidad más alta y el comportamiento del **Algoritmo genético** podría ser difícil de predecir, se han diseñado recientemente varios operadores especiales para imitar las acciones de la cruza y la mutación en los alfabetos binarios procurando igualarlos e incluso excederlos en cuanto a eficiencia (véase [9]).

Función fitness

El otro aspecto fundamental en el desarrollo de un algoritmo genético es la elección de una buena función de costo o función fitness, la cual debe ser diseñada para cada problema de manera específica. Dado un cromosoma particular, la función de adaptación le asigna un número real, que se supone refleja el nivel de adaptación al problema del individuo representado por el cromosoma.

El **fitness** es una medida de desempeño de la solución; es decir, permite decidir que tan buena es una solución con respecto a otra.

Aquellos individuos cuyo valor de función de fitness sea mejor, tendrán más posibilidades de ser seleccionado para construir la siguiente población y por lo tanto, para pasar sus características a los individuos de la siguiente generación.

Operadores Evolutivos

Los operadores genéticos proporcionan el mecanismos de búsqueda de los algoritmos genéticos. Los algoritmos genéticos están compuestos por tres operadores principales: **selección**, **cruce** y **mutación**, los cuales se describirán a continuación:

1. **Selección**

   La selección es un proceso por medio del cual los individuos son escogidos de la población de acuerdo al valor de su fitness, para someterse a la acción futura de otros operadores. Existen varias técnicas para el proceso de selección: **selección por ruleta**, **torneo**, **métodos de rango**, etc.

   Entre los métodos de selección más utilizados en la representación binaria está la selección por ruleta, desarrollada por Holland (véase [10]), donde a cada uno de los individuos de la población se le asigna una parte proporcional a su fitness en una ruleta, de tal forma que la suma de los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Generalmente, la población está ordenada en base al ajuste por lo que las porciones más grandes se encuentran al inicio de las ruleta.

   Para seleccionar a un individuo basta con generar un número aleatorio en $[0, 1]$ y devolver el individuo situado en esa posición de la ruleta. Esta posición se suele obtener recorriendo los individuos de la población y acumulando sus porciones hasta que la suma exceda el valor obtenido.

   La probabilidad de selección de un individuo es:

   $$p_s(i) = \frac{P(i)}{\sum_{i=1}^{n} P(i)}$$

   donde $P(i)$ es el fitness del individuo i.

Ejemplo 1.1.2 *Maximizar* $f(x) = x^2$ *sobre los enteros* $[0, 31]$.

N°	**Población**	x	**fitness**	**porcentaje**
1	01101	13	169	14,4
2	11000	24	576	49,2
3	01000	8	64	5,5
4	10011	19	361	30,9
Total			**1170**	**100,0**

Selección por ruleta

Calculamos la suma de los fitness, $sumfitness = 1170$

Calculamos la probabilidad de selección de cada individuo:

$$p_s(1) = \frac{169}{1170} = 0{,}14 \qquad p_s(2) = \frac{576}{1170} = 0{,}49$$
$$p_s(3) = \frac{64}{1170} = 0{,}054 \qquad p_s(4) = \frac{361}{1170} = 0{,}31$$

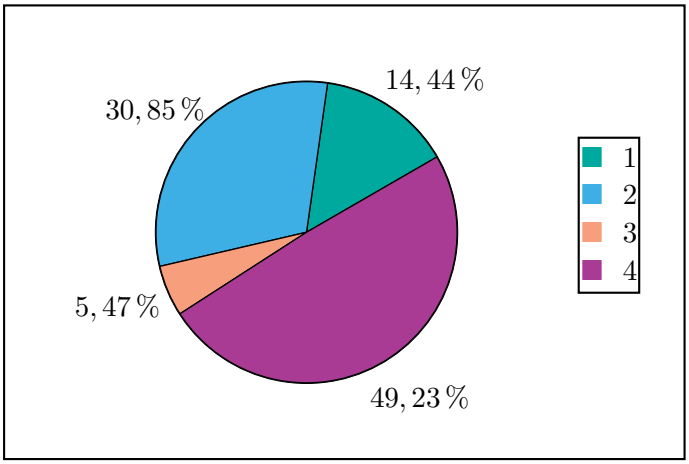


Figura 1.1: Selección por ruleta

El número de veces que esperamos que ocurra un evento es $p_s(i)N$*, donde* N *es el tamaño de la población. Así,* 0,56*;* 1,96*;* 0,21*;* 1,24 *serán los valores esperados de los individuos* $1, 2, 3, 4$*, respectivamente.*

Para seleccionar al individuo que entra en el conjunto población, generamos un número aleatorio r*, supongamos que* $r = 0{,}62$ *(representa el* 62 % *de la ruleta), ahora recorremos secuencialmente los individuos sumando los valores esperados.*

Hasta el individuo 1*, la suma es* $0{,}56 < r$*, por lo tanto, no se selecciona. En el primer individuo se cubre de* 0 *al* 14,4 % *de la ruleta como podemos observar en la Figura 1.1; así* 62 *no pertenece a este rango.*

Hasta el individuo 2 *la suma es* $2{,}52 > r$*, de esta forma, este individuo es seleccionado para la acción de próximos operadores. Observe que en el segundo individuo se cubre del* 14,4 % *al* 63,6 % *de la ruleta.*

La técnica de la ruleta limita el uso del algoritmo genético a problemas de maximización ya que la función de evaluación debe mapear los valores sobre $\mathbb{R}^+$, para el caso de minimización Goldberg ha propuesto algunas extensiones (véase [10]).

La selección por torneo, trabaja mediante la selección aleatoria de una solución j con reemplazo de la población y coloca a la mejor solución j en la nueva población. Este proceso culmina cuando se han seleccionado **N** individuos (tamaño de la población).

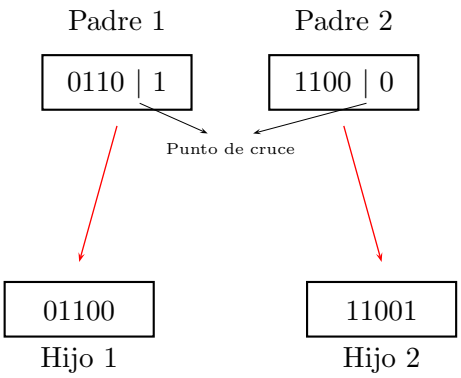
Los métodos de rango solo requieren la evaluación de la función para mapear la solución a un conjunto ordenado, aquí la población se ordena en subpoblaciones o niveles que definen la jerarquía de cada solución, es decir, si el individuo pertenece al primer nivel tendrá más probabilidad de ser seleccionado. Los métodos de rango asignan la probabilidad de selección a una solución cuando todas las soluciones son ordenadas.

2. **Cruza**

   La cruza es un proceso complejo que ocurre en parejas de cromosomas. Estos cromosomas se alinean, luego se fraccionan en ciertas partes y posteriormente intercambian fragmentos entre sí. La aplicación de este operador depende de la representación básica usada, ya sea binaria, entera o real.

   En la representación binaria se usa con frecuencia la cruza de un punto, la cual se inicia a partir de la obtención, mediante la selección por ruleta, de los individuos que serán sometidos al operador. El cruzamiento requiere la selección de dos cadenas como padres, el cual se realiza por medio de la ruleta, luego se selecciona un punto de cruce, tal que dicho punto se encuentre entre 1 y $l-1$, donde l es la longitud de la cadena. Los padres son cortados en este punto para originar cuatro subcadenas, que se intercambian para generar dos hijos.

   Ejemplo 1.1.3 *Inicialmente seleccionamos a cuatro individuos de la población del ejemplo 1.1.2, esto lo hacemos aplicando la selección por ruleta. Supongamos que los individuos seleccionados son:* $1, 2, 2, 4$*; procedemos a seleccionar a los padres durante este proceso, aplicamos el método de la ruleta para conocer con quien emparejaremos a cada individuo. Supongamos que la pareja seleccionada son los individuos* 1 *y* 2*, aplicamos nuevamente el método de la ruleta para determinar el punto de cruce entre* 1 *y* $l-1$*, donde* $l = 5$ *y realizamos el cruce:*



Entre los operadores de cruce más usados en la representación real se encuentran:

- **Simulación de cruce binaria (¨Simulated Binary Crossover (SBX)¨)**, y
- **Cruce de Mezcla (¨Blend Crossover (BLX-α)¨)**.

- **Simulación cruce binaria**

La simulacion binaria (SBX) ([1], [9]) trabaja con dos pares de soluciones padres y crea dos descendientes. El SBX simula el trabajo del operador del cruce de un punto de la representación binaria.

Consideraremos $x_i^{k,t}$ como coordenada i-ésima de un individuo o cromosoma de la población proveniente del padre k en la generación t. El proceso de cálculo de la descendencia $x_i^{1,t+1}$ y $x_i^{2,t+1}$ de las soluciones padres $x_i^{1,t}$ y $x_i^{2,t}$ se describe como sigue: se define un factor β_i, como el radio de la diferencia absoluta entre los valores de la descendencia de los padres, esto es,

$$\beta_i = \left| \frac{x_i^{2,t+1} - x_i^{1,t+1}}{x_i^{2,t} - x_i^{1,t}} \right| \tag{1.1}$$

En primer lugar, se escoge un número aleatorio entre 0 y 1. Posteriormente, a partir de una distribución de probabilidad, se encuentra β_{qi} a fin de que el área bajo la curva de probabilidad en el intervalo $[0, \beta_{qi}]$ sea igual al número escogido al azar.

La probabilidad de distribución utilizada para crear la descendencia es:

$$p(\beta_i) = \begin{cases} \frac{1}{2}(\eta_c + 1)\beta^{\eta_c} & \beta_i \leq 1, \\ \frac{1}{2}(\eta_c + 1)\frac{1}{\beta^{\eta_c+1}} & \text{en otro caso.} \end{cases} \tag{1.2}$$

Esta función de densidad de probabilidad se deriva de tener un poder de búsqueda similar a la del cruce simple binario. El parámetro η_c es cualquier número real no negativo. Un valor grande de η_c nos da una mayor probabilidad para la creación de las soluciones cerca de los padres y un valor pequeño permite que soluciones lejanas a los padres sean seleccionadas como descendencia. En las poblaciones iniciales puede ocurrir que la descendencia este alejada de los padres, pero a medida que las soluciones comienzan a converger no se permite que los descendientes se alejen de los padres. Así el SBX enfoca la búsqueda a regiones reducidas. Esencialmente el SBX tiene dos propiedades:

a) La diferencia entre la descendencia es proporcional a la de los padres.

b) Las soluciones cercanas a los padres son escogidas en lugar de las soluciones lejanas. (La descendencia conserva las características de los padres).

De la función de densidad (1.2) obtenemos:

$$\beta_{qi} = \begin{cases} 2(u_i)^{\frac{1}{\eta_c+1}} & u_i \leq 0{,}5, \\ (\frac{1}{2(1-u_i)})^{\eta_c+1} & \text{en otro caso.} \end{cases} \tag{1.3}$$

Luego la descendencia es calculada como sigue:

$$x_i^{1,t+1} = \frac{1}{2}[(1+\beta_{qi})x_i^{1,t} + (1-\beta_{qi})x_i^{2,t}] \tag{1.4}$$

$$x_i^{2,t+1} = \frac{1}{2}[(1-\beta_{qi})x_i^{1,t} + (1+\beta_{qi})x_i^{2,t}] \tag{1.5}$$

Notar que los descendientes son simétricos a los padres, esto evita la inclinación hacia cualquier solución padre particular en el cruce de un punto.

Se puede describir el SBX de forma general como:

- Seleccionar dos cromosomas padres.
- Generar un número aleatorio entre 0 y 1.
- Generar la descendencia usando una función de densidad de probabilidad.

Cruce de mezcla (BLX-α)

Eshelman y Shaffer sugieren un operador para la representación real el BLX-α (véase [6], [9]). El BLX-α trabaja con un par de soluciones padres $x_i^{1,t}$ y $x_i^{2,t}$ (se asume que $x_i^{1,t} < x_i^{2,t}$) y genera los descendientes $x_i^{1,t+1}$ y $x_i^{2,t+1}$ usando una distribución uniforme; para esto se genera aleatoriamente una solución en el rango $[x_i^{1,t} - \alpha(x_i^{2,t} - x_i^{1,t}), x_i^{2,t} + \alpha(x_i^{2,t} - x_i^{1,t})]$.

El proceso de cálculo del descendiente es como sigue:
Generar un número aleatorio u_i entre 0 y 1 y calcular el descendiente como:

$$x_i^{(1,t+1)} = (1-y_i)x_i^{(1,t)} + y_i x_i^{(2,t)}$$

donde $y_i = (1+2\alpha)u_i - \alpha$,$x_i^{2,t+1}$ se calcula de manera similar. Se puede observar que si α es cero, el cruce genera una solución en el rango $(x_i^{(1,t)}, x_i^{(2,t)})$.

