**INSTITUTO TECNOLÓGICO DE CIUDAD MADERO**
**División de estudios de posgrado e investigación**

"POR MI PATRIA Y POR MI BIEN"

# ESTUDIO DEL IMPACTO DE PATRONES DE DISEÑO EN LA IMPLEMENTACIÓN DE UN FRAMEWORK DE OPTIMIZACIÓN PARA APOYO A LA TOMA DE DECISIONES

**TESIS**

Para Obtener el Título de:

**Maestro en Ciencias de la Computación**

**Presenta:**
**I.S.C. RICARDO ROJAS HERNÁNDEZ**
**G08070810**

**Director de Tesis:**
**DR. NELSON RANGEL VALDEZ**

**Co-Directora de Tesis:**
**DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN**

**Comité Tutorial:**
**DRA. GUADALUPE CASTILLA VALDEZ**
**DRA. LAURA CRUZ REYES**
**DRA. CLAUDIA GÓMEZ SANTILLÁN**

Cd. Madero, Tamaulipas Mayo de 2017

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Cd. Madero, Tamps; a 12 de Mayo de 2017.

OFICIO No.: U5.068/17
AREA: DIVISIÓN DE ESTUDI
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

**ING. RICARDO ROJAS HERNÁNDEZ**
**NO. DE CONTROL G08070810**
**P R E S E N T E**

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su examen de grado de Maestría en Ciencias de la Computación, el cual está integrado por los siguientes catedráticos:

| | |
|---|---|
| PRESIDENTE : | DRA. LAURA CRUZ REYES |
| SECRETARIO : | DRA. GUADALUPE CASTILLA VALDEZ |
| VOCAL : | DR. NELSON RANGEL VALDEZ |
| SUPLENTE | DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN |
| | |
| DIRECTOR DE TESIS: | DR. NELSON RANGEL VALDEZ |
| CO-DIRECTORA DE TESIS: | DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN |

Se acordó autorizar la impresión de su tesis titulada:

**"ESTUDIO DEL IMPACTO DE PATRONES DE DISEÑO EN LA IMPLEMENTACIÓN DE UN FRAMEWORK DE OPTIMIZACIÓN PARA APOYO A LA TOMA DE DECISIONES "**

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta.

Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

**A T E N T A M E N T E**
"POR MI PATRIA Y POR MI BIEN"®

**DRA. ADRIANA ISABEL REYES DE LA TORRE**
**JEFA DE LA DIVISIÓN**

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO
E INVESTIGACIÓN

c.c.p.- Archivo
Minuta

AIRT'NLCO 'jar

Ave. 1° de Mayo y Sor Juana I. de la Cruz Col. Los Mangos, C.P. 89440 Cd. Madero, Tam.
Tel. (833) 357 48 20. e-mail: itcm@itcm.edu.mx
www.itcm.edu.mx

## Declaración de Originalidad

Declaro y prometo que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derecho de publicación, derechos de autor, patentes y similares.

Además, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

Además, en caso de infracción a los derechos de terceros derivados de este documento de tesis, acepto la responsabilidad de la infracción y relevo de ésta a mi director y co-director de tesis, así como al Instituto Tecnológico de Cd. Madero y sus autoridades.

Mayo de 2017, Cd. Madero, Tamaulipas.

I.S.C. Ricardo Rojas Hernández

## *Agradecimientos*

Ofrezco todo mi agradecimiento y respeto a cada una de las personas que contribuyeron de manera directa o indirecta en la elaboración de este trabajo, ya que sin su apoyo jamás se habría llegado al objetivo.

Le agradezco a mi familia que me brindó su apoyo en todo momento; a mi madre Ofelia Hernández Martínez que siempre me alentó a seguir adelante; mi padre que Julián Rojas Hernández por sus sabios consejos; y a mis hermanos Jorge Luis Ontiveros Hernández y María Guadalupe Rojas Hernández por su gran solidaridad.

También agradezco a mis compañeros de generación que me brindaron su apoyo y amistad incondicional; a Leonor Hernández Ramírez por sus comentarios asertivos; Enith Martínez Cruz por sus grandes aportaciones; y a Lemuel Rodríguez Moya por su gran amistad.

Agradezco a mi director de tesis el Dr. Nelson Rangel Valdez, por su enorme apoyo y asesoramiento brindado, a lo largo de estos dos últimos años en el desarrollo de este trabajo. Además también agradezco sus valiosas aportaciones y observaciones, que me permitieron alcanzar el objetivo de este trabajo.

Agradezco también a mi Co-asesora, Dra. Claudia Guadalupe Gómez Santillán por creer en mí en todo momento, por brindarme consejos que me orientaron de manera acertada y su gran paciencia al asesorarme cuando lo necesitaba.

Gracias a las Dras. Laura Cruz Reyes y Guadalupe Castilla Valdez por su enorme ayuda incondicional, que me han otorgado a lo largo de todo este tiempo. También por sus valiosas observaciones que me permitieron mejorar aún más mi trabajo de tesis.

Mi más profundo agradecimiento al Consejo Nacional de Ciencia y Tecnología (CONACYT), y al Instituto Tecnológico de Ciudad Madero (ITCM), por haberme otorgado el apoyo en la realización de este proyecto.

Por último le doy gracias a Dios por haberme dado la fuerza y entendimiento necesario, en alcanzar una más de mis metas más anheladas en el ámbito profesional.

# CAPÍTULO 1

## 1. INTRODUCCIÓN

Hoy en día, es posible observar cómo la solución a problemas de optimización complejos juega un papel importante dentro de la actividad diaria desarrollada tanto en el sector productivo como en el académico. Por mencionar algunos de esta gran variedad de problemas están, diseño de circuitos [Bhatt&Leighton, 1984], programación de horarios [Sotskov et al., 2002], pruebas de software [Cohen et al., 1996; Michael et al.,1997], entre otros.

Para lidiar con la complejidad inherente de los problemas de optimización en general [Garey& Johnson, 1979], existe una gran diversidad de estrategias que lo pueden resolver tanto de forma óptima (e.g. en problemas de etiquetado hay ramificación y poda [Martí et al., 2008], modelos de programación lineal entera y programación dinámica [Saxe, 1980], entre otros), como aproximada (e.g. algoritmos voraces [Gibbs et al., 1976], metaheurísticas [Rodriguez et al., 2008], etc.).

Tradicionalmente, los tipos de algoritmos previamente descritos resuelven problemas de optimización mediante mecanismos que comúnmente definen un espacio de búsqueda particular, por medio de estructuras especializadas (e.g. en estructuras de estrategias como [Gibbs et al., 1976; Saxe, 1980;Martí et al., 2008;Rodriguez et al., 2008]). Las estructuras normalmente empleadas en la solución de los problemas de optimización, como parte del diseño de un algoritmo, son estructuras rígidas basadas en condicionales y ciclos, y reglas de decisión que guían de forma imperativa hacia donde debe moverse el algoritmo.

Una alternativa al uso de reglas rígidas basadas en condicionales, es el uso de estrategias basadas en control de conocimiento [Sohrabi et al., 2014], y en aprendizaje [Figueira et al., 2005]. Las estrategias basadas en control de conocimiento son comunes en el área de inteligencia artificial, y hacen uso de lenguajes como PDDL (del inglés *Planning Domain Definition Language*) para codificar preferencias que guíen la búsqueda de soluciones [Huang&Selman, 1999]. De acuerdo al análisis hecho sobre la literatura revisada, se establece que esta alternativa tiene como desventajas los siguientes aspectos: a) está acotada comúnmente a un conjunto pequeño de lenguajes de programación; b) suele utilizarse para resolver sólo tareas de planeación y de logística; y c) las preferencias no pueden cambiar a lo largo de su ejecución.

Por otro lado, las estrategias basadas en aprendizaje, pueden ser flexibles y permitir incorporar nuevo conocimiento al cambiar los conjuntos de referencia (historial de casos, o casos de prueba) [Fernandez & Navarro, 2012]. Un área de oportunidad para el desarrollo de este último tipo de estrategias es permitir el cambio dinámico del conjunto de referencia, es decir, permitir un ajuste dinámico del modelo de aprendizaje, cuando se busca la solución del problema. El uso de los clasificadores comúnmente utilizados en inteligencia artificial puede hacer costosa esa operación (por la complejidad del modelo), e incluso limitar el área de aplicación. Por el otro lado, los modelos de preferencia, empleados en investigación de operaciones [Figueira et al., 2005; Fernandez et al., 2013], son lo suficientemente sencillos para centrar la atención en ellos para el desarrollo de la presente investigación, los detalles al respecto se dan a continuación.

Desde hace varias décadas, el modelado de preferencias es una actividad inevitable en numerosas áreas como economía [Armstrong, 1939], sociología [Chisholm& Sosa, 1966], inteligencia artificial [Doyle & Wellman, 1992], ciencias de la computación [Fishburn, 1999], investigación de operaciones [Perny & Roy, 1992], programación matemática [Perny & Spanjaard, 2002], entre otras. La construcción de estos modelos sirve para entender mejor y representar de mejor manera el conocimiento derivado de una situación particular [Dias et al., 2002], ya sea por decisiones que hay que tomar, o por información que hay que obtener. A través de los modelos de preferencia se hace posible el comparar la similitud entre objetos, de tal manera que se contribuya en la automatización de procesos para la solución de problemas donde dichas diferencias sean importantes [Fernandez et al., 2013].

Para ampliar el dinamismo del aprendizaje, y la adaptabilidad con diferentes algoritmos, se considera que es posible incorporar mecanismos empleados en investigación de operaciones basados en modelos de preferencias; los cuales definen un conjunto de parámetros que caracterizan estrategias de búsquedas (o preferencias) de acuerdo a un decisor [Perny & Spanjaard, 2002]. Como ya se ha visto en la literatura, es posible resolver problemas de optimización a través del apoyo de modelos de preferencia, como en [Fernandez et al., 2013], sin embargo, la capacidad de solución está limitada en el número de estrategias que hacen uso de dichos modelos y los problemas que se resuelven (sólo observado en cartera de proyectos) [Fernandez et al., 2013]. Por esta razón, este proyecto considera el estudio de estrategias existentes, y explora la posibilidad de generación de nuevas estrategias apoyadas en métodos basados en el área de toma de decisiones (el caso particular del uso de modelos de preferencias).

Ahora bien, a través de un análisis lógico es posible la comprensión y representación de la estructura de un problema de optimización. De poder llevar a cabo la caracterización de ese conocimiento, se podrían plantear *agentes decisores* (es decir, tomadores de decisiones derivados del conocimiento general de los objetivos del problema, y no exclusivamente de las preferencias de un humano) que faciliten la búsqueda de soluciones a problemas de optimización en general. De esta manera, un modelo del *agente decisor*, basado en modelos de preferencia, se podría incorporar en una estrategia de solución para problemas de optimización en general, y reducir considerablemente el espacio de búsqueda en ellos. Sin embargo, de acuerdo a la literatura revisada hasta nuestro conocimiento, no existe una estrategia de optimización que incorpore las preferencias de un *agente decisor* como guía de la búsqueda, y que dichas preferencias se vayan ajustando automáticamente.

Un primer acercamiento involucraría estudiar las estrategias para apoyo a la toma de decisiones, observando primero su versatilidad para incorporarse como modelos de *agentes decisores*, posteriormente analizando la robustez de sus soluciones, y finalmente empleándose para el diseño de nuevas estrategias de optimización, permitiendo la incorporación automática de preferencias. La importancia de este trabajo consiste en proponer un nuevo modelo de solución a problemas, que integre estrategias de apoyo a la toma de decisiones, en estrategias de optimización basadas en heurísticas y/o metaheurísticas con el fin de reducir el espacio de soluciones explorado.

La innovación es en la disciplina de Ciencias de la Computación, en particular en *Optimización Inteligente*, ya que el conocimiento generado involucra la creación de nuevas estrategias de solución para problemas de optimización, considerando un nuevo nivel de inteligencia que hace uso de las preferencias características del problema.

Para llevar a cabo la integración de modelos de optimización y estrategias de apoyo a la toma de decisiones, se propone el desarrollo de un framework de optimización. El desarrollo del framework contempla las siguientes actividades: a) Análisis de patrones de diseño aplicables en el área de optimización; b) Desarrollo de un diseño experimental genérico que sea aplicable a diferentes problemas de optimización, y metaheurísticas de solución; c) Implementación del framework en un lenguaje orientado a objetos; y d) Validación del framework y su funcionalidad esperada. A lo largo del desarrollo de estas etapas se contemplan llevar a cabo estudios de la robustez (la calidad de la solución obtenida), validaciones, y pruebas sobre los modelos implementados en el framework, así como de su flexibilidad, y facilidad de implementación.

Finalmente, el impacto del conocimiento generado por este proyecto, que consiste en los esquemas basados en patrones de diseño que permitirán hacer flexible la implementación de nuevas estrategias, incorporarlas al framework, y validarlas a través de un diseño experimental igualmente flexible de elaborar.

## 1.1 Antecedentes del proyecto

Este proyecto forma parte de un macro-proyecto que involucra a la Universidad Autónoma de Sinaloa (UAS), la Universidad Autónoma de Nuevo León (UANL) y el Instituto Tecnológico de Ciudad Madero (ITCM) con el objetivo de probar la posibilidad de una colaboración en conjunto. El problema principal que se busca tratar en dicho macro-proyecto es la optimización de carteras de proyectos, lo cual requiere la aplicación de modelos y técnicas pertenecientes a diferentes disciplinas pertenecientes a las ciencias de la computación y las matemáticas.

La selección de carteras se ha definido en cuatro fases principales, donde la interacción con el usuario es de vital importancia en cada una de estas etapas. Las primeras dos fases son la *captura de proyectos* y la *evaluación de proyectos,* estableciendo los objetivos a evaluar, las propuestas del proyecto y haciendo un análisis para definir la aportación y factibilidad de cada uno de los objetivos. Por último se lleva a cabo las fases de *optimización de carteras* y la *presentación de la recomendación.* En estas fases se buscan las mejores carteras de proyectos que satisfagan los criterios del tomador de decisiones, dichas carteras se le muestran al DM para que defina sus preferencias con respecto a ellas. El ITCM está apoyando el desarrollo de algoritmos de optimización para resolver el problema de cartera de proyectos sociales (SPP por sus siglas en ingles), aprovechando la experiencia de sus miembros en la creación de algoritmos para la resolución de problemas de alta complejidad.

Un prototipo de un entorno de trabajo para un sistema que apoye al usuario a tomar una decisión fue propuesto por Balderas [Balderas, 2011], en donde se le apoya al tomador de decisiones para que descubra sus preferencias mediante módulos relacionados a la interacción, apoyándose de la interacción visual, todo esto de forma intuitiva.

## 1.2 Justificación

El desarrollo del Framework de optimización, bajo el análisis de patrones de diseño que se puedan incorporar en su implementación, impactará en las siguientes actividades:

- Disminuirá el tiempo de implementación de nuevas estrategias, al permitir reutilizar el código ya existente;
- Aumentará la flexibilidad de estrategias existentes, al permitir extenderlas; y
- Permitirá configurar procesos de experimentación de forma rápida, y obtener un resumen de sus resultados fácilmente.

La mejora de las actividades comentadas previamente impactará en cualquier trabajo de investigación, ya que permitirá llevar a cabo la prueba y validación de nuevas procesos en menor tiempo.

## 1.3 Objetivos del proyecto

### 1.3.1 Objetivo general

Implementar un Framework de optimización que facilite la integración de nuevas estrategias, y la extensión de estrategias existentes, dentro del área de apoyo a la toma de decisiones.

### 1.3.2 Objetivos específicos

- Analizar patrones de diseño en optimización para diseñar estrategias basadas en metaheurísticas que apoyen a la toma de decisiones.

- Desarrollar un conjunto de indicadores que permitan medir la calidad del uso de los patrones de diseño en la implementación y prueba de metaheurísticas para apoyo a la toma de decisión.

- Seleccionar el conjunto de patrones de diseño que permita diseñar el Framework de optimización bajo un esquema de código reutilizable, y de fácil implementación.

- Implementar el Framework de optimización, y validar su impacto en la creación de nuevas estrategias, y en la experimentación.

## 1.4 Alcances y limitaciones

- Dos problemas de optimización. La primera propuesta se abordará el Problema de Cartera de Proyectos Públicos, el cual fue descrito en la sección 3.3; la segunda es el problema de Job Shob Scheduling, que será descrito en la sección 3.4;

- Realización de al menos dos algoritmos multi-objetivos y dos mono-objetivos.

- Un subconjunto de patrones de diseño que permitan reutilizar código existente tanto para la generación de nuevas estrategias de solución, como para llevar a cabo el proceso de experimentación; y
- Un prototipo del Framework que sea funcional, y permita validar el funcionamiento correcto de la propuesta.

El desarrollo del Framework de la propuesta está acotado por los siguientes aspectos:

- La validación contemplará instancias pequeñas del problema de optimización con un máximo de 3 objetivos y 25 proyectos;

- La interfaz de usuario se implementará en un ambiente local; y

- El uso del lenguaje de programación será orientado a objetos, y en particular el Lenguaje Java;

### 1.5 Contenido de la tesis

**Capítulo 2:** Marco Conceptual. En esta sección se presenta las bases teóricas relacionadas a este trabajo de tesis, como por ejemplo: problemas de toma de decisiones, optimización multi-objetivo, patrones de diseño y diseño experimental.

**Capítulo 3:** Descripción del Problema y Estado del Arte. Esta sección está compuesta en dos partes. La primera parte se define y se describe el problema a resolver en este trabajo. En la segunda parte se expone la literatura investigada, que fue necesaria en la resolución del problema establecido. Las literatura más sobresaliente en esta investigación, fueron los distintos patrones de diseño que se utilizan en la resolución de problemas de optimización, y los frameworks de optimización más sobresalientes para nuestro propósito, que posteriormente se realiza la comparación de sus características con respecto al de este trabajo.

**Capítulo 4:** Propuesta de Solución. En esta sección se presenta la metodología utilizada para la selección de los patrones de diseño más adecuados, en la construcción de framework de optimización. Algunos de los elementos utilizados en esta metodología son: el Lenguaje de Modelado Unificado (UML), utilizado para representar las clases contenidas en el framework; el algoritmo genético, utilizado para evolucionar los patrones de diseño; y

claramente, un conjunto tentativo de patrones de diseño. Por otro lado, también se muestra una tabla que describe las características de todos los módulos de optimización contenidos en el framework.

**Capítulo 5:** Diseño de la Solución. En esta sección de presenta la forma de cómo se evolucionaran los patrones de diseño, mediante el algoritmo genético. Este proceso está dividido en 5 etapas: la primera aborda la construcción de la solución, que describe la forma en cómo serán abordados los patrones de diseño en dicha solución; la segunda etapa es la función de aptitud, en donde se emplea una métrica para medir la aptitud de cada solución; la tercera etapa es la de selección, en donde se realiza un proceso de clasificación de las soluciones que pertenecerán a nuestra población; la cuarta etapa es la cruza, la cual permite entrelazar la soluciones padres para generar nuevas soluciones hijas; y por último la etapa de muta, la cual permite perturbar algunas soluciones hijas.

**Capítulo 6:** Experimentación y Resultados. En esta sección se divide en dos secciones: la primera es el ambiente experimental, que son las condiciones donde se llevaron a cabo las pruebas, y la segunda etapa, es la experimentación, donde se describen los resultados obtenidos por el algoritmo genético y el análisis de sus resultados.

**Capítulo 7:** Conclusiones y Trabajos Futuros. En esta sección se presenta las conclusiones del trabajo de tesis y los trabajos futuros que podrían darle seguimiento a este proyecto.

**Apéndice A:** Instancias utilizadas. En esta parte se presentan las instancias utilizadas en las experimentaciones de este trabajo y su descripción.

**Apéndice B:** Incorporación de otros frameworks de optimización. En esta sección se llevó a cabo una nueva experimentación, pero considerando otras arquitecturas de frameworks, como son: JAMESv3.1, JMetalv3.1 y JMetalv5.0.

# CAPÍTULO 2

## 2. MARCO CONCEPTUAL

Dentro de esta apartado se definen los conceptos más importantes relacionados, con la definición de la presente propuesta de investigación.

### 2.1 Optimización

De acuerdo a Scenna et al. 2015, define optimización como: "al proceso de seleccionar, a partir de un conjunto de alternativas posibles, aquella que mejor satisfaga el o los objetivos propuestos". De tal modo que al querer solucionar un problema de optimización, es muy deseado encontrar la mejor solución entre el conjunto de posibles soluciones. Por tal motivo se propone un framework capaz de incorporará algoritmos y técnicas pertenecientes al área de MCDA; y que a su vez permitirán resolver problemas de optimización, los cuales estarán dentro de toma de decisiones.

### 2.2 Toma de decisiones

La toma de decisiones no es sólo un resultado, sino el desarrollo complejo de la selección de criterios (y sus medidas), determinación de alternativas (u opciones) de recolección, evaluación y procesamiento de información. Además, también es la forma de producción y evaluación de resultados parciales o intermedios, reconsideración de criterios, alternativas e información sobre la base de los resultados obtenidos, repitiendo (reciclando) el proceso hasta que se alcance un resultado útil (una decisión). Cada decisión parcial influye y enmarca el contexto de la siguiente [Zeleny, 2008].

