SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Ciudad Madero

INSTITUTO TECNOLÓGICO DE CUIDAD MADERO

DEPARTAMENTO DE POSGRADO

"POR MI PATRIA Y POR MI BIEN"

ALGORITMO DE PROCESAMIENTO CELULAR PARA SOLUCIÓN DEL PROBLEMA DE CARTERA DE PROYECTOS

OPCIÓN I

TESIS PROFESIONAL

Para obtener el Título de

Maestro en Ciencias de la Computación

Presenta

Ing. Jorge Alberto Cerecedo Cordoba
G08070774

Asesor

Dra. Claudia Guadalupe Gómez Santillán

Cd. Madero, Tamps. **Noviembre 2015**

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

"2015, Año del Generalísimo José María Morelos y Pavón"

Cd. Madero, Tamps; a 28 de Octubre de 2015.

OFICIO No.: U5.256/15
AREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

ING. JORGE ALBERTO CERECEDO CORDOBA
NO. DE CONTROL G08070774
P R E S E N T E

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

PRESIDENTE :	DR. RODOLFO ABRAHAM PAZOS RANGEL
SECRETARIO :	DRA. GUADALUPE CASTILLA VALDEZ
VOCAL :	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
SUPLENTE	DRA. LAURA CRUZ REYES
DIRECTORA DE TESIS :	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
CO-DIRECTORA DE TESIS:	DRA. LAURA CRUZ REYES

Se acordó autorizar la impresión de su tesis titulada:

"ALGORITMO DE PROCESAMIENTO CELULAR PARA SOLUCIÓN DEL PROBLEMA DE CARTERA DE PROYECTOS"

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

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

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

Ma. Yolanda Chávez Cinco

M. P. MARÍA YOLANDA CHÁVEZ CINCO
JEFA DE LA DIVISIÓN

S.E.P.
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACION
I T C M

c.c.p.- Archivo
Minuta

MYCHC 'NLCO ' jar

ALGORITMO DE PROCESAMIENTO CELULAR PARA SOLUCIÓN DEL PROBLEMA DE CARTERA DE PROYECTOS

I.S.C. Jorge Alberto Cerecedo Cordoba

Tesis presentada como requisito para optar al título de:

Maestro en ciencias de la computación.

Directora:

Dra. Claudia Guadalupe Gómez Santillán

Codirectora:

Dra. Laura Cruz Reyes

Línea de Investigación:
Optimización Inteligente

Resumen

El problema de selección de carteras de proyectos es una cuestión la cual enfrentan diversas organizaciones de múltiples ramas. Es un problema que no carece de importancia al ser vital para la gerencia de las organizaciones, una elección pobre de los proyectos puede llevar a derroches significativos de recursos.

La selección adecuada de los proyectos es una tarea compleja, se maneja como encontrar el balance entre los diferentes aspectos aportados por los proyectos contando además con una restricción de recursos.

En el presente trabajo se propone un algoritmo de procesamiento celular para la resolución de la tarea de seleccionar un conjunto reducido de carteras de proyectos, delimitados por las preferencias de un tomador de decisiones, el cual representa a la persona o grupo de personas que tomarán la decisión final. Además se incluyen pruebas experimentales donde se concluye que el algoritmo propuesto es un algoritmo de competitividad al compararse con trabajos relacionados.

Contenido

Lista de figuras

Lista de tablas

Capítulo 1. Introducción

En este capítulo se abordan los temas necesarios para introducir los algoritmos de procesamiento celular dando inicio con la motivación y la justificación del trabajo de tesis, también se incluyen los antecedentes de la investigación, el planteamiento de metas y las limitaciones para este documento.

1.1 Motivaciones

Idealmente las organizaciones buscan obtener el mayor beneficio de todas las operaciones que efectúan. Las operaciones son reflejadas en proyectos que a su vez son agrupados en carteras que compiten por presupuesto. Distribuir el presupuesto entre diferentes elementos que lo solicitan se convierte en una tarea compleja por diversas cuestiones como: ¿Cuenta con recursos suficientes para apoyar a los proyectos seleccionados?, ¿Qué se puede hacer para repartir equitativamente los recursos a diferentes áreas de necesidad?, ¿Cómo se justifica que la selección efectuada es la mejor cartera?

Una decisión de esta naturaleza no debe ser tomada a la ligera, si la búsqueda de la mejor cartera es para el beneficio de una organización entonces pudieran surgir las siguientes cuestiones ¿Cómo influyen las necesidades de la organización en la selección? y ¿Cómo influyen las preferencias de la organización en la búsqueda? Entonces, para llegar a un estado de conformidad en la organización es necesario incluir un sistema de preferencias, el cual se encargue de asegurar que las carteras sean del agrado de la persona o personas que tomarán la decisión final.

La o las personas encargadas para la toma de decisiones son conocidas como decisor (Decision Maker, DM), y sus deseos, preferencias y opiniones deben ser considerados en la búsqueda de una cartera integral.

Para identificar la mejor cartera no es suficiente realizar una comparación de los proyectos de manera individual, sino que se deben comparar grupos de proyectos para poder seleccionar la cartera que realice una aportación mayor a los objetivos de la organización.

Resulta de vital importancia el desarrollo de tecnologías diseñadas para resolver de una manera efectiva el problema de selección de cartera de proyectos (PSP, Portfolio Selection Problem), por este motivo, se ha desarrollado este trabajo de tal manera que la implementación de un algoritmo de procesamiento celular permita ampliar la gama de herramientas disponibles.

El algoritmo de procesamiento celular (Celular Processing Algorithm, CPA) permitirá manipular un conjunto mayor de soluciones en un menor tiempo comparado con un algoritmo genético simple, debido a la subdivisión del trabajo en células de procesamiento (Processing Cell, PCell), aunado al uso de preferencias del DM para guiar la búsqueda del algoritmo con el fin de obtener mejores carteras.

1.2 Justificación

La distribución financiera destinada a proyectos por parte de las organizaciones es una tarea trascendental, aún más cuando el presupuesto es limitado. Es precisa la ayuda de herramientas capaces de brindar apoyo a la decisión que reduzcan el espacio inmensurable de posibles carteras a un grupo de unas cuantas.

El trabajo actual brinda una herramienta para utilizar técnicas metaheurísticas con el fin de restringir el catálogo de carteras “buenas” a solo unas cuantas, de manera que sea una tarea más sencilla para el DM seleccionar la "mejor" cartera (de acuerdo a sus criterios).

Aunque actualmente existen trabajos relacionados con la cartera de proyectos, estos no presentan un enfoque de procesamiento paralelo. Este trabajo está encaminado a explorar este enfoque a través del algoritmo propuesto, el cual logra una ejecución pseudoparalela con el uso de "células" de procesamiento. Esto permite atacar el problema desde diferentes ángulos, produciendo incógnitas como las siguientes: ¿Cómo influye la división de la población en el resultado final? o ¿Qué efectos produce evolucionar independientemente las poblaciones? Se busca resolver estas cuestiones en el documento presente.

1.3 Antecedentes

En esta investigación, el problema de optimización de carteras es estudiado como un problema de decisión en donde el objetivo es brindar al DM un conjunto de soluciones que den un mejor compromiso a sus preferencias; los primeros trabajos en esta área daban como resultado un conjunto de soluciones con un cardinal muy grande, lo que dificultaba al DM la elección del mejor compromiso o de un compromiso satisfactorio.

Debido a la magnitud del problema es inviable el uso de métodos exactos de una manera práctica, por ello, se opta comúnmente por el uso de técnicas metaheurísticas. El algoritmo de procesamiento celular brinda la oportunidad de explorar estas alternativas, donde su base fundamental es el uso de metaheurísticas y la cooperación de las mismas con un solo fin; explorar la capacidad del algoritmo para resolver el problema referido.

Este proyecto es parte de un macro-proyecto de la red académica de Optimización y Apoyo a la Decisión (OAD). Dicha red está formada por cuatro centros de investigación: Universidad Autónoma de Nuevo León (UANL), Universidad Autónoma de Sinaloa (UAS), Universidad de Occidente (UDO) y el Instituto Tecnológico de Ciudad Madero (ITCM).

La red tiene como finalidad explorar las posibilidades de colaboración conjunta en torno al problema de selección de cartera de proyectos, además de propiciar un acercamiento fructífero entre los miembros de las universidades (Balderas, 2012). La Figura 1 muestra las áreas de aportación de cada institución.

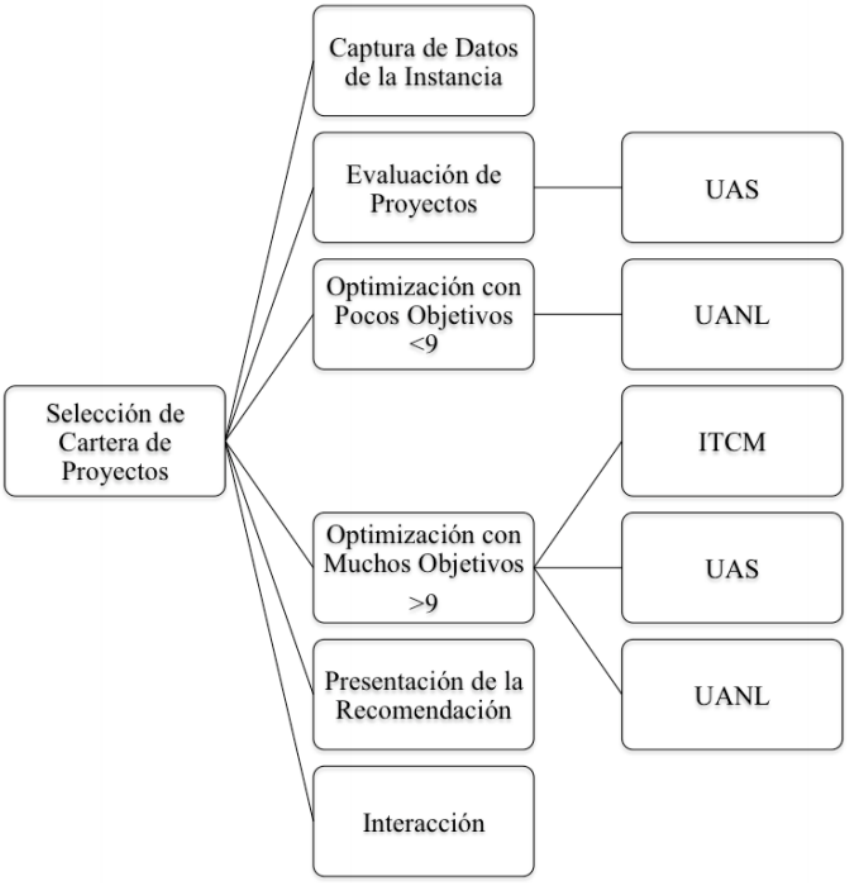

Figura 1. Desarrollo de la red académica OAD (Balderas, 2012)

1.4 Objetivo general

Diseñar e implementar un algoritmo de procesamiento celular con capacidad de solución del problema de cartera de proyectos, que alcance la competitividad de los trabajos reportados en el estado del arte.

1.5 Objetivos específicos

- Analizar los Metaheurísticos poblacionales para la selección de estrategia más adecuada.
- Diseñar las celdas de procesamiento celular.

- Diseñar el método de comunicación entre células.
- Diseñar el método de detección de estancamiento.
- Implementar el método de solución integrando las estrategias anteriores, para que obtengan un grupo de soluciones del mejor compromiso.
- Aplicar un método de ajuste de parámetros al algoritmo celular.
- Comparar el desempeño del algoritmo propuesto con el del algoritmo del estado del arte desarrollado por Rivera, Cruz, Fernández, Gómez y Pérez (2014).

1.6 Alcances y delimitaciones

Alcances:

- El algoritmo deberá tratar instancias de gran escala (de 100 a 500 proyectos).

Delimitaciones:

- Sólo se utilizará el modelo reportado por Rivera et al. (2014).
- Sólo se usarán algoritmos poblacionales en la construcción de las células.
- En la experimentación sólo se compararán los resultados contra Rivera et al. (2014).
- Sólo se usarán estrategias de diseño multi-objetivo a posteriori.