Deb reportó ([6]) que para un valor de α igual a 0,5, se tiene una buena representación de cruce, sin embargo, para un valor de α pequeño la descendencia comienza a coincidir en las generaciones iniciales con el padre lo cual no permite que el algoritmo mejore rápidamente en las proximas iteraciones y para α grande, la descendencia se aleja bastante de los padres. El BLX-α tiene la propiedad de localizar la descendencia según la diferencia de las soluciones padres, esto es:

$$x_i^{(1,t+1)} - x_i^{(1,t)} = y_i(x_i^{(2,t)} - x_i^{(1,t)})$$

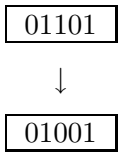
Si la diferencia entre las soluciones padres es pequeña, la diferencia entre la descendencia y el padre solucion es pequeña; es decir, si los padres son cercanos, el hijo estará cerca del padre y viceversa.

3. **Mutación**

La mutación es utilizada como un operador cuyo propósito es la exploración aleatoria de nuevas porciones de espacios de busqueda. Este operador es el encargado de introducir nuevo material genético en la búsqueda de soluciones, ya que el cruce no introduce ningún material nuevo.

Cuando la codificación utilizada es binaria el operador de mutación simplemente modifica el valor de un bit en la cadena con una probabilidad dada.

Ejemplo 1.1.4



Entre los operadores de mutación mas usados en la representación real se encuentran:

a) Mutación polinomial.

b) Mutación Uniforme.

c) Mutación Gaussiana.

a) **Mutación polinomial**

En la representación real uno de los métodos más conocidos es la mutación polinomial, en el cual se usa una función de densidad de probabilidad:

$$p(\delta) = 0{,}5(\eta_m + 1)(1 - |\delta|)^{\eta_m} \tag{1.6}$$

El parámetro η_m determina que tan lejos o cerca se encuentra el individuo con el gen mutado del padre. Para valores muy pequeños del η_m, la distancia es grande y además en algunas ocasiones puede incurrir en problemas de infactibilidad en cuanto al descendiente, sin embargo, a medida que este valor crece la distancia entre padre y descendiente disminuye y preserva la factibilidad.

Similar al SBX se genera un número aleatorio y posteriormente a partir de la distribución de probabilidad se encuentra δ_i a fin de que el área bajo la curva de probabilidad en el intervalo $[0, \delta_i]$ sea igual al número aleatorio r_i. Así:

$$\delta_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m}+1} - 1 & r_i < 0{,}5 \\ 1 - [2(1-r_i)]^{\frac{1}{\eta_m+1}} & r_i \geq 0{,}5 \end{cases} \tag{1.7}$$

El cálculo de la posición mutada será:

$$y_i^{1,1+t} = x_i^{1,t+1} + (x_i^U - x_i^L)\delta_i \tag{1.8}$$

Los valores de las cotas mínimas y máximas de x_i están representadas por x_i^L, x_i^U.

Se puede describir el algoritmo en forma general como:

- Seleccionar un padre.
- Generar un número aleatorio entre 0 y 1.
- Generar el valor de la posición mutada usando una función de densidad de probabilidad.

En este trabajo, los parámetros relacionados del cruce η_c y la mutación η_m, se les asignará el valor 20 recomendado por Deb (véase [9]).

b) **Mutación Uniforme**

La mutación más simple es la creación aleatoria de un descendiente en el espacio de busqueda mediante la sustitución de un gen por un número aleatorio, esto es:

$$y_i^{(1,t+1)} = r_i(x_i^U - x_i^L)$$

donde r_i es un número entre $[0, 1]$.

c) **Mutación Gaussiana**

Dado un padre y un gen seleccionado para la mutación se genera un descendiente como:

$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + N(0, \sigma_i)$$

donde $N(0, \sigma_i)$ es una distribución normal con media cero y desviación estandar σ_i, esta desviación estandar es un parámetro fijo y definido por el usuario. En este trabajo se usa σ_i igual a 0,5.

Ajuste de parámetros

Los parámetros principales de un algoritmo genético son:

1. Tamaño de la población.

2. Porcentaje de cruza.

3. Porcentaje de mutación.

La forma óptima de definir estos parámetros en un algoritmo genético ha sido motivo de investigaciones desde los origenes mismos de la técnica, y no existe hasta la fecha una solución satisfactoria a este problema.

El tamaño de la población es un parámetro crucial para una aplicación exitosa de los algoritmos genéticos.

Experimentalmente Deb ([9]) observó que al maximizar una función bimodal:

$$f(x) = c_1 N(x, a_1, b_1) + c_2 N(x, a_2, b_2)$$

Donde $N(x, a_i, b_i)$ es la función gaussiana con media a_i y desviación estandar b_i, un tamaño de población pequeña hace que el algoritmo converja al máximo local, pero a medida que aumenta el número de individuos este converge al máximo global. Mas aún, el tamaño de la población está relacionado con la complejidad del problema, sin embargo, una población muy grande requiere muchos recursos computacionales. Aunque normalmente se elige un tamaño fijo de la población, también se puede considerar un tamaño de población variable. En este trabajo, se usará una población de 100 individuos, parámetro recomendado por Baran ([2]). De esta misma forma se debe establecer un número óptimo de generaciones (iteraciones) para asegurar la convergencia cercana a la solución óptima; Deb ([9]) recomienda un número de generaciones entre 250 y 1000, puesto que los algoritmos evolutivos han demostrado tener un buen desempeño en estos casos. Además, se establecerá un número de generaciones entre 1000 y 3000, sugeridos por Baran ([2]), debido a la alta dimensionalidad del problema.

Los parámetros sugeridos según DeJong, Shaffer y Goldberg ([7]) para cruce y mutación son:

• porcentaje de cruza de $0{,}65 - 0{,}95$

• porcentaje de mutación de $0{,}001 - 0{,}005$.

Sin embargo, Deb ([9]) recomienda un porcentaje de cruza de 0,9 y de mutación $1/n$ o $1/l$, donde n es el número de variables de decisión para la representación real y l es la longitud del string para la representación binaria. Estos parámetros permiten al algoritmo genético tener un desempeño razonablemente bueno.

1.2. Métodos tradicionales de búsqueda y algoritmos genéticos

Existen muchas técnicas de optimización clásica, para resolver problemas de optimización, tanto en el área restringida como no restringida, tales como el método simplex, para problemas

de optimización lineal y no lineal; el método de máximo descenso, para encontrar el óptimo de una función no restringida; o el método de Newton, entre otros. Estos métodos suelen ser eficientes para problemas cuyas características específicas se adaptan a los principios teóricos que rigen a estas técnicas, sin embargo, existen problemas que por su naturaleza y por la limitación de herramientas computacionales son difíciles de abordar con estas técnicas clásicas.

Uno de los problemas de las técnicas clásicas de optimización es que suelen requerir información que no siempre está disponible. Por ejemplo, métodos como el del gradiente conjugado requieren de la primera derivada de la función objetivo. Otros como el de Newton, requieren además de la segunda derivada. Por tanto, si la función objetivo no es diferenciable (y en algunos problemas de la vida real, ni siquiera está disponible en forma explicita), estos métodos no pueden aplicarse.

Desventajas de los métodos tradicionales.

1. Suposiciones restrictivas de continuidad, existencia de derivadas, entre otros, para la función objetivo.

2. Buscan la solución a partir de un punto dado.

3. Poseen alcance local, el óptimo que buscan son los mejores en las cercanías del punto actual, por lo que puedan con facilidad perderse en el evento actual.

Cuando enfrentamos espacios de búsqueda tan grande, como en el caso del problema del agente viajero, los algoritmos más eficientes que existen para resolverlo requieren tiempo exponencial, por lo que las técnicas clásicas de optimización resultan ineficientes y recurrir a la heurística puede ser una buena alternativa.

En los ultimos años han aparecido una serie de métodos, bajo el nombre de metaheurísticas, los cuales son estrategias para diseñar o mejorar los procedimientos heurísticos con miras a obtener un alto rendimiento.

El término metaheurístico fue introducido por Fred Glover en 1986, y a partir de entonces han aparecido muchas propuestas para diseñar mejores procedimientos de solución.

Entre las metaheurísticas más comunes se encuentran los algoritmos genéticos los cuales han demostrado resolver problemas lineales y no lineales, mediante la exploración de todas las regiones o espacios de solución y explotación de manera exponencial a través de los operadores genéticos.

1.3. Diferencias entre métodos tradicionales de búsqueda y los algoritmos genéticos

Los algoritmos genéticos (GAs) son diferentes de la optimización tradicional en los procedimientos de búsquedas:

1. (GAs) no necesitan conocimiento específico sobre sobre el problema que intentan resolver, como derivada de la función o dimensión del problema.

2. (GAs) buscan una población de puntos solución, no un solo punto, lo cual los hace menos sensible a quedar atrapados en mínimos o máximos locales.

3. (GAs) usa reglas de transición probabilísticas no determinísticas.

4. Los (GAs) requieren de un conjunto de parámetros naturales, tales como parámetros de mutación, cruce, tamaño de la población y número de generaciones los cuales permiten al algoritmo un desempeño razonablemente bueno.

1.4. Algoritmo genético básico

Los algoritmos genéticos parten de una población inicial generada aleatoriamente donde cada elemento de la población se le asigna un valor fitness, este permite seleccionar a los individuos más aptos para la próxima generación, la cual evoluciona mediante la acción de los operadores genéticos.

El algoritmo básico es el siguiente:

1. Generar (aleatoriamente una población inicial).

2. Calcular la aptitud de cada individuo.

3. Seleccionar (probabilísticamente en base a la aptitud).

4. Aplicar operadores genéticos (cruza y mutación para generar la población siguiente).

5. Repetir los pasos $2, 3$ y 4, hasta que verifique cierta condición.

1.5. Optimización multiobjetivo

Problemas con objetivos múltiples aparecen de forma natural en muchos problemas de optimización y constituyen un área que ha adquirido gran relevancia en el mundo de la investigación científica y en aplicaciones de ingeniería, un ejemplo, lo constituye el crecimiento del consumo y

variedad de los servicios de telecomunicaciones, los cuales generan una necesidad cada vez mayor de implementar herramientas eficientes para la planificación de las redes de comunicación a fin de minimizar los altos costos de inversión y mantenimiento.

El problema de optimización multiobjetivo (POM), también llamado multicriterio o vectorial, puede definirse como el problema de encontrar un vector de variables de decisión que satisface un cierto conjunto de restricciones y optimice un conjunto de funciones objetivos.

Definición 1.5.1 *Un problema de optimización multiobjetivo tiene M funciones objetivos, las cuales son minimizadas o maximizadas. En general se define como:*

$$\begin{array}{|c|c|c|}\hline \text{Minimizar-Maximizar} & f_m(x) & m = 1, 2, \ldots, M \\ s.a. & g_j(x) \leq 0 & j = 1, 2, \ldots, J \\ & h_k(x) = 0 & k = 1, 2, \ldots, K \\ & x_i^L \leq x_i \leq x_i^U & \\ \hline \end{array} \tag{1.9}$$

Una solución x es un vector de n variables de decisión $x = (x_1, x_2, \ldots, x_n)^T$ que satisface las restricciones donde cada x_i toma valores entre x_i^L y x_i^U, cotas inferior y superior, respectivamente.

Ejemplo 1.5.1 *Deseamos encontrar, un número real x que sea un mínimo, en el problema de optimización multiobjetivo planteado por Schaffer ([9]). Minimizar:*

$$f(x) = \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$$

con

$$-10^3 \leq x \leq 10^3$$

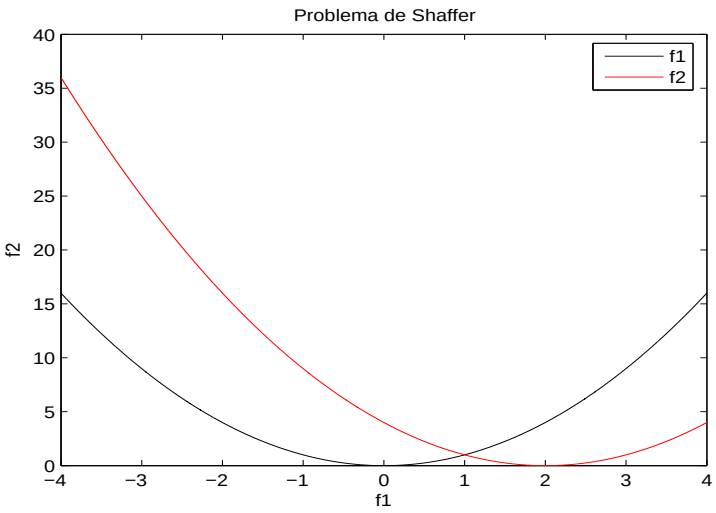


Figura 1.2: Schaffer

En la Figura 1.2 se observa que los mínimos de $f_1(x)$, $f_2(x)$ son 0 y 2, respectivamente, y a medida que $f_2(x)$ tiende a su mínimo global, $f_1(x)$ se aleja de su mínimo global y viceversa. Mas aún para:

$$\begin{cases} x = 0, & f(0) = (0, 4) \\ x = 2, & f(2) = (4, 0) \end{cases}$$

no se puede establecer cual de las dos soluciones es mejor, por esta razón la solución de un problema multiobjetivo suele ser un conjunto de soluciones, el cual se denomina **frente de Pareto**. Este conjunto está formado por aquellas soluciones que no encuentran otra solución que las mejore en todas las funciones que se busca optimizar. Sin embargo, en algunos problemas de optimización se presenta el caso de que una sola variable de decisión optimiza todas las funciones objetivos. A estas soluciones se le llama **vector ideal**.