#### 2.2.1 El problema de toma de decisiones

Hablando en rasgos generales, tomar una decisión reside en seleccionar la mejor opción o alternativa de entre un conjunto de alternativas posibles. La duda suele ser una compañera presente en los procesos de toma de decisiones, el cual produce malestar e inseguridad a las personas que deben escoger las decisiones [Sánchez, 2007].

La toma de decisiones en base a Keeney & Raiffa, (1993), tiene como objetivo ayudar a los individuos a tomar decisiones difíciles y complejas de una forma racional. Debido a la racionalidad, necesita el desarrollo de métodos y/o modelos que permitan plasmar de una forma fiel cada problema y analizar las diferentes alternativas con criterios objetivos. Por

tal motivo, es necesario partir de las disciplinas clásicas como la Estadística, la Economía y la Matemática, a la cual se le ha unido la Inteligencia Artificial, para desarrollar teorías y modelos en el campo de la toma de decisiones. Estas han permitido estructurar, de forma lógica y racional, el proceso de toma de decisión y disminuir esta tarea a los individuos encargados de llevarla a cabo [Sánchez, 2007].

En cualquier problema de decisión, al encargado de tomar la decisión final se le llama "Decision-Maker" (DM). Este elemento central es una persona o grupo, cuyo sistema de preferencias es determinante en la solución de problemas que consideran varios objetivos, los cuales posiblemente se encuentren en conflicto entre sí [Fernández et al., 2011]. Estas preferencias van a tener una repercusión en el resultado final; pues como se ha mencionado con anterioridad, al haber conflicto en los objetivos, no existirá una solución óptima.

### 2.2.2 Análisis de Decisión Multicriterio (MCDA)

Hoy en día es bien aceptado que el proceso de toma de decisiones se extiende más allá del modelo clásico: la optimización de una sola función objetivo sobre un conjunto soluciones factibles [Guitouni & Martel, 1998]. De tal manera que al manejar muchos aspectos en contraste en un mismo tiempo, no es posible encontrar una solución óptima, pero si una satisfactoria. Esta solución puede ser encontrada bajo un conjunto de técnicas conocidas dentro del área de MCDA.

De acuerdo en el libro de Keeney and Raiffa's, (2006), definen a MCDA como "una extensión de la teoría de la decisión que cubre cualquier decisión con múltiples objetivos. Una metodología para evaluar alternativas en persona, a menudo contradictorios criterios, y combinarlas en una sola evaluación global…". A sí mismo, MCDA comprende un amplio conjunto de enfoques metodológicos procedentes de la investigación de operaciones [Köksalan et al., 2011].

Una de las técnicas perteneciente al MCDA que se revisa en este trabajo, es la de ELECTRE III. La relación outranking de éste modelo puede ser interpretada como una relación difusa [Figueira et al., 2005]. La construcción de esta relación requiere la definición del *índice de credibilidad* $\sigma(x, y)$, que se caracteriza por la afirmación de que "$x$ es al menos tan bueno como $y$ desde el punto de vista del *DM*" [Fernández et al. 2010]. Este se encuentra definido por los índices: *concordancia* y *discordancia*, denotados por $c(x, y)$ y $d(x, y)$ respectivamente.

En este trabajo propone el diseño de un módulo que permita incorporar relaciones de preferencia definidas por modelos como ELECTRE III, en algoritmos de optimización, como el Non-Outranked-Sorting Genetic Algorithm (NOSGA), que está basado en NSGA II (Non-dominated Sorting Genetic Algorithm II) [Deb et al., 2002].

### 2.2.3 Modelos de preferencias

El apoyo a la decisión está basado en la elaboración de la información preferencial. La idea básica en la metodología de ayuda a la decisión es que, dado un problema de decisión, recopilamos información preferencial del tomador de decisiones de tal manera que su sistema de valores está representado fielmente o críticamente construido y por lo tanto somos capaces de construir un modelo que, cuando se aplica, debe de hacer la recomendación de acción para el tomador de decisiones.

***Modelado de preferencias:*** Tiene como objetivo construir relaciones de preferencia en un conjunto de acciones, conjunto *A*, que se evalúan con respecto a un criterio. La representación de las preferencias del tomador de decisiones sobre el conjunto $A$ constituye un paso crucial en la ayuda a la decisión. Dependiendo del contexto del problema, la naturaleza de la información que somos capaces de manejar y las expectativas del tomador de decisiones, diferentes situaciones pueden aparecer.

***Estructuras de Preferencia:*** El concepto matemático de *relación binaria* es comúnmente usado como una representación formal de relaciones de preferencia (modelos) definida en un conjunto finito dado que dichas relaciones son el resultado de la comparación de dos elementos.

Por lo tanto, teniendo en cuenta que $A$ es un conjunto finito, la comparación por pares de sus elementos pueden resultar en relaciones binarias indiferentes que tienen propiedades diferentes. Decimos que tales relaciones construyen una estructura de preferencias, si cumplen ciertas condiciones.

Una estructura de preferencias [Oztürk et al., 2005] es un conjunto de relaciones binarias $\{S_1, \ldots, S_m\}$ definida en el conjunto $A$ tal que:

$\forall\, x, y \in A \exists\, i \in \{1, \ldots, m\}$, $xS_iy$ o $yS_ix$. Significa para cada pareja *x, y* en *A*; al menos una relación se cumple;
$\forall\, x, y \in A$, $xS_iy \Rightarrow \forall j \neq i$ , no $(xS_jy)$ y no $(yS_jx)$, lo que significa que para cada pareja *x, y* en *A*, si una relación es satisfecha, otra no puede ser satisfecha.

Una relación binaria grande, también llamado una relación de *outranking*, denotado por *S*, se puede utilizar para caracterizar una estructura de preferencia. Esta relación se interpreta como "a es al menos tan buena como *b*", denotado como $a \ \underline{\mathbf{f}}\ b$ o *aSb*, tal que $\underline{\mathbf{f}} \subset A \times A$.

Las reglas de preferencia del DM no están bien definidas. La existencia de zonas de incertidumbre en la mente del DM puede deberse a creencias imprecisas, conflictos o aspiraciones que se oponen entre sí [Roy, 1990]. Roy [Roy, 1996] describe situaciones donde se presenta este comportamiento no ideal en el DM. Debido a esto usa un sistema preferencial compuesto de varias relaciones binarias. A continuación se describen dichas relaciones:

***Indiferencia*:** Corresponde a la existencia de razones claras que justifican una equivalencia entre las dos alternativas. Se denota como *xIy*.

***Preferencia estricta*:** Corresponde a la existencia de razones claras que justifican una preferencia significativa en favor de alguna de las dos alternativas. La declaración "*x* es estrictamente preferida sobre *y*" se denota como *xPy*.

***Preferencia débil*:** Corresponde a la existencia de razones claras para preferir a *x* sobre *y*, pero que no son lo suficientemente significativas como para justificar una preferencia estricta. Se utiliza cuando la indiferencia y la preferencia estricta no pueden ser claramente distinguidas. Es denotada como *xQy*.

***Incomparabilidad:*** Ninguna de las situaciones anteriores predomina. Esto es porque faltan razones claras que justifiquen cualquiera de las relaciones anteriores. Se denota como *xRy*.

***Sobreclasificación**:* Corresponde a la existencia de razones que justifican la declaración "*x* es al menos tan bueno como *y*", pero sin una división claramente establecida entre preferencia estricta, preferencia débil o indiferencia. La sobreclasificación se denota como *xSy*.

***k-Preferencia**:* Corresponde a la existencia de razones claras que justifican ya sea una preferencia estricta o una incomparabilidad, pero sin existir una diferencia establecida entre ambas situaciones. Se denota como *xKy*.

***No preferencia:*** Corresponde a la situación en la cual tanto la indiferencia como la incomparabilidad son posibles, sin existir una diferencia clara entre ellas. Se denota como *x~y*.

## 2.3 Optimización multi-objetivo

Una gran parte de los problemas de decisión del mundo real implican el procesamiento simultáneo de múltiples criterios [Pérez et al., 2007]. Los criterios, también llamados objetivos, con frecuencia están en conflicto, es decir, al mejorar uno de los objetivos, los demás objetivos se pueden ver afectados (en forma negativa).

El término optimización, según Kalyanmoy Deb [Deb, 2001], es la acción de encontrar una o más soluciones factibles que corresponden a valores extremos de uno o más objetivos. Cuando un problema de optimización comprende solamente una función objetivo, se le llama optimización mono-objetivo, cuya meta es encontrar la solución óptima. Cuando nos encontramos con un problema de optimización, donde se involucra más de una función objetivo, se le conoce como optimización multi-objetivo, cuya meta es la de encontrar una o más soluciones óptimas.

### 2.3.1 Optimalidad de pareto

En un problema de optimización multi-objetivo (MOP) por sus siglas en inglés *Multi-objetive Optimization Problems*; se busca un conjunto de soluciones que presenten un equilibrio óptimo entre los diferentes objetivos. A la gráfica de éste conjunto de soluciones-equilibradas presentadas en el espacio de objetivos se le denomina Frente de Pareto, por esta razón, estas soluciones son conocidas como soluciones óptimas de Pareto o soluciones no-dominadas [Van Veldhuizen & Lamont, 1998].

En problemas multi-objetivo, es muy común que los objetivos se encuentren en conflicto entre sí. Por ello, se buscan normalmente *soluciones compromiso* en vez de una única solución. Debido a esto, la noción de óptimo es diferente en estos casos. La noción de óptimo más comúnmente aceptada es la que propuso Francis Ysidro Edgeworth [Edgeworth, 1881], la cual más tarde fue generalizada por Vilfredo Pareto [Pareto, 1896]. Aquella noción es comúnmente llamada con el término de *optimalidad de Pareto*.

Se dice que un vector de variables de decisión: $\overrightarrow{x^*} \in \mathcal{F}$ es un óptimo de Pareto existente en la región factible $\mathcal{F}$ si (asumiendo minimización):

$$f_i(\overrightarrow{x^*}) \leq f_i(\vec{x}), \forall i \in [1,2,\dots,k] \text{ y}$$
$$f_j(\overrightarrow{x^*}) < f_j(\vec{x}), \exists j \in [1,2,\dots,k]$$

En palabras, $\overrightarrow{x^*}$ es un óptimo de Pareto si no existe otro vector factible de variables de decisión $\vec{x} \in \mathcal{F}$ que mejore algún criterio sin causar el deterioro simultáneo de al menos otro criterio. Sin embargo, este concepto casi siempre ofrece no una, sino varias soluciones

llamadas *Conjunto de óptimos de Pareto* (*P**). Los vectores $\overrightarrow{x^*}$ correspondientes a las soluciones incluidas en el conjunto de óptimos de Pareto son llamados *no-dominados*. La imagen del conjunto de óptimos de Pareto bajo las funciones objetivo es llamada frente de Pareto.

### 2.3.2 Dominancia de pareto

En algoritmos multi-objetivo, se utiliza con mucha frecuencia el concepto de *dominancia de Pareto* al contrastar dos soluciones y decidir si una domina a otra o no. Se dice que una solución $\overrightarrow{x_a}$ domina a otra $\overrightarrow{x_b}$ si se cumplen las siguientes condiciones (para el caso de minimización):

1. La solución $\overrightarrow{x_a}$ no es peor que $\overrightarrow{x_b}$ en todos los objetivos, o:

$$f_i(\overrightarrow{x_a}) \leq f_i(\overrightarrow{x_b}), \quad \forall_i \in [1,2,\dots,k]$$

2. La solución $\overrightarrow{x_a}$ es estrictamente mejor que $\overrightarrow{x_b}$ en al menos un objetivo, o:

$$f_i(\overrightarrow{x_a}) < f_i(\overrightarrow{x_b}), \quad \exists_i \in [1,2,\dots,k]$$

Si alguna de las condiciones 1 y 2 no se cumple, entonces se dice que la solución $\overrightarrow{x_a}$ no domina a la solución $\overrightarrow{x_b}$. En otras palabras, para que una solución domine a otra, es necesario que sea estrictamente mejor en al menos un objetivo, y no peor en ninguno de ellos. Por ello, al comparar dos soluciones A y B, sólo pueden existir tres posibles soluciones:

- A domina a B
- A es dominada por B
- A y B no se dominan (son no dominadas entre sí)

La representación de las funciones objetivo cuyos vectores son no-dominados y además están en el conjunto de óptimos de Pareto es llamado el Frente de Pareto. Para un problema multi-objetivo planteado y un conjunto de óptimos de Pareto, el frente de Pareto se define como:

$$\mathcal{F}P^* = \left\{F(\vec{x}) = \left(f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})\right) \mid \vec{x} \in P^*\right\}$$

Hablando en general, no existe un método que sea completamente eficiente para encontrar el frente de Pareto, y la mejor forma de realizarlo, es hacer pruebas con todos y cada uno de los puntos en la zona factible Por lógica, el espacio de búsqueda es demasiado grande, por lo cual este proceso es incosteable y de ahí surge la necesidad de implementar heurísticas para generar aproximaciones del frente de Pareto.

A continuación, en la Figura 1 [Sánchez, 2017], se muestra de forma gráfica, el frente de Pareto, tanto para problemas de minimización como de maximización.

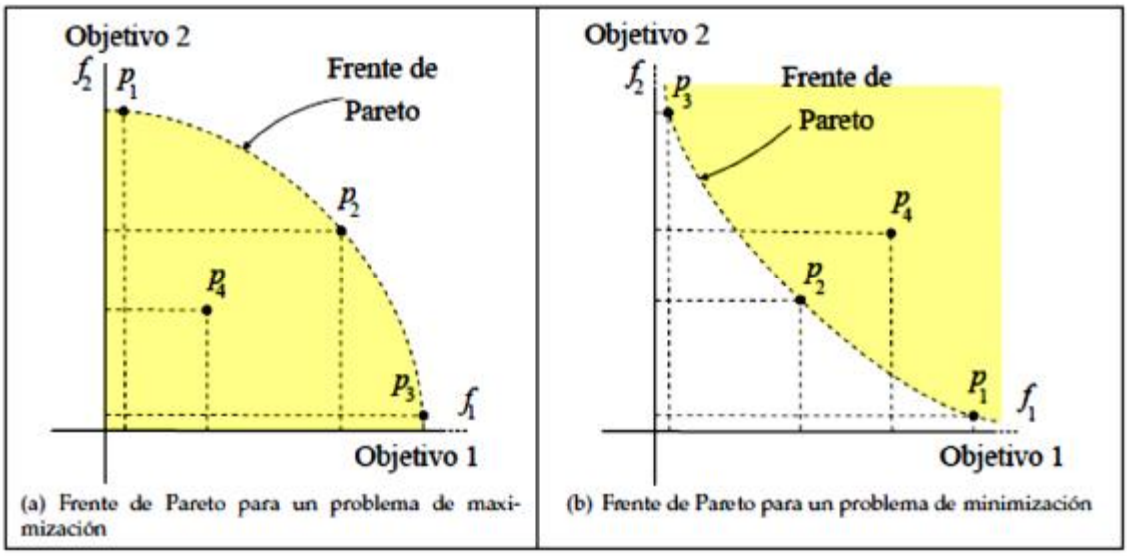



**Figura 1.** Frente de Pareto

### 2.3.3 Modelo Matemático para un Problema Multi-objetivo

Formalmente, en un problema multi-objetivo se busca el vector de variables de decisión

$\vec{x} = [x_i, x_2, \dots, x_n]^T$ que optimice a $F(\vec{x})$ [Serrano, 2007]. Por lo tanto, el modelo matemático quedaría de la siguiente manera:

$$F(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})], f_i: \mathbb{R}^n \rightarrow \mathbb{R}$$

Sujeto a:

$$g_i(\vec{x}) \leq 0; i = 1,2, \dots, m;$$
$$h_j(\vec{x}) = 0; j = 1,2, \dots, p$$

Donde:

$k$: es el número de funciones objetivo
$n$: es el número de variables de decisión
$m$: es el número de restricciones de desigualdad
$p$: es el número de restricciones de igualdad.

En otras palabras, se busca determinar el conjunto de todos aquellos números que satisfagan las restricciones y que optimicen todas las funciones objetivo. Las restricciones

definen la región factible del problema y todo vector que se encuentre en esta región será considerado como una solución factible [Santana, 2004]. Los problemas multi-objetivo son de tres tipos:

- Minimizar todas las funciones objetivo
- Maximizar todas las funciones objetivo
- Minimizar algunas y maximizar las funciones objetivo restantes

Para abordar la solución de MOPs, existen algoritmos exactos y aproximados. Dentro de los algoritmos aproximados, se encuentran estrategias deterministas y estocásticas. Una de las estrategias estocásticas de mayor popularidad los constituyen los algoritmos evolutivos, los cuales serán base de esta investigación.

### 2.3.4 Estrategias de optimización multi-objetivo

La optimización estocástica es una clase general de algoritmos que utiliza algún grado de aleatoriedad para conseguir soluciones óptimas (o las más cercanas posibles) a problemas de optimización difíciles. Las metaheurísticas son la clase más general de este tipo de algoritmos, algunos ejemplos muy utilizados en la literatura, son el recocido simulado, búsqueda tabú, optimización mediante colonia de hormigas y algoritmos evolutivos [Luke, 2009]. Sin embargo, debido a la forma en que trabajan dichos algoritmos, las soluciones que generan no siempre son las mejores posibles y, en general, su optimalidad no puede ser respaldada [Serrano, 2007].

Aunque a simple vista, esto pareciera otorgar una gran desventaja al respecto, muchos resultados experimentales han mostrado que una metaheurística bien diseñada, tendrá una gran probabilidad de proporcionar soluciones muy cercanas a las óptimas en tiempos de cómputo sensatos [Gendreau & Potvin, 2005].

*Metaheurísticas evolutivas*

Dentro de las metaheurísticas se encuentran los algoritmos evolutivos, que son una clase de métodos de optimización estocástica que simulan el proceso de la evolución natural, entre las metodologías propuestas destacan los algoritmos genéticos [Zitzler, 1999]. Los algoritmos evolutivos se caracterizan por contar con una población de posibles soluciones y un proceso de reproducción, el cual permite la combinación de soluciones existentes para generar nuevas soluciones. Los algoritmos evolutivos cuentan con una técnica de selección, la cual determina qué individuos de la población en turno participan en la nueva población. Este proceso permite encontrar varios miembros del conjunto de óptimos de Pareto en una sola ejecución del algoritmo, en lugar de tener que realizar una serie de ejecuciones por

separado, como ocurre con los algoritmos clásicos y con técnicas estocásticas no poblacionales (p.ej., el Recocido Simulado) [Serrano, 2007].

De acuerdo a [Goldberg, 1989], las principales ventajas que presenta el uso de los algoritmos evolutivos en la resolución de problemas de optimización son entre otras, las siguientes:

- Operan sobre una población (o conjunto de soluciones) lo que evita que la búsqueda se quede atascada en óptimos locales.
- No requieren conocimiento previo sobre el problema a resolverse.
- Pueden combinarse con otras técnicas de búsqueda para mejorar su desempeño.
- Permiten su paralelización de forma sencilla.
- Son conceptualmente fáciles de implementar y usarse.

*Algoritmos evolutivos multi-objetivo (MOEAs)*

Los algoritmos evolutivos para optimización multi-objetivo (MOEAs) por sus siglas en inglés (Multi-objective Evolutionary Algorithms) han mostrado su potencial para alcanzar la meta trazada por la comunidad de optimización multi-objetivo: aproximación bien distribuida y buena convergencia [Laumanns et al., 2002; Li, 2012; Tan, 2008]. Sin embargo, para fines prácticos, el DM no está interesado en que se le presente todo el frente de Pareto, sólo quiere una solución que corresponda con sus preferencias. Con la finalidad de encontrar esa única solución, nuevos métodos, que en su mayoría son variantes de los MOEAs existentes, utilizan las preferencias del DM para guiar la búsqueda hacia una porción preferida del frente, por ejemplo la región de interés (Region of Interest, ROI) definida en [Adra et al., 2007].

*Algoritmos evolutivos multi-objetivo basados en preferencias*

Se mencionarán algunas de las técnicas evolutivas que se han propuesto para incorporar preferencias en optimización multi-objetivo, dividiéndolas según la clasificación propuesta por Cohon y Marks [Cohon & Marks, 1975].

- *Técnicas a priori:* Las preferencias del usuario tienen que ser conocidas antes de que comience la búsqueda.
- *Técnicas progresivas:* En estas técnicas, las preferencias se van dando conforme la búsqueda avanza y el DM indica si una solución le parece adecuada o no y el proceso actualiza las preferencias conforme el DM lo va indicando, guiando así el proceso de búsqueda.
- *Técnicas a posteriori*: En estas técnicas, las preferencias se expresan al final y el DM recibe una información completa de los resultados para así entonces tomar la decisión

que mejor le convenga. Es decir, los resultados intentan mostrar todos los compromisos posibles entre las funciones objetivo tratando de generar el verdadero frente de Pareto o al menos una aproximación razonablemente buena.

