

**Instituto Tecnológico de Ciudad Madero**

División de Estudio de Posgrado e Investigación

***“Hiper-heurístico para Resolver el Problema  
de Cartera de Proyectos Sociales”***

Tesis para obtener el grado de  
***Maestro en Ciencias en Ciencias de la  
Computación***

Presenta:

**Rogelio García Rodríguez**

Director:

Dra. Laura Cruz Reyes

Co-Director:

Dr. Fernando López Irarragorri

Cd. Madero, Tampaulipas, México

Noviembre 2010

## 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 derechos 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.

En caso de infracción de 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-directores de tesis, así como al Instituto Tecnológico de Cd. Madero y sus autoridades.

18 de Noviembre del 2010

---

Ing. Rogelio García Rodríguez

# Índice general

Índice general	III
Índice de tablas	VII
Índice de figuras	IX
<b>1. Introducción</b>	<b>1</b>
1.1. Antecedentes . . . . .	2
1.2. Problema de Investigación . . . . .	3
1.3. Hipótesis . . . . .	3
1.4. Justificación . . . . .	4
1.5. Objetivos . . . . .	5
1.5.1. Objetivo General . . . . .	5
1.5.2. Objetivos Específicos . . . . .	5
1.6. Alcances y Limitaciones . . . . .	6
<b>2. Marco Conceptual</b>	<b>7</b>
2.1. Problema de la Mochila . . . . .	8
2.1.1. Problemas Complejos . . . . .	8
2.1.2. Problema de la Mochila 0/1 . . . . .	9
2.2. Problema de Cartera de Proyectos . . . . .	11
2.2.1. Proyectos Sociales . . . . .	11
2.3. Optimización Multicriterio . . . . .	12
2.3.1. Frente de Pareto . . . . .	13
2.3.2. Algoritmos Evolutivos Multiobjetivos . . . . .	14
2.3.3. Modelo de Optimización Entera . . . . .	15
2.4. Hiper-heurísticos . . . . .	22
2.4.1. Tipos de Hiper-heurísticos . . . . .	23

2.4.2. Hiper-heurísticos de Selección . . . . .	23
2.5. Métricas de Desempeño de Algoritmos . . . . .	25
<b>3. Estado del Arte</b>	<b>27</b>
3.1. Problema de la Mochila . . . . .	28
3.1.1. Algoritmo Determinista . . . . .	28
3.1.2. Algoritmo Hiperheurístico . . . . .	28
3.2. Problema de Cartera de Proyectos . . . . .	29
3.2.1. NOSGA . . . . .	29
3.2.2. NSGA-II . . . . .	32
3.2.3. Algoritmo Evolutivo Multiobjetivo . . . . .	32
3.3. Conclusiones . . . . .	33
3.3.1. Mochila y Cartera de Proyectos Sociales . . . . .	33
3.3.2. Hiper-heurísticas . . . . .	34
<b>4. Propuesta de Solución</b>	<b>37</b>
4.1. Problema de la Mochila 0/1 . . . . .	38
4.1.1. Representación de la solución . . . . .	39
4.1.2. Heurísticas de Bajo Nivel . . . . .	40
4.1.3. Heurística de Alto Nivel . . . . .	42
4.2. Problema de Cartera de Proyectos Sociales . . . . .	44
4.2.1. Representación de la solución . . . . .	44
4.2.2. Heurísticas de Bajo Nivel . . . . .	45
4.2.3. Heurística base para el hiper-heurístico . . . . .	46
4.3. Mejoras al algoritmo . . . . .	47
4.3.1. Métricas del Comportamiento . . . . .	48
4.3.2. Mejoras al Algoritmo . . . . .	52
<b>5. Experimentación y Resultados</b>	<b>55</b>
5.1. Ambiente General . . . . .	55
5.2. Problema de la Mochila . . . . .	56
5.3. Problema de Cartera de Proyectos Sociales . . . . .	58
5.3.1. Instancias de Prueba . . . . .	58
5.3.2. Prueba con Instancias de 4 Objetivos . . . . .	61
5.3.3. Pruebas con Instancias de 9 Objetivos . . . . .	62

<i>ÍNDICE GENERAL</i>	v
-----------------------	---

<b>6. Conclusiones y Trabajos Futuros</b>	<b>65</b>
6.1. Beneficios . . . . .	65
6.2. Trabajos Futuros . . . . .	66
<b>Bibliografía</b>	<b>67</b>



# Índice de tablas

2.1.	Datos de los proyectos disponibles . . . . .	18
2.2.	Alternativas de solución al problema de cartera . . . . .	18
2.3.	Matriz de concordancia $C_{x,y}$ . . . . .	19
2.4.	Matriz de discordancia $d_1$ . . . . .	20
2.5.	Matriz de discordancia $d_2$ . . . . .	20
2.6.	Matriz de discordancia $N(d(x, y))$ . . . . .	20
2.7.	Matriz $\sigma$ del ejemplo . . . . .	21
3.1.	Matriz $\sigma$ y el número de alternativas a las que dominan y que son dominadas ( $cad(S_0)$ ) . . . . .	30
3.2.	Comparativa de los diferentes trabajos del estado del arte . . .	34
3.3.	Principales trabajos de hiper-heurísticos de selección . . . . .	35
4.1.	Representación de la solución del KP . . . . .	39
4.2.	Ejemplo de un cromosoma del algoritmo genético hiper- heurístico . . . . .	42
4.3.	Representación de la solución del SPP . . . . .	45
4.4.	Heurística de Bajo Nivel y su tipo de movimiento . . . . .	46
4.5.	Ejemplo del cromosoma de un individuo del HHGA_SPP. . . .	47
4.6.	Características del algoritmo genético de HHGA_SPP . . . . .	48
5.1.	Características del equipo usado para los experimentos . . . .	55
5.2.	Instancias para KP de OR-Library . . . . .	56
5.3.	Descripción de las instancias . . . . .	59
5.4.	Descripción de los proyectos en las instancias . . . . .	60
5.5.	Comparación de resultados entre el Algoritmo Exacto, el algoritmo NOSGA y el HHGA_SPP . . . . .	62
5.6.	Resultados del HHGA_SPP . . . . .	62

5.7. Comparación de resultados entre el Algoritmo Exacto, NSGA-II, NOSGA y el HHGA_SPP . . . . .	63
--	----



# Índice de figuras

2.1.	Frente de Pareto de una función con dos objetivos . . . . .	14
2.2.	Clasificación de los hiper-heurístico . . . . .	23
2.3.	Esquema general de un hiper-heurístico . . . . .	24
2.4.	Espacios en que operan los hiper-heurísticos . . . . .	25
4.1.	Diagrama de representación del Hiper-heurístico . . . . .	39
4.2.	Ejemplo del funcionamiento del HHGA_KP . . . . .	43
4.3.	Diagrama de representación del Hiper-heurístico . . . . .	44
4.4.	Métrica $T$ de las instancias 1,2 y 3 . . . . .	50
4.5.	Métrica $F_{hg}$ de la instancia 1 . . . . .	51
4.6.	Métrica $F_{hg}$ de la instancia 2 . . . . .	51
4.7.	Métrica $F_{hg}$ de la instancia 3 . . . . .	51
5.1.	Algoritmo determinista de Pisinger vs HHGA_SPP . . . . .	57
5.2.	Porcentaje de error en el Beneficio obtenido por el HHGA_KP con respecto al algoritmo determinista de Pisinger . . . . .	58
5.3.	Fragmento del archivo de una instancia del SPP . . . . .	59



# Resumen

La correcta selección de proyectos sociales de una cartera de proyectos es uno de los problemas más importantes de las organizaciones públicas, dicho problema es similar al problema típico de la mochila pero más complejo debido a las múltiples restricciones y múltiples objetivos utilizados en el modelo para representar las preferencias del “*Tomador de Decisiones*”. En el presente artículo se explica y ejemplifica el modelo de Fernández *et al* del Problema de Cartera de Proyectos Sociales y su posterior resolución utilizando las nuevas tendencias en investigación en heurísticas y meta-heurísticas, particularmente las llamadas hiper-heurísticas.

Las hiper-heurísticas son procedimientos de búsqueda de muy alto nivel que se sitúan por encima de las metaheurísticas y permiten elevar la generalidad de aplicación haciendo algoritmos más robustos. Al tener un algoritmo robusto sería posible resolver los diferentes modelos propuestos para el Problema de Cartera de Proyectos Sociales. Además, en este trabajo, se proponen dos métricas que permiten estudiar y conocer el comportamiento del algoritmo hiper-heurístico propuesto y que sirven de guía para proponer nuevas estrategias que incrementen la calidad de los resultados.

Palabras clave: Cartera de Proyectos Sociales, Optimización Multiobjetivo, Relación de Dominancia Difusa, Algoritmos Hiper-heurísticos.



# Agradecimiento

*A Dios*

*A mis padres Rogelio García Balderas y  
María del Carmen Rodríguez Castillo  
por todo su apoyo.*

*A todos los que me apoyaron  
en este gran proyecto*



# Capítulo 1

## Introducción

La correcta selección de proyectos sociales de una cartera es uno de los problemas más importantes de las organizaciones públicas y privadas. Dicho problema es similar en su estructura al Problema de la Mochila 0/1 pero más complejo en función de que el modelo utilizado representa preferencias y restricciones que el “Tomador de Decisiones” tiene sobre la forma de seleccionar los proyectos. Por lo tanto el Problema de la Mochila es un límite inferior de la complejidad del Problema de Cartera de Proyectos Sociales.

Se resolvió el problema de carteras de proyectos utilizando las nuevas tendencias en investigación en heurísticas y metaheurísticas, particularmente las llamadas hiper-heurísticas. Desde alrededor del año 2000 se ha empezado a utilizar algoritmos hiper-heurísticos para resolver problemas complejos. Dichos algoritmos son más robustos y sencillos que los algoritmos típicos para resolver este tipo de problemas. Los principales problemas que se han resuelto con hiper-heurísticos son de programación de horarios y programación de personal, este trabajo es uno de los primeros en abordar el Problema de Cartera de Proyectos Sociales.

En este trabajo se plantea primero resolver el Problema de la Mochila 0/1 utilizando un hiper-heurístico que va a funcionar como un *Director de Orquesta* de una serie de heurísticas y metaheurísticas y selecciona en cada momento las mejores heurísticas y metaheurísticas que den las mejores

soluciones al problema. Después de tener experiencia sobre el uso de hiper-heurísticos en un problema fácil se procederá a resolver el problema central de este trabajo que es el Problema de Cartera de Proyectos Sociales.

## 1.1. Antecedentes

Este proyecto es parte de un macro-proyecto que involucra a la Universidad Autónoma de Nuevo León (UANL), Universidad Autónoma de Sinaloa (UAS), Universidad de Occidente (UDO) y el Instituto Tecnológico de Ciudad Madero (ITCM) y tiene como finalidad explorar las posibilidades de colaboración conjunta, además de propiciar un acercamiento fructífero entre los miembros de las universidades. El problema central del macro-proyecto trata sobre la Optimización de Carteras de Proyectos Sociales Financiados por Grandes Organizaciones Públicas. Dar solución a este problema requiere de la aplicación de técnicas y modelos que pertenecen a diferentes disciplinas de las matemáticas y las ciencias de la computación.

En los primeros trabajos se proponen los primeros modelos matemáticos para el problema de cartera de proyectos sociales. Los primeros trabajos fueron presentados por el Dr. Eduardo Fernández, líder del cuerpo académico de la UAS y contó con la colaboración de los Dres. Fernando López Irragorri, de la UANL y Juan Carlos Leyva, líder del cuerpo académico de la UDO. Otros trabajos fueron desarrollados por el Dr. Fernando López y el cuerpo académico de la UANL y participando además el Dr. Eduardo Fernández y los Dres. Igor Litvinchev y Ada Álvarez. En estos trabajos se aborda el problema de Cartera de Proyectos de Investigación y Desarrollo (I&D).

La propuesta actual del macro-proyecto parte de los resultados obtenidos en los trabajos anteriores y tiene como objetivo científico extender los resultados obtenidos tanto para el problema de cartera de proyectos sociales como de I&D, donde el número de objetivos involucrados puede llegar ser de varias decenas. Otro factor que constituye un reto es la evaluación de los proyectos sociales, dado que no existen trabajos previos que se ocupen de este tema, y la evaluación se basa en características que son de naturaleza



totalmente subjetiva.

La colaboración del ITCM va dirigida al desarrollo de algoritmos que permitan resolver los Problemas de Cartera de Proyectos Sociales y de I&D, dada la experiencia que tienen los miembros de la misma en crear algoritmos que resuelvan problemas de alta complejidad, utilizando las técnicas más novedosas en el ámbito de las ciencias computacionales.

En el caso particular de este trabajo se aborda el problema de Cartera de Proyectos Sociales y se resuelve usando Hiper-heurísticos. La contribución más importante de este trabajo es el uso y estudio del hiper-heurístico para resolver tanto el Problema de la Mochila como el de Cartera de Proyectos Sociales.

## 1.2. Problema de Investigación

El problema consiste en diseñar e implementar un algoritmo hiper-heurístico que permita dar soluciones aceptables para el *Tomador de Decisiones* del Problema de Cartera de Proyectos Sociales presentado por Eduardo Fernández *et al* en el artículo “Evolutionary multiobjective optimization using an outranking-based dominance generalization” [13].

## 1.3. Hipótesis

Los algoritmos hiper-heurísticos son consideradas herramientas efectivas para resolver problemas complejos pertenecientes a una misma familia. Al utilizar un hiper-heurísticos basados en algoritmos genéticos ¿ Es posible obtener resultados competitivos que los mejores algoritmos del estado del arte para el Problema de la Mochila?, si esto es así y se usa el mismo algoritmo para el Problema de Cartera de Proyectos Sociales, ¿ Es posible obtener resultados mejores que los algoritmos del estado del arte para el Problema de Cartera de Proyectos Sociales?

Por otro lado, la comunidad científica tiene un interés creciente en entender el comportamiento de algoritmos heurísticos con la finalidad de aplicar el conocimiento adquirido al diseño y generación de teorías. En este sentido, se plantea el interrogante de si ¿Es posible formular métricas que permitan analizar el comportamiento de un hiper-heurístico con el objetivo de mejorarlo?

## 1.4. Justificación

Una de las principales tareas de dirección en las organizaciones de gobierno a cualquier nivel, organizaciones descentralizadas, fundaciones, instituciones que realizan investigación-desarrollo y otras, es evaluar un conjunto de proyectos con impacto social que compiten por apoyo financiero, recurso que es generalmente escaso. Con una cantidad a distribuir inferior a la demanda no se puede otorgar el beneficio a todos los proyectos en competencia, aun cuando fueran aceptables individualmente. La decisión sobre la distribución de los recursos regularmente esta en manos de una persona que **tomará la decisión** de dar o no recursos a un proyecto determinado. En el marco de ciertas restricciones determinadas por la orientación de las políticas públicas, es preciso formar Carteras de Proyectos de calidad donde se maximice el impacto (con connotaciones ideológicas del *Tomador de Decisiones*) de la solución escogida. Se trata de un problema de enorme importancia social en que el costo de pobres soluciones es sencillamente inmenso, aunque su complejidad ha impedido hasta ahora avances verdaderos para resolverlo. La controversia habitual entre los Poderes Ejecutivo y Legislativo por el presupuesto de los Estados Unidos Mexicanos es un excelente ejemplo de la magnitud del reto.

