UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

ALGORITMOS EVOLUTIVOS MULTIOBJETIVO Y EVALUADORES DE DESEMPEÑO CON EL APOYO DE UN PRODUCTO DE SOFTWARE

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

AARÓN MARTÍN CASTILLO MEDINA

TUTORA

DRA. KATYA RODRÍGUEZ VÁZQUEZ

2017

Resumen

Se proporciona una introducción concisa relativa a la definición, constitución y ejemplificación de los Algoritmos Evolutivos Multiobjetivo **(en inglés Multi-Objective Evolutionary Algorithms ó simplemente M.O.E.A.s)** así como sus evaluaciones de desempeño.

Para poder realizar las operaciones pertinentes se toma como base un entorno gráfico específicamente elaborado para este trabajo, el cual lleva por denominación **M.O.E.A. Software**.
Con base en dicho producto de software se realizan pruebas para analizar y corroborar las medidas de desempeño contempladas en la teoría.

A su vez el producto de software en sí constituye una pieza de divulgación con la finalidad de acercar a los interesados en el estudio de este tipo de técnicas a través de un ambiente dinámico.
Dicho lo anterior, se incluye información relacionada con la construcción y ejecución del programa antes mencionado para guiar a aquéllos que deseen adentrarse en lo concerniente al ámbito técnico del producto.

Agradecimientos

En construcción

Dedicatoria

En construcción

Tabla de Contenido

Capítulo 1
Introducción

El presente documento tiene como meta otorgar un preámbulo preciso relativo a los Algoritmos Evolutivos Multiobjetivo **(en inglés Multi-Objective Evolutionary Algorithms ó simplemente M.O.E.A.s)** haciendo énfasis en su descripción, modelado y análisis mediante evaluadores de desempeño, apoyándose en un programa propiamente creado para este trabajo nombrado **M.O.E.A. Software**.

La motivación detrás de dicho desarrollo se debe a que, desde el punto de vista del autor, la inmersión a este tema con un enfoque pragmático enriquece los comentarios, discusiones y retroalimentaciones suscitadas alrededor de este proyecto; por otra parte la visualización de resultados de manera expedita facilita la comprensión de los elementos que conforman tanto a este documento como al producto de software.

Entonces, concretando ideas, se persiguen principalmente dos objetivos:

- Crear un producto de software que emule las características y funcionalidades de los M.O.E.A.s **(M.O.E.A. Software)** con la finalidad de incentivar a las personas a acercarse y operar con este tipo de técnicas.
- Usando el producto de software antes mencionado, realizar un análisis básico sobre los resultados derivados del uso de los M.O.E.A.s para determinar su comportamiento y desempeño tanto teórico como práctico, tomando como fundamento las métricas desarrolladas para esta finalidad.

Con respecto del primer objetivo, es importante añadir que, si bien el producto de software por sí solo representa un tópico por separado, para fines de este proyecto se le tratará como el medio y no el fin; no obstante se agregan algunos recursos que ilustran su construcción y uso con la finalidad de que quien así lo desee pueda modificar o potenciar su estructura.
Lo anterior se encuentra orientado hacia una perspectiva técnica.

Apuntando al segundo objetivo, para lograr un acercamiento preciso del tema sin saturar de información al lector se lleva a cabo la revisión de los cuatro M.O.E.A.s más representativos, los cuales son:

- V.E.G.A. **(Vector Evaluated Genetic Algorithm)**.
- M.O.G.A. **(Multi-Objective Genetic Algorithm)**.
- S.P.E.A. 2 **(Strength Pareto Evolutionary Algorithm 2)**.
- N.S.G.A. II **(Non-Dominated Sorting Genetic Algorithm II)**.

La selección determinada con anterioridad implica la existencia de un listado extenso de técnicas que si bien son importantes, no son imprescindibles para una primera aproximación por parte del lector.
A pesar de esto, se proporcionarán las referencias correspondientes para que se puedan estudiar adicionalmente.
El procedimiento es similar para con las métricas de desempeño.

Un detalle importante a considerar es que durante el desarrollo del escrito se usarán los términos "lector" y "usuario" **(este último sobre todo en la sección Apéndice)**; para estos fines ambos términos se referirán a la persona que incursione en este escrito independientemente de su curiosidad teórica o práctica.
Análogamente dentro de la parte técnica los términos "programa", "producto" y "producto de software" se considerarán sinónimos, ya que éstos describen a **M.O.E.A Software**.

Recapitulando, la obra completa consta de los siguientes elementos:

- Trabajo Escrito.
- M.O.E.A. Software **(código fuente y ejecutables, para mayores detalles véase la subsección Características Técnicas del Apéndice)**
- Manual Técnico de M.O.E.A. Software **(anexado en un documento aparte)**.

Cada uno de estos elementos funge como un complemento de los demás para garantizar la dispersión adecuada de información sin redundancias innecesarias, además en éstos se proporciona terminología exclusiva que pudiera entorpecer la comprensión del tema si se localizara en otras secciones.

Es menester mencionar que, el Manual Técnico abarca única y exclusivamente el contenido de M.O.E.A Software, indicando al usuario los componentes, relaciones con los demás y el detalle de sus funcionalidades.

En lo concerniente al contenido de la parte escrita, el lector notará la existencia de 4 capítulos adicionales, la Bibliografía y el Apéndice.
En el Capítulo 2 se muestran todos los fundamentos que dan pie a los Algoritmos Evolutivos Multiobjectivo, es decir, los temas que, combinados entre sí originan el tema de estudio principal de este trabajo de tesis.
Debido a que cada uno por sí mismo se considera un tema aparte sólo se tratarán las características necesarias para poder enlazar adecuadamente un tópico con los demás y

así converger apropiadamente al tema en cuestión.

El Capítulo 3 es la sección dedicada enteramente al M.O.E.A., en ésta se encuentran algunos elementos adicionales como son ciertos criterios de selección específicos **(Shared Fitness ó Fitness Compartido)**, evaluadores de desempeño, los algoritmos más representativos mencionados con anterioridad, los conjuntos de pruebas para determinar su desempeño **(M.O.P, Multi-Objective Problem ó Problema Multiobjetivo)** así como una introducción al uso del producto de software.

El Capítulo 4 está dirigido hacia las pruebas y análisis de los algoritmos con el uso del programa tomando como punto de referencia los criterios de desempeño establecidos en el Capítulo 2.

En el Capítulo 5 se localizan las conclusiones y manifiestos del trabajo futuro no sólo con base en los resultados del Capítulo 4, sino en general con los elementos de toda la obra.

Con respecto de la Bibliografía es preciso detallar que también contiene Mesografía ya que una considerable cantidad de las fuentes utilizadas se obtuvo mediante medios electrónicos.

Finalmente el Apéndice recopila las características técnicas alusivas al producto de software, indicando los paquetes y versiones empleadas para la construcción de dicho programa, así como una ligera introducción al diseño del mismo; para referencias técnicas más sucintas es recomendable que se revise el Manual Técnico.

En lo relativo a los conocimientos previos a este trabajo que se deben adquirir, aún tomando en cuenta el hecho de que se abordarán los conceptos necesarios con una explicación suficiente, es recomendable que el lector indague un poco en tópicos relativos a conjuntos, funciones y operaciones elementales con vectores, lo anterior para poder comprender más a profundidad las explicaciones propiciadas.

En lo que concierne al producto de software sólo es necesario un conocimiento básico en el lenguaje de programación Python y por ende en las sentencias elementales de codificación.
Lo anterior es relevante sólo si el usuario decide introducirse en el código fuente proporcionado, ya que de hecho se incluyen programas ejecutables cuya función es la de otorgar un ambiente de software totalmente separado de aspectos técnicos y de esta manera el usuario únicamente se avoque a la parte analítica.