## 2.4 Problema de cartera de proyectos públicos (PPP)

Un problema particular de optimización multi-objetivo es al que se conoce como: Problema de Cartera de Proyectos Públicos (Public Project Portfolio, PPP), el cual consiste en la selección de los proyectos públicos que más beneficios provean a la sociedad, por ejemplo, proyectos abocados a educación, salud y transporte público. Debido a su naturaleza, el problema de selección de proyectos ha sido comúnmente abordado mediante algoritmos multi-objetivo [Carazo et al., 2010; Doerner et al., 2004; Reiter, 2010] que se enfocan a identificar un conjunto de soluciones en el frente de Pareto.

### 2.4.1 Proyecto

Un proyecto es un proceso temporal, único e irrepetible que persigue un conjunto específico de objetivos [Carazo et al., 2010]. Así mismo, una cartera de proyectos es un conjunto de proyectos realizados en el mismo lapso de tiempo [Carazo et al., 2010]. Por tal motivo, los proyectos que se encuentran en una misma cartera, comparten los recursos disponibles en la organización, además, pueden complementarse entre ellos. Este es el motivo por el cual no es suficiente comparar los proyectos de manera individual, sino que se deben comparar grupos de proyectos para lograr identificar cuál cartera realiza una aportación mayor a los objetivos de la organización. La adecuada selección de proyectos para integrar la cartera, en la cual se invertirán los recursos de la organización, es uno de los problemas más importantes de decisión tanto para instituciones públicas como privadas [Castro, 2007; García, 2010].

### 2.4.2 Características del problema de cartera de proyectos públicos

Dentro del problema de cartera de proyectos públicos, el DM es el encargado de evaluar las acciones de política pública y sus consecuencias; la evaluación es a través de su particular visión que está influenciada principalmente por sus juicios, creencias e interpretación de la realidad. Por esta razón, se considera que el DM es el factor subjetivo del problema de decisión [Fernández et al., 2011].

En particular, los proyectos públicos se caracterizan por perseguir objetivos cuyo cumplimiento favorece a la sociedad. Dichos objetivos son generalmente intangibles, tal

como la repercusión social y científica, así como la formación de recursos humanos, entre otros, dejando de lado el posible beneficio económico como elemento principal de medida.

La selección de proyectos de una cartera de proyectos públicos necesita de un tratamiento especial por las siguientes razones [Fernández y Navarro, 2002]:

1. La calidad de los proyectos es generalmente descrita por múltiples criterios que frecuentemente están en conflicto.

2. Comúnmente los requerimientos no son conocidos con exactitud. Muchos conceptos no tienen un soporte matemático por ser de naturaleza totalmente subjetiva.

3. La heterogeneidad, o diferencia entre los objetivos perseguidos por los proyectos, dificulta compararlos.

El problema que es el objeto de estudio en esta investigación es un caso particular del problema de cartera de proyectos, cuya definición formal se presenta a continuación.

Se considera una cartera de proyectos a desarrollar, de los cuales el DM debe elegir algunos proyectos de dicho conjunto; esto con el fin de obtener un mayor grado de beneficios, sin excederse de los recursos establecidos. Considérese un vector $p$-dimensional $f(i) = \langle f_1(i), f_2(i), f_3(i), \ldots, f_p(i)\rangle$, donde cada $f_j(i)$ representa la contribución del proyecto $i$ para el objetivo $j$, por otra parte el portafolio es comúnmente modelado como un vector binario $x = \langle x_1, x_2,\ldots,x_N\rangle$ donde $x_i = 1$ si el proyecto $i$ es apoyado y $x_i = 0$ en caso contrario.

Considerando un límite de recursos $B$ y un costo $c_i$ para cada proyecto, la restricción de presupuesto se puede establecer de la siguiente manera:

$$\left(\sum_{i=1}^{N} x_i c_i\right) \leq B \qquad (1)$$

Además de la restricción de presupuesto, también deben tenerse en cuenta las restricciones por área, en otras palabras, que rango (máximo y mínimo) de la cantidad total del presupuesto se puede destinar a dicha área, para eso se formula la siguiente restricción:

$$L_k \leq \sum_{i=1}^{N} x_i g_i(k) c_i \leq U_k \qquad (2)$$

Donde $g$ está definido por:

$$g_i(k) = \begin{cases} 1 & if \quad a_i = k, \\ 0 & otherwise \end{cases} \tag{3}$$

De la misma forma, cada proyecto corresponde a un región geográfica que beneficiara, a estas regiones también se les aplica rangos máximos y mínimos y estas mismas restricciones. Se puede percibir que la solución al problema de cartera de proyectos requiere la toma de decisiones, la cual se puede abordar a través de la teoría de la argumentación, concepto que se aborda en la siguiente sección.

## 2.5 Problema de job shop scheduling

El problema de calendarización (scheduling) se refiera a la asignación de recursos compartidos sobre el tiempo de actividades competentes. Es conveniente adoptar la terminología fabricación, donde los *trabajos* (jobs) representan actividades y maquinas (*machines*) representan recursos, sin embargo, existe una amplia gama de áreas de aplicación de la teoría de calendarización, el cual no solo se limita a computadoras y fabricación, si no que incluye transportes, servicios, etc. [Yamada, 2003].

### 2.5.1 Descripción general

El objetivo de Jop Shop Scheduling Problem (JSSP) es organizar de manera óptima el orden de procesamiento y los tiempos de inicio de las operaciones para optimizar una determinada medida de desempeño.

Se tienen dos tipos de restricciones en este tipo de problema:

- *Restricción de secuencia.-* Expresa que las relaciones de precedencia entre las operaciones de un trabajo deben ser garantizadas

- *Restricción de recurso.-* Refiere que no más de un trabajo puede ser ejecutado en una máquina al mismo tiempo.

Un problema de JSSP se considera completamente solucionado si los tiempos de inicio de todas las operaciones son determinados y las restricciones de secuencia y recurso no son violadas [Marquez, 2012].

### 2.5.2 Características del problema de Job Shop Scheduling

Un JSSP está definido por el tamaño *j x m* donde *j* es el número de trabajos $J = \{J_1, J_2, J_3,\ldots J_j\}$ y *m* es el número de máquinas del problema $M = \{M_1, M_2, M_3,\ldots M_m\}$.

Para todo elemento del conjunto trabajos *j* deberá haber su correspondiente relación con cada uno de los elementos del conjunto de máquinas *M*. La relación de cada trabajo con cada máquina conforma una relación binaria de $J_j \rightarrow M_m$ que es llamada ***Operación (j,m)*** [Tellez, 2007]. El total de operaciones del problema es denotado por $O_{JM} = \{O_{1,1}, O_{1,2}, O_{1,3},\ldots, O_{j,m}\}$.

El conjunto de operaciones *O* correspondientes a un mismo trabajo tiene una secuencia de procesamiento llamada: ***Secuencia tecnológica S***, tal que $S = \{S_1, S_2, S_3,\ldots, S_m\}$ donde *m* es el número de máquinas del problema. Por lo tanto, para cada trabajo *j* tendremos la secuencia tecnológica: $O_{JS} = \{O_{j1}, O_{j2}, O_{j3},\ldots, O_{js}\}$.

Cada operación ***(j, m)*** tiene asociado un tiempo denominado: ***tiempo de procesamiento***, y es denotado por la letra *p*. Este parámetro indica el tiempo requerido de ese trabajo *j* en la máquina *m*.
En resumen:

| | |
|---|---|
| $M = \{M_1, M_2, M_3,\ldots M_m\}$ | Conjunto de máquinas |
| $J = \{J_1, J_2, J_3,\ldots J_j\}$ | Conjunto de trabajos |
| $O_{JM} = \{O_{1,1}, O_{1,2}, O_{1,3},\ldots, O_{j,m}\}$ | Conjunto de operaciones |
| $O_{JS} = \{O_{j,1}, O_{j,2}, O_{j,3},\ldots, O_{j,s}\}$ | Secuencia tecnológica. |

Para cada operación $i \in O$ tenemos ligadas un trabajo $j$ $i \in O$ al que pertenece y una máquina $m$ $i \in M$ en la que debe realizarse consumiendo un tiempo $p$ $i \in \mathbb{R}$ ininterrumpido y positivo. Además, es dada una relación de precedencia binaria PREC que descompone *O* en cadenas, una para cada trabajo. Encontrar un tiempo de comienzo ***si*** para cada operación $i \in O$ tal que se minimice el ***makespan***.

El objetivo es programar las operaciones para el procesamiento en las máquinas de tal manera que el tiempo total para completar todos los trabajos, es decir el ***makespan*** sea minimizado. El ***makespan*** de una solución *s* por lo general es denotado por la mayoría de los autores de esta forma $C_{max}(s)$ [Montgomery, 2007]. Por lo tanto la función objetivo se denota como:

Minimizar $C_{max}(s) = \text{Max } i \in O\ (si + pi)$

Dónde:

*s* es la solución.

*C* es el makespan de la solución *s*.

Sujeto a:

$sj \geq si + pi$ para todo $i, j \in O$ con *i* PREC *j* (1)

$sj \geq si + pi$ o $si \geq sj + pj$ para todo $i, j \in O$ con $mi = mj$ (2)

(1) Esta relación refleja el hecho de que un trabajo se compone de operaciones que tienen que efectuarse en un orden preciso y no en otro. Podemos decir entonces, que PREC es una relación de precedencia fijada por el problema entre operaciones que pertenezcan al mismo trabajo, tal que, si *i* PREC *j* entonces *j* no puede comenzar antes que *i* finalice.

(2) Implica que ninguna máquina puede procesar dos operaciones al mismo tiempo [Debels et al., 2006].

## 2.6 Problema multi-objetivo con enfoque mono-objetivo

En el trabajo de Cruz-Reyes, Fernandez y Rangel-Valdez (2017), proponen un método basado en algoritmos de optimización mono-objetivo para resolver un problema multi-objetivo (ver Figura 2). El método explota la prioridad preferente establecida para el problema; este encuentra el mejor valor para el primer objetivo ($E_1$), entonces este valor es usado como límite cuando el segundo objetivo ($E_2$) es minimizado. Por último, el objetivo tres ($E_3$), es minimizado manteniendo sus valores mínimos, que previamente se obtuvieron de $(E_1, E_2)$.

Minimizar $(E_1, E_2, E_3)$
$(\eta,\lambda,\beta,\varepsilon) \in P_{fr}$

Dónde:
$P_{fr}$ es la región factible de los valores de los parámetros, con derecho preferente de $E_1$ sobre $E_2$, y $E_2$ sobre $E_3$.

**Figura. 2**. Problema multi-objetivo

De acuerdo con la Figura 3, se expone el procedimiento general mostrado por este enfoque. Cada algoritmo metaheurísticos resuelve $P$=($DM_{sim}$, $T$), por ejemplo, una instancia del problema (ver Figura 2) formada por los valores de los parámetros que definen el modelo de preferencia para un $DM$ simulado, denotado como $DM_{sim}$, y su conjunto de referencia $T$, intentando de minimizar solamente el objetivo $E_1$. El mejor valor $Best_{E1}$ es usado como límite en los siguientes pasos del método; en este paso el mismo algoritmo de búsqueda minimiza el objetivo $E_2$ mientras valida que las soluciones alcanzadas en el valor de redimiendo $E_1$ es igual o mayor que $Best_{E1}$. Finalmente, en la última etapa, el algoritmo nuevo minimiza el tercer objetivo del problema (Figura 2), por ejemplo $E_3$, mientras mantiene los valores para $E_1$ y $E_2$ menores que $Best_{E1}$ y $Best_{E2}$ respectivamente. La mejor solución ($\eta^*$, $\lambda^*$, $\beta^*$, $\varepsilon^*$) seguirá para los vectores de parámetros obtenidos para este procedimiento.

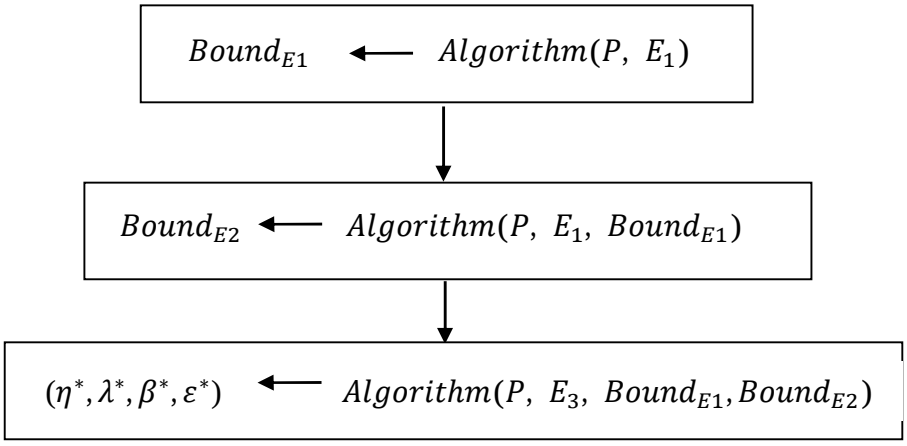



**Figura 3.** Estrategia utilizada para resolver problemas multi-objetivo (Figura 2) usando algoritmos mono-objetivos.

Los algoritmos que se utilizaron en el trabajo de Cruz-Reyes et al. (2017), fueron: a) Algoritmo Genético; Optimización por Enjambre de Partículas; c) Búsqueda Tabú; y d) Recocido Simulado. Estas estrategias tiene dos principales componentes en común: el problema codifica y evalúa la función. La función evaluada en cada etapa del procedimiento general que está relacionada a un objetivo particular $E_i$ del problema (ver Figura 2).

### 2.7 Patrones de diseño

De acuerdo con la definición propuesta por Alexander, menciona que “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, y describe el núcleo de

la solución de ese problema, de tal manera que se puede utilizar esa solución un millón de veces otra vez, sin tener que hacerlo de la misma manera dos veces" [Alexander, 1977]. Por tal motivo, cada patrón de diseño es aquel que mediante sus soluciones propuestas en cada problema, van modelando el desarrollo del software. A si también, una cuidadosa selección de estos patrones, facilitara el mantenimiento futuro del mismo.

### 2.7.1 Definición y clasificación de patrones de diseño

La implementación de los patrones de diseño es la base de muchas soluciones a problemas comunes en el desarrollo de software [Tedeschi, s.f.]. En otras palabras, proporcionan una solución que ha sido probada y documentada a problemas de desarrollo de software que están sujetos a entornos similares. Algunos de los elementos primordiales que se deben tener cuenta de los patrones de diseño, son los siguientes:

- Su nombre.
- El problema (cuando aplicar un patrón).
- La solución (descripción abstracta del problema).
- Las consecuencias (costos y beneficios).

Hoy en día existen varios patrones de diseño popularmente conocidos, los cuales se clasifican como se muestra a continuación [Tedeschi, s.f.]:

a) *Patrones Creacionales:* Inicialización y configuración de objetos.
b) *Patrones Estructurales*: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
c) *Patrones de Comportamiento:* Más que describir objetos o clases, describen la comunicación entre ellos.

En la Tabla 1 se describe brevemente en qué consisten los distintos tipos de patrones de diseño, y cuáles son sus fines.

**Tabla 1.** Patrones de diseño y su descripción

| PATRÓN | DESCRIPCIÓN |
|---|---|
| **CREACIONALES** | |
| Fábrica abstracta (**Abstract Factory**) | Crea una instancia de varias familias de clases. |
| Método de fábrica (**Factory Method**) | Crea una instancia de varias clases derivadas. |

| | |
|---|---|
| Prototipo (**Prototipo**) | Una instancia totalmente inicializado a ser copiada o clonada. |
| Constructor (**Builder**) | Separa la construcción objeto de su representación. |
| Conjunto de objetos (**Object Pool**) | Evita la adquisición costosa y liberación de los recursos mediante el reciclaje de objetos que ya no están en uso. |
| Instancia única(**Singleton**) | Una clase de la que sólo puede existir una sola instancia. |
| **ESTRUCTURALES** | |
| Adaptador (**Adapter**) | Interfaces que coinciden de diferentes clases. |
| Compuesto (**Composite**) | Una estructura de árbol de objetos simples y compuestos. |
| Fachada (**Facade**) | Una única clase que representa todo un subsistema. |
| Datos de clase privada (**PrivateClass Data**) | Restringe de acceso / Acceso mutador. |
| Puente (**Bridge**) | Separa la interfaz de un objeto de su aplicación. |
| Decorador (**Decorator**) | Añade responsabilidades a los objetos de forma dinámica. |
| Peso mosca (**Flyweight**) | Un fino ejemplo de intercambio eficiente. |
| Proxy**(Proxy)** | Un objeto que representa otro objeto. |
| **COMPORTAMIENTO** | |
| Cadena de responsabilidad (**Chain of Responsibility**) | Una manera de pasar una petición entre una cadena de objetos. |
| Intérprete (**Interpreter**) | Una manera de incluir elementos del lenguaje a un programa. |
| Mediador(**Mediator**) | Define la comunicación simplificada entre clases. |
| Objeto nulo(**NullObject**) | Diseñado para actuar como un valor por defecto de un objeto. |
| Estado(**State**) | Modificar el comportamiento de objetos cuando sus cambia se estado. |
| Método Plantilla (**Templatemethod**) | Aplazar los pasos exactos de un algoritmo para una subclase. |
| Comando(**Command**) | Encapsular una solicitud de comando como un objeto. |
| Iterador (**Iterator**) | Acceder Secuencialmente a los elementos de una colección. |
| Recuerdo(**Memento**) | Capturar y restaurar el estado interno de un objeto. |
| Observador(**Observer**) | Una manera de notificar el cambio a un número de clases. |
| Estrategia(**Strategy**) | Encapsula un algoritmo dentro de una clase. |
| Visitante(**Visitor**) | Define una nueva operación para una clase sin cambio. |

## 2.8 Diagrama de clases UML

El **diagrama de clase** explica los tipos de objetos que se encuentran en un sistema y las diversas clases de relaciones estáticas que existen entre ellos [Fowler & Scott, 1999]. De acuerdo con los autores Fowler & Scott, (1999), existen dos principales relaciones estáticas las cuales son:

- Asociaciones. Por ejemplo, un cliente puede rentar dos automóviles.
- Subtipos. Por ejemplo, un arquitecto es un tipo de empleado.

Los diagramas de clase también presentan los atributos y operaciones de una clase y las restricciones que se ven sujetas, descuerdo a la forma en que se relacionan los objetos.

### 2.8.1 Elementos de un diagrama de clases

Un diagrama de clase está compuesto por los siguientes elementos [Salinas, 1996]:

- **Clase:** atributos, métodos y visibilidad.
- **Relaciones:** Herencia, Composición, Agregación, Asociación y Uso.

**Clase:**

Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

En UML, una clase es representada por un rectángulo que posee tres divisiones:

| "Nombre de la clase" |
|---|
| "Atributos" |
| "Métodos" |

**Clase Abstracta:**

Una clase abstracta se expresa con el nombre de la clase y de los métodos con letra "*itálica*". Esto indica que la clase definida no puede ser instanciada, ya que posee métodos abstractos (aún no han sido definidos, es decir, sin implementación). La única forma de utilizarla es definiendo subclases, que implementan los métodos abstractos definidos.

| "clase *Abstracta*" |
|---|
| |
| "*metodos*" |

**Relaciones:**

Una vez definido el concepto de Clase, es necesario explicar cómo se pueden relacionar dos o más clases (cada uno con características y objetivos diferentes).

Antes de empezar, es imprescindible explicar el concepto de cardinalidad de relaciones: En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:

- **uno o muchos**: 1..* (1..n)
- **0 o muchos**: 0..* (0..n)
- **número fijo**: m (m denota el número)

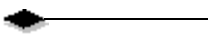
Enseguida se describe de manera general las relaciones que contiene el diagrama de clases:

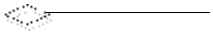
ü Herencia(Especialización/Generalización): 

Indica que una subclase hereda los métodos y atributos especificados por una Super Clase, por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected).

ü Agregación: 

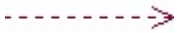
Para modelar objetos complejos, no bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades:

- **Por Valor**: 
  Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada **Composición** (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").

- **Por Referencia**: 
  Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada **Agregación** (el objeto base utiliza al incluido para su funcionamiento).

ü Asociación: 

La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

- Dependencia o Instanciación (uso): 

Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase). Se denota por una flecha punteada.

**2.9 Diseño experimental**

Generalmente en todos los campos de estudios, los investigadores llevan a cabo experimentos para descubrir algo acerca de un proceso o sistema en particular. En otras palabras, un experimento puede hacer referencia a una serie pruebas, en las que se pueden hacer modificaciones de las variables de un sistema o proceso para observar e identificar los motivos de cambios que pudieran observarse en la respuesta de salida [Montgomery, 2004].

En esta parte, los usuarios pueden fácilmente seleccionar las metaheurísticas más adecuadas que resuelvan un determinado problema, de tal manera, que también sea fácilmente configura los parámetros de cada una de las metaheurísticas; con el fin de seleccionar los mejores valores de cada parámetro que maximicen la solución del problema.