El contar con algoritmos cada vez más rápidos y eficientes permitiría que problemas de alta complejidad, como lo es el Problema de Cartera de Proyectos Sociales, sean resueltos más rápido generando las mejores soluciones posibles. El contar con Sistemas de Toma de Decisiones fundamentados en las matemáticas y ciencias de la computación permitirá que las personas encargadas de tomar las decisiones cuenten con herramientas robustas y efectivas que permitan distribuir de mejor manera los recursos públicos.

Una tendencia vanguardista en el ámbito de las metaheurísticas es utilizar enfoques en los que las metaheurísticas colaboran entre sí de un modo u otro, tal como los hiper-heurísticos [10]. Con este proyecto se espera contribuir al crecimiento de esta área de reciente creación con la finalidad de contar con herramientas computacionales más robustas y rápidas que permitan resolver problemas complejos como el de Cartera de Proyectos Sociales de manera más exacta.

En el ámbito del macro-proyecto existe una variedad de modelos matemáticos que modelan diferentes tipos de carteras (Proyectos Sociales, I&D) y cada modelo cuenta con sus particularidades. El desarrollar un hiper-heurístico permitirá usar un solo algoritmo para todos los modelos propuestos y por proponer sin mucho esfuerzo.

## 1.5. Objetivos

### 1.5.1. Objetivo General

Diseñar e implementar un hiper-heurístico que resuelva el Problema de Cartera de Proyectos de Índole Social tomando como base un Modelo de Optimización que refleje las preferencias y creencias del *Tomador de Decisiones* para medir el beneficio social.

### 1.5.2. Objetivos Específicos

Los objetivos específicos para este trabajo de investigación son los siguientes:

1. Contar con un banco de pruebas (instancias y algoritmos de solución) que permita evaluar la calidad de los hiper-heurístico que se plantean en éste trabajo de investigación.

2. Diseñar e implementar un hiper-heurístico competitivo con el algoritmo del estado del arte para el Problema de la Mochila.
3. Adaptar el hiper-heurístico del Problema de la Mochila al Problema de Cartera de Proyectos Sociales que sea competitivo con el algoritmo del estado del arte.
4. Proponer métricas que permitan conocer el funcionamiento y comportamiento del hiper-heurístico para el Problema de Cartera de Proyectos Sociales.

## 1.6. Alcances y Limitaciones

A continuación se listan los alcances y limitaciones de este trabajo de investigación:

1. Para la formulación matemática del problema se utilizó el modelo de Fernández *et al*[13].
2. En este trabajo sólo se resuelve el Problema de la Mochila y de Cartera de Proyectos Sociales aunque la naturaleza del hiper-heurístico permita ser utilizado para resolver otros problemas.
3. El hiper-heurístico propuesto tiene como base un algoritmo genético.
4. Las métricas propuestas no garantizan de ninguna manera que el algoritmo tenga el comportamiento indicado, aunque dan un buen indicio de su funcionamiento y posibles áreas de mejora.

# Capítulo 2

## Marco Conceptual

En este Capítulo se describirán los conceptos más importantes que engloban a este trabajo. Este trabajo está incluido en los trabajos de una red de investigadores que aborda el problema de asignación de recursos públicos, por lo tanto, los conceptos centrales son el Problema de Cartera de Proyectos Sociales y los algoritmos hiper-heurísticos.

Debido a la inexperiencia en el conocimiento y uso de hiper-heurísticos se decidió primero resolver un problema con menor complejidad que el Problema de Cartera de Proyectos. Dado que los problemas de Cartera de Proyectos están dentro de la familia del problema de la Mochila, primero se resolvió el Problema de la Mochila 0/1.

En esta Sección también se incluyen otros conceptos como la optimización multicriterio y el frente de pareto importantes para este trabajo.

## 2.1. Problema de la Mochila

### 2.1.1. Problemas Complejos

Desde fines de la década de los setenta se han dedicado estudios a la caracterización matemática de los problemas combinatorios. Como fruto de estos estudios se han clasificado dichos problemas en diversas categorías. La clase  $P$  es el subconjunto de problemas cuyos algoritmos de solución presentan una complejidad computacional polinómica, dicho de otro modo, el tiempo de ejecución de estos algoritmos crece de forma polinómica con el tamaño del problema. Este tipo de problemas se puede resolver eficientemente utilizando algoritmos exactos. Sin embargo, para la mayoría de los problemas que tienen interés práctico o científico no se conoce un algoritmo exacto con complejidad polinómica que los resuelva en un tiempo razonable. Los problemas de aplicación de este tipo pertenecen a una clase conocida como  $NP - \text{duros}$  [10].

Dado el interés práctico que tiene la resolución de muchos problemas pertenecientes a la clase  $NP - \text{duros}$  y la dificultad que existe en resolver dichos problemas de forma exacta, se plantea la opción de encontrar soluciones de alta calidad en tiempos razonables, aunque estas soluciones no sean óptimas. Para esta tarea se necesita utilizar algoritmos aproximados que permitan resolver los problemas combinatorios [10].

Los algoritmos heurísticos, según Zanakakis et al. [31], son “*Procedimientos simples, a menudo basados en el sentido común, que se supone obtendrán una buena solución (no necesariamente óptima) a problemas difíciles de un modo sencillo y rápido*”. El principal inconveniente al utilizar algoritmos heurísticos es su incapacidad de escapar de óptimos locales.

Para solventar este problema, se introducen otros algoritmos de búsqueda más inteligentes (denominados metaheurísticas [18]) que evitan, en la medida de lo posible, caer en óptimos locales. Este tipo de algoritmos son procedimientos de alto nivel que guían a métodos heurísticos conocidos, evitando que éstos queden atrapados en óptimos locales.

El Problema de la Mochila es uno de los problemas clásicos de optimización combinatoria *NP-duros* más estudiados, siendo el Problema de Cartera de Proyectos Sociales muy similar al Problema de la Mochila, esté también es un problema *NP-duros*.

### 2.1.2. Problema de la Mochila 0/1

Imagine que un excursionista se dispone a llenar su mochila antes de viajar. Para ello ha seleccionado algunos objetos de entre un conjunto  $N = 1, 2, \dots, n$  de objetos posibles, donde hay una unidad de cada objeto. El objeto  $j \in N$  pesa  $p_j$  kgs., y el excursionista ha fijado un límite de  $P$  kgs. en el peso que está dispuesto a llevar a la espalda en la mochila. Por otra parte, cada objeto  $j \in N$  tiene para el excursionista un valor  $c_j$ . El problema que se plantea es seleccionar tal cantidad de objetos que se llevarán en la mochila tal que se maximice el valor total sin sobrepasar la capacidad de la mochila [21].

#### Formulación

El Problema de la Mochila 0/1 (*Knapsack Problem*, KP) se formula de la siguiente manera:

$$\text{maximizar } Z = \sum_{j=1}^N c_j x_j \quad (2.1)$$

$$\text{sujeto a } \sum_{j=1}^N x_j p_j \leq P$$

$$x_j \in \{0, 1\}, j \in N = \{1, 2, \dots, n\}$$

donde:

$N$ : es el número de objetos disponibles.

$c_j$ : es el beneficio de llevar el objeto  $j$ .

$p_j$ : es el peso de llevar el objeto  $j$ .

$P$ : es el límite de Kgs. de la mochila.

$x_j$ : es tal que  $x_j = 0$  si el objeto  $j$  no esta en la mochila,  $x_j = 1$  en caso contrario.

Pocos excursionistas han usado dicho modelo, sin embargo, si se cambian las palabras “excursionista” por “empresa”, “objetos” por “proyectos”, y “mochila” por “cartera de proyectos”, se obtiene un modelo de selección óptima de proyectos que sí resulta de interés aplicado. Además de ésta, se han encontrado multitud de aplicaciones para el KP, re interpretando el modelo con imaginación en distintos contextos [24].

### Núcleo del KP

El KP puede ser fácilmente resuelto colocando los objetos en orden descendente de acuerdo a su *rendimiento*  $e_j = c_j/p_j$  y usando un algoritmo voraz, es decir, un algoritmo que seleccione los objetos con mejor rendimiento a cada vez para llenar la mochila. El primer ítem  $b$  que no es incluido en la mochila es denotado como el *ítem del cambio*. La solución óptima para el KP estaría dada por los ítem 1 hasta  $b-1$  y algunos otros ítemes mayores a  $b$ . La solución entera correspondiente al ítem de cambio es conocida como *solución de cambio* y los valores de las variables son:  $x_j = 1$  para  $j = 1, \dots, b-1$  y  $x_j = 0$  para  $j = b, \dots, n$  [25].

Balas y Zemel [11] observaron que la solución óptima al KP generalmente correspondía a la *solución de cambio* excepto por algunos pocas variables alrededor de  $b$  que cambiaban. Es en esté subconjunto de objetos donde es más difícil decidir si los objetos entran o no a la mochila y por lo tanto es el **núcleo** del KP [26] [25].



## 2.2. Problema de Cartera de Proyectos

La correcta selección de proyectos para integrar una cartera de proyectos es uno de los problemas más importantes de decisión tanto para instituciones públicas como privadas. Los principales modelos económicos y matemáticos para el problema de cartera de proyectos suponen que se tiene un conjunto definido  $N$  de proyectos, cada proyecto perfectamente caracterizado con costos e ingresos, de los cuales la distribución en el tiempo es conocido. En caso de riesgos, el “Tomador de Decisiones” (*Decision Maker*, DM) debe conocer la probabilidad de distribución de los beneficios [15]. El DM es una persona o un grupo de personas a cargo de seleccionar, a su juicio y dada su experiencia, las mejores soluciones.

### 2.2.1. Proyectos Sociales

La selección de proyectos de una cartera de proyectos sociales necesita de un tratamiento especial por las siguientes razones[15]:

- a) La calidad de los proyectos es generalmente descrita por *múltiples criterios* que frecuentemente son conflictivos.
- b) Frecuentemente, los requerimientos no son conocidos con exactitud. Muchos conceptos no tienen un soporte matemático por ser de *naturaleza totalmente subjetiva*.
- c) La *heterogeneidad* o diferencia entre los posibles proyectos de una misma cartera, dificulta compararlos.

El *impacto en el bienestar social* es el concepto más importante del Problema de Cartera de Proyectos de Índole Social (*Social Porfolio Problems*, SPP) y dicha variable es de naturaleza subjetiva; depende del DM dar un valor a esta variable por cada proyecto. Una forma de modelar las preferencias que el DM tiene es utilizando la *Lógica Difusa* creando predicados tales como

“El proyecto A es muy bueno” o “El proyecto B es malo” para evaluar los proyectos.

Los puntos anteriores son características de proyectos sociales tales como proyectos abocados a educación, salud, transporte público y en general para el bienestar social. Hasta nuestro conocimiento, sólo existen dos aproximaciones científicas concernientes a este tipo de cartera de proyectos[15]:

1. El más usado es el análisis de costo-beneficio. Usando algunas técnicas estadísticas es posible reducir o agrupar el número de variables de un proyecto para representarlas con un valor monetario. Así que el modelo clásico de cartera de proyectos puede ser usado, maximizando el valor de los proyectos seleccionados.
2. Usando un método multicriterio que permita evaluar por completo los proyectos de acuerdo a los costos y beneficios de cada uno y ordenarlos para después distribuir los recursos siguiendo el orden de prioridad de los proyectos.

### 2.3. Optimización Multicriterio

En el mundo real los problemas de optimización son sumamente complejos con muchos **atributos** a evaluar y múltiples **objetivos** a optimizar. Los atributos corresponden a valores cuantitativos que describen al problema y están expresados en función de las variables de decisión. Los objetivos son las direcciones de mejora de los atributos y pueden ser de maximizar o minimizar.

Los deseos y creencias que el DM tienen sobre el problema se expresan, en parte, en el nivel de aspiración que tiene sobre cierto atributo del problema. De este modo muchos problemas de optimización pueden ser representados con una perspectiva multiobjetivo. Pero, aunque se resuelva el problema multiobjetivo todavía no se resuelve el problema central para el DM.

En muchos casos, debido a la naturaleza conflictiva de los atributos no es posible obtener una sola solución y por consecuencia la solución ideal

de un problema multiobjetivo no puede ser alcanzado debido a que no se tiene una única solución al problema. Típicamente, al resolver el problema multiobjetivo se tienen una serie de soluciones o alternativas buenas dado que alcanzaron cierto nivel de aspiración esperado por el DM (frente de Pareto) [5].

Es en este momento en donde surge otro problema, decidir de las múltiples alternativas cual es la que mejor se ajusta a las creencias y preferencias del DM. Los problemas de optimización multicriterio engloban tanto al problema de optimización multiobjetivo como al problema de seleccionar la mejor alternativa. La mayoría de los métodos de Investigación de Operaciones para problemas multicriterio se pueden clasificar en [13]:

1. Técnicas que *a priori* toman en cuenta las preferencias del DM.
2. Métodos interactivos que toman en cuenta las preferencias del DM de manera progresiva.
3. Técnicas que *a posteriori* toman en cuenta las preferencias del DM.

### 2.3.1. Frente de Pareto

La optimización multiobjetivo, como lo es el KP, no se restringe a la búsqueda de una única solución, sino de un conjunto de soluciones llamadas *soluciones no-dominadas*. Cada solución de este conjunto se dice que es un *óptimo de Pareto* y, al representarlas en el espacio de los valores de las funciones objetivos, conforman lo que se conoce como *frente de Pareto*. El obtener el frente de Pareto es una de las principales finalidades de los problemas de optimización multiobjetivo [23].

En la Figura 2.1 se representa, con trazo grueso, el frente de Pareto de una función con 2 objetivos. El área  $T$  representa la imagen de dicha función objetivo. Se puede observar que no existe ningún punto perteneciente a  $T$  que mejore en el sentido de Pareto, a algún punto del Frente: eligiendo un punto de  $T$  de forma arbitraria, por ejemplo  $p3$ , se puede trazar la vertical hasta obtener el punto de corte con el Frente de Pareto, en este caso  $p1$ ;

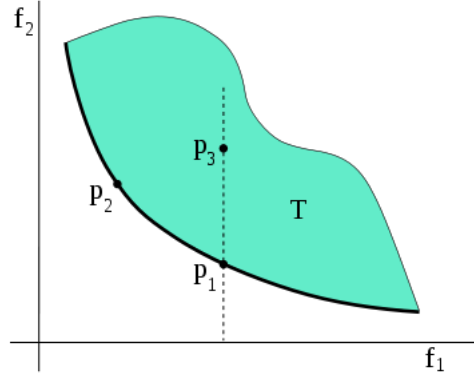


Figura 2.1: Frente de Pareto de una función con dos objetivos

dicho punto de corte siempre tendrá el mismo valor de  $f_1$  y un valor mejor de  $f_2$ . También se puede observar que para 2 puntos cualesquiera del Frente de Pareto, nunca habrá uno que mejore de forma simultánea los dos objetivos respecto al otro punto.

### 2.3.2. Algoritmos Evolutivos Multiobjetivos

Los algoritmos evolutivos multiobjetivo (*Multi-Objective Evolutionary Algorithms*, MOEA) se han convertido en una técnica muy popular para resolver problemas multiobjetivo. Los MOEAs son muy atractivos para resolver problemas multiobjetivo debido a que tratan de forma simultánea un conjunto de posibles soluciones lo cual permite obtener una aproximación del frente de Pareto en la ejecución de un único algoritmo. De esta manera, usando MOEAs el DM no necesita realizar una serie de optimizaciones para cada objetivo como normalmente se hace en los métodos de Investigación de Operaciones [6]. Sin embargo, una limitante en los MOEAs es el hecho de que sólo involucra el proceso de búsqueda de la solución sin considerar un aspecto más importante, el **proceso de decisión**. La mayoría de los enfoques actuales de los MOEAs se concentran en encontrar una aproximación al conjunto óptimo del frente de Pareto, sin embargo, la búsqueda de dicho conjunto no resuelve el problema. El DM todavía tiene que elegir, del

conjunto, la mejor solución. Esta no es una tarea muy difícil cuando se trata de problemas con dos o tres objetivos. Sin embargo, cuándo el número de objetivos aumenta, se plantean dos dificultades importantes [13]:

1. La capacidad de los algoritmos para encontrar el frente de Pareto se degrada rápidamente.
2. Se hace más difícil o incluso imposible para el DM establecer juicios válidos para comparar soluciones con objetivos en conflicto.