Definición 1.5.2 *Cada una de las M funciones objetivos tiene una solución optimal diferente. Un vector que contiene estos valores objetivos individuales constituye el vector objetivo ideal.*

En general el vector ideal es inalcanzable, excepto en el caso en que no existe ningún conflicto entre las funciones objetivos del problema.

Al tener varias funciones objetivos, la noción de óptimo cambia debido a que en los problemas de optimización multiobjetivo, estamos tratando de obtener un conjunto de buenas soluciones en vez de una solución única como óptimo global.

1.6. Optimalidad de Pareto

Definición 1.6.1 *Un vector $\vec{u} = \overrightarrow{(u_1, \ldots, u_k)}$ domina a otro $\vec{v} = \overrightarrow{(v_1, \ldots, v_k)}$, denotado mediante $\vec{u} \preceq \vec{v}$ si y sólo si $\vec{u}$ es parcialmente menor $\vec{v}$, es decir, $\forall i \in \{1, \ldots, k\}$ $u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\}$ tal que $u_i < v_i$.*

Definición 1.6.2 *Una solución $x^* \in \Omega$ es óptimo de Pareto, si y sólo si no existe otro $x \in \Omega$ tal que $f_i(x)$ domine a $f_i(x^*)$, donde $i \in \{1, 2, \ldots, k\}$.*

Definición 1.6.3 *Para un problema de optimización multiobjetivo dado, el* **conjunto óptimo de Pareto** *P^* se define como:*

$$P^* = \{x \in \Omega / \nexists x^1 \in \Omega : \overrightarrow{f(x^1)} \leq \overrightarrow{f(x)}\}$$

donde Ω es la región factible.

Definición 1.6.4 *Para un problema multiobjetivo dado y un conjunto de optimos de Pareto P^*, el* **frente de Pareto** $F(P^*)$ *se define como:*

$$F(P^*) = \{\overrightarrow{f(x)} = \overrightarrow{(f_1(x), \ldots, f_M(x))} \mid x \in P^*\}$$

Definición 1.6.5 *El* **rango** *o cuenta de dominación del individuo i será el número de individuos que lo dominan.*

En las últimas décadas se ha desarrollado una amplia gama de algoritmos genéticos para optimización multiobjetivo, también llamados algoritmos evolutivos multiobjetivos. Estos problemas cuentan con complejidades propias, por ejemplo, objetivos contradictorios, que los distinguen de los problemas monobjetivos, y por ello los algoritmos evolutivos multiobjetivo tienen características diferentes a los algoritmos genéticos.

1.7. Algoritmos evolutivos multiobjetivos

Los algoritmos evolutivos multiobjetivo surgen como una extensión de los algoritmos genéticos para un objetivo, utilizando fundamentalmente varios conceptos relacionados con el tratamiento de las funciones multimodales por parte de algoritmos genéticos para un objetivo tales como la selección y operadores genéticos.

Los **algoritmos genéticos multiobjetivos**, se pueden clasificar en:

Los algoritmos no basados en Pareto

Son aquellos en los que el algoritmo evolutivo no incorpora el concepto de óptimo de Pareto en su mecanismo de selección. Un ejemplo es el algoritmo Vector Evaluated Genetic Algorithm (VEGA) propuesto por Schaffer a mediados de los 80s. El algoritmo VEGA divide la población de un algoritmo genético en tantas subpoblaciones como objetivos existan y en la etapa de selección, cada individuo es seleccionado de acuerdo al objetivo relevante en su subpoblación.

Los algoritmos basados en Pareto

Tienen su origen en 1989 en la idea de Goldberg ([10]) y estan basados en el concepto de nivel de no-dominancia de Pareto. Desde el punto de vista Pareto el primer nivel lo constituye el conjunto de soluciones no dominadas y se le asigna el rango más alto. El segundo rango, lo constituye el conjunto de soluciones no dominadas de toda la población a la que se le han quitado los individuos del primer nivel, y este procedimiento se aplica sucesivamente hasta ordenar la población. El elemento fundamental para seleccionar a un individuo de la población será el

nivel en que el individuo se encuentre. Lo anterior y los operadores de diversidad, componen las características más importantes de los algoritmos genéticos multiobjetivos.

En particular nosotros estaremos interesados en los algoritmos evolutivos multiobjetivos basados en Pareto, debido a que estos son los que han demostrado mejor desempeño computacionalmente (véase [9], [13]).

1.7.1. Esquema general de los algoritmos evolutivos basados en Pareto

1. Generar una población inicial (dar un conjunto de soluciones).
2. Crear el conjunto de no dominados (Población externa el cual tendrá un tamaño preestablecido).
3. Generar una nueva población.
4. Volver al paso 2 hasta satisfacer el criterio de parada.

A diferencia de los algoritmos genéticos simples que buscan una solución única, los algoritmos genéticos multiobjetivos tratan de encontrar tantos elementos del conjunto de Pareto como sea posible. Para ello es necesario mantener:

- Convergencia de las soluciones hacia el conjunto óptimo de Pareto.
- Mantener la diversidad en las soluciones dentro de un conjunto de óptimos de Pareto.

Para poder cumplir con estas metas los algoritmos genéticos multiobjetivos utilizan herramientas que mantienen la diversidad de la población, para evitar la convergencia hacia una única solución.

El primer intento para mantener la diversidad de la población surge con la idea de Holland, al introducir el concepto de agrupamiento (**crowding**), donde se intenta identificar situaciones donde lo individuos se concentran en espacios o nichos (esto último hace referencia a lo que sucede en la naturaleza, donde diferentes especies de animales se agrupan en función de sus características). Pero el fundamento de los algoritmos multiobjetivos actuales se basa en los conceptos de Goldberg ([10]), que permite obtener un valor modificado de aptitud para cada individuo. Para esto se ordena la población en diferentes subpoblaciones (nichos) y la aptitud de una solución en particular, depende de su ubicación dentro del nicho y de la cantidad de soluciones vecinas que tenga. A la cantidad de soluciones vecinas que tenga se le llama contador de nicho (nc_i) y la cercanía entre las soluciones se puede medir dentro del espacio de las funciones objetivos mediante una métrica Euclidiana. El valor de aptitud de el individuo i se puede obtener

como:

$$Fi = \frac{f_i}{nc_i}$$

donde $nc_i = \sum_{j=1}^{M} sh(d_{ij})$ y f_i es el valor del fitness del individuo i. M indica el número de individuos localizados en la vecindad del i-ésimo individuo. $Sh(d_{ij})$ se define como la función compartimiento (**sharing function**) y se obtiene su valor de:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{\text{shared}}}\right)^{\alpha} & d_{ij} < \sigma_{\text{shared}} \\ 0 & \text{en otro caso} \end{cases}$$

donde $\alpha = 1$; d_{ij} es una medida de la distancia entre los individuos i y j; σ_{shared} es un parámetro que determina la cercanía permitida entre individuos. Observemos que cuando d_{ij} es cero, las soluciones son muy cercanas y $sh(d_{ij})) = 1$; entre más lejanas sean las soluciones $sh(d_{ij}))$ tiende a cero.

1.8. Descripción de los algoritmos

En esta sección se presentan brevemente las características principales de algunos de los algoritmos evolutivos multiobjetivos más usados.

1. Algoritmo genético multiobjetivo (MOGA):

   El algoritmo MOGA fue propuesto por Fonseca y Fleming en 1993. Este algoritmo propone una variación de la técnica de los niveles de dominación de Goldberg ([10]), en el algoritmo MOGA la jerarquía de dominación o rango de un individuo r_i es igual al número de individuos de la población que lo dominan n_i más uno. Es claro que cada individuo de el conjunto de Pareto de la población tiene un valor de jerarquía o rango igual a uno ya que no son dominados por otros individuos.

   Este algoritmo usa la función fitness sharing para preservar la diversidad de la población.

2. Algoritmo genético de solución no dominada (NSGA):

   Fue propuesto por Kalyanmoy Deb y uno de sus estudiantes de postgrado en 1994. El NSGA se basa en el uso de niveles de dominación propuesta por Goldberg ([10]).

   A los individuos de la población se les asigna un rango de dominación basado en el concepto de dominancia de Pareto, los individuos no dominados con respecto a toda la población son removidos y se les asigna un rango de dominación igual a cero puesto que ningún individuo en la población lo domina. Posteriormente, se obtienen los individuos no dominados de la población restante y se le asigna un rango de dominación igual a uno, mayor a los individuos

del primer nivel. Luego se remueven de la población. El proceso continúa hasta que todos los individuos son clasificados.

En este algoritmo se usa la función fitness sharing para mantener la diversidad de la población.

3. NSGAII:

   Propuesto por Kalyanmoy Deb ([9] es una versión mejorada del NSGA. Este algoritmo realiza el proceso de ordenamiento de las soluciones con una complejidad computacional de $O(MN^2)$, donde M es el número de funciones objetivos y N es el tamaño de la población, mientras que el algoritmo NSGA lo hace en $O(MN^3)$ y más aún no requiere el uso de la función fitness sharing para mantener la diversidad de la población. Este algoritmo se describirá en detalle en el proximo capítulo.

4. Algoritmo genético Niched-Pareto (NPGA):

   Fue propuesto por Jeffry Horn en 1993, este algoritmo difiere de un algoritmo genético en la etapa de selección. En esta etapa se usa la técnica de torneo binario aplicando el concepto de rango de dominación y contador de nicho.

   La técnica del torneo inicia con la selección aleatoria de dos individuos, estos se comparan con otros individuos que estan dentro de una subpoblación de la población total (típicamente el 10 %). Si uno de los individuos domina a todos los individuos de la población, entonces es el ganador. Si lo dominan o ambos individuos son dominados, entoces se aplica el contador de nicho y gana el individuo que este en el nicho de menor densidad.

5. **NPGA II**:

   Usa jerarquización de Pareto pero mantiene la selección mediante torneo del NPGA original. Se usa una nueva función de aptitud en la que se calculan los conteos de nichos usando individuos de la siguiente generación en vez de usar los de la generación actual. A esto se le llama **fitness sharing**.

6. Algoritmo evolucionario Pareto fuerte (SPEAII):

   Eckart Zitzler propuso el SPEAII como un método que integra diferentes algoritmos evolutivos multiobjetivos, buscando combinar lo mejor de cada uno de ellos.

   SPEAII usa un archivo de soluciones no dominadas obtenidas previamente que se conoce como archivo externo. A cada generación se copian individuos no dominados a este archivo. Para cada individuo en este archivo se calcula un valor de fortaleza que es número de soluciones a las que un individuo domina.

La aptitud de cada individuo en la población se calcula de acuerdo a las fortalezas de todos los individuos del archivo externo a los cuales un cierto individuo domine. Adicionalmente, se usa una técnica de truncamiento para mantener un número fijo de individuos en la población.

Capítulo 2

NSGAII (Nondominated sorting genetic Algorithm-II)

Se trata de un algoritmo genético multiobjetivo propuesto por Deb ([9]) es una versión mejorada de su antecesor, el NSGA, desarrollado también por Deb. Básicamente, el NSGAII mejora la versión anterior en tres aspectos fundamentales:

1. Mejora el proceso de ordenamiento de las soluciones no-dominadas; en esta etapa, el NSGA tiene una complejidad computacional de $O(MN^3)$, donde M es el número de objetivos y N el tamaño de la población, por $O(MN^2)$ del NSGAII.
2. Adiciona el elitismo, esto es, las mejores soluciones pasan a la próxima generación.
3. Ausencia de parámetro σ_{shared} para incrementar la variedad de la población, y para el cual no suelen definirse métodos sistematizados de elección.

A continuación se presenta el pseudocodigo:

ALGORITMO NSGAII

1. Dado Nind, Gen /*Donde Nind representa el tamaño de individuos en la población. Gen representa el número de generaciones.
2. Generar población inicial (Po), con un procedimiento aleatorio.
3. Generar descendientes (Qo), usando torneo binario y mutación uniforme.
4. Mientras el número de poblaciones (t) sea menor que Gen,

   4.1 Determinar la población intermedia (Rt) mediante la unión de la población (Pt) y (Qt).

4.2 Generar el conjunto (F) conformado por los diversos frentes (Fi). Ordenando las soluciones según el principio de no dominancia.