# CAPÍTULO 3

## 3. DESCRIPCION DEL PROBLEMA Y ESTADO DEL ARTE

Para poder llevar a cabo el desarrollo del framework de optimización, es necesaria la fácil incorporación de distintos problemas y algoritmos, que de tal manera puedan trabajar en conjunto.

### 3.1 Definición del problema

Dado un conjunto de patrones de diseño *PD*, un conjunto de problemas *P* derivados del área de Apoyo a la Toma de Decisiones, y un conjunto de metaheurísticas *M* de solución para *P*.

¿Es posible desarrollar un Framework de optimización que por medio de *PD* permita implementar *M* fácilmente, y validarlo experimentalmente, en la solución de *P*?

### 3.2 Descripción del problema

Dicho lo anterior, se propone la unificación de estos problemas y algoritmos bajo un esquema de patrones de diseño, permitiendo una buena comunicación entre ellos. Para una mayor profundización de este planteamiento, se expone enseguida un ejemplo que ilustre este concepto.

**Ejemplo:**

De acuerdo con el diagrama de clases que se expone en la Figura 4, se observa que las clases *Algoritmo_1, Algoritmo_2,* hasta el *Algoritmo_n* que heredan de la clase abstracta *Algoritmo*. De tal manera que cada algoritmo que se deseé implementar, derivará de la clase *Algoritmo,* como por ejemplo: el Recocido Simulado, Búsqueda Tabú, Algoritmo Genético, NSGA II, etc. Sin embargo, dada a la naturaleza de que cada algoritmo maneja comportamiento y atributos diferentes; estos suelen compartir métodos semejantes como *inicializar()* y *ejecutar()*. De tal manera, estos métodos se pueden ser extrapolados para todo tipo de algoritmo, haciendo de esta manera las clases más generales. Estas características pueden ser explotadas a favor de la creación de patrones de diseño en diferentes módulos de optimización, de tal manera que se puede reducir el tiempo de desarrollo en nuevas estrategias, y posteriormente su validación.

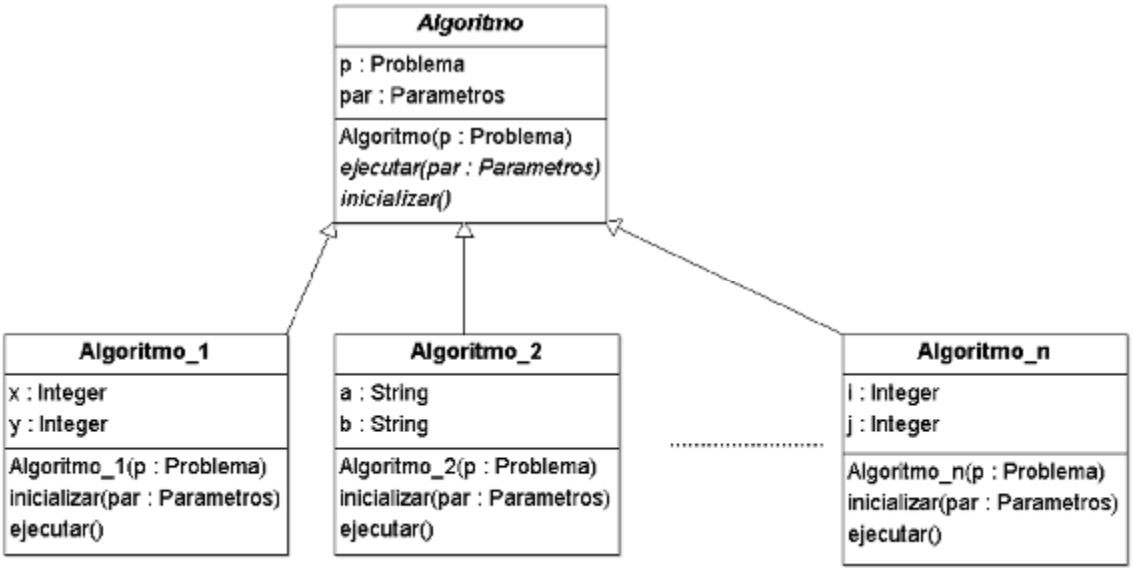



**Figura 4**. Ejemplo de clases de tipo Algoritmo

Como se aprecia en la Figura 4, el método *inicializar()* reciben como parámetro un objeto de la clase *Parametros*, el cual permite configurar a cada algoritmo según sus parámetros necesarios. En la Figura 5 se muestra la clase *Parametros*, la cual permitió estandarizar los datos de entrada en cada algoritmo.

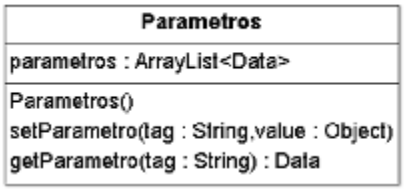



**Figura 5**. Clase parámetros

Por ultimo en la Figura 6, nos encontramos con las clases *Problema_1*, *Problema_2* hasta la clase *Problema_n* heredan de la clase abstracta *Problema*. De la misma forma que las clases de tipo algoritmo, cada problema deberá de derivarse de la clase *Problema,* como por ejemplo: el problema de Cartera de proyectos, el Agente viajero, la Mochila, Calendarización de horarios, etc. Así mimo, dada a la naturaleza de cada problema, estos contienen comportamiento y atributos diferentes; sin embargo, también suelen compartir métodos semejantes como *setParametro()* y *getParametro().* De tal forma que estos métodos pueden ser extrapolados para todo tipo de problema, haciéndolas de esta manera las clases más generales.

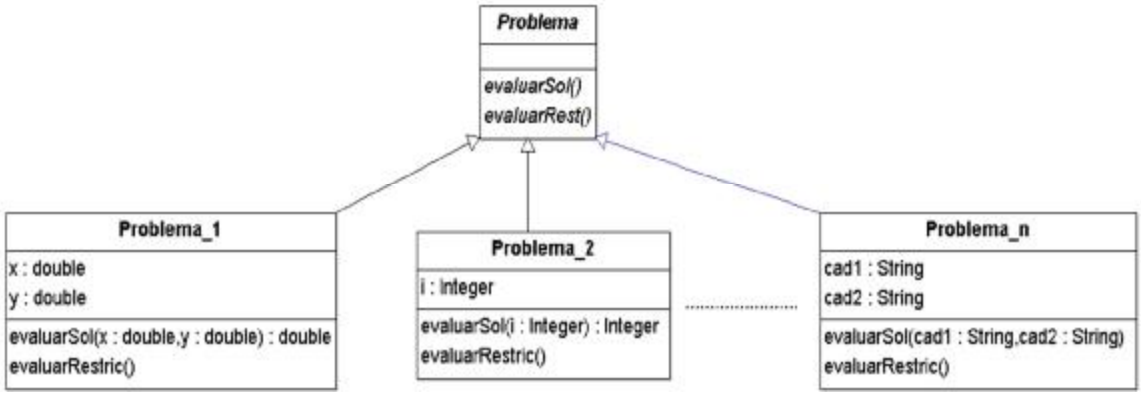



**Figura 6**. Ejemplo de clases de tipo Problema

Todos estos elementos combinados pueden influir en la creación del framework, y en la integración con nuevos algoritmos y/o problemas, que es lo que se busca en la presente investigación.

## 3.3 Trabajos relacionados

En las siguientes subsecciones se exponen los temas más relevantes abordados en esta investigación.

### 3.3.1 Patrones de Diseño

Los trabajos encontrados en la literatura se clasificaron como: Evolución de Patrones de Diseño, Leguaje de Patrones de Diseño y Definición de Patrones de Diseño, según su enfoque; y estas son descritas en las Tablas 2, 3 y 4 respectivamente.

**Tabla 2.**Evolución de Patrones de Diseño

| **Evolución de Patrones Diseño** | | |
|---|---|---|
| **Título** | **Autores** | **Descripción** |
| A Model Transformation Approach for Design Pattern Evolutions | Dong et al., 2006 | En este artículo, se define el proceso de evolución de patrones de diseño en términos de dos niveles de transformación, con lo que son posibles las evoluciones de cada patrón de diseño explícito. Además, se automatizan los procesos de evolución como son las transformaciones XSLT, que puede convertir un modelo UML de una aplicación de patrón de diseño, en un evolucionado modelo UML del patrón. Tanto el UML original y los modelos evolucionados, son representados en *Intercambio de metadatos* de XML (XMI) formato para facilitar las transformaciones. Por otra parte, se verifica la consistencia de los resultados de la evolución utilizando *Java Theorem Prover.* |

| | | |
|---|---|---|
| Software Development Based on Software Pattern Evolution | Kobayashi & Saeki, 1999 | En este trabajo, discuten una técnica para modelar patrones de software para el apoyo de patrones basados en el desarrollo de software. El desarrollo de software puede considerarse como la evolución de artefactos a producir. Los patrones de software son estructuras generales que aparecen con frecuencia en los artefactos; y los patrones también están siendo evolucionados como los artefactos. En este enfoque, consideran a un patrón de software que consiste en la estructura de un patrón (diagramas de clases y/o un diagrama de objetos) y manipulación de operaciones sobre la estructura del patrón. Estas operaciones son para la instanciación del patrón (aplicando un patrón para un problema real) y para la evolución del patrón (evolucionando los artefactos de las etapas anteriores en uno nuevo). |

**Tabla 3.** Lenguaje de Patrones de Diseño

| **Lenguaje de Patrones Diseño** | | |
|---|---|---|
| **Título** | **Autores** | **Descripción** |
| A Visual Languaje for Design Pattern Modelling and Instantiation | Maplesden et al., 2007 | En este trabajo se describe el *Design Pattern Modelling Language* (DPML), una notación de apoyo en la especificación de soluciones de patrones de diseño y cualquier instanciación en los modelos de diseño UML. El DPML utiliza un simple conjunto de abstracciones visuales y fácilmente se presta como herramienta de apoyo. Las especificaciones de solución los patrones de diseño DPML, son usadas para la construcción visual y especificaciones formales de patrones de diseño. Los diagramas de instanciación de DPML, son utilizadas para vincular a una especificación de solución de patrón de diseño, a instancias de un modelo UML, indicando las funciones desempeñadas por los diferentes elementos UML, en la solución de un patrón de diseño genérico. |
| A meta-modeling approach to specifying patterns | Kim, 2004. | El objetivo de esta investigación, es el desarrollo de una práctica y rigurosa técnica de especificación del patrón, que apoya el uso sistemático de los patrones durante el modelado de diseño. En este contexto “Practico” significa que la técnica está basada en un lenguaje de modelado ampliamente conocido. Para lograr el objetivo de un leguaje, el Role-Based Metamodeling Language (RBML), especifica con precisión las soluciones del patrón que fueron desarrollados. El lenguaje tiene una sintaxis basada en el Unified Modeling Language (UML). Una especificación del patrón RBML, define una especialización del metamodelo UML que caracteriza el conjunto de |

| | | |
|---|---|---|
| | | modelos de solución UML para el patrón. Un modelo que se ajusta a la especificación del patrón, satisface las propiedades declaradas en la especificación del patrón. |

**Tabla 4.** Definición de Patrones de Diseño

| **Definiciones de Patrones Diseño** | | |
|---|---|---|
| **Título** | **Autores** | **Descripción** |
| Design Patterns: Elements of Reusable Object-Oriented Software | Gamma et al., 1997 | Este es un libro de patrones de diseño que describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. |
| Applied Java Patterns | Maassen & Stelting, 2001 | Este libro describen los patrones de diseño esenciales, y cómo se pueden utilizar en las aplicaciones en Java. Por otra parte, este libro muestra en dónde y por qué se utilizan los patrones en las API de tecnología en Java. |
| Pattern-Oriented Software Architecture – A System of Patterns | Buschmann et al., 2001 | En este libro habla acerca de patrones en la arquitectura de software. |

### 3.3.2 Frameworks de Optimización

En la Tabla 5 se muestran los Frameworks de optimización que se encontraron en la literatura.

**Tabla 5.** Frameworks de Optimización

| **Desarrollo de un Framework de Optimización** | | |
|---|---|---|
| **Título** | **Autores** | **Descripción** |
| JAMES: A modern object-oriented Java framework for discrete Optimization using local search metaheuristics | Beukelear et al., 2015 | Se menciona un moderno Framework llamado JAMES, hecho en Java orientado a objetos para la optimización discreta. Este utiliza algoritmos de búsqueda local que aprovecha las generalidades de estas Metaheurísticas, separando claramente la aplicación de búsqueda y especificación del problema. Disponen de una amplia gama de búsquedas locales genéricos, incluyendo (estocásticos) como *sube la colina*, *búsqueda tabú*, *búsqueda de vecindad variable* y *Parallel Tempering*. El rendimiento de diferentes algoritmos de búsqueda puede ser evaluada y comparadas a fin de seleccionar una adecuada estrategia de optimización. Además, la influencia de los valores de los parámetros puede ser estudiado. |
| jMetal: A Java framework for | Durillo & Nebro, 2011 | En este trabajo hablan de jMetal, como un Framework basado en Java orientado a objetos para el desarrollo, |

| | | |
|---|---|---|
| multi-objective optimization | | experimentación y estudio de metaheurísticas que resuelven problemas de optimización multi-objetivo. Este Framework incluye la generación automática de información estadística de los resultados obtenidos y aprovechando la disponibilidad actual de procesadores multi-core para acelerar el tiempo de ejecución de los experimentos. |
| Redesigning the jMetal Multi-Objective Optimization Framework | Nebro A. et al., 2015 | JMetal es un entorno de código abierto basado en Java para Optimización multi- objetivo con metaheurísticas. Su arquitectura tiene un nuevo diseño basado en patrones de diseño, y un mejor uso de las características del lenguaje Java. Entre las nuevas características incorporadas, jMetal soporta en viva interacción con algoritmos de ejecución y ejecución paralela de los algoritmos. |
| JCLEC: a Java framework for evolutionary computation. | Ventura et al., 2008 | En este artículo describen a JCLEC, como un sistema de software en Java para el desarrollo de aplicaciones de computación evolutiva. Este sistema ha sido diseñado como un Framework, aplicando patrones de diseño para maximizar su reutilización y adaptación para nuevos paradigmas con un mínimo esfuerzo en programación. La arquitectura JCLEC comprende tres principales módulos:<br>**a)** el núcleo contiene todas las definiciones de tipo abstracto y su aplicación;<br>**b)** ejecución de experimentos en un entorno scripting para ejecutar algoritmos en modo por lotes;<br><br>Gen-Lab una interfaz gráfica de usuario que permite a los usuarios configurar un algoritmo y ejecutar en forma interactiva para visualizar los resultados obtenidos. |
| ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics∗ | Cahon et al., 2004 | En su trabajo presentan la caja blanca de ParadisEO, un Framework orientado a objetos dedicado al diseño reutilizable de metaheurísticas paralelas y distribuidas (PDM). ParadisEO proporciona un amplia gama de características que incluyen algoritmos evolutivos (EA), búsquedas locales (LS), los más comunes modelos paralelos y distribuidos y mecanismos de hibridación, etc. |
| The EvA2 optimization framework. | Kronfeld et al., 2010 | Presentan a EvA2, como un completo Framework de optimización metaheurística con énfasis en Algoritmos Evolutivos. Esta presenta una estructura modular de interfaces y clases abstractas para la implementación tanto problemas de optimización como solvers. Esta se |

| | | |
|---|---|---|
| | | encuentra libremente disponible bajo una licencia de código abierto (LGPL). |

### 3.4 Análisis de trabajos relacionados

En la literatura científica se encontraron algunos Frameworks de optimización que presentan una relevancia para el estudio propuesto en éste trabajo. Uno de estos trabajos corresponde al Framework JAMES presentado en [Beukelaer et al., 2015], este framework está hecho en Java, y se caracteriza por separar la búsqueda de la solución del problema. Otro trabajo relacionado es el Framework jMetal (ver [Nebro et al., 2015]), framework dirigido al desarrollo, experimentación, y estudio de metaheurísticas para la resolución de problemas de optimización multi-objetivo, el cuál sirvió de base para la propuesta de este trabajo.

La Tabla 6 compara los Frameworks revisados en la literatura en función las características deseables de este trabajo. La principal diferencia encontrada estriba en la incorporación de los módulos de optimización de preferencias y diseño experimental, los cuales no han sido considerados en las demás arquitecturas.

**Tabla 6.** Comparación de Frameworks

| **Características de los Frameworks** | | | |
|---|---|---|---|
| **Características** | **JAMES v0.2** | **jMetal v5.0** | **Presente trabajo** |
| Optimización Mono-objetivo | ✔ | ✔ | ✔ |
| Optimización Multi-objetivo | ✔ | ✔ | ✔ |
| Preferencias | ✘ | ✘ | ✔ |
| Diseño experimental | ✘ | ✘ | ✔ |
| Patrones de diseño | ✘ | ✔ | ✔ |
| Algoritmos paralelos | ✔ | ✔ | ✘ |

Finalmente, al revisar las características de estos Frameworks, se observó que JAMES no emplea explícitamente de patrones de diseño, mientras que jMetal sí; por lo menos en optimización. Además, ninguno de esto Frameworks están enfocados al uso extensivo de patrones de diseño en procesos como diseño experimental y/o manejo de preferencias, lo cual se puede considerar como parte de la contribución de este trabajo.

# CAPÍTULO 4

## 4. PROPUESTA DE SOLUCIÓN

La presente propuesta hace mención al uso de patrones de diseño para la construcción del Framework; los cuales se utilizaron como base en la incorporación de diversos módulos de optimización, como por ejemplo: Diseño Experimental, Preferencias, Metaheurísticas, etc. En la Figura 7, se expone un diagrama de actividades que representa la metodología utilizada para seleccionar los patrones de diseño más idóneos, en la construcción de la arquitectura del framework.

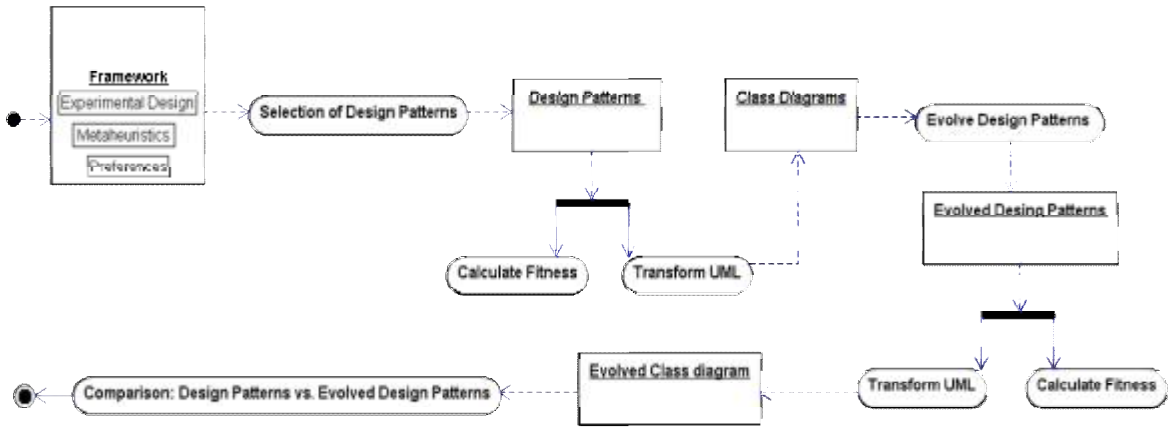



**Figura 7.** Metodología para seleccionar los patrones de diseño

En base a los patrones de diseño seleccionados, permitirán dar una mayor flexibilidad en la creación y modificación de las clases pertenecientes a los módulos de optimización. Esta metodología propuesta se describe brevemente en la Tabla 7.

**Tabla 7.** Descripción de la metodología

| **Proceso Evolutivo** | |
|---|---|
| **Etapa** | **Descripción** |
| **1** | Se selecciona los patrones de diseño que permitan manipular los módulos de optimización del Framework, calculando posteriormente su *fitness* (beneficio). |
| **2** | Se procede a transformar los patrones de diseño a diagramas de clases UML. |
| **3** | Mediante un algoritmo genético, se procede a evolucionar los patrones de diseño elegidos en la etapa 1; calculando nuevamente su *fitness.* |
| **4** | Se procede transformar los patrones de diseño evolucionados a diagramas de clases UML. Dando como resultado un diagrama de clases evolucionado. |
| **5** | Por último, se compararan los patrones de diseño de la primera etapa contra los de la tercera etapa. Esto con el fin de observar el impacto que existe en utilizar unos u otros patrones de diseño en la arquitectura del Framework. |

Enseguida se detallan las etapas utilizadas en la metodología descrita de la Tabla 7:

- **Etapa 1.-** Las Figuras 8 y 9 muestran algunos ejemplos de los patrones de diseño seleccionados, los cuales se marcan con un círculo azul las clases que hacen referencia a dichos patrones.