Capítulo 2

Fundamentos

Para comprender la composición y funcionamiento de un M.O.E.A. **(Multi-Objective Evolutionary Algorithm ó Algoritmo Evolutivo Multiobjetivo)** es necesario adentrarse en sus raíces, más en específico éste tiene sus fundamentos principalmente en las siguientes disciplinas:

- Algoritmos Evolutivos
- Optimización Multiobjetivo

Cada una de manera individual comprende por sí mismo una colección de conocimiento, por lo que se explicarán de manera breve pero precisa enfocando los detalles de tal manera que lo que se describa en este capítulo empate con la generalización del M.O.E.A. en el siguiente episodio. A continuación se muestran cada uno estos tópicos:

2.1. Algoritmos Evolutivos

Se denominan Algoritmos Evolutivos al conjunto de heurísticas [1] basadas en procesos progresivos tales como las teorías de Darwin relativas la selección natural [2] y la supervivencia del individuo más apto [3], de hecho citándolo en su *magnum opus* [1] se puede tomar la siguiente frase que engloba todo el procedimiento:

"Es interesante contemplar un enmarañado ribazo cubierto por muchas plantas de varias clases, con aves que cantan en los matorrales, con diferentes insectos que revolotean y con gusanos que se arrastran entre la tierra húmeda, y reflexionar que estas formas, primorosamente construidas, tan diferentes entre sí, y que dependen mutuamente de modos tan complejos, han sido producidas por leyes que obran a nuestro alrededor.

[1] Aproximación de una solución donde es difícil ó imposible hallar una idónea bajo las condiciones originales.

[2] Concepto que corresponde a una colaboración con Alfred Wallace previo al trabajo de Darwin.

[3] Término acuñado por Herber Spencer posterior a la publicación de la obra de Darwin.

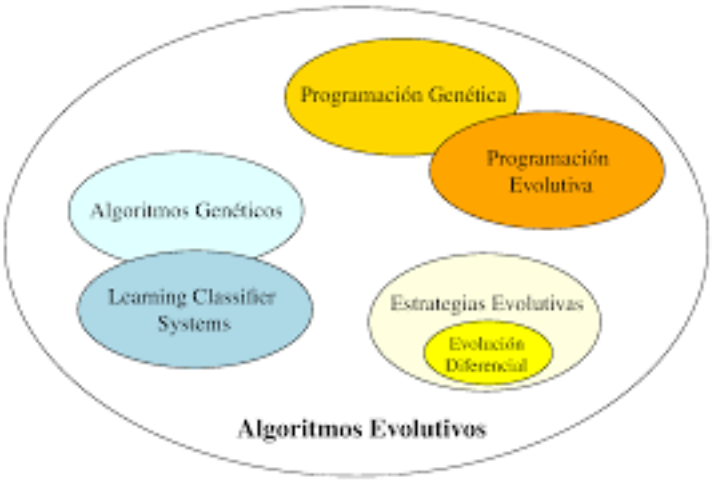


Figura 2.1: Esbozo gráfico de las distintas ramificaciones del Cómputo Evolutivo.

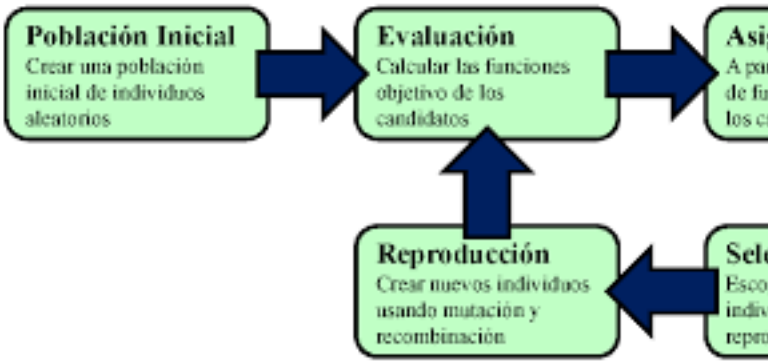
Estas leyes, tomadas en un sentido más amplio, son: la de crecimiento con reproducción; la de herencia, que casi está comprendida en la de reproducción; la de variación por la acción directa e indirecta de las condiciones de vida y por el uso y desuso; una razón del aumento, tan elevada, tan grande, que conduce a una lucha por la vida, y como consecuencia a la selección natural, que determina la divergencia de caracteres y la extinción de las formas menos perfeccionadas".

De esta manera los Algoritmos Evolutivos toman estos conceptos y los trasladan a un entorno cuantificable y tecnológico; así, el objetivo reside entonces en encontrar la solución "óptima" **(aquélla que sea idónea o la que se le aproxime mejor)** ante una problemática establecida, esto a través de mecanismos de selección y recombinación que permitan un adecuado refinamiento y por tanto la obtención de respuestas cada vez más cercanas a las requeridas.

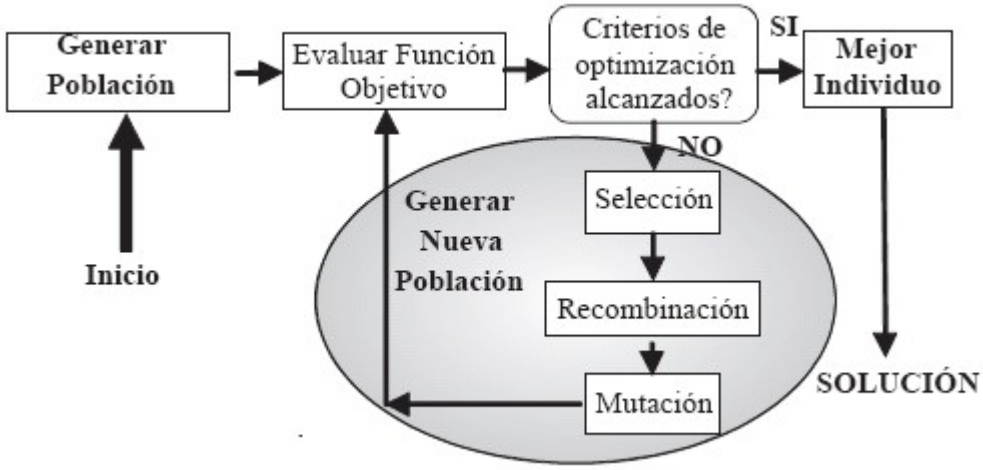
Abordando un poco de contexto histórico eśtas surgen en la década de 1950, sin embargo tuvieron su momento álgido hasta 1960, debido en gran medida a la creación de computadoras cada vez más potentes ya que como el lector podrá notar en secciones posteriores los recursos para llevar a cabo la ejecución de estos algoritmos son no pocos. Por otra parte la concepción de dicho género pertenece a una rama denominada Cómputo Evolutivo en la cual han intervenido varios científicos para su forjamiento, sólo por mencionar algunos se tiene a Lawrence Jerome Fogel, John Henry Holland e Ingo Rechenberg.

A su vez los Algoritmos Evolutivos constituyen en sí una considerable rama de especialidades como se puede apreciar a continuación:

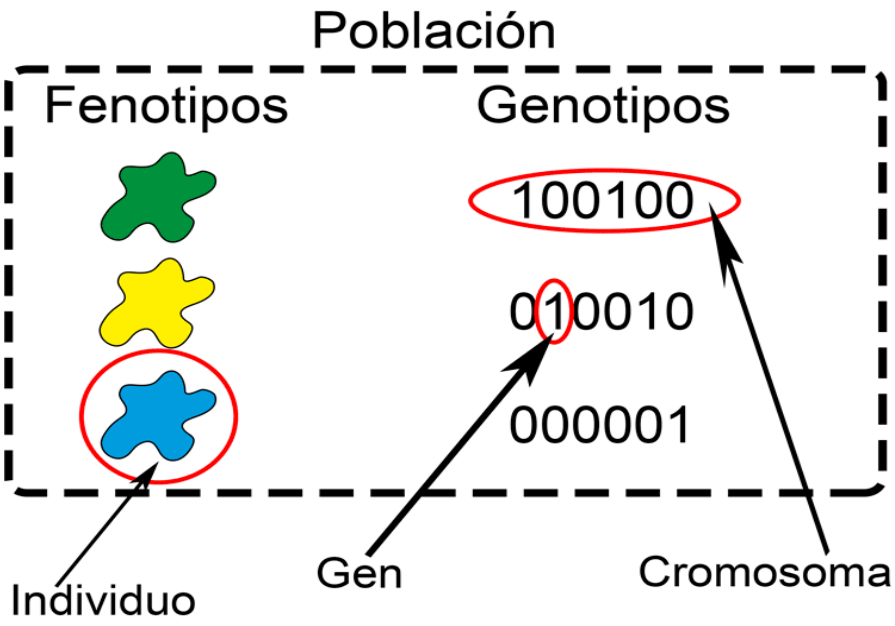
Ahondando en la consistencia de los Algoritmos Evolutivos, el siguiente diagrama ejem-



plifica el proceso de evolución entre soluciones.
Es menester incluir esta otra imagen para garantizar la comprensión en lo que a una solu-



ción se refiere:
Se añaden individuo, cromosoma, gen, alelo, imagen, poblacion y haciendo conexión directa con MoEA sSoftware, Mencionar que en este se encuentran individuo, Comunidad haciendo referencia a las clases blablabla. Además se tienen los siguientes conceptos: Como dato para posteriores referencias, un gen hace referencia a una casilla del cromosoma mientras que un alelo es el valor que puede existir en un gen.