### 2.3.3. Modelo de Optimización Entera

Algunos modelos se han propuesto para resolver el SPP basados en ordenamiento [15] [12] [14]. En estos modelos se ha utilizado la lógica difusa dada su flexibilidad para comparar dos proyectos o carteras por medio de predicados, tal como “El proyecto A es mejor que el proyecto B”, de acuerdo a las preferencias que tiene el DM evitando la dificultad de evaluaciones numéricas.

Fernández *et al* [13] proponen un modelo para SPP en el cual caracteriza *a priori* las preferencias del *Tomador de Decisiones* utilizando el concepto de **relación de dominancia difusa** para discernir entre un conjunto de soluciones cuales son las mejores, es decir, buscar un subconjunto de soluciones que ninguna otra solución sea mejor que ellas. Las comparaciones entre las diferentes alternativas se realizan en base a los métodos ELECTRE. El subconjunto de las mejores soluciones es solo una zona del frente de Pareto que corresponde con las mejores soluciones de acuerdo a las preferencias del *Tomador de Decisiones*. Esta **relación de dominancia difusa** sustituye al mecanismo de posición dominante clásicos de los Algoritmos Evolutivos Multiobjetivo (*Multi-Objective Evolutionary Algorithms*, MOEA) y permite dirigir la búsqueda al subconjunto de soluciones que concuerdan con las preferencias del *Tomador de Decisiones* evitando hacer una búsqueda exhaustiva en todo el frente de Pareto. En lo sucesivo al modelo propuesto por Fernández *et al* se le denominará PORTADOR ( Optimización de cartera de proyectos publicos basado en la descripción multi-atributos de los

proyectos, *Public pORtfolio opTimizAtion baseD on Multi-attribute prOject descRiption*).

### Relación de Dominancia Difusas

Dado  $G$  como un conjunto de funciones objetivos de un problema de optimización multicriterio y  $O$  como su espacio de solución, un elemento  $x \in O$  es un vector  $(x_1, \dots, x_n)$ , donde  $x_i$  es el  $i$ -ésimo valor del objetivo. Supone que por cada criterio  $j$  existe una relación de preferencia o indiferencia  $(P_j, I_j)$  de dicho criterio. Esto es,  $\forall (x_j, y_j) \in G \times G$  se cumple una y solo una de las siguientes sentencias [13]:

- $x_j P_j y_j$
- $y_j P_j x_j$
- $x_j I_j y_j$

En PORTADOR, se supone que  $x_j P_j y_j \Rightarrow x_j > y_j$ . Así, se puede establecer la siguiente premisa: para cada  $(\mathbf{x}, \mathbf{y}) \in O \times O$ , el DM puede crear un predicado difuso que modele el grado de verdad del siguiente enunciado “ $\mathbf{x}$  es al menos tan bueno como  $\mathbf{y}$  de acuerdo al punto de vista del DM”. Existen muchos métodos para crear este predicado, PORTADOR utiliza un método basado en los métodos ELECTRE [13].

### ELECTRE

Uno de los métodos más conocidos en el análisis de la decisión con múltiples criterios es ELECTRE (*ELimination and Choice Expressing REality, por sus siglas en ingles*). Estos métodos eligen una o varias alternativas diferentes. En el caso del problema de Cartera de Proyectos Sociales dichas alternativas son los proyectos a desarrollar, de entre un conjunto de alternativas no dominadas, es decir que no existan otras

alternativa mejores, o también pueden ordenar dichas alternativas de mejor a peor basados en ciertas creencias o preferencias del DM [29].

El método ELECTRE I fue desarrollado inicialmente por Benayoun (1966) y mejorado por Roy (1971). El método define un subconjunto de las mejores soluciones a partir del conjunto inicial de soluciones no dominadas. El DM debe escoger finalmente de ese conjunto cuál es la solución por implementar. La definición de este conjunto se hace a partir de una relación binaria  $P$  denominada relación de sobreclasificación, *que es la comparación de dos alternativas respecto a todos los criterios mediante el uso de relaciones binarias* [29].

### Método Basado en ELECTRE para PORTADOR

A continuación se explica de forma breve el método modificado de ELECTRE que se utiliza para obtener la relación de dominancia difusa en PORTADOR. La descripción se basa en [13] con apoyo de [7] y [29].

En PORTADOR se dice que una alternativa  $x$  tiene una relación de sobreclasificación sobre una alternativa  $y$  ( $xSy$ ) si el DM tiene mucha satisfacción al escoger  $x$  y no  $y$  y poca insatisfacción al escoger  $x$  por encima de  $y$ . La relación de sobreclasificación se establece por medio de los índices de concordancia y discordancia. El índice de concordancia  $c(x, y)$  es la medida ponderada del número de criterios para los cuales  $x$  es preferida a  $y$ . El índice de discordancia  $d(x, y)$  representa la insatisfacción que el DM experimenta cuando se elige la alternativa  $x$  y no la  $y$ . Por lo tanto, el predicado  $xSy$  ( $x$  supera a  $y$  ó  $x$  es al menos tan bueno como  $y$ ) puede ser expresado por la equivalencia lógica de la Ecuación 2.2.

$$xSy \Leftrightarrow C(x, y) \wedge \sim D(x, y) \quad (2.2)$$

El grado de verdad del predicado  $xSy$  es calculado por la Ecuación 2.3

$$\sigma(x, y) = c(x, y) \cdot N(d(x, y)) \quad (2.3)$$

donde  $N(d(x, y))$  denota el grado de verdad de la negación del índice de discordancia.

El método exige que el DM asigne para cada objetivo un factor de ponderación asociado a las preferencias relativas respecto a los otros objetivos. Asumiendo que  $G = 1, \dots, n$  representa el conjunto de los  $n$  criterios y  $w_i, i = 1, \dots, n$  representa el conjunto de factores de ponderación asociados con los  $n$  criterios.

Por ejemplo, se tiene un problema de cartera de proyectos en donde existen cuatro posibles proyectos y se deben maximizar dos objetivos, los datos se muestran en la Tabla 2.1.

<i>Proyecto</i>	$o_1$	$o_2$
1	75	10
2	25	5
3	100	10
4	25	10

Tabla 2.1: Datos de los proyectos disponibles

De esos cuatro proyectos se han obtenido tres posibles soluciones al problema cada una con dos objetivos a maximizar. En la Tabla 2.2 se muestran las tres posibles alternativas, por ejemplo, en la alternativa A están incluida los proyectos 1 y 2, en la alternativa B los proyectos 1, 3 y 4 y en la alternativa C los proyectos 1 y 4. El peso de cada objetivo es  $W = (0.6, 0.4)$ .

<i>Cartera</i>	$o_1$	$o_2$	PC
A	100	15	1,2
B	200	30	1,3,4
C	100	20	1,4

PC: Proyectos que componen a la alternativa

Tabla 2.2: Alternativas de solución al problema de cartera

Así, el índice de concordancia se calcula como en la Ecuación 2.4

$$c(x, y) = \sum_{j \in C_{x,y}} w_j \quad (2.4)$$



donde  $C_{x,y} = \{j \in G \text{ tal que } x_j P_j y_j \vee x_j I_j y_j\}$  denota los proyectos en donde  $x$  es preferible a  $y$ ;  $w_j$  denotan el peso ponderado del objetivo  $j$ .

Para el ejemplo, la matriz de concordancia se obtiene con la Ecuación 2.4 y se muestra en la Tabla 2.3. El término  $c(1, 3)$  resulta del cálculo de  $c(1, 3) = 0.6/2 + 0 = 0.3$ . Como en A y C el valor del objetivo 1 vale 100, el peso de este objetivo se divide entre 2.

	A	B	C
A	-	0	0.3
B	1	-	1
C	0.7	0	-

Tabla 2.3: Matriz de concordancia  $C_{x,y}$

Sea  $D_{x,y} = \{j \in G \text{ tal que } y_j P_j x_j\}$  la coalición de discordancia con  $xSy$ . El índice de discordancia es medido en comparación entre el umbral de veto  $v_j$ , el cual es la máxima diferencia de  $y_j - x_j$  compatible con  $\sigma(x, y) > 0$ . El índice de discordancia se calcula como en la Ecuación 2.5

$$N(d(x, y)) = \min_{j \in D_{x,y}} [1 - d_j(x, y)] \quad (2.5)$$

$$d_j(x, y) = \begin{cases} 1 & \text{si } \nabla_j \geq v_j \\ (\nabla_j - u_j)/(v_j - u_j) & \text{si } u_j \leq \nabla_j \leq v_j \\ 0 & \text{si } \nabla_j \leq u_j \end{cases} \quad (2.6)$$

donde  $\nabla_j = y_j - x_j$  y  $u_j$  es el umbral de discordancia.

Siguiendo con el ejemplo, para el cálculo de las matrices de discordancia se toman los valores para el umbral de veto  $U = (10, 2)$  y el umbral de discordancia  $V = (50, 5)$ . Posteriormente se generaran dos matrices  $d_1$  y  $d_2$  que se muestran en las Tablas 2.4 y 2.5 respectivamente. Por ejemplo, para el término  $d_1(A, B)$ , como  $\nabla_j = y_j - x_j = 200 - 100 = 100$  es mayor que  $v_1$  el valor de  $d_1(A, B) = 1$ , así para cada objetivo y para cada alternativa.

	A	B	C
A	-	1	0
B	0	-	0
C	0	1	-

Tabla 2.4: Matriz de discordancia  $d_1$ 

	A	B	C
A	-	1	1
B	0	-	0
C	0	1	-

Tabla 2.5: Matriz de discordancia  $d_2$ 

Al tener  $d_1$  y  $d_2$  se puede calcular  $N(d(x, y))$  y se muestra en la Tabla 2.6.

	A	B	C
A	-	0	0
B	1	-	1
C	1	0	-

Tabla 2.6: Matriz de discordancia  $N(d(x, y))$ 

Al obtener el grado de verdad del predicado  $xSy$  se tiene suficiente información para realizar una clasificación completa de las alternativas. Cada término de la matriz del grado de verdad  $\sigma(x, y)$  es equivalente al término de la matriz de concordancia  $c(i, j)$  afectado en virtud de los “desacuerdos”, respecto a algunos criterios, con el predicado “la alternativa  $x$  es al menos tan buena como la alternativa  $j$ ”.

En el ejemplo, al tener  $c(x, y)$  y  $N(d(x, y))$  es posible calcular  $\sigma(x, y)$ , el resultado se muestra en la Tabla 6. Para el elemento  $\sigma(C, A) = c(C, A) \cdot N(d(C, A)) = 0.7 * 1 = 0.7$ .

Para realizar la clasificación completa, debe determinarse un “valor de credibilidad”  $\lambda$  tal que,  $\sigma(x, y) \geq \lambda$  define una relación estricta de dominancia  $xSy$ . Con  $\lambda = 0.75$  se obtienen declaraciones creíbles de dominancia, es decir, *dominancias fuertes* y con  $\lambda = 0.67$  se obtiene una relación de domi-

	A	B	C
A	-	0	0
B	1	-	1
C	0.7	0	-

Tabla 2.7: Matriz  $\sigma$  del ejemplo

nancia débil.  $\sigma(x, y) > \approx 0.5$  definen una dominancia dudosa,  $\sigma(x, y) < 0.5$  significa que definitivamente no es dominada. Esto es:

$xSy \wedge \sim ySx \Leftrightarrow \sigma(x, y) > \lambda \wedge \sigma(y, x) < \lambda \Rightarrow$  una presunta preferencia hacia  $x$ .

En el ejemplo los valores de  $\sigma(B, A) = 1, \sigma(B, C) = 1, \sigma(C, A) = 0.7$  quedan por encima de  $\lambda = 0.65$ , por lo tanto se puede decir que existen solo dominancias fuertes.

En PORTADOR se asume la existencia de un umbral  $\beta > 0$  tal que si  $\sigma(x, y) \geq \lambda$  y también  $\sigma(y, x) \leq (\sigma(x, y) - \beta)$ , entonces está es una relación de preferencias asimétricas que favorecen a  $x$  que será denotada por  $xP(\lambda, \beta)y$ . Al tener soluciones muy parecidas es conveniente utilizar un umbral  $\beta > 0$  tal que si  $\sigma(x, y) \geq \lambda$  y que  $\sigma(y, x) \leq (\sigma(x, y) - \beta)$ , entonces hay una relación a favor de la preferencia que favorece a  $x$  y es denotada por  $xP(\lambda, \beta)y$ . Por lo tanto, para algunos valores de  $\lambda$  y  $\beta$  la condición definida  $xP(\lambda, \beta)y$  es un buen argumento para justificar una relación de preferencias estrictas. La **Relación de Dominancia Difusa** de este modelo,  $xP(\lambda, \beta)y$ , es:

- (i)  $x$  domina a  $y$
- (ii)  $\sigma(x, y) \geq \lambda_s \wedge \sigma(y, x) < \lambda_i; \lambda_s = 0.67, \lambda_i = 0.5$
- (iii)  $\sigma(x, y) \geq \lambda_s \wedge (\lambda_i \leq \sigma(y, x) < \lambda_s) \wedge (\sigma(x, y) - \sigma(y, x)) \geq \delta; \lambda_s = 0.67, \lambda_i = 0.5$

$\delta$  es un parámetro para la dominancia estricta el cual depende del

número de criterios. Para la consistencia de ii y iii,  $\delta$  debe ser mayor que  $(0.67 - 0.5) = 0.17$ .

## 2.4. Hiper-heurísticos

Los hiper-heurísticos son heurísticas (simples o de alto nivel) las cuales, dado un problema particular y un número de Heurísticas de Bajo Nivel que actúan sobre el problema, seleccionan y aplican la Heurística de Bajo Nivel más apropiada en cada punto de decisión. Mientras que las metaheurísticas usualmente (pero no necesariamente) trabaja directamente con la solución del problema, un hiper-heurístico trabaja con métodos de solución del problema (típicamente heurísticas simples). Las metaheurísticas usualmente modifican la solución directamente mientras que los hiper-heurísticos modifican la solución de manera indirecta usando las Heurísticas de Bajo Nivel a sus disposición. Las hiper-heurísticas cuentan con un alto nivel de abstracción y generalidad que la mayoría de las metaheurísticas actuales. La hiper-heurística puede ser tanto heurísticas de búsqueda simple como metaheurísticas más complejas como son los Algoritmos Genéticos [30].

El término hiper-heurístico fue acuñado por primera vez en 1997 por Jörg Denzinger, Matthias Fuchs y Marc Fuchs [9]. Ellos usaron el término para describir un protocolo que selecciona y combina varios métodos de Inteligencia Artificial. Tiempo después en el año 2000, Cowling y Soubeiga usaron el término de hiper-heurístico para describir la idea de "heurísticas que seleccionan heurísticas" en el contexto de optimización combinatoria [8].

La idea detrás de los hiper-heurísticos no es nueva. Fue a inicios de los 60's cuando Fisher y Thompson creyeron y probaron que usando aprendizaje probabilístico combinado con una serie de reglas de programación de horarios era mejor que seleccionar cualquiera de las reglas por separado. Aunque no se utilizó el término, este fue el primer hiper-heurístico [16] [17].