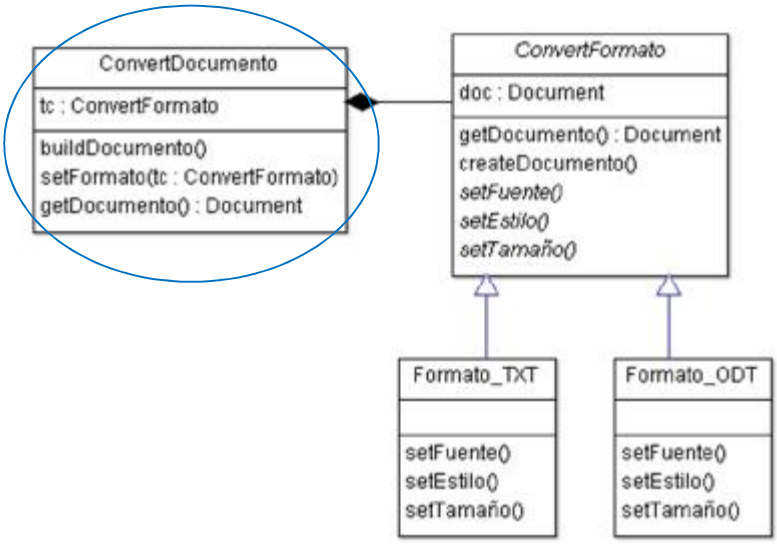



**Figura 8.** Ejemplo del patrón de diseño Builder

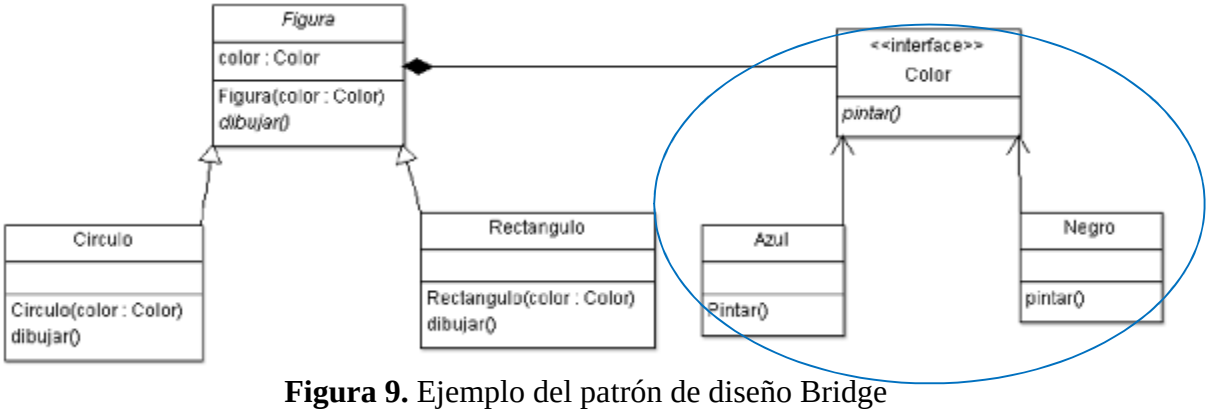



**Figura 9.** Ejemplo del patrón de diseño Bridge

- **Etapa 2.-** Se procede a transformar los patrones seleccionados a diagramas de clases UML. Las Figuras 9 y 10 se presentan dos posibles maneras diseñar la arquitectura del Framework, en donde cada una de sus clases fueron identificadas con el número del paquete al que pertenecen.

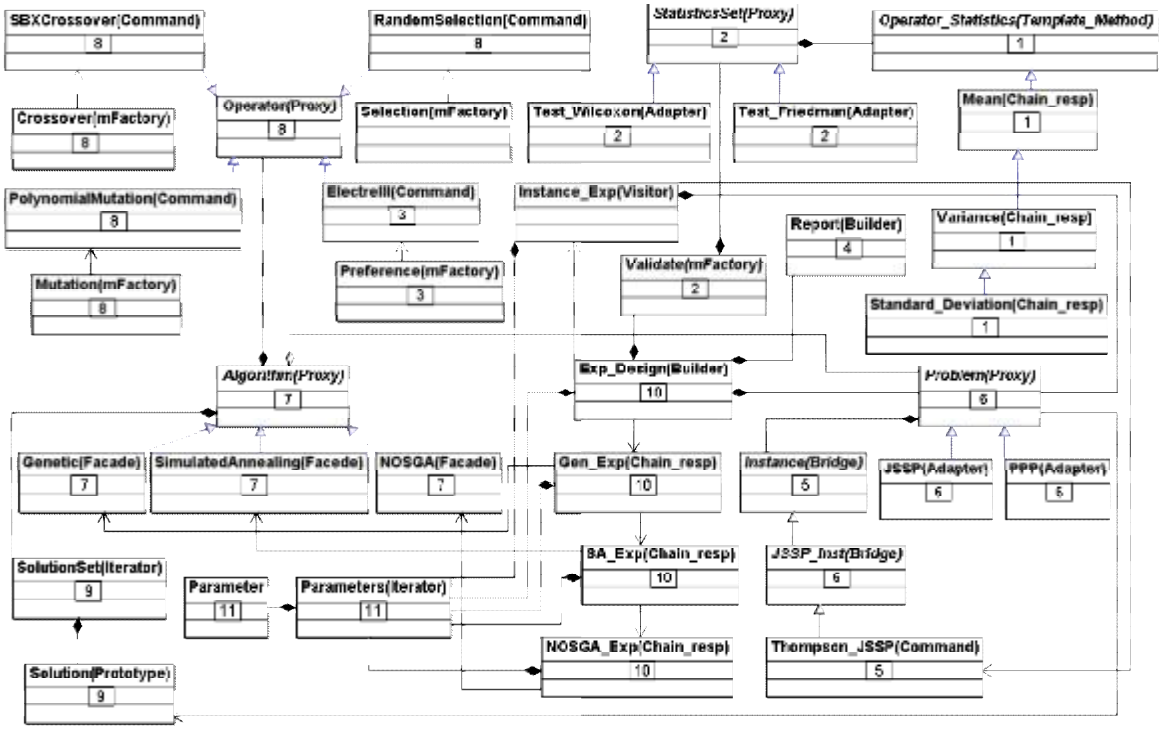



**Figura 9.** Primera arquitectura del Framework de optimización.

Los patrones seleccionados fueron elegidos estratégicamente para maximizar la reutilización y extensibilidad del Framework. Por ejemplo, gracias a la implementación del patrón de diseño **proxy** en la clase *Problem*, los algoritmos podrán acceder a sus clases heredadas, sin que estas sean declaradas en los algoritmos. De la misma forma, las demás clases del Framework son diseñadas con patrones de diseño, pudiendo impactar en gran medida en algunos criterios de modularidad, como es la *cohesión* y *acoplamiento*.

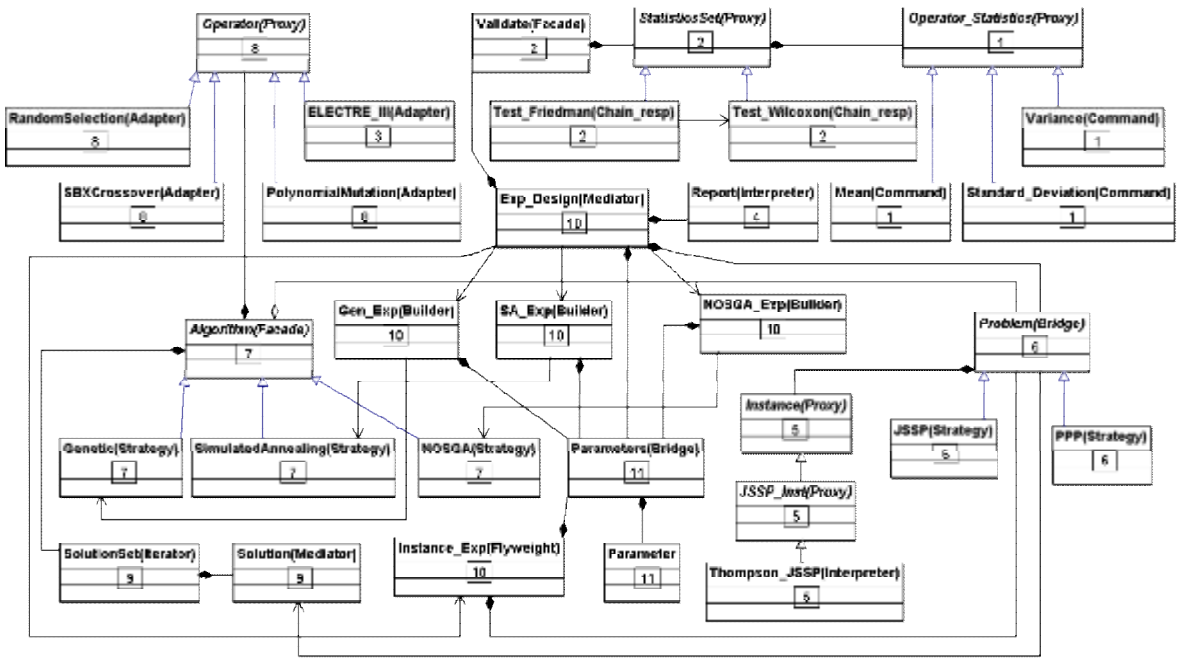



**Figura 10.** Segunda arquitectura del Framework de optimización.

- **Etapas 3, 4 y 5.-** Estas etapas se describen en los capítulos posteriores. La etapa 3 se describe en el capítulo 5, y las últimas dos etapas, en el capítulo 6.

La Tabla 8 describe la funcionalidad que tiene cada uno de los módulos pertenecientes al Framework de optimización; nombrando las clases más sobresalientes de cada uno.

**Tabla 8.** Módulos de optimización del Framework

| **Características de los módulos de optimización** | |
|---|---|
| **Módulo** | **Descripción** |
| Validación<br>**1** | Mediante el uso de pruebas estadísticas como Friedman y Wilcoxon, permitirá seleccionar las mejores soluciones creadas por los algoritmos, en función de cada problema. Además, mediante la generalización algunos operadores estadísticos como la media, varianza y desviación estándar, facilitara la creación de nuevas pruebas estadísticas. Este módulo esta compuesto por las clases: *Validate*, *StatisticsSet*, *Test_Friedman*, *Test_Wilcoxon*, *Operator_Statistics*, *Mean*, *Variance* y *Standard_Desviation*. |
| Preferencia<br>**2** | Este módulo aborda un sistema relacional de preferencias compuesto de varias relaciones binarias [Roy, 1996] como: Indiferencia, Preferencia estricta, Preferencia débil, Incomparabilidad, K-preferencia y No preferencia. Este módulo ayudará a implementar diferentes técnicas pertenecientes a MCDA; el cual se compone por la clases: *Preference* y *ELECTRE III*. |
| Reporte<br>**3** | Este módulo presentara los resultados obtenidos del módulo “Validación”, de tal manera que puedan ser revisados y comparados para su posterior |

| | |
|---|---|
| | análisis. Este módulo está compuesto por la clase: *Report*. |
| Instancia<br><br>**4** | Este módulo está diseñado para contener la instancia de cualquier problema, de tal manera al generalizar su información, pueda ser abordado por el módulo "Problema"; sin importar el formato que utilicen los autores de cada instancia. Actualmente este módulo está compuesto por las clases: *Instance*, *JSSP_Inst* y *Thompson_JSSP*. |
| Problema<br><br>**5** | Este módulo permite generalizar distintos problemas de optimización, enfatizando sus características comunes. Actualmente este módulo está compuesto por las clases: *Problem*, *JSSP* y *PPP*. |
| Algoritmo<br><br>**6** | Este módulo permite manipular la información generada por módulo "Problema", según el comportamiento de cada algoritmo. Este permite generalizar distintos algoritmos, enfatizando sus características comunes, facilitando la creación y extensión de sus clases. Este módulo está compuesto por las clases: Algorithm, *Genetic*, *SimulatedAnnealing* y *NOSGA*. |
| Operadores<br><br>**7** | Este módulo contiene un conjunto de operadores en el área de optimización, como p. e. Cruza, Mutación, Selección, etc. Estos son necesarios para llevar a cabo la ejecución de múltiples algoritmos como p. e. el Genético, Recosido Simulado y NOSGA (todos estos dentro del módulo "Algoritmo"). Este módulo está compuesto por las clases: *Operator*, *Crossover*, *Mutation*, *Selection*, etc. |
| Solución<br><br>**8** | Este módulo permite generalizar las soluciones creadas por el módulo "Algoritmo"; que de otra manera, se tendría que crear tantas clases de soluciones como existan algoritmos en el Framework. Este módulo está compuesto por las clases: *Solution* y *SolutionSet*. |
| Diseño Experimental<br><br>**9** | Este módulo es uno de los más importantes de este Framework, ya que permite realizar una experimentación sencilla y cómoda. Además, también permite configurar fácilmente los parámetros pertenecientes a módulos de optimización (p. e. "Problema", "Algoritmo", etc.). De esta manera evitemos la configuración directa a esos módulos. Actualmente este módulo está compuesto por las clases: *Parameter*, *Parameters*, *Exp_Design*, *Gen_Exp*, *SA_Exp* y *NOSGA_Exp* e *Instance_Exp*. |

En la Tabla 9 se presenta dos posibles maneras de diseñar la arquitectura del Framework, mediante el uso de patrones de diseño.

**Tabla 9.** Caracterización del Framework de optimización.

| Framework | | | | | |
|---|---|---|---|---|---|
| **Module** | | **Class** | | **Design patterns** | |
| **ID** | **Name** | **Package** | **Name** | **Option 1** | **Option 2** |
| **1** | **Validación** | 1 | *Operator_Statistic* | Template method | Proxy |
| | | | Mean | Chain responsibility | Command |
| | | | Variance | Chain responsibility | Command |
| | | | Standard_Desviation | Chain responsibility | Command |
| | | 2 | Validate | Factory method | Facade |
| | | | *StatisticSet* | Proxy | Proxy |
| | | | Test_Fridman | Adapter | Chain responsibility |
| | | | Test_Wilcoxon | Adapter | Chain responsibility |
| **2** | **Preferencia** | 3 | Preference | Factory method | |
| | | | ELECTRE III | Command | Adapter |
| **3** | **Reporte** | 4 | Report | Builder | Interpreter |
| **4** | **Instancia** | 5 | *Instance* | Bridge | Proxy |
| | | | JSSP_Inst | Bridge | Proxy |
| | | | Thompson_JSSP | Command | Interpreter |
| **5** | **Problema** | 6 | *Problem* | Proxy | Bridge |
| | | | JSSP | Adapter | Strategy |
| | | | PPP | Adapter | Strategy |
| **6** | **Algoritmo** | 7 | *Algorithm* | Proxy | Facade |
| | | | Genetic | Facade | Strategy |
| | | | SimulatedAnnealing | Facade | Strategy |
| | | | Nosga | Facade | Strategy |
| **7** | **Operadores** | 8 | *Operator* | Proxy | Proxy |
| | | | Crossover | Factory method | |
| | | | Mutation | Factory method | |
| | | | Selection | Factory method | |
| | | | SBXCrossover | Command | Adapter |
| | | | PolynomialMutation | Command | Adapter |
| | | | RandomSelection | Command | Adapter |
| **8** | **Solución** | 9 | Solution | Prototype | Madiator |
| | | | SolutionSet | Iterator | Iterator |
| **9** | **Diseño Experimental** | 10 | Exp_Desing | Builder | Madiator |
| | | | Gen_Exp | Chain responsibility | Builder |
| | | | SA_Exp | Chain responsibility | Builder |
| | | | NOSGA_Exp | Chain responsibility | Builder |
| | | | Instance_Exp | Visitor | Flyweight |
| | | 11 | Parameters | Iterator | Bridge |
| | | | Parameter | | |

# CAPÍTULO 5

## 5. PROPUESTA DE SOLUCIÓN

En este capítulo se aborda el proceso evolutivo de los patrones de diseño, el cual permitió encontrar los patrones más adecuados en la construcción del framework propuesto. Para llevar a cabo este proceso, se eligió el algoritmo genético como medio en la generación de soluciones, las cuales minimizaran el acoplamiento entre paquetes. En las siguientes subsecciones se describen los elementos comprendidos de dicho algoritmo.

**Nota:** Todos los valores expuestos en este capítulo con excepción del 5.1, son utilizados solamente confines representativos.

### 5.1 Construcción de la solución

Para llevar a cabo la construcción de la solución, se seleccionaron aleatoriamente para cada paquete, uno de las dos opciones que se muestra en la columna *Design patterns* de la Tabla 9. Cada paquete representará una posición de la solución, tal como se aprecia en la Figura 11; y en conjunto representara un individuo de la población. Este individuo (o solución), será representado mediante un número binario; donde el *cero* representara la opción 1 y el *uno* representara la opción 2.

| Modulo | 1 | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Opción | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 2 |
| Solución | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |

**Figura 11.** Configuración de la solución.

### 5.2 Función de aptitud

Para llevar a cabo la evaluación de la solución, se implementó la métrica llama *Distancia de la Secuencia Principal (D)*, la cual mide el nivel acoplamiento que existe entre paquetes. Esta métrica fue introducida por Robert C. Martin (2000), la cual se define entre los rangos [0, 0.707] y puede ser calculada por la ecuación 1.

$$D = \frac{|A+I-1|}{\sqrt{2}} \qquad 1)$$

Dónde:

***I*** es la métrica de *Inestabilidad*, que se define entre los rangos [0,1] y es calculada por la ecuación 2:

$$I = \frac{Ae}{Aa + Ae} \qquad 2)$$

Dónde:

- *Aa* es el *Acoplamiento Aferente.* Es el número de clases que se encuentran fuera del paquete y que dependen de las clases que se encuentran dentro del paquete (es decir, dependencias entrantes).
- *Ae* es el *acoplamiento Eferente.* Es el número de clases que se encuentran fuera del paquete y que las clases que se encuentran dentro, son dependientes de estas (es decir, dependencias salientes).

**Nota:** Si no existen dependencias salientes, entonces será cero y el paquete será estable. Si no existen dependencias entrantes, entonces será uno y el paquete será inestable [Martin, 2000].

***A*** es la métrica de *Abstracción,* que se define entre los rangos [0,1] y es calculada por la ecuación 3:

$$A = \frac{Na}{Nc} \qquad 3)$$

Dónde:

- *Nc* es el número de clases en el paquete.
- *Na* es el número de clases abstractas en el paquete.

Para una mejor comprensión en la implementación de esta métrica, se presenta un Ejemplo de una pequeña representación de dicha implementación.

**Ejemplo:**

**Tabla 10.** Datos requeridos en la métrica.

| **Paquete** | **Total de Clases** | **Total de Clases Abstractas** | **Acoplamiento Aferente** | **Acoplamiento Eferente** |
|---|---|---|---|---|
| 6 | 6 | 1 |   |   |
| 9 | 4 | 0 | | |

Una vez extrayendo los valores de la instancia Aplicando los valores que se observa en la Tabla 10, se procede a implementar la métrica *Distancia de la secuencia principal (D)*; calculando el nivel de acoplamiento que tiene el paquete 9 con respecto al paquete 6 (ver Tabla 11).

**Tabla 11.** Implementación de la métrica.

| **Métricas** | | |
|---|---|---|
| **Inestabilidad** | **Abstracción** | **Distancia de la secuencia principal** |
| $I = \frac{1}{6 + 1} = 0.143$ | $A = \frac{0}{4} = 0$ | $D = \frac{\lvert 0 + 0.143 - 1 \rvert}{\sqrt{2}} = 0.605$ |

De la misma forma se procede a calcular todos los demás paquetes, obteniendo como resultado la suma de todas sus evaluaciones.

### 5.3 Selección

El proceso de selección nos permitirá elegir las soluciones padres de la población, tomando la mitad de la población. En este proceso se implementó el método de la *ruleta* [DeJong, 1975], la cual se explica a continuación:

**a)** Antes de seleccionar los padres, primero se calcula el *fitness* de la población como se muestra en la Figura 12, empleando la métrica de la sección 5.2.

| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | *fitness(f)* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sol 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 10 |
| Sol 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 15 |
| Sol 3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 20 |
| Sol 4 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 5 |
| Sol 5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 30 |
| Sol 6 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 25 |
| Sol 7 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 30 |
| Sol 8 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 40 |

**Figura 12.** Población de soluciones.

**b)** Después se suma el *fitness* de cada solución como se muestra en la ecuación 1

$$\sum_{i=1}^{8} f_i = 10 + 15 + 20 + 5 + 30 + 25 + 30 + 40 = 175 \quad (1)$$

**c)** Calculamos $\bar{f}$ :

$$\bar{f} = \frac{175}{8} = 21.875 \tag{2}$$

**d)** Calculamos $T$:

$$T = \sum_{i=1}^{8} Ve_i = \sum_{i=1}^{8} \frac{f_i}{\bar{f}} = 0.46 + 0.68 + 0.91 + 0.22 + 1.37 + 1.14 + 1.37 + 1.83 = 7.98 \tag{3}$$

**e)** Generamos un número aleatorio($r$) entre [0.0, $T$]:

$r = 2.03$

**f)** Se busca la posición del primer sumando de $Ve_i$ que sea mayor a $r$; por ejemplo, si $r < \sum_{i=3}^{8} Ve_3$, entonces seleccionamos la solución ubicada en esa posición y se considerará como una solución padre. Este ejemplo se puede aprecia en la Figura 13.

| | **Solución** | $\sum_{i=1}^{8} Ve_i$ |
|---|---|---|
| | 1 | 0.46 |
| | 2 | 1.14 |
| → | 3 | 2.05 |
| | : | : |
| | 8 | 7.98 |

**Figura 13.** Selección de padres.

De esta manera se seguirán seleccionando los padres, hasta a completar la mitad de la población. En la Figura 14 se muestra un ejemplo de los padres seleccionados de la población.

| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Solución 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Solución 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| Solución 3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Solución 4 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**Figura 14.** Soluciones Padres.