4.3 Generar la nueva población.

4.3.1 Mientras el número de individuos en la nueva población ($Pt + 1$) sea menor o igual que Nind.

4.3.1.1 Asignar distancia de agrupamiento (crowding-distance-assignment) a cada uno de los individuos en los diversos frentes (Fi).

4.3.1.2 Determinar la nueva población mediante la unión de los frentes.

4.4 Generar la descendencia ($Qt + 1$) ejecutando OPERADORES GENÉTICOS.

El algoritmo NSGAII parte de una población inicial (Po) que se genera aleatoriamente tomando en cuenta las funciones que conforman el problema multiobjetivo y el rango de variación de las variables involucradas.

Una vez generada esta población se debe determinar los conjuntos de no dominancia.

Enfoque rápido de ordenamiento no-dominado

La población es ordenada en base a los principios de no dominacion donde a cada solución se la asigna un valor de bondad, igual al nivel de su dominación. Una vez obtenida la población inicial se ejecuta el operador de selección de enfoque de rápido ordenamiento, el cual, consiste en ordenar a la población de tamaño N en diferentes niveles de no dominación. El primer nivel o frente de no dominación se obtienen comparando cada uno de los individuos de la población con el resto de la misma para determinar si es o no dominado. Para cada elemento de la población se determinan el valor de dos entidades:

1. **np** que es el número de individuos que dominan al individuo **p**.
2. **Sp** que representa el conjunto que contiene a los individuos que son dominados por **p**.

Todas las soluciones del primer frente no dominado tendrán su cuenta de dominación igual a cero puesto que ninguna otra solución la domina. Por cada solución del primer frente se revisa cada integrante (denominado q) del conjunto Sp y se reduce en uno su grado de dominación. Después de este proceso todos los elementos q cuyos contadores de dominacion alcancen el valor de cero, serán colocados en una lista Q, todos los integrantes de Q pertenecerán al segundo frente no dominado. Ahora se continua con el procedimiento anterior con cada miembro de Q y se forma el tercer frente no dominado. Este procedimiento se mantiene, hasta que todos los frentes son identificados.

Durante el proceso de convergencia hacia el conjunto óptimo de pareto es deseable mantener una gran diversidad de elementos de la población debido a que de esta forma podemos recorrer gran parte del el espacio de búsqueda. El NSGAII usa el enfoque denominado crowded-comparison. Esta técnica no requiere ningún valor definido por el usuario como el NSGA que usa el parámetro shared para incrementar la diversidad de la población. Definiremos antes el concepto de estimación de densidad.

Estimación de densidad

Para obtener una estimación de la densidad de los individuos alrededor de un punto en particular, se calcula la distancia promedio de dos puntos vecinos, uno en cada lado del punto de interés, utilizando el valor de sus funciones objetivos. Esta cantidad, i.distance, sirve como una estimación del perímetro de un cuboide tomando como vértice el punto vecino más cercano, es decir, mide el grado de aislamiento de una solución, así entre más pequeño sea el perímetro del cuboide se tiene una solución con más vecinos lo cual es poco deseable.

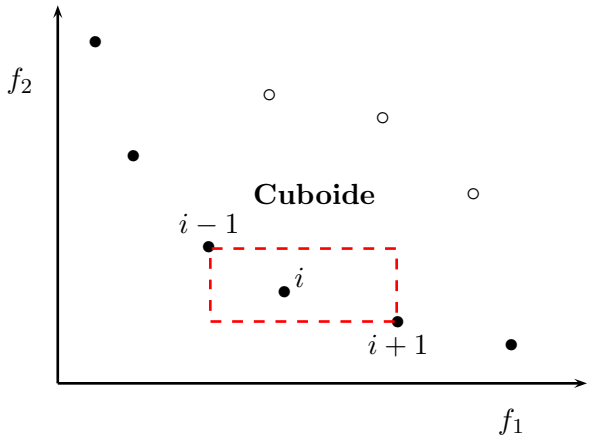


Figura 2.1: Cálculo del crowding distancia

En la Figura 2.1, el valor de crowding distance, del i-ésimo individuo en su frente (marcado con círculos llenos), es la longitud promedio del cuboide que lo contiene. El proceso computacional para hallar la **crowding-distance**, requiere ordenar la población de acuerdo con el valor de cada función objetivo, en orden de magnitud ascendente tantas veces como funciones objetivos tenga. Después, por cada función objetivo se asignan valores de distancia infinitos a los individuos que están en los extremos (individuos con el menor valor objetivo). A todas las demás soluciones, se les asigna una distancia, igual al valor absoluto de la diferencia (normalizada) de valor de las funciones correspondientes a los puntos adyacentes. Estos cálculos se hacen con todas las funciones objetivo. El valor **crowding-distance** total se calcula como la suma de los valores de todas las distancias de los elementos correspondientes a cada objetivo. Se normaliza cada función objetivo antes del cálculo de la **crowding-distance**.

Proceso General del crowding-distance (distancia de agrupamiento)

1. Ordenar a la población en orden ascendente.
2. Asignar valores infinitos a los individuos en los extremos.
3. Asignar al resto de la población la distancia de agrupamiento correspondiente.

2.0.1. Algoritmo para la asignación de distancia de agrupamiento

1. Hacer $l = |I|$ /*Número de soluciones en I (el frente).
2. Para cada i; $I[i]$.distance $= 0$/*se inicializa el contador de distancia.
3. Por cada objetivo m.

   3.1 $I = \text{sort}(I, m)$ /*se ordena usando el valor de cada función objetivo.

   3.2 $I[1]$.distance $= I[l]$.distance $= \infty$ /* Acotamiento para los valores extremos.

   3.3 for $i = 2 \rightarrow (l - 1)$ /*Para todos los puntos.

      3.2.1 $I[i]$.distance $= I[i]$.distance $+ (I[i + 1].m - I[i - 1].m)/(f_m^{\text{máx}} - f_m^{\text{mín}})$.

En el pseudocódigo anterior, $I[i].m$ y $I[i]$.distancia se refieren al valor de la m-ésima función objetivo y el valor de la distancia correspondiente al individuo i-ésimo, en el conjunto **I**. Los parámetros $f_m^{\text{máx}}$ y $f_m^{\text{mín}}$ son el máximo y el mínimo valor de la m-ésima función objetivo. La complejidad computacional de este proceso, está relacionada directamente con el proceso de ordenamiento, ya que se requieren M ordenamientos independientes de a lo más N individuos. Después que todos los miembros del conjunto **I** tienen asignados un valor de distancia se pueden comparar 2 integrantes por su separación o proximidad con otros individuos. Una solución con valor de distancia pequeños tiene más vecinos y por lo tanto es menos deseable.

Operador crowded-comparison

El operador **crowded-comparison** ($\prec_n$) guía el proceso de selección encaminando directamente al algoritmo hacia un frente óptimo de Pareto. Se asume que cada individuo i de la población tiene dos atributos:

1. Jerarquía de no-dominación (i_{rank});
2. Distancia de agrupamiento (i_{distance})

Definición 2.0.1 *Se define un orden parcial* $\prec_n$ *como:*

$$i \prec_n j = \begin{cases} (i_{\text{rank}} < j_{\text{rank}}) & \text{si } (i_{\text{rank}} \neq j_{\text{rank}}), \\ (i_{\text{distance}} > j_{\text{distance}}) & \text{si } (i_{\text{rank}} = j_{\text{rank}}). \end{cases}$$

Lo anterior significa que entre dos soluciones con diferentes rangos de no-dominación, esto es, pertenecen a distintos frentes, se prefiere la solución con el menor rango. En otro caso, si dos soluciones pertenecen al mismo frente, entonces se prefiere la que está ubicada en el área de menor densidad. El orden parcial es importante puesto que en ocasiones un frente no basta para hacer el llenado de la población. Una vez seleccionada la población P_{t+1} de tamaño N (mejores soluciones) mediante el **crowded-comparison** se le aplica los operadores de selección, cruzamiento y mutación para formar una nueva población Q_{t+1} de tamaño N.

Operadores genéticos

Inicialmente se genera un número aletoario entre 0 y 1 para determinar si se efecturá cruce o mutación, esto es debido a que daremos un porcentaje alto para el cruce y uno menor a la mutación , si resulta cruce seleccionamos dos cromosomas padres a los cuales se aplicará cruce genético SBX para obtener dos cromosomas hijos en caso contrario seleccionamos un padre para aplicar mutación polinomial para producir un cromosoma hijo.

A continuación se muestra el pseudocodigo:

OPERADORES GENÉTICOS

1. Dado M, V, l-limit, u-limit, Po y Q /*Donde M es el número de funciones objetivos, V es el número de variables de decisión, l-limit y u-limit son vectores cuyas componentes representan los limites inferior y superior por cada variable, Po representa la matriz población y Q es la matriz formada por los descendientes.

2. Por cada individuo en la matriz población (Po).

   2.1 Generar un número aleatorio r.

   2.2 Si $r <$ 0,9 aplique **CRUCEOG**
   caso contrario aplique **MUTACIÓNOG**

A continuación se presenta los pseudocodigo del proceso de cruce y mutación:

CRUCEOG

1. Dado l-limit,u-limit,Po,Q /*Donde l-limit y u-limit son vectores cuyas componentes representan los limites inferior y superior por cada variable, Po representa la matriz población y Q es la matriz formada por los descendientes.

2. Seleccionar aleatoriamente de Po dos cromosomas padres x^k diferentes.

3. Hacer L =longitud del cromosoma.

4. Haga desde $i = 1$ hasta L.

   a) Genere un número aleatorio.

   b) Aplicar **SBX** para obtener la k-ésima componente del cromosoma hijo x_i^k donde $k = 1, 2$.

   c) Si $x_i^k < L$ entonces $x_i^k = l - limit$
   en caso contrario si $x_i^k > u - limit$ entonces $x_i^k = u - limit$.

5. Actualizar Q con x^k donde $k = 1, 2$.

MUTACIONOG

1. Dado l-limit,u-limit,Po,Q /*Donde l-limit y u-limit son vectores cuyas componentes representan los limites inferior y superior por cada variable, Po representa la matriz población y Q es la matriz formada por los descendientes.

2. Seleccionar aleatoriamente de Po el cromosoma padre x^k.

3. Hacer L =longitud del cromosoma.

4. Haga desde $i = 1$ hasta L.

   a) Genere un número aleatorio.

   b) Aplicar **mutación polinomial** para obtener la k-ésima componente del cromosoma hijo x_i^k donde $k = 1$.

   c) Si $x_i^k < L$ entonces $x_i^k = l - limit$
   en caso contrario si $x_i^k > u - limit$ entonces $x_i^k = u - limit$.

5. Actualizar Q con x^k donde $k = 1$.

Ciclo completo del NSGAII

Inicialmente y de manera aleatoria, se crea una población P_0. Esta población es ordenada en base al principio de no-dominación. A cada solución se le asigna un valor de bondad (o jerarquía), igual al nivel de su dominación (el nivel 1 es el mejor, el nivel 2 es el siguiente mejor y así sucesivamente). En el proceso anterior se aplica la minimización del fitness o valor aptitud. Al principio se aplica selección por torneo; es decir, se escogen dos soluciones al azar que compiten según su grado de dominación y se escoje un ganador, luego se aplican los operadores de cruzamiento y mutación, para crear una población de descendientes Q_0 de tamaño N. El concepto de elitismo se aplica al comparar a la población actual con la mejor población de individuos no-dominados encontrados previamente. El procedimiento difiere después de producir la generación inicial, como se describe a continuación con la i-ésima generación.

Primero se obtiene una población formada por $R_t = P_t \bigcup Q_t$ de tamaño $2N$. Luego, la población R_t es ordenada por niveles de acuerdo a los principios de no-dominación. Así la combinación de las poblaciones previas y actual R_t aseguran el proceso de elitismo. Después de que son ordenadas, las mejores soluciones pertenecen al conjunto F_1 y éstas van a ocupar el papel preponderante durante el proceso. Los siguientes mejores individuos pertenecerán al conjunto F_2 y así sucesivamente hasta formar el último frente F_l. Si el tamaño de F_1 es menor a N, entonces se deben considerar todos sus elementos para formar una nueva población P_{t+1}. Los espacios pendientes en la nueva población serán llenados por los subsiguientes frentes no-dominados. Este procedimiento continúa hasta que se complete la población. Seguramente que la suma de los conjuntos F_1 a F_l es mayor que N. Para solucionar esta situación se ordena en forma descendente el último conjunto F_l usando el operador de crowded-comparison para seleccionar las mejores soluciones y llenar exactamente el espacio de la nueva población de tamaño N. El NSGAII se describe gráficamente en la Figura 2.2. A la nueva población P_{t+1} se le aplican los operadores de cruzamiento y mutación para formar una nueva población Q_{t+1} de tamaño N. Es importante mencionar que en la técnica de selección de torneo binario se aplica el operador crowded-comparison. Este operador requiere que se conozca la jerarquía y la distancia de agrupamiento, de cada solución de la población.