### 2.4.1. Tipos de Hiper-heurísticos

Los hiper-heurísticos pueden ser clasificados en dos grandes categorías, como se muestra en la Figura 2.2. En la primera categoría están las *heurísticas que seleccionan heurísticas*, este tipo de hiper-heurísticos cuenta con un conjunto existente de heurísticas que resuelven, completamente o de manera parcial, el problema central. El objetivo es descubrir la secuencia de como aplicar dichas heurísticas para resolver de manera eficiente el problema.

En la segunda categoría están las *heurísticas que generan heurísticas*. En esta clase de hiper-heurísticos también cuenta con un conjunto de heurísticas constructivas para el problema central. Primero se inicia con una solución vacía y de manera inteligente se van seleccionando heurísticas constructivas para ir mejorando gradualmente la solución. Este proceso continua hasta que se tiene una solución completa al problema central.

$$\text{Hiper-heurísticos} = \begin{cases} \text{Selección} & \text{heurísticas que seleccionan heurísticas} \\ \text{Generación} & \text{heurísticas que generan heurísticas} \end{cases}$$

Figura 2.2: Clasificación de los hiper-heurístico

### 2.4.2. Hiper-heurísticos de Selección

Las hiper-heurísticas de selección son procedimientos de búsqueda de muy alto nivel que se sitúan por encima de las metaheurísticas debido a que elevan la generalidad que tienen las metaheurísticas permitiendo crear algoritmos más robustos. Las hiper-heurísticas se encargan de seleccionar, en cada momento y para cada problema, el método heurístico o meta-heurístico más adecuado a aplicar. Esta selección se hace, por ejemplo, en función de la calidad de la solución que consigue cada heurística o meta-heurística o en función del tiempo que tarda cada una de ellas. [10][2].

La idea básica que reside detrás de las hiper-heurísticas consiste en utilizar una población de métodos heurísticos o meta-heurísticos (denominados *heurísticas de bajo nivel*, HBN) que actúan sobre un problema determinado, de tal forma que en cada momento la hiper-heurística (denominada *heurística*

*de alto nivel*, HAN) decide cuál es el mejor método para aplicar al problema, cambiando el estado de dicho problema. Los métodos heurísticos son procedimientos simples y basados en el sentido común que permiten obtener una solución de calidad (no necesariamente óptima) en un tiempo razonable [31].

A continuación, se presenta la plantilla general de un hiper-heurístico [3]:

1. Construir un conjunto  $H$  de heurísticas/meta-heurísticas, las cuales se pueden aplicar a un problema concreto.
2. Inicializar el problema a un estado conocido  $S_0$ .
3. Aplicar al estado actual  $S_i$  del problema el algoritmo de  $H$  más adecuado, haciendo pasar el problema al estado  $S_j$ .
4. Si el problema está resuelto, parar. En otro caso ir a 3.

En el fondo, el hiper-heurístico planifica una serie de acciones que se ejecutan sobre un determinado problema, realimentándose con los resultados obtenidos.

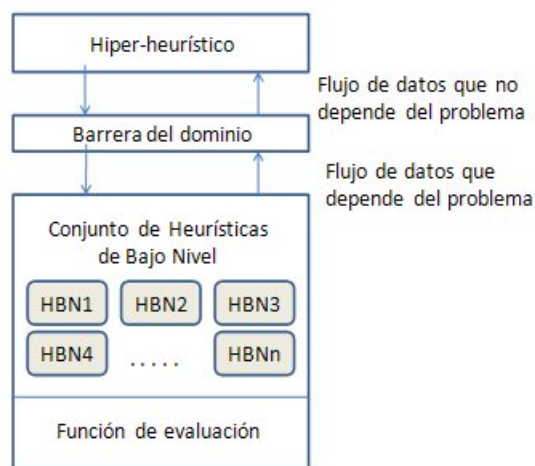


Figura 2.3: Esquema general de un hiper-heurístico

En la Figura 2.3 se presenta el esquema general de un hiper-heurístico, se puede observar que existe una barrera entre la hiper-heurística y el conjunto de heurísticas/metaheurísticas. En este marco, el hiper-heurístico preguntaría cómo haría cada algoritmo su trabajo, de tal forma que pueda decidir que algoritmo (o conjunto) aplicar en cada momento[10].

En la Figura 2.4 se ilustra la situación en la que operan los hiper-heurísticos. El espacio de búsqueda de los hiper-heurísticos esta en las HBN, es decir, encontrar la serie de HBN que genere las mejores soluciones por medio de una Lista de HBN (LHBN). El espacio de búsqueda de las HBN corresponde a las soluciones del problema que se quiere resolver, es decir, al espacio de las variables de decisión  $\Omega$  que va a estar en función de las variables  $x(x_1, x_2, \dots, x_n)$ . Dada las variables de decisión y una función objetivo  $F$  que se desee optimizar se obtendrán los valores de las funciones objetivos  $y(y_1, y_2, \dots, y_k)$  del espacio de las funciones objetivo  $\Lambda$ .

Para aplicar el hiper-heurístico a un problema distinto, lo único que habrá que hacer será cambiar el conjunto de HBN, dado que éstos son dependientes del problema, también sería necesario cambiar la función de evaluación que indica la calidad de una solución, dado que ésta también es dependiente

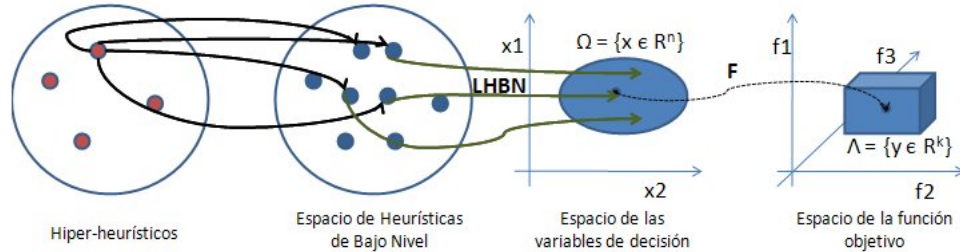


Figura 2.4: Espacios en que operan los hiper-heurísticos

## 2.5. Métricas de Desempeño de Algoritmos

El desempeño de un algoritmo basado en heurísticas se determina por la eficiencia y la efectividad mostradas por éste al resolver diferentes

instancias de un problema. La efectividad de un algoritmo se refiere a la calidad de la solución encontrada o a su confiabilidad en la tarea de encontrar soluciones adecuadas y es altamente dependiente de la estructura del problema. La eficiencia por otra parte, caracteriza el comportamiento del algoritmo en tiempo de ejecución (por ejemplo el tiempo computacional y los requerimientos de memoria) y está muy relacionada con el conocimiento que se tenga del dominio y la complejidad del problema [27].

La caracterización del problema y del proceso de solución es una parte esencial en el análisis de desempeño de algoritmos, y permite identificar cuáles son los factores que influyen en el comportamiento algorítmico. Un análisis de desempeño de calidad requiere la definición de métricas adecuadas que cuantifiquen las características indicadoras del desempeño[28]..

Al detectar los factores que influyen en el comportamiento de los algoritmos y proponer métricas que midan esos factores se podría conocer mejor el comportamiento de los algoritmos y así poder detectar comportamientos que no son deseados y que no permiten que el algoritmo sea eficiente. El análisis de estas métricas permitiría mejorar los algoritmos actuales. Las métricas propuestas para conocer el comportamiento de los Hiper-heurísticos se describen en el Capítulo 4 en la sección 4.3.1.



# Capítulo 3

## Estado del Arte

El aporte más importante de este trabajo es el uso de nuevas técnicas de algoritmos heurísticos denominadas hiper-heurísticos para resolver problemas complejos tal como el de la Mochila y de Cartera de Proyectos. Dado que el uso de estas técnicas es nuevo, no existen trabajos que aborden dichos problemas usando hiper-heurísticos de selección. Por lo tanto, en la Sección 3.1 se describen los trabajos más importantes del problema de la Mochila incluyendo una aproximación usando un hiper-heurístico de generación y en la Sección 3.2 se describe los trabajos relacionados con el Problema de Carteras incluyendo un algoritmo determinista. En la Sección 3.3 se dan algunas conclusiones para los problemas mencionados incluyendo una breve reseña de los trabajos de hiper-heurísticos de selección más importantes en donde es claro ver que no existen trabajos para problemas de la familia del Problema de la Mochila.

### 3.1. Problema de la Mochila

#### 3.1.1. Algoritmo Determinista

David Pisinger [25] propone un algoritmo determinista que tiene la propiedad de obtener el núcleo mínimo del KP y además aplica técnicas que mejoran el tiempo para el ordenamiento necesario en la reducción del problema al núcleo mínimo. El algoritmo es basado en el enfoque de programación dinámica donde se inicia con un tamaño del núcleo mínimo y éste es extendido conforme es necesario. Según Pisinger, este algoritmo es el mejor para resolver el KP hasta ese momento (1994).

El primer paso del algoritmo es encontrar el objeto de cambio  $b$ . El proceso consiste en ir seleccionando repetidamente objetos que entren en el núcleo del KP, representados por  $s-1$  o  $t+1$  (al inicio,  $s$  y  $t$  se igualan a  $b$ ), para obtener un conjunto  $X_{s,t}$  que representan los objetos en el núcleo y sus estados (0 para los objetos que no están en la mochila o 1 en caso contrario).

Para seleccionar que objetos de  $s-1$  y  $t+1$  entran en  $X_{s,t}$  se utilizan algunas técnicas de límites y relajación lineal. Al contar con el núcleo  $[s, t]$  y el conjunto  $X_{s,t}$  el problema queda resuelto si:

1. Si  $X_{s,t} = \emptyset$ , esto es, la solución óptima es la solución de cambio y no existe un núcleo del problema.
2. Si todos los objetos  $j \in [1, s-1]$  están en la mochila ( $x_j = 1$ ) y todos los objetos  $j \in [t+1, n]$  no están en la mochila ( $x_j = 0$ ).

#### 3.1.2. Algoritmo Hiperheurístico

En el trabajo “Evolution of Hyperheuristics for the Biobjective 0/1 Knapsack Problem by Multiobjective Genetic Programming” [19], Kumar pretende mostrar que un Sistema de Programación Genética, trabajando como un hiper-heurístico, puede desarrollar un conjunto de heurísticas que

puedan dar la solución del frente de Pareto para el KP y en general para cualquier problema de optimización combinatoria multiobjetivo.

El Sistema de Programación Genética propuesto por Kumar desarrolla una heurística la cual decide, usando otras heurísticas, si agrega o no objetos a la mochila en caso de que la solución final sea una solución óptima del frente de Pareto. Los resultados indican que el frente de Pareto obtenido por el Sistema de Programación Genética es comparable con otras heurísticas desarrolladas.

Los individuos usados en la población son arboles que consisten de las funciones aritméticas básicas (suma, resta, multiplicación y división) y terminales. La población inicial es generada de manera aleatoria y consta de 5 individuos. El algoritmo mantiene siempre la mejor solución obtenida hasta el momento en la población actual y reemplaza los dos peores individuos por otros dos mejores encontrados. La salida es un árbol de funciones aritméticas que permite determinar cuáles objetos deben entrar(+) o salir (-) de la mochila para encontrar la solución óptima (o más cercana a la óptima) del problema.

## 3.2. Problema de Cartera de Proyectos

### 3.2.1. NOSGA

Para dar solución al SPP formulado en el modelo PORTADOR, Fernández *et al* utilizan una variación del muy conocido algoritmo NSGA-II, denominado NOSGA. El principal cambio realizado al algoritmo fue la incorporación de un tipo de relación de dominancia difusa con la finalidad de obtener las mejores soluciones de acuerdo a las preferencias del DM y no solo el frente de Pareto como lo hace el NSGA-II. Esta relación se describe detalladamente en la Sección 2.3.3.

La relación de dominancia difusa  $yP(\lambda, \beta)x$  se traduce en el algoritmo como el número de soluciones preferidas con respecto a una solución dada. Sea  $A$  un subconjunto del espacio de soluciones de  $O$ , si no existe una solución

$y \in A$  tal que  $yP(\lambda, \beta)x$ , se puede decir que  $x$  es una solución no dominada en  $A$ , estrictamente dentro de umbrales de credibilidad dados por  $\lambda y \beta$ . El conjunto de soluciones no dominadas en  $O$  es un subconjunto del frente de Pareto. Para una solución  $x \in O$ , el conjunto que lo domina es definido como  $S_0 = \{y \in O \text{ tal que } yP(\lambda, \beta)x\}$ . La cardinalidad de este conjunto se denota por  $card(S_0)$ , esta es una función entera que depende de  $x$ .

La mejor solución del problema de optimización multiobjetivo debe ser una solución no dominada de  $O$ , donde cada solución es estrictamente dominada, la mejor solución debe estar en el conjunto  $x$  con  $card(S_0)$  mínimo.

Continuando con el ejemplo mostrado en el Capítulo 2, en la Tabla 3.1 se muestra el número de soluciones a las que domina y que son dominadas estrictamente por cada solución. Se puede observar que la alternativa  $B$  domina a las soluciones  $A$  y  $C$  y está no es dominada por ninguna; la solución  $C$  domina a la solución  $A$  pero es dominada por la solución  $B$ . La solución  $A$  no domina a ninguna otra solución y es dominada por  $B$  y  $C$ . Por lo tanto, en orden ascendente quedaría primero la solución  $B$ , después la solución  $C$  y por ultimo la solución  $A$ .

	A	B	C	domina a
A	-	0	0	0
B	1	-	1	2
C	0.7	0	-	1
$card(S_0)$	2	0	1	

Tabla 3.1: Matriz  $\sigma$  y el número de alternativas a las que dominan y que son dominadas ( $card(S_0)$ )

A diferencia de NSGA-II, NOSGA no está interesado en encontrar un frente de Pareto uniforme, sino que busca un frente de Pareto consistente con las preferencias impuestas por el DM. El pseudo-código de NOSGA se muestra en el Algoritmo 1.

Cabe señalar que en las pruebas experimentales del algoritmo NOSGA superó ampliamente al NSGA-II para el SPP. Las principales diferencias entre NSGA-II y NOSGA son:

---

**Algoritmo 1** NOSGA (K, Numero\_de\_generaciones)

---

```

1: Inicializar la Población  $P$ 
2: Generar de manera aleatoria una población inicial de  $K$  individuos
3: Evaluar los valores de los objetivos
4: Evaluar  $\sigma$  en  $P \times P$ 
5: para todo individuo  $x \in P$  hacer
6:   Calcular  $card(S_0)$ 
7: fin para
8: Generar frentes de soluciones con valores iguales de  $card(S_0)$ 
9: Asignar a los frentes una categoría basada en  $card(S_0)$ 
10: Generar la siguiente población  $Q$  de  $K$  individuos
11:   Seleccionar los padres por torneo
12:   Cruzar y mutar a los individuos
13: para  $i = 1$  hasta Numero_de_generaciones hacer
14:   Asignar  $P' = P \cup Q$ 
15:   Evaluar  $\sigma$  en  $P' \times P'$ 
16:   para todo padre e hijo  $\in P$  hacer
17:     Calcular  $card(S_0)$ 
18:     Asignar a los frentes una categoría basada en  $card(S_0)$ 
19:     Agregar soluciones a la siguiente generación hasta  $k$  individuos
        encontrados
20:   fin para
21:   Remplazar  $P$  por los  $k$  individuos encontrados
22:   Generar la siguiente población  $Q$  de  $K$  individuos
23:     Seleccionar los padres por torneo
24:     Cruzar y mutar a los individuos
25: fin para

```

---

1. El uso de  $\sigma$  en NOSGA como la Aptitud de los individuos.
2. El ordenamiento basado en dominancia del frente de Pareto es sustituido por la dominancia estricta.