Ahora se explican las etapas que conforman a este tipo de algoritmos:

2.1.1. Aptitud

Se entiende por aptitud **(también denominado *fitness* y a partir de ahora utlizando sólo esta denominación)** a un valor escalar que indica la calidad del individuo, esto es, a mayor fitness, mayor es la probalidad de que éste sea la solución óptima.
Indirectamente, esto nos indica que un individuo con un fitness alto tiene más probabilidades de ser elegido y propagar su carga genética; así el criterio para escoger a un individuo está basado comúnmente en su fitness.

Mencionar el uso del rank
Los tipos creados para este proyecto son:

2.1.1.1. Proporcional

El fitness está dado por la siguiente función:

$$fitness(individuo) = \frac{F_0(individuo)}{\sum_{i=1}^{tamaño\_población} F_0(individuo_i)}$$

Donde:

- F_0 es conocido como el valor de la función objetivo del individuo. Nótese que F_0 debe ser proporcional al fitness del individuo.

De acuerdo a la información provista anteriormente, la asignación es llamada así porque, como dice el nombre, el fitness de un individuo corresponde a la parte proporcional de la cantidad total de F_0 de la población.

2.1.1.2. Ranking Lineal

Es denominado así porque el fitness se asigna con una función lineal que tiene como fundamento la posición que ocupa el individuo dentro de la población.
El procedimiento es: los individuos se ordenan de acuerdo la evaluación en su función objetivo ¿? y entonces el fitness se basa en la posición que cada uno de los individuos ocupa. Más en específico, el fitness está proporcionado por la siguiente fórmula:

$$fitness(individuo) = 2 - SP + \frac{2 \cdot (SP-1) \cdot posición(individuo)}{tamaño\_población - 1}$$

Donde:

- **SP (Selective Pressure ó Presión Selectiva)** es un valor que oscila entre 1 y 2.
- **Posición(individuo)** es la que ocupa el individuo en la población.

Haciendo un análisis somero en la fórmula, se puede apreciar que los individuos con mejor fitness serán aquéllos que se encuentren en las últimas posiciones.

2.1.1.3. Ranking No Lineal

El fitness se constituye ordenando los elementos de la población con base en su función objetivo y después tomando la posición del individuo y una función polinomial **(la cual es una función no lineal, de ahí el nombre)**. La fórmula es la siguiente:

$$fitness(individuo) = \frac{TP \cdot X^{posición(individuo)}}{\sum_{i=1}^{TP} X^{i-1}}$$

Donde:

- **TP** es el tamaño de la población.
- **Posición(individuo)** es la que ocupa éste en la población.
- **X** es la solución al polinomio: $(SP-TP) \cdot X^{TP-1} + SP \cdot X^{TP-2} + ... + SP \cdot X + SP = 0$
- **SP (Selective Pressure ó Presión Selectiva)** varía entre 1 y 2.

Haciendo un análisis somero en la fórmula, al igual que con el fitness anterior se puede apreciar que los individuos con mejor fitness serán aquéllos que se encuentren en las últimas posiciones.

2.1.2. Selección

Durante dicha operación la importancia de la elección radica en el fitness de cada individuo, además un individuo puede ser seleccionado más de una vez si la causa lo amerita. De esta manera se elegirán tantos individuos como elementos haya en la población.

El objetivo radica en mantener el equilibrio entre una "selección justa" y la oportunidad de permitir a los individuos con una calidad media o baja la propagación de su carga genética.

Para este trabajo los tipos de selección desarrollados son:

2.1.2.1. Ruleta

También es llamado Proportional Selection **(ó Selección Proporcional)**.
En la función se distinguen dos etapas principales: construir la ruleta y "ponerla a girar" para que se elija al elemento.

Para la primera etapa se toma como base el Valor Esperado **(ó Expected Value)** de cada individuo.

El Valor Esperado para fines de este proyecto es el número de "hijos" que un individuo puede ofrecer. Éste se calcula de la siguiente forma:

$$Valor\_Esperado(individuo) = \frac{tamaño\_población \cdot fitness(individuo)}{\sum_{i=1}^{tamaño\_población} fitness(individuo_i)}$$

Al final aquéllos con Valores Esperados altos tendrán lugar a mayores espacios en la ruleta y por ende su probabilidad de elección aumenta.

Para recorrer la ruleta en realidad se toma un valor aleatorio entre 0 y la suma de los Valores Esperados. Entonces se van sumando los Valores Esperados de los individuos hasta que se exceda el valor aleatorio mencionado antes. Aquel elemento cuyo Valor Esperado haya excedido la suma se considera el elegido y es seleccionado para la etapa de cruza.

2.1.2.2. Torneo Probabilístico

Tal como lo sugiere el nombre, la selección será llevada a cabo en forma de competencia directa entre los individuos.
Tradicionalmente se comparan sus fitness y de esta manera el individuo ganador es aquél con la cantidad mayor de fitness, pero dado que se maneja un esquema probabilístico la decisión no depende totalmente del factor antes mencionado.

De esta manera se pueden recapitular los siguientes pasos:

- Tomar k $(2 \leqslant k \leqslant tamaño\_población)$ individuos de la población.

- Realizar el torneo de manera secuencial entre los elementos seleccionados anteriormente, esto es, tomar el elemento A y enfrentarlo con B, al resultado de la batalla anterior enfrentarlo con C y así sucesivamente.

Para ello por cada encuentro se crea un número aleatorio entre 0 y 1, si el número es menor a 0.5 se toma al elemento con menor fitness, de lo contrario se elige al de mayor fitness.
La operación se lleva a cabo hasta que se tenga un ganador de los k individuos.

Los dos pasos anteriores se repiten hasta que se hayan obtenido tantos individuos como el tamaño de la población.

2.1.2.3. Muestreo Estocástico Universal

Primero que nada es menester mencionar que es necesario el uso del Expected Value **(ó Valor Esperado)** de cada individuo.

Para fines concernientes a este proyecto, se trata del número de "hijos" que un individuo puede ofrecer. Éste se calcula de la siguiente forma:

$$Valor\_Esperado(individuo) = \frac{tamaño\_población \cdot fitness(individuo)}{\sum_{i=1}^{tamaño\_población} fitness(individuo_i)}$$

Con base a lo anterior, el método consiste en lo siguiente:

- Se selecciona un valor aleatorio entre 0 y 1, a éste se le llamará Pointer **(ó Puntero)**
- De manera secuencial se seleccionarán tantos individuos como el tamaño de la población, los cuales deben estar igualmente espaciados en su Valor Esperado tomando como referencia el valor de Pointer.