### 5.4 Cruza

El proceso de cruza consiste en combinar un conjunto de padres y generar el mismo número de hijos. Este proceso se llevó a cabo utilizando el método de *cruza de un punto* [Holland, 1975], el cual se explica a continuación:

**a)** Primero se establece una probabilidad de cruza, por ejemplo 0.8. Esto nos indicará que padre es apto para la cruza.

**b)** Después se genera un número aleatorio entre [0,1] y si este es $\leq 0.8$, entonces se selecciona el padre (ver Figura 15); de lo contrario se descarta y se procede a evaluar al siguiente. En la Figura 10 se presenta un ejemplo de los padres seleccionados, tomando en consideración las soluciones padres de la Figura 14.

| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Solución 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Solución 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

**Figura 15.** Padres seleccionados.

**c)** En esta parte se realizará la *cruza de un punto*. Este método consiste en generar un número aleatorio entre [1, 11], el cual nos indicara hasta que elemento estarán compuestos los hijos. Por ejemplo, como se muestra en la Figura 16, se generó el número aleatorio 8.

| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Solución 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Solución 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Hijo 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Hijo 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

**Figura 16.** Cruza de un punto.

### 5.5 Muta

La mutación se realiza haciendo modificaciones en las soluciones hijos. Estos hijos tendrán cambios en su estructura, es decir; si uno de sus elementos es cero, se remplazara por uno, y viceversa. Sin embargo, al igual que la cruza, es necesario establecer previamente un rango de probabilidad, el cual nos indicara que hijo será mutado.

Empleando los hijos generados de la Figura 16, se implementará el método de *mutación por intercambio recíproco*. Este método consiste en seleccionar aleatoriamente dos elementos de solución hijo e intercambiarlos de posición (ver Figura 17).

| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Hijo 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Hijo 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Hijo 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| Hijo 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

**Figura 17.** Mutación por intercambio recíproco.

# CAPÍTULO 6

## 6. EXPERIMENTACIÓN Y RESULTADOS

Antes de comenzar, es importante mencionar que las experimentaciones se realizaron mediante un diseño experimental, el cual fue implementado en la primera arquitectura del framework, que se expone en la Figura 9. Gracias a esta implementación se logró configurar fácilmente el algoritmo genético, estableciendo como parámetro la clase problema: *Framework_Design* (ver Figura 18). Esta clase permite crear y evaluar las soluciones compuestas de patrones de diseño, que posteriormente servirán en la construcción del framework. Así mismo, esta clase recibe como parámetro la clase instancia: Framework_Inst (ver Figura 19), la cual recopila la información de las arquitecturas del framework (ver Figuras 9 y 10), las cuales son contenidas en la instancias que se describen en el Apéndice A.

### 6.1 Ambiente experimental

La siguiente configuración corresponde a las condiciones bajo las que se desarrollaron las pruebas de este trabajo:

1. **Hardware:** Procesador Intel Core (TM) i3-5015U CPU 2.10 GHZ y 6 GB RAM.

2. **Software:** Sistema operativo: Windows 10; Lenguaje de programación: Java versión 1.8.0_92; IDE: NetBeans 8.1.

3. **Instancias:** La instancia utilizada en esta experimentación se muestra en la Figura 18 Apéndice A. Esta instancia recopila información de las dos que arquitecturas propuestas de las Figuras 9 y 10.

4. **Medida de rendimiento:** El rendimiento fue medido a través de la métrica *Distancia de la Secuencia Principal (D),* la cual fue implementada dentro de la clase *Framework_Design.* Esta clase fue incorporada dentro del módulo Problema, heredando los atributos y métodos de la clase ***Problem*** (ver Figuras 9 y 10) como las demás clases. Sin embargo, a diferencias de las otras clases, está no cuenta con restricciones, pero sí con su función objetivo. Esta función objetivo minimiza las soluciones creadas por el algoritmo genético, permitiendo seleccionar las soluciones que mayor minimicen el acoplamiento entre paquetes.

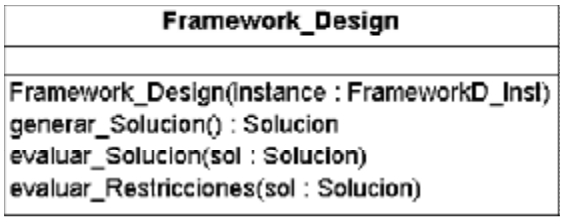



**Figura 18.** Representación de la clase *Framework_Desing*

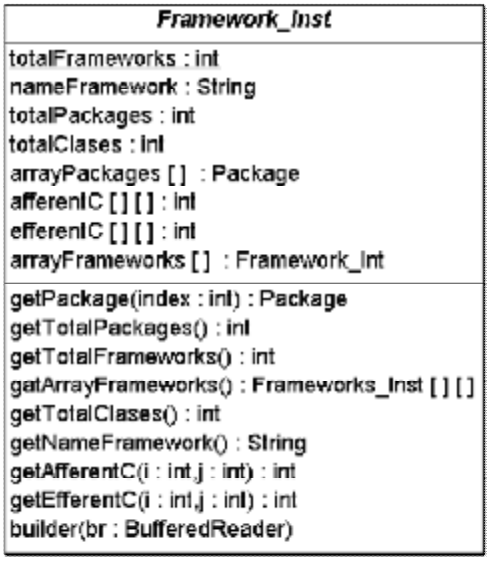



**Figura 19.** Representación de la clase *Framework_Inst*

### 6.2 Experimentación

Como se mencionó al principio de este capítulo, se implementó un diseño experimental que permitirá llevar a cabo las experimentaciones de manera sencilla. Por tal motivo, en la Figura 20 se representa la clase implementada del diseño experimental, llamada *Experimental_Design*, y sus configuraciones son expuestas en las Tablas 12 y 13.

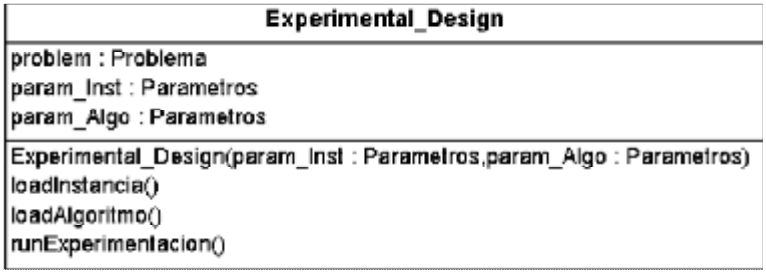



**Figura 20.** Representación de la clase *Experimental_Design*

**Tabla 12.** Parámetros del algoritmo genético.

| Parámetro | Valor |
|---|---|
| Número de Experimentaciones | 50 |
| Número de Generaciones | 100 |
| Tamaño de la población | 100 |

**Tabla 13.** Parámetros de los operadores.

| Operador | Método | Probabilidad |
|---|---|---|
| Selección | Ruleta | |
| Cruza | Cruza de un punto | 0.7 |
| Muta | Mutación por intercambio recíproco | 0.4 |

A demás se implementaron otras clases que ayudaron a configurar el algoritmo genético, como por ejemplo; la implementación de una clase de tipo Problema que permitió evaluar las arquitecturas del Frameworks, y otra de tipo Instancia que permitió agrupar la información de las arquitecturas. Estas arquitecturas se evaluaron mediante la métrica *Distancia de la Secuencia Principal (D)*; las cuales se presentan en la Tabla 14.

**Tabla 14.** Evaluación de las dos Arquitecturas.

| Framework | Solución | *fitness* |
|---|---|---|
| 1 | 0 0 0 0 0 0 0 0 0 0 0 0 | 55.7513 |
| 2 | 1 1 1 1 1 1 1 1 1 1 1 1 | 54.8716 |

Ahora se procede a llevar a cabo las 50 experimentaciones del Algoritmo Genético, obteniendo los resultados de las experimentaciones 1, 15, 30 y 50, que se presentan en las Tablas 15, 16, 17 y 18 respectivamente.

**Tabla 15.** Resultados de la Experimentación 1.

| Individuo | Solución | *fitness* |
|---|---|---|
| 1 | 0 0 1 1 0 0 0 1 1 0 1 | 53.6877 |
| 2 | 1 0 1 1 0 0 0 1 1 0 0 | 53.6877 |
| 3 | 1 0 0 1 1 1 0 1 1 0 0 | 54.0412 |
| 4 | 1 1 1 0 0 1 0 1 1 0 0 | 54.1002 |
| : | : | : |
| 49 | 0 1 0 0 1 0 0 1 0 0 0 | 55.3376 |
| 50 | 0 1 0 1 1 1 1 1 0 0 1 | 55.3376 |
| 51 | 1 1 1 0 1 1 1 1 0 0 0 | 55.3376 |
| 52 | 1 1 0 1 1 0 0 0 1 1 0 | 55.3442 |
| : | : | : |
| 97 | 0 1 1 0 1 1 1 0 0 1 0 | 56.5817 |
| 98 | 1 1 1 0 0 1 1 0 0 1 0 | 56.5817 |

| | | |
|---|---|---|
| 99 | 0 0 0 0 1 1 1 0 0 1 0 | 56.8763 |
| 100 | 0 0 0 0 1 0 1 0 0 1 0 | 56.8763 |

**Tabla 16.** Resultados de la Experimentación 15.

| **Individuo** | **Solución** | ***fitness*** |
|---|---|---|
| 1 | 1 1 1 1 0 1 0 1 1 0 1 | 53.7466 |
| 2 | 1 0 1 0 1 1 0 1 1 0 0 | 54.0412 |
| 3 | 0 0 0 1 0 1 0 1 1 0 1 | 54.0412 |
| 4 | 1 1 0 1 1 0 0 1 1 0 0 | 54.1002 |
| : | : | : |
| 49 | 1 0 0 1 0 0 0 0 1 1 1 | 55.2853 |
| 50 | 1 0 1 0 0 1 0 0 1 1 0 | 55.2853 |
| 51 | 0 1 0 0 0 0 0 1 0 0 1 | 55.3376 |
| 52 | 1 0 1 1 0 0 0 1 0 1 1 | 55.3430 |
| **:** | **:** | **:** |
| 97 | 0 0 0 0 0 0 1 0 0 1 0 | 56.8763 |
| 98 | 1 0 0 0 1 1 1 0 0 1 1 | 56.8763 |
| 99 | 1 1 0 0 1 1 1 0 0 1 1 | 56.9352 |
| 100 | 0 1 0 0 1 0 1 0 0 1 1 | 56.9352 |

**Tabla 17.** Resultados de la Experimentación 30.

| **Individuo** | **Solución** | ***fitness*** |
|---|---|---|
| 1 | 1 0 1 1 0 0 0 1 1 0 0 | 53.6877 |
| 2 | 1 1 1 1 0 0 0 1 1 0 0 | 53.7466 |
| 3 | 0 0 0 1 0 0 0 1 1 0 0 | 54.0412 |
| 4 | 1 1 0 1 1 1 0 1 1 0 0 | 54.1002 |
| : | : | : |
| 49 | 1 0 0 1 1 1 0 0 1 1 1 | 55.2853 |
| 50 | 1 0 1 0 1 1 0 0 1 1 0 | 55.2853 |
| 51 | 0 1 0 1 0 0 1 1 0 0 1 | 55.3376 |
| 52 | 0 1 0 1 1 1 1 1 0 0 1 | 55.3376 |
| : | **:** | **:** |
| 97 | 0 1 0 1 0 0 1 0 0 1 1 | 56.5817 |
| 98 | 0 1 0 1 1 0 1 0 0 1 1 | 56.5817 |
| 99 | 0 1 0 1 0 1 1 0 0 1 0 | 56.5817 |
| 100 | 0 1 0 0 1 0 0 0 0 1 0 | 56.5817 |

**Tabla 18.** Resultados de la Experimentación 50.

| **Individuo** | **Solución** | ***fitness*** |
|---|---|---|
| 1 | 0 1 1 1 0 1 0 1 1 0 0 | 53.7466 |
| 2 | 0 0 0 1 1 1 0 1 1 0 1 | 54.0412 |
| 3 | 0 1 1 1 1 1 1 1 1 0 1 | 54.1002 |

| | | |
|---|---|---|
| 4 | 1 1 1 1 0 0 1 1 1 0 0 | 54.1002 |
| : | : | : |
| 49 | 0 0 1 1 1 1 1 0 1 1 0 | 55.2853 |
| 50 | 1 1 0 0 1 0 0 1 0 0 0 | 55.3376 |
| 51 | 1 0 1 1 0 1 1 0 0 0 0 | 55.3978 |
| 52 | 1 0 0 1 1 0 0 0 0 0 1 | 55.3978 |
| : | : | : |
| 97 | 0 1 1 0 1 0 1 0 0 1 0 | 56.5817 |
| 98 | 1 1 1 0 0 0 1 0 0 1 0 | 56.5817 |
| 99 | 0 0 0 0 0 0 1 0 0 1 1 | 56.8763 |
| 100 | 1 1 0 0 0 1 1 0 0 1 0 | 56.9352 |

Los valores de las experimentaciones se ordenaron de menor a mayor para una mejor apreciación de las evaluaciones. Por otro lado, se pudo observar que los valores resultantes de estas experimentaciones, pueden cambiar de acuerdo a los patrones de diseño que se utilicen en cada evaluación; ya que estos pueden alteran el número de clases y el comportamiento de las mismas, impactando en gran medida en algunos criterios de modularidad, como la *cohesión y acoplamiento* [Joyanes, 1996].

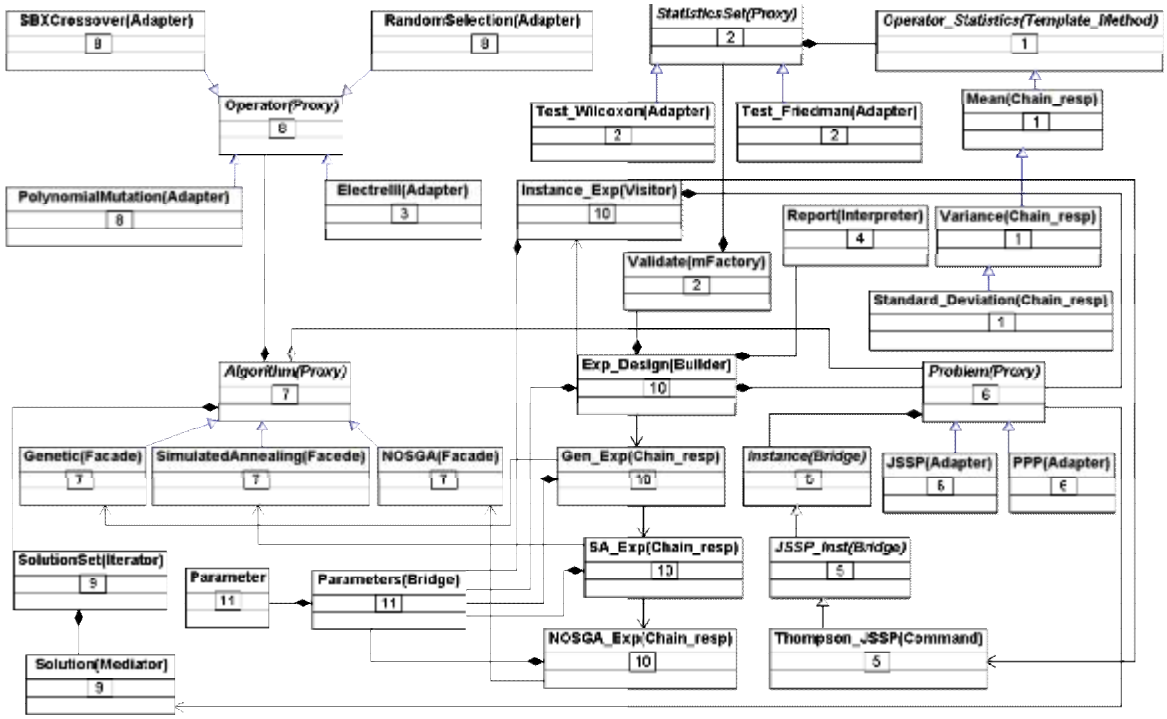



**Figura 21.** Arquitectura evolucionada del Framework de optimización.

Por otro lado, se llevaron a cabo otras experimentaciones del framework optimización, pero incluyendo las arquitecturas de JMetal v5.0, JMetal v3.1 y JAMES v0.2. Estas experimentaciones se detallan en el Anexo A.

## 6.3 Validación del framework de optimización

La validación de software se utiliza para demostrar que el sistema se ajusta a su especificación y que cumple con las expectativas del usuario que lo adquirirá [Sommerville, 2005]. Así mismo, este trabajo fue validado mediante la comprobación de su funcionalidad esperada, la cual se logró incorporando distintos problemas y algoritmos bajo un esquema de patrones de diseño. Uno de estos algoritmos implementados, fue el algoritmo genético, utilizado para evolucionar los patrones de diseño como se aprecia en la sección anterior, y la resolución de distintos problemas de optimización. Por otra parte, también se implementaron en el framework diversos problemas de optimización, como por ejemplo: el problema de Job Shop Sheduling (JSSP).

### 6.3.1 Implementación del problema de Job Shop Scheduling (JSSP)

Al igual que el diseño experimental y el algoritmo genético, el problema de Job Shop Scheduling también fue implementado en la primera arquitectura del framework (ver Figura 8). Este problema se resolvió utilizando el algoritmo genético, el cual se configuró de acuerdo a los parámetros que se muestran en las Tablas 19 y 20.

**Tabla 19.** Parámetros del algoritmo genético.

| Parámetro | Valor |
|---|---|
| Número de Experimentaciones | 5 |
| Número de Generaciones | 5 |
| Tamaño de la población | 100 |

**Tabla 20.** Parámetros de los operadores.

| Operador | Método | Probabilidad |
|---|---|---|
| Selección | Torneo binario | |
| Cruza | Simulated Binary Crossover | 0.9 |
| Muta | Simple mutación | 0.2 |

Las instancias utilizadas en esta experimentación fueron: **ft06-mt06, ft10- mt10, and ft20-mt20** de Fischer & Thompson, (1963), las cuales se describen a continuación:

1. Las primeras 6 líneas es el encabezado de la instancia.
2. En la 7 se especifica el número de tareas y el número de máquinas.
3. Después, en cada línea subsiguiente se enmarca una máquina y su tiempo de procesamiento en cada tarea comprendida por el trabajo. Las máquinas están numeradas del 1 al 6.

Enseguida en la Figura 22, se presenta el formato que contienen las instancias anteriormente menciionadas, utilizando como ejemplo la instancia **ft06-mt06:**

```
+++++++++++++++++++++++++++
instance ft06
+++++++++++++++++++++++++++

Fisher and Thompson 6x6 instance, alternate name (mt06)
machine (processing time)
 6 6
 3  1  1  3  2  6  4  7  6  3  5  6
 2  8  3  5  5 10  6 10  1 10  4  4
 3  5  4  4  6  8  1  9  2  1  5  7
 2  5  1  5  3  5  4  3  5  8  6  9
 3  9  2  3  5  5  6  4  1  3  4  1
 4  3  4  3  6  9  1 10  5  4  3  1
```

**Figura 22.** Instancia ft06-mt06

En la Tabla 21 se muestran las características más generales que contiene cada instancia utilizada en esta experimentación:

**Tabla 21.** Características de las instancias utilizadas.

| ID | INSTANCE | JOBS | MACHINES | MSC |
|---|---|---|---|---|
| 1 | ft06-mt06 | 6 | 6 | 55 |
| 2 | ft10-mt10 | 10 | 10 | 930 |
| 3 | ft20-mt20 | 20 | 5 | 1165 |

**MSC:** Mejor solución conocida.

Una vez realizado la implementación y configuración adecuada, se procede a llevar a cabo el proceso de experimentación, mostrando los resultados en la columna *fitness* en las Tablas 22, 23, 24, 25, 26 y 27. Realizando una comparación de estos resultados, con respecto a las *mejores soluciones conocidas* (MSC) de la Tabla 21, se puede observar que los valores obtenidos "son mejores que las MSC"; más sin embargo, esto no es así. Esto posible porque son alcanzados, porque se generaron soluciones aleatorias, sin verificar su posible factibilidad. No obstante, se le agregó un indicador que midiera la calidad de estas soluciones, calificando las veces en que estas violaban alguna restricción. Por lo tanto al generar la siguiente población, se hacían preferencia a las soluciones con menos infactibilidad.