1.7 Organización del documento

En el Capítulo 2, se presentan todas las definiciones necesarias para la comprensión del trabajo, aquí se explican los conceptos relacionados con la optimización, optimización multi-objetivo, el problema de decisión evaluado, formas de interpretarlo e incluso posibles formas de solución de estos problemas. Enseguida en el Capítulo 3, se muestran los trabajos de mayor relevancia para esta tesis, se abarcan algoritmos que dan solución a la cartera de proyectos guiados con y sin preferencias. Continuando con el Capítulo 4, se presenta la propuesta de esta tesis; un algoritmo de procesamiento celular para el problema de la cartera de proyectos; el contenido muestra el algoritmo utilizado y detalles del mismo como lo son

las características, heurísticas usadas, estrategia de selección de padres, forma de recombinación de soluciones y técnicas de mutaciones. En el Capítulo 5, se muestran las experimentaciones de este trabajo y sus configuraciones en preparación de las conclusiones. Finalmente en el Capítulo 6, las conclusiones son descritas recapitulando las cualidades positivas y negativas del Algoritmo de Procesamiento Celular aplicado a la cartera de proyectos, además se manejan recomendaciones para futuros trabajos y se mencionan algunas líneas de investigación abiertas.

Capítulo 2. Marco Teórico

En este capítulo se introducen y describen algunos elementos de la teoría y la terminología en el dominio de esta investigación. Primeramente, se presenta el marco teórico asociado a los problemas de optimización mono-objetivo, bi-objetivo y multi-objetivo. Se prosigue describiendo los conceptos asociados a las estrategias existentes de solución de estos problemas. Finalmente, se concluye con una breve exposición acerca de métodos de solución.

2.1 Problemas de Optimización

La optimización es uno de los temas importantes en la teoría de la computación, surge de la necesidad de resolver problemas "difíciles" o NP-Duros (Garey & Johnson, 1979), el principal beneficio de optimizar un problema de decisión se refleja en los costos asociados al problema. Una efectiva exploración del espacio de soluciones permite ahorros considerables en la inversión final.

El término optimización, según Kalyanmoy Deb (Deb, Pratap, Agarwal, & Meyarivan, 2002), es la acción de encontrar una o más soluciones factibles que corresponden a valores extremos de uno o más objetivos. Cuando un problema de optimización involucra solamente una función objetivo, se le llama optimización mono-objetivo. En el caso donde el problema de optimización involucra dos objetivos se le conoce como bi-objetivo y a más de dos funciones objetivo, se le conoce como optimización multi-objetivo (Deb K. , 2001) (Collette & Siarry, 2013).

Además se puede considerar una subdivisión de los problemas multi-objetivos dividiendolos en problemas de pocos objetivos (de dos a tres) y problemas de muchos objetivos (cuatro o más). La separación de multi-objetivo a muchos objetivos se debe a que a partir de cuatro objetivos el comportamiento de los problemas cambia debido a la alta

dimensionalidad de los mismos, por ello algunos autores consideran esta clasificación como la más acertada (Ishibuchi, Tsukamoto, & Nojima, 2008).

2.2 Problemas de Optimización Multi-objetivo

En términos del mundo real, los problemas mono-objetivo son raramente aplicados, una representación confiable de los problemas comunes son los problemas multi-objetivo, en éstos, las funciones objetivo pueden o no estar en conflicto y varían desde dos objetivos a cientos de objetivos, una aproximación de estos problemas se da al optimizar el primer objetivo, luego el segundo y así sucesivamente hasta manipular todos los objetivos (Benayoun, De Montgolfier, Tergny, & Laritchev, 1971).

Una vez definido un problema multi-objetivo se debe cuestionar que es lo que se considerará como una solución óptima. En un *problema de optimización multi-objetivo* (MOP, Multiobjetive Optimization Problem) se busca un conjunto de soluciones que presenten un equilibrio óptimo entre los diferentes objetivos (Coello, Van Veldhuizen, & Lamont, 2002).

Para poner en contexto un problema multi-objetivo considérese un ejemplo; el problema de la mochila (Garfinkel & Nemhauser, 1972) (Karp, 1972) consiste en seleccionar un conjunto de "objetos" con algunas características (peso, valor, importancia, volumen, etc.) para ser introducidos en un contenedor (mochila), resulta evidente que al minimizar el peso de la mochila la solución sería una mochila sin elementos la cual carece de valor alguno, por el contrario, si maximizamos únicamente el valor de la mochila consideraríamos una mochila con todos los objetos (considerando que no existen restricciones) por lo que no existe un problema a resolver.

Entonces generalmente no existe una solución única que satisfaga un problema multi-objetivo debido a que los objetivos normalmente se encuentran en conflicto, y dentro del conjunto de soluciones óptimas todas las opciones son igual de válidas (al menos en el aspecto teórico) (Coello, Van Veldhuizen, & Lamont, 2002).

Un problema de optimización multi-objetivo se define como: "Encontrar un vector de variables de decisión que satisfaga las restricciones dadas y optimice un vector de funciones cuyos elementos representan las funciones objetivo" (Osyczka, 1985).

Esas funciones forman una descripción matemática de los criterios a optimizar y generalmente se encuentran en conflicto entre sí. Por lo tanto, el término *optimizar* significa encontrar las soluciones que darían valores aceptables para todas las funciones objetivo.

2.2.1 Variables de decisión

Son aquellos valores numéricos que se eligen para un problema de optimización. Estos valores pueden ser denotados como: $x_j \; para \; j = 1,2, \dots, n$. Así entonces, el vector de n variables de decisión $\vec{x}$ es representado por: $\vec{x} = [x_1, x_2, \dots, x_n]^T$ (Santana, 2004).

2.2.2 Restricciones

En la mayoría de los MOP existen restricciones emulando las características particulares del entorno o de los recursos disponibles. Estas restricciones (Santana, 2004) están representadas de forma matemática como:

$$\text{desigualdades} \quad g_i(\vec{x}) \leq 0; i = 1,2 \dots\dots, m \quad \text{ó}$$

$$\text{igualdades} \quad h_j(\vec{x}) = 0; j = 1,2 \dots\dots, p$$

2.2.3 Modelo matemático de un problema multi-objetivo

Formalmente, en (Serrano, 2007) un problema multi-objetivo busca el vector de variables de decisión $\vec{x} = [x_1, x_2, \dots, x_n]^T$ que optimice a:

$$F(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})], \qquad f_i: \mathbb{R}^n \rightarrow \mathbb{R} \tag{1}$$

Sujeto a: $$g_i(\vec{x}) \leq 0; \quad i = 1,2 \ldots\ldots, m$$

$$h_j(\vec{x}) = 0; \quad j = 1,2 \ldots\ldots, p$$

Dónde:

k es el número de funciones objetivo.

n es el número de variables de decisión.

m es el número de restricciones de desigualdad.

p es el número de restricciones de igualdad.

En otras palabras, se intenta determinar el conjunto de todos aquellos números que satisfagan las restricciones y que optimicen todas las funciones objetivo. Las restricciones son las que definen la región factible del problema y cualquier vector $\vec{x}$ que esté en esta región se considera como una solución factible (Santana, 2004). Existen tres tipos de problemas multi-objetivo:

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

En el conjunto de soluciones óptimas, si se aprecia una degradación de un objetivo se reflejará una mejora en otro, esto se puede asegurar gracias a los conceptos de optimalidad de Pareto y frente de Pareto.

2.2.4 Optimalidad de Pareto y Frente de Pareto

El concepto más comúnmente aceptado de "óptimo" en un problema multi-objetivo se propuso por el economista Vilfredo Pareto, él desarrolló el concepto de óptimo para la situación donde no se puede beneficiar algún objetivo sin afectar otro y busca un punto o conjunto de puntos (soluciones) que permiten obtener el máximo beneficio hasta el punto que empieza a afectar otros objetivos (Goldberg, 1989) (Coello C. A., 2015).

Se dice que una solución 'u' domina a una solución 'v' (suponiendo maximización) si y solo si se cumplen las siguientes condiciones:

1. $f_i(\vec{u}) \geq f_i(\vec{v}),\ \forall\, i \in \{1,2,\dots,k\}$
2. $f_i(\vec{u}) > f_i(\vec{v}),\ \exists\, i \in \{1,2,\dots,k\}$

Esta relación es llamada Dominancia de Pareto y se representa como $\succ$. La Figura 2 ilustra este concepto. Un punto en el espacio de búsqueda $\overrightarrow{x^*} \in \Omega$ se considera óptimo de Pareto si no existe otro $\vec{x} \in \Omega$ que lo domine.

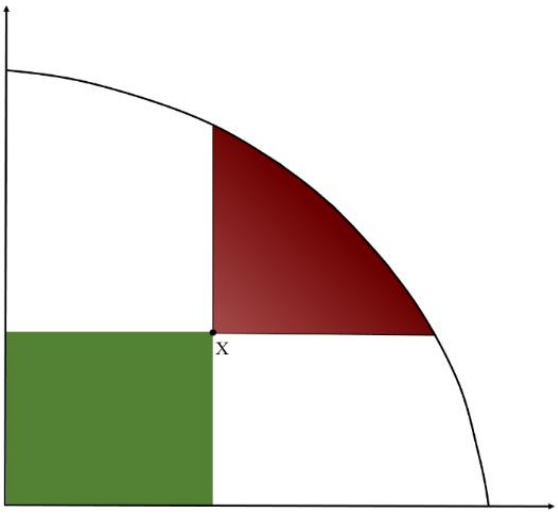

Figura 2. Dominancia de Pareto para un punto X en el espacio (Maximización), Todas las soluciones en la región superior derecha a X mientras que los elementos de la región inferior izquierda son dominados por X.

Para un problema multi-objetivo dado $f_i(\overrightarrow{x^*})$, el conjunto de óptimos de Pareto (P^*) se define como $P^* := \{x \in \Omega \mid \neg\exists\, x^* \in \Omega, x^* \preceq x\}$. Es decir, una solución pertenece a los óptimos de Pareto si no existen soluciones que mejoren la solución en al menos un objetivo sin empeorar en los demás. La Figura 3 muestra una representación gráfica de la no dominancia para tres soluciones (suponiendo maximización).

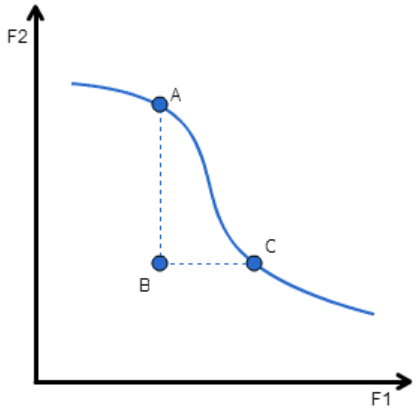

Figura 3. Representación de No Dominancia con dos objetivos F1 y F2, El punto A domina a B, C domina a B. Por el contrario, A no domina a C ni C domina a A.

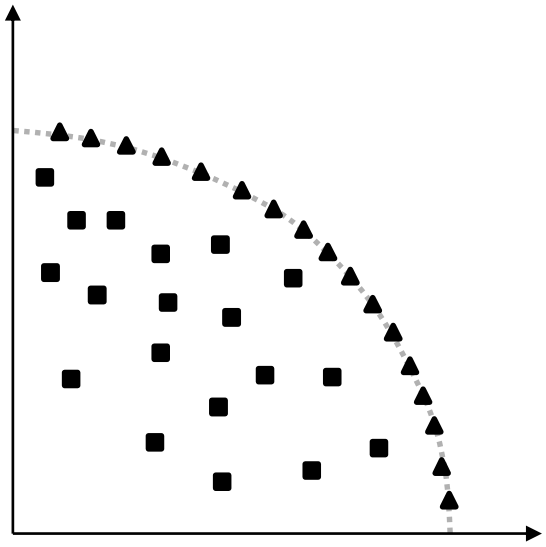

Figura 4. Ejemplo de Frente de Pareto (Triángulos), y su dominancia sobre soluciones del Espacio (Cuadrados).

Entonces la representación de las funciones objetivo de las soluciones no dominadas son llamadas Frente de Pareto y se define como $PF^* := \{f(x^*) \mid x^* \in P^*\}$ como se ilustra en la Figura 4.

El frente de Pareto nos permite discriminar las soluciones con los mejores valores objetivo de las demás. Para dar una idea de la calidad del frente existen técnicas llamadas métricas de desempeño con la función de medir características del frente.