Es importante aclarar el segundo punto, así que se abordará desde una perspectiva computacional:

- Se deben tener variables adicionales que indiquen la acumulación tanto del Pointer **(CP, Cumulative Pointers)** como de los Valores Esperados **(CEV, Cumulative Expected Value)** así como al individuo actual que está siendo seleccionado **(I)**.
- Para averiguar si un individuo está igualmente espaciado en su Valor Esperado con respecto de los demás basándose en Pointer, basta con corroborar que:

$$CP + Pointer > CEV + EV$$

- Si la condición descrita es verdadera los valores EV e I deben actualizarse **(I se ajusta al siguiente individuo)** ya que esto indica que se buscará al siguiente individuo espaciado equitativamente con el valor Pointer. No se hace nada si la condición es falsa.

- Independientemente del valor de la condición anterior, CP y CEV deben actualizarse durante todo el ciclo.

Cabe mencionar que si la lista de individuos se agota, se puede volver a iterar desde el inicio teniendo cautela en conservar CEV y CP.

2.1.3. Cruza

Prosiguiendo con el ciclo de creación de una nueva población, es en este apartado donde se lleva a cabo la concepción de nuevos individuos.
Debido a esto se busca crear "hijos" más aptos que respondan mejor ante la problemática fundamentada, es decir, concebir soluciones que se adapten mejor a los criterios establecidos por el usuario desde un inicio basándose en las soluciones predecesoras.

Es menester mencionar que esta función es meramente binaria, lo cual significa que siempre deben haber dos padres y siempre debe regresar dos hijos. Las implementaciones desarrolladas son:

2.1.3.1. N Puntos

Su funcionamiento consiste en construir a los descendientes usando sub-bloques de cromosomas de cada uno de los padres, determinados éstos por una cierta cantidad de puntos de corte, de ahí el nombre.
Aterrizando lo anterior de una manera concisa se tiene lo siguiente:

- Consideremos a los cromosomas de los padres Padre I: $I_1 I_2 ... I_n$ y Padre J: $J_1 J_2 ... J_n$
- Posteriormente se determinan aleatoriamente los puntos de corte, cabe mencionar que si los cromosomas son de tamaño n, pueden existir máximo n - 1 puntos. Supongamos que se crean k puntos $(1 \leqslant k \leqslant n-1)$ y por lo tanto cada cromosoma queda separado en k + 1 bloques.
  De esta manera obtenemos:
  - Padre I en bloques **(BI)**: $BI_1 BI_2 ... BI_{k+1}$;
  - Padre J en bloques **(BJ)**: $BJ_1 BJ_2 ... BJ_{k+1}$.
- Finalmente cada hijo constará de la alternancia de bloques de manera secuencial comenzando por el bloque inicial de un padre determinado, dicho de otra forma, los hijos estarán constituidos de la siguiente manera:
  - Para el hijo H_1: $BI_1 BJ_2 ... BI_{k+1}$
  - Para el hijo H_2: $BJ_1 BI_2 ... BJ_{k+1}$

2.1.3.2. Uniforme

La característica de este procedimiento es crear nuevos individuos intercambiando secuencialmente los genes de sus padres; visto de una manera más estructurada consiste en lo siguiente:

- Tenemos a los cromosomas de los padres Padre A: $A_1A_2...A_n$ y Padre B: $B_1B_2...B_n$
- Ahora, cada hijo será construido con genes de uno y sólo uno de los padres a menos que se indique lo contrario; este movimiento será posible con una variable denominada Pmask **(Pm)** que toma valores de 0 a 1 y una probabilidad de Pmask **(Pp)** que también toma valores de 0 a 1. Entonces lo anterior se puede declarar así:
- Para el hijo (H_1) que tomará sus genes del padre A **(PA)**:
  - Si $Pm \leqslant Pp \; entonces \; H_1(i) = A_i, \; en \; otro \; caso \; H_1(i) = B_i; 1 \leqslant i \leqslant n$
- Para el hijo (H_2) que tomará sus genes del padre B **(PB)**:
  - Si $Pm \leqslant Pp \; entonces \; H_2(i) = B_i, \; en \; otro \; caso \; H_1(i) = A_i; 1 \leqslant i \leqslant n$

2.1.4. Mutación

Retomando el proceso de creación de una nueva población, es aquí donde una vez obtenidos los hijos, se modifican pequeñas porciones **(genes)** de sus cromosomas de manera individual.
Con ésto se persigue principalmente que estas ínfimas alteraciones permitan incrementar la exploración del material genético y por ende otorgar individuos aún más aptos sin caer en el peligro de perder características valiosas en la población.

Considerando lo anterior, lo primero que hay que tomar en cuenta es que la operación de Mutación es unaria, esto significa que sólo se puede mutar el cromosoma de un individuo a la vez.
Para este trabajo las implementaciones son las siguientes:

2.1.4.1. Binaria

El procedimiento es el siguiente:

- Se trata cada gen individualmente y se modifica de acuerdo a una probabilidad de Mutación asignada, si ésta es suficiente se procede a hacer el cambio, en otro caso se deja el alelo asociado al gen intacto.
- Retomando el caso en que se puede modificar el alelo del gen se verifica su valor actual y ya que se maneja una representación Binaria su transformación es muy simple: si se encuentra un 0, el alelo toma el valor 1 y viceversa.

2.1.4.2. Punto Flotante

El procedimiento es el siguiente:

- Se trata cada gen individualmente y se modifica de acuerdo a una probabilidad de Mutación asignada, si ésta es suficiente se procede a hacer el cambio, en otro caso se deja el alelo asociado al gen intacto.
- Retomando el caso en que se puede modificar el alelo del gen se verifica los límites de la variable de decisión que está ligada a éste, así como la precisión decimal. Entonces se crea el nuevo número con la precisión decimal requerida y se sustituye por el anterior.

Conviene mencionar además algunos conceptos extra que servirán en las secciones posteriores de este trabajo, tales como:
Elitismo
Especiación
Presión Selectiva

2.2. Optimización Multiobjetivo

En un lenguaje coloquial, la Optimización Multiobjetivo consiste en, dado un listado de objetivos, encontrar la solución que optimice el desempeño de cada uno de éstos bajo un cierto conjunto de restricciones.
Las condiciones de búsqueda son variadas, pero por lo general los objetivos tendrán conflictos entre sí, lo que quiere decir que el hallar una solución excelente para un objetivo puede significar la paupérrima para otro, por ello es que se debe ser cauteloso en la adquisición de soluciones.

Lo anterior aterrizado en un lenguaje matemático se desarrolla de la siguiente manera. Tenemos un vector de funciones objetivo:

$$F(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})]^T;\ n \geqslant 1.$$

Donde:

$$\vec{x} = [x_1, x_2, ..., x_k]^T;\ k \geqslant 1.$$

Representa al vector de variables de decisión que cada función objetivo recibe como parámetro. La meta es encontrar un vector especial en el espacio de funciones objetivo:

$$\vec{z^*} = \vec{F^*}(\vec{x^*}) = [f_1^*(\vec{x^*}), f_2^*(\vec{x^*}), ..., f_n^*(\vec{x^*})]^T;\ n \geqslant 1.$$

Basado en el vector de variables de decisión definido como:

$$\vec{x^*} = [x_1^*, x_2^*, ..., x_k^*]^T;\ k \geqslant 1.$$

Tal que:

$$f_i^*(\vec{x^*}) \leqslant f_i(\vec{x});\ 1 \leqslant i \leqslant n;\ \forall f \in F \quad .$$

Dicho de otra forma, se debe encontrar el vector de variables de decisión que minimize todas y cada una de las funciones objetivo en existencia. Adicionalmente, todo vector de variables de decisión debe estar sujeto a las restricciones:

$$h_i(\vec{x}) = 0;\ 1 \leqslant i \leqslant p\ \ (restricciones\ de\ igualdad).$$
$$g_i(\vec{x}) \leqslant 0;\ 1 \leqslant i \leqslant m\ \ (restricciones\ de\ desigualdad).$$

Las cuales para fines de este proyecto son aquéllas a las que se encuentran afianzadas las variables de decisión.

Algo importante a mencionar es que en las definiciones se trata únicamente la minimización de funciones objetivo porque, en caso de querer la maximización, simplemente se realiza la sustitución:

$$f_i'(\vec{x}) = -f_i(\vec{x});\ 1 \leqslant i \leqslant n,\ para\ alguna\ f \in F.$$

Es decir, minimizando la función negativa se obtiene el máximo. El producto de software elaborado ya contempla este tipo de casos.

