Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Solución de problemas de programación disyunta mediante optimización metaheurística

por

Perla Cecilia Hernández Lara

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León septiembre 2011

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Solución de problemas de programación disyunta mediante optimización metaheurística

por

Perla Cecilia Hernández Lara

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León septiembre 2011

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Solución de problemas de programación disyunta mediante optimización metaheurística», realizada por la alumna Perla Cecilia Hernández Lara, con número de matrícula 1074487, sea aceptada para su defensa como opción al grado de Maestría en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

Dr. Óscar L. Chacón Mondragón
Asesor

Dr. Igor S. Litvinchev
Revisor

Dr. Hugo J. Escalante
Revisor

Vo. Bo.

Dr. Moisés Hinojosa Rivera
División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, septiembre 2011

A mi Familia, amigos y a la fuerza que me mueve día a día
mis amores Jaime y Mabel.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Primeramente quiero agradecer a las instituciónes de las cuales recibí beca para poder realizar mis estudios de maestría:

- El Consejo Nacional de Ciencia y Tecnología (CONACYT)
- Facultad de Ingeniería Mecánica y Eléctrica (FIME)
- Universidad Autónoma de Nuevo León (UANL)

También agradezco enormemente al Dr. Óscar L. Chacón Mondragón por todo el apoyo durante la realización de este trabajo de tesis, por sus consejos y el conocimiento que compartió conmigo.

A los miembros del comité de tesis por sus consejos y sugerencias.

Profesores de PISIS gracias por todos los conocimientos aportados durante mi estudio de maestria y siempre aclarar mis dudas.

Compañeros de PISIS gracias por el compañerismo, el trabajo en equipo y la amistad brindada, Amigos FCFM gracias por siempre darme animos y confiar en mi.

Gracias a mi Familia por siempre darme su apoyo incondicional, alentarme a ser mejor y ayudarme a cumplir mis metas.

Jaime y Mabel, faltan palabras para agradecer a ustedes dos que son mi TODO, gracias mis amores por ayudarme a completar esta meta, que no es solo mia si no

de ustedes también, prueba superada!.

Y por último pero no menos importante, gracias a Dios por darme conocimiento y tiempo para realizar este proyecto.

Resumen

Perla Cecilia Hernández Lara.

Candidato para el grado de Maestro en Ciencias
con especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Solución de problemas de programación disyunta mediante optimización metaheurística

Número de páginas: 66.

Objetivos y método de estudio: Los objetivos del presente trabajo se listan a continuación:

- Adquirir el conocimiento de los problemas de programación disyunta, la estructura y propiedades.
- Dar solución a problemas de programación disyunta sin transformar el problema de ninguna forma, sin agregar variables, sin reformular el problema o hacer relajaciones; nuestro objetivo es resolverlo bajo su estructura matemática original.

- Encontrar un método para tratar las disyunciones, se consideran las soluciones, no solo se evaluan en base a la función de mérito sino también en base a su factibilidad; es en este último contexto donde se requiere la correcta operación para el manejo de la infactibilidad en un conjunto de disyuntos.
- Dar solución al problema independientemente del tipo de variables que tenga: variables continuas, enteras, discretas o binarias.
- Implementar el método propuesto en un lenguaje matemático donde la dificultad no sea la solución de funciónes lineales o no lineales (los problemas de programación disyunta pueden ser lineales o no lineales).

Metodología:

- Estudio amplio sobre la téoria de programación disyunta, características principales, propiedades, formulaciones.
- Estudio de los métodos propuestos para dar solución a este tipo de problemas.
- Estudio amplio y profundo de Algoritmos Evolutivos, en particular Algoritmos Genéticos, así como también del lenguaje computacional Octave, para su implementación y operación.
- Elaboración del procedimiento para dar solución a problemas de Programación Disyunta mediante Algoritmos Genéticos.
- Implementación del algorimo en Octave.

CONTRIBUCIONES Y CONLUSIONES: La contribución más importante es el procedimiento seguido para evualar al elementos de la población, su jerarquía en las soluciones, orientado a la factibilidad que aporta la solución, y a la identificación del disyunto más apto dentro del conjunto de disyuntos al que pertenece asi como la función de mérito (objetivo del Problema de Programación Disyunta). Para esto

se creo un nuevo método de solución basado en poblaciones para dar solución a problemas de programacíon disyunta con su implementación en octave.

Se probó el método propuesto para problemas con variables reales y enteras quedando como trabajo a futuro probar el método para variables binarias y discretas (Ejemplo de conjunto de variables discretas 2, 4, 6, 10). Las pruebas se realizarón manteniendo la regla de parada con el número de generaciones. Se probarón dos tipos de selección de la nueva generación, primero utilizando el criterio $80-20$, esto es, separando los individuos factibles de los infactibles y eligiendo el $80\,\%$ de los individuos de la nueva generación de los individuos factibles y el $20\,\%$ de los individuos infactibles. Después la nueva generación se seleccionó mediante el criterio Ψ el cual considera una relación entre su nivel de infactibilidad y el valor de la función de mérito. Con el criterio Ψ se obtuvierón buenos resultados con pocas generaciones.

Firma del asesor: _____

Dr. Óscar L. Chacón Mondragón

Capítulo 1

Introducción

En problemas de optimización, en los cuales se requiere encontrar la solución óptima entre un espacio de soluciones factibles, comunmente se identifica como Programación Matemática (**PM**) a la estructura matemática consistente de un objetivo numérico a ser minimizado ó maximizado, sujeto a restricciones funcionales y de estado(variables de decisión).

Cuando un problema de PM es formulado a traves de ecuaciones lineales tanto en restricciones como en objetivo, con variables reales, es llamado Programación Lineal (**LP** por sus siglas en inglés).

Un programa lineal entero-mixto (**MILP** por sus siglas en ingles) permite restringir las variables a valores enteros, discretos y/o binarios.

Programación Disyunta es una extensión de estas dos (LP, MILP), que permite restricciones disyuntas esto es $x \geq 4 \wedge x \leq 2$.

Generalizando, Programación disyunta es optimización sobre una unión de poliedros [4, 16, 5]

Los problemas de Programación disyunta son NP-hard ya que, si bien un conjunto de los disyuntos es convexo, la unión de dichos disyuntos no necesariamente lo es, agregando a esto que las variables pueden ser enteras, discretas o binarias.

Muchos problemas se pueden representar mediante Programación Disyunta; estos problemas surgen cuando tenemos más de una opción para llevar a cabo un

fin. Como ejemplo dare un caso muy simple, el caso de un carpintero que en su taller tiene madera, clavos, pintura y el sabe hacer 3 diferentes de diseños de juegos de puerta y ventana, pero no sabe la cantidad que necesita hacer de cada diseño, de tal manera que sus ganancias aumenten, ¿Cómo se podria modelar este caso?

Modelado con programación disyunta:

- Primero, establecemos las variables de decisión que son la cantidad de puertas y ventanas, llamémoslas x_pi y x_vi.
- Para este ejemplo no tendriamos restricciones generales.
- Los conjuntos de restricciones se forman(para este caso) de acuerdo al diseño a realizar, ya que cada diseño necesita de cierta cantidad de materiales; esto es, en cada diseño:
  - Para madera $M_{pi}x_{pi} + M_{vi}x_{vi} \leq M, i = 1, 2, 3$
  - Para clavos $C_{pi}x_{pi} + C_{vi}x_{vi} \leq C, i = 1, 2, 3$
  - Para pintura $P_{pi}x_{pi} + P_{vi}x_{vi} \leq P, i = 1, 2, 3$
  - Como son el juego de puerta y ventana, entonces $x_{pi} = x_{vi}$
- Funión objetivo $maxZ = \sum_{i=1}^{3}(Ax_{pi+} + Bx_{vi})$

Donde: $i = 1, 2, 3$ es el conjunto de subindices para los diseños.

M_{pi} es la cantidad de madera a utilizar para la puerta en el diseño i.

M_{vi} es la cantidad de madera a utilizar para la ventana en el diseño i.

C_{pi} es la cantidad de clavos a utilizar para la puerta en el diseño i.

C_{vi} es la cantidad de clavos a utilizar para la ventana en el diseño i.

P_{pi} es la cantidad de pintura a utilizar para la puerta en el diseño i.

P_{vi} es la cantidad de pintura a utilizar para la ventana en el diseño i.

A Ganancia por puerta.

B Ganancia por ventana.

La Formulación como problema de programación disyunta de este ejemplo es la siguiente:

$$\begin{array}{rcl} \max & \mathrm{Z} & = \sum_{i=1}^{3}(Ax_{pi+} + Bx_{vi}) \\ \vee_{i=1}^{3} & \left[\begin{array}{c} x_{pi} = x_{vi} \\ M_{pi}x_{pi} + M_{vi}x_{vi} \leq M \\ C_{pi}x_{pi} + C_{vi}x_{vi} \leq C \\ P_{pi}x_{pi} + P_{vi}x_{vi} \leq P \end{array}\right] & x_{pi} \quad y \quad x_{vi} \quad \in \end{array}$$

$\mathbb{N}$ (1.1)

Este tipo de problemas se han resuelto bajo diversos métodos de reformulación, entre los cuales el más eficiente es el del envolvente convexo (convex-hull) de Balas, pero este método es difícil de implementar debido a que:

- Nuevas variables deben ser introducidas.
- Restricciones deben ser modificadas.
- Nuevas restricciones deben ser agregadas.

El método de la Big-M es de los más utilizados pero no siempre los resultados obtenidos son muy buenos, también otros de los métodos utilizados son heurísticas y con relajaciones.

Nuestro objetivo es crear un algoritmo génetico que resuelva el problema de programación disyunta en su forma estandar, sin transformarlos en problemas de Programación Entera Mixta. La aportación más importante es el ordenar la población con respecto a la factibilidad de los individuos siendo la dificultad el asignar un val-

or adecuado cuando se tienen un conjunto de disyuntos y al mismo tiempo el orden basado en el valor que genera en la función objetivo.

Otros aspectos importantes son la representación de los individuos ya que las variables pueden ser reales, enteras, discretas o binarias, así como también representar la factibilidad o infactibilidad del individuo respecto a cada disyunto, y como se trataran las restricciones generales.

El algoritmo genético propuesto se puede aplicar para todo tipo de variables así como también para funciones tanto lineales como no lineales, siempre resolviendo tal cual el problema en su forma estandar.

El siguiente trabajo de tesis se estrucura de la siguiente manera: En el capítulo 2 titulado *Programación Disyunta* se describen conceptos básicos de programación disyunta así como métodos no heurísticos utilizados para resolverlos. Después, en el Capítulo 3 titulado *Algoritmos Evolutivos* se presentan los conceptos más importantes de los Algoritmos Evolutivos en específico Algoritmos Genéticos. Posteriormente, el Capítulo 4 con el nombre de *Problemas de programación disyunta mediante algoritmos genéticos* en el punto 4.1 se discute sobre las consideraciónes de los problemas a resolver; en el punto 4.2 se plantea el algoritmo genético propuesto, detallando cada punto. El Capítulo 5 *Resultados experimentales* trata sobre los problemas resueltos así como los resultados arrojados por estos, y finalmente el Capítulo 6 *Conclusiones y trabajo a futuro* se presenta las conclusiones y trabajo a futuro propuesto.

Capítulo 2

Programación Disyunta

Programación Disyunta *Es la optimización de una función de mérito en una región de factibilidad que considera disyuntos. También la podemos entender como: Optimización bajo union de poliedros, mientras los poliedros son convexos, sus uniones seguramente no lo son, ya que la no convexidad es debido a las disyunciones[5]. Los problemas de optimización disyunta están entre los más difíciles en optimización.*

2.1 Conjuntos Disyuntos

Trabajaremos con los mencionados conjuntos disyuntos mediante operadores lógicos.

2.1.1 Definiciones

Disyunto *Conjunto de restricciones.*

Preposición *Es cualquier expresión lógica y consiste en un conjunto de disyunciones d_i, $i = 1, 2, ..., n$ que son relacionadas mediante operadores lógicos* ***OR*** $(\vee)$ ***AND***$(\wedge)$ ***IMPLICATION***$(\rightarrow)$.

Conjunto Disyunto *Conjunto de restricciones separadas por un operador*

OR.[1]

$$D = \bigvee_{i \in N} [h_i(x) \leq 0] \, x \in \mathbb{R}^{\mathbb{N}}$$

Asumiendo que $h_i(x)$ es una función continua y convexa, y considerando que D es solamente un conjunto de desigualdades, se tiene las siguientes definiciones:

Región factible de un término disyunto: *Conjunto de puntos que satisfacen la siguiente desigualdad.*

$$R_i = \{x | h_i(x) \leq 0\}$$

.

Poliedro: *Un conjunto disyunto puede ser expresado de formas lógicamente equivalentes. Denotemos el subespacio por:*

$$H^+ = \{x \in \mathbb{R}^n | ax \geq a_0\},$$

donde $a \in \mathbb{R}^n, a_0 \in \mathbb{R}$. *La intersección de una colección finita de subespacios es un conjunto de la forma:*

$$P = \bigcap_{i \in M} H_i^+ = \left\{x \in \mathbb{R}^n | a^i x \geq a_{i0}, i \in M\right\}$$

es conocida como poliedro.