2.3 Métricas de desempeño

2.3.1 Generalized Spread

Generalized Spread (Zhou, Jin, Zhang, Sendhoff, & Tsang, 2006) es una métrica de desempeño la cual nos da una idea de la dispersión o diversidad de las soluciones a través del frente. Funciona por medio del cálculo de las distancias más cortas de todos los puntos del frente (FP') a los puntos del frente óptimo (FP^*).

$$GS(FP', FP^*) = \frac{\sum_{i=1}^{k} d(e_i, FP') + \sum_{i=1}^{|FP^*|} |d(FP_i^*, FP') - \bar{d}|}{\sum_{i=1}^{k} d(e_i, FP') + |FP^*|\bar{d}} \qquad (2)$$

Donde $\{e_1, e_2, \dots, e_k\}$ son k soluciones extremas en FP^* y $\bar{d}$ es:

$$\bar{d} = \frac{1}{|FP^*|} \sum_{i=1}^{|PF^*|} d(FP_i^*, FP') \qquad (3)$$

La distancia de un punto a un conjunto se conoce como la menor distancia encontrada entre los puntos del conjunto y el punto.

El *Spread* es inversamente proporcional a la calidad del frente, es decir, un frente óptimo en diversidad tendrá un valor de cero.

2.3.2 Generational Distance

La métrica de convergencia *Generational Distance* (Van Veldhuizen & Lamont, 1998) estima la distancia que existe entre el frente aproximado (FP') y el frente óptimo (FP^*). El cálculo es realizado tomando las distancias de los n puntos de FP' al punto más cercano correspondiente al FP^*, el promedio de todas las distancias arroja el valor por esta métrica, la Fórmula 4 muestra el proceso.

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \quad (4)$$

Donde d_i es la distancia euclidiana del punto i al más cercano en FP^*. Al igual que en la métrica *Generational Spread*, la *Generational Distance* es inversamente proporcional a la calidad del frente, es decir, un frente óptimo en convergencia tendrá un valor de cero.

2.4 Problema de Cartera de Proyectos

Un proyecto es un proceso temporal, único e irrepetible que persigue un conjunto específico de objetivos (Carazo, y otros, 2010). Un proyecto es una unidad atómica indivisible y en el contexto del problema actual persigue objetivos específicos y tiene un costo asociado; además, un proyecto puede pertenecer a un área y una región.

Una cartera de proyectos es un conjunto de proyectos realizados en el mismo lapso de tiempo (Carazo, y otros, 2010). Por esta razón, los proyectos que están en una misma cartera, comparten y compiten por los recursos disponibles en la organización. Se deben comparar grupos de proyectos para poder seleccionar la cartera que realiza una aportación mayor a los objetivos de la organización.

Para abordar la solución de este problema primeramente se debe definir formalmente.

2.5 Definición formal de Selección Cartera de Proyectos

La selección de cartera de proyectos es un problema de programación lineal entera (Schrijver, 1998) de optimización multi-objetivo (Deb K. , 2001). Una representación general del problema es la siguiente:

Suponga una cartera de n proyectos $\vec{x} = (x_1, x_2, \dots x_n)$ $donde$ $\vec{x} \in \{0,1\}^n$ para el cual:

$$\vec{x}_i = \left\{ \begin{array}{l} 1 \; si \; el \; proyecto \; i \; pertenece \; a \; la \; cartera \\ 0 \; si \; el \; proyecto \; i \; no \; pertenece \; a \; la \; cartera \end{array} \right\}$$

Además considérese un presupuesto B y un vector de costos $c = (c_1, c_2, \dots c_n)$ donde $c_i \in \mathbb{N}$ representa el costo de seleccionar el proyecto i en la cartera. Asimismo un par de vectores $a = (a_{11}, a_{12}, \dots, a_{1m}, a_{21}, \dots, a_{nm})$ y $r = (r_{11}, r_{12}, \dots, r_{1l}, r_{21}, \dots, r_{nl})$ donde $a_{ij} \in \{0,1\}$ representa si el proyecto i pertenece al área j y $r_{ik} \in \{0,1\}$ representa si el proyecto i pertenece a la región k.

Entonces el problema se resuelve al encontrar una cartera $\vec{x}$ que satisfaga las siguientes restricciones:

$$\sum_{i=0}^{n} x_i c_i \leq B \tag{5}$$

$$A_1^L \leq \sum_{i=0}^{n} x_i c_i a_{i1} \leq A_1^U \qquad R_1^L \leq \sum_{i=0}^{n} x_i c_i r_{i1} \leq R_1^U$$

$$A_2^L \leq \sum_{i=0}^{n} x_i c_i a_{i2} \leq A_2^U \qquad R_2^L \leq \sum_{i=0}^{n} x_i c_i r_{i2} \leq R_2^U$$

$$\vdots \qquad\qquad \vdots$$

$$A_m^L \leq \sum_{i=0}^{n} x_i c_i a_{im} \leq A_m^U \tag{6} \qquad R_l^L \leq \sum_{i=0}^{n} x_i c_i r_{ik} \leq R_l^U \tag{7}$$

Donde:

$$A_j^U = Límite\ superior\ de\ inversión\ en\ área\ j$$

$$A_j^L = Límite\ inferior\ de\ inversión\ en\ área\ j$$

$$R_j^U = Límite\ superior\ de\ inversión\ en\ región\ j$$

$$R_j^L = Límite\ inferior\ de\ inversión\ en\ región\ j$$

Asimismo, incluye el vector $b = (b_{11}, b_{12}, \dots, b_{1o}, b_{21}, \dots, b_{no})$ donde b_{ij} indica el beneficio aportado en el objetivo j por el proyecto i. Entonces, la función objetivo se define como:

$$f(x) = (f_1(\vec{x}), f_1(\vec{x}), \dots, f_p(\vec{x})) \quad (8)$$

Donde

$$f_i(x) = \sum_{j=0}^{n} x_j b_{ji} \quad (9)$$

La mejor solución compromiso se define al resolver $max_{x \in R_F}(f(\vec{x}))$ donde R_f es el espacio de soluciones factibles. La resolución del problema nos brinda un conjunto de soluciones consideradas óptimas, este es un conjunto de soluciones que crece exponencialmente con la dimensionalidad del problema, para facilitar la toma de decisión se han definido estrategias las cuales delimitan el frente óptimo a través de preferencias.

2.6 Estrategias de solución para problemas multi-objetivo

La optimización multi-objetivo está fuertemente relacionada con la articulación de preferencias y cuando se toman; esto puede ser: nunca, antes, después o durante el proceso de optimización (Andersson, 2000).

Históricamente se han creado diferentes metodologías para resolver un problema multi-objetivo: sin articulación de preferencias, articulación de preferencias a priori, articulación de preferencias a posteriori y articulación de preferencias progresivas (Andersson, 2000). En la Figura 5 se observa la clasificación de las diferentes metodologías de solución.

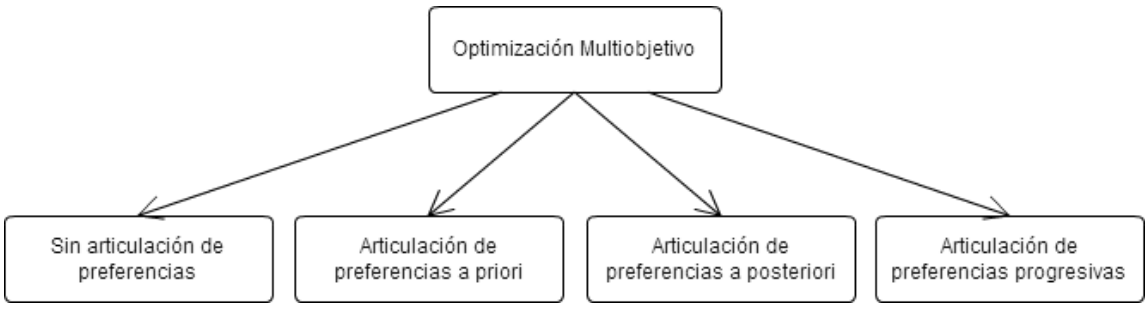

Figura 5. Clasificación de los métodos de articulación de preferencias (Andersson, 2000).

2.7 Estrategias de solución para el problema de la cartera de proyectos con preferencias a priori

Los métodos de solución encontrados en la literatura hasta este momento se clasifican en: Función valor, Relaciones de preferencias difusas y Ranking de prioridades. La Figura 6 muestra esta clasificación.

A continuación se describirán los métodos de evaluación para la toma de decisiones.

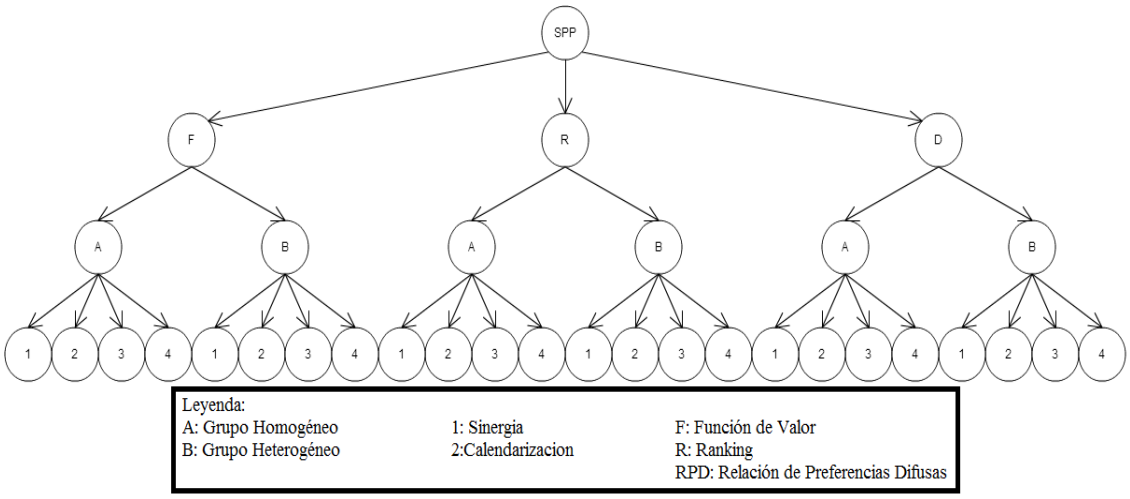

Figura 6. Métodos de evaluación para toma de decisiones (Bastiani M. , 2013).

2.7.1 Función Valor

El método función de valor es una estrategia que permite transformar el desempeño en valor y tomar la decisión en términos del valor que nos aporta una determinada opción y no únicamente en términos de su desempeño. En preparación de esta técnica es necesario conocer o modelar la función valor para optimizar a través de ella (Bastiani M. , 2013).

2.7.2 Ranking de prioridades

El Ranking de prioridades es una técnica de medida de beneficio, el cual es denominado Método basado en pesos y ordenación. Este modelo permite determinar una jerarquía u

orden (ranking) de preferencia de proyectos candidatos, basado en un conjunto de criterios, para que posteriormente el DM, en función de los recursos disponibles seleccione los proyectos en orden, hasta agotar los recursos. Para establecer un ranking entre proyectos, esta técnica evalúa cada uno de los proyectos en función de los criterios que deben tenerse en cuenta en el proceso de decisión (Fernández, Gómez, & Guerrero, 2008).

2.7.3 Relación de Preferencias Difusas (RPD)

Este modelo de preferencias permite utilizar la metodología de las técnicas fuzzy o difusas para tratar la información sobre las apreciaciones difusas del experto sobre las alternativas y sus comparaciones. En algunas ocasiones un DM no está seguro de sus preferencias por lo cual, dentro de este modelo, estas preferencias se pueden modelar con relaciones de preferencia ajustadas a la lógica difusa; a estas relaciones se les denomina "Relaciones de Preferencia Difusa" (Fernández, López, & Coello, 2010) (García, 2010).

En este modelo en lugar de que se tengan dos valores para hacer una representación de la relación de preferencia, se puede tomar cualquier valor comprendido en el intervalo cerrado [0, 1]; es decir que si el DM no está seguro de preferir B a A, se puede asumir un valor por ejemplo, de 0.8. Este es un valor que favorece al candidato B, este valor puede ser mayor o menor, dependiendo de qué tan fuerte o débil sea para el DM la relación de preferencia entre B y A (Díaz M. , 2006). Esta estrategia es la abordada en el problema de investigación actual.