También es preciso añadir la diferencia entre los términos Multiobjetivo y Multicriterio; mientras que el primero contempla a más de una función objetivo, el segundo hace lo propio con respecto de las variables de decisión, por ello es que en un problema de optimización pueden encontrarse ambos escenarios; si bien en este trabajo de tesis se le da prioridad a la faceta Multiobjetivo en pruebas ulteriores se tratarán con ejemplos considerados Multicriterio también.

2.2.1. Optimalidad de Pareto

Como se ha podido apreciar hasta este punto la Optimización Multiobjetivo prácticamente se sustenta en comparaciones y operaciones entre vectores; de lo anterior se deriva la necesidad de establecer un criterio para determinar la superioridad de un vector frente a otro y así otorgar la solución idónea.
Si bien existen considerables maneras de lograr este objetivo, para fines del proyecto presentado se toma en cuenta aquélla conocida como la **Eficiencia de Pareto** u **Optimalidad de Pareto**.

Haciendo alusión al contexto histórico este concepto fue creado por el economista italiano Vilfredo Pareto, quien basado en los trabajos de Edgeworth [2] concibió en 1896 una propuesta de equilibrio en el cual se beneficia a un elemento sin perjudicar a otro

y viceversa, así cuando esta situación sea insostenible entonces el equilibrio deja de tener sentido.

Concretando el enunciado previo con base en definiciones la primera que se utiliza es la de **Dominancia de Pareto** o simplemente *dominancia* entre vectores, para ello se toman los vectores $U = (u_1, u_2, ..., u_k)$ y $V = (v_1, v_2, ..., v_k)$ y se dice que **U domina a V ó V es dominada por U** si:

$$\forall i \in \{1, ..., k\} \;\; u_i \leqslant v_i \wedge \exists i \in \{1, ..., k\}; \;\; u_i < v_i.$$

Esto significa que U debe ser menor que V en cada uno de sus componentes para garantizar la dominancia.
La simbología que se suele usar para identificar este hecho es $u \succ v$.
Conviene mencionar la existencia de dos tipos de dominancias: fuerte y débil. La primera es prácticamente la definición construida para la comparación entre vectores, mientras que la seguna únicamente prescinde de la condición de existencia menor estricta.
Esto sólo se menciona con fines ilustrativos ya que a pesar de esta distinción únicamente se trabajará con la dominancia fuerte, si bien computacionalmente toma más tiempo calcularla, es más precisa y por ende se arrojan resultados más concisos que usando la contraparte débil.

Tomando el enunciado previo sobre dominancia y aquéllos creados en la sección de Optimización Multiobjetivo, se introducen los conceptos de **Óptimo de Pareto** y **Frente de Pareto**.
El primero, también denominado **Conjunto Óptimo de Pareto** ($P^* o \; P_{true}$) es aquél que cumple con la siguiente condición:

$$P^* := \{\vec{x} \mid \vec{F}(\vec{x}) \succ \vec{F}(\vec{x}'); \; \forall \vec{x}' \in FS\}$$

Nótese que de hecho el Conjunto Óptimo de Pareto es equivalente a la obtención de todos los vectores $\vec{x^*}$ de la sección de Optimización Multiobjetivo, además FS es conocido como la *región factible* o lo que es lo mismo el espacio plausible de todas las variables de decisión; como consecuencia se tiene que $P^* \subseteq FS \subseteq I\!R^n$.

Ahora el Frente de Pareto es el conjunto descrito a continuación:

$$PF^* := \{\vec{z^*} = \vec{F^*}(\vec{x^*}) = [f_1^*(\vec{x^*}), f_2^*(\vec{x^*}), ..., f_n^*(\vec{x^*})] \mid x^* \in P^*\}$$

El cual no es otra cosa que la recopilación de las evaluaciones del Óptimo de Pareto en las funciones objetivo.
Dicho de otra forma, el Óptimo de Pareto corresponde al conjunto de las soluciones en el espacio de variables de decisión, mientras que el Frente de Pareto es lo análogo con respecto del espacio de funciones objetivo.

Generalizando, al final un problema de Optimización Multiobjetivo se reduce en encontrar el Frente de Pareto, no obstante de acuerdo con el Dr. Coello [3] no existe una

forma analítica de obtener el Óptimo de Pareto **(y por ende el Frente de Pareto)**, lo que significa que se deben utilizar aproximaciones para encontrar el resultado deseado. Es aquí donde entra el uso de los Algoritmos Evolutivos descritos en la sección anterior, destacando además que ésta es sólo una de las numerosas aproximaciones elaboradas [4] para encontrar el ya mencionado Frente de Pareto.
Debido al alcance de este proyecto, dichas técnicas alternativas no se mencionan.

Capítulo 3

M.O.E.A.

Con base en el capítulo anterior, el funcionamiento de un M.O.E.A. **(resolver un problema de optimización multiobjetivo usando algoritmos evolutivos)** generalmente se lleva a cabo de la siguiente manera:

1. Usando una Representación Cromosómica, crear la Población Padre y evaluar cada uno de los Individuos respecto a las funciones objetivo.

2. Asignar un Ranking a los Individuos de la Población Padre.

3. Con base en el Ranking, asignar la Aptitud **(en inglés Fitness)** a cada uno de los Individuos.

4. Tomando en cuenta el Fitness, aplicar las operaciones de Selección, Cruza y Mutación con la finalidad de crear una Población Hija. Todos los métodos empleados en este punto deben funcionar acorde a la Representación Cromosómica del punto 1.

5. **(Opcional)** Utilizar el Fitness Compartido **(en inglés Shared Fitness)** para aplicar una elección más minuciosa de los mejores Individuos en la Población Hija.

6. Designar a la población Hija como la nueva población Padre.

7. Repetir los pasos 2 a 6 hasta haber alcanzado un número límite de generaciones **(iteraciones)**.

A grandes rasgos la diferencia entre un M.O.E.A. y otro es la Presión Selectiva **(en inglés Selective Pressure)** que se aplica durante el procedimiento, para fines de este proyecto se trata de la tolerancia para seleccionar a los Individuos de calidad media o baja frente a los mejores. Una baja Presión Selectiva permite elegir Individuos no tan aptos; el caso es análogo para una alta Presión Selectiva.

3.1. Fitness Compartido

3.2. Algoritmos

3.2.1. V.E.G.A.

La forma de proceder del algoritmo es la siguiente:

1. Se crea la Población Padre (de tamaño n).
2. Tomando en cuenta las k funciones objetivo y la Población Padre, se crean k subpoblaciones de tamaño n/k cada una, si este número llega a ser irracional se pueden hacer ajustes con respecto de la distribución de los Individuos.
3. Por cada subpoblación, se aplica la técnica de Selección y obtienen los n/k Individuos, terminado esto se deben unificar todos los seleccionados de nuevo en una súper Población.
4. Con la súper Población del paso 3, se crea a la población Hija, la cual pasará a convertirse en la la nueva Población Padre.
5. Se repiten los pasos 2 a 4 hasta haber alcanzado el número de generaciones **(iteraciones)** límite.

Como se puede apreciar es una implementación muy sencilla de optimización multiobjetivo, sin embargo el inconveniente que tiene es la fácil pérdida de material genético valioso.
Lo anterior significa que un Individuo que en una generación previa era el mejor para una función objetivo i al momento de ser separado y seleccionado en una subpoblación j **(y por ende analizado bajo la función objetivo** j**)** puede ser muy malo en calidad y por tanto no ser seleccionado; perdiéndose la ganancia genética hasta el momento obtenida para la función objetivo i; $i \neq j$.

Por ello es que se puede decir que V.E.G.A. genera soluciones promedio que destacan con una calidad media para todas las funciones objetivo.

Finalmente hay que comentar que para este algoritmo no se requiere aplicar un Ranking específico, no obstante, se ha decidido utilizar el de Fonseca & Flemming **(véase Model/Community/Community.py)** pues es el más sencillo de implementar.