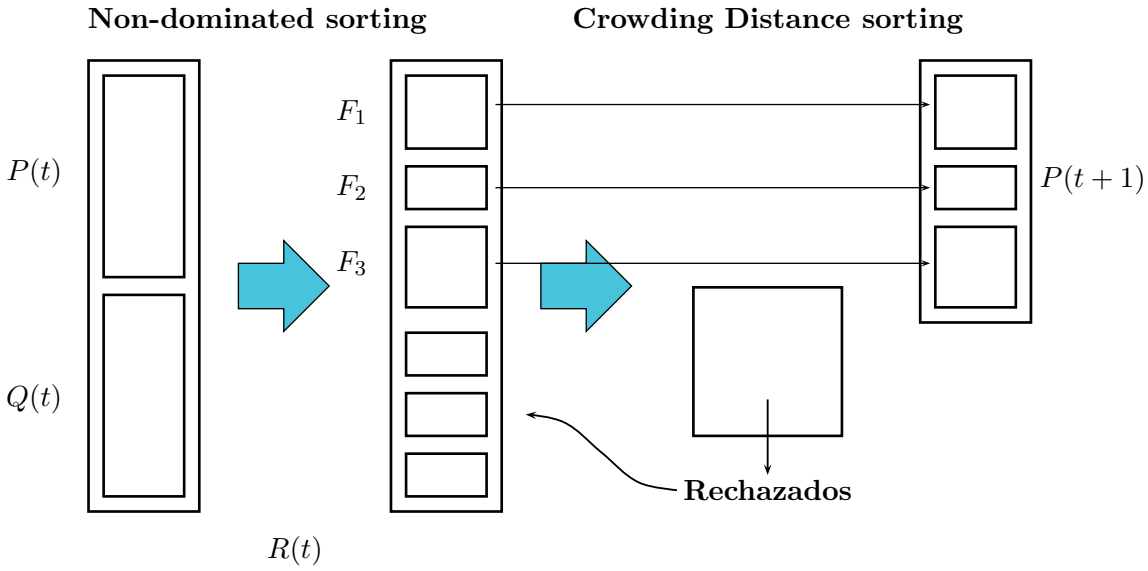


Figura 2.2: Ciclo completo del NSGAII

Capítulo 3

Simulación

La idea principal de este trabajo consistió en aplicar el algoritmo NSGAII, descrito en el capítulo anterior, a algunos de los problemas clásicos de optimización cuya naturaleza no lineal y alta dimensionalidad los convierte en problemas difíciles de resolver, así mismo se aplico el algoritmo NSGAII al problema de ubicar centrales telefónicas en Cabudare estado Lara, estudiando mediante la experimentación numérica el comportamiento del algoritmo, proponiendo algunas variantes y analizando el resultado obtenido.

El conjunto de problemas pruebas, se escogió de tal manera de investigar sistemáticamente los distintos aspectos del algoritmo NSGAII, más aún el conocimiento exacto del frente de Pareto se encuentra disponible (véase [9]). El conjunto de problemas pruebas intento abarcar una amplia gama de características. En general, se trabajo con frentes de Pareto continuos, discontinuos, convexos y no convexos.

Problemas de Zitzler-Deb-Thiele

Zitzler, Deb y Thiele ([11], [12]) diseñaron un conjunto de problemas de diferente complejidad para evaluar los algoritmos multiobjetivos. Estos problemas tienen varias características, pero la principal motivación de su uso la constituye el hecho de poseer frentes de Paretos calculables analíticamente, lo cual permite evaluar de forma precisa la calidad de las soluciones obtenidas por el algoritmo. La formulación de estos problemas se ofrece en la Tabla 3.1.

El problema ZDT1 tiene 30 variables en el rango $[0, 1]$. Su frente de Pareto es convexo, y está determinado por $0 \leq x_1^* \leq 1$ y $x_i^* = 0$, para $i = 2, \ldots, 30$. Este problema tiene un frente de Pareto continuo y su complejidad esta dada por el gran número de variables que maneja.

El problema ZDT2 tiene 30 en el rango $[0, 1]$ su frente de Pareto es no convexo y esta determinado por $0 \leq x_1^* \leq 1$ y $x_i^* = 0$, para $i = 2, \ldots, 30$. Este problema similar al ZTD1 tiene un frente de Pareto continuo y su complejidad esta dada por el gran número de variables que

Nombre	**N°**	**Dominio**	**Planteamiento del problema**	**Función** $g(x)$
ZTD1	30	$0 \leq x_i \leq 1$	Minimizar $(f_1(\mathbf{x}), f_2(\mathbf{x}))$ donde: $f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \left(\sqrt{\frac{x_1}{g(\mathbf{x})}}\right)\right)$	$1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$
ZTD2	30	$0 \leq x_i \leq 1$	Minimizar $(f_1(\mathbf{x}), f_2(\mathbf{x}))$ donde: $f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \left(\frac{x_1}{g(\mathbf{x})}\right)^2\right)$	$1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$
ZTD3	30	$0 \leq x_i \leq 1$	Minimizar $(f_1(\mathbf{x}), f_2(\mathbf{x}))$ donde: $f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})h(\mathbf{x})$, con $h(\mathbf{x}) = 1 - \sqrt{\frac{x_1}{g(\mathbf{x})}} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x_1)$	$1 + \frac{9}{N-1}\sum_{i=2}^{N} x_i$

Tabla 3.1: Problemas prueba

maneja.

El problema ZDT3 tiene 30 variables en el rango $[0, 1]$. Su frente de Pareto es discontinuo, y está determinado por $0 \leq x_1^* \leq 1$ y $x_i^* = 0$, para $i = 2, \ldots, 30$.

Problemas de Viennet

Viennet ([9], [11]) presenta un conjunto de problemas con tres funciones objetivos, de los cuales en este trabajo se consideran dos problemas cuya formulación se ofrece en la Tabla 3.2.

Nombre	**N°**	**Dominio**	**Planteamiento del problema**
VNT2	2	$-4 \leq x_i \leq 4$	Minimizar $(f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))$ donde: $f_1(\mathbf{x}) = \frac{(x_1-2)^2}{2} + \frac{(x_2+1)^2}{13} + 3$ $f_2(\mathbf{x}) = \frac{(x_1+x_2-3)^2}{36} - \frac{(-x_1+x_2+2)^2}{8} - 17$ $f_3(\mathbf{x}) = \frac{(x_1+2x_2-1)^2}{175} - \frac{(-x_1+2x_2)^2}{17} - 13$
VNT3	2	$-3 \leq x_i \leq 3$	Minimizar $(f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))$ donde: $f_1(\mathbf{x}) = \frac{(x_1^2-x_2^2)}{2} + \sin(x_1^2+x_2^2)$ $f_2(\mathbf{x}) = \frac{(3x_1-2x_2+4)^2}{8} - \frac{(x_1-x_2+1)^2}{27} + 15$ $f_3(\mathbf{x}) = \frac{1}{(x_1^2+x_2^2+1)} - 1{,}1e^{(-x_1^2-x_2^2)}$

Tabla 3.2: Problemas prueba

Los problemas VNT2 y VNT3 tienen dos variables de decisión en los rangos $[-4, 4]$ y $[-3, 3]$, respectivamente. Sus frentes de Pareto son discontinuo en VNT2 y no convexo en VNT3, su importancia es más que nada la observación de frentes de Pareto con tres objetivos.

Durante la aplicación se planteó la posibilidad del estudio de funciones no contradictorias, en estas los padres destinados al cruce coinciden a medida que el número de generaciones aumenta

Nombre	**N**	**Dominio**	**Planteamiento Del Problema**
	1	$-4 \leq x \leq 4$	Minimizar $(f_1(\mathbf{x}), f_2(\mathbf{x}))$ donde: $f_1(\mathbf{x}) = x^2$ $f_2(\mathbf{x}) = x^2$

Tabla 3.3: Problema prueba

y en el algoritmo implementado este fenómeno causaba estructuras ciclicas indefinidas lo cual no permitia el estudio de problemas tan simples como el presentado en la Tabla 3.3.

En el paso [2] del algoritmo CRUCEOG descrito en el capítulo anterior se eliminó la condición que seleccionaba solo padres diferentes para el cruce genético resultando ser una modificación significativa en la estructura del algoritmo implementado, pues en este caso al aplicar el algoritmo NSGAII es posible encontrar el vector ideal para funciones no contradictorias sin afectar la diversidad de la población. Mas aún se realizarón cambios en la estrutura de los operadores genéticos en el paso [4.b] del algoritmo CRUCEOG original, realizando pruebas con la técnica de cruce BLX-0.5 y en el paso [4.b]del algoritmo MUTACIONOG se cambio la mutación polinomial por la uniforme para representación real este cambio se muestra en el NSGAII.v1 y la técnica de cruce el BLX-0.5 y mutación gaussiana también para representación real en NSGAII.v2.

Las pruebas numéricas se efectuaron con los siguientes parámetros:

Número de generaciones	500
Tamaño de la población	100
Prob. de cruce	0.9
Prob. de mutación	0.1
Parámetro de mutación (nm)	20
Parámetro de cruce (nc)	20

Y los tiempos promedio de ejecución para las versiones NSGAII, NSGAII.v1 y NSGAII.v2

se muestran a continuación:

Tiempo promedio de ejecución en centésimas de segundos

Problema prueba	NSGAII	NSGAII.v1	NSGAII.v2
ZDT1	235.953	284.938	269.922
ZDT2	284.734	286.313	268.859
ZDT3	266.422	350.688	340.031
VNT2	236.344	297.016	300.828
VNT3	239.031	303.547	308.031

Observe que en la versión original del NSGAII el tiempo de ejecución es menor que en las versiones NSGAII.v1 y NSGAII.v2, sin embargo, la calidad de las soluciones es mucho mejor en estas últimas versiones como puede observarse a continuación:

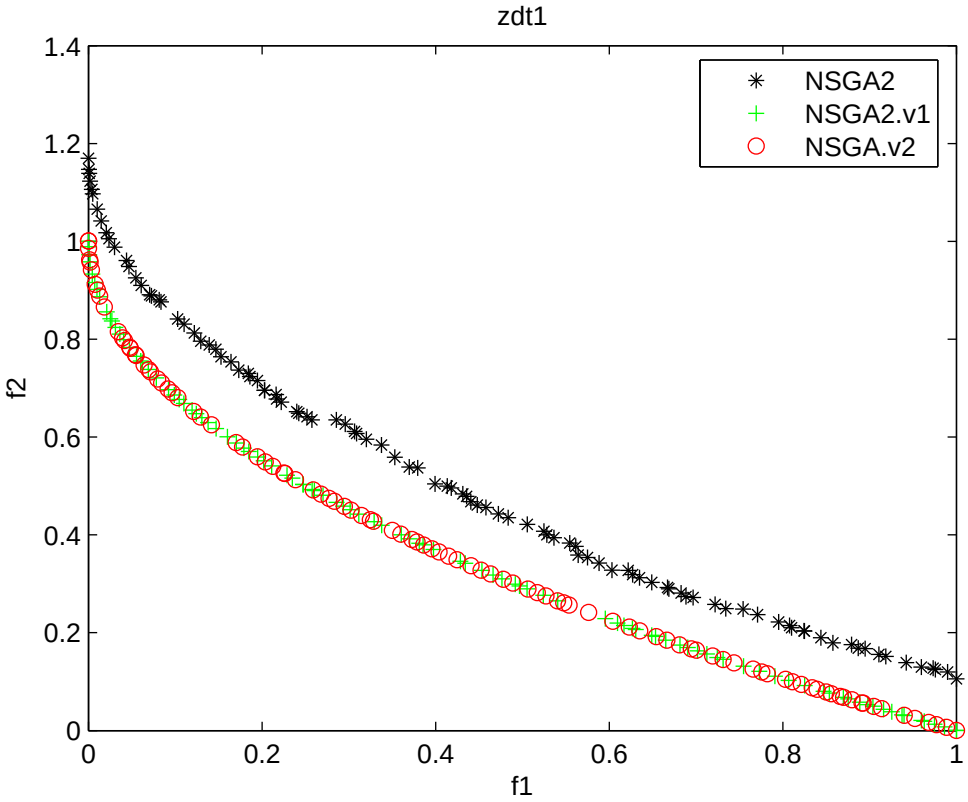


Figura 3.1: Problema prueba ZDT1

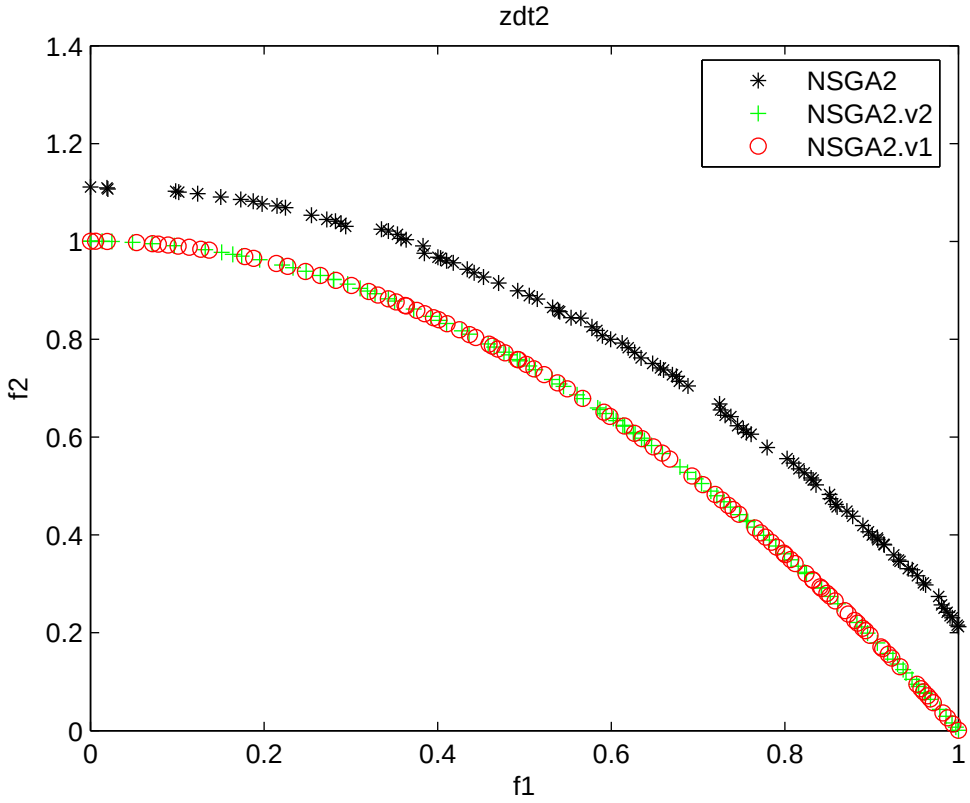


Figura 3.2: Problema prueba ZDT2

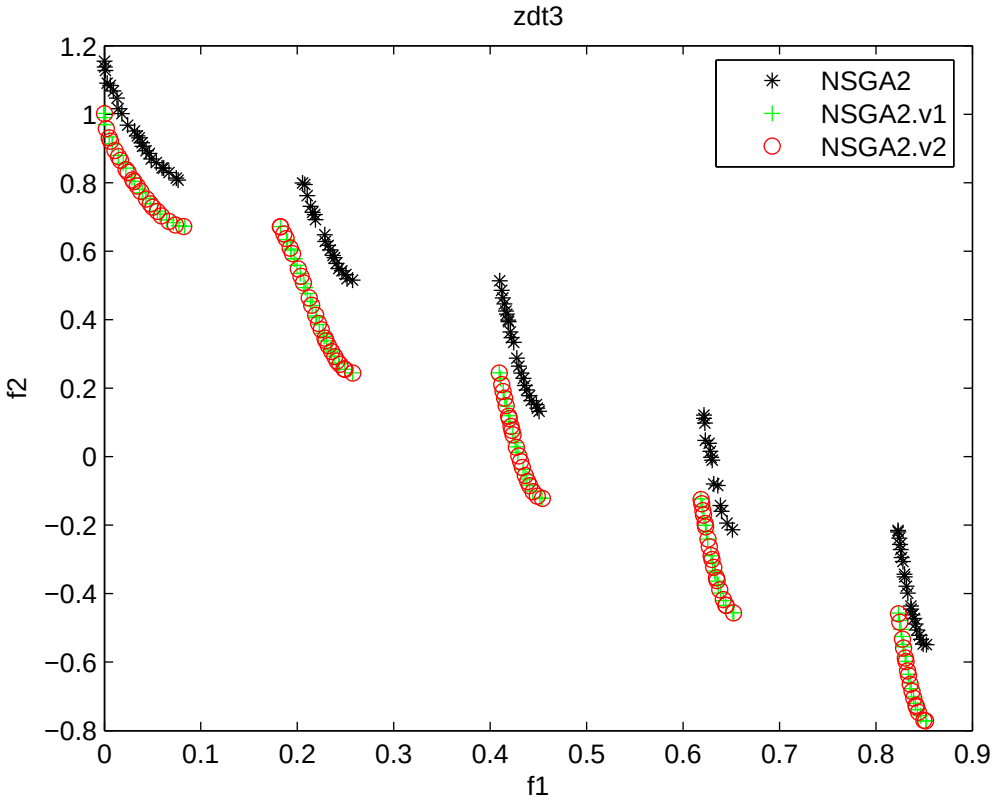


Figura 3.3: Problema prueba ZDT3

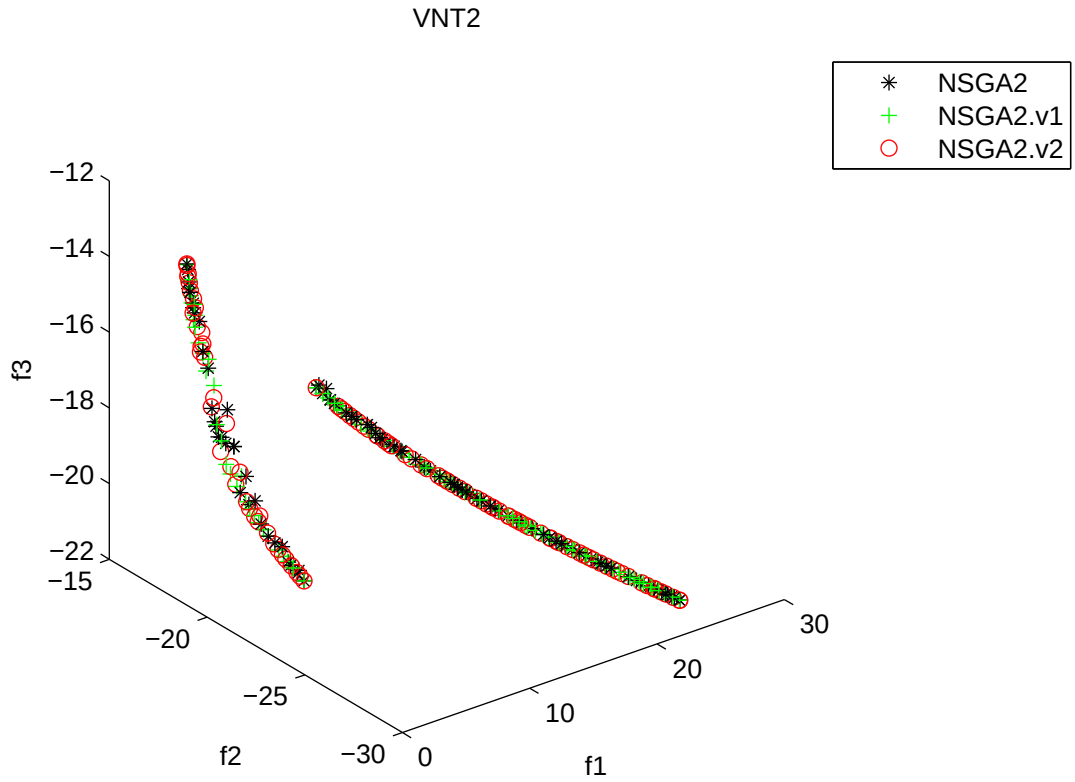


Figura 3.4: Problema prueba VNT2

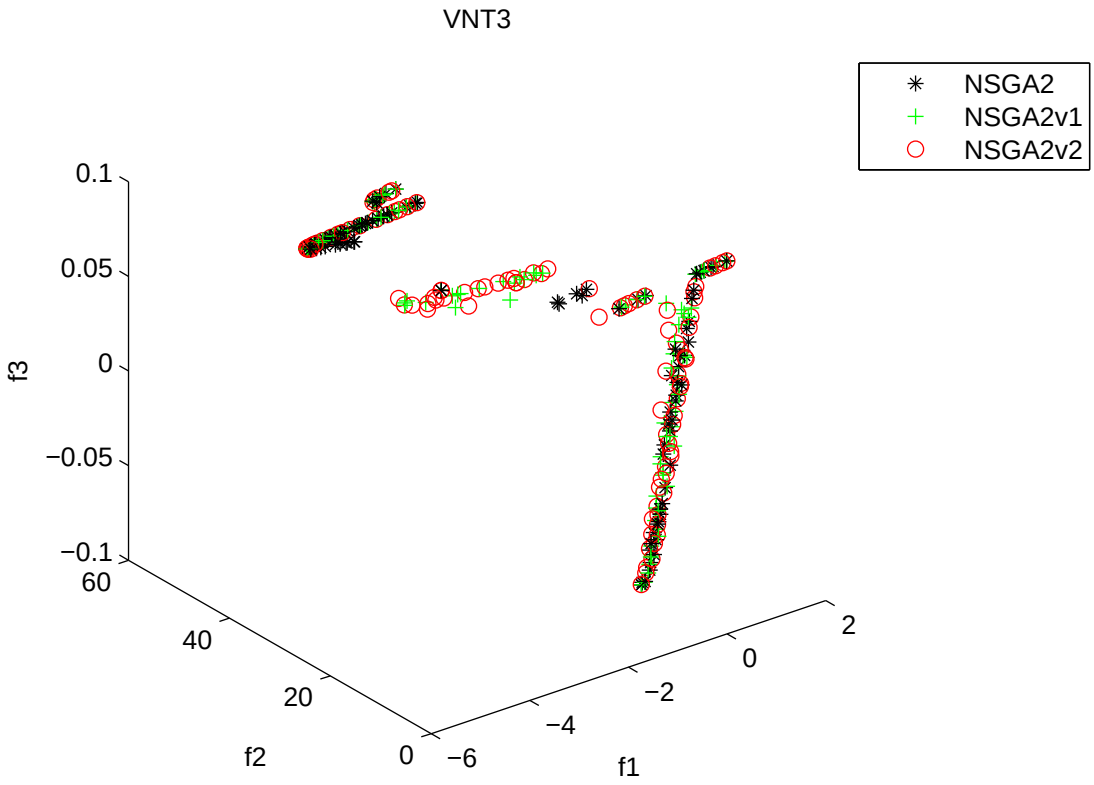


Figura 3.5: Problema prueba VNT3

Observe que para los problemas ZDT1, ZDT2 y ZDT3 cuya formulación se ofrece en la Tabla 1 la calidad de las soluciones es mucho mejor para las versiones NSGAII.v1 y NSGAII.v2 que para la versión NSGAII, esto nos induce a pensar que debido a la naturaleza no lineal de los problemas (cocientes de funciones, funciones exponenciales y otros) la variante en la técnica de cruce es significativa ya que no solo constituye una buena aproximación al frente de Pareto real sino que se aproxima en menos iteraciones. Es importante mencionar que la versión original NSGAII se aproxima al frente de Pareto real con más de 500 generaciones lo cual nos lleva a pensar que la técnica de cruce BLX resulta ser más eficiente que el SBX aplicado en la versión original. En los problemas VNT2 y VNT3 (ver Tabla 3.2) el NSGAII, NSGAII.v1 y NSGAII.v2 tienen un comportamiento similar pero es importante señalar que en este caso las funciones son menos complejas que en los casos ZDT1, ZTD2 y ZTD3 y el número de variables es menor.

El operador de cruce juega un papel central en los algoritmos evolutivos. De hecho puede considerarse como una de las características que definen a estos algoritmos y es uno de los componentes a tener en cuenta para mejorar su eficiencia.

Problema de Almeida-Amarilla-Barán (AAB)

Almeida, Amarilla y Baran ([2]) diseñaron un problema para la optimización multiobjetivo en la planificación de centrales telefónicas en la ciudad de la Asunción en Paraguay, este problema es tratado en este trabajo, y aplicado en Cabudare estado Lara, Venezuela. A continuación se define el problema de la siguiente forma:

Minimizar

$$y = (\sum_{i=1}^{6} c_i(x), \sum_{i=1}^{6} c_i(x), \sum_{i=1}^{6} c_i(x))$$

donde

$$x \in X \subset \mathbb{R}^n$$

$$0 \leq x \leq m$$

n ... número máximo de centrales;

m ... número máximo de cuadrículas en que se divide el área de estudio;

x_i ... designa la ubicación de una central dentro del área en estudio ($0 \leq x_i \leq m$)

y_i ... representa la inversión acumulada hasta el año considerado.

donde:

$c_1(x)$: costo total de planta externa, definidas por el vector de decisión x.