2.7.4 Integración de preferencias del Tomador de Decisiones

En el trabajo presente aun cuando se encuentra un Frente de Pareto Óptimo esto no resuelve el problema, aun se tiene que hacer una elección de la solución que más se adapta a las preferencias del DM. Un problema multi-objetivo puede tener un frente de Pareto que varía en magnitud desde un par de soluciones a miles de soluciones igualmente óptimas bajo la perspectiva de optimización de Pareto (Coello, Van Veldhuizen, & Lamont, 2002).

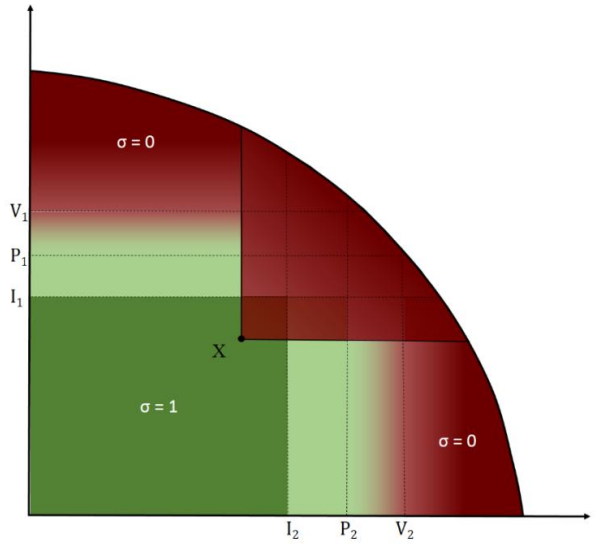

Figura 7. Representación del valor del grado de verdad de la afirmación "x es al menos tan buena como y" bajo el método ELECTRE III bi-objetivo.

La propuesta de Fernández en (Fernández, López, & Coello, 2010) está basada en el Sistema de preferencias descritas por Roy (Roy, 1991). Este sistema de preferencias incluye las relaciones descritas en la Tabla 1 y es necesario el grado de verdad de la afirmación "x es al menos tan buena como y" y puede ser calculado con métodos probados como ELECTRE, una representación del cálculo de sigma para un caso bi-objetivo se muestra en la Figura 7.

Relación	Nomenclatura	Significado	Condiciones de la relación
Preferencia Estricta	xPy	"x se prefiere estrictamente sobre y"	$x\ domina\ y$ $\vee (\sigma(x,y) \geq \lambda \wedge \sigma(y,x) < 0.5)$ $\vee (\sigma(x,y) \geq \lambda \wedge [0.5 \leq \sigma(y,x) < \lambda] \wedge [\sigma(x,y) - \sigma(y,x)] \geq \beta)$
Indiferencia	xIy	"x es indiferente a y"	$(\sigma(x,y) \geq \lambda \wedge \sigma(y,x) \geq \lambda)$ $\wedge (\lvert\sigma(x,y) - \sigma(y,x)\rvert \leq \epsilon)$
Preferencia Débil	xQy	"x es débilmente preferido sobre y"	$(\sigma(x,y) \leq \lambda \wedge \sigma(x,y) \geq \sigma(y,x))$ $\wedge (\neg xPy \wedge \neg xIy)$
Incomparable	xRy, yRx	"DM no puede discernir entre estas soluciones"	$\sigma(x,y) < 0.5 \wedge \sigma(y,x) < 0.5$
k-preferencia	xKy	"x tiene una k-preferencia sobre y"	$(0.5 \leq \sigma(x, y) \leq \lambda)$ $\wedge (\sigma(y, x) < 0.5)$ $\wedge (\sigma(x, y) - \sigma(y, x) > 2)$

Tabla 1. Relaciones de preferencia (Roy, 1991) (Fernández, Gómez, & Guerrero, 2008).

Entonces para el conjunto de soluciones factibles O, se definen los siguientes conjuntos:

$S(O,x) = \{y \in O \mid yPx\}$ está compuesto de las soluciones que se prefieren estrictamente sobre x.

$NS(O) = \{x \in O \mid S(O,x) = \emptyset\}$ es conocida como la frontera no superada, donde una solución se considera *no superada* si no existe otra solución que se prefiera estrictamente sobre ella.

$W(O,x) = \{y \in NS(O) \mid yQx \wedge yKx\}$ está compuesto de las soluciones que son débilmente superadas por la frontera no superada.

$NW(O) = \{x \in O \mid W(O,x) = \emptyset\}$ es conocida como la frontera débilmente no superada, donde una solución se considera *no superada débilmente* si no existe otra solución que se prefiera débilmente sobre ella.

Además, se incluye una medida para identificar las preferencias del DM dentro de NS llamada flujo neto (Fernández, López, & Coello, 2010). El flujo neto asociado a una alternativa a ∈ C está definido como:

$$F_n(x) = \sum_{y \in NS(O)\{x\}} [\sigma(x,y) - \sigma(y,x)] \qquad (10)$$

Donde $F_n(x) > F_n(y)$ indica una preferencia del DM de x sobre y.

Fernández propone en (Fernández, López, & Coello, 2010) la minimización lexicográfica del vector $(|S(O,x)|, |W(O,x)|, |F(O,x)|)$ como función objetivo y demostró que la mejor solución compromiso es una solución que es no dominada y a su vez no superada.

Independientemente de la estrategia de solución del problema, es necesario el mecanismo algorítmico de navegación del espacio de soluciones con el cual daremos solución del problema de decisión.

2.8 Métodos de solución de problemas de optimización

Existe diversidad de propuestas algorítmicas para la solución de problemas, un algoritmo puede ser de solución exacta o aproximada. Un algoritmo exacto garantiza la obtención de la solución óptima, por el contrario, un algoritmo aproximado no puede garantizar la obtención del óptimo (Woeginger, 2003).

2.8.1 Métodos Exactos

Los métodos exactos buscan la mejor solución (óptimo) posible a un problema determinado, con este propósito estos métodos navegan por todo el espacio de soluciones factibles; son considerados imprácticos en problemas de gran tamaño dado los altos tiempos de cómputo necesarios para su terminación.

Entre los métodos exactos se puede nombrar métodos de programación lineal entera, Branch and Bound e incluso búsquedas por fuerza bruta.

2.8.2 Métodos basados en algoritmos metaheurísticos

Los algoritmos metaheurísticos han probado ser una herramienta eficaz en la búsqueda de aproximaciones al resolver problemas de optimización. El término metaheurística fue introducido por (Glover, 1986).

Los problemas NP-Duros no pueden abordarse de forma realista con algoritmos exactos dada la gran magnitud de combinaciones que conforman el espacio de búsqueda, por consecuencia el uso de los algoritmos aproximados es aceptado en la resolución de problemas NP-Duros (Garey & Johnson, 1979). Al utilizar algoritmos aproximados sacrificamos la certeza de optimalidad a cambio de soluciones aproximadas, pero en tiempos de respuesta relativamente rápidos.

Los algoritmos metaheurísticos son una técnica utilizada para la solución de problemas de optimización combinatoria ya que exploran grandes espacios de soluciones, encontrando

soluciones aproximadas en tiempos razonables para problemas que son difíciles de resolver (Feo & Resende, 1995) (Resende & González, 2003).

Diversas clasificaciones de algoritmos metaheurísticos pueden ser encontradas en (Blum & Roli, 2003), algunas clasificaciones son las siguientes:

- Inspiradas en la naturaleza o no inspiradas en la naturaleza.
- Basadas en poblaciones o basadas en solución única.
- De función dinámica o función estática.
- De una o varias estructuras de vecindad.

Existe infinidad de algoritmos metaheurísticos para resolución de problemas complejos, uno de las metaheurísticos más utilizados son las llamados algoritmos genéticos.

2.8.3 Algoritmos Genéticos

Los algoritmos genéticos (Genetic Algorithm, GA) o algoritmos genéticos secuenciales, fueron introducidos en (Goldberg, 1989) y son usados ampliamente para la resolución de problemas combinatorios de alta complejidad mostrando buenas soluciones. Se clasifican como metaheurísticos y son inspirados en la selección natural y la genética. Buscan obtener soluciones “buenas” a través de la manipulación de una población (conjunto de soluciones), dicha población es evaluada y las mejores soluciones son seleccionadas para un proceso de cruza; el derivado de la cruza es una nueva generación (un nuevo conjunto de soluciones).

Un operador de mutación suele ser implementado pero se considera un evento suplementario, el proceso de selección y cruza pretende guiar la búsqueda hacia soluciones de calidad. Estos algoritmos están basados en el proceso darwiniano de evolución donde buenos individuos sobreviven y malos individuos perecen.

Las soluciones generalmente están compuestas por un vector el cual se conoce como cromosoma; este representa las variables del problema y pueden variar en tamaño y tipo dependiendo del problema. El pseudocódigo de un algoritmo genético general se aprecia en el Algoritmo 1.

Algoritmo 1. Estructura general de un algoritmo genético (Goldberg, 1989).

```
1    Inicio
2        Mientras Condición de parada no se cumpla
3           Evaluar P
4           Padres ← Seleccionar N/2 pares de soluciones
5           Hijos ← Ejecutar operador de Cruza
6           Mutación (Hijos)
7           P ← Hijos
8        Fin Mientras
9        Mostrar P
10   Fin
```

2.8.4 Algoritmo genético distribuido

Los algoritmos genéticos distribuidos no manejan una población única, más bien manipulan un conjunto de subpoblaciones que evolucionan independientemente unas de las otras. Los algoritmos genéticos son una buena opción al manejar problemas de complejidad alta (Coello, Van Veldhuizen, & Lamont, 2002), varias técnicas se han implementado en la literatura con la intención de mejorarlos, con ese propósito fueron ideados los Algoritmos Genéticos Paralelos (Parallel Genetic Algorithms, PGAs). La filosofía detrás de estos algoritmos es la de "divide y vencerás", el proceso de división varía de algoritmo a algoritmo.

La clasificación más aceptada en la literatura es la de (Cantú-Paz, 1998) (Alba & Troya, 1999), la cual indica que existen tres diferentes clases de PGA: (1) población única global con GAs maestro-esclavos, (2) población única de grano fino o celulares y (3) de múltiples poblaciones o distribuidos.

- PGAs de población única (master-slave)

Este tipo de algoritmo consta de una sola población (Master) y hace las operaciones básicas del GA, los esclavos (slaves) solamente evalúan la aptitud de los individuos. En la Figura 8 se aprecia un esquema de este tipo de algoritmo.

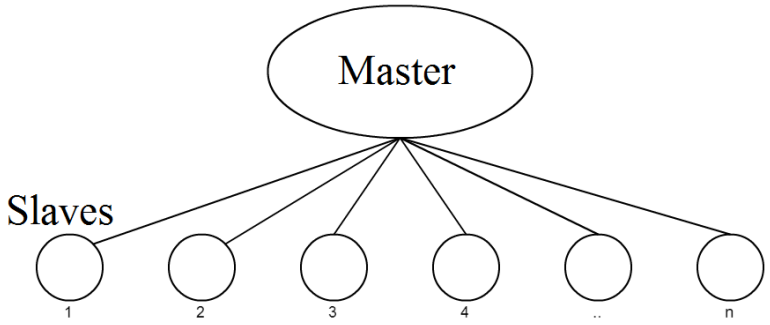

Figura 8. Un esquema de un PGA de población única (Master-Slaves) (Cantú-Paz, 1998).

- PGAs de población única de grano fino (PGAs Celulares).

Los GA comúnmente cuentan con una población panmíctica (no existe una estructura en la población), mientras que los PGAs Celulares cuentan con una estructura de población y es su principal estrategia de paralelismo. El algoritmo contiene las operaciones de cruza y mutación comunes de los GA, con la restricción de que las operaciones genéticas solo pueden ocurrir entre individuos de un mismo vecindario (soluciones cercanas) (Alba & Dorronsoro, Cellular genetic algorithms, 2009) (Díaz B. , 2007).

El solapamiento de los vecindarios provee al PGA celular un mecanismo implícito de migración ya que la información genética de las mejores soluciones se extenderán suavemente por toda la población, permitiendo mantener la diversidad genética en la población por más tiempo respecto a otros algoritmos evolutivos no estructurados (Díaz B. , 2007).