3.2.2. M.O.G.A.

Su funcionamiento es el siguiente:

1. Se crea la Población Padre, se evalúan las funciones objetivo de sus correspondientes Individuos.

2. Se asigna a los Individuos un Ranking **(Fonseca & Flemming)** y posteriormente se calcula el Niche Count de la Población Padre.

3. Tomando en cuenta los valores del punto 2 se obtiene el Fitness para cada Individuo y posteriormente su Shared Fitness.

4. Se aplica el operador de selección sobre la Población Padre para determinar los elegidos para dejar descendencia.

5. Se crea la Población Hija, se evalúan las funciones objetivo de sus correspondientes Individuos.

6. Se asigna a los Individuos un Ranking **(Fonseca & Flemming)** y posteriormente se calcula el Niche Count de la Población Hija.

7. Tomando en cuenta los valores del punto 6 se obtiene el Fitness para cada Individuo y posteriormente su Shared Fitness.

8. La Población Hija pasará a ser la nueva Población Padre.

9. Se repiten los pasos 4 a 8 hasta que se haya alcanzado el número límite de generaciones **(iteraciones)**.

Como se puede apreciar, la implementación de este algoritmo es muy sencilla, además se rige casi en su totalidad por el Shared Fitness **(ó Fitness Compartido)**, por lo que la Presión Selectiva **(ó Selective Pressure)** incluida dependerá en gran medida de la función de Distancia que se utilice, así como de la magnitud indicada por el usuario.

Finalmente es menester mencionar que para esta implementación el Ranking utilizado debe ser estrictamente el de Fonseca & Flemming

3.2.3. S.P.E.A. 2

Se desarrolla la implementación de la técnica M.O.E.A. conocida como S.P.E.A. II **(Strength Pareto Evolutionary Algorithm ó Algoritmo Evolutivo de Fuerza de Pareto)**.
El funcionamiento del algoritmo es el siguiente:

1. Se inicializa una población llamada P y un conjunto inicialmente vacío llamado E **(E albergará Individuos también)**; ambos son de tamaño n.

2. Se asigna el Fitness a los Individuos de P y E **(para ello se evalúan las funciones objetivo de los Individuos de ambos conjuntos y se asigna el Ranking Zitzler & Thiele)**.

3. A continuación se funden *P* y *E* en una súper Población **(llamémosle S también señalado en el algoritmo como Mating Pool, de tamaño n)**.Para ello primero se añaden los Individuos *NO DOMINADOS* de *P* en *S* y posteriormente los *NO DOMINADOS* de *E* en *S*.
   Aquí se distinguen dos casos:

   - Si llegasen a faltar Individuos se añaden al azar Individuos *DOMINADOS* de *P* en *S* hasta completar la demanda.
   - Si después de la fusión el número de Individuos supera a n, entonces se hace un truncamiento en *S* hasta ajustar su tamaño a n.

4. *S* será la nueva *E*, además se crea la población Hija de la recién creada *E* **(E-Child)**.

5. E-Child será la nueva P.

6. Se repiten los pasos 2 a 5 hasta que se haya alcanzado el límite de generaciones **(iteraciones)**.

Finalmente lo que se regresa es *E*, ya que ahí es donde se han almacenado los mejores Individuos de todas las generaciones.

La característica de este algoritmo es que tiene una Presión Selectiva alta ya que se da prioridad a los Individuos no dominados **(de ahí el nombre de Fuerza de Pareto ó los más fuertes con respecto al principio de Pareto)**, y el hecho de mezclar a *E* y *P* en una súper Población garantiza la conservación de los mejores Individuos sin importar el transcurso de las generaciones **(a eso se le conoce como Elitismo)**, pero también da una tolerancia, aunque mínima, a los Individuos de menor calidad como en el punto 3. Además al momento de actualizar *S* a *E* y E-Child a *P* se tiene una especie de seguro de vida, es decir, si en algún momento la población E-Child llegara a tener una calidad baja se tiene el respaldo de *E* para una generación posterior para formar *S*.

Se debe tener en cuenta que el algoritmo originalmente no contempla ni una súper Población *S* ni E-Child sino que en los pasos 3 y 4 se utiliza solamente *E* para referirse tanto a E-child como a *S*, sin embargo para no confundir al usuario en la funcionalidad del método se decidió colocar contenedores extra para poder diferenciar más precisamente a los elementos involucrados.

Algo muy importante a mencionar es que en el paso 1 y al momento de crear la población E-Child es necesario evaluar las funciones objetivo, asignar un Ranking y posteriormente un Fitness para que se puedan aplicar los operadores geneticos **(véase Model/GeneticOperator)**, para este caso el Ranking es estrictamente el de Zitzler & Thiele;

3.2.4. N.S.G.A. II

La forma de proceder del método es la siguiente:

1. Se crea una Población Padre **(de tamaño n)**, a la cual se le evalúan las funciones objetivo de sus Individuos, se les asigna un Ranking **(Goldberg)** y posteriormente se les otorga un Fitness.

2. Con base en la Población Padre se aplica el operador de Selección para elegir a los Individuos que serán aptos para reproducirse.

3. Usando a los elementos del punto 2, se crea una Población Hija **(de tamaño n)**.

4. Se crea una súper Población **(llamémosle S, de tamaño 2n)** que albergará todos los Individuos tanto de la Población Padre como Hija; a *S* se le evalúan las funciones objetivo de sus Individuos, se les asigna un Ranking **(Goldberg)** y posteriormente se les otorga un Fitness.

5. La súper Población *S* se divide en subcategorías de acuerdo a los niveles de dominancia que existan, es decir, existirá la categoría de dominancia 0, la cual almacena Individuos que tengan una dominancia de 0 Individuos **(ningún Individuo los domina)**, existirá la categoría de dominancia 1 con el significado análogo y así sucesivamente hasta haber cubierto todos los niveles de dominancia existentes.

6. Se construye la nueva Población Padre, pare ello constará de los Individuos de *S* donde la prioridad será el nivel de dominancia, es decir, primero se añaden los elementos del nivel 0,luego los del nivel 1 y así en lo sucesivo hasta haber adquirido n elementos. Se debe aclarar que la adquisición de Individuos por nivel debe ser total, esto significa que no se pueden dejar Individuos sueltos para el mismo nivel de dominancia.
   Supongamos que a un nivel k existen tantos Individuos que su presunta adquisición supera el tamaño n, en este caso se debe hacer lo siguiente:

   a) Se crea una Población provisional **(Prov)** con los Individuos del nivel k, se evalúan las funciones objetivo a cada uno de sus Individuos, se les asigna un Ranking **(Goldberg)** y posteriormente se les asigna el Fitness.
   Con los valores anteriores se calcula el Niche Count **(véase Model/SharingFunction)** de los Individuos; una vez hecho ésto se seleccionan desde Prov los Individuos faltantes con los mayores Niche Count, esto hasta completar el tamaño n de la nueva Población Padre.

7. Al haber conformado la nueva Población Padre, se evalúan las funciones objetivo de sus Individuos, se les asigna el Ranking correspondiente **(Goldberg)** y se les atribuye su Fitness.

8. Se repiten los pasos 2 a 7 hasta haber alcanzado el límite de generaciones **(iteraciones)**.

Como su nombre lo indica, la característica de este algoritmo es la clasificación de los Individuos en niveles para su posterior selección.
Esto al principio propicia una Presión Selectiva moderada por la ausencia de elementos con dominancia baja que suele existir en las primeras generaciones, sin embargo en iteraciones posteriores se agudiza la Presión Selectiva ya que eventualmente la mayoría de los Individuos serán alojados en las primeras categorías de dominancia, cubriendo casi instantáneamente la demanda de Individuos necesaria en el paso 6, por lo que las categorías posteriores serán cada vez menos necesarias con el paso de los ciclos.

Por otra parte la fusión de las Poblaciones en *S* garantiza que siempre se conserven a los mejores Individuos independientemente de la generación transcurrida, a eso se le llama Elitismo.
Por cierto que en el algoritmo original no existe un nombre oficial para *S* sino más bien se señala como una estructura genérica, sin embargo se le ha formalizado con un identificador para guiar apropiadamente al usuario en el flujo del algoritmo.