Conjunto disyunto elemental: *Se llama así a la unión de una colección finita de subespacios, esto es un conjunto de la forma:*

$$D = \bigcup_{i \in M} H_i^+ = \left\{ x \in \mathbb{R}^n | \bigvee_{i \in M} (a^i x \geq a_{i0}) \right\},$$

Un conjunto disyunto puede ser expresado de muchas formas diferentes, logrando llegar de una expresión a otra por ser considerada F como expresión lógica, entre estas formas equivalentes, se tienen dos extremos [1, 4] que son:

1. La forma normal conjunta (CNF)

$$F = \bigcap_{i \in T} D_i,$$

donde cada D_i es un conjunto disyunto elemental.

2. La forma normal disyunta (DNF)

$$F = \bigcup_{i \in Q} P_i,$$

donde cada P_i es un poliedro.

2.1.2 Propiedades

Si la unión de las regiones factibles de los términos disyuntos es igual a uno de estos términos, el cual tiene la región más grande, entonces, el conjunto disyunto es llamado *impropio*, en otro caso el conjunto disyunto es llamado *propio*. Siendo R_i la región factible del disyunto i, el conjunto disyunto impropio puede ser reescrito como sigue:

$$\bigcup_{i \in D} R_i = R_j$$

El conjunto disyunto *impropio* también tiene la siguiente propiedad $R_i \subseteq R_j \quad \forall i \neq j$ Lo que quiere decir que todas las regiones factibles de i con $i \neq j$ en el conjunto disyunto son incluidas en la $j - esima$ región factible, de esto, un conjunto impropio puede ser reducido a:

$$\{x | h_j(x) \leq 0\}$$

El conjunto disyunto *propio* es el conjunto en el cual la intersección de regiones factibles es vacia, o no vacia pero sin que se cumpla que la intersección sea igual a uno de los disyuntos, esto es:

$$\bigcap_{i \in D} R_i = \emptyset \quad ó \quad \left\{ \bigcap_{i \in D} R_i \neq \emptyset \quad \bigcap_{i \in D} R_i \neq R_j \right\}$$

2.2 Antecedentes

El modelo de programación disyunta a considerar tiene la siguiente forma: una función objetivo (o de mérito) que puede ser lineal o no lineal, conjunto de disyuntos

también lineales o no lineales, y variables que pueden ser reales, enteras, discretas y/o binarias. En este capítulo se mencionan algunos de los métodos utilizados para resolver algunos de los casos. La estrategia más conocida hasta el momento para resolver este tipo de problemas, es reformular el problema llevandolo a un equivalente MILP, para los que ya se cuenta con *solvers* eficientes[6].

2.2.1 Método de la Big M

El método Big-M establece que un problema dado se puede reformular a un equivalente con restricciones entera-mixta, de la siguiente manera (Ver [1]):

$$\begin{aligned} A^1x - b^1 &\leq M^1(1 - y_1) \\ A^2x - b^2 &\leq M^2(1 - y_2) \\ y_1 + y_2 &= 1 \end{aligned}$$

donde $y_i \in \{0, 1\}$ y M^i son los parámetros Big-M, que son las cotas superiores sobre $A^ix - b^i$. Sea $i \in D$ siendo D el conjunto de subíndices para los disyuntos $D = 1, ..., n$. Supongamos $y_1 = 1$ y $y_2 = 0$ la primer desigualdad se transforma a $A^1x - b^1 \leq 0$ que es la restricción original del primer disyunto, mientras que la segunda desigualdad se transforma en $A^2x - b^2 \leq M^2$ la cual es trivialmente satisfecha ya que M^2 es una cota superior del lado izquierdo para el segundo disyunto, por esto, este disyunto es ignorado.

La eficiencia computacional de este método esta directamente relacionada con la calidad de los parámetros Big-M, y cantidad ya que los M^i son vectores; la mayoría de las veces se establecen valores arbitrariamente altos para estos parámetros, lo que ocasiona crear un espacio solución mucho mayor.

Los valores más ajustado para los M^i pueden ser calculados por:

$$M^i = max\left\{A^ix - b^i | x^L \leq x \leq x^U\right\}$$

2.2.2 Método de Beaumont surrogate

Beaumont propuso una desigualdad válida para el conjunto disyunto, teniendo un valor valido de las M^i calculándolo de la misma manera que en el método de la big-M, dividiendo cada restricción $i \in D$ por M_i y sumando bajo $i \in D$, este método es interesante por no involucrar variables binarias, dicha desigualdad queda como sigue:

$$\sum_{i \in D} \frac{h_i(x)}{M_i} \leq N - 1$$

donde $N = |D|$. Beaumont mostro que la desigualdad que propuso es equivalente al método Big-M en el espacio continuo x cuando las restricciones son lineales [1].

2.2.3 Método del envolvente convexo

Este método es una de las técnicas de reformulación más eficientes, el *Convex-hull* de Balas [1], mas sin embargo tiene muchos inconvenientes cuando se lleva a la práctica, como la introducción de nuevas variables, restricciones deben ser modificadas así como nuevas restricciones necesitan ser agregadas, ocasionando que el álgebra de la reformulación sea complicada y propensa a errores, incluso en los problemas pequeños; es difícil reconocer como la salida representa la restricción original.

La envolvente convexa cubre todos los puntos factibles en el menor espacio posible. Imaginemos una cinta elástica que tiene que envolver una cierta cantidad de puntos, toma los puntos más lejanos como soporte para mantener dentro a todos los puntos. Para ejemplificar mejor lo que el método hace, ver figuras 2.1 y 2.2.

2.2.4 Lift-and-Project

Programación pura y entera mixta pueden ser representadas como programación disyunta, pero lo mismo puede decirse para instancias de problemas completamente lineales; la atención de este método se enfoca solamente a programación pura y programación mixta 0-1. El método Lift-and-Project se basa principalmente

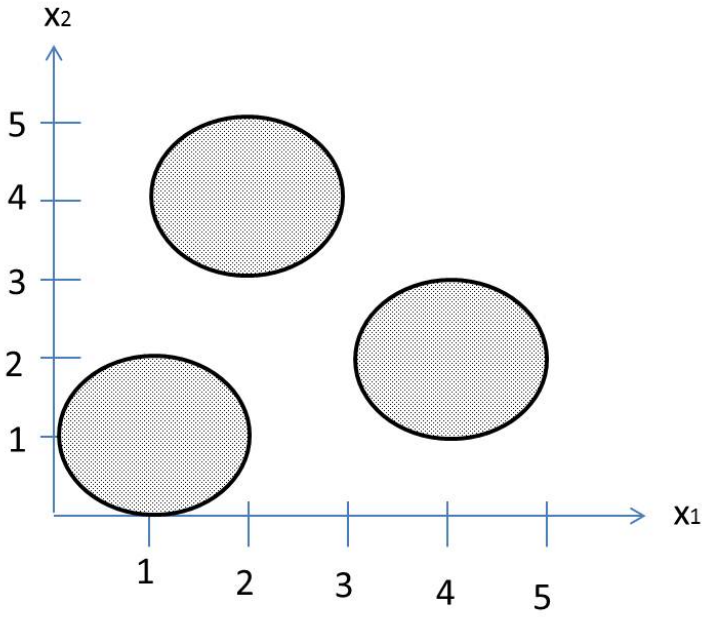

Figura 2.1: Ejemplo de espacio solución de problema de programación disyunta.

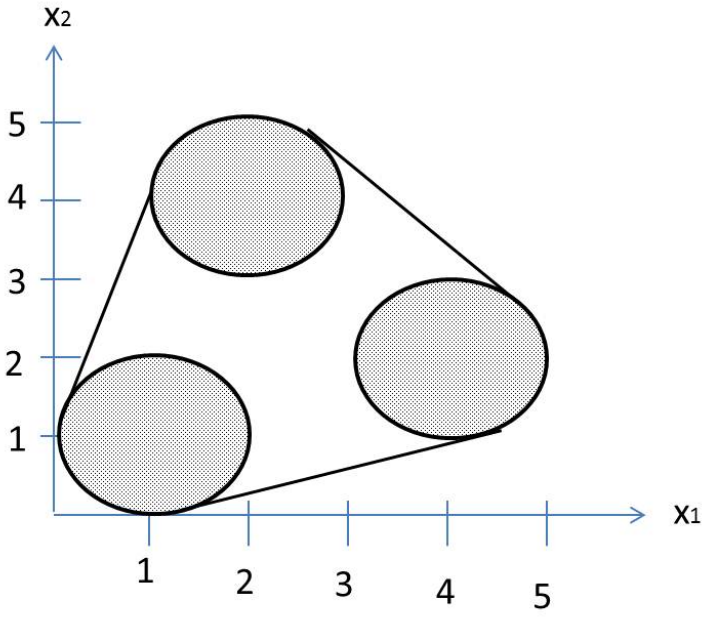

Figura 2.2: Envolvente convexa de problema de programación disyunta.

en dos ideas principales, donde la primera usa la forma normal disyunta (DNF) mientras la segunda usa la forma normal conjunta (CNF)[5]:

1. Hay una representación compacta del envolvente convexo de una unión de poliedros en un espacio dimensional mayor, que a su vez se puede proyectar de nuevo en el espacio original. El primer paso de esta operación puede ser vista como un levantamiento, el segundo paso como proyección. Como resultado obtenemos el envolvente convexo en el espacio original.

2. Una gran clase de conjuntos disyuntos, llamada *facial* puede ser convexificado secuencialmente, esto es, sus envolventes convexos pueden ser derivados de imponer los disyuntos uno a la vez, generando cada vez el envolvente convexo del conjunto actual.

2.2.5 Automatización de transformaciones de programación matemática

La idea general de este método es que la dicotomía expresada por las disyunciones sea considerada en lugar de la naturaleza discreta de las variables enteras. Una variable binaria entera $y_i \in \{0, 1\}$ es asociada con cada $i - ésimo$ disyunto de una disyunción y la disyunción es reemplazada por conjunción, solo un y_i es requerido para ser 1 y solo las disyunciones del correspondiente disyunto se aplican. Los disyuntos $j \neq i$ se reducen a tautologías. Se consideran algunos métodos específicos realizando algunas variaciones. Todos conservan restricciones lineales que es importante para la eficiencia del solver [13].

En la definición del método Big-M requiere la introducción de dos sentencias:

1. Se necesita una operación para el cálculo de los parámetros Big-M.

2. Define una operación para convertir cualquier restricción a una desigualdad en la forma Big-M.

2.2.6 Solución paralela de problemas de programación disyunta

Los problemas de optimización disyunta comunmente pueden ser divididos en un número contable de subproblemas que son automáticamente convertidos a la forma conjunta, lo que los hace fácil de solucionar por la existencia de *solvers*. Estos problemas pueden ser resueltos separadamente y la mejor solución es la solución óptima global del problema original, las técnicas de optimización en común resuelven estos problemas secuencialmente haciendo la complejidad a tiempo alto.

La idea de soluciones en paralelo siempre ha sido una dificultad, empezando por el analisis de complejidad, haciendo casi cualquier problema de optimización combinatoria solucionable en tiempo polinomial. La construcción de una máquina que resuelva en paralelo es de cualquier manera apenas téorica aunque si se han hecho intentos por construir los más rigurosos *solvers* en paralelo. J. Björkqvist y T.Westerlund proponen conectar computadoras estándar a una red como internet puede proveer el hardware para alcanzar esta máquina [6].

La implementación propuesta consta principalmente de dos partes:

1. Parte Servidor, que es responsable de enumerar el problema original en subproblemas y distribuirlos.

2. Parte Cliente, responsable de la solución en si de los subproblemas.

Los subproblemas pueden ser resueltos con técnicas convencionales tales como el *branch and bound* y programación lineal.

La estructura de este método propuesto se muestra en la figura 2.3.

