UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

ALGORITMOS EVOLUTIVOS MULTIOBJETIVO Y EVALUADORES DE DESEMPEÑO A TRAVÉS 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

2020

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 las pautas de desempeño establecidas en el Capítulo 3.

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 contiene 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

M.O.E.A.

Una vez que se han descrito los dos preceptos más importantes de M.O.E.A., entonces podemos definir a un Multi-Objective Evolutionary Algorithm **(Algoritmo Multiobjetivo Evolutivo)** como un problema de optimización multiobjetivo donde para su resolución se emplean algoritmos evolutivos.

Como se ha visto con anterioridad, la gama de técnicas y en general de opciones ya sea de un antecedente o del otro es muy amplia y por ende existe también en ésta una variedad considerable de métodos [4, Coello 145-146], no obstante y para fines de este trabajo se toma una muestra representativa, destacando:

- 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)**.

De esta manera, se indican en este capítulo las características de cada una y se llevan a cabo comparaciones con métricas que se tratan más adelante.

Se tiene que comentar además que el algoritmo evolutivo debe cambiar su estructura ya que como se mencionó en el capítulo anterior éste lidia con la minimizacion de un solo objetivo mientras que en este caso se tratan con múltiples funciones, lo cual indica que se deben introducir otras comparativas para obtener a los mejores individuos.

Así, se proporciona un algoritmo genérico para introducir al M.O.E.A. [4, Coello 147-148]:

1. Fase de inicialización donde se generan N individuos en una población $P^0(i), i = 0$ y se evalúa el fitness. La codificación de cada individuo puede ser binaria, entera o real.

2. Se eliminan los individuos dominados con base en las comparaciones basadas en Pareto en $P^0(i), i = 0$ y tomando en cuenta las evaluaciones multiobjetivo. $P^0(i) \rightarrow P^1(i)$.

3. Se descartan los individuos inviables de $P^1(i)$ o se "reparan" basándose en otras condiciones. $P^1(i) \rightarrow P^2(i)$.

4. Se usa un algoritmo de clusterización para limitar el número de individuos en $P^2(i)$ en pequeñas regiones del PF_{known} ó P_{known} de la población actual y las cuales incluyen al *niching*, *aptitud compartida* y *crowding* con valores paramétricos asociados.

5. Se llevan a cabo operaciones de M.O.E.A. (cruza, mutación, etc.) para crear una nueva población usando los apropiados valores paramétricos; $P^2(i) \rightarrow P^3(i)$. Para elegir a estos individuos se pueden emplear técnicas de selección tales como elección aleatoria, torneo binario, selección, elitismo, etc.

6. Se toman los individuos que conformen la siguiente generación (población $P^4(i)$) mediante operaciones en $P^3(i)$ o $P^2(i) \cup P^3(i)$. Para esto se pueden usar técnicas de selección binaria, elección por torneo (con ó sin reemplazo), rankeo o elitismo para que se limite el tamaño de $P^4(i)$.

7. Si la condición de término no se cumple (ya sea un número límite de generaciones o algún criterio de convergencia) se asigna $P^4(i) a P^0(i), i = i+1$ y se regresa al paso 2.

8. Se eliminan los individuos dominados bajo la comparativa de Pareto y en general los individuos inviables de $P^4(i)$ o se "reparan". Se asigna $P^4(i) a P^0(i)$.

Es importante comentar que en los pasos 3 a 5 es posible auxiliarse de algún entorno de almacenamiento secundario para almacenar los individuos no dominados de P^3 y que sirvan como "reparación" o reemplazo para conformar las poblaciones nuevas.
A continuación se describen los nuevos términos añadidos:

2.1. Rankeo

El rankeo (ranking) en estructo sentido funge como una métrica que permite asignar un valor escalar a un individuo tomando como base resultados de multiples objetivos.

Es este concepto el que se introduce en los M.O.E.A.'s porque tradicionalmente en un algoritmo evolutivo se persigue la optimización de un sólo objetivo, mientras que en este caso al generalizarse a múltiples funciones, surge un problema de represantividad, es decir, se requiere de un elemento que proporcione la eficiencia o falta de ésta de una solucion bajo multiples objetivos. En otras palabras el ranking lleva acabo una reduccion de dimensionalidad

poner la formula.
Como suele darse el caso en ese tipo de tópicos, se proporciona un conjunto clave de ejemplificaciones de rankeo, destacando:
Ranking Fonseca y Flemming
Ranking Sistler and Thiele
RAnking Goldberg

2.2. Aptitud Compartida

También llamado Shared Fitness, es un método que se añade para los M.O.E.A.'s que va ligado al de ranking.
Una definicion ad-hoc al asunto evolutivo india que el Shared Fitness trata de realzar una seleccion mas profunda cuando dos individuos tienen el mismo ranking. De este modo eñ Shared Fitness permite llevar a cabo un desempate
Mencionar el Fitness De la poblacion, hay de dos tipos: fenotipico y genotipico, poner la formula.

2.3. Algoritmos

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 cualitativa 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.
Habiendo integrado todos los elementos anteriores, a continuacion se exponen distintas maneras de llevar a cabo un proceso evolutivo multiobjetivo, cada una con sus propias caracteristicas intrinsecas y oportunidades de mejora

2.3.1. 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.

¿Mencionar esto? 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 pues es el más sencillo de implementar.

2.3.2. 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

2.3.3. 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;

2.3.4. 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

2.4. M.O.P.

Un Problema MultiObjetivo (MultiObjective Problem) es un escenario bien estudiado que contiene funciones objetivo y ariables de decision que se ejecutan bajo determinadas coniciones iniciales. La meta de un M.O.P consiste en ver el comportamiento de la solución bajo ciertos algoritmos, en estecaso los que se toman como ejemplificaciones (MOGA; SPEA2, NSGA, VEGA), Ademas, para fines de este trabajo, el papel del M.O.P. es fundamental para el desarrollo del producto de software puesto que esas comparaciones permitieron calibrar el programa para volverlo útil y práctico para el lector.
Para esta entrega, los M.O.Ps empleados son:
Añadir tabla

2.5. Indicadores de Desempeño

Los indicadores de desempeño con

2.6. Introducción a M.O.E.A Software

Habiendo detallado todos los elementos

M.O.E.A. Software surge como una solución ante la necesidad inicial de tener visualizadores adecuados donde se puedan interpretar adecuadamente los M.O.E.A's y sus correspondientes modificaciones.
Si bien existen tanto en el mundo de codigo libre como privativo soluciones asequibles, no han abordado aun de manera grafica escenarios multobjetivo por lo que es común tener limitantes con los analisis.
Para abordar detalles de índole más técnica se recomienda visitar la sección **Características Técnicas** perteneciente al Apéndice.

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] hola mundo

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

[4] 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.

[5] 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.

2.7. Mesografía

[6] Verhoef, Herman A. *Community Ecology*. March, 2015. *http://www.oxfordbibliographies.com/view/document/obo-9780199830060/obo-9780199830060-0042.xml*

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

[8] 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.

2.8. 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.

2.9. 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 desoftware 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 podemos señalar 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 métodosrelacionados 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 Windows y GNU/Linux por su popularidad [8], 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 quese 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 crearonpor 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.