Para finalizar se señala que el uso del ranking de Goldberg

3.3. M.O.P.

3.4. Indicadores de Desempeño

3.5. Introducción a M.O.E.A Software

Habiendo detallado todos los elementos
M.O.E.A. Software surge como una solución ante la problemática blablabla.
Para abordar detalles de índole más técnica se recomienda visitar la sección **Características Técnicas** perteneciente al Apéndice.

Capítulo 4

Experimentación y Análisis de Resultados

Una vez oscultada la teoría de los la herramienta con la que se llevan a cabo las pruebas, se procede a

Capítulo 5

Conclusiones

Con base en los resultados anteriores se puede primeramente verificar que

5.1. Trabajo Futuro

Una vez que se han concretado los méritos y metas cumplidas del presente trabajo escrito conviene mencionar los posibles escenarios de expansión del proyecto, ante lo cual se pueden dividir en dos categorías: analítica y técnica.

En lo concerniente a la primera
Ahora en consideración a la segunda el tratamiento se enfoca principalmente en las características relacionadas con M.O.E.A. Software.
En primer lugar, debido a la carencia de tiempo y limitaciones tecnológicas, las restricciones que están sujetas a las variables de decisión hasta el cierre de este trabajo corresponden únicamente a valores escalares aún teniendo en cuenta que las definiciones vistas en un principio contemplan el uso de funciones. Por este motivo se debe tener en mente la inclusión de esta característica para futuras versiones del programa.
Se manifiesta un escenario parecido con las funciones objetivo ya que el producto de software no considera aquéllas definidas a trozos que, aunque no forman parte de la versión actual, son imprescindibles puesto que muchos de los M.O.P.'s adicionales localizados en las fuentes de consulta****** contienen funciones de este tipo.

Desde una perspectiva gráfica, la interfaz elaborada hasta el momento resulta insuficiente debido a que durante el proceso se hallaron varias fallas originadas en el uso de bibliotecas que carecían de mantenimiento, no obstante se utilizaron debido a su portabilidad con todos los sistemas operativos.
Es por esta razón que se puede sugerir el uso de alguna otra tecnología gráfica pudiendo incluso ser considerado algún microservicio en la red **(con Node.js por ejemplo)** o aplicación móvil **(usando Android ó Swift)** con la finalidad de hacer mas eficiente la

interacción con el usuario y por otro lado garantizar una mayor distribución en la divulgación de las técnicas mostradas durante el desarrollo de este trabajo.

Bibliografía

[1] Charles Darwin. *El Origen de las Especies*. (Spanish) [*On the Origin of Species*]. Translator: Antonio de Zulueta, Biblioteca Virtual Miguel de Cervantes, pages 460–461, 1859.

[2] Francis Ysidro Edgeworth. *Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences*. Kegan Paul and Co., London, 1881.

[3] Carlos A. Coello Coello, Gary B. Lamont, David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, Second edition, September 2007, ISBN 978-0-387-36797-2.

[4] Juan Arturo Herrera, Katya Rodríguez Vázquez. *Técnicas de Programación Matemática para Optimización Multi-Criterio: Estado del Arte*. Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Reporte Técnico EMO07-01.

[5] Kalyanmoy Deb. *Software for Multi-Objective NSGA-II Code in C*. `http://www.iitk.ac.in/kangal/codes.shtml`

[6] Gartner Inc. *Windows Comes Up Third in OS Clash Two Years Early*. `http://www.computerworld.com/article/3050931/microsoft-windows/windows-comes-up-third-in-os-clash-two-years-early.html`

Apéndice

Esta sección corresponde a la incursión de detalles elementales relacionados con el producto de software denominado **M.O.E.A. Software**, la cual recopila las características de dicho producto tales como el diseño y datos alusivos a la construcción y ejecución del programa.
Con base en lo anterior, es preciso mencionar que en este apartado existe terminología meramente técnica que se contempla para garantizar una mayor comprensión del proyecto.

Habiendo proporcionado el anterior preámbulo, la finalidad es, a grandes rasgos, otorgar una guía rápida y concisa al usuario en lo concerniente a la motivación e implementación del producto de software, la idea detrás de ésto radica en notificar al usuario de todos los pormenores encontrados en el desarrollo del programa y de esta manera lograr que su interacción con el producto sea afable, evitando para ello la mayor cantidad de contratiempos posible.

Es importante mencionar que en caso de querer ahondar más minuciosamente en los componentes del programa se ofrece un **Manual Técnico** que contiene precisadas todas las funcionalidades con su respectiva explicación, las cuales han sido creadas para poder erigir el producto de software .
Dicho manual se ha constituido en un documento independiente al trabajo de tesis que, aunque guarda una cierta distancia con los temas que se revisan aquí al estar asociado a un panorama más técnico, también contiene referencias de carácter analítico para poder enlazar adecuadamente lo estipulado en este medio con aquél.

A continuación se abarca más detalladamente cada uno de los rasgos del producto de software.

1.1. Características de Diseño

En esta parte se considera todo lo relacionado con la arquitectura de diseño.
Este concepto corresponde al tipo de organización que se suele emplear para agrupar y comunicar apropiadamente cada uno de los componentes del producto de software con la finalidad de minimizar los tiempos de corrección y actualización del código, así como proporcionar una presentación digerible para cualquiera que desee familiarizarse con las

partes codificadas.

Para este proyecto se ha elegido la arquitectura denominada MVC **(Model-View-Controller ó Modelo-Vista-Controlador)**.
Siguiendo este tipo de organización, se colocan las funcionalidades en tres categorías principales, que son:

- **Model (ó Modelo)**, se almacenan todos los elementos que realizan el proceso analítico, en este caso todo lo relacionado con la ejecución de M.O.E.A.'s y la recolección de resultados.
- **View (ó Vista)**, se coloca todo aquéllo asociado a la interfaz gráfica del programa y en el caso del proyecto, la graficación apropiada de resultados.
- **Controller (ó Controlador)**, se guarda toda la parafernalia relativa al control de las comunicaciones entre la Vista y el Modelo.

El proceso usual de interacción entre dichas categorías es el siguiente:

1. El usuario inserta las configuraciones pertinentes en la Vista, las cuales permitirán obtener resultados detallados del M.O.E.A. que se fuera a ejecutar.
2. El Controlador obtiene las configuraciones previamente insertadas por el usuario; durante esta etapa se realiza una verificación y saneamiento de dichas configuraciones. Si el proceso fue exitoso se procede ir al paso **(3)**, en cualquier otro caso se retrocede al paso **(1)** con una notificación de error.
3. El Modelo se encarga de ejecutar el algoritmo precisado por el usuario en **(1)**, para ello se le proporcionan todas las configuraciones adjuntas. Una vez concluido el proceso el Modelo le regresa los resultados al Controlador.
4. El Controlador toma los resultados y a su vez los transfiere a la Vista, la cual se encarga de mostrar al usuario los datos finales de manera gráfica.

Señalando nuevamente al Manual Técnico, en éste el usuario notará que las funcionalidades están colocadas siguiendo un listado basado en la arquitectura antes mencionada. Esto permite vislumbrar de manera secundaria los elementos y sus relaciones.

1.2. Características Técnicas

Adentrándose en una perspectiva tecnológica el programa **M.O.E.A. Software** está elaborado usando el lenguaje de programación Python en su versión 2.7.3.
(http://www.python.org)

Conviene primeramente explicar la motivación detrás del uso de dicho lenguaje; desde

el punto de vista del autor éste es sintácticamente fácil de aprender lo cual implica un esfuerzo reducido en la lectura y comprensión del código fuente por lo que de esta manera el usuario puede familiarizarse rápidamente con las funcionalidades elaboradas.
Por otra parte, al ser un lenguaje de alto nivel, éste permite la agrupación de varias instrucciones en pocos comandos **(a esto se le conoce también como azúcar sintáctica)**, como resultado se crea un código no tan extenso que de nueva cuenta agiliza su interpretación por parte del usuario.