Trabajos relacionados: [9, 10, 2, 20, 21, 24, 22, 14, 7]

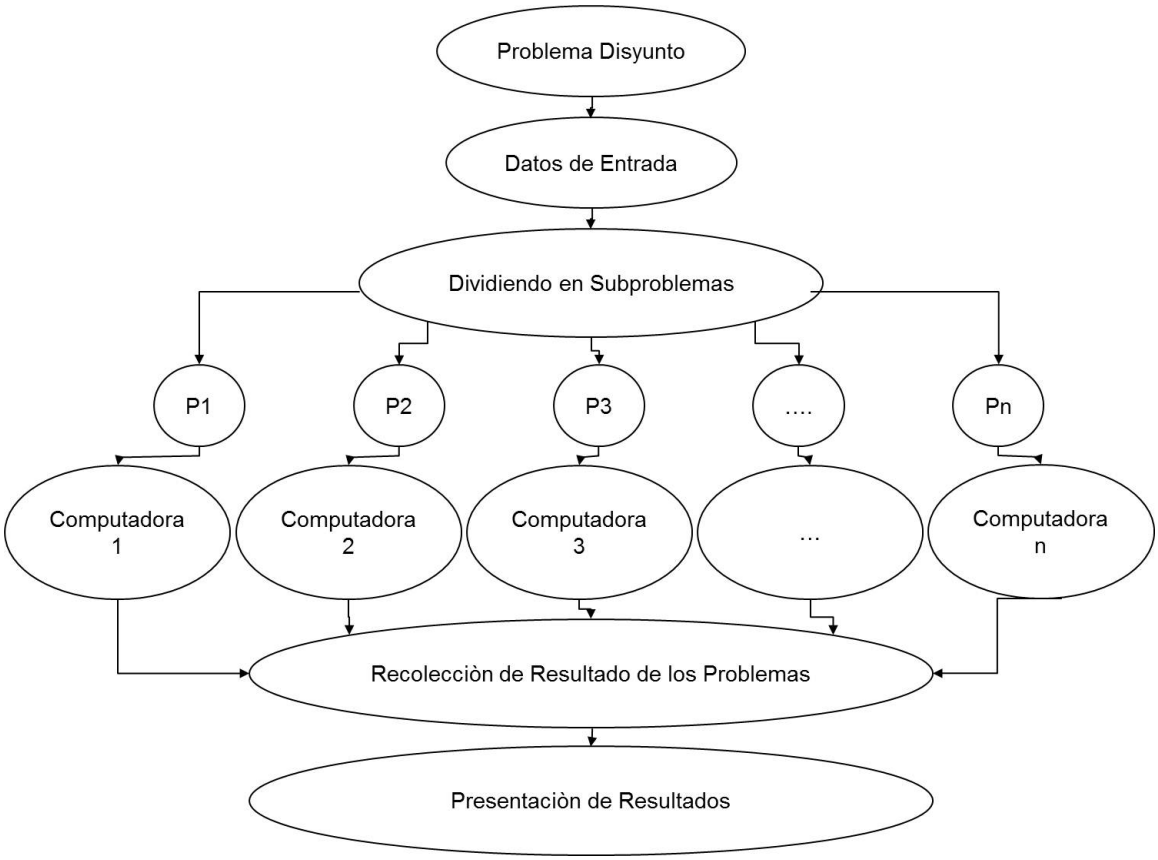

Figura 2.3: Grafico de la estructura de Solución paralela de problemas de programación disyunta.

Capítulo 3

Algoritmos Evolutivos

Los algoritmos evolutivos constituyen una técnica general de resolución de problemas de búsqueda y optimización inspirada en la teoría de la evolución de las especies y la selección natural. Estos algoritmos permiten abordar problemas complejos que surgen en las ingenierias y campos científicos como problemas de planificación de tareas, horarios, tráfico aéreo y ferroviario, búsqueda de caminos óptimos, optimización de funciones, etc [3].

3.1 Algoritmos Genéticos

La idea que sustenta los algoritmos genéticos es emular lo que la naturaleza hace con respecto a la evolución de las especies, tomando el ejemplo dado en [17], consideremos una población de conejos, algunos de ellos son más rápidos y más inteligentes que otros; esta clase es menos probable de ser capturada por los zorros y por esto la mayoría de ellos sobrevivirán. Seguramente algunos de los conejos más lentos y tontos también sobrevivirán y esto solamente por que tuvieron suerte, esta población de sobrevivientes comenzará a reproducirse, los resultados de la reproducción serán una buena mezcla de material génetico de los conejos: conejos lentos se cruzarán con conejos rápidos, algunos rápidos con rápidos, conejos inteligentes con conejos tontos y así sucesivamente, la mejor parte de esto es que la naturaleza produce liebres mediante la mutación del material génetico de los conejos. El resultado de esta nueva población podrían ser bebés conejos más rápidos y más inteligentes

que los de la población original y esto porque los más rápidos e inteligentes de la población original sobrevivieron a la persecución de los zorros. Los algoritmos genéticos siguen paso a paso el procedimiento que se asemeja a la historia de los conejos.

Los algoritmos genéticos son aquellos que se sustentan en bases estocásticas de búsqueda, que modelan fenómenos naturales como lo es la genética de la herencia y la competencia Darwiniana por la supervivencia del más fuerte, es un método de búsqueda con independecia del dominio que presenta un balance entre la explotación y la exploración del espacio de búsqueda de la solución.

El algoritmo génetico realiza una búsqueda multidireccional manteniendo una población de soluciones potenciales, motivando la adecuación de la información e intercambiando las direcciones de búsqueda. Esta población realiza una evolución simulada: en cada iteración las soluciones relativamente buenas se reproducen en tanto las soluciones relativamente malas, mueren. Para distinguir entre las diferentes soluciones se utiliza una funcion objetivo (conocida también función de mérito, de adaptación, de aptitud.), que modela el medio ambiente natural en un proceso de supervivencia.

Un algoritmo genético (como cualquier algoritmo evolutivo) para un problema en particular debe tener los siguientes componentes:

- Una representación genética para las posibles soluciones del problema.
- Un método para crear una población inicial de posibles soluciones.
- Una función de evaluación que desempeñe el papel de medio ambiente, clasificando soluciones por medio de aptitudes (*fitness*).
- Operadores genéticos que modifican la composición de los hijos.
- Valores para diversos parámetros que utiliza el algoritmo genético, como: tamaño de la población, probabilidades de aplicar operadores genéticos, etc.

El algoritmo general es el siguiente:

ALGORITMO GENÉTICO

1: Generar la población inicial *PopIni*

2: Hacer t=0, *PopOld* = *PopIni*

3: **while** $t \leq MaxNumGen$

do

4: Evaluar *PopOld*

5: Operación de reproducción *PopNew*

6: Operación de cruzamiento *PopNew*

7: Operación de mutación *PopNew*

8: *PopOld* = *PopNew*

9: **end while**

Un algoritmo genético puede presentar diversas variaciones, dependiendo de como se aplican los operadores genéticos, de como se realiza la selección y de como se decide el reemplazo de los individuos para formar la nueva población. En general, el pseudocódigo consiste de los siguientes pasos:

1. **Inicialización**: Se genera aleatoriamete la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que dentro de la población inicial se tenga la diversidad estructural de estas soluciones, para tener una mayor representación de la mayor parte de la población posible, o al menos evitar la convergencia prematura.

2. **Evaluación**: A cada uno de los cromosomas de esta población se aplicará la función de aptitud, para saber que tan buena es la solución que se está codificando.

3. **Condición de término**: El algoritmo genético de deberá detener cuando se alcance la solución óptima, pero ésta generalmente se desconoce, por lo que se deben utilizar otros criterios de terminación del proceso:

- Correr el AG un número máximo de iteraciones (Generaciones).
- Detenerlo cuando no haya cambios en la población.

Mientras no se cumpla la condición de término se hace lo siguiente:

4. **Selección**: Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con la mejor aptitud tienen mayor probabilidad de ser seleccionados. Los mecanismos de selección más frecuentemente utilizados se presentan a continuación:

   - **Selección de ruleta**: Es también conocida como la selección proporcional a la función de desempeño del i_{esimo} individuo. La probabilidad asociada a su selección (cuando se esta maximizando) esta dada por:

     $$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

     Esta selección permite que a los mejores individuos les corresponda una mayor probabilidad de ser elegidos, pero al mismo tiempo permite a los peores individuos ser elegídos, esto puede ayudar a mantener diversidad en la población, a excepción del caso en el que existe una pequeña fracción de la población que posee una medida de desempeño excesivamente superior al resto; este caso permite pérdida de diversidad y conduce a la convergencia prematura.

   - **Selección basada en ranking**: En esta selección los individuos se ordenan en base a su medida de desempeño, y después en base a este orden se les asigna una segunda medida de desempeño (Probabilidad), inversamente proporcional a su posición (Esto es, otorgando una mayor medida de desempeño a los mejores individuos). Los individuos son seleccionados proporcionalmente en base a esta probabilidad. Este método de selección reduce el riesgo de convergencia prematura que da la selección de ruleta. La probabilidad asociada a su selección esta dada por:

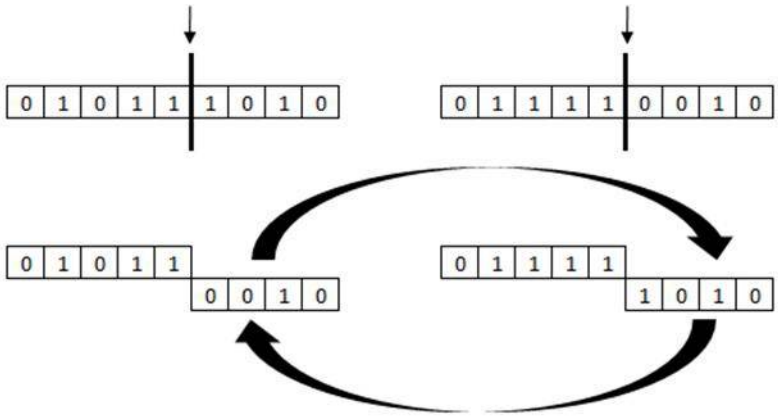

Figura 3.1: Ejemplo de cruzamiento en un punto.

$$P_{rank}(i) = \frac{2-s}{N} + \frac{2i(s-1)}{N(N-1)}$$

Con s como parametro que puede estar entre [1, 2].

- **Selección por torneo**: Esta selección se realiza mediante un torneo o comparación, entre un pequeño subconjunto de individuos elegidos al azar desde la población. Los beneficios de esta selección es la velocidad de aplicación y la capacidad de prevenir, en cierto grado, la convergencia prematura. Su principal desventaja es la necesidad de establecer el parámetro correspondiente al tamaño del subconjunto.

5. **Cruzamiento**: El cruzamiento es el principal operador génetico, este operador representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos descendientes que contengan características de ambos padres.

   - **Cruzamiento en un punto**: Se selecciona un punto en el vector del primer parental, todos los valores más allá de este punto (en el vector que representa al individuo) se intercambian entre los dos vectores parentales. Los individuos resultantes son los hijos, por ejemplo la figura 3.1.
   - **Cruzamiento en dos puntos**: Se seleccionan dos puntos en el vector parental, todos los valores entre estos dos puntos se intercambian entre

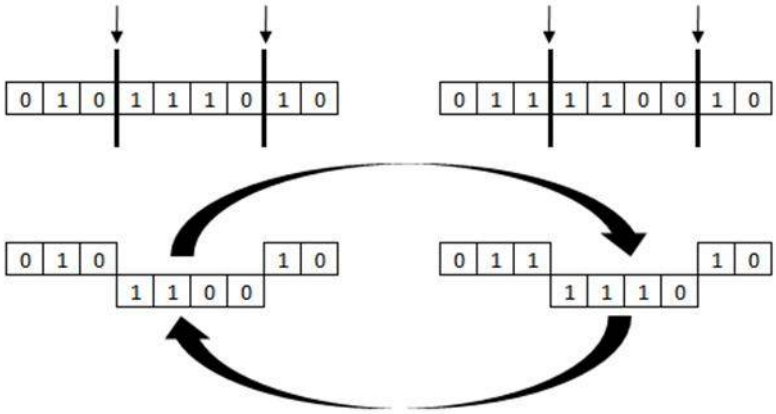

Figura 3.2: Ejemplo de cruzamiento en dos puntos.

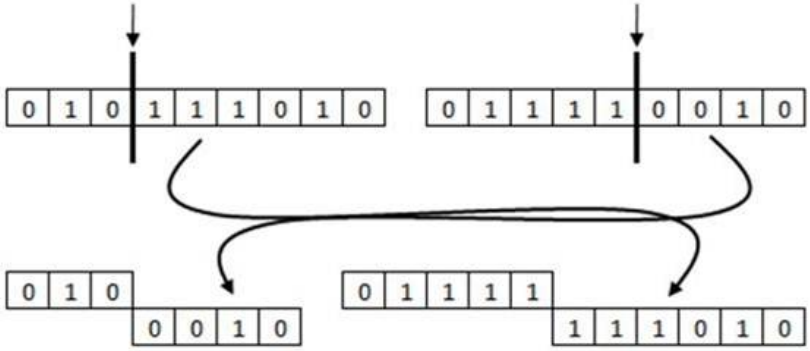

Figura 3.3: Ejemplo de cruzamiento corte y empalme.

los organismos parentales, creando dos individuos hijos, por ejemplo la figura 3.2.

- **Corte y empalme**: Este es otro tipo de cruzamiento en el cual el enfoque de cortar y empalmar ocasiona un cambio de la longuitud de vectores de los hijos, la razón de esta diferencia es que se selecciona un punto de corte diferente para cada vector parental, por ejemplo la figura 3.3
- **Cruzamiento uniforme y uniforme medio**: En ambos de estos esquemas los dos padres se combinan para producir los descendientes. En el esquema de cruzamiento uniforme, los bits se comparan individualmente entre ambos padres. Los bits se intercambian con una probabilidad fija

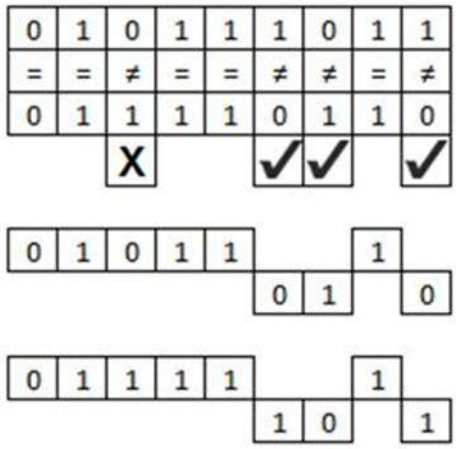

Figura 3.4: Ejemplo de cruzamiento uniforme.