$c_2(x)$: costo de terrenos donde serán instaladas las centrales.

$c_3(x)$: costo de edificios donde serán instaladas las centrales.

$c_4(x)$: costo de ingeniería que conlleva la instalación de las centrales;

$c_5(x)$: costo de equipos de conmutación;

$c_6(x)$: costos de equipos de transmisión entre las centrales definidas por x.

Un vector de decisión, que representa una solución, se establece de la siguiente forma: $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ este vector contiene como máximo n centrales, $x_i \geq 0$ indica en que ubicación de la matriz tenemos una central, $x_i = 0$ indica que no existe central.

Para la evaluación del costo de planta externa $c_1(x)$, se calcula las distancias de cada abonado a la central más cercana.

Problema Prueba

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1				1	2	3	4	5				
2				6	7	8	**9**	10	11	12		
3		13	14	15	16	17	18	19	20	21	22	
4		23	24	25	26	27	28	29	30	31	32	33
5	34	35	36	**37**	38	39	40	41	42	43	44	45
6		46	47	48	49	50	51	52	53	54	**55**	56
7			57	58	59	60	61	62	63	64	65	66

Tabla 3.4: Cabudare

En la tabla 3.4 se muestra al municipio Palavecino y las parroquias Agua Viva (zona en azul), Cabudare (zona en rojo) y José Gregorio Bastida (zona en amarillo). El vector de decisión para este caso es $\mathbf{x} = [9, 37, 55]$, en donde se ve en el mapa de Cabudare Figura (3.4) que las centrales están ubicadas en las posiciones $9, 37$ y 55 respectivamente.

A continuación se hace un procedimiento para calcular cuales son las cuadrículas que pasarán a formar parte del área de servicio de una central. En cada una de las cuadrículas existe una demanda conocida.

Las cuadrículas que formarán parte del área de servicio de una central, son aquellas que tienen el costo mínimo de conexión cuando están conectadas a dicha central. Cada una de estas cudrículas denotadas en adelante x_t, atiende a la condición: $1 \leq x_t \leq m$. A cada cuadrícula x_t van asociados dos valores que representan sus coordenadas (X_t, Y_t) en una matriz de 7 filas y 12 columnas. El cálculo del costo de conectar los abonados que están en una cuadrícula x_t a una central, se realiza conforme a:

$$c_{it} = d_t p(|X_i - X_t| + |Y_i - Y_t| + 1)$$

donde:

c_{it} ... costo de conectar los abonados pertenecientes a la cuadrícula x_t a la central x_i;

d_t ... cantidad de abonados de la cuadrícula x_t;

p ... costo de planta externa por unidad de longitud,por cada abonado;

(X_i, Y_i) ... coordenadas de la central x_i;

(X_t, Y_t) ... coordenadas de la cuadrícula x_t.

El área de servicio de cada central contiene aquellas cuadrículas con menores distancias a dicha central, de forma a minimizar el costo de planta externa $c_1(x)$. Por lo tanto, el costo total de planta externa, para el ejemplo considerado, se calcula conforme:

$$c_1(x) = \sum_{i=1}^{n} \sum_{t=1}^{m} c_{it} h_{it}$$

$$h_{it} = \begin{cases} 1 & \text{si el sitio } x_t \text{ está conectado a la central } x_i \\ 0 & \text{en caso contrario} \end{cases}$$

Se asume que cada sitio x_t puede estar conectado a una central x_i, por lo tanto:

$$\sum_{i=1}^{n} h_{it} = 1$$

con $x_t = 1, 2, \ldots, m$.

El costo del terreno es el producto del costo por m^2 y el área del edificio de la central, conforme:

$$c_2(x) = \sum_{i=1}^{n} q_i g_i w_i$$

q_i ... área en m^2 a ser ocupada por la central x_i.

g_i ... costo del terreno por m^2 en la cuadrícula x_i.

$$w_i = \begin{cases} 1 & \text{si la central está en el sitio } x_i \\ 0 & \text{en caso contrario} \end{cases}$$

Los demás costos de la ecuación, se calcularon de la siguiente forma:

$$c_3(x) = \sum_{i=1}^{n} c_e w_i$$

$$c_4(x) = \sum_{i=1}^{n} c_{ing} w_i$$

$$c_5(x) = \sum_{t=1}^{m} c_{eq} d_t$$

$$c_6(x) = \frac{c_{tr} k(k-1)}{2}$$

$$k = \sum_{i=1}^{n} w_i$$

donde:

c_e ... costo de construcción del edificio de la central;

c_{ing} ... costo de la ingeniería de planifcación de centrales;

c_{eq} ... costo de los equipamiento de la central, por abonado;

c_{tr} ... costo de equipamento de transmsión;

k ... cantidad de centrales ($k \leq n$);

d_t ... demanda telefónica de la cuadrícula x_t.

De esta forma cada costo de inversión y_j para el año considerado será:

$$y_j = \sum_{i=1}^{n} \sum_{t=1}^{m} d_t p(|X_i - X_t| + |Y_i - Y_t| + 1) h_{it} + \sum_{i=1}^{n} q_i g_i w_i +$$

$$\sum_{i=1}^{n} c_e w_i + \sum_{i=1}^{n} c_{ing} w_i + \sum_{t=1}^{m} c_{eq} d_t + \frac{c_{tr} k(k-1)}{2}$$

Observar que la ubicación de la central, depende del costo de conectar los abonados de una determinada cuadrícula a la central, ya que los cinco últimos costos que son valores contantes tendrán un papel preponderante, solo si tienen valores númericos muy grandes. Esto es, la función de inversión depende de la demanda en la cuadricula y la distancia a la central. Más aún al considerar el mismo objetivo no habrá contradicción entre las funciones lo cual permitirá al algoritmo encontrar un único vector solución, que minimice el problema (vector ideal), cabe destacar que en esta situacón el algoritmo ubica la central telefónica en la cudrícula 28 que representa el centro geometrico de Cabudare, muy próxima a la ubicación actual (cuadrícula 18). A medida que comenzamos a variar la demanda por cuadrícula en cada objetivo, creamos situaciones conflictivas y esto es posible debido a que si consideramos el hecho de que Cabudare es una zona en expansión debido a que el municipio palavecino cuya capital es Cabudare vincula los corredores viales y comerciales entre el occidente y el oriente del Estado Lara, representando un importante receptor y distibuidor de productos comestibles de origen agrícola hacia otras regiones dentro y fuera del estado más aún desarrollo urbanístico ha generado, por parte de la población asentada, una gran presión para demandar servicios básicos y especializados, de acuerdo al Consejo Municipal de Palavecino, se estima alrededor de 2000 establecimientos comerciales y de servicios, incluyendo franquicias, cadenas comerciales de alimentos, comidas rápidas, farmacias, etc; el municipo cuenta con los servicios públicos básicos mientras que los otros servicios, son abastecidos desde la ciudad de Barquisimeto, Estado Lara. En consecuencia, el problema principal a ser resuelto consiste en encontrar la cantidad de centrales y la ubicación óptima de estas centrales en el área de estudio. Si existen m sitios posibles, existen 2^m alter-

nativas de ubicación de centrales. Aún, si se retringe la atención para ubicar n centrales en m sitios, el número de ubicación de centrales es:

$$\begin{pmatrix} m \\ n \end{pmatrix} = \frac{m!}{(m-n)!n!}$$

En el ejemplo para 66 cuadrículas validas y 3 centrales, existen unas 45760 alternativas de ubicación de centrales.

Para calcular la cantidad óptima de centrales, se debe aplicar la función de costo para todas las situaciones posibles de tal forma que el cálculo de costo $y_1(x)$, $y_2(x)$, $y_3(x)$ sean mínimos para cada uno de los objetivos.

Debido a la complejidad del problema, y la cantidad de calculos necesarios, se optó por realizar la búsqueda de soluciones utilizando algoritmos evolutivos multiobjetivos, en particular para este trabajo se usó el NSGAII por ser una herramienta de dominio público, disponible en la página central de MATLAB creado por Aravind Seshadri y citada en múltiples reportes de investigación evolutiva multiobjetivo, debido a su reconocida eficiencia (véase [9][11][13]).

El algoritmo fue corrido bajo el sistema operativo Windows-XP de Microsoft, para que pudiera ejecutarse junto con Matlab en el mismo entorno.

Sin embargo durante la implementación númerica se realizarón algunas modificacones en los pasos [2] y [4.4] del NSGAII. A continuación se presenta el pseudocodigo del NSGA.v3:

ALGORITMO NSGAII.v3

1. Dado Nind, Gen /*Donde Nind representa el tamaño de individuos en la población. Gen representa el número de generaciones.
2. Generar población inicial (Po) con un procedimiento heurístico.
3. Generar Descendientes (Qo) usando torneo binario y mutación uniforme.
4. Mientras el número de poblaciones (t) sea menor que Gen
   - 4.1 Determinar la población intermedia (Rt) mediante la unión de la Población Pt y Qt.
   - 4.2 Generar el conjunto (F) conformado por los diversos frentes (Fi). Ordenando las soluciones según el principio de no dominancia.
   - 4.3 Generar la nueva población
     - 4.3.1 Mientras el número de individuos en la nueva población ($Pt + 1$) sea menor o igual que Nind.

4.3.1.1 Asignar distancia de agrupamiento (crowding-distance-assignment) a cada uno de los individuos en los diversos frentes (Fi).

4.3.1.2 Determinar la nueva población mediante la unión de los frentes

4.4 Generar la descendencia (Qt) aplicando OPERADORES GENÉTICOS.

En general el algoritmo NSGAII.v3 tiene la misma estructura que el NSGAII implementado por Seshadri, pero para el caso de Cabudare la población es generada con un procedimiento heurístico (paso [2] del NSGAII) descrito por Baran (véase [2]). A continuación se presenta el método:

Población inicial

La población inicial, cuyo tamaño se denotará como *nind* (número de individuos), es generada por un algoritmo heurístico de inicialización, en donde *nmax* indica el número máximo de centrales para cada vector de decisión. Este algoritmo genera una población inicial de forma inteligente de manera de obtener individuos que se aproximen al conjunto de soluciones Pareto óptimas. Para cada individuo de la población, se realiza un sorteo para saber cuantas centrales tendrá esa solución, y se ubican las centrales de tal forma que las mismas se encuentren en los centros de demandas a fin de minimizar los costos de conexión de los abonados a su central correspondiente. El algoritmo de inicio se describe a continuación:

ALGORITMO HEURÍSTICO DE LA POBLACIÓN INICIAL

1. Dado Nind, nmax /*Donde Nind representa el tamaño de individuos en la población. nmax representa el número de centrales.
2. Ordenar la población de acuerdo al número de habitantes.
3. Mientras no se satisfaga el critero de parada.

   3.1 Generar un número aleatorio **N** entre 1 y nmax.

   3.2 Dividir a la población total en **N** partes.

   3.2.1 Elegir una cuadrícula x_i entre las ubicaciones más pobladas.

   3.2.2 Encontrar la distancia euclideana de x_i a todas las ubicaciones de la matriz población.

   3.2.3 Ordenar las distancias en orden creciente.

   3.2.4 Inicializar **población**.

   3.2.4.1 Mientras no se satisfaga el criterio de parada.

4.3.4.2 Sumar a **población** la población de las ubicaciones más próximas a x_i.

3.2.5 Eliminar de la matriz población las ubicaciones que se agregaron a población.

3.2.6 Hallar el centro geométrico P_i de todas las ubicaciones que se agregaron a población.

3.2.7 Hacer la central x_i igual al centro geométrico P_i.

4. Eliminar centrales repetidas de cada individuo de la población inicial y ordenar las centrales.

Mas aún se realizarón cambios en la estrutura de los operadores genéticos en el paso [4.b] del algoritmo CRUCEOG original, realizando pruebas con la técnica de cruce BLX-0.5 y en el paso [4.b]del algoritmo MUTACIONOG se cambio la mutación polinomial por la uniforme para representación real este cambio se muestra en el NSGAII.v3 y la técnica de cruce el BLX-0.5 y mutación gaussiana también para representación real en NSGAII.v4.

En las corridas realizadas del algoritmo NSGAII, NSGAII.v3 y NSGAII.v4 se utilizarón los siguientes parámetros de entrada:

Número de generaciones	500
Tamaño de la población	100
Número máximo de centrales	3
Prob. de cruce	0.9
Prob. de mutación	0.1
Parámetro de mutación (nm)	20
Parámetro de cruce (nc)	20

En la experimentacón se usaron los siguientes costos fijos en dolares americanos:

Costo de planta externa por Km (p)	70
Costo de construcción de edificio (c_e)	70000
Costo de ingeneria (c_{ing})	30000
Costo fijo de equipos de conmutación por cada 1000 abonados (c_{eq})	585000
Costo de transmisión se enlace (c_{tr})	100000

A continuación se presenta los experimentos realizados con el problema de Almeida, Amarilla y Barán aplicado en Cabudare con la finalidad de mostrar el comportamiento del frente de Pareto, cabe destacar que los costos de terreno empleados no fueron variados en los tres objetivos además

las situaciones que se presentan son ficticias, esto es debido a que al considerar los costos a corto, mediano y largo plazo lo único que se vario fue la demanda por objetivo.