La utilización de vecindarios es posible gracias a la distribución de la población en forma de toroide. El grado de explotación y exploración puede ser controlado redefiniendo la estructura de población y vecindad (Alba & Dorronsoro, Cellular genetic algorithms, 2009) (Díaz B. , 2007), una visualización de la población se encuentra en la Figura 9c, en contraste a una población panmítica mostrada en la Figura 9a.

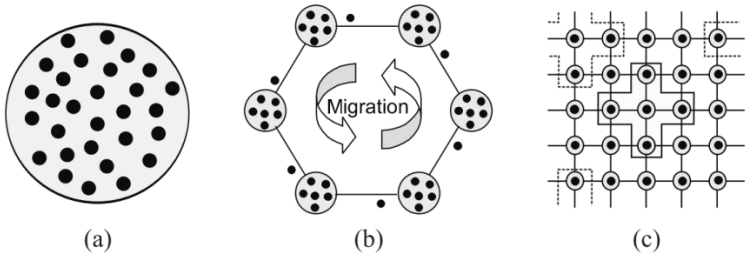

Figura 9. Un AG Panmíctico con sus individuos (puntos negros) en una sola población (a) Un PGA basado en islas(b) y un PGA celular(c) (Alba & Dorronsoro, Cellular genetic algorithms, 2009)

- PGAs distribuidos (islas).

Los Algoritmos Paralelos distribuidos tienen la principal característica de dividir la población en subpoblaciones o islas. El GA evoluciona las subpoblaciones con una comunicación entre poblaciones de forma periódica llamada migración, la Figura 9b muestra una representación del algoritmo.

Los principales parámetros para este tipo de algoritmos involucran la cantidad y frecuencia de la migración, la selección y remplazo de los individuos migrados y la topología de las conexiones entre islas (Alba & Dorronsoro, Cellular genetic algorithms, 2009).

Al surgimiento de este algoritmo la configuración indicaba un mismo plan reproductivo para todas las islas, pero existe una corriente en la literatura que consiste en ejecutar cada isla con diferente parametrización con el fin de explorar diferentes regiones del espacio de búsqueda; este tipo de algoritmo es conocido como heterogéneo (Alba & Dorronsoro, Cellular genetic algorithms, 2009) (Alba, Luna, & Nebro, 2004).

2.8.5 Algoritmo de procesamiento Celular

El algoritmo de Procesamiento Celular detallado en (Terán, Fraire, Carpio, Puga, & Martínez, 2012) consiste en una ejecución paralela en células de procesamiento. Cada célula de procesamiento es una heurística específica que navega el espacio de búsqueda, además, cada célula manipula conjuntos diferentes de soluciones o subpoblaciones, las células permiten comunicación entre ellas.

Las principales características son simplicidad, paralelismo y localidad. Simplicidad se refiere a que el procesamiento es realizado por un conjunto de células sin un control central de los individuos, paralelismo significa que las células funcionan de forma independiente y localidad implica que las células solo pueden comunicarse con las células cercanas.

Las PCell están distribuidas en un grafo representadas por los vértices y las aristas representan las conexiones entre las diferentes PCell. Generalmente la comunicación se efectúa cuando las células se encuentran en un estado de “estancamiento” logrando un intercambio de información genética.

Capítulo 3. Estado del Arte

En esta sección se presentan los trabajos relacionados con el problema de la cartera de proyectos. Se clasifican en dos tipos de algoritmos, los que no toman las preferencias del DM y las que toman en cuenta las preferencias del DM.

3.1 Algoritmos para PSP que no toman en cuenta las preferencias del DM

3.1.1 P-ACO

Doerner (Doerner, y otros, 2004) proponen P-ACO (Pareto Ant Colony Optimization), un algoritmo basado en la conocida metaheurística de Colonia de Hormigas para generar el frente de Pareto para las carteras de proyectos más eficientes. Cada hormiga de la colonia genera una cartera candidata, y la cantidad de feromona depositada por tal hormiga es inversamente proporcional a la cantidad de soluciones que la dominan. El algoritmo almacena las soluciones que nunca han sido dominadas las cuales al final formarán la aproximación del frente de Pareto. P-ACO es capaz de tratar con la sinergia entre proyectos, característica sobresaliente en el algoritmo.

3.1.2 SS-PPS

Carazo (Carazo, y otros, 2010) proponen SS-PPS (Scatter Search for Project Portfolio Selection), un algoritmo basado en búsqueda dispersa que aborda el problema de formación de carteras de proyectos. El algoritmo está especialmente diseñado para tratar con la calendarización de los proyectos, el financiamiento parcial así como la sinergia entre ellos. Carazo (Carazo, y otros, 2010) no sólo proponen el algoritmo, sino también un modelo matemático para el problema de calendarización y selección de carteras. SS-PPS tiene

básicamente dos etapas, en la primera genera un conjunto de puntos iniciales eficientes usando Búsqueda Tabú, y en la segunda etapa mejora el conjunto inicial mediante Búsqueda Dispersa.

3.2 Algoritmos que toman en cuenta las preferencias del DM.

3.2.1 NOSGA-II

Para dar solución al problema de selección de cartera de proyectos, (Fernández, López, & Coello, 2010) proponen NOSGA-II (Non-Outranked-Sorting Genetic Algorithm– II), un algoritmo basado en NSGA-II (Non-Dominated Sorting Genetic Algorithm – II) y NOSGA (Fernández, López, & Coello, 2010).

El principal cambio realizado al algoritmo, con respecto a NSGA-II, fue la incorporación de relaciones difusas con la finalidad de obtener las mejores soluciones de acuerdo a las preferencias del DM y no sólo el frente de Pareto como lo hace NSGA-II.

La incorporación de las preferencias se hace a través de las reglas preferencia definidas por el mismo Fernández, esto permite al algoritmo generar frentes de superación los cuales sustituyen a los frentes de dominancia característicos de NSGA-II.

Dado que en el trabajo de Fernández no se busca un frente disperso en la frontera de Pareto, él proporciona una métrica $\eta = |W(O,x)| + |F(O,x)|$ la cual nos permite hacer distinción entre soluciones con un mismo nivel de superación estricta. El Algoritmo 2 muestra el pseudocódigo de NOSGA-II.

Algoritmo 2. Pseudocódigo de NOSGA-II (Fernández, López, & Coello, 2010)

```
1    NOSGA-II (P)
2    Inicio
3      Inicializar la población P_t de tamaño L
4      Evaluar σ en P_t × P_t
5      Por cada x ∈ P_t, calcular |S(O,x)|
6      Generar Frentes de valores iguales de |S(O,x)|
7      Asignar a estos frentes un nivel basado en |S(O,x)|
```

```
8       Para cada nivel en Ci hacer:
9         Para cada x ∈ Ci, calcular |W(Ci,x)|, |F(Ci,x)|,η
10      Generar población de hijos Qt con tamaño L
11      Aplicar operadores genéticos
12      Mientras no cumplir condición de parada:
13        Rt = Pt ∪ Qt
14        Evaluar σ en Rt × Rt
15        Para cada x ∈ Rt
16          Calcular |S(O,x)|,|W(O,x)|,|F(O,x)| ,η
17          Asignar a x un nivel
18        Mientras |Pt+1| < L :
19          Añadir a Pt+1 siguiente solución con mejor nivel
20        Generar Población Qt+1 a partir de Pt+1
21        t = t + 1
22    Fin
```

3.2.2 HHGA-SPP

García (García, 2010) propone un algoritmo genético hiperheurístico para la selección de cartera de proyectos (Hyper-Heuristic Genetic Algorithm for SPP, HHGA-SPP) con un enfoque multicriterio en el cual cada individuo representa la secuencia en que las Heurísticas de Bajo Nivel (HBN) deben de ejecutarse. Las HBN son algoritmos que interactúan directamente con la solución de SPP. Ninguna de las HBN tiene por sí sola la capacidad de resolver SPP, pero cuando varias de ellas se realizan en secuencia, pueden satisfacer el problema de manera satisfactoria. La mejor secuencia es elegida por una Heurística de Alto Nivel, en este caso un algoritmo genético.

3.2.3 NO-ACO

En el trabajo de Rivera (Rivera, Cruz, Fernandez, Gómez, & Pérez, 2014) proponen NO-ACO, un algoritmo de colonia de hormigas para el problema de cartera de proyectos. El algoritmo utiliza hormigas que representan una solución para navegar el espacio de búsqueda simulando el proceso natural de las hormigas a través de una tabla de feromonas y simulando la evaporación de esta mediante las reglas de depósito de feromona y evaporación de feromona.

Este algoritmo a diferencia de otras colonias de hormigas incorpora relaciones difusas que reflejan las preferencias de DM (Fernández, López, & Coello, 2010) lo cual permite la localización de zonas de preferencia y guiar la búsqueda de las hormigas. Además de la formulación de la cartera de proyectos incluye otras características como el manejo de apoyo parcial, calendarización y sinergia entre proyectos.

3.2.4 SS-SSP

SS-SSP (Martínez, 2015) es un algoritmo basado en un Scatter Search, este utiliza los principios descritos por (Fernández, López, & Coello, 2010) con tal de implementar un sentido de dirección a la búsqueda a través del método de diversificación y actualización del método de referencia, además hace uso de un conjunto de referencia basado en la no superación.

3.2.5 ACO-SOP

Bastiani propuso un algoritmo de colonia de hormigas para la resolución del problema de cartera de proyectos públicos ACO-SOP (Ant Colony Optimization for Solving portfolio problems with Ordinal information about Projects), en el se utilizan reglas de evaporación de feromona y de depósito de la misma (Bastiani, Cruz, Fernandez, Gómez, & Rivera, 2015).

A diferencia de NO-ACO, esté usa como única fuente de información un ranking de los proyectos, funciona de este modo para ser empleado en los casos donde no existe mucha información de los mismos.

La mejor cartera es generada minimizando las discrepancias existentes entre la cartera seleccionada y una cartera "ideal" que se forma con los mejores proyectos según el ranking. El término discrepancia refleja el efecto negativo que exhorta el DM por el hecho que un proyecto, comparado con otros, parezca tener méritos suficientes para ser incluido en la solución y no aparezca en ella.

En la Tabla 2 se resumen todos los algoritmos mencionados en esta sección, además se incluye el trabajo actual el cual hace uso de las preferencias del DM y maneja una cantidad de valores objetivos de hasta nueve objetivos, la limitante de nueve objetivos se debe a falta de experimentación con instancias de mayor tamaño.

Trabajo	Tipo algoritmo	Considera preferencias de DM	Proyectos	Objetivos
Doerner et al. [2004]	Colonia de Hormigas		20-30	5-10
Carazo et al. [2010]	Montecarlo		12-18	2
Deb et al., [2001]	Genético		10-70	2-6
Fernández et al. [2010]	Genético	✓	20-100	6-9
García [2010]	Híperheurístico	✓	20-100	6-9
Rivera et al. [2014]	Colonia de Hormigas	✓	20-500	5-15
Bastiani et al. [2015]	Genético	✓	25-100	6-10
Este trabajo	Algoritmo de procesamiento celular	✓	20-500	4-9

Tabla 2. Tabla comparativa de los algoritmos para cartera de proyectos.

Capítulo 4. Propuesta de Solución

En este capítulo se plantean las estrategias utilizadas para la resolución del problema. Estrategias que son integradas por un algoritmo de procesamiento celular incluyendo sus características esenciales y agregando a él las heurísticas de selección, cruza y mutación seleccionadas.

Además se maneja la incrustación de las preferencias en el proceso de búsqueda.

4.1.1 Algoritmo De Procesamiento Celular

Algunas propiedades que distinguen al algoritmo de Procesamiento Celular de un GA Distribuido son: (1) la frecuencia de migración es controlada por la detección de estancamiento (si las células se encuentran estancadas se procede a la comunicación), (2) permite el intercambio de información genética entre PCells con métodos generación de soluciones lineales y (3) las PCells controlan la evolución de cada subpoblación, lo que permite tener múltiples algoritmos o heurísticas contribuyendo en la resolución del problema.

La estructura general de un algoritmo genético se presenta en la Figura 10.