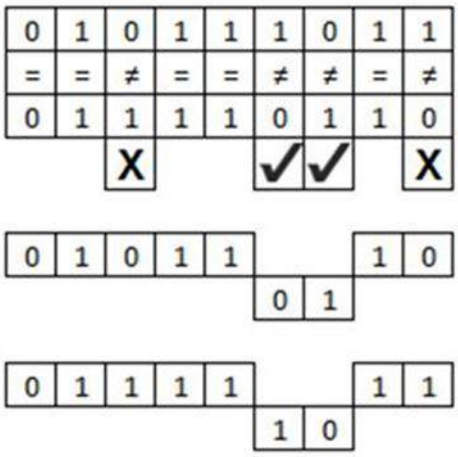

Figura 3.5: Ejemplo de Cruzamiento uniforme medio.

(usualmente 0.5) ver fgura 3.4.

En el esquema de cruzamiento uniforme medio exactamente la mitad de los bits que son diferentes se intercambian, esto se muestra en la figura 3.5.

- **Cruzamiento de cromosomas ordenados**: Este tipo de cruzamiento depende de como se represente la solución del problema, un cambio directo puede no ser posibe. Un ejemplo donde se puede aplicar este cruzamiento es el caso del problema viajero, ya que la solución es una lista ordenada

de las ciudades visitadas.

6. **Mutación**: Este operador genético modifica al azar parte del cromosoma de los individuos, esto para permitir alcanzar zonas del espacio de busqueda que no se alcanzan con los individuos de la población actual. La técnica más clásica para realizar la mutación es la modificación de un bit en el algoritmo génetico, dicha modificación se realiza con una probabilidad preestablecida, llamada probabilidad de mutación. El proposito de este operador génetico es proveer un mecánismo para escapar de los óptimos locales, así como también desplazar a los individuos hacia otros espacios de búsqueda que aun no han sido alcanzados por los anteriores operadores géneticos.

7. **Reemplazo (Nueva generación)**: Una vez que se aplicaron los operadores géneticos se seleccionan los mejores individuos para la siguiente generación.

Capítulo 4

Problemas de Programación disyunta

En este capítulo se plantea la propuesta de tesis; en la sección 4.1 planteamos el modelo matemático de un problema con disyunciones, así como los casos que el algoritmo propuesto podrá resolver, siguiendo en la sección 4.2 en la cual se describe el algoritmo genético propuesto para resolver problemas de programación disyunta y finalmente en la sección 4.3 se presentan los problemas que utilizamos para hacer pruebas de dicho algoritmo.

4.1 Consideraciones

Los problemas que estamos interesados en resolver son los problemas de programación disyunta generalizada, donde un programa como tal puede ser formulado como sigue [18],[23]:

$$min\, Z = \sum_{k \in K} C_k + f(x)$$

sujeto a: $\quad g(x) \leq 0$

$$\vee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \leq 0 \\ C_k = \gamma_{ik} \end{bmatrix} \quad k \in K$$

$$\Omega(Y) = V$$

$$\mathrm{x} \in \mathbb{R}^n,\ Y_{ik} \in \{V,\ F\}^m,\ C_k \geq 0$$

(4.1)

Siendo D_k el *k-esimo* conjunto de disyuntos (ver sección 2.1.1) y las elecciones discretas son expresadas con variables booleanas Y_{ik} en términos de disyunciones y preposiciones lógicas $\Omega(Y)$. La característica atractiva de programación disyunta generalizada es que permite una representación simbólica/cuantitativa de problemas de optimización discretos y continuos.

4.2 Algoritmo Genético

El Algoritmo propuesto sigue el procedimiento de algoritmos genéticos, por lo que se plantea cada componente de estos algoritmos y al final el esquema general.

4.2.1 Componentes del Algoritmo Genético

1. **Representación genética.** Los elementos fundamentales de toda heurística son la definición correcta del objetivo y su representación matemática, así como la representación de la solución [19]. Para el algoritmo genético propuesto la solución se representará mediante una cadena de caracteres de elementos binarios. A esta cadena de caracteres, por su comparación a procesos genéticos, se les denomina *cromosoma*, y a cada uno de sus elementos *gen* y el valor que toman *allelo*.

En sistema binario los enteros positivos $\mathbb{Z}^+$ se pueden representar en la forma de un cromosoma; lo que se hace es convertir la cadena de base 2 (binaria) a base 10 (decimal). En este sentido sea $[c_0...c_i...c_{m-1}]$ un cromosoma de m genes, para el cual el valor entero positivo que representa es el siguiente:

$$\hat{x} = c_0 2^0 + c_1 2^1 + ... + c_i 2^i + ... + c_{m-1} 2^{m-1} \tag{4.2}$$

donde $c_i \in \{0, 1\}$, por ejemplo: El cromosoma de 5 genes [11101] representa el valor $1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 = 1 + 0 + 4 + 8 + 16 = 29$. Ahora para representar números reales mediante cadenas binarias, es necesario establecer el rango de la variable x , $[x^l, x^u]$ donde x^l es la cota inferior y x^u la cota superior; sea también q el número de cifras decimales de aproximación para un intervalo deseado de separación de valores desde su rango dado. La primera tarea a realizar es encontrar el número de particiones en los que se puede dividir el intervalo de la variable x, esto es, el número de genes para el cromosoma, y este se obtiene mediante la siguiente relación:

$$2^{m-1} < \frac{(x^u - x^l)}{10^{-q}} \leq 2^m - 1 \tag{4.3}$$

Ejemplo: Sea $x \in [-2, 4]$ donde $x^l = -2$ y $x^u = 4$, $q = 2$. Aplicando (4.3) obtenemos:

$$2^{m-1} < \frac{6}{0.01} = 600 \leq 2^m - 1$$

Se cumple la desigualdad con $2^{m-1} = 2^{10-1} = 2^9 = 512$ y $2^m - 1 = 2^{10} - 1 = 1024 - 1 = 1023$, de donde $m = 10$, esto es, el número de intervalos está entre 0 y $2^{10} - 1 = 1023$, denotemos al número de intervalos $\hat{x}$ mediante la ecuación (4.2). Ahora, el valor de la variable x se obtiene de la siguiente manera:

$$x = x^l + \frac{(x^u - x^l)}{2^{m-1}} \hat{x}; \qquad \hat{x} \in [0, 2^m - 1] \tag{4.4}$$

De todo lo anterior podemos observar que la variable x se puede representar como un cromosoma a través de la variable $\hat{x}$ que toma valores enteros en el intervalo $[0, 2^m - 1]$, obteniendo 0 del cromosoma [000000000] y $2^m - 1$ de [1111111111], de los cuales obtenemos las cotas inferior y superior de la variable

de decisión x.

$\hat{x} = 0 \rightarrow x = -2 + \frac{6}{1023}(0) = -2; \qquad \hat{x} = 1023 \rightarrow x = -2 + \frac{6}{1023}(1023) = 4;$

Para nuestro problema dado en el cual las variables pueden ser continuas, enteras, discretas, o binarias, (más en todos los casos se genera aleatoriamente una cadena de caracteres de elementos binarios), al generar la población inicial se tienen que considerar todos los casos:

a) **Variables continuas:** Para este caso se sigue el procedimiento antes planteado, obteniendo el valor de la variable de decisión mediante la ecuación (4.4), tomemos el ejemplo anterior, consideremos el cromosoma [010101101] $\hat{x} = 151$ resolviendo (4.4) $x = 2.94$, de esta manera se obtienen las variables continuas.

b) **Variables enteras:** Se obtienen de la misma manera que las variables continuas sólo que una vez aplicada la ecuación (4.4) redondeamos la variable mediante la función de octave *round*, para el ejemplo anterior $round(x = 2.94) = 3$.

c) **Variables discretas:** Primero que nada se ordena el conjunto de las variables discretas. Para generar estas variables se genera también una cadena de elementos binarios, donde m=número de valores que puede tomar dicha variable, la diferencia es que esta cadena solamente tiene un valor igual a 1 y el resto son ceros, de esta manera la posición del valor igual a 1 nos indica la posición del valor de la variable discreta. Ejemplo: sea $x \in \{2, 3, 6, 9, 13, 17\}$, consideremos el cromosoma 000100 el cual nos indica que x tomara el cuarto valor del conjunto ordenado de las variables discretas, esto es, $x = 9$.

d) **Variables binarias:** Se genera de la misma manera que las variables enteras con limites fijos $x^u = 1$ y $x^l = 0$, redondeando la variable mediante la función de octave *round*, siendo $x = 0$ para valores de $x \leq 0.5$ y $x = 1$ para valores de $x > 0.5$.

2. **Población inicial.** El número de individuos de que constará la población inicial es un valor fijo establecido.

3. **Función de adaptación (o evaluación).** Uno de los problemas presentes en la optimización de funciones es la inclusión de restricciones. En los algoritmos evolutivos, esta condición ha sido estudiada ampliamente y diversos métodos de manejo de las restricciones han sido sugeridos [8] [15]. Un problema adicional en el presente caso es la inclusión de alternativas (Conjuntos de restricciones-disyuntos) mutuamente excluyentes que da lugar a nuevos métodos de manejo de restricciones. Si algún elemento de la población incumple con alguna(s) restricción(es), entonces dicho elemento dará lugar a una solución infactible, estableciéndose entonces un compromiso entre la solución óptima de la función de aptitud y la factibilidad de la solución proporcionada por el elemento de la población. En este contexto este trabajo proporciona un nuevo término ψ (ver sección 4.2.1, ecuación 4.8) que establece dicha reación entre los elementos proporcionando una función de aptitud pseudofactible, para lograr un buen ordenamiento de los elementos basados en este principio.

   Para los problemas a resolver, el ambiente al que deben sobrevivir los individuos no solo consta de la función objetivo, sino tambien es de vital importancia considerar si la solución generada es factible en al menos un disyunto; para considerar la calidad de una solución (buena o mala) se requiere 2 evaluaciones: De las Restricciones - Infactibilidad y de la Función Objetivo - Solución Óptima

   a) **Restricciones**

   La primera tarea a realizar es la evaluación de las restricciones (generales, de los disyuntos, etc.) en la forma ≤ 0 para cada uno de los individuos de la población (soluciones: valores de las variables de decisión). Una vez evaluadas todas las restricciones para todos los elementos de la población se realiza una representación de la factibilidad de cada elemento para cada una de las restricciones de la siguiente manera:

   1) Generar una matriz de valores de todas las restricciones para todos

los elementos de la población; como referencia de este paso se muestra la Tabla 4.1.

Individuo	**...**	**Restricciones Generales**	**...**	**Restricciones Disyunto**	**...**
1	...	$g_{1,j}$	...	$r_{1,k,j}$	...
...	...	...	...	...	...
l*	...	$g_{2,l*}$	...	$r_{l*,k,j}$	...
...	...	...	...	...	...
N	...	$g_{N,j}$	...	$r_{N,k,j}$	...

Tabla 4.1: Evaluación de la j-ésima restricción general $g_{i,j}$ y de la restricción $r_{i,k,j}$ del k-ésimo disyunto, para cada individuo i=1,...,N

2) Siendo las restricciones de desigualdad ≤ 0, se establece la medida de la *infactibilidad* como los valores positivos; los valores negativos se convertirán en cero una vez normalizados de tal forma que para la restricción factible el elemento de la población aporte una *infactibilidad* con valor de cero.

3) Una vez evaluadas todas las restricciones (generales, disyuntos, etc) estas se normalizan dividiendo cada columna de valores entre el mayor valor del individuo en esa restricción; al efectuar la división se realiza con el valor absoluto (el valor mayor puede ser negativo), esto para no alterar el signo de los datos y poder identificar los individuos que son infactibles. Recordemos que las restricciones se establecen de forma estandar como menor o igual que cero (≤ 0). Una vez realizado este proceso de normalización, todos los valores negativos se transforman en 0 - restricciones factibles para el elemento de la población considerado o bien de *infactibilidad* 0. Estos resultados se consideran en la forma presentados en la Tabla 4.2 en donde

$$\hat{b}_{s,j} = \frac{g_{s,j}}{|max\{g_{i,j}, \quad i=1,...,N\}| \quad \forall s}$$

$$b_{s,k,j} = \frac{r_{s,k,j}}{|max\{r_{i,k,j}, \quad i=1,...,N\}| \quad \forall s,k}$$

Individuo	**Restricciones Generales**			**Restricciones Disyunto k**		
1	...	$\hat{b}_{1,j}$	...	...	$b_{1,k,j}$	...
...	...	...	..	...	...	...
s	...	$\hat{b}_{s,j}$	...	...	$b_{s,k,j}$	...
...	...	...	...	...	...	...
N	...	$\hat{b}_{N,j}$	...	...	$b_{N,k,j}$	...