Llegados a este punto se pudiera uno cuestionar la existencia de otros productos de software cuyo fin se pareciera o fuera el mismo que el que se muestra en el presente trabajo de tesis.
Si bien existen paquetes que realizan operaciones relacionadas tanto con la Optimización Multiobjetivo como con los Algoritmos Evolutivos, como los siguientes:

- **DEAP, Distributed Evolutionary Algorithms in Python (https://pypi.python.org/pypi/deap)**.
- **PyGMO, Python Parallel Global Multiobjective Optimizer (http://esa.github.io/pygmo/)**.
- **jMetal, Metaheuristic Algorithms in Java (http://jmetal.sourceforge.net/)**.
- **MOEA Framework (http://moeaframework.org/)**.
- **Borg MOEA (http://borgmoea.org/)**.

Se pueden percibir ciertas diferencias, en primera instancia se señala el lenguaje de programación utilizado, pues jMetal y MOEA Framework se han elaborado usando el lenguaje de programación Java; desde otra perspectiva el paquete PyGMO aunque ofrece técnicas de Optimización Multiobjetivo éstas no se centran en el uso de Algoritmos Evolutivos precisamente sino que ofrece una gama aparte de métodos relacionados con el primer tema.
Enfocándose en la disponibilidad Borg MOEA resulta ser un programa muy completo, sin embargo su uso se restringe sólo a ciertas operaciones de tipo comercial o estudiantil por lo que en ese sentido su extensión se encuentra limitada.
Finalmente el paquete que más se acerca, DEAP, aunque maneja un catálogo mayor de Algoritmos Evolutivos no considera la graficación de resultados además de que es preciso aprender la sintaxis misma del paquete para poder operar con las funcionalidades y ello implica que el uso está dirigido hacia un público más especializado en el tema.

Dicho lo anterior, sin desmerecer el esfuerzo y dedicación que se ha puesto en cada uno de los trabajos antes mencionados, con base en estas comparaciones se podrá aseverar que el desarrollo de **M.O.E.A. Software** tiene finalidades distintas, ya que, como se ha

mencionado con anterioridad, su uso se enfoca hacia el acercamiento inicial de los Algoritmos Evolutivos Multiobjetivo desde un panorama gráfico donde para ello el usuario tiene total libertad de revisar y/o modificar el código fuente anexado sin necesidad de ninguna capa intermedia de aprendizaje debido a que todo el proyecto ha sido escrito usando la sintaxis más simple.

En lo concerniente al contenido del producto de software, lo que se incluye en el proyecto son tanto los ejecutables como el código fuente.
La principal diferencia entre éstos radica en que los primeros son programas que contienen todas las dependencias necesarias ya anexadas para que el usuario únicamente se enfoque en la ejecución e interacción con el producto de software, por otra parte el código fuente almacena las funcionalidades desarrolladas de **M.O.E.A. Software** pero para su ejecución es preciso que el usuario instale algunas dependencias en su sistema operativo.

Para los ejecutables se ha determinado enfocarse en sistemas operativos de escritorio Windows y GNU/Linux por su popularidad y en algunos casos gratuidad [**?**], sin embargo, debido a una falla de origen con una de las dependencias usadas en el producto de software **(Matplotlib)** se ha tenido la necesidad de crear un ejecutable por cada sistema operativo. De esta manera los que se encuentran soportados son:

- **Windows**.
- **Debian**.
- **Ubuntu**.
- **CentOS**.
- **Fedora**.

No importa la versión utilizada siempre y cuando sea del mismo linaje de sistema operativo empleado.

Algo muy importante a mencionar es que los ejecutables han sido creados para sistemas operativos de 32 bits, esto ya que en teoría los programas hechos para sistemas de 32 bits son admisibles en sistemas de 64 bits, no así el caso contrario.
A pesar de esto, mientras este factor en los sistemas operativos Windows no presenta problema, en los relativos a GNU/Linux de 64 bits sí puesto que se ha detectado que éstos no contienen los paquetes necesarios para poder ejecutar programas de 32 bits.
Dado que este proyecto no contempla este tipo de paquetes y hasta el término de este proyecto no se han podido integrar a los ejecutables es imprescindible alertar al usuario sobre estas condiciones.
Lo ideal sería el uso de un sistema operativo de 32 bits pero en caso de no ser posible se deben instalar los paquetes que permitan la ejecución de programas de 32 bits.

Es menester mencionar el programa utilizado para la elaboración de los ejecutables, para

este proyecto se ha utilizado la herramienta **PyInstaller (versión 3.2)** **(http://www.pyinstaller.org)**.
A grandes rasgos lo que realiza este ejecutable es introducir el código fuente, paqueterías y demás aditamentos necesarios dentro de un contenedor que eventualmente llega a ser el ejecutable.
Dicho de otra forma, PyInstaller crea un ínfimo ambiente controlado en el que se puede ejecutar **M.O.E.A. Software** dando la ilusión de que se ha creado un programa ejecutable independiente.

En el peor de los escenarios, si no se tiene la posibilidad de operar con los archivos ejecutables ya sea por el problema antes mencionado o porque los sistemas soportados no coinciden con el sistema del usuario se ofrece el código fuente, al estar elaborado en Python y éste a su vez ser un lenguaje multiplataforma **(se puede instalar en cualquier sistema operativo)** como consecuencia lógica se indaga que el código fuente puede ser ejecutado en cualquier sistema si se cuenta con las dependencias adecuadas.

Como se ha mencionado anteriormente el producto de software ha sido escrito usando la sintaxis más simple y ello implica que se ha construido utilizando sólo las dependencias necesarias para garantizar que aún empleando el código fuente se conciba la máxima independencia posible del producto de software.

Para utilizar el código fuente es necesario, además de la instalación obligatoria de Python 2.7.3 **(otras versiones causarían problemas de incompatibilidad)** las siguientes dependencias:

- **Tkinter**, versión 8.5 **(http://tkinter.unpythonic.net/wiki/How_to_install_Tkinter)**.
- **Matplotlib**, versión 1.1.1rc2 **(http://matplotlib.org/)**.

La preferencia por las versiones de las dependencias no es estricta, no obstante se recomienda seguir estas indicaciones lo más fielmente posible para garantizar resultados satisfactorios.
De manera similar para con los archivos ejecutables, dado que los sistemas operativos en los que se use el programa pueden variar sólo se indican las bibliotecas empleadas, le corresponde al usuario instalarlas en su sistema operativo.

Por otra parte el desarrollo del programa se llevó a cabo primordialmente usando entornos virtuales con la ayuda de la herramienta **VirtualBox** **(https://www.virtualbox.org/)**
Entonces, los sistemas operativos creados fueron:

- **Windows 7 *Home Premium* (32 bits)**; 1GB Memoria RAM, Procesador Intel Core i5.
- **Debian 7 *Wheezy* (32 bits)**; 1GB Memoria RAM, Procesador Intel Core i5.

Tanto la construcción del código como las pruebas consecuentes fueron hechas en estos sistemas debido a que se necesitaba asegurarse que el producto de software funcionara en los entornos principales Windows y GNU/Linux.

Se considera importante mencionar las características de los sistemas operativos creados con la finalidad de enfatizar la elaboración del producto de software bajo escenarios con condiciones paupérrimas; de esta manera se garantiza un rendimiento favorable en sistemas con los mismos o mayores recursos.

Para construir los ejecutables de los demás sistemas operativos se crearon por igual sistemas virtualizados, sin embargo dado que sólo fueron utilizados para los fines señalados no se considera importante profundizar más en sus características.

Finalmente es menester mencionar que las instrucciones y aditamentos ilustrativos en la interfaz gráfica del programa se encuentran plasmados en el idioma inglés, el objetivo de tal acción tiene dos metas: pulir las habilidades lingüísticas del autor e incentivar a los estudiantes con la práctica de dicho idioma, de todas formas la gramática y vocabulario empleados son sencillos y no se encontrará dificultad alguna en la comprensión del tema.