El algoritmo de procesamiento celular cuenta con una estrategia de estancamiento local y global, al estancarse todas las PCell el algoritmo fuerza la interacción de las células con motivo de que escapen de la optimalidad local, si el estancamiento global es irremediable el algoritmo termina. El Algoritmo 3 muestra el pseudocódigo general de un algoritmo de procesamiento celular.

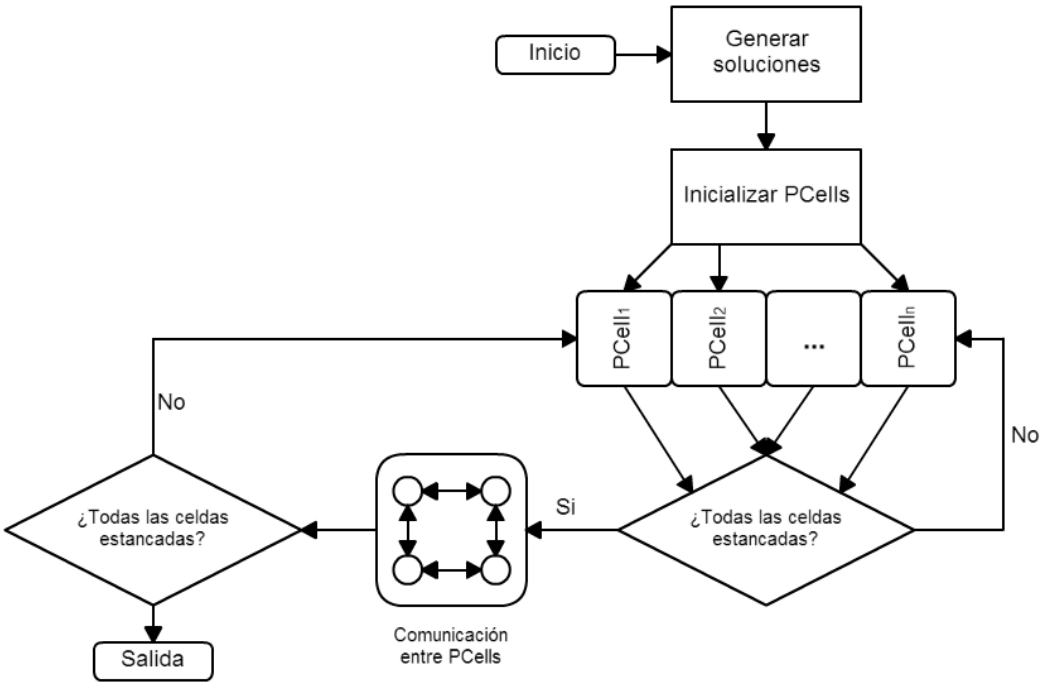

Figura 10. Estructura de un algoritmo de procesamiento celular propuesto por (Terán, Fraire, Carpio, Puga, & Martínez, 2012)

Algoritmo 3. Algoritmo de procesamiento celular (Terán, Fraire, Carpio, Puga, & Martínez, 2012).

```
1   InitializePCells()
2   Repeat
3       Repeat
4           for i = 1 to numberOfPCells do
5               if IsNotStagnated(PCell_i) then
6                   PCell_i()
7               end if
8           end for
9       until AllPCellAreStagnated()
10      PCellInteraction()
11      Re-structureCells()
12  Until StopCondition()
```

El uso de PCells permite la manipulación de diferentes heurísticas en una sola ejecución sin tomar tiempo de procesamiento extra, además muestra los beneficios aportados por los PGAs basados en islas. En el trabajo actual se desarrolló un algoritmo de procesamiento celular para la selección de carteras de proyectos (PSP-CPA).

4.2 Comunicación

La comunicación entre PCells es una técnica de exploración que permite el envío de información entre diferentes subpoblaciones con el fin de salir de un óptimo local. Este proceso es importante en el desarrollo del Algoritmo de Procesamiento Celular y constituye una de sus principales características. Sin la comunicación no sería posible que las diferentes células coexistieran.

La comunicación puede diferir de implementación a implementación debido que su tarea es transmitir individuos o información de una célula a otra sin definir la mecánica del proceso. Incluso diversos aspectos del algoritmo de procesamiento celular no se encuentran especificados dado que es una idea flexible y de gran maleabilidad.

Los principales parámetros correspondientes a la comunicación son los siguientes (Delaossa, Gamez, & Puerta, 2006).

- Número de PCells: el número de células determina el número de subpoblaciones y tamaño de las subpoblaciones.
- Porcentaje de comunicación: determina el número de individuos que cada subpoblación comunica a las demás.
- Topología: determina el número de conexiones de las PCells.
- Política de Remplazo: determina como se remplazan las soluciones de las PCells después de la comunicación, la política más común es sustituir los elementos entrantes por las peores soluciones de la población.

El ajuste de los parámetros mencionados resulta crucial para la paralelización del algoritmo como se ha demostrado en diversos estudios en la literatura (Delaossa, Gamez, & Puerta, 2006) (Cantú-Paz, 1998).

4.2.1 Migración

La migración (Jozefowiez, Semet, & Talbi, 2002) es el concepto más elemental de comunicación entre poblaciones, consiste en el intercambio de individuos entre un par de células dadas. La Figura 11 ilustra el proceso de comunicación.

En el proceso es posible tomar una o varias soluciones de una subpoblación para transferirlas a otra subpoblación conectada topológicamente. La política de remplazo juega un rol importante en todas las comunicaciones estableciendo las bases para la sustitución o agregación de nuevas soluciones, inclusive si las soluciones son borradas de la subpoblación original.

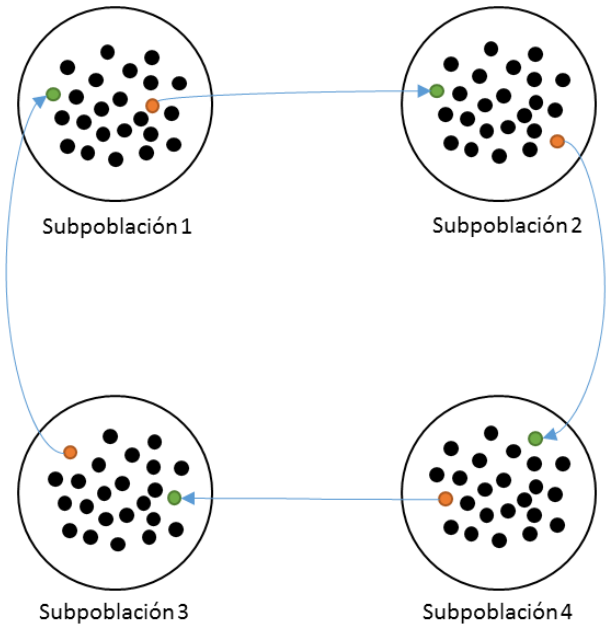

Figura 11. Esquema de migración con cuatro subpoblaciones en una topología de anillo y una solución como porcentaje de comunicación (verde como solución entrante y naranja como solución saliente).

Con respecto al algoritmo de procesamiento celular; la migración se efectúa cuando las células se consideren “estancadas”, es entonces cuando se realiza la comunicación a través de todas las conexiones topológicas. En lugar de la migración se pueden efectuar diversas mecánicas de transferencia, una de ellas es el Path Relinking.

4.2.2 Path Relinking

Originalmente utilizado en Scatter Search (Glover, 1986) , Path Relinking es utilizado como estrategia de comunicación entre subpoblaciones (Delaossa, Gamez, & Puerta, 2006) se basa en la idea de que soluciones de buena calidad pueden producir soluciones de buena calidad. Consiste en explorar las soluciones que se pueden generar transformando la solución origen hasta convertirla en la solución destino, lo cual se logra realizando cambios en la solución origen acortando la distancia entre ambas soluciones, todos los cambios se consideran dentro del camino o “Path”.

La Figura 12 proporciona una idea del método en cuestión. Considérese que las soluciones intermedias pueden salir de la región factible pero queda abierta la posibilidad de eliminarlas o corregirlas a una solución factible.

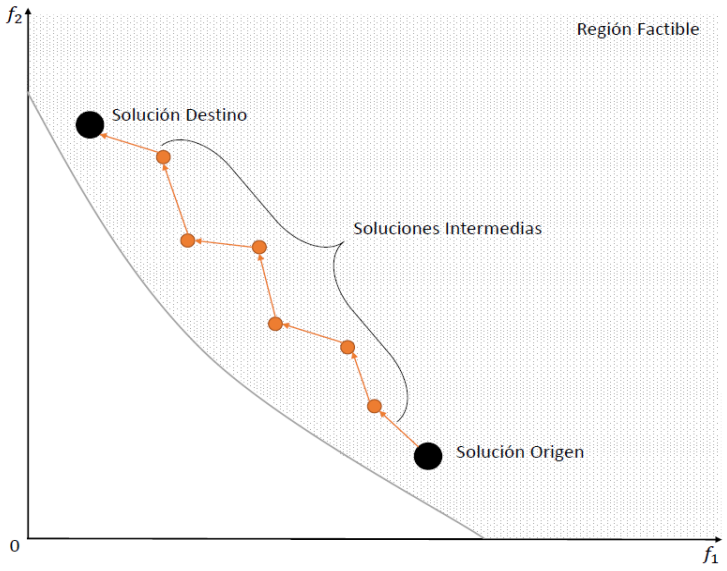

Figura 12. Ejemplo de pathrelinking con las soluciones generadas

4.3 Célula de procesamiento propuesta para resolución del problema

Dado que el algoritmo de procesamiento celular utiliza heurísticas como células de procesamiento se propuso el uso de un algoritmo basado en NOSGAII, el pseudocódigo se muestra en el Algoritmo 4.

Algoritmo 4. Pseudocódigo de PCellNOSGAII

1	run_PCellNOSGAII( ):	
2	F ← NonOutRankingSort(R)	*//Ordenamiento en frentes por xPy*
3	$P \leftarrow \{\}$	
4	$i \leftarrow 1$	
5	$while \ \|P\| < (n / 2)$	*//llenado por Frentes*
6	$P \leftarrow Fi$	
7	$F_n Sort(P)$	*//Ordenamiento por Flujo Neto*
8	$P = P[1:N]$	*//Primeros N elementos*
9	Q=Crossover(P)	
10	$R_{+1} = P \cup Q$	*//Unión Padres e Hijos(Estrategia Elitista)*
11	$return \ R_1$	*//Nueva población*

El algoritmo propuesto, al igual que NOSGAII y NSGAII, es un algoritmo genético. Para que la heurística sea compatible con el algoritmo de procesamiento celular, PCellNOSGA tiene que permitir la ejecución de generaciones fuera de un ciclo principal con el fin de que el programa principal decida detener o continuar su ejecución a conveniencia.

Se puede apreciar que el algoritmo no muestra ciclos de generaciones; esto se debe a que el CPA se encarga de manejarlas, es decir CPA mantiene el flujo del programa y determina cuando ejecutar las diferentes PCells y cuando considerarlas estancadas.

4.4 Representación de la solución

La forma de representación de la solución consiste en un vector de valores binarios $\vec{x}$ los cuales simbolizan el estado de la cartera; un valor verdadero en un índice i lo indica la selección del proyecto i dentro de la cartera. En esta representación cada valor del arreglo es llamado alelo y el arreglo completo es llamado cromosoma, esto basado en la literatura relacionada con algoritmos genéticos (Goldberg, 1989). La Figura 13 muestra la representación de la solución.

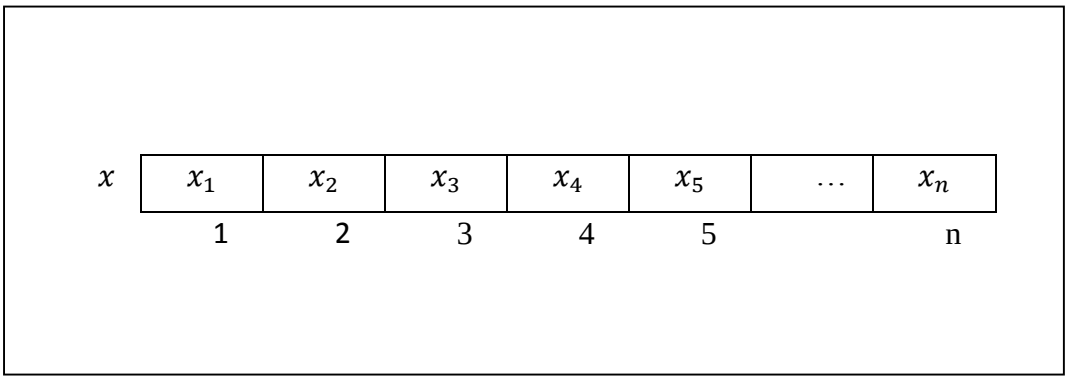

Figura 13. Ilustración de un cromosoma que representa una solución

4.5 Recombinación de soluciones

La recombinación de soluciones es una actividad que es indispensable para algoritmos genéticos, representa la reproducción de especies en la naturaleza, se aplica este concepto en la optimización con el fin de obtener mejores conjuntos de soluciones o generaciones de la población.