Tabla 4.2: Normalización $\hat{b}_{s,j}$ de la j-ésima restricción general $g_{s,j}$ y la $\mathrm{b}_{s,k,j} laj - ésima restricción \mathrm{r}_{s,k,j}$ del k-ésimo disyunto, para cada individuo $s = 1, ..., N$

4) Se genera una representación de *infactibilidad* de cada elemento de la población respecto a las restricciones generales y los disyuntos, sumando los valores normalizados de las restricciones generales y por separado las restricciones en cada disyunto respectivamente, generando agregación de infactibilidad para las restricciones generales y para las restricciones de cada disyunto. La infactibilidad para cada caso se determina de la siguiente forma:

$$InfactG_s = \sum_{j=1}^{N_G} \hat{b}_{1,j} \tag{4.5}$$

$$InfactDy_s = min\{\sum_{j=1}^{N_{D_k}} b_{1,k,j}, \quad k = 1, \ldots, K\} \tag{4.6}$$

$$Infact_s = InfactG_s + InfactDy_s \tag{4.7}$$

La infactibilidad del s-ésimo individuo $Infact_s$ se toma como la infactibilidad de las restricciones generales $InfactG_s$ y la infactibilidad ocasionada por el conjunto de disyuntos $InfactDy_s$, el cual se tomaría -ver etapa 5- como el menor valor de la infactibilidad entre los K disyuntos. Ver Tabla 4.3 para la obtención de la infactibilidad de las restricciones generales y de los disyuntos.

5) Para cada individuo, en el conjunto de disyuntos, se selecciona aquel disyunto que contenga la menor infactibilidad (operador lógico OR,

Individuo	**Infactiblidad Rest. Gral.**			**Infactibilidad Rest. Disy. k**		
1	...	$\sum_{j=1}^{N_G} \hat{b}_{1,j}$	...	...	$\sum_{j=1}^{N_{D_k}} b_{1,k,j}$	...
...	...	...	...	...	...	...
s	...	$\sum_{j=1}^{N_G} \hat{b}_{s,j}$	...	...	$\sum_{j=1}^{N_{D_k}} b_{s,k,j}$	...
...	...	...	.. .	...	...	...
N	...	$\sum_{j=1}^{N_G} \hat{b}_{N,j}$	...	...	$\sum_{j=1}^{N_{D_k}} b_{N,k,j}$	...

Tabla 4.3: Infactibilidad para cada individuo de las restricciones generales y la del k-ésimo disyunto,

ver ecuación (4.6)) y se registra su posición (entre los disyuntos) y el valor de la función objetivo del individuo en cuestión.

6) Una vez que se dispone para cada individuo de la población el valor de la infactibilidad (ecuación (4.7)) y el valor C_k que se agrega al de la función objetivo del elemento en cuestión, se dispone ahora de información para realizar un ordenamiento de los elementos de la población, necesario en el proceso de selección.

En este proceso se dispone ya de un valor del criterio de infactibilidad para cada uno de los elementos de la población. El orden de importancia de los elementos se logrará incluyendo el segundo criterio que es el valor de la función objetivo.

b) **Función objetivo**

Se obtiene el valor en la función objetivo en cada individuo $f(x_i)$. Asociado a este valor, se tiene el valor de la menor infactibilidad y a que disyunto pertenece.

c) **Función de la adaptación**

La función de adaptación se establece a través de una relación entre estos dos criterios de evaluación: su valor en la función objetivo y su valor de infactibilidad. Para ello consideremos la función de adaptación como

(dado que el objetivo es maximizar):

$$\psi_s = \begin{cases} F(x_s) & , Si\ Indfact_s = 0. \\ \dfrac{F(x_s)}{Indfact_s} & , En\ otro\ caso. \end{cases} \tag{4.8}$$

Con esta función de adaptación podemos observar que el ordenamiento de los elementos de la población se realiza primero mediante la función objetivo para aquellos elementos que son factibles (*Infact=0*), y posteriormente los restantes se ordenan de acuerdo a esta función de adaptación (*Infact*$\neq 0$).

4. Operadores genéticos. La tarea principal de estos operadores es mantener la diversidad de individuos así como garantizar la evolución. Operadores genéticos aplicados:

   - Selección: Representa la supervivencia del más apto, la selección la realizaremos aplicando *ranking lineal* a la evaluación realizada, donde los individuos son ordenados inversamente a su medida de desempeño, correspondiendole al peor individuo el índice $i = 0$ mientras que al individuo con mejor evaluación le corresponde el índice $i = N - 1$ (Siendo la población de N individuos). A cada individuo se le asigna una probabilidad para ser seleccionado, de manera que los mejores individuos tengan una mayor probabilidad, la probabilidad que se le asigna es mediante la siguiente fórmula:

$$P_{rank}(i) = \frac{2-s}{N} + \frac{2i(s-1)}{N(N-1)} \tag{4.9}$$

   Ejemplo: Consideremos una población de 4 individuos para los cuales sus evaluaciones se muestran en la tabla 4.4 En la tabla 4.5 se muestran los individuos ordenandos para agregar probabilidad de *ranking*. Para aplicar la probabilidad de *ranking* tomemos $s = 1.2$. El tamaño de la población sabemos que es $N = 4$; a manera de ejemplo obtengamos la probabilidad

Individuo	**F**
1	30
2	10
3	40
4	20

Tabla 4.4: Ejemplo de selección aplicando *ranking lineal*: población

Individuo	**F**	**Índice i**
2	10	0
4	20	1
1	30	2
3	40	3

Tabla 4.5: Ejemplo de selección aplicando *ranking lineal*: índice.

del individuo 1 que tiene índice de *ranking* 2:

$$P_{rank}(2) = \frac{2 - 1.2}{4} + \frac{2 \times 2(1.2 - 1)}{4(4 - 1)} = 0.266$$

de la misma manera se aplica para todos los individuos. Se muestran los resultados de probabilidad asignada a los individuos en la tabla 4.6.

Individuo	**F**	**Índice i**	P_{rank}
2	10	0	0.200
4	20	1	0.233
1	30	2	0.266
3	40	3	0.300

Tabla 4.6: Ejemplo de selección aplicando *ranking lineal*: Probabilidad.

Una vez aplicando el ranking se generan valores aleatorios (entre 0 y 1) para seleccionar los individuos a reproducirse, procurando siempre tener diversidad.

- Cruzamiento: También conocido como sobrecruzamiento o recombinación, juega el papel de la reproducción sexual. El cruzamiento a realizar fue el cruzamiento en un punto, y se describe en el capítulo 3.
- Mutación: La mutación que realizamos en el algoritmo propuesto es la modificación de un gen, solamente a un porcentaje de la población. A cada individuo se le asigna una probabilidad de ser mutado así como también cada gen tiene la misma probabilidad de ser modificado. Ejemplo: Consideremos 5 cromosomas, 1. − [11000110] 2. − [11101001] 3. − [00100011] 4. − [10111101] 5. − [11000011] cada uno con la misma probabilidad ($P_m(i) = 0.2$) de sufrir mutación. Supongamos que sólo el 40 % de la población sufre mutación (Este valor es preestablecido), esto es sólo 2 de los 5 cromosomas sufriran mutación, de los que salen selecionados el cromosoma 2 y el 5, ahora, sólo un gen sera mutado y todos los genes tienen la misma probabilidad de ser mutados ($P_m(j) = 0.125$) de donde supongamos que se mutarán el gen 7 del cromosoma 2 y el gen 3 del cromosoma 5, resultando la población final como: 1. − [11000110] 2. − [11101011] 3. − [00100011] 4. − [10111101] 5. − [11100011]. De esta manera se realiza la mutación.

5. Parámetros del algoritmo genético.

   El algoritmo necesita de los siguientes parámetros de entrada:

   - Tamaño de la población.
   - Rango para detener algoritmo: Número de poblaciones.
   - Parametro s para el *ranking* de la selección.
   - Datos de las variables:
     - Número de variables continuas, enteras, discretas y binarias
     - Cotas de cada una de ellas.
     - Decimales de aproximación.
     - Número de genes.

4.2.2 Ejemplo

A lo largo del capítulo planteamos los componentes más importantes del algorimo genético; ahora con el fin de que sea más claro el algoritmo propuesto para solucionar problemas de programación disyunta, se expondra un ejemplo, siguiendo el pseudocódigo general de un algoritmo genético planteado en el capítulo 3. Consideremos el siguiente problema:

$$
\begin{gathered}
\max \quad \mathrm{F} = \mathrm{C}_{i^*} - 12x_1 + 23x_2 \\
\text{sujeto a:} \\
\mathrm{x}_1 + x_2 \geq \tfrac{1}{2} \\
\mathrm{x}_1 \leq 2 \\
\mathrm{x}_2 \leq 2 \\
\mathrm{x}_1 \geq 0, x_2 \geq 0 \\
\begin{bmatrix} -2x_1 + x_2 \leq 1 \\ x_1 \leq \frac{1}{2} \\ x_2 \geq 1 \\ C = 5 \end{bmatrix} \vee \begin{bmatrix} 2x_1 + x_2 \leq 5 \\ x_1 \geq \frac{3}{2} \\ x_2 \geq 1 \\ C = 6.5 \end{bmatrix} \vee \begin{bmatrix} 2x_1 - x_2 \geq \frac{3}{2} \\ x_1 \leq 2 \\ x_2 \geq 0 \\ C = 7.5 \end{bmatrix} \\
\mathrm{x}_1 \in \mathbb{R}, \qquad x_2 \in \mathbb{I}
\end{gathered}
\tag{4.10}
$$

La primer tarea a realizar es llevar las restricciones a la forma ≤ 0:

$$
\begin{gathered}
\text{-}\mathrm{x}_1 - x_2 + \tfrac{1}{2} \leq 0 \\
\mathrm{x}_1 - 2 \leq 0 \\
\mathrm{x}_2 - 2 \leq 0 \\
\text{-}\mathrm{x}_1 \leq 0, -x_2 \leq 0
\end{gathered}
$$

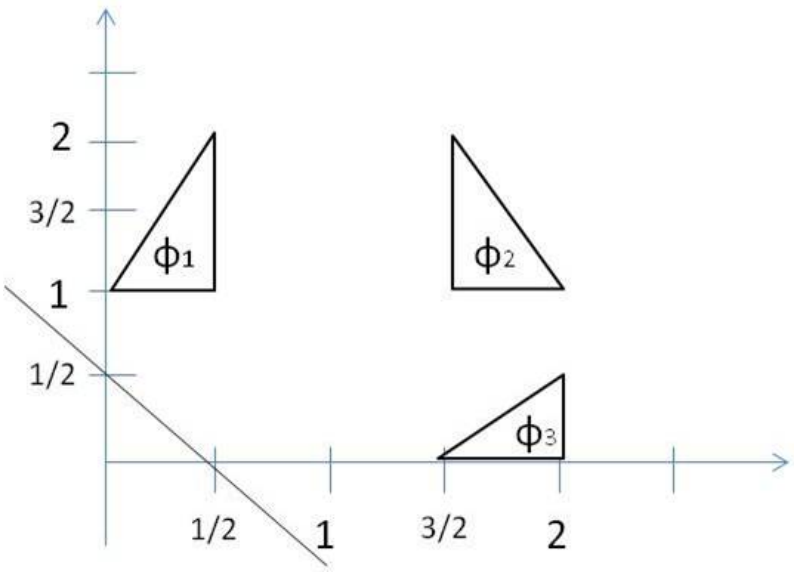

Figura 4.1: Espacio solución del ejemplo.

$$\begin{bmatrix} -2x_1 + x_2 - 1 \leq 0 \\ x_1 - \frac{1}{2} \leq 0 \\ -x_2 + 1 \leq 0 \\ C = 5 \end{bmatrix} \vee \begin{bmatrix} 2x_1 + x_2 - 5 \leq 0 \\ -x_1 + \frac{3}{2} \leq 0 \\ -x_2 + 1 \leq 0 \\ C = 6.5 \end{bmatrix} \vee \begin{bmatrix} -2x_1 + x_2 + \frac{3}{2} \leq 0 \\ x_1 - 2 \leq 0 \\ -x_2 \leq 0 \\ C = 7.5 \end{bmatrix}$$

Gráficamante el espacio soluión del ejemplo se muestra en la figura 4.1.

Consideremos también los siguientes parámetros de entrada:

- Tamaño de la población = 10
- Número de generaciones = 1
- Parametro s para el ranking de la selección = 1.2.
- Datos de las variables: variable real $x_1 \in [0, 2]$, variable entera $x_2 \in \{0, 1, 2\}$, para ambos casos se manejarán 2 cifras de aproximación, esto es q = 2, de lo que se obtiene m=8.

1. Inicialización: Generamos la población inicial como se expuso en el punto 4.2.1, resultando los siguientes cromosomas, de los cuales obtenemos mediante (4.4) el valor real para la variable x_1 y el valor entero de la variable x_2 mediante redondeo del resultado de la ecuación (4.4). Los resultados se muestra en la tabla 4.7.