### 3.2.2. NSGA-II

El NSGA-II (*Nondominated Sorting Genetic Algorithm-II*) es uno de los algoritmos más populares para resolver problemas multiobjetivo debido a su simplicidad y su efectividad. El algoritmo primero genera una población competitiva de individuos que después es ordenada de acuerdo al nivel de dominancia que tenga el individuo en la población. Este nivel de dominancia genera diferentes frentes, el primer frente con las soluciones no dominadas.

Las soluciones de este primer frente pasan a la siguiente generación (elitismo) junto con otras soluciones de tal manera que exista diversidad. Como todo Algoritmo Genético, se le aplican operadores evolutivos (cruza y mutación, entre otros). Al finalizar, las soluciones no dominadas de la ultima generación serán una aproximación al frente de Pareto. Este algoritmo es de los más usados para resolver problemas multiobjetivo ha servido de base para el diseño de muchos otros. [4] [20]

### 3.2.3. Algoritmo Evolutivo Multiobjetivo

En el trabajo presentado por Madaglia[22], “Multiobjective evolutionary approach for linearly constrained project selection under uncertainty”, se propone un método evolutivo para el problema de selección de proyectos de una cartera que tengan múltiples criterios, independencia entre los proyectos y restricciones lineales de los recursos. El método esta basado en las preferencias que el *Tomadores de Decisiones* tiene sobre los proyectos y es capaz de generar múltiples soluciones que se acerquen al frente de Pareto.

El algoritmo propuesto esta basado en el NSGA-II pero con algunas mejoras tales como:

1. *Estrategias de elitismo*. Estas estrategias permiten conservar los mejores individuos de la población.
2. *Parámetros de diversificación*. Estos parámetros son útiles para extender la búsqueda y evitar estancamientos en óptimos locales.
3. *Mecanismo rápido de dominancia estricta*. Este mecanismo toma en cuenta las preferencias impuestas por el DM para una rápida comparación entre soluciones.
4. *Eficiente mecanismo de manejo de las restricciones*. Este mecanismo permite descartar rápidamente soluciones infactibles.

Medaglia reporta que este método se comparó con el método de investigación del espacio de parámetros estocásticos (PSI) usando para ello una cartera de proyectos de I&D con incertidumbre evaluados con simulación Montecarlo. Los resultados fueron que este método es más rápido y robusto y provee soluciones de alta calidad.

### 3.3. Conclusiones

#### 3.3.1. Mochila y Cartera de Proyectos Sociales

La evaluación de los proyectos sociales la realiza un *Tomador de Decisiones* que los valora de acuerdo a su experiencia y preferencias. La principal forma de abordar el problema de cartera de proyectos sociales a sido con Algoritmos Genéticos que dan como resultado los proyectos seleccionados y el monto a invertir. Las tendencias actuales muestran que se empiezan a utilizar las hiper-heurísticas para resolver problemas complejos como KP o SPP. En la Tabla 3.2 se muestra una comparativa entre los diferentes trabajos del estado del arte.

Autor/ Año	Problema	Nivel de comparación	Evaluación	Multi objetivo	Algoritmo de solución
Fernández <i>et al</i> 2010	SPP	Carteras	Difusa	✓	Genético (NOSGA)
Medaglia <i>et al</i> 2007	R&DPP	Proyectos	Cuantitativa	✓	Genético (NSGA-II)
Pisinger 1994	Mochila	Objetos	Cuantitativa	NA	Determinista
Kumar <i>et al</i> 2008	Mochila	Objetos	cuantitativa	✓	Hiper-heurístico
Esta tesis	SPP	Carteras	Difusa	✓	Hiper-heurístico
	Mochila	Objetos	Cuantitativa		

Tabla 3.2: Comparativa de los diferentes trabajos del estado del arte

### 3.3.2. Hiper-heurísticas

Los trabajos más importantes en el ámbito de los hiper-heurísticos de selección se muestran en la Tabla 3.3. Cowling *et al*, para el problema de Programación de Personal, propone una selección aleatoria de las Heurísticas de Bajo Nivel (HBN) en cada paso. Por otro lado, Nareyek (2003) para el problema de Planeación propone usar un mecanismo de Aprendizaje Reforzado como método de selección de las HBN; dando o quitando puntos a cada una de ellas según el valor de la solución generada.

Burke (2003) propone usar, para el problema de Programación de Horarios, Aprendizaje Reforzado con una lista Tabú. El algoritmo podrá seleccionar de la lista Tabú las HBN disponibles. La lista tabú podrá bloquear HBN por algunas iteraciones si así lo cree conveniente. Ayob y Kendall (2003) utilizan el método Montecarlo para realizar la selección de las HBN. Bai y Kendall (2005), para el problema de Auto Asignación de Espacio, utilizan el algoritmo de Recocido Simulado como Heurística de Alto Nivel para seleccionar las HBN [2].

Es importante destacar que, después de una búsqueda exhaustiva, no se encontraron trabajos de hiper-heurísticos para problemas de Cartera de Proyectos o inclusive para la familia de problemas de la Mochila. Por lo que en



Problema	Autor(es)/Año
Asignación de Canales ( <i>Channel Assignment</i> )	Kendall y Mohamad (2004)
Colocación de Componentes ( <i>Component Placement</i> )	Ayob y Kendall (2003)
Programación de Personal ( <i>Personnel Scheduling</i> )	Cowling <i>et al</i> (2000), Cowling y Chakhlevitch (2003), Han y Kendall (2003), Burke <i>et al</i> (2003), Bai <i>et al</i> (2007)
Empacado ( <i>Packing</i> )	Dowsland <i>et al</i> (2007), Bai <i>et al</i> (2007)
Planeación ( <i>Planning</i> )	Nareyek (2003)
Auto Asignación de espacios ( <i>Self Space Allocation</i> )	Bai y Kendall (2005), Bai <i>et al</i> (2008)
Programación de Horarios ( <i>Timetabling</i> )	Burke <i>et al</i> (2003, 2005), Bilgin <i>et al</i> (2006), Chen <i>et al</i> (2007), Bai <i>et al</i> (2007)
Problema de Ruteo de Vehículos ( <i>Vehicle Routing Problems</i> )	Pisinger y Ropke (2007)

Tabla 3.3: Principales trabajos de hiper-heurísticos de selección

este trabajo se demuestra que la utilización de algoritmos hiper-heurísticos no esta restringida a problemas clásicos, sino que pueden ser usados para resolver nuevos problemas complejos, como lo es el Problema de la Mochila y el Problema de Cartera de Proyectos Sociales, dando excelentes resultados.

En ninguno de los trabajos anteriores se ha intentado explicar el comportamiento de un hiper-heurístico por medio de métricas. En este trabajo se formulan dos métricas que permiten conocer como se comportan las HBN en el transcurso de la ejecución del hiper-heurístico. Este conocimiento permitió realizar mejoras al algoritmo propuesto.



# Capítulo 4

## Propuesta de Solución

En este Capítulo se describe con detalle el diseño e implementación del Hiper-heurístico para el KP y su posterior adaptación para resolver el SPP formulado en PORTADOR.

El hiper-heurístico propuesto está basado en un algoritmo genético básico. La motivación para seleccionar algoritmos genéticos como la heurística de alto nivel se basa en que la estructura de éstos se adapta a la perfección al concepto de hiper-heurísticas de selección. Para resolver PORTADOR es necesario contar con cierto número de soluciones que al compararse entre sí es posible obtener un frente de soluciones no dominadas. Al ser los algoritmos genéticos heurísticas poblacionales, es decir, que buscan  $k$  soluciones simultáneamente por generación, siendo  $k$  el tamaño de la población, facilitan la obtención del frente buscado.

Además, dada la característica de los algoritmos genéticos de evolucionar a mejores soluciones, ésta también permite que gradualmente el frente de soluciones se mueva hacia el frente de Pareto para encontrar las soluciones óptimas. A su vez, los individuos del algoritmo genético permiten que en lugar de ir seleccionando una sola heurística de bajo nivel a la vez, se puedan seleccionar  $m$  número de heurísticas de bajo nivel, siendo  $m$  el tamaño del individuo, permitiendo generar suficientes soluciones diferentes por cada generación.

Cabe destacar que la estrategia usada por otros investigadores para resolver SPP y otros problemas multiobjetivo ha sido el uso de algoritmos genéticos dada su facilidad de implementación y adaptación a los diferentes problemas aunado a su eficiencia y rapidez.

El objetivo central de este trabajo de investigación es resolver el SPP, sin embargo, se ha optado resolver primeramente el KP dado que su dificultad de solución es un límite inferior de la complejidad de SPP. Las dos principales razones para esta decisión son:

1. *Es más fácil plantear una estrategia de solución para el KP que para SPP.* Creemos que sería más fácil resolver primero un problema sencillo utilizando hiper-heurísticos y después llevar este conocimiento a un problema más complicado en su formulación.
2. *Probar que los hiper-heurísticos son robustos.* Si realizamos un hiper-heurístico para el KP y después utilizamos este mismo hiper-heurístico quedará demostrado su robustez.

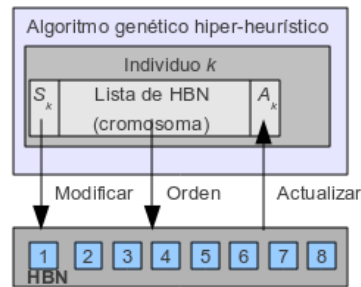
Considerando que los hiper-heurísticos propuestos están basados en algoritmos genéticos, se decidió denominarlos como sigue:

- Hiper-heurístico basado en Algoritmos Genéticos para el Problema de la Mochila 0/1 (*Hyper-heuristic based on Genetic Algorithm for Knapsack Problems*, HHGA\_KP).
- Hiper-heurístico basado en Algoritmos Genéticos para el Problema de Cartera de Proyectos Sociales (*Hyper-heuristic based on Genetic Algorithm for Social Portfolio Problems*, HHGA\_SPP)

## 4.1. Problema de la Mochila 0/1

Cada individuo del algoritmo genético representa una secuencia de como las HBN se van a ejecutar y son estas las que interactúan con la solución

del problema agregando o quitando objetos a la mochila. Las HBN son muy básicas y solo permiten sacar o meter objetos en la mochila de diferentes maneras. Ninguna de estas heurísticas pueden resolver el KP por si solas.



$S_k$  : solución asociada al individuo  $k$ .  
 $A_k$  : Aptitud de la solución del individuo  $k$ .

Figura 4.1: Diagrama de representación del Hiper-heurístico

En la Figura 4.1 se muestra el diagrama que representa al hiper-heurístico, que es un algoritmo genético y su objetivo principal es encontrar la serie de HBN que generan mejores soluciones del KP. El algoritmo genético nunca manipula directamente las soluciones aun cuando tiene en sus individuos soluciones asociadas a dicho problema. Las HBN son las encargadas de ir modificando las soluciones al KP para generar nuevas soluciones.

#### 4.1.1. Representación de la solución

La representación de la solución para el KP es un arreglo binario en donde cada casilla representa a un objeto, si en dicha casilla hay un 1 esté indica que el objeto esta dentro de la mochila, si es un 0 indica el que objeto no esta en la mochila.

Objeto	1	2	3	4	5	6	7	8
Solución	1	0	1	1	1	0	0	1

Tabla 4.1: Representación de la solución del KP

En la Tabla 4.1 se muestra un ejemplo de la representación de la solución para una instancia de 8 objetos del KP, como se puede observar el objeto 1, 3, 4, 5 y 8 se encuentran en la mochila.

Como se mencionó en el Capítulo 2, el objetivo principal del KP es maximizar el beneficio total de los objetos que quedan dentro de la mochila. Como valor aptitud de una solución se tomará el beneficio de los objetos dentro de la mochila sobre la suma de los beneficios de todos los objetos. Este valor sera un número que esta entre 0 y 1. Por ejemplo, si la suma del beneficio de los objetos dentro de la mochila es de 35 y el valor de la suma de los beneficios de todos los objetos es de 58 entonces el valor de la aptitud de esta solución sera:  $aptitud = 35/58 = 0.6034$ .

#### 4.1.2. Heurísticas de Bajo Nivel

Un hiper-heurístico es una heurística que tiene a su disposición un conjunto de HBN y va seleccionando, de acuerdo a algún criterio, la secuencia en que las heurísticas de bajo nivel se van ejecutando. Cada individuo del algoritmo genético representa una secuencia de como se ejecutarán los algoritmo de bajo nivel para obtener una solución del problema.

Las HBN deben ser técnicas sencillas que permitan moverse entre el conjunto de soluciones del problema a resolver. Tradicionalmente las heurísticos cuentan con fases de exploración, que permiten moverse a otras regiones inexploradas de las soluciones, y fases de intensificación, que permiten buscar en los vecinos cercanos a una buena solución. Se deben de diseñar e implementar heurísticos de bajo nivel con técnicas de exploración e intensificación que permitan moverse por todo el espacio de soluciones para encontrar la solución optima. Siguiendo los lineamientos anteriores, los algoritmos de bajo nivel para KP se pueden clasificar en dos grupos: (1) *Agregar*, (2) *Quitar*.

Las heurísticas de **Agregar** permiten ingresar objetos a la mochila mientras sea posible. Dos de estas heurísticas agregan solo un elemento a la mochila, el que tiene mejor rendimiento o el segundo de mejor rendimiento. Las otras dos heurísticas de Agregar agregan más de dos elementos hasta

tantos como sean posible. Las heurísticas de **Quitar** sacan elementos de la mochila para dar espacio y permitir meter otros elementos que permitan maximizar la función objetivo. Dos de estas heurísticas sólo sacan un objeto a la vez elegido de manera aleatoria o el objeto que tenga el peor rendimiento de los que están en la mochila. Las otras dos heurísticas sacan más de dos objetos cada vez (hasta un máximo de doce), los objetos pueden ser seleccionados de manera aleatoria tomando los que tengan menor rendimiento.

En total son 8 las heurísticas de bajo nivel propuestos:

1. **agrega**: Agrega el objeto con mejor rendimiento.
2. **quitar\_aleatorio**: Saca de manera aleatoria un objeto.
3. **agregar\_mrm1**: Agrega el segundo mejor objeto con mejor rendimiento.
4. **quitar\_mr**: Saca el objeto con menor rendimiento.
5. **agrega\_mrmn**: Agrega los objetos con mejor rendimiento, tantos como sea posible.
6. **quitar\_aleatorio\_n**: Saca  $n$  objetos de manera aleatoria,  $n$  es un número entre 2 y 12.
7. **agrega\_aleatorio**: Agrega  $n$  objetos de manera aleatoria,  $n$  es un número aleatorio entre 2 y 12.
8. **quitar\_mr\_n**: Saca  $n$  objetos con el menor rendimiento,  $n$  es un número entre 2 y 12.

El número que acompaña al nombre y descripción de cada heurística es utilizado para crear los cromosomas para el algoritmo genético.

### 4.1.3. Heurística de Alto Nivel

El meta heurístico que sirve de base para crear el hiper-heurístico es un algoritmo genético. Cada individuo de una población del algoritmo genético representa el orden y el tipo de HBN que se va a ejecutar.