Se espera que recombinar soluciones de "buena" calidad produzca descendientes de mejor calidad o de calidad aproximada a la de sus padres. El proceso permite que características positivas se hereden pero también permite la herencia de las características negativas, entonces, seleccionar la estrategia de cruza o recombinación adecuada resulta una tarea importante.

4.5.1 Cruza en un punto

Esta técnica define un punto de corte en los cromosomas padre, para así, generar hijos que son copias fieles de uno de ellos hasta el punto de corte e idénticos al otro padre en la parte restante de su cromosoma (Holland, 1975).

Un punto favorable para esta técnica es que es de fácil aplicación, pero si el punto de corte se encuentra muy cerca de los extremos del cromosoma la cruza no tiene relevancia ya que la solución hijo tendrá un alto grado de similitud a algún padre. La Figura 14 muestra un ejemplo de este método.

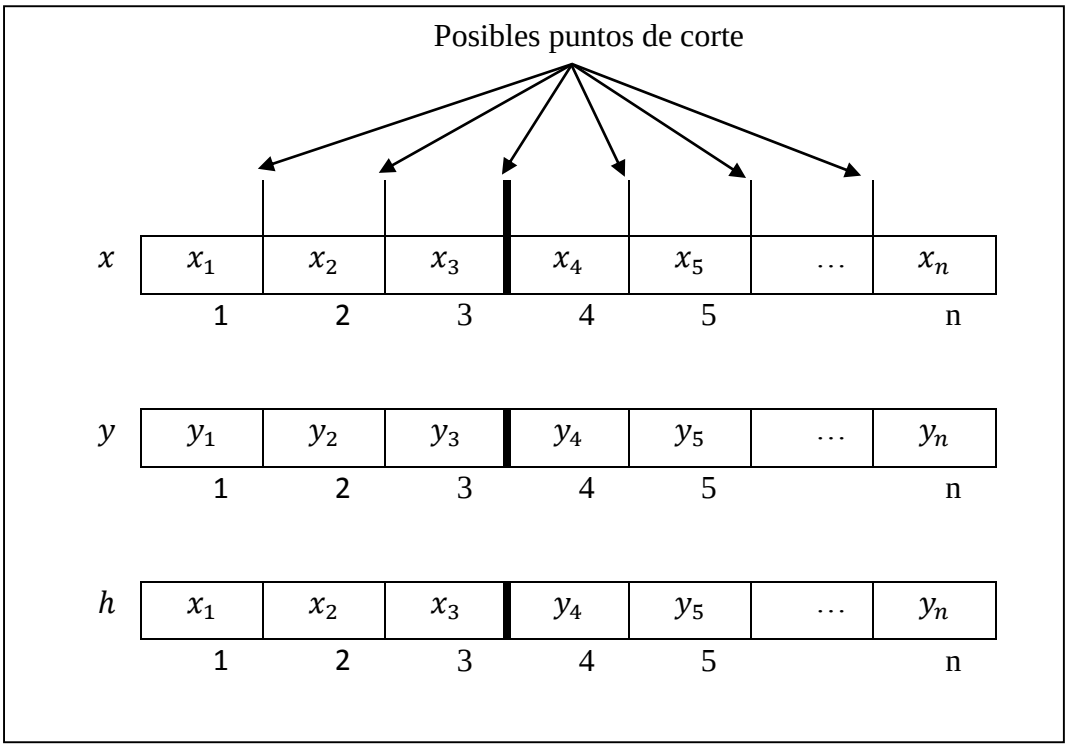

Figura 14. Ejemplo de cruza de un punto, se seleccionó el punto tres para el corte, h es el producto de la recombinación de *x* y *y*.

4.5.2 Cruza Uniforme

Originalmente propuesto por (De Jong & Spears, 1991) la cruza uniforme busca aproximar la cantidad de información genética de cada padre a un 50% dentro de la nueva

solución. Por cada alelo se tiene un 50% de probabilidad de copiarlo del primer padre y 50% de copiarlo del segundo padre. Al completar el cromosoma aproximadamente la mitad de los alelos pertenecerán al primer padre y la otra mitad al segundo. La Figura 15 muestra un ejemplo de la cruza uniforme.

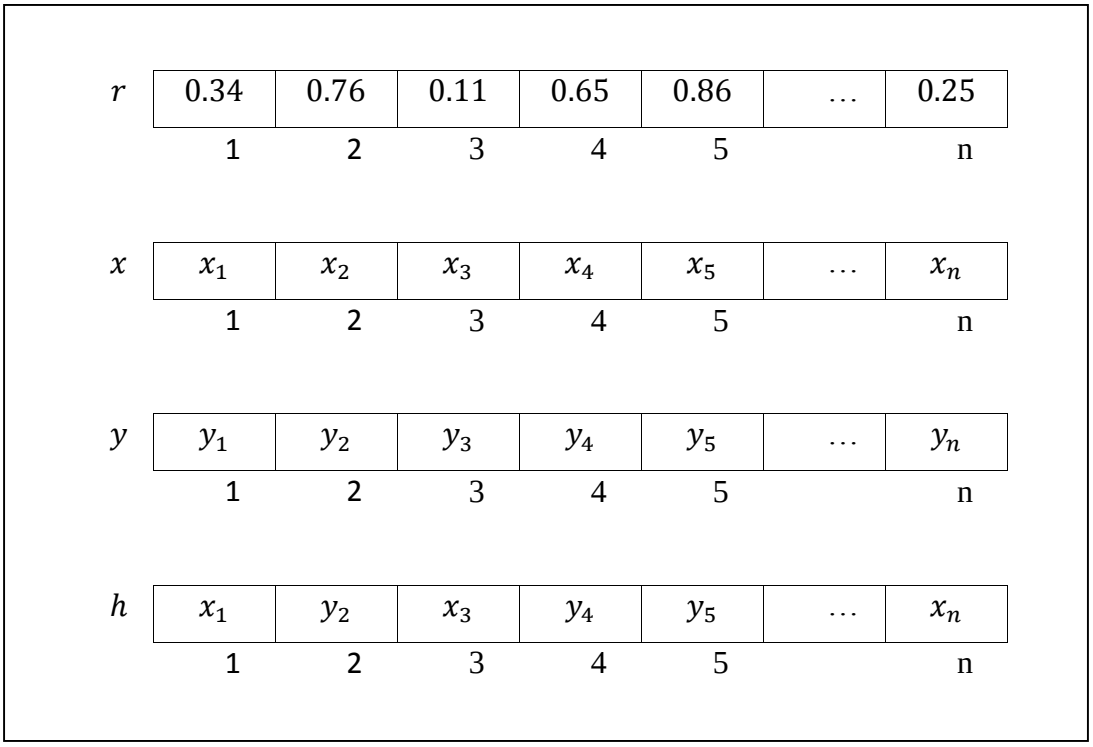

Figura 15. Ejemplo de cruza uniforme, el vector r representa números aleatorios que determinan si se copia el alelo de x o de y (x si r<0.5).

4.6 Selección de Padres

La selección de padres es la herramienta principal para guiar el crecimiento de la población, mediante este proceso se selecciona pares de padres de la población, las recombinaciones de estos padres generarán una siguiente población idealmente de mayor calidad a la actual.

4.6.1 Selección mediante Torneo binario

La selección a través de esta técnica (Blickle & Thiele, 1995) se implementa seleccionando aleatoriamente un par de soluciones de la población obteniendo el mejor par; esto se hace con un comparador de dominancia o en el caso de utilizar FastDominantedSort (Anexo A) se puede usar el ranking producido como criterio.

Se escoge la mejor solución con el fin de ejercer presión selectiva en la búsqueda aunque también existe una probabilidad de escoger la peor de las soluciones seleccionadas con el fin de preservar diversidad genética en la población. El proceso produce la selección de un padre; es necesario repetir el proceso hasta obtener el número suficiente de padres para efectuar la recombinación de soluciones.

4.7 Reparación de soluciones

Al ser un problema con un gran número de restricciones, hacer cambios en las soluciones conlleva una alta probabilidad de obtener soluciones infactibles, por lo que contar con un método eficaz para la reparación de las soluciones es importante para el desarrollo de este trabajo.

4.7.1 Reparación aleatoria

Este método se basa en el cambio de bits aleatorios hasta que se logre llevar la solución a una zona de factibilidad. El beneficio de este método es que en el mejor de los casos es un método de ejecución rápida, pero no hay certeza de cuanto procesamiento utilizará ya que teóricamente puede tomar un tiempo infinito (García, 2010).

Se puede utilizar un número máximo de intentos de reparación pero implica que la solución no se repare y por lo tanto se quede en zona de infactibilidad.

4.7.2 Reparación por restricciones

El método propuesto es un método iterativo que consiste en identificar las restricciones que más alejan a la solución de la factibilidad y guían hacia la factibilidad quitando o incluyendo proyectos en las zonas de necesidad. Recordando que en este problema las restricciones incluyen áreas y regiones, y pueden cometer infracción desde un límite inferior y superior, la selección de la restricción a ajustar es dinámica.

El proceso de ajuste de la solución inicia al evaluar la magnitud de la violación de las restricciones (sea negativa o positiva), entonces se selecciona alterar (agregar o quitar) un proyecto de la región seleccionada; la decisión de que proyecto se modifica es aleatoria solamente siendo restringido por la necesidad de retirar o incrementar el apoyo, es decir si le faltan recursos a una región para ser factible se agregará un proyecto de esa región, en caso contrario se quitará un proyecto. Este proceso de selección es ilustrado por la Figura 16.

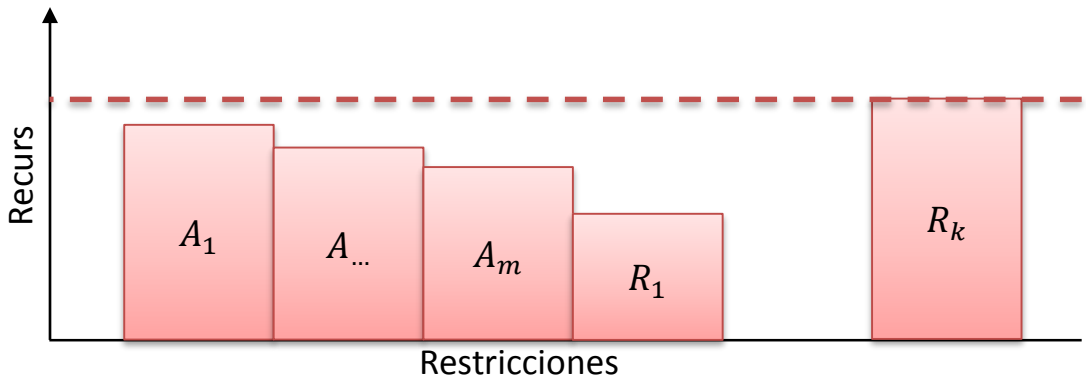

Figura 16. Magnitudes de la violación total en diferentes regiones, si una región no se muestra implica que esa restricción no aporta a la infactibilidad,

Entra las ventajas de este método es que al limitar la aleatoriedad a solo los proyectos que benefician a la reparación de la solución se acelera el proceso, además se utiliza de forma inteligente la aleatoriedad.

4.8 Operador de mutación.

El operador de mutación ayuda en la exploración de la región de búsqueda al distorsionar las soluciones levemente para colocarse en otro punto del espacio de búsqueda.