Individuo	**Cromosoma p/x_1**	**Cromosoma p/x_2**	x_1	x_2
1	[11001000]	[00111110]	1.56	0
2	[01011101]	[10001111]	0.72	1
3	[01100011]	[11000111]	0.77	2
4	[00011100]	[11100000]	0.21	2
5	[11000110]	[11101001]	1.55	2
6	[00100000]	[01011101]	0.25	1
7	[11010100]	[01110011]	1.66	1
8	[01110111]	[00101010]	0.93	0
9	[01101101]	[00011111]	0.85	0
10	[00100011]	[10111101]	0.27	1

Tabla 4.7: Población inicial del ejemplo (4.6)

2. Evaluación: Se evalúa cada individuo primero en la función objetivo F en cada disyunto, ya que recordemos que la función objetivo no depende solamente de las variables x_1, x_2 sino también de los parámetros C_k; las evaluaciones se muestran en la tabla 4.8.

   La seguna evaluación es mediante las restricciones. Se evalua cada individuo en todas las restricciones generales y en las de cada disyunto; la tabla 4.9 muestra la evaluación en las restricciones generales, la tabla 4.10 las del disyunto 1, la tabla 4.11 las del disyunto 2 y la tabla 4.12 las del disyunto 3.

   Ahora, cada restricción se normaliza (tanto en las restricciones generales como

Individuo	**F(Disyunto 1)**	**F(Disyunto 2)**	**F(Disyunto 3)**
1	-13.72	-12.22	-11.22
2	19.36	20.85	21.85
3	41.76	43.26	44.26
4	48.48	49.98	50.98
5	32.4	33.9	34.9
6	25	26.5	27.5
7	8.08	9.58	10.58
8	-6.16	-4.66	-3.66
9	-5.20	-3.70	-2.70
10	24.76	26.26	27.26

Tabla 4.8: Evaluación de la función objetivo, ejemplo(4.6)

en cada disyunto), dividiendo el valor de la restricción en todos los individuos entre el mayor valor de ellos en valor absoluto, de lo que resultan las tablas 4.13, 4.14, 4.15, 4.16.

En las tablas anteriores, los valores negativos se convierten a cero, ya que si es ≤ 0 indica que el individuo es factible en esa restricción. El valor positivo representará por consiguiente la infactibilidad de cada individuo en dicha restricción. En cada disyunto se suma la infactibilidad de cada individuo, como se muestra en la tabla 4.17.

El siguiente paso es aplicar los operadores genéticos, para lo cual es necesario ordenar los individuos en base a su medida de desempeño: El valor de su función objetivo y su infactibilidad. Sabremos que un individuo es factible si en su evaluación de infactibilidad, las restricciones generales y en al menos un disyunto se tiene suma igual a 0. Si es factible en más de un disyunto, al individuo se le asociará el valor de la función objetivo del disyunto en el cual se tiene la mejor evaluación de este (el objetivo es maximizar). Para los individuos factibles su medida de desempeño será el valor de la función objetivo

Restricciones generales			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-1.06	-0.44	-2
2	-1.22	-1.28	-1
3	-2.27	-1.23	0
4	-1.71	-1.79	0
5	-3.05	-0.45	0
6	-0.75	-1.75	-1
7	-2.16	-0.34	-1
8	-0.43	-1.07	-2
9	-0.35	-1.15	-2
10	-0.77	-1.73	-1

Tabla 4.9: Evaluación de las restricciones generales, ejemplo(4.6)

correspondiente al disyunto en el que es factible. A los individuos infactibles consideramos otra medida de desempeño que es la relación $\frac{F}{Infac}$ perteneciente al disyunto en el que el individuo tiene menor infactibilidad, ver ecuación (4.8) y tabla 4.18.

3. Selección: Se asigna la probabilidad de *ranking* (4.5) para realizar la selección de los individuos que se cruzarán, ver tabla 4.19.

   Se generan valores aleatorios entre 0-1 para seleccionar a los individuos que se cruzarán, ver tabla 4.20.

4. Cruzamiento: El cruzamiento se realiza en un punto solamente como lo vimos en el capítulo 3, y se genera aleatoriamente los puntos de cruce; como cada individuo depende de dos variable generamos dos puntos de cruce por pareja como se muestra en la tabla 4.21. De donde al realizar el cruzamiento resulta la tabla 4.22 para la variable 1 y la tabla 4.23 para la variable 2.

   Su representación en valores reales y enteros se muestran en la tabla 4.24.

Disyunto 1			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-4.12	1.06	1
2	-1.44	0.22	0
3	-0.54	0.27	-1
4	0.58	-0.29	-1
5	-2.1	1.05	-1
6	-0.50	-0.25	0
7	-3.32	1.16	0
8	-2.86	0.43	1
9	-2.7	0.35	1
10	-0.54	-0.23	0

Tabla 4.10: Evaluación restricciones disyunto 1, ejemplo(4.6)

5. Mutación: Para estas pruebas no aplicaremos mutación.

6. Nueva Población: Para elegir la nueva población primero tenemos que evaluar a los hijos, siguiendo los pasos anteriores, de donde se obtiene la tabla 4.25. Con la población inicial y los hijos se genera una población de 16 individuos los cuales se ordenan primero los factibles y despues los infactibles (en negritas) en base a su función de mérito, obteniendo el resultado que se muestra en la tabla 4.26.

   Se elige el 80 % de individuos fáctibles y el 20 % de infactibles, o lo que es lo mismo los mejores 8 individuos factibles y los mejores 2 infactibles, (esto porque la población es de tamaño 10). De donde la nueva población la conforman los individuos: 6, 10, 11, 12, 7, 9, 13, 8, 15, 16.

   Aqui termina una iteración (o generación) del algoritmo donde obtuvimos que el mejor individuo para esta iteración es el individuo **6** que es factible en el disyunto **1** con un valor de mérito de **25**.

Disyunto 2			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-1.88	-0.06	1
2	-2.56	0.78	0
3	-1.46	0.73	-1
4	-2.58	1.29	-1
5	0.10	-0.05	-1
6	-3.50	1.25	0
7	-0.68	-0.16	0
8	-3.14	0.57	1
9	-3.30	0.65	1
10	-3.46	1.23	0

Tabla 4.11: Evaluación restricciones disyunto 2, ejemplo(4.6)

Disyunto 3			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-1.62	-0.44	0
2	1.06	-1.28	-1
3	1.96	-1.23	-2
4	3.08	-1.79	-2
5	0.40	-0.45	-2
6	2	-1.75	-1
7	-0.82	-0.34	-1
8	-0.36	-1.07	0
9	-0.20	-1.15	0
10	1.96	-1.73	-1

Tabla 4.12: Evaluación restricciones disyunto 3, ejemplo(4.6)

Restricciones generales			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-3.03	-1.29	-2
2	-3.49	-3.76	-1
3	-6.49	-3.62	0
4	-4.89	-5.26	0
5	-8.71	-1.32	0
6	-2.14	-5.15	-1
7	-6.17	-1	-1
8	-1.23	-3.15	-2
9	-1	-3.38	-2
10	-2.20	-5.09	-1

Tabla 4.13: Normalización de la evaluación restricciones generales, Ejemplo(4.6)

Disyunto 1			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-7.19	0.91	1
2	-2.48	0.19	0
3	-0.93	0.23	-1
4	1	-0.25	-1
5	-3.62	0.91	-1
6	-0.86	-0.22	0
7	-5.72	1	0
8	-4.93	0.37	1
9	-4.66	0.30	1
10	-0.93	-0.20	0

Tabla 4.14: Normalización de la evaluación disyunto 1, Ejemplo(4.6)

Disyunto 2			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-18.80	-0.05	1
2	-25.60	0.60	0
3	-14.60	0.57	-1
4	-25.80	1	-1
5	1	-0.04	-1
6	-35	0.97	0
7	-6.80	-0.12	0
8	-31.40	0.44	1
9	-33	0.50	1
10	-34.60	0.95	0

Tabla 4.15: Normalización de la evaluación disyunto 2, Ejemplo(4.6)

Disyunto 3			
Individuo	**restricción 1**	**restricción 2**	**restricción 3**
1	-0.53	-1.29	0
2	0.34	-3.76	-1
3	0.64	-3.62	-2
4	1	-5.26	-2
5	0.13	-1.32	-2
6	0.65	-5.15	-1
7	-0.27	-1	-1
8	-0.12	-3.15	0
9	-0.06	-3.38	0
10	0.64	-5.09	-1

Tabla 4.16: Normalización de la evaluación disyunto 3, Ejemplo(4.6)

Infactibilidad				
Individuo	**Restricciones grals.**	**Disyunto 1**	**Disyunto 2**	**Disyunto 3**
1	0	1.91	1	0
2	0	0.19	0.60	0.34
3	0	0.23	0.57	0.64
4	0	1	1	1
5	0	0.91	1	0.13
6	0	0	0.97	0.65
7	0	1	0	0
8	0	1.37	1.44	0
9	0	1.30	1.50	0
10	0	0	0.95	0.64

Tabla 4.17: Infactibilidad polación inicial, ejemplo(4.6)

Ordenamiento					
Individuo	**Infact.**	**Disyunto**	**F**	ψ	**Orden**
1	0	3	-11.22	-11.22	6
2	0.19	1	19.36	101.89	9
3	0.23	1	41.76	181.56	8
4	1	3	50.98	50.98	10
5	0.13	3	34.9	268.46	7
6	0	1	25	25	1
7	0	3	10.58	10.58	3
8	0	3	-3.66	-3.66	5
9	0	3	-2.70	-2.70	4
10	0	1	24.76	24.76	2

Tabla 4.18: Ordenamiento por criterio, ejemplo(4.6)

Probabilidad ranking				
Individuo	ψ	**Índice ranking**	$Prob_{rank}$	**Rango**
6	25	9	0.150	$(0 - 0.150)$
10	24.76	8	0.139	$(0.150 - 0.289)$
7	10.58	7	0.128	$(0.289 - 0.417)$
9	-2.70	6	0.117	$(0.417 - 0.533)$
8	-3.66	5	0.106	(0.5330.639)
1	-11.22	4	0.094	$(0.639 - 0.733)$
5	268.46	3	0.083	$(0.733 - 0.817)$
3	181.56	2	0.072	$(0.817 - 0.889)$
2	101.89	1	0.061	$(0.889 - 0.950)$
4	50.98	0	0.050	$(0.950 - 1)$

Tabla 4.19: Probabilidad de ranking, ejemplo(4.6)

Padres seleccionados			
P_{padre}	P_{madre}	**Padre**	**Madre**
0.087	0.355	6	7
0.441	0.902	9	2
0.768	0.757	5	5

Tabla 4.20: Padres seleccionados, ejemplo(4.6)

Punto de cruce	
Variable 1	**Variable 2**
5	2
6	7
4	5

Tabla 4.21: Punto de cruce, ejemplo(4.6)

Cruzamiento Variable 1			
Padre	**Madre**	**Hijo 1**	**Hijo 2**
[**00100**000]	[11010**100**]	[00100100]	[11010000]
[**01101**01]	[01011**01**]	[01101101]	[01011101]
[**1100**0110]	[1100**0110**]	[11000110]	[11000110]

Tabla 4.22: Cruzamiento variable 1, Ejemplo(4.6)

Cruzamiento Variable 2			
Padre	**Madre**	**Hijo 1**	**Hijo 2**
[**01**011101]	[01**110011**]	[01110011]	[01011101]
[**0001111**1]	[1000111**1**]	[00011111]	[10001111]
[**11101**001]	[11101**001**]	[11101001]	[11101001]

Tabla 4.23: Cruzamiento variable 2, Ejemplo(4.6)

Hijos		
Hijo	x_1	x_2
11	0.28	1
12	1.63	1
13	0.85	0
14	0.72	1
15	1.55	2
16	1.55	2

Tabla 4.24: Hijos, Ejemplo(4.6)

F. Mérito Hijos		
Hijo	**F. Mérito**	**Disyunto**
11	24.61	1
12	10.92	3
13	-2.76	3
15	274	2
16	274	2
14	94.92	1

Tabla 4.25: F. Mérito hijos, ejemplo (4.6)

1er. Generación			
Individuo	**F. Mérito**	**Individuo**	**F. Mérito**
6	25	**15**	**274**
10	24.76	**16**	**274**
11	24.61	**5**	**268.73**
12	10.92	**3**	**179.41**
7	10.58	**2**	**102.08**
9	-2.70	**14**	**94.92**
13	-2.76	**4**	**50.98**
8	-3.66		
1	-11.22		

Tabla 4.26: Primera Generación, ejemplo(4.6)

1er. Generación		
Individuo	**F. Mérito**	**Individuo**
F. Mérito		
1	[00100000]	[01011101]
2	[00100011]	[10111101]
3	[00100100]	[01110011]
4	[11000110]	[11101001]
5	[11010100]	[01110011]
6	[01101101]	[00011111]
7	[01101101]	[00011111]
8	[01110111]	[00101010]
9	[11000110]	[01110101]
10	[11000110]	[01110101]

Tabla 4.27: Nueva Generación, ejemplo(4.6)

Capítulo 5

Resultados experimentales

En el presente capítulo se exponen los resultados obtenidos en la experimentación del método propuesto; en la sección 5.1 se presentan los problemas resueltos y en la sección 5.2 los resultados obtenidos.