Heurística de bajo nivel	2	3	8	7	1	4	3	5
Gen	1	2	3	4	5	6	7	8

Tabla 4.2: Ejemplo de un cromosoma del algoritmo genético hiper-heurístico

La Tabla 4.2 representa un cromosoma en el algoritmo genético hiper-heurístico. En este cromosoma, el primer gen indica que primero se ejecutará la heurística de bajo nivel número 1, es decir, **agrega**, el segundo gen representa que se ejecutara la heurística 3, **agregar mrm1** y así sucesivamente para los demás genes. A cada individuo se le asocia una solución del KP, esta solución es la obtenida con la aplicación de las heurísticas de bajo nivel del cromosoma.

La aptitud de cada individuo esta dada por el valor de la solución del KP asociada a cada individuo resultante de la aplicación de la serie de heurísticas de bajo nivel del cromosoma.

El algoritmo genético utilizado como base para crear el hiper-heurístico es el más básico de los algoritmos genéticos. El número de generaciones es un parámetro estático, los individuos siempre se cruzan, la cruce es en un punto y la selección de los individuos para la cruce es de manera aleatoria. El porcentaje de mutación es de 25 % y la mutación es intercambiar una heurística por otra seleccionada de manera aleatoria. Además existe un elitismo del 5 % de la población. Los valores de los parámetros se tomaron de la literatura[REF]. El pseudo-código del algoritmo genético hiper-heurístico es mostrado en el Algoritmo 2.

En la Figura 4.2 se ilustra como el funcionamiento del hiper-heurístico. Se inicia con una solución  $S_0$ , que es generada por un algoritmo voraz. Se genera la población de individuos del Algoritmo Genético de manera aleatoria. Los individuos representan en que orden y que heurísticas se van aplicar a  $S_0$  para



**Algoritmo 2** HHGA\_KP

- 
- 1: Generar una solución inicial  $S_0$  utilizando un algoritmo voraz
  - 2: Generar de manera aleatoria una población inicial de  $n$  individuos
  - 3: **mientras** Generación < Número\_generaciones **hacer**
  - 4:   **para todo** cromosoma  $k$ ,  $k \leq 100$ : **hacer**
  - 5:     Aplicar la serie de heurísticas a la solución  $S_0$
  - 6:     Almacenar la solución resultante  $S_k$
  - 7:     Calcular la aptitud del individuo de acuerdo a la solución  $S_k$
  - 8:   **fin para**
  - 9:   Compara  $S_k$  con  $S_0$ , si  $S_k > S_0$ ,  $S_0 = S_k$
  - 10:   Seleccionar a los padres y cruzar
  - 11:   Mutar
  - 12:   Agregar los individuos más aptos a una lista élite.
  - 13: **fin mientras**
- 

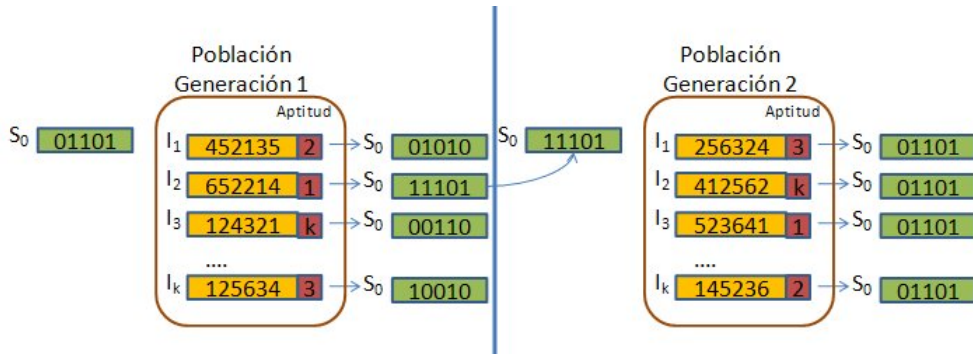
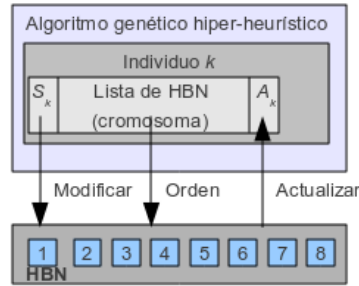


Figura 4.2: Ejemplo del funcionamiento del HHGA\_KP

generar nuevas soluciones. Se calcula la aptitud de las soluciones generadas y se ordenan, la que tiene mejor aptitud pasa a ser la solución  $S_0$  en la siguiente generación. La siguiente población es generada de la cruce y mutación de la generación anterior. Cabe destacar que el Algoritmo Genético trabaja en el espacio de las Heurísticas de Bajo Nivel y busca aquellas que generen las mejores soluciones. Las Heurísticas de Bajo Nivel son las que van a modificar la solución al problema de carteras directamente.

## 4.2. Problema de Cartera de Proyectos Sociales

En esta sección se describe a detalle los hiper-heurísticos utilizado para SPP. En general, la base del hiper-heurístico es un algoritmo genético. Los individuos representan una secuencia de como los algoritmos de bajo nivel se van a ejecutar. Los algoritmos de bajo nivel van a interactuar con la solución del problema agregando, quitando o intercambiando proyectos. Ninguna de estas heurísticas de bajo nivel pueden resolver el SPP por si solas.



$S_k$  : solución asociada al individuo  $k$ .  
 $A_k$  : Aptitud de la solución del individuo  $k$ .

Figura 4.3: Diagrama de representación del Hiper-heurístico

En la Figura 4.3 se muestra el diagrama que representa al hiper-heurístico, que es un algoritmo genético y su objetivo principal es encontrar la serie de HBN que generan mejores soluciones del SPP. El algoritmo genético nunca manipula directamente las soluciones aun cuando tiene en sus individuos soluciones asociadas a dicho problema. Las HBN son las encargadas de ir modificando las soluciones del SPP para generar nuevas soluciones.

### 4.2.1. Representación de la solución

Para el SPP, la representación de la solución es un arreglo binario de tamaño  $n$  en donde cada casilla representa a un proyecto, el dominio de

valores para indicar si el proyecto recibe o no apoyo son  $[0,1]$ , 0 si el proyecto no recibe apoyo y 1 si el proyecto recibe apoyo. Los proyectos están ordenados de mayor a menor según su costo. Así, el proyecto 1 es el que necesita más presupuesto y el 8 el que necesita menos presupuesto.

Proyecto	1	2	3	4	5	6	7	8
Solución	1	0	1	1	1	0	0	1

Tabla 4.3: Representación de la solución del SPP

En la Tabla 4.3 se muestra un ejemplo de la representación de la solución para una instancia de 8 proyectos del *SPP*, como se puede observar los proyectos 1, 3, 4, 5 y 8 son los que se apoyaran.

El objetivo principal del SPP es maximizar el beneficio social de los proyectos seleccionados para recibir apoyo. Dado que en el modelo PORTADOR solo permite evaluaciones entre carteras, no tiene sentido evaluar un solo proyecto y por lo tanto la solución individual no tiene ninguna *Aptitud* asociada.

#### 4.2.2. Heurísticas de Bajo Nivel

Las HBN para SPP se pueden clasificar en dos grupos: (1) *Exploración* y (2) *Explotación* o *Intensificación*. El número que acompaña al nombre y descripción de cada heurística es utilizado para crear los cromosomas del algoritmo genético. En total son 7 las heurísticas de bajo nivel propuestas:

1. **Cambio Aleatorio:** Cambia un proyecto por otro de manera aleatoria.
2. **Genera Aleatorio:** Genera una nueva solución al azar.
3. **Cambio Izquierda:** Cambia un proyecto seleccionado al azar por el primer proyecto que haga factible la solución buscando los proyectos de izquierda a derecha.

4. **Cambio Derecha:** Cambia un proyecto seleccionado al azar por el primer proyecto que haga factible la solución empezando buscando de derecha a izquierda.
5. **Cambio x Región:** Saca al azar un proyecto de cada región y los sustituye por otro proyecto de la misma región.
6. **Cambio x Área:** Saca al azar un proyecto de cada área y los sustituye por otro proyecto de la misma área.
7. **Cambio Opuesto:** Selecciona cuatro proyectos al azar, si un proyecto esta en la cartera lo saca, en caso contrario lo mete en la cartera.

Las heurísticas de Exploración (1,2) permiten generar nuevas soluciones diferentes a las actuales, permitiendo dar grandes saltos en el espacio de búsqueda de las soluciones y así explorar nuevas soluciones. Las heurísticas de Explotación (5,6,7) permiten buscar soluciones vecinas a las soluciones actuales en búsqueda de mejores soluciones en el mismo vecindario. Sólo las heurísticas 3 y 4 tienen cierto equilibrio entre exploración y explotación. En la Tabla 4.4 se muestra la HBN y el tipo de movimiento que realiza en el espacio de búsqueda.

HBN	Tipo movimiento
Cambio Aleatorio	Exploración
Genera Aleatorio	Exploración
Cambio Izquierda	Exploración / Explotación
Cambio Derecha	Exploración / Explotación
Cambio x Región	Explotación
Cambio x Área	Explotación
Cambio Opuesto	Explotación

Tabla 4.4: Heurística de Bajo Nivel y su tipo de movimiento

### 4.2.3. Heurística base para el hiper-heurístico

El meta heurístico que sirve de base para crear el hiper-heurístico es un algoritmo genético. Cada individuo de una población del algoritmo genético

representa el orden y el tipo de HBN que se va a ejecutar. La Tabla 4.5 ejemplifica el cromosoma de un individuo en el algoritmo genético hiper-heurístico. En éste, el primer gen indica que primero se ejecutará la heurística de bajo nivel número 1, es decir, **Cambio Aleatorio**, el segundo gen indica que se ejecutará la heurística 3, **Cambio Izquierda** y así sucesivamente para los demás genes. A cada individuo se le asocia una solución del Problema de Cartera de Proyectos Sociales, esta solución cambiará de acuerdo a las heurísticas de bajo nivel de su cromosoma.

	1	3	6	2	7	4	3	5
Gen	1	2	3	4	5	6	7	8

Tabla 4.5: Ejemplo del cromosoma de un individuo del HHGA\_SPP.

La aptitud de cada individuo está en función del número de soluciones de la población que dominan de manera estricta a la solución. Así por ejemplo una *aptitud\_dominancia* = 5 indica que la solución es **dominada estrictamente** por otras 5 soluciones de la población. Para el caso del HHGA\_SPPv1, se ha denominado *aptitud\_dominancia* al valor dado por la función  $card(S_0)$  del modelo PORTADOR.

El algoritmo genético utilizado como base para crear el hiper-heurístico es el más básico de los algoritmos genéticos. El número de generaciones es un parámetro estático, los individuos siempre se cruzan, la cruce es en un punto y la selección de los individuos para la cruce es de manera aleatoria. El porcentaje de mutación es de 25 % y la mutación es intercambiar una heurística por otra seleccionada de manera aleatoria. Además existe un elitismo del 10 % de la población, ver Tabla 4.6. En el Algoritmo 3 se muestra el HHGA\_SPPv1.

### 4.3. Mejoras al algoritmo

En los experimentos realizados con el Algoritmo HHGA\_SPPv1 para las instancias de 4 objetivos no se obtuvieron los resultados esperados (el experimento se describe con mayor detalle en la Sección 5.2.2). El

Característica	Valor
Num. individuos	100
Num. generaciones	100
Selección	Aleatoria
Cruza	En 1 punto
Mutación	5 %, intercambio aleatorio
Elitismo	10 %

Tabla 4.6: Características del algoritmo genético de HHGA\_SPP

comportamiento del Algoritmo fue ineficiente al no encontrar buenas soluciones al problema. Por tal motivo, se propusieron dos métricas para analizar el comportamiento de las HBN dentro del HHGA\_SPPv1 que den indicios de como mejorar el algoritmo propuesto.

#### 4.3.1. Métricas del Comportamiento

Los resultados mostrados por el HHGA\_SPPv1 u otros algoritmos podrían ser mejorados a través del estudio de los factores que afectan el desempeño, y al identificar dichos factores sería posible implementar estrategias que mejoren el su desempeño[28].

Para analizar el comportamiento del HHGA\_SPPv1, se formularon dos métricas, en particular para determinar las fortalezas y debilidades del algoritmo y en consecuencia poder realizar adaptaciones que permitan mejorar su desempeño.

#### Tendencia de Uso de HBN por Generación

Esta métrica permite saber que tipo de HBN se usa más en cada generación aprovechando que están organizadas por tipo de búsqueda. Primero aparecen las HBN que permiten hacer más exploración (HBN 1 y 2), enseguida las que equilibran la exploración e intensificación (HBN 3 y 4) y

**Algoritmo 3** HHGA\_SPPv1

- 
- 1: Generar una solución inicial  $S$  del SPP de manera aleatoria
  - 2: Generar de manera aleatoria una población inicial de 30 individuos
  - 3: **para todo** individuo  $k$ ,  $k \leq 30$  **hacer**
  - 4:   Aplicar a la solución  $S$  la serie de heurísticas codificados en el cromosoma del individuo  $k$
  - 5:   Aplicar técnicas de **diversificación**
  - 6:   Almacenar la solución resultante  $S_k$
  - 7: **fin para**
  - 8: Calcular la *aptitudDominancia* de cada individuo de acuerdo a la solución  $S_k$  asociada
  - 9: Identificar las soluciones no dominadas
  - 10:   Insertar  $S_k$  en  $LS$ ;  $LS$  es la lista de las mejores soluciones
  - 11: Generar una nueva población
  - 12:   Seleccionar a los padres y cruzar aplicando diferentes técnicas
  - 13:   Mutar los individuos
  - 14:   Agregar los individuos más aptos a una lista elite
  - 15: Regresar al paso 3 hasta llegar a la condición de paro
- 

finalmente las de intensificación (HBN 5, 6 y 7). Así, si la métrica es pequeña se puede concluir que el algoritmo está explorando más y, si es grande que está haciendo más intensificación. La métrica se denomina Tendencia de Uso de Heurísticas de Bajo Nivel por Generación y su función es:

$$T_g = \frac{1}{tp} \sum_{p=0}^{tp} \frac{1}{ti} \sum_{k=0}^{ti} I_{pk}, \forall g \quad (4.1)$$

donde:

$T_g$  : Tipo de HBN usada en promedio en la generación  $g$ .

$I_{pk}$  : Es la HBN del gen  $k$  del  $p$ -ésimo individuo de la generación  $g$ .

$tp$  : Tamaño de la población.

$ti$  : Tamaño del individuo.

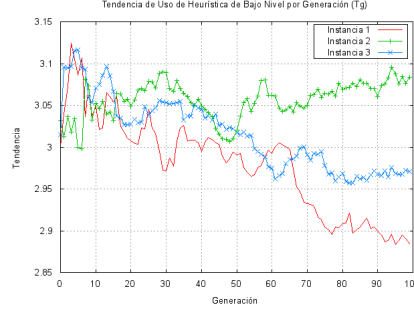


Figura 4.4: Métrica  $T$  de las instancias 1,2 y 3

En la Figura 4.4 se muestra la métrica  $T$  para las instancias 1,2 y 3. El HHGA\_SPPv1 tuvo mejores resultados en las instancias 1 y 2, para las instancias antes mencionadas se percibe cierta oscilación en el valor de la  $T_g$ , esto significa que el HHGA\_SPPv1 usa HBN de intensificación y exploración de manera cíclica. Al parecer esta oscilación permite que el HHGA\_SPPv1 encuentre más y mejores soluciones. En la instancia 3, no es suficiente la oscilación que realiza para encontrar las soluciones óptimas. En general, aun cuando el algoritmo utiliza HBNs de exploración parece que no es suficiente; constituyéndose en una gran área de oportunidad para el hiper-heurístico genético.