**Tabla 22.** Resultados de la Experimentación 1 de la instancia **ft06-mt06**.

| **Individuo** | ***fitness*** | **Calificación** |
|---|---|---|
| 1 | 43.00 | -27.000 |
| 2 | 53.00 | -31.000 |
| 3 | 67.00 | -34.000 |
| 4 | 54.00 | -34.000 |
| : | : | : |
| 49 | 57.00 | -29.000 |

| | | |
|---|---|---|
| 50 | 70.00 | -33.000 |
| 51 | 45.00 | -31.000 |
| 52 | 50.00 | -30.000 |
| : | : | : |
| 97 | 51.00 | -33.000 |
| 98 | 58.00 | -29.000 |
| 99 | 59.00 | -27.000 |
| 100 | 46.00 | -33.000 |

**Tabla 23.** Resultados de la Experimentación 5 de la instancia **ft06-mt06**.

| **Individuo** | ***fitness*** | **Calificación** |
|---|---|---|
| 1 | 43.00 | -30.000 |
| 2 | 57.00 | -28.000 |
| 3 | 62.00 | -30.000 |
| 4 | 58.00 | -32.000 |
| : | : | : |
| 49 | 62.00 | -30.000 |
| 50 | 55.00 | -29.000 |
| 51 | 52.00 | -34.000 |
| 52 | 71.00 | -30.000 |
| : | : | : |
| 97 | 44.00 | -31.000 |
| 98 | 50.00 | -31.000 |
| 99 | 55.00 | -27.000 |
| 100 | 44.00 | -30.000 |

**Tabla 24.** Resultados de la Experimentación 1 de la instancia **ft10-mt10**.

| **Individuo** | ***fitness*** | **Calificación** |
|---|---|---|
| 1 | 899.00 | -92.000 |
| 2 | 679.00 | -88.000 |
| 3 | 844.00 | -92.000 |
| 4 | 754.00 | -89.000 |
| : | : | : |
| 49 | 856.00 | -88.000 |
| 50 | 1060.00 | -84.000 |
| 51 | 762.00 | -87.000 |
| 52 | 958.00 | -94.000 |
| : | : | : |
| 97 | 842.00 | -87.000 |
| 98 | 737.00 | -89.000 |
| 99 | 839.00 | -91.000 |
| 100 | 648.00 | -94.000 |

**Tabla 25.** Resultados de la Experimentación 5 de la instancia **ft10-mt10**.

| **Individuo** | ***fitness*** | **Calificación** |
|---|---|---|
| 1 | 814.00 | -90.000 |
| 2 | 797.00 | -95.000 |
| 3 | 882.00 | -85.000 |
| 4 | 692.00 | -94.000 |
| : | : | : |
| 49 | 934.00 | -93.000 |
| 50 | 995.00 | -89.000 |
| 51 | 887.00 | -92.000 |
| 52 | 724.00 | -89.000 |
| : | : | : |
| 97 | 718.00 | -88.000 |
| 98 | 987.00 | -86.000 |
| 99 | 684.00 | -91.000 |
| 100 | 892.00 | -84.000 |

**Tabla 26.** Resultados de la Experimentación 1 de la instancia **ft20-mt20**.

| **Individuo** | ***fitness*** | **Calificación** |
|---|---|---|
| 1 | 1366.00 | -81.000 |
| 2 | 1288.00 | -78.000 |
| 3 | 1579.00 | -85.000 |
| 4 | 1380.00 | -84.000 |
| : | : | : |
| 49 | 1219.00 | -80.000 |
| 50 | 1231.00 | -84.000 |
| 51 | 1625.00 | -77.000 |
| 52 | 1610.00 | -79.000 |
| : | : | |
| 97 | 1255.00 | -84.000 |
| 98 | 1573.00 | -89.000 |
| 99 | 1384.00 | -85.000 |
| 100 | 1219.00 | -71.000 |

**Tabla 27.** Resultados de la Experimentación 5 de la instancia **ft20-mt20**.

| **Individuo** | ***fitness*** | **Calificación** |
|---|---|---|
| 1 | 1178.00 | -79.000 |
| 2 | 1276.00 | -89.000 |
| 3 | 1252.00 | -80.000 |
| 4 | 1361.00 | -80.000 |
| : | : | : |
| 49 | 1341.00 | -77.000 |
| 50 | 1450.00 | -82.000 |

| | | |
|---|---|---|
| 51 | 1234.00 | -88.000 |
| 52 | 1297.00 | -79.000 |
| : | : | : |
| 97 | 1253.00 | -82.000 |
| 98 | 1219.00 | -77.000 |
| 99 | 1156.00 | -81.000 |
| 100 | 1307.00 | -86.000 |

Tal como se muestran en las experimentaciones de las tablas anteriores, es posible agregar otros problemas y algoritmos, sin la necesidad de modificar a los que ya están incorporados en el framework. Esto se logra corroborar, implementado los problemas de TSP y PPP, sin necesidad de modificar al algoritmo genético, ni al problema de Jop Shop Scheduling.

# CAPÍTULO 7

## 7. CONCLUSIONES Y TRABAJOS FUTUROS

Mediante la implementación del proceso evolutivo descrito en la Tabla 7, se pudo constatar el impacto que resulta utilizar diferentes patrones de diseño, en la construcción del framework de optimización; ya que estos pueden alteran el número y forma en que se comunican las clases, impactando en gran medida en algunos criterios de modularidad, como son la *cohesión y acoplamiento* [Joyanes, 1996].

Gracias a la implementación del proceso evolutivo, se podrá elegir la arquitectura que conlleve un menor acoplamiento entre paquetes, facilitando en un futuro el mantenimiento del framework. Por tal motivo, el framework resultante tendrá la capacidad de crear, integrar y extender fácilmente nuevas estrategias en MCDA, que permitan a resolver problemas de toma de decisión.

Por último, queda como trabajo a futuro la implementación de estrategias pertenecientes al área de MCDA, para la resolución de problemas de toma de decisiones; y la realización de sus experimentaciones.

# APÉNDICE A

## A. INSTANCIAS UTILIZADAS

En esta sección se presentan dos instancias utilizadas en este trabajo. La primera instancia se construyó mediante la información recopilada de las arquitecturas vistas en las figuras 9 y 10. La segunda instancia se construyó recabando la información contenida en la primera instancia, más la información de las arquitecturas de los Frameworks JAMES v0.2 [Beukelaer et al., 2015], JMetal v3.1 [Durillo & Nebro, 2011] y JMetal v5.0 [Nebro et al., 2015]. Estas instancias se aprecian en la figuras 23 y 24 respectivamente; y son configuradas de acuerdo a la información requerida para la implementación de la métrica [Martin, 2000]. Enseguida se explican las partes en que se compone la primera instancia, las mismas que son utilizadas en la segunda.

a) En la primera línea se establece el número de arquitecturas que comprende la instancia.
b) En la segunda línea, se establece el nombre que identifica la arquitectura a leer.
c) En la tercera, el número de paquetes que contiene dicha arquitectura.
d) En la cuarta, el número total de clases que tiene la arquitectura.
e) En la quinta línea se describen las columnas que contendrán las próximas líneas, desde la 6 hasta la 16. En la primera, segunda y tercera columna contendrá el número del paquete, el número total de clases y el número de total de clases abstractas del paquete al que hace referencia, respectivamente.
f) A partir de la línea 18 a la 28, se especifican los acoplamientos aferentes que tienen los paquetes entre sí; por ejemplo, en la línea 18 se muestra el acoplamiento aferente del paquete 1 con el resto de los otros paquetes, en la línea 19 se presenta el acoplamiento aferente del paquete 2 con el resto de los demás paquetes, y así sucesivamente hasta el paquete 11.
g) En la línea 30 hasta la 40, se presenta de la misma forma que el inciso anterior, pero con la diferencia a que se hace referencia al acoplamiento eferente.
h) Por último, de la misma forma en que se explicó la arquitectura anterior, se procede a explicar la segunda arquitectura, iniciando en la línea 41 con la descripción del inciso ***b*** y terminando en la línea 79 con el inciso ***f***.

| Línea | Información de la Instancia |
|---|---|
| **1** | 2 |
| **2** | Framework_v1 |
| **3** | 11 |
| **4** | 82 |
| **5** | Package/Total Class/Abstract Class: |
| **6** | 1 4 1 |
| **7** | 2 4 1 |

| | |
|---|---|
| **8** | 3 5 1 |
| **9** | 4 1 0 |
| **10** | 5 14 8 |
| **11** | 6 6 1 |
| **12** | 7 8 1 |
| **13** | 8 22 5 |
| **14** | 9 4 0 |
| **15** | 10 11 2 |
| **16** | 11 3 0 |
| **17** | Afferent coupling: |
| **18** | 0 2 0 0 0 0 0 0 0 0 0 |
| **19** | 0 0 0 0 0 0 0 0 0 1 0 |
| **20** | 0 0 0 0 0 0 1 0 0 0 0 |
| **21** | 0 0 0 0 0 0 0 0 0 1 0 |
| **22** | 0 0 0 0 0 5 0 0 0 3 0 |
| **23** | 0 0 0 0 0 0 8 0 3 11 0 |
| **24** | 0 0 0 0 0 0 0 0 0 8 0 |
| **25** | 0 0 0 0 0 0 6 0 0 6 0 |
| **26** | 4 4 1 0 0 6 8 17 0 7 1 |
| **27** | 0 0 0 0 0 0 0 0 0 0 0 |
| **28** | 0 0 0 0 6 0 8 1 0 9 0 |
| **29** | Efferent coupling: |
| **30** | 0 0 0 0 0 0 0 0 1 0 0 |
| **31** | 3 0 0 0 0 0 0 0 1 0 0 |
| **32** | 0 0 0 0 0 0 0 0 1 0 0 |
| **33** | 0 0 0 0 0 0 0 0 0 0 0 |
| **34** | 0 0 0 0 0 0 0 0 0 0 1 |
| **35** | 0 0 0 0 5 0 0 0 1 0 0 |
| **36** | 0 0 1 0 0 1 0 3 2 0 2 |
| **37** | 0 0 0 0 0 0 0 0 2 0 1 |
| **38** | 0 0 0 0 0 1 0 0 0 0 0 |
| **39** | 0 0 0 1 4 5 8 0 0 0 1 |
| **40** | 0 0 0 0 0 0 0 0 0 0 0 |
| **41** | Framework_v2 |
| **42** | 11 |
| **43** | 78 |
| **44** | Package/Total Class/Abstract Class: |
| **45** | 1 4 1 |
| **46** | 2 4 1 |
| **47** | 3 4 1 |
| **48** | 4 1 0 |
| **49** | 5 14 8 |
| **50** | 6 6 1 |
| **51** | 7 8 1 |
| **52** | 8 17 5 |
| **53** | 9 8 1 |
| **54** | 10 9 1 |
| **55** | 11 3 0 |
| **56** | Afferent coupling: |
| **57** | 0 2 0 0 0 0 0 0 0 0 0 |
| **58** | 0 0 0 0 0 0 0 0 0 1 0 |
| **59** | 0 0 0 0 0 0 1 0 0 0 0 |
| **60** | 0 0 0 0 0 0 0 0 0 1 0 |
| **61** | 0 0 0 0 0 5 0 0 0 1 0 |
| **62** | 0 0 0 0 0 0 8 0 3 9 0 |
| **63** | 0 0 0 0 0 0 0 0 0 8 0 |
| **64** | 0 0 0 0 0 0 6 0 0 7 0 |

| | |
|---|---|
| **65** | 4 4 1 0 0 6 8 17 0 7 1 |
| **66** | 0 0 0 0 0 0 0 0 0 0 0 |
| **67** | 0 0 0 0 6 0 8 1 0 9 0 |
| **68** | Efferent coupling: |
| **69** | 0 0 0 0 0 0 0 0 1 0 0 |
| **70** | 2 0 0 0 0 0 0 0 1 0 0 |
| **71** | 0 0 0 0 0 0 0 0 1 0 0 |
| **72** | 0 0 0 0 0 0 0 0 1 0 0 |
| **73** | 0 0 0 0 0 0 0 0 0 0 1 |
| **74** | 0 0 0 0 5 0 0 0 1 0 0 |
| **75** | 0 0 0 0 0 1 0 3 2 0 2 |
| **76** | 0 0 0 0 0 0 0 0 2 0 1 |
| **77** | 0 0 0 0 0 1 0 0 0 0 0 |
| **78** | 0 0 0 0 4 5 5 0 0 0 1 |
| **79** | 0 0 0 0 0 0 0 0 0 0 0 |

**Figura 23.** Primera instancia

La segunda instancia se presenta en la Figura 24, la cual se construyó utilizando la información de la Figura 18, más la información de los Frameworks JAMES v0.2 [Beukelaer et al., 2015], JMetal v3.1 [Durillo & Nebro, 2011] y JMetal v5.0 [Nebro et al., 2015].

| Línea | Información de la Instancia |
|---|---|
| **1** | 5 |
| **2** | Framework_v1 |
| **3** | 11 |
| **4** | 82 |
| **5** | Package/Total Class/Abstract Class: |
| **6** | 1 4 1 |
| **7** | 2 4 1 |
| **8** | 3 5 1 |
| **9** | 4 1 0 |
| **10** | 5 14 8 |
| **11** | 6 6 1 |
| **12** | 7 8 1 |
| **13** | 8 22 5 |
| **14** | 9 4 0 |
| **15** | 10 11 2 |
| **16** | 11 3 0 |
| **17** | Afferent coupling: |
| **18** | 0 2 0 0 0 0 0 0 0 0 0 |
| **19** | 0 0 0 0 0 0 0 0 0 1 0 |
| **20** | 0 0 0 0 0 0 1 0 0 0 0 |
| **21** | 0 0 0 0 0 0 0 0 0 1 0 |
| **22** | 0 0 0 0 0 5 0 0 0 3 0 |
| **23** | 0 0 0 0 0 0 8 0 3 11 0 |
| **24** | 0 0 0 0 0 0 0 0 0 8 0 |
| **25** | 0 0 0 0 0 0 6 0 0 6 0 |
| **26** | 4 4 1 0 0 6 8 17 0 7 1 |
| **27** | 0 0 0 0 0 0 0 0 0 0 0 |
| **28** | 0 0 0 0 6 0 8 1 0 9 0 |
| **29** | Efferent coupling: |
| **30** | 0 0 0 0 0 0 0 0 1 0 0 |
| **31** | 3 0 0 0 0 0 0 0 1 0 0 |
| **32** | 0 0 0 0 0 0 0 0 1 0 0 |

```
33  0 0 0 0 0 0 0 0 0 0 0
34  0 0 0 0 0 0 0 0 0 0 1
35  0 0 0 0 5 0 0 0 1 0 0
36  0 0 1 0 0 1 0 3 2 0 2
37  0 0 0 0 0 0 0 0 2 0 1
38  0 0 0 0 0 1 0 0 0 0 0
39  0 0 0 1 4 5 8 0 0 0 1
40  0 0 0 0 0 0 0 0 0 0 0
41  Framework_v2
42  11
43  78
44  Package/Total Class/Abstract Class:
45  1 4 1
46  2 4 1
47  3 4 1
48  4 1 0
49  5 14 8
50  6 6 1
51  7 8 1
52  8 17 5
53  9 8 1
54  10 9 1
55  11 3 0
56  Afferent coupling:
57  0 2 0 0 0 0 0 0 0 0 0
58  0 0 0 0 0 0 0 0 0 1 0
59  0 0 0 0 0 0 1 0 0 0 0
60  0 0 0 0 0 0 0 0 0 1 0
61  0 0 0 0 0 5 0 0 0 1 0
62  0 0 0 0 0 0 8 0 3 9 0
63  0 0 0 0 0 0 0 0 0 8 0
64  0 0 0 0 0 0 6 0 0 7 0
65  4 4 1 0 0 6 8 17 0 7 1
66  0 0 0 0 0 0 0 0 0 0 0
67  0 0 0 0 6 0 8 1 0 9 0
68  Efferent coupling:
69  0 0 0 0 0 0 0 0 1 0 0
70  2 0 0 0 0 0 0 0 1 0 0
71  0 0 0 0 0 0 0 0 1 0 0
72  0 0 0 0 0 0 0 0 1 0 0
73  0 0 0 0 0 0 0 0 0 0 1
74  0 0 0 0 5 0 0 0 1 0 0
75  0 0 0 0 0 1 0 3 2 0 2
76  0 0 0 0 0 0 0 0 2 0 1
77  0 0 0 0 0 1 0 0 0 0 0
78  0 0 0 0 4 5 5 0 0 0 1
79  0 0 0 0 0 0 0 0 0 0 0
80  JMetal_v3.1
81  4
82  29
83  Package/Total Class/Abstract Class:
84  6 6 1
85  7 7 1
86  8 9 1
87  9 7 2
88  Afferent coupling:
89  0 9 0 0
```

| | |
|---|---|
| **90** | 0 0 0 0 |
| **91** | 0 9 0 0 |
| **92** | 6 9 0 0 |
| **93** | Efferent coupling: |
| **94** | 0 0 0 2 |
| **95** | 1 0 5 1 |
| **96** | 0 0 0 0 |
| **97** | 0 0 0 0 |
| **98** | JMetal_v5.0 |
| **99** | 4 |
| **100** | 26 |
| **101** | Package/Total Class/Abstract Class: |
| **102** | 6 6 1 |
| **103** | 7 7 1 |
| **104** | 8 9 1 |
| **105** | 9 4 4 |
| **106** | Afferent coupling: |
| **107** | 0 9 0 0 |
| **108** | 0 0 0 0 |
| **109** | 0 9 0 0 |
| **110** | 6 9 0 0 |
| **111** | Efferent coupling: |
| **112** | 0 0 0 1 |
| **113** | 1 0 5 1 |
| **114** | 0 0 0 0 |
| **115** | 0 0 0 0 |
| **116** | JAMES_v0.2 |
| **117** | 5 |
| **118** | 13 |
| **119** | Package/Total Class/Abstract Class: |
| **120** | 6 5 1 |
| **121** | 7 3 0 |
| **122** | 8 2 0 |
| **123** | 9 1 0 |
| **124** | 11 2 0 |
| **125** | Afferent coupling: |
| **126** | 0 3 0 0 0 |
| **127** | 0 0 0 0 0 |
| **128** | 0 1 0 0 0 |
| **129** | 4 3 2 0 0 |
| **130** | 0 2 0 0 0 |
| **131** | Efferent coupling: |
| **132** | 0 0 0 1 0 |
| **133** | 1 0 1 1 2 |
| **134** | 0 0 0 1 0 |
| **135** | 0 0 0 0 0 |
| **136** | 0 0 0 0 0 |

**Figura 24.** Segunda instancia

Esta segunda instancia se construyó en el mismo orden que la primera instancia, solo que a diferencia de esta última, se le incorporaron nuevas arquitecturas pertenecientes a otros frameworks.

# APÉNDICE B

## B. INCORPORACIÓN DE OTROS FRAMEWORKS DE OPTIMIZACIÓN

En esta sección se presenta la segunda experimentación, utilizando como datos de entrada la segunda instancia que se presentó en el Apéndice A. En esta experimentación, se llevó a cabo la construcción de la solución seleccionando aleatoriamente una de las cinco opciones que se muestran en la Tabla 28, según el paquete deseado. Cada paquete representará un elemento de la solución, como se aprecia en la Figura 25; y esta solución será representada por cinco números; donde el *uno* representara la opción 1, el *dos* representará la opción 2, el *tres* representará la opción 3, el *cuatro* representará la opción 4 y el *cinco* que será representará por la opción 5. Cada una de estas opciones representara el Framework elegido en cada paquete.

| Modulo | 1 | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Paquete | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Opción | 1 | 2 | 1 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 5 |
| Solución | 1 | 2 | 1 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 5 |

**Figura 25.** Configuración de la solución.

Los paquetes en cada solución solo se representarán por los Frameworks que contenga el paquete en determinada ubicación. Como se aprecia en la Tabla 28, los paquetes 1, 2, 3 y 4 solo podrán ser elegidos por los Frameworks en las opciones 1 y 2, ya que los demás Frameworks no cuentan con estos paquetes.

Posteriormente, al igual que la primera experimentación, se utilizó la métrica descrita en la sección 5.2, evaluando los paquetes de las dos arquitecturas del Framework propuesto, junto con los de JAMESv3.1, JMetalv3.1 y JMetalv5.0; presentado el *fitness* de cada uno en la Tabla 28.

**Tabla 28.** Evaluación de las cinco Arquitecturas.

| **Opción** | **Frameworks** | **Packages** | ***fitness*** |
|---|---|---|---|
| 1 | Arquitectura 1 | 1 2 3 4 5 6 7 8 9 10 11 | 55.7513 |
| 2 | Arquitectura 2 | 1 2 3 4 5 6 7 8 9 10 11 | 54.8716 |
| 3 | JMetal v3.1 | 6 7 8 9 | 5.2191 |
| 4 | JMetal v5.0 | 6 7 8 9 | 3.8217 |
| 5 | JAMES v0.2 | 6 7 8 9 11 | 11.3373 |

Cabe mencionar que los paquetes de JAMESv3.1, JMetalv3.1 y JMetalv5.0, fueron creados mediante la agrupación de sus clases; utilizando como criterio los módulos que se describen en la Tabla 8. Estas clases fueron tomadas utilizando la literatura perteneciente a cada framework.