5.1 Problemas resueltos

- **Ejemplo 1**

  El primer ejemplo fue formulado originalmente como un problema MINLP convexo [12], la solución optima es $Y = (Falso, Verdadero, Falso)$, $x^* = (1, 1)$ y $Z^* = 3.5$. La formulación como problema de programación disyunta es la siguiente:

$$\begin{aligned} \min \quad & \mathrm{Z} = \mathrm{C} + \mathrm{x}_1^2 + x_2^2 \\ \text{sujeto a:} \quad & (\mathrm{x}_1 - 2)^2 - x_2 \leq 0 \end{aligned}$$

$$\begin{bmatrix} Y_1 \\ x_1 - 2 \geq 0 \\ x_1 - x_2 \leq 4 \\ C = 1 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ x_1 - x_2 \leq 0 \\ x_1 - 1 \geq 0 \\ x_2 - 1 \geq 0 \\ C = 1.5 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ x_1 - x_2 \leq 4 \\ x_1 + x_2 \geq 3 \\ x_1 - 1 \geq 0 \\ C = 0.5 \end{bmatrix}$$

$$\Omega(Y) = Verdadero$$

Para correr el algoritmo se transforman todas las desigualdades a la forma ≤ 0, y la función objetivo a maximizar $-Z = -C - x_1^2 - x_2^2$, el ejemplo 1 queda de la siguiente manera:

$$\text{maximizar} \quad \text{-Z = -C - } x_1^2 - x_2^2$$
$$\text{sujeto a:} \quad (x_1 - 2)^2 - x_2 \leq 0$$

$$\begin{bmatrix} Y_1 \\ 2 - x_1 \leq 0 \\ x_1 - x_2 - 4 \leq 0 \\ C = 1 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ x_1 - x_2 \leq 0 \\ 1 - x_1 \leq 0 \\ 1 - x_2 \geq 0 \\ C = 1.5 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ x_1 - x_2 - 4 \leq 0 \\ -x_1 - x_2 + 3 \leq 0 \\ 1 - x_1 \leq 0 \\ C = 0.5 \end{bmatrix}$$

$$\Omega(Y) = Verdadero$$

Ejemplo 2 La solución optima para este ejemplo es: $Y = (Verdadero, Falso, Falso)$, $x^* = (3.0, 1.0)$ y $Z^* = 6$

$$\min \quad \text{Z = C + } (x_1 - 2)^2 + (x_2 - 1)^2$$
$$\text{sujeto a:}$$

$$\begin{bmatrix} Y_1 \\ (x_1 - 4)^2 \leq 0 \\ -(x_1 - 2) + x_2 \leq 0 \\ C = 5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2x_1 + x_2 - 4 \leq 0 \\ 2 - x_2 \leq 0 \\ C = 7 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ (x_1 - 4)^2 - x_2 \leq 0 \\ x_1 - x_2 \leq 0 \; C = 9 \end{bmatrix}$$

$$0 \leq x_1, \qquad x_2 \leq 5, \qquad 0 \leq C$$
$$\mathrm{Y}_j \in \{verdadero, falso\}, j = 1, 2, 3$$

Para este ejemplo ya todas las desigualdades se encuentran en la forma $\leq$ solamente la función objetivo se transforma de minimizar a maximizar, quedando: $max\ -Z = -C - (x_1 - 2)^2 - (x_2 - 1)^2$.

Ejemplo 3 La solución optima para este ejemplo es: $Y = (Falso, Verdadero, Falso)$, $x^* = (1.5, 2)$ y $Z^* = 7$

$$\max \quad \mathrm{Z} = \mathrm{C} - \mathrm{x}_1 + x_2$$
$$\text{sujeto a:} \qquad \mathrm{x}_1 + x_2 \geq \tfrac{1}{2}$$

$$\begin{bmatrix} Y_1 \\ -2x_1 + x_2 \leq 1 \\ x_1 \leq \frac{1}{2} \\ x_2 \geq 1 \\ C = 5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2x_1 + x_2 \leq 5 \\ x_1 \geq \frac{3}{2} \\ x_2 \geq 1 \\ C = 6.5 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ x_1 - x_2 \geq \frac{3}{2} \\ x_1 \leq 2 \\ x_2 \geq 0 \\ C = 7.5 \end{bmatrix}$$

$$\mathrm{x}_1, x_2 \geq 0, \quad x_1, x_2 \leq 2, \quad c \geq 0$$
$$\mathrm{Y}_j \in \{verdadero, falso\}, j = 1, 2, 3$$

Para correr el algoritmo, se transforman todas las desigualdades a la forma ≤ 0, y la función objetivo no sufre cambio ya que se desea maximizar, el ejemplo 3 queda de la siguiente manera:

$$\max \quad \mathrm{Z} = \mathrm{C} - \mathrm{x}_1 + x_2$$

$$\text{sujeto a:} \qquad \tfrac{1}{2} - \mathrm{x}_1 - x_2 \leq 0$$

$$-\mathrm{x}_1 \leq 0, \quad -x_2 \leq 0, \quad x_1 - 2 \leq 0, \quad x_2 - 2 \leq 0$$

$$\begin{bmatrix} Y_1 \\ -2x_1 + x_2 - 1 \leq 0 \\ x_1 - \frac{1}{2} \leq 0 \\ 1 - x_2 \leq 0 \\ C = 5 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2x_1 + x_2 - 5 \leq 0 \\ \frac{3}{2} - x_1 \leq 0 \\ 1 - x_2 \leq 0 \\ C = 6.5 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ \frac{3}{2} - x_1 + x_2 \leq 0 \\ x_1 - 2 \leq 0 \\ -x_2 \leq 0 \\ C = 7.5 \end{bmatrix}$$

$$\mathrm{Y}_j \in \{verdadero, falso\}, j = 1, 2, 3$$

Ejemplo 4

Como Ejemplo 4 resolvimos el problema 3 solo que con x_1 como variable real y x_2 como variable entera, donde la solucón es $Y = (Falso, Verdadero, Falso)$, $x^* = (1.5, 2)$ y $Z^* = 70.5$.

5.2 Experimentación

Para realizar la experimentación, resolvimos los 4 ejemplos planteados, realizando variaciones en el número de generaciones y el número de población asi como también en el método de selección de la nueva generación, en primera instancia la nueva generación se eligio 80 % los mejores factibles, 20 % de infactibles en base a su infactibilidad y en segunda instancia en base al criterio ψ. Cabe mencionar que en todos los casos el algoritmo paro con un número de generaciones fijo. Se probo el algoritmo para variables reales y enteras, dejando como trabajo a futuro probar las variables discretas y binarias.

Los resultados obtenidos fuerón los siguientes:

- Ejemplo 1:

  Primero se resolvió realizando la selección de la siguiente generación 80 % los mejores factibles, 20 % menos infactibles.

  Población=50. Se muestra en la tabla A.1.

  Población=75. Se muestra en la tabla A.2.

  Población=100. Se muestra en la tabla A.3.

  Población=150. Se muestra en la tabla A.4.

  Después se resolvió realizando la selección de la siguiente generación 80 % los mejores factibles, 20 % de infactibles en base a la razón de infactibilidad ψ.

  Población=50. Se muestra en la tabla A.5.

  Población=75. Se muestra en la tabla A.6.

  Población=100. Se muestra en la tabla A.7.

  Población=150. Se muestra en la tabla A.8.

- Ejemplo 2:

  Primero se resolvió realizando la selección de la siguiente generación 80 % los mejores factibles, 20 % menos infactibles.

  Población=50. Se muestra en la tabla A.9.

  Población=75. Se muestra en la tabla A.10.

  Población=100. Se muestra en la tabla A.11.

  Población=150. Se muestra en la tabla A.12.

  Después se resolvió realizando la selección de la siguiente generación 80 % los mejores factibles, 20 % de infactibles en base a la razón de infactibilidad r.

  Población=50. Se muestra en la tabla A.13.

  Población=75. Se muestra en la tabla A.14.

  Población=100. Se muestra en la tabla A.15.

  Población=150. Se muestra en la tabla A.16.

Ejemplo 3:

Se resolvió realizando la selección de la siguiente generación 80 % los mejores factibles, 20 % de infactibles en base a la razón de infactibilidad ψ.

Población=50. Se muestra en la tabla B.1.

Población=75. Se muestra en la tabla B.2.

Población=100. Se muestra en la tabla B.3.

Población=150. Se muestra en la tabla B.4.

Ejemplo 4:

Se resolvió realizando la selección de la siguiente generación 80 % los mejores factibles, 20 % de infactibles en base a la razón de infactibilidad r.

Población=50. Se muestra en la tabla B.5.

Población=75. Se muestra en la tabla B.6.

Población=100. Se muestra en la tabla B.7.

Capítulo 6

Conclusiones y trabajo a futuro

6.1 Conclusiones

El presente trabajo se realizó para proponer un nuevo método de solución a probemas de programación disyunta, ya que como se menciona en el Capitulo 2 existen diversos métodos que dan solución a este tipo de problemas, más sin embargo en todos los casos se realizaban transformaciones al problema original, por esta razón nuestro objetivo principal era dar solución a los problemas de programación disyunta mediante su formulación original basandonos en algoritmos geneticos.

En el presente trabajo nos centramos en problemas de programación disyunta con funcion objetivo y restricciones lineales, y problemas que solo tienen un conjunto de disyuntos.

Probamos el funcionamiento del método propuesto mediante 4 ejemplos, en los cuales los primeros 3 tienen variables reales y el 4 tiene una variable real y una entera, comparando los resultados de los ejemplos 1 y 2 obtenidos por [11].

Después de la experimentación realizada con los ejemplos mencionados anteriormente llegamos a las sigiuentes conclusiones:

El algoritmo da muy buenas soluciones en buenos tiempos, y como se puede comparar los mejores resultados obtenidos fueron mediante la selección basada en el criterio ψ.

Utilizando el criterio ψ el número de generaciones necesarias para dar buenas soluciones disminuye considerablemente en comparación con el criterio $80 - 20$

También podemos concluir que el método utilizado por [11] obtiene soluciones optimas en cuanto el método propuesto en el presente trabajo no obtiene soluciones optimas en todos los casos, mas sin embargo las soluciones obtenidas son muy cercanas a las optimas en cuanto la población es mas grande, por otro lado los resultados obtenidos por [11] son optimos relajando el problema original.

6.2 Trabajo a futuro

Complementar el algoritmo para variables discretas y binarias, asi como problemas en los que se involucren conjuntos de disyuntos.

Posibilidad de adecuar el codigo a una interfaz amigable con el usuario.

Probar el algoritmo con problemas con mas variables.

En el proceso de la determinación de infactibilidades para los elementos de la población se recomienda el estudio de funciones de agregación tipo normas triangulares que puede proporcionar un proceso mas adecuado para este tipo de problema.

Apéndice A

Tablas de Resultados: Ejemplos 1 y 2

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[1.0009, 1.5000]	$Z = 4.75182$	t=57.633
75	x^*=[1.0002, 1.0314]	$Z = 3.5636$	t=86.330
100	x^*=[1.0001, 1.0313]	$Z = 3.5636$	t=86.330
150	x^*=[1.0000, 1.1875]	$Z = 3.9102$	t=163.22

Tabla A.1: Resultados experimentales del ejemplo 1 con Polación de 50

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[1.0010, 1.5000]	$Z = 4.7520$	$t = 104.20$
75	x^*=[1.0000, 1.1875]	$Z = 3.9100$	$t = 152.13$
100	x^*=[1.0004, 1.0313]	$Z = 3.5640$	$t = 200.29$
150	x^*=[1.0009, 1.0313]	$Z = 3.5652$	$t = 292.51$

Tabla A.2: Resultados experimentales del ejemplo 1 con Polación de 75

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[1.0000, 1.1875]	$Z = 3.9100$	$t = 159.54$
75	x^*=[1.0001, 10313.]	$Z = 3.5636$	$t = 234.89$
100	x^*=[1.0000, 1.0110]	$Z = 3.5200$	$t = 314.37$
150	x^*=[1.0000, 1.0310]	$Z = 3.5635$	$t = 455.76$

Tabla A.3: Resultados experimentales del ejemplo 1 con Polación de 100

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[1.0000, 1.0000]	$Z = 3.5030$	$t = 329.56$
75	x^*=[1.0000, 1.0300]	$Z = 3.5630$	$t = 465.07$
100	x^*=[1.0003, 1.0310]	$Z = 3.5640$	$t = 607.62$
150	x^*=[1.0000, 1.0310]	$Z = 3.5630$	$t = 876.90$

Tabla A.4: Resultados experimentales del ejemplo 1 con Polación de 150

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.1800, 1.2500]	$Z = 4.4770$	$t = 36.272$
50	x^*=[1.0007, 1.0012]	$Z = 3.5030$	$t = 86.557$
75	x^*=[1.0020, 1.0040]	$Z = 3.5141$	$t = 110.13$
100	x^*=[1.0007, 1.0020]	$Z = 3.5058$	$t = 132.14$
150	x^*=[1.0007, 1.1250]	$Z = 3.7677$	$t = 265.47$