En la Figura 3.6, se puede observar como el *vector ideal* es alcanzable cuando las funciones objetivos no son contradicctorias, es decir, que para nuestro caso particular las zonas con menos demanda relativa en la actualidad, permanece con menos demanda relativa a mediano y largo plazo, mas aún, en este caso la central telefónica fue ubicada en la cuadrícula 28 que es el centro geométrico de Cabudare, muy próxima a la ubicación actual de la central en Cabudare que es en la cuadrícula 18.

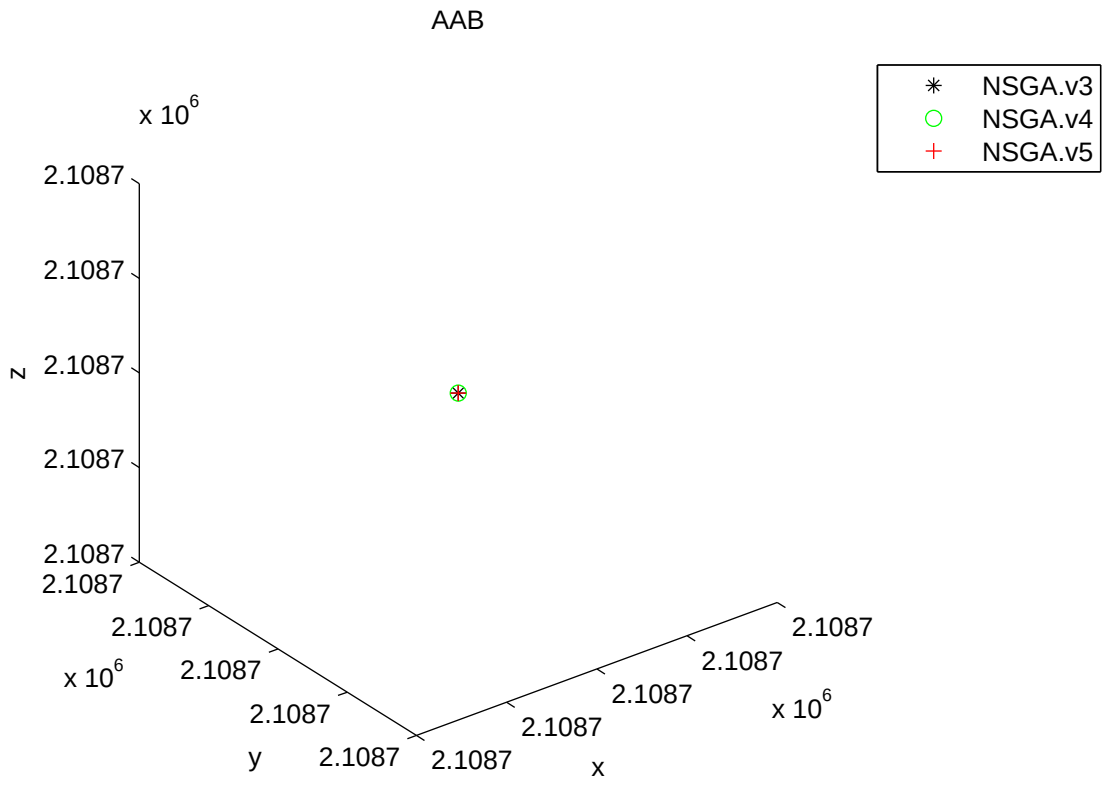


Figura 3.6: Cabudare demanda fija

Al aumentar la demanda en una cuadrícula específica para un año considerado, el algoritmo tenderá a ubicar la central en cuadrículas cercanas a esta y al centro geométrico en búsqueda de una minimización de costos. A medida que la demanda comienza a cambiar entre un plazo y otro(por ejemplo,una zona con mucha demanda relativa a corto plazo a largo plazo tendrá una demanda menor relativa), las funciones son mas contradicctorias y por tanto el frente de pareto requiere mas de un vector solución para la minimización de los objetivos. De esta forma obtenemos frentes de pareto como en la figura 3.7.

A continuación se muestran los datos suministrados por la alcaldía de Palavecino Dirección de Planificación y Desarrollo Urbano, en los cuales se observa la población de Palavecino para

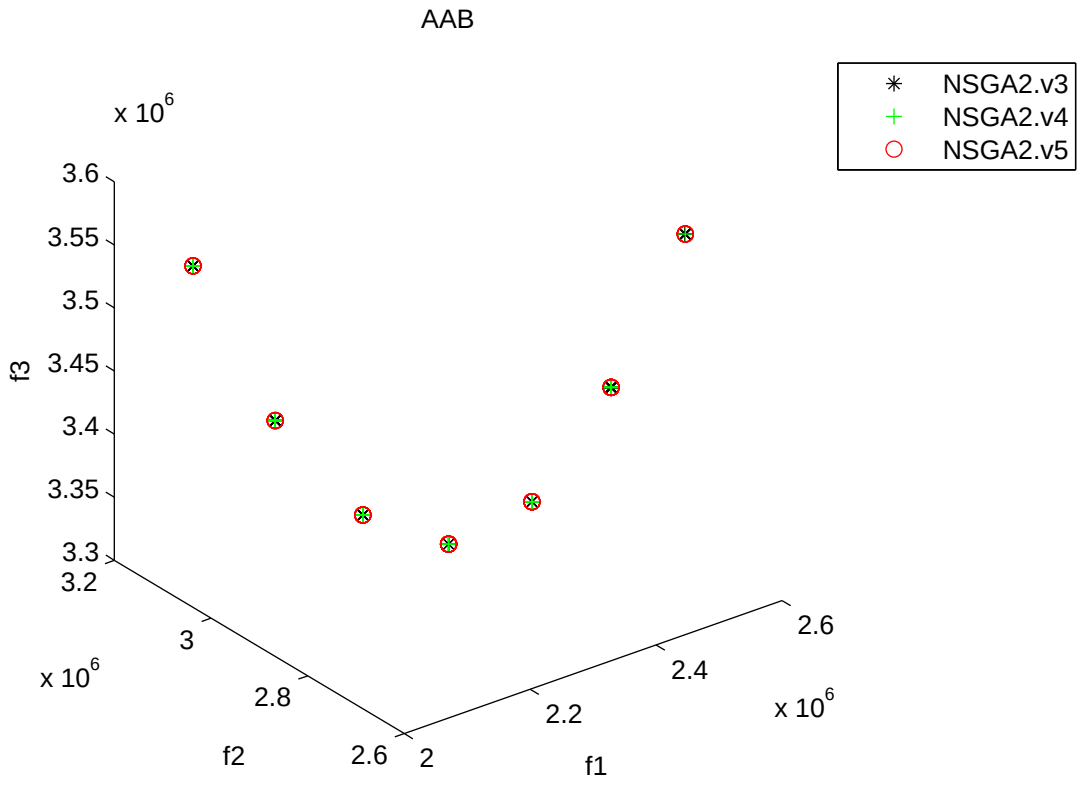


Figura 3.7: Cabudare demanda variable

los años 1981, 2008 y proyecciones para el año 2024.

Años	1981	2008	2024
Parroquias			
Cabudare	59671	64247	10000
José Gregorio Bastidas	61439	76969	122500
Agua Viva	13290	17528	27500

Figura 3.8: Población del Municipio Palavecino por parroquia

Tomando en cuenta estas proyecciones de demanda se generaron demandas aleatorias a corto, mediano y largo plazo manteniendo proporcionalmente las ratas de crecimiento de la población de acuerdo a los datos mostrados en la tabla 3.8. Así la demanda se simuló para las tres parroquias generando un número aleatorio uniformente de la siguiente manera:

La ejecución del algoritmo dio como resultado las siguientes ubicaciones: $\mathbf{x} = [0, 0, 29]$ y $\mathbf{x} = [0, 0, 54]$. Observamos que la ubicación actual de la central telefónica no es la mejor alternativa, sin embargo es una opción viable debido a que es cercana a las cuadriculas óptimas 28 y 29 pero si aumentamos la demanda en cuadrículas como las 56,65 y 66 el algoritmo tenderá a

Parroquias	Corto plazo	**Mediano plazo**	**Largo plazo**
Agua viva	$0 \leq r \leq 13{,}290$	$0 \leq r \leq 17{,}580$	$0 \leq r \leq 27{,}500$
Cabudare	$0 \leq r \leq 59{,}671$	$0 \leq r \leq 64{,}247$	$0 \leq r \leq 100{,}000$
José Gregorio Bastidas	$0 \leq r \leq 61{,}439$	$0 \leq r \leq 76{,}969$	$0 \leq r \leq 122{,}500$

ubicar la central en cuadrículas mas cercanas a las cuadrículas anteriores. Las soluciones del problema AAB proponen ubicar solo una central en Cabudare cercana al centro geométrico y a las cuadrículas de mayor densidad de población.

Esto nos lleva a pensar que la ubicación de la central telefónica cuadrícula 18, donde se encuentra actualmente pertenece a la parroquia de Cabudare y se puede observar en la tabla (3.8) que las proyecciones para el año 2024 indican que la Parroquia José Gregorio Bastidas tendrá una mayor densidad de población por lo cual seria conveniente una ubicación de central telefónica en esta área lo cual es razonable debido a que la zona tiene una alta zona de expansión.

Conclusiones

1. EL Algoritmo NSGAII puede emplearse en la ubicación de cualquier recurso en un área geográfica determinada.

2. El operador de cruce juega un rol crucial en la eficiencia del algoritmo, al encontrar el conjunto solución. Las versiones NSGAII.v1 y NSGAII.v2 usan como operador de cruce el BLX-α, mostrando ser en la practica más eficiente que el NSGAII que usa el SBX como operador de cruce.

3. El efecto de variar el operador de mutación no mostró cambios significativos respecto al número de iteraciones requeridas por el algortimo NSGAII para encontrar el frente de Pareto.

4. En la experimentación númerica se observo que cuando los objetivos no son contradictorios el vector ideal es alcanzable, más aún coincide con el enfoque clásico donde se obtiene una solucón unica.

5. El uso de este tipo de modelo así como la aplicación del algoritmo NSGAII puede ser útil en la planificación de los servicios del estado.

6. Sería interesante estudiar el comportamiento del algoritmo NSGAII para la uicación de las celdas de telefonía celular.

Bibliografía

[1] Agrawal S., Deb K., Meyarivan T. and Pratap Amrit. *A Fast Elitist Multiobjective Genetic Algorithm NSGAII.* IEEE Transactions on Evolutionary Computation, Vol 6, (2002).

[2] Almeida C. y Barán B. *Optimización Multiobjetivo en la planificación de las centrales telefónicas.* Universidad Nacional de Asunción. San Lorenzo, Paraguay, (2003)

[3] Arroyo J. e Armentano V. *Um Algoritmo Genetico para Problemas de Otimizacao Combinatoria Multiobjetivo.* XXXIII Simposio Brasileriro de Pesquisa Operacional. Campos do Jordao-SP: Noviembre, (2001).

[4] Barán B. and Duarte S. *Multiobjetive Network Design Optimization using Parallel Evolutionary Algorithms.* Centro nacional de Computación, Universidad Nacional de Asunción. San Lorenzo, Paraguay. Agosto, (2002).

[5] Bagchi T. and Deb K. *Calibration de GA parameters: The design of experiments approach. Computer Science and informatics.* 26(4): 46-59, (1996).

[6] Bhusan R., Deb K. *Simulated Binary Crossover for continuous Search Space.* Kanpur India, (1994)

[7] Coello C. *Introducción a la computación Evolutiva.* San Zacatenco, México, (2006).

[8] Cooper L. and Steinberg D. *Methods and Appof Linear Programming.* Saunders, Philadephia, (1974).

[9] Deb K. *Multi-objetive Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, (2001).

[10] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, (1989).

[11] Nesmachnow S. *Una versión paralela del Algoritmo Evolutivo para optimización Multiobjetivo NSGAII y su aplicación al diseño de redes de comunicaciones confiables.* Centro de

cálculo, Instituto de Computación Facultad de Ingeniería. Universidad de la República, Uruguay, (2003).

[12] Pradyumn K. and Deb K. *On Finding Multiple Pareto-Optimal Solutions Using Classical and Evolutionary generating Methods.* Disponible en linea htt:/www.iitkac.in/Kandal/pub.html.

[13] Ziztzler E., Laumanns M., and Thiele L. *SPEAII: Improving the Strength Pareto Evolutionary Algorithm, technical Report 103.* Computer Engineering and Networks Laboratory, Swiss Federal Institute of Tecnology. Zurich, Switzerland, (2001).