### Frecuencia de Uso de HBN por Generación

Esta métrica permite saber cual es la HBN más frecuentemente usada por generación de HHGA\_SPPv1. Al conocer que HBN se utilizan con mayor frecuencia permite conocer el comportamiento que tiene el HH Genético en cada generación. La métrica se denomina Frecuencia de Uso de Heurísticas de Bajo Nivel por Generación y corresponde a la Ecuación 4.2.

$$F_{hg} = \sum_{p=0}^{tp} HBN_{hpg}, \forall h, \forall g \quad (4.2)$$

donde:



$F_{hg}$  = Frecuencia de uso de la HBN  $h$  para cada generación  $g$ .

$HBN_{hpg} = 1$  si la HBN  $h$  se usa en el individuo  $p$  de la generación  $g$  y 0 si no se usa.

$tp$  = Tamaño de la población.

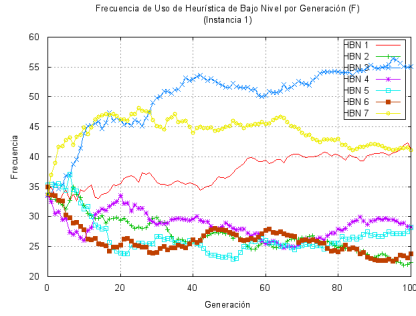


Figura 4.5: Métrica  $F_{hg}$  de la instancia 1

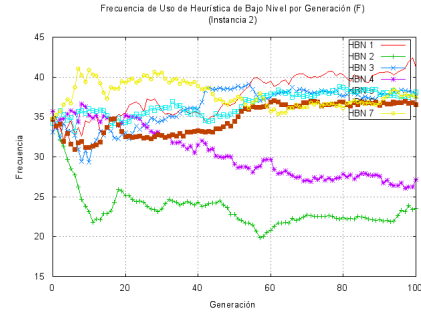


Figura 4.6: Métrica  $F_{hg}$  de la instancia 2

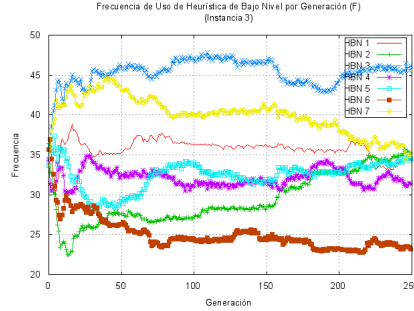


Figura 4.7: Métrica  $F_{hg}$  de la instancia 3

En las Figuras 4.5, 4.6 y 4.7 se muestran las frecuencias de uso de las HBN para las instancias de prueba 1, 2 y 3. La HBN más usada en las 3 instancias son las marcadas con los números 3 y 7. Las instancias menos usadas son las numeradas con 2, 4 y 6. Como conclusión al analizar las figuras mencionadas se observa que el HHGA\_SPPv1 “prefiere” las HBN de intensificación y solo un poco de exploración con la HBN 1. Esto puede ser la causa de el estancamiento de las soluciones.

### 4.3.2. Mejoras al Algoritmo

El conocer el comportamiento que tienen las HBN en el transcurso de la ejecución del Hiper-heurístico propuesto para SPP ha permitido proponer dos mejoras al algoritmo que permitan diversificar la búsqueda en el espacio de soluciones y de esta manera encontrar todas las NOS en cada instancia. La primera modificación consiste en variar de vez en vez la solución inicial  $S_0$  que genera las nuevas soluciones para los individuos del hiper-heurístico, es decir, en vez de siempre utilizar la mejor solución encontrada hasta el momento, utilizar una solución generada de manera aleatoria como  $S_0$ . Esta modificación permitirá que se busque en otras regiones no exploradas del espacio de búsqueda permitiendo salir de óptimos locales u óptimos globales y encontrar todas las NOS. La modificación se muestra en el Algoritmo 4 en la línea 5 y 6.

La segunda mejora cambia la forma de seleccionar los padres y cruzar. En la primera versión la selección de los padres era aleatoria en esta segunda versión la siguiente generación se compone de cuatro cambios. El primer 10 % de la población es la población elite de la generación anterior. El siguiente 30 % de la población corresponde a la cruce de los mejores individuos. El siguiente 30 % corresponde a la cruce aleatoria de todos los individuos. El 15 % siguiente se compone de la cruce de padres malos y el 15 % final de generar nuevos individuos de manera aleatoria. La modificación al algoritmo se muestra en la línea 12 del Algoritmo 4.

---

**Algoritmo 4** HHGA\_SPPv2

---

```
1: Generar una solución inicial  $S_0$  utilizando un algoritmo voraz
2: Generar de manera aleatoria una población inicial de  $n$  individuos
3: mientras Generación < Número_generaciones hacer
4:   para todo cromosoma  $k$ ,  $k \leq 100$ : hacer
5:     Usar la solución  $S_0$  ó
6:     generar una solución  $S_0$  aleatoria
7:     Aplicar la serie de heurísticas a la solución  $S_0$ 
8:     Almacenar la solución resultante  $S_k$ 
9:     Calcular la aptitud del individuo de acuerdo a la solución  $S_k$ 
10:   fin para
11:   Compara  $S_k$  con  $S_0$ , si  $S_k > S_0$ ,  $S_0 = S_k$ 
12:   Seleccionar a los padres usando diferentes técnicas de
     selección y cruzar
13:   Mutar
14:   Agregar los individuos más aptos a la lista élite.
15: fin mientras
```

---



# Capítulo 5

## Experimentación y Resultados

En este Capitulo se describen las instancias y los experimentos realizados para comprobar el funcionamiento de los algoritmos propuestos en la sección anterior.

### 5.1. Ambiente General

Todos los experimentos se realizaron en una laptop marca ACER bajo el sistema operativo Linux. Las características de la computadora se describen en la Tabla 5.1.

Característica	
Marca	Acer Aspire 5536
Sistema Operativo	Fedora 13 (64 bits)
Compilador	gcc 4.4.4
Procesador	AMD Athlon X2
Memoria RAM	3 GB

Tabla 5.1: Características del equipo usado para los experimentos

## 5.2. Problema de la Mochila

Para realizar las pruebas se utilizaron las instancias de *OR-Library*, la cual ofrece una colección de datos de pruebas para una gran variedad de problemas de Investigación de Operaciones. Esta colección de pruebas se ha compartido entre los investigadores que trabajan en el área de Investigación de Operaciones en Reino Unido y se ha extendido a otros países del mundo vía correo electrónico [1]. En total son 240 instancias para el KP que van desde 1000 a 20000 objetos. En la Tabla 5.2 se describe el número y tamaños de las instancias disponibles para KP.

Tipo de instancias	Cantidad de instancias	Número de objetos
A	40	1000
B	40	2500
C	40	5000
D	40	7500
E	40	10000
F	40	20000

Tabla 5.2: Instancias para KP de OR-Library

El mejor algoritmo del estado del arte es uno desarrollado por David Pisinger, dicho algoritmo tiene la propiedad de que obtiene el núcleo mínimo del Problema utilizando programación dinámica. Este algoritmo supera a cualquier algoritmo de la mochila conocido[25].

El HHGA\_KP se ejecutó 30 veces por cada instancia. Los resultados mostrados en la Figura 5.1 corresponde al beneficio promedio de las 30 ejecuciones. De las 240 instancias disponibles se logró igualar en 140 instancias los resultados de Pisinger, en las 100 instancias restantes el error promedio fue de 0.03 %. El tiempo máximo de ejecución para una instancia fue de 1 segundo; siendo el promedio de 0.38 segundos.

En la Figura 5.1 se muestra el número de instancia y el beneficio obtenido tanto por el algoritmo determinista de Pisinger como por el HHGA\_KP. Se puede notar que las gráficas de beneficio están empalmadas; esto se debe a

que ambos algoritmos obtienen casi los mismo resultados.

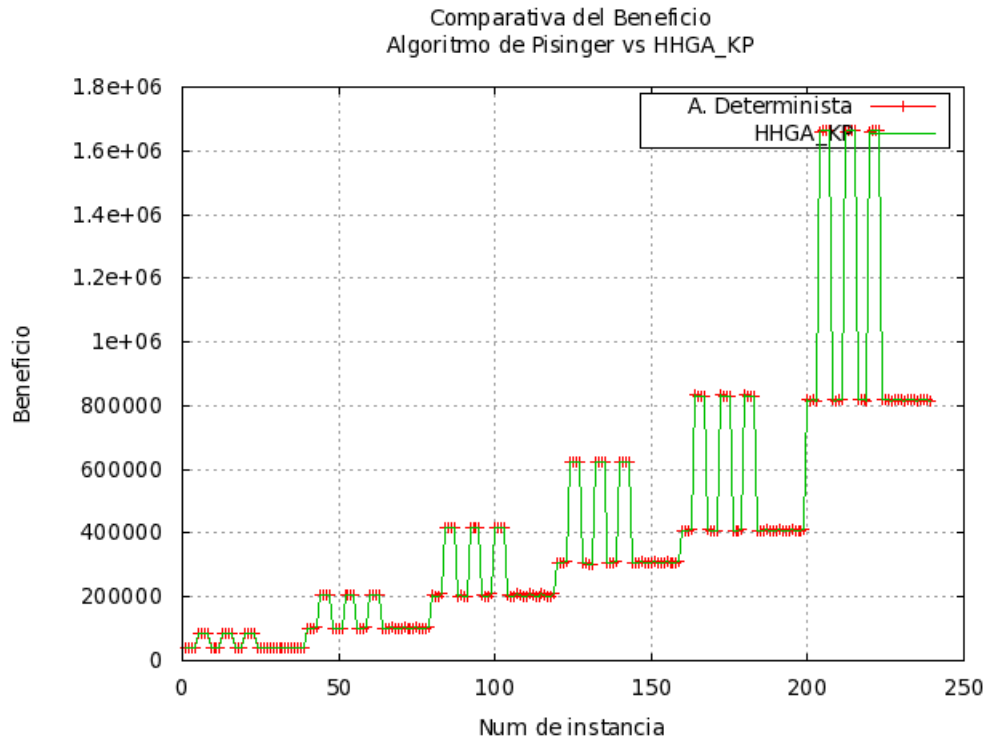


Figura 5.1: Algoritmo determinista de Pisinger vs HHGA\_SPP

En la Figura 5.2 se muestra el error promedio de HHGA\_KP con respecto al algoritmo de Pisinger. Es interesante resaltar que son las instancias de menor tamaño en donde el error es más grande, esto es, no se alcanzan los resultados obtenidos por el algoritmos determinista. Por el contrario, en las instancias de mayor tamaño el error es menor.

Es importante recalcar que en este trabajo de investigación el problema central no es el de la mochila y por lo tanto no se pretende encontrar las soluciones óptimas. El resolver primeramente el KP permitió reunir los conocimientos necesarios para abordar el SPP usando algoritmos hiper-heurísticos.

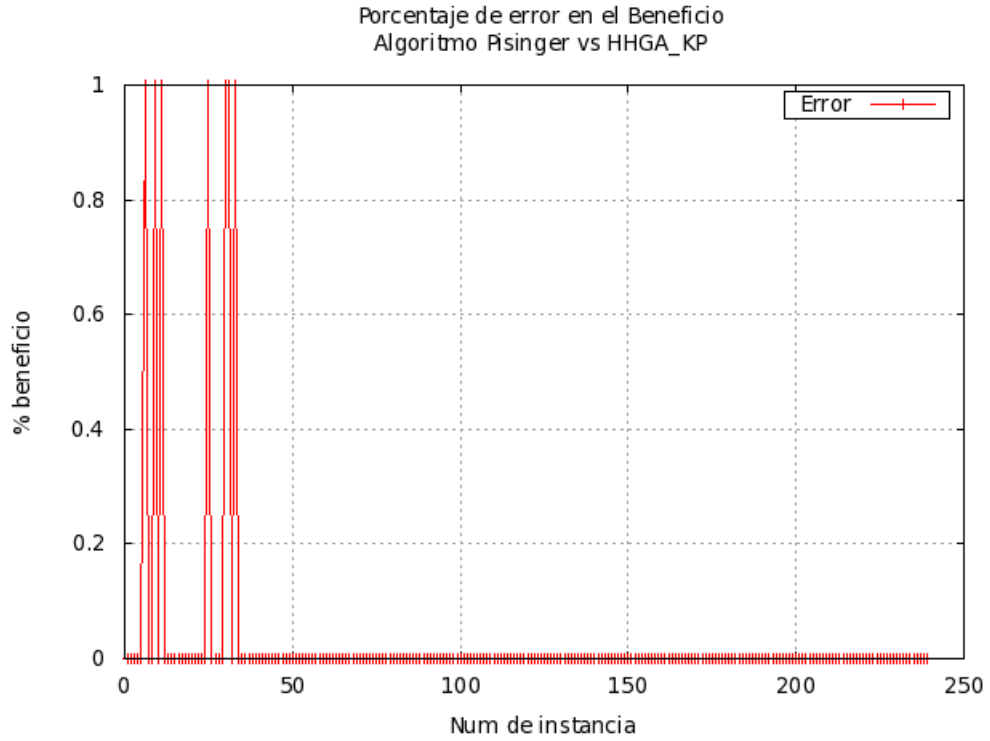


Figura 5.2: Porcentaje de error en el Beneficio obtenido por el HHGA\_KP con respecto al algoritmo determinista de Pisinger

### 5.3. Problema de Cartera de Proyectos Sociales

#### 5.3.1. Instancias de Prueba

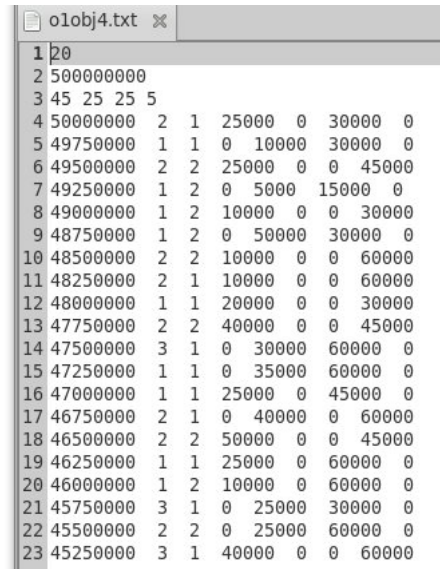
Las instancias de SPP empleadas en las pruebas de este trabajo fueron proporcionadas por Fernández y son las mismas que utilizó para realizar sus experimentos del modelo PORTADOR en [13]. Estas instancias se describen en la Tablas 5.3.

En cada archivo de instancia se indica el numero de proyectos, el monto total del presupuesto y la descripción de cada proyecto.



No. Instancia	Nombre	Proyectos	Objetivos	Presupuesto
1	o1obj4	20	4	500,000,000
2	o2obj4	20	4	500,000,000
3	o3obj4	20	4	500,000,000
4	o1obj9	20	9	750,000,000
5	o2obj9	20	9	750,000,000
6	o3obj9	20	9	750,000,000

Tabla 5.3: Descripción de las instancias



```

o1obj4.txt x
1 20
2 500000000
3 45 25 25 5
4 50000000 2 1 25000 0 30000 0
5 49750000 1 1 0 10000 30000 0
6 49500000 2 2 25000 0 0 45000
7 49250000 1 2 0 5000 15000 0
8 49000000 1 2 10000 0 0 30000
9 48750000 1 2 0 50000 30000 0
10 48500000 2 2 10000 0 0 60000
11 48250000 2 1 10000 0 0 60000
12 48000000 1 1 20000 0 0 30000
13 47750000 2 2 40000 0 0 45000
14 47500000 3 1 0 30000 60000 0
15 47250000 1 1 0 35000 60000 0
16 47000000 1 1 25000 0 45000 0
17 46750000 2 1 0 40000 0 60000
18 46500000 2 2 50000 0 0 45000
19 46250000 1 1 25000 0 60000 0
20 46000000 1 2 10000 0 60000 0
21 45750000 3 1 0 25000 30000 0
22 45500000 2 2 0 25000 60000 0
23 45250000 3 1 40000 0 0 60000

```

Figura 5.3: Fragmento del archivo de una instancia del SPP

En la Tabla 5.3 se muestra un fragmento del archivo de la instancia o1obj4.txt del SPP. Se puede observar que en la línea 1 aparece el número de proyectos, en la línea 2 el presupuesto total asignado, en la línea 3 los pesos de los objetivos, de la línea 4 en adelante aparece la descripción de cada proyecto empezando por el monto necesario, el área y la región y los valores de los criterios. Cada proyecto tiene esta incluido en un área y en una región, cada área y región tienen un límite del presupuesto total.