Tabla A.5: Resultados experimentales del ejemplo 1 con Polación de 50, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.0029, 1.0100]	$Z = 3.5259$	$t = 74.222$
50	x^*=[1.0010, 1.0032]	$Z = 3.5092$	$t = 121.33$
75	x^*=[1.0015, 1.0154]	$Z = 3.5349$	$t = 223.10$
100	x^*=[1.0469, 1.0600]	$Z = 3.7254$	$t = 334.01$

Tabla A.6: Resultados experimentales del ejemplo 1 con Polación de 75, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	$x^* = [1.0007, 1.0012]$	$Z = 3.5039$	$t = 85.806$
50	$x^* = [1.0007, 1.0012]$	$Z = 3.5039$	$t = 268.78$
75	$x^* = [1.0007, 1.0012]$	$Z = 3.5039$	$t = 312.58$

Tabla A.7: Resultados experimentales del ejemplo 1 con Polación de 100, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
10	x^*=[1.0029, 1.0090]	$Z = 3.5240$	$t = 107.38$
20	x^*=[1.0007, 1.0012]	$Z = 3.5038$	$t = 213.33$

Tabla A.8: Resultados experimentales del ejemplo 1 con Polación de 150, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[3.3500, 1.0000]	$Z = 6.0710$	$t = 27.564$
75	x^*=[3.0470, 0.9960]	$Z = 6.0960$	$t = 36.660$
100	x^*=[3.0110, 0.9990]	$Z = 6.0230$	$t = 52.853$
150	x^*=[3.1870, 0.9920]	$Z = 6.4100$	$t = 77.214$

Tabla A.9: Resultados experimentales del ejemplo 2 con Polación de 50

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[3.3780, 1.0150]	$Z = 6.8990$	$t = 47.216$
75	x^*=[3.0120, 1.0010]	$Z = 6.0243$	$t = 68.364$
100	x^*=[3.0470, 0.9750]	$Z = 6.0973$	$t = 88.468$
150	x^*=[3.0238, 0.9974]	$Z = 6.0481$	$t = 285.33$

Tabla A.10: Resultados experimentales del ejemplo 2 con Polación de 75

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[3.0230, 1.0150]	$Z = 6.0484$	$t = 140.370$
75	x^*=[3.0238, 0.9992]	$Z = 6.0481$	$t = 159.87$
100	x^*=[3.0470, 1.0001]	$Z = 6.0967$	$t = 289.89$
150	x^*=[3.0120, 0.9850]	$Z = 6.0240$	$t = 453.10$

Tabla A.11: Resultados experimentales del ejemplo 2 con Polación de 100

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[3.0230, 1.0157]	$Z = 6.0484$	$t = 318.13$
75	x^*=[3.0470, 0.9330]	$Z = 6.1010$	$t = 474.17$
100	x^*=[3.0941, 0.9339]	$Z = 6.2014$	$t = 607.16$
150	x^*=[3.0472, 1.0157]	$Z = 6.0969$	$t = 865.23$

Tabla A.12: Resultados experimentales del ejemplo 2 con Polación de 150

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[3.03, 1.00]	$Z = 6.071$	$t = 27.564$
75	x^*=[3.05, 1.00]	$Z = 6.096$	$t = 38.660$
100	x^*=[3.01, 0.99]	$Z = 6.023$	$t = 52.853$
150	x^*=[3.19, 0.99]	$Z = 6.410$	$t = 77.214$

Tabla A.13: Resultados experimentales del ejemplo 2 con Polación de 50, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[3.38, 1.01]	$Z = 6.899$	$t = 47.216$
75	x^*=[3.01, 1.00]	$Z = 6.024$	$t = 68.364$
100	x^*=[3.05, 0.98]	$Z = 6.097$	$t = 88.468$
150	x^*=[3.02, 1.00]	$Z = 6.048$	$t = 285.33$

Tabla A.14: Resultados experimentales del ejemplo 2 con Polación de 75, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
50	$x^* = [3.02, 1.01]$	$Z = 6.048$	$t = 140.37$
75	$x^* = [3.05, 1.00]$	$Z = 6.097$	$t = 159.87$
100	$x^* = [3.01, 1.00]$	$Z = 6.048$	$t = 289.89$
150	$x^* = [3.01, 0.99]$	$Z = 6.024$	$t = 453.10$

Tabla A.15: Resultados experimentales del ejemplo 2 con Polación de 100, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
50	x^*=[3.04, 1.01]	$Z = 6.097$	$t = 318.13$
75	x^*=[3.09, 0.93]	$Z = 6.201$	$t = 474.17$
100	x^*=[3.05, 0.93]	$Z = 6.101$	$t = 607.16$
150	x^*=[3.02, 1.01]	$Z = 6.048$	$t = 857.15$

Tabla A.16: Resultados experimentales del ejemplo 2 con Polación de 150, en base a ψ

Apéndice B

Tablas de Resultados: Ejemplos 3 y 4

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.576, 1.848]	$Z = 6.772$	$t = 37.106$
50	x^*=[1.541, 1.918]	$Z = 6.877$	$t = 64.324.$
75	x^*=[1.510, 1.979]	$Z = 6.969$	$t = 86.38$
100	x^*=[1.501, 1.998]	$Z = 6.997$	$t = 125.783$

Tabla B.1: Resultados experimentales del ejemplo 3 con Polación de 50, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.545, 1.91]	$Z = 6.865$	$t = 83.152$
50	x^*=[1.528, 1.944]	$Z = 6.916$	$t = 112.364$
75	x^*=[1.519, 1.962]	$Z = 6.943$	$t = 145.413$
100	x^*=[1.509, 1.982]	$Z = 6.973$	$t = 210.546$

Tabla B.2: Resultados experimentales del ejemplo 3 con Polación de 75, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.530, 1.940]	$Z = 6.910$	$t = 102.454$
50	x^*=[1.512, 1.976]	$Z = 6.964$	$t = 234.633$
75	x^*=[1.509, 1.982]	$Z = 6.973$	$t = 315.562$
100	x^*=[1.509, 1.982]	$Z = 6.973$	$t = 465.689$

Tabla B.3: Resultados experimentales del ejemplo 3 con Polación de 100, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.511, 1.978]	$Z = 6.967$	$t = 167.842$
50	x^*=[1.509, 1.982]	$Z = 6.973$	$t = 357.568$
75	x^*=[1.501, 1.998]	$Z = 6.997$	$t = 475.680$
100	x^*=[1.500, 1.999]	$Z = 6.999$	$t = 612.923$

Tabla B.4: Resultados experimentales del ejemplo 3 con Polación de 150, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.542, 0]	$Z = 5.958$	$t = 38.549$
50	x^*=[1.532, 1]	$Z = 5.698$	$t = 95.734$
75	x^*=[0.5, 2]	$Z = 6.5$	$t = 127.450$
100	x^*=[1.501, 1]	$Z = 5.999$	$t = 283.504$

Tabla B.5: Resultados experimentales del ejemplo 4 con Polación de 50, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[0.419, 1]	$Z = 5.581$	$t = 74.245$
50	x^*=[1.528, 1]	$Z = 5.972$	$t = 119.647$
75	x^*=[1.501, 1]	$Z = 5.999$	$t = 232.579$
100	x^*=[1.500, 1]	$Z = 6.000$	$t = 331.893$

Tabla B.6: Resultados experimentales del ejemplo 4 con Polación de 75, en base a ψ

Generaciones	x^*	Z	**Tiempo en segundos**
20	x^*=[1.509, 1]	$Z = 5.991$	$t = 108.345$
50	x^*=[1.510, 1]	$Z = 5.990$	$t = 327.693$
75	x^*=[0.5, 2]	$Z = 6.500$	$t = 529.892$
100	x^*=[1.5, 2]	$Z = 7.000$	$t = 683.457$

Tabla B.7: Resultados experimentales del ejemplo 4 con Polación de 100, en base a ψ

Bibliografía

[1] Aldo Vecchietti, S. L. y I. E. Grossmann, «Modeling of Discrete/Continuos Optimization Problems: Characterization and Formulation of Disjunctions and their Relaxations», *Computers and Chemical Engineering*, **27**, 2003.

[2] Andramonov, M., «Solving Problems with Min-Type Functions by Disjunctive Programming», *Global Optimization*, **24**, 2002.

[3] Araujo, L. y C. Cervigon, *Algoritmos Evolutivos. Un enfoque práctico*, primera edición, RA-MA Editorial, Madrid, España, 2009.

[4] Balas, E., «Disjunctive Programming and a hierarchy of relaxations for discrete optimization problems», *SIAM journal on Algebraic and Discrete Methods*, **6**, 1985.

[5] Balas, E. y M. Perregaard, «Lift-and-Project for Mixed 0-1 Programming: Recent Progress», *Discrete Applied Mathematics*, **123**, 2002.

[6] Björkqvist, J. y T. Westerlund, «Parallel Solution of disjunctive MINLP problems», *Cem. Eng. Comm*, 2000.

[7] Ceria, S. y J. Soares, «Covex programming for disjunctive convex optimization», *Mathematical Programming*, **86**, 1999.

[8] Coello, C. A. C., «Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art», *Computers methods in applied Mechanical and Engineering*, **19**(11), 2002.

[9] ENDO, S. y A. OHUCHI, «Disjunctive Form Concept Learning System based on Genetic Algorithm», *IEEE International Conference on Systems man and cybertetics*, **5**, 1995.

[10] GROSSMANN, I. E. y S. LEE, «New Algorithms for Nonlinear Generalized Disjunctive Programming», *Computers and Chemical Engineering*, **24**(9), 2000.

[11] GROSSMANN, I. E. y S. LEE, «Generalized disjunctive programming: nonlinear convex hull relaxation and algoritms», *Computational optimization and applications*, **26**, 2003.

[12] LEE, S. y I. E. GROSSMANN, «Global optimization of nonlinear generalized disjunctive programming mith bilinear equality constraints: applications to process networks», *Computers and Chemical Engineering*, **27**, 2003.

[13] LIANG PI, Y. P. y L. SHI, «Hybrid Nested Partitions and Mathematical Programming Approach and Its Aplications», *IEEE Transactions on Automation Science and Engineering*, **5**(4), págs. 573–586, 2008.

[14] MELNIKOV, B. F., «Multiheuristic approach to discrete optimization problems», *Cybernetics and Systems Analysis*, **42**(3), págs. 335–341, 2006.

[15] MEZURA-MONTES, E., *Constraint-Handling in evolutionary Optimization*, primera edición, Springer-Verlag, Heidelberg, Germany, 2009.

[16] MICHAEL JÜNGER, T. L., *50 years of integer programming*, primera edición, Amazon, Aussois, France, 2008.

[17] MICHALEWICZ, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, tercera edición, Springer, Charlotte, USA, 1996.

[18] RAMAN, R. y I. E. GROSSMAN, «Modelling and computational techniques for logic based integer programming», *Computers and Chemical Engineering*, **18**, 1994.

[19] ÓSCAR LEONEL CHACÓN MONDRAGÓN, «Algoritmos Genéticos», Impreso para clase .ªlgoritmos Genéticos"de PISIS 2009, 2009.

[20] SMAUS, J.-G., «On Boolean Functions Encodable as a Single Linear Pseudo-Boolean Constraint», CPAIOR´07, Volume 4510 of LNCS, Springer Verlag, pp 288-302, 2007, 2007.

[21] STUBBS, R. A. y S. MEHRUTRA, «A branch and cut method for 0-1 mixed convex programming», *Mathematical Programming*, **86**(3), págs. 515–532, 1999.

[22] TOSCANO, G. y C. A. C. COELLO, «A Constraint-Handling Mechanism for Particle Swarm Optimization», *IEEE Congress on Evolutionary Computation*, **2**, 2004.

[23] TÜRKAY, M. y I. E. GROSSMANN, «Logic-based MINLP algorithms for the optimal synthesis of process networks», *Computers and Chemical Engenieering*, **20**(8), págs. 959–978, 1996.

[24] VECCHIETTI, A. y I. E. GROSSMAN, «Modeling issues and implementation of language for disjunctive programming.», *Computers and Chemical Engineering*, **24**, 2000.

Ficha autobiográfica

Perla Cecilia Hernández Lara

Candidato para el grado de Maestro en Ingeniería
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Solución de problemas de programación disyunta mediante optimización metaheurística

Estudie la Licenciatura en Matemáticas en la Facultad de Ciencias Físico Mateáticas (FCFM) de la Universidad Autónoma de Nuevo León (UANL) en el periodo Agosto 2003-Diciembre 2007. En Enero del 2008 ingrese al Posgrado en Ingenieria de Sistemas (PISIS) en la Facultad de Ingenieria Mecánica y Eléctrica (FIME) de la UANL, teniendo como asesor de tesis al Dr. Óscar Leonel Chacón Mondragón, en el PISIS participe en el programa de seminarios exponiendo los avances del presente trabajo en Abril y noviembre del 2009.