Ahora se procede a realizar las experimentaciones del algoritmo genético, utilizando las configuraciones de las Tablas 12 y 13. El resultado de las experimentaciones se presenta en las Tablas 29, 30, 31 y 32.

**Tabla 29.** Resultados de la Experimentación 1.

| **Individuo** | **Solución** | ***fitness*** |
|---|---|---|
| 1 | 2 2 2 1 1 1 3 2 4 1 1 | 47.9467 |
| 2 | 1 2 2 1 2 1 2 2 4 1 1 | 48.3675 |
| 3 | 2 1 1 1 1 4 3 1 4 1 5 | 48.5372 |
| 4 | 1 1 2 2 1 2 3 2 4 2 5 | 48.6591 |
| : | : | : |
| 49 | 2 1 1 2 2 4 4 5 3 1 1 | 55.6664 |
| 50 | 2 1 1 1 2 4 1 1 5 1 5 | 55.6756 |
| 51 | 2 1 1 1 1 2 1 1 1 1 2 | 55.7513 |
| 52 | 2 1 1 2 1 2 3 3 3 2 1 | 55.8289 |
| : | : | : |
| 97 | 2 2 2 1 1 2 2 5 1 2 5 | 58.6601 |
| 98 | 2 1 1 1 2 4 5 4 1 2 5 | 59.1708 |
| 99 | 1 1 2 2 2 2 5 5 5 2 1 | 59.5272 |
| 100 | 2 2 1 2 2 3 5 5 1 2 1 | 59.5440 |

**Tabla 30.** Resultados de la Experimentación 15.

| **Individuo** | **Solución** | ***fitness*** |
|---|---|---|
| 1 | 1 1 2 1 1 5 2 1 4 1 2 | 48.4159 |
| 2 | 1 1 2 1 2 2 2 1 4 1 5 | 48.7813 |
| 3 | 2 2 1 1 2 4 3 1 4 1 1 | 48.9496 |
| 4 | 2 1 1 2 1 2 3 1 4 2 2 | 49.1318 |
| : | : | : |
| 49 | 1 2 1 1 1 2 1 5 3 1 1 | 55.9694 |
| 50 | 2 1 2 1 2 2 3 4 2 1 2 | 56.0929 |
| 51 | 1 2 1 1 1 4 2 4 3 2 2 | 56.1319 |
| 52 | 2 1 2 2 2 1 2 1 1 2 5 | 56.1692 |
| : | : | : |
| 97 | 2 2 1 2 2 1 3 5 5 2 1 | 58.6938 |
| 98 | 2 1 2 2 2 2 5 4 5 2 5 | 58.7415 |
| 99 | 1 1 1 1 1 1 2 5 1 2 5 | 58.9547 |
| 100 | 2 2 1 2 2 3 5 5 5 2 2 | 59.6450 |

**Tabla 31.** Resultados de la Experimentación 30.

| Individuo | Solución | *fitness* |
|---|---|---|
| 1 | 2 1 2 2 2 1 4 2 4 1 1 | 47.8877 |
| 2 | 1 1 2 2 2 5 1 2 4 2 2 | 48.0076 |
| 3 | 2 2 2 2 2 4 4 2 4 2 1 | 48.1877 |
| 4 | 1 2 1 2 2 4 1 1 4 1 2 | 48.3099 |
| : | : | : |
| 49 | 2 1 2 2 1 1 2 3 3 2 5 | 55.5427 |
| 50 | 1 1 2 1 2 2 1 3 3 2 2 | 55.5427 |
| 51 | 2 1 2 2 2 4 4 2 5 2 2 | 55.5534 |
| 52 | 1 1 1 2 1 4 2 1 5 1 1 | 55.6756 |
| : | : | : |
| 97 | 2 2 1 2 2 2 3 5 1 2 1 | 58.5928 |
| 98 | 1 1 1 1 1 4 3 5 1 2 1 | 58.7106 |
| 99 | 2 2 1 1 2 5 5 5 2 2 2 | 58.9430 |
| 100 | 1 1 2 2 2 4 5 5 1 2 5 | 59.2494 |

**Tabla 32.** Resultados de la Experimentación 50.

| Individuo | Solución | *fitness* |
|---|---|---|
| 1 | 2 2 1 2 2 3 3 2 4 1 5 | 48.0056 |
| 2 | 1 2 2 1 1 5 1 2 4 2 2 | 48.4200 |
| 3 | 2 1 2 1 2 3 2 2 4 2 2 | 48.7854 |
| 4 | 1 2 2 2 2 2 4 4 4 1 2 | 49.3586 |
| : | : | : |
| 49 | 2 2 1 1 1 5 2 4 3 2 5 | 55.9434 |
| 50 | 2 1 1 1 2 2 5 1 3 2 1 | 56.1355 |
| 51 | 1 2 2 2 1 1 1 4 2 2 5 | 56.2835 |
| 52 | 2 1 1 2 1 3 2 5 3 2 2 | 56.3873 |
| : | : | : |
| 97 | 1 1 2 1 1 4 4 5 1 2 5 | 58.3571 |
| 98 | 1 2 1 2 1 4 5 5 1 1 5 | 58.5369 |
| 99 | 2 2 1 2 1 4 5 3 1 2 5 | 58.8762 |
| 100 | 2 2 2 1 1 2 5 5 2 2 2 | 58.9547 |

Como se pudo observar, los resultados obtenidos dan mejores *fitness* en comparación de los resultados de la primera experimentación. Esto es posible porque la cantidad de clases utilizadas en estos nuevos frameworks, proporcionan una menor cantidad de clases y una distinta configuración de estas, impactando en gran medida el proceso de evaluación. Sin embargo, este comportamiento también se puede conseguir utilizando distintos patrones, ya que estos también pueden alterar el número y configuración de clases.

# REFERENCIAS


[Adra et al., 2007] Adra, S. F., Griffin, I., Fleming, P. J. (2007). A comparative study of progressive preference articulation techniques for multiobjectiveoptimisation. In Proceedings of the 4th international conference on Evolutionary Multi-criterion Optimization (EMO'07), Springer, Matsushima, Japan, pp. 908–921.

[Alexander, 1977] Alexander, C. (1977). *A pattern language: towns, buildings, construction*. Oxford University Press.

[Armstrong, 1939] Armstrong, W.E. (1939). The determinateness of the utility function. The Economic Journal, 49:453–467.

[Balderas, 2011] Balderas F. (2011) Sistema de Apoyo a la Decisión para la Selección de Carteras de Proyectos en Organizaciones Públicas basado en Agentes. Tesis de maestría, Instituto Tecnológico de Cd Madero, Tamaulipas, México.

[Beukelaer et al., 2015] Beukelaer H., Davenport G., Meyer G., & Fack V. (2015). JAMES: A modern object-oriented Java framework for discrete optimization using local search metaheuristics.

[Bhatt & Leighton, 1984] Bhatt, S., Leighton, F., (1984). A framework for solving VLSI graph layout problems. J. Comput. Sytem Sci. 28(2), 300–343.

[Cahon et al., 2004] Cahon, S., Melab, N., & Talbi, E. G. (2004). Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of heuristics*,*10*(3), 357-380.

[Carazo et al., 2010] Carazo, A. F., Gómez, T., Molina, J., Hernández-Díaz, A. G., Guerreo, F. M., Caballero, R. (2010). Solving a comprehensive model for multiobjective project portfolio selection. *Computers & Operations Research*, 37(4):630–639.

[Chisholm & Sosa, 1966] Chisholm, R.M. and Sosa E., (1966). On the logic of intrinsically better. American Philosophical Quarterly, 3:244–249.

[Cohen et al., 1996] Cohen, D., Dalal, S., Parelius, J., Patton, G. (1996). The combinatorial design approach to automatic test generation. IEEE Softw. 13(5), 83–88.

[Cruz-Reyes, L., Fernandez, E., & Rangel-Valdez, N., 2017] Cruz-Reyes, L., Fernandez, E., & Rangel-Valdez, N. (2017). A metaheuristic optimization-based indirect elicitation of preference parameters for solving many-objective problems.

[Deb, 2001] Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. Interscience series in systems and optimization. John Wiley & Sons, LTD.

[Design Patterns, s.f.] Design Patterns. (s.f.). Recuperado el 4 octubre de 2015 de: https://sourcemaking.com/design_patterns

[Debels et al., 2006] Debels, D., De Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, *169*(2), 638-653.

[Dias et al., 2002] L. Dias, V. Mousseau, J. Figueira, J. Clímaco, and C.G. Silva, (2002) . IRIS 1.0 software. Newsletter of the European Working Group "Multicriteria Aid for Decisions", 3(5):4–6.

[Doerner et al., 2004] Doerner, K., Gutjahr, W. J., Hartl, R., Strauss, C., Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals OR*, 131:79–99.

[Dong et al., 2006] Dong, J., Yang, S., & Zhang, K. (2006, March). A model transformation approach for design pattern evolutions. In Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on (pp. 10-pp). IEEE.

[Durillo & Nebro, 2011] Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760-771.

[Edgeworth, 1881] Edgeworth, F. Y. (1881). Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences. London: Kegan Paul.

Ferenc et al., 2005] Ferenc R., Beszédes Á., Fulop L. & Lele J. (2005). Design Pattern Mining Enhanced by Machine Learning. Software Maintenance, 44, 295 - 304.

[Fernández et al., 2011] Fernández, E., López, E., Navarro, J., Vega, I. (2011): "Aplicación de metaheurísticas multiobjetivo a la solución de problemas de cartera de proyectos públicos con una valoración multidimensional de su impacto", Gestión y Política Pública, vol. XX, no. 2, pp. 381-432.

[Fernandez & Navarro, 2012] E. Fernandez, J. Navarro, G. M., (2012). Evolutionary multi-objective optimization for inferring outranking model's parameters under scarce reference information and effects of reinforced preference. Foundations of Computing and Decision Sciences, 37(3), 163–197.

[Fernandez et al., 2013] Fernandez, E., Lopez, E., Mazcorro, G., Olmedo, R., Coello Coello, C. A., (2013). Application of the non-outranked sorting genetic algorithm to public project portfolio selection, Information Sciences, 228(10):131-149.

[Figueira et al., 2005] Figueira, J., Greco, S., y Ehrogott, M., (2005). Multi Criteria Decision Analysis: State of the Art Surveys. New York. Springer-Verlag.

[Fishburn, 1999] Fishburn, P.C, (1999). Preference structures and their numerical presentations. Theoretical Computer Science, 217:359–389.

[Fischer & Thompson, 1963] Fischer, C., & Thompson. , G. (1963). Probabilistic learning combinations of local job-shop scheduling rules. En J. Muth, & G. Thompson, *Industrial Scheduling* (págs. 225–251). Englewood Cliffs, N.J., USA: Prentice-Hall.

[Hugan & Selman, 1999] Huang, Y.C. and Selman B., (1990). Control Knowledge in Planning: Benefits and Tradeoffs. Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence. pp. 511 – 517. Orlando, Florida, USA.

[Gamma et al., 1997] Gamma, E., Johnson, R., Helm, R. & Vlissides, J.,(1997). Design patterns: Elements of reusable object-oriented software. Reading: Addison-Wesley, 49(120), 11.

[García, 2010] García, R. (2010). Hiper-heurístico para resolver el problema de cartera de proyectos sociales. Tesis de Maestría, Instituto Tecnológico de Cd. Madero.

[Garey & Johnson, 1979] M.R. Garey and D.S. Johnson., (1979). Computers and Intractability: A Guide to the Theory of NP Completeness. Freeman: San Francisco., 1979. ISBN: 0-

7167-1045–5.

[Gendreau & Potvin, 2005] Gendreau, M., & Potvin, J-Y. (2005). Meta-heuristics in combinatorial optimization, Annals of Operations Research 140 (1) 189-213.

[Gibbs et al., 1976] Gibbs, N., Poole, W., Stockmeyer, P., (1976). An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM Journal on Numerical Analysis 13 (2), 236–250.

[Goldberg, 1989] Goldberg, E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. AddisonWesley Publishing Company, Reading, Massachusetts.

[Guitouni & Martel, 1998] Guitouni, A., & Martel, J. M. (1998). Tentative guidelines to help choosing an appropriate MCDA method. *European Journal of Operational Research*, *109*(2), 501-521.

[Joyanes, 1996] Joyanes Aguilar, L. (1996). *Programación orientada a objetos*. España: McGraw-Hill.

[Keeney & Raiffa, 1993] Keeney, R.L. & Raiffa, H. (1993). Decisions with Multiple Objetives: Prefereces and Value Tradeoffs. Cambridge University Press.

[Kim, 2004] Kim, D. K. (2007). *A meta-modeling approach to specifying patterns* (Doctoral dissertation, Colorado State University. Libraries).

[Kobayashi & Saeki,1999] Kobayashi, T., & Saeki, M. (1999). Software development based on software pattern evolution. In Software Engineering Conference, 1999.(APSEC'99) Proceedings. Sixth Asia Pacific (pp. 18-25). IEEE.

Köksalan et al., 2011 Köksalan, M. M., Wallenius, J., & Zionts, S. (2011). *Multiple criteria decision making: from early history to the 21st century*. World Scientific.

[Kronfeld et al., 2010] Kronfeld, M., Planatscher, H., & Zell, A. (2010). The EvA2 optimization framework. In *Learning and Intelligent Optimization* (pp. 247-250). Springer Berlin Heidelberg.

[Laumanns et al., 2002] Laumanns, M., Thiele, L., Deb, K., Zitzler, E. (2002) Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation, 10(3):263 – 282.

[Li, 2012; Tan, 2008] Li, K., Kwong, S., Cao, J., Li, M., Zheng, J., Shen, R. (2012). Achieving balance between proximity and diversity in multi-objective evolutionary algorithm, Information Sciences 182 (1): 220–242.

[Luke, 2009] Luke, S., (2009). Essentials of Metaheuristics. Available at http://cs.gmu.edu/sean/book/metaheuristics/.

[Maplesden, D. et al., 2007] Maplesden, D., Eden, M., Hosking, J., & Grundy, J. (2007). A Visual Language for Design Pattern Modelling and Instantiation.

[Marquez, 2012] Marquez Delgado, J. E. (2012). *Optimización de la programación (scheduling) en Talleres de Mecanizado* (Doctoral dissertation, Agronomos).

[Martí et al., 2008] Martí, R., Campos, V., Piñana, E., (2008). A branch and bound algorithm for the matrix bandwidth minimization. European Journal of Operational Research 186 (2), 513–528.

[Martin, 2000] Martin, R. C. (2000). Design principles and design patterns. Object Mentor, 1, 34.

[Michael et al.,1997] Michael, C., McGraw, G., Schatz, M. & Walton, C., (1997) Genetic algorithms for dynamic test data generation. In: Automated Software Engineering, 1997. Proceedings., 12th IEEE International Conference, pp. 307–308.

[Montgomery, 2004] Montgomery, D. C. (2004). *Diseño y análisis de experimentos*. México DF, México: Limusa.

[Montgomery, 2007] Montgomery, J. (2007). Alternative solution representations for the job shop scheduling problem in ant colony optimisation. In *Progress in Artificial Life* (pp. 1-12). Springer Berlin Heidelberg.

[Nebro et al., 2015] Nebro, A. J., Durillo, J. J., & Vergne, M. (2015, July). Redesigning the jMetal multi-objective optimization framework. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* (pp. 1093-1100). ACM.

[Nicholson et al., 2013] Nicholson J., Eden A., Gasparis E. & Kazman R. (2013). Automated verification of design patterns: A case study. *Science of Computer Programming*, *vol* (80), pp. 211-222. doi.org/10.1016/j.scico.2013.05.007.

[Oztürk et al., 2005] Oztürk, M., Tsoukiàs, A. & Vincke, Ph., (2005). Preference Modelling, In: State of the Art in Multiple Criteria Decision Analysis, M. Ehrgott, Greco, S. and Figueira, J. (Ed.). Wiley Series on Intelligent Systems. Springer-Verlag. pp. 27 – 72.

[Pareto, 1896] Pareto, V. (1896). Cours d'economie politique, volume 1 & 2. Lausanne.

[Pérez et al., 2007] Pérez, F. Molina, J., Caballero, R., Coello, C., & Hernández, A., (2007). Hibridación de métodos exactos y heurísticos para el problema multiobjetivo. Journal Economic Literature: C61; C63. XV Jornadas de ASEPUMA y III Encuentro Internacional.

[Perny & Roy, 1992] Perny, P. and Roy, B., (1992). The use of fuzzy outranking relations in preference modelling. Fuzzy Sets and Systems, 49:33–53.

[Perny & Spanjaard, 2002] Perny, P. and Spanjaard, O., (2002). Preference-based search in state space graphs. In Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'2002), pages 751–756, Edmonton, Canada, July 2002. AAAI Press, Cambridge.

[Reiter, 2010] Reiter, P. (2010). Metaheuristic Algorithms for Solving Multi-objective/Stochastic Scheduling and Routing Problems. Tesis Doctoral, University of Wien.

[Rodriguez et al., 2008] Rodriguez-Tello, E., Hao, J., Torres-Jimenez, J., (2008). An improved simulated annealing algorithm for bandwidth minimization. European Journal of Operational Research 185 (3), 1319–1335.

[Roy, 1990] Roy, B. (1990) Reading in Multiple Criteria Decision Aid, chapter The Outranking Approach and the Foundations of ELECTRE methods, pages 155–183. Spinger Verlag.

[Sánchez, 2007] Sánchez, P. (2007). Modelos para la combinación de preferencias en toma de

decisiones: herramientas y aplicaciones. Tesis Doctoral, Universidad de Granada.

[Sánchez, 2017] Sánchez, J. (2016). Nuevos métodos de incorporación de preferencias en meta-heurísticas multiobjetivo para la solución de problemas de cartera de proyectos. Tesis de doctorado, Instituto Tecnológico de Cd. Madero, Tamaulipas, México.

[Santana, 2004] Santana, L. (2004). Un Algoritmo Basado en Evolución Diferencial para Resolver Problemas Multiobjetivo. Tesis de Maestría, Centro de Investigación y de Estudios avanzados del Instituto Politécnico Nacional, Departamento de Ingeniería Eléctrica sección de Computación.

[Saxe, 1980] Saxe, J., (1980). Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time. Journal on Algebraic and Discrete Methods 1 (4), 363–369.

[Serrano, 2007] Serrano, V. (2007). Métodos para reducir evaluaciones en algoritmos evolutivos multi-objetivo, basados en aproximación de funciones. Tesis de Maestría, Centro de Investigación y de Estudios avanzados del Instituto Politécnico Nacional, Departamento de Ingeniería Eléctrica sección de Computación.

[Scenna et al., 2015] Scenna, N. J., Aguirre, P. A., Benz, S. J., Chiotti, O. J., Espinosa, H. J., Ferrero, M. B., & Salomone, H. E. (2015). Modelado, simulación y optimización de procesos químicos.

[Sotskov et al., 2002] Sotskov, Y.N., Tanaev, V.S., Werner, F., (2009). Scheduling problems and mixed graph colorings. Optimization 51(3), 597–624

[Sohrabi et al., 2014] Sohrabi, S., Udrea, O., and Riabov, A., (2014). Knowledge Engineering for Planning-Based Hypothesis Generation. In Proceedings of the Automated Planning and Scheduling (ICAPS) Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), pages 46-53.

[Sommerville, 2005] Sommerville, I. (2005). *Ingeniería del software*. Pearson Educación.

[Sridaran et al., 2009] Sridaran R., Padmavathi G. & Iyakutti K., (2009). "A Survey of Design Pattern Based Web Applications", in Journal of Object Technology, vol. 8, no. 2, March-April 2009, pp. 61-7

[Tedeschi, s.f.] Tedeschi, N. (s.f.). ¿Qué es un Patrón de Diseño? . Recuperado el 4 octubre de 2015 de: https://msdn.microsoft.com/es-es/library/bb972240.aspx

[Téllez, 2007] Téllez Enríquez, E. (2007). *Uso de una Colonia de Hormigas para resolver Problemas de Programación de Horarios.* Tesis de Maestría, Laboratorio Nacional de Informática Avanzada, Xalapa, Veracruz.

[Van Veldhuizen & Lamont, 1998] Van Veldhuizen, D., & Lamont, G. (1998). Evolutionary Computation and Convergence to a Pareto Front. In: Late Breaking Papers at the Genetic Programming Conference, University of Wisconsin, Madison, Wisconsin, USA, Stanford University Bookstore. John R. Koza, J.R. (Ed.). Morgan Kaufmann pp. 221-228.

[Ventura, S. et al., 2008] Ventura, S., Romero, C., Zafra, A., Delgado, J. A., & Hervás, C. (2008). JCLEC: a Java framework for evolutionary computation. Soft Computing, 12(4), 381-392.

[Yamada, 2003] Yamada, T. (2003). Studies on metaheuristics for jobshop and flowshop scheduling problems.

[Zeleny, 2008] Zeleny, M. (2008). The KM-MCDM interface in decision design: tradeoffs-free conflict dissolution. *International Journal of Applied Decision Sciences*, *1*(1), 3-23.

[Zitzler, 1999] Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications.

.