Es importante destacar que el peso de cada objetivo es un dato

proporcionado por el tomador de decisiones. Para los experimentos de este trabajo se han utilizado los mismos valores que en [13]. Otro dato que debe ser proporcionado por el tomador de decisiones pero que se mantienen fijos en el programa son los límites del presupuesto por región y por área. Cada área y cada región tiene un límite inferior y superior del total de presupuesto que permiten que éste sea distribuido equitativamente entre todas las áreas y regiones.

Estos dos datos son los que convierten al problema multi-objetivo en un problema multi-criterio. Los valores de estos datos van a permitir generar soluciones que concuerden con las creencias y preferencias que el tomador de decisiones quiere para distribuir el presupuesto.

En la Tabla 5.4 se muestra la descripción de cada proyecto en las instancias. La primera columna es el monto del presupuesto que dicho proyecto necesita para llevarse a cabo. La segunda es el área y región a la que pertenece. Cada proyecto tiene cuatro o nueve criterios a maximizar, cada criterio mide las personas beneficiadas por la realización de dicho proyecto en base a su nivel social y al impacto del proyecto. Para las instancias de cuatro objetivos se manejan dos niveles sociales: Pobres (P) y Clase Media (CM); y dos niveles de impacto: Impacto Alto (IA) o Impacto Bajo (BI), de modo que se tienen cuatro posibles combinaciones: P\_AI, P\_BI, CM\_AI y CM\_BI. Par las instancias de nueve objetivos se tiene tres niveles sociales: Pobres (P), Clase Media (CM) y Clase Alta (CA) y tres niveles de impacto: Impacto Alto (IA), Impacto Medio (IM) e Impacto Bajo (IB) de modo que se tienen 9 combinaciones: P\_IA, P\_IM, P\_IB, CM\_IA, CM\_IM, CM\_IB, CA\_IA, CA\_IM y CA\_IB.

Monto	Área	Región	P_AI	P_BI	CM_AI	CM_BI
50000000	2	1	25000	0	30000	0
49750000	1	1	0	10000	30000	0
49500000	2	2	25000	0	0	45000
49250000	1	2	0	5000	15000	0

Tabla 5.4: Descripción de los proyectos en las instancias

### 5.3.2. Prueba con Instancias de 4 Objetivos

Para las primeras pruebas del algoritmo solo se usaron las instancias de 4 objetivos, es decir, el problema es maximizar los 4 objetivos que corresponden al número de personas beneficiadas por aplicar algún proyecto.

Con el HHGA\_SPPv1 (primera versión del hiper-heurístico), cada instancia se resolvió 30 veces y los resultados mostrados en la Tabla 5.7 corresponden al número de Soluciones No Dominadas (NOS, *No Outranking Solutions*) encontradas en promedio y en forma acumulada. Los resultados del HHGA\_SPPv1 se compararon con los de un Algoritmo Exacto y con los del algoritmo NOSGA propuesto por Fernández *et al.* El Exacto y NOSGA fueron ejecutados una sola vez; para ellos se muestra el número de NOS encontrados en la única corrida. Aunque NOSGA no obtuvo todas las NOS esperadas en todas las instancias, sus resultados fueron mejores que los obtenidos por HHGA\_SPPv1. Sin embargo, dado que HHGA\_SPPv1 esta soportado en un algoritmo genético básico, sus resultados, aunque son de menor calidad que NOSGA, son prometedores.

La columna *aptitud promedio* que es el promedio de NOS en la ultima generación, de la Tabla 5.7, nos indica que los resultados del HHGA\_SPPv1 están muy cerca de 0. La columna *NOS Acumuladas* nos indica que se están encontrando muchas soluciones distintas en las diferentes corridas. Como se verá en el siguiente experimento el uso de técnicas avanzadas de algoritmos genéticos y métricas de comportamiento, como las descritas en la Sección 4.3 mejoran considerablemente el desempeño del algoritmo propuesto.

En la Tabla 5.6 se muestran el número de veces, de las 30 corridas, que el HHGA\_SPP encontró una NOS (columna 1 NOS) o dos NOS (columna 2 NOS) por cada instancia. Se puede observar que en la instancia 1 el algoritmo tuvo su mejor desempeño. Lo que se puede concluir al observar estos resultados es que el algoritmo se estanca cuando encuentra soluciones muy buenas y queda atrapado en óptimos locales de los que no puede salir.

Como conclusión final de este experimento se puede decir que el algoritmo no obtuvo buenos resultados para el SPP. Es por eso que fue necesario realizar algunas adaptaciones y mejoras al algoritmo para mejorar estos resultados.

Instancia	Exacto NOS	NOSGA NOS	HHGA_SPP		
			NOS Promedio	NOS Acumuladas	Aptitud Promedio
1	4	4	0.76	4	2
2	3	1	0.33	3	0.73
3	10	7	0.03	1	1.87

NOS: Soluciones no dominadas

Tabla 5.5: Comparación de resultados entre el Algoritmo Exacto, el algoritmo NOSGA y el HHGA\_SPP

Instancia	1 NOS	2 NOS
1	13	5
2	10	0
3	1	0

Tabla 5.6: Resultados del HHGA\_SPP

Las mejoras realizadas permiten al algoritmo expandir el área de búsqueda a regiones inexploradas además de permitir salir de óptimos tanto locales como globales para encontrar más de una solución óptima. Esta última mejora es muy importante dado que se está resolviendo un problema que tiene múltiples soluciones óptimas y es importante encontrarlas.

### 5.3.3. Pruebas con Instancias de 9 Objetivos

Con la nueva versión del algoritmo HHGA\_SPPv2 se resolvieron las instancias de 4 objetivo y de 9 objetivos. Cada instancia se resolvió 30 veces y los resultados se muestran en la Tabla 5.7. Los resultados del HHGA\_SPP se compararon con los de un Algoritmo Exacto y con los del algoritmo NOSGA. El Exacto y NOSGA fueron ejecutados una sola vez y para ellos se muestra el número de NOS encontrados en la única corrida.

En la Tabla 5.7 se muestran el número de NOS encontradas por cada algoritmo (Exacto, NOSGA y HHGA\_SPP). Se puede observar que en las

instancias 1, 2, 4, 5 y 6 se llega al mismo resultado que el algoritmo Exacto, cabe destacar que dicho algoritmo tarda en resolver cada instancia más de 5 minutos. En las instancias 2 y 3 supera claramente a NOSGA.

<b>Instancia</b>	<b>Exacto NOS</b>	<b>NSGA-II* NOS</b>	<b>NOSGA NOS</b>	<b>HHGA_SPPv2 NOS</b>
1	4	3	4	4
2	3	1	1	3
3	10	0	7	9.06
4	6	3	6	6
5	1	1	1	1
6	4	0	4	4

\* Resultados de NSGA-II obtenidos de [20]

NOS: Soluciones no dominadas

Tabla 5.7: Comparación de resultados entre el Algoritmo Exacto, NSGA-II, NOSGA y el HHGA\_SPP

Como conclusión final de este experimento se puede decir que el algoritmo obtuvo excelentes resultados para el SPP superando a los resultados obtendidos por los trabajos del estado del arte. Las dos mejoras realizadas a partir de la información proporcionadas por las métricas propuestas han permitido mejorar en gran medida el desempeño del hiper-heurístico dándole la capacidad de resolver problemas tanto mono objetivos como multi-objetivos.



# Capítulo 6

## Conclusiones y Trabajos Futuros

### 6.1. Beneficios

Se aportan los siguientes productos de investigación:

- a) Un hiper-heurístico básico para resolver el problema de la mochila basado en un algoritmo genético. El algoritmo tiene a su disposición siete heurísticas de bajo nivel que permiten insertar o remover elementos de la mochila. El genético es el heurístico de alto nivel que selecciona en cada iteración la mejor forma de aplicar las heurísticas de bajo nivel para llegar paulatinamente a mejores soluciones.
- b) Un hiper-heurístico extendido para resolver el problema de cartera de proyecto sociales. El algoritmo cuenta con ocho heurísticas de bajo nivel enfocadas a realizar intercambios en los proyectos que formaran las carteras. El genético es el heurístico de alto nivel que selecciona en cada iteración la mejor forma de aplicar las heurísticas de bajo nivel para llegar paulatinamente a las carteras que satisfagan mejor las preferencias del tomador de decisiones.

- c) Dos métricas de desempeño que permiten medir aspectos relacionados con el comportamiento de las heurísticas de bajo nivel en el hiper-heurístico. La primera métrica mide la tendencia de uso de las heurísticas de bajo nivel por generación y la segunda cuantifica la frecuencia de uso de las heurísticas de bajo nivel también generación. Las métricas propuestas permiten identificar si existe un balance entre el nivel de exploración y explotación en el proceso de búsqueda de las mejores soluciones realizado por el algoritmo.
- d) Una estrategia de solución para el problema de cartera de proyectos sociales desarrollada con la colaboración de un grupo de trabajo multidisciplinario. El grupo colaboran especialistas en matemáticas, computación e investigación de operaciones.

## 6.2. Trabajos Futuros

Algunas recomendaciones para trabajos futuros son:

- Crear nuevos hiper-heurísticos usando otro tipo de heurísticas de alto Nivel tales como búsqueda Tabú, Recocido Simulado y Colonias de Hormigas, entre otros.
- Proponer nuevas heurísticas de bajo nivel con estrategias inteligentes tales como búsqueda local y metaheurístico.
- Resolver otro tipo de problemas complejos de la misma familia de asignación de recursos para verificar la robustez y alcance. Ambos objetivos son retos que se plantean para este nuevo tipo de algoritmos inteligentes.
- Proponer nuevas métricas de desempeño para realizar un estudio teórico y experimental a profundidad sobre el comportamiento de los hiper-heurísticos.



# Bibliografía

- [1] BEASLEY, J. E. Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society* 41, 11 (1990).
- [2] BURKE, E., HYDE, M., KENDALL, G., OCHOA, G., OZCAN, E., AND QU, R. A survey of hyper-heuristics. *Computer Science Technical Report* (2009).
- [3] BURKE, E., KENDALL, G., NEWALL, J., HART, E., ROSS, P., AND SCHULENBURG, S. *Hyper-heuristic: An emerging direction in modern search technology*. Kluwer Academic Publishers, 2003.
- [4] C. COELLO, A. V. V., AND LAMONT, B. *Evolutionary Algorithms for solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.
- [5] COELLO, A., LAMONT, B., AND VELDHIJZEN, A. V. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.
- [6] COELLO, C. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal* (1999).
- [7] CORTÉS, C. Técnicas de decisión multicriterio: Métodos multicriterios discretos. Tech. rep., Dpto. Economía General y Estadística, Universidad de Huelva, 2006.
- [8] COWLING, P., AND SOUBEIGA, E. Neighborhood structures for personnel scheduling: A summit meeting scheduling problem.

- [9] DENZINGER, J., FUCHS, M., AND FUCHS, M. High performance atp system by combining several ai methods. In *International Joint Conference on Artificial Intelligence*.
- [10] DUARTE, A., PANTRIGO, J., AND GALLEG0, M. *Metaheurísticas*. Universidad Rey Juan Carlos, 2006.
- [11] E., B., AND ZEMEL, E. An algorithm for large zero-one knapsack problem. *Operations Research* (1980).
- [12] FERNÁNDEZ, E., AND LEYVA, C. Method based on multiobjective optimization for deriving a ranking from a fuzzy preference relation. *European Journal of Operational Research* (2004).
- [13] FERNÁNDEZ, E., LOPEZ, E., BERNAL, S., COELLO, C., AND NAVARRO, J. Evolutionary multiobjective optimization using an outranking-based dominance generalization. *Computers & Operations Research* 37, 2 (2010).
- [14] FERNÁNDEZ, E., LÓPEZ, F., NAVARRO, J., AND DURANTE, A. Técnicas inteligentes para la selección de proyectos de i&d en las grandes organizaciones públicas. *Computación y Sistemas* (2006).
- [15] FERNÁNDEZ, E., AND NAVARRO, J. A genetic search for exploiting a fuzzy preference model of portfolio problem with public projects. *Kluwer Academic Publishers* (2003).
- [16] FISHER, H., AND THOMPSON, G. Probabilistic learning combinations of local job-shop scheduling rules. In *Factory Scheduling Conference*.
- [17] FISHER, H., AND THOMPSON, G. *Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules, Industrial Scheduling*. Prentice Hall, Inc, New Jersey, 1963.
- [18] GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* (1986).
- [19] KUMAR, R., BANKA, K., JOSHI, H., AND ROCKETT, I. Evolution of hyperheuristics for the biobjective 0/1 knapsack problem by multiobjective genetic programming. *GECCO* (2008).

- [20] LÓPEZ, E. *Incorporación de Preferencias en Algoritmos Evolutivos MultiObjetivo Utilizando Información de una Relación Borrosa de Sobreclasificación*. PhD thesis, Universidad Autónoma de Sinaloa, 2008. Culiacán, Sinaloa, México.
- [21] MARTELLO, S., AND TOTH, P. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, 1990.
- [22] MEDAGLIA, A., GRAVES, S., AND RINGUEST, J. Multiobjective evolutionary approach for linearly constrained project selection under uncertainty. *European Journal of Operational Research* (2007).
- [23] NEBRO, A., ALBA, E., AND LUNA, F. *Optimización multi-objetivo y computación grid*. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga, 2004.
- [24] NIÑO, J. *Optimización combinatoria. El problema de la mochila*, 2007.
- [25] PISINGER, D. A minimal algorithm for the 0-1 knapsack problem. *Operations Research* (1994).
- [26] PUCHINGER, J., RAIDL, R., AND PFERSCHY, U. The core concept for the multidimensional knapsack problem. - (2006).
- [27] PÉREZ, V. *Modelo Causal de Desempeño de Algoritmos Metaheurísticos en Problemas de Distribución de Objetos*. PhD thesis, Instituto Tecnológico de Ciudad Madero, 2007. Ciudad Madero, Tamaulipas, México.
- [28] QUIROZ, M. *Caracterización de los Factores de Desempeño de Algoritmos de Solución de BPP*. PhD thesis, Instituto Tecnológico de Ciudad Madero, 2009. Ciudad Madero, Tamaulipas, México.
- [29] SMITH, R., MESA, O., DYNER, I., JARAMILLO, P., PROVEDA, G., AND VALENCIA, D. *Decisiones con multiples objetivos e incertidumbre*. Universidad Nacional de Colombia, 2000.
- [30] SOUBEIGA, E. *Development and Aplication of Hyperheuristics to Personnel Scheduling*. PhD thesis, The University of Nottingham, 2003.
- [31] ZANAKIS, H., STELIOS, J., AND EVANS, A. Heuristic optimization: Why, when and how to use it. *Interfaces* (1981).