4.8.1 Mutación por cambio de bit

La mutación por cambio de bit es una operación sencilla donde se toma un bit de la solución y se cambia por la negación del bit. Existe una probabilidad de que el evento de mutación ocurra, pero si sucede, generalmente ocurre inmediatamente después de la recombinación de soluciones. Si se deseara incrementar el efecto de la mutación se puede incrementar el número de bits alterados y la probabilidad de mutación. Además se puede añadir métodos de control de estos parámetros para que cambien a lo largo de la experimentación (Holland, 1975).

4.9 Métodos de Estancamiento.

Se ha demostrado a través de la literatura **Fuente especificada no válida.** que los GAs y las metaheurísticas en general son eficaces al tratar con problemas multi-objetivo, pero examinar la convergencia del frente generado es difícil sin técnicas adecuadas. Normalmente un GA toma como condición de parada un número de generaciones o un valor esperado de algún indicador de calidad.

En casos donde el número de generaciones o el valor esperado no sean correctamente seleccionados el GA puede tomar tiempo de cómputo innecesario o parar cuando todavía existe una mejora prometedora.

Con el fin de encontrar herramientas que permitan determinar efectivamente el punto de parada de un algoritmo se han hecho algunas propuestas en la literatura.

4.9.1 Rudenko and Schoenauer: Medida de Estabilidad.

En el trabajo de (Roudenko-Schoenauer, 2004) definieron una medida de estabilidad basada en NSGA-II (Deb, 2001), ellos demostraron que la estabilización del *Crowding Distance*(CD) máximo del Frente de Pareto es una métrica de convergencia efectiva. Para ello se almacenan los valores máximos de CD de cada generación, el cálculo de la métrica involucra la desviación estándar de los CD máximos y se compara con un valor de tolerancia predefinido.

Esta propuesta se utilizó en fases tempranas de este trabajo, al reproducir la experimentación de este trabajo se encontró una métrica efectiva y estable cuando se trabajaron dos objetivos, se encontró que al incrementar el número de objetivos esta métrica perdía consistencia. Se optó por cambiar el uso de este método por uno donde se utilicen las preferencias del decisor.

4.9.2 Detección por número de cambios favorables.

Se propuso el cambio de método de estancamiento por uno donde se tome en cuenta al DM, en este método se guarda las soluciones NS de la generación, para considerar una subpoblación “estancada” se compara el frente NS de la generación anterior con la actual, si el número de soluciones nuevas no supera un margen preestablecido se puede considerar estancada. Además, se puede hacer un historial del número de soluciones nuevas de las últimas generaciones y utilizar la desviación estándar para considerar si existe o no estancamiento; esto con el fin de tener una medida más estable.

Capítulo 5. Experimentación y Resultados

En este capítulo se presenta las características de hardware y software utilizadas, inmediatamente se describen los experimentos realizados; en el experimento 1 se usó el software JMetal (Anexo 2) como estándar para analizar la factibilidad del algoritmo propuesto. El experimento 2 muestra la experimentación comparativa entre NO-ACO y la propuesta actual, mientras que el experimento 3 muestra la comparación de MSS de Martínez (2015) contra PSP-CPA.

5.1 Características de: hardware, software, algoritmos e instancias

El equipo utilizado para la experimentación es una computadora personal modelo SVT11115 con un procesador Intel Core i5 1.7 GHz de cuatro núcleos con 4GB de memoria RAM.

El software utilizado es un sistema operativo Windows 7 con Java 1.8 como lenguaje de programación, se utilizó el IDE Neatbeans 8.01.

5.2 Experimento 1: Jmetal

El propósito del experimento es comprobar la factibilidad del algoritmo, para determinar que el funcionamiento es correcto independiente del problema, se decidió utilizar el problema de la mochila multi-objetivo (Gandibleux & Freville, 2000) debido a su similitud

con el problema de selección de cartera de proyectos (García, 2010), además existen instancias con óptimos conocidos, las instancias a utilizar son las instancias de la mochila multi-objetivo propuestas por (Gandibleux & Freville, 2000)[1].

La configuración del experimento es la siguiente:

- Tamaño de la población: 500
- Número de generaciones: 100
- Número de ejecuciones del algoritmo por instancia: 30
- Instancias de prueba: set 1A y 1B-A de MOKP

En la Tabla 3 se resumen las configuraciones de los algoritmos utilizados en la experimentación.

	NSGAII jMetal	**Config1**	**Config2**	**Config3**	**Config4**	**Config5**
Flujo de Prog	N/A	N/A	CPA	CPA	N/A	CPA
Heurística	NSGAII				Grano F.	
Selección	Torneo Binario					
Cruza	Un punto	Uniforme				
Mutación	Flip de bits					
N° PCells	N/A	1	2	4	1	4
Detección estancamiento	x	x	Crowding Distance		N° Cambios Fav.	
Comunicación	x	x	Migración		x	Path R.

Tabla 3. Configuraciones de los algoritmos

La experimentación se realizó en dos etapas, en la primera etapa se compararon diversas implementaciones:

- La implementación de NSGAII de JMetal.

[1] Las instancias utilizadas están disponibles en http://xgandibleux.free.fr/MOCOlib/MOKP.html

- Una célula de procesamiento de NSGAII.
- Dos células de procesamiento de NSGAII con migración.
- Cuatro células de procesamiento de NSGAII con migración.

Se utilizaron dos métricas para medir la calidad de los algoritmos en función de la diversidad (GS, Spread) (Zhou, Jin, Zhang, Sendhoff, & Tsang, 2006) y distancia al frente real (GD, Generational Distance) (Van Veldhuizen & Lamont, 1998). La aproximación a cero de los valores en el cálculo de las métricas representa el óptimo.

A	**NSGAII (jMetal)**		**Config1**		**Config2**		**Config3**	
Instancia	GS	GD	GS	GD	GS	GD	GS	GD
2KP50-50	0.77	0.36	0.70	0.01	0.70	0.01	0.66	0.01
2KP100-50	0.84	0.63	0.79	0.01	0.73	0.01	0.69	0.01
2KP50-1A	0.87	0.50	0.21	0.00	0.42	0.02	0.44	0.00
2KP100-1A	0.87	0.41	0.81	0.01	0.75	0.01	0.74	0.01
2KP150-1A	0.93	0.55	0.83	0.02	0.81	0.02	0.77	0.01
2KP200-1A	0.90	0.68	0.86	0.03	0.81	0.02	0.76	0.02
2KP250-1A	0.93	0.43	0.86	0.05	0.82	0.02	0.82	0.02
2KP300-1A	0.91	0.85	0.87	0.06	0.87	0.02	0.79	0.02
2KP350-1A	0.93	0.69	0.88	0.08	0.87	0.03	0.81	0.03
2KP400-1A	0.94	0.62	0.88	0.11	0.85	0.03	0.81	0.03
2KP450-1A	0.94	0.68	0.90	0.12	0.83	0.04	0.80	0.04
2KP500-1A	0.96	0.58	0.84	0.12	0.86	0.04	0.83	0.04
Promedios	0.90	0.58	0.79	0.05	0.78	0.02	0.74	0.02

Tabla 4. Resultados: NSGAII jMetal), Config1, Config2 y Config3.

Como se observa en la **¡Error! No se encuentra el origen de la referencia.**Tabla 4 los resultados obtenidos por los algoritmos muestran que la configuración 1 supera en ambas métricas al algoritmo de referencia NSGAII de JMetal.

Después de hacer una comparación del desempeño de PSP-CPA contra el NSGAII de JMetal con una población única, se propuso distribuir la población en un mayor número de células con el objetivo de validar la aportación de la comunicación entre ellas; así como tratar de configurar el número de células a usar.

En la Tabla 5 se observan los resultados del incremento del número de células concluyendo que en esta experimentación los mejores resultados se obtuvieron al

incrementar el número de células, obteniendo una mejoría del 3% al comparar los resultados de las configuraciones 1 y 3.

B	Config4		Config5	
Instancia	GS	GD	GS	GD
2KP50-50	0.74	0.10	0.77	0.12
2KP100-50	0.85	0.06	0.85	0.05
2KP50-1A	0.66	0.01	0.78	0.02
2KP100-1A	0.86	0.08	0.88	0.04
2KP150-1A	0.88	0.09	0.90	0.07
2KP200-1A	0.90	0.10	0.89	0.08
2KP250-1A	0.90	0.13	0.91	0.12
2KP300-1A	0.92	0.14	0.92	0.17
2KP350-1A	0.90	0.16	0.92	0.22
2KP400-1A	0.92	0.25	0.92	0.27
2KP450-1A	0.93	0.19	0.92	0.29
2KP500-1A	0.94	0.17	0.93	0.30
Promedios	0.94	0.17	0.88	0.15

Tabla 5. Experimentación con Genético celular y PathRelinking

La segunda etapa del experimento se realizó al comparar las siguientes implementaciones:

- La implementación de NSGAII de JMetal.
- Una célula de procesamiento de un genético de *Grano Fino*.
- Cuatro células de procesamiento de un genético de *Grano Fino*.

La Tabla 5 describe los resultados de la experimentación realizada con PCells. En la columna *Config4* corresponde a un algoritmo genético de malla celular, mientras que la columna *Config5* corresponde a un algoritmo genético celular con población distribuida en cuatro *PCells*. Los resultados preliminares indican que el algoritmo celular (*Config4*) muestra un desempeño inferior a la implementación de *Config1*, sin embargo, al subdividir la población los resultados mejoran en un 4% entre *Config4* y *Config5*, validando la efectividad de la comunicación entre células.

5.3 EXPERIMENTO 2: NO-ACO CONTRA CELULAR

Se propone el desarrollo de un algoritmo celular para la resolución del problema de cartera de proyectos, el cual busca mejorar las estrategias actuales de solución con un enfoque de pseudo-paralelización el cual ayuda al escape de óptimos locales y permite una eficiente búsqueda del frente de Pareto.

La configuración del PSP-CPA es la siguiente:

- Tamaño de la población: 200
- Numero de generaciones: 500
- Número de ejecuciones del algoritmo por instancia: 5

La configuración del NO-ACO es la siguiente:

- Numero de iteraciones : 500
- Numero de hormigas: 200

La Tabla 6 proporciona información acerca del algoritmo PSP-CPA, se infiere que es el caso más básico de algoritmo de procesamiento celular al solo manejar una célula lo cual inhabilita la comunicación entre células.

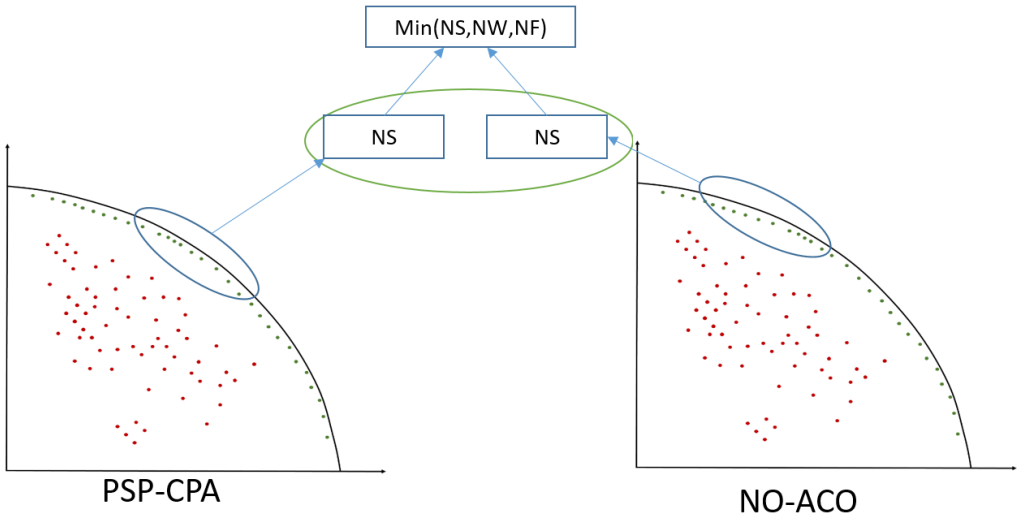

Figura 17. Los experimentos se basan en la obtención de los frentes NS de ambos algoritmos haciendo después el análisis lexicográfico de MIN(NS,NW,NF).

Número de PCells	**1**
Soluciones por PCells	200
Comunicación	No existente

Tabla 6. Características de PSP-CPA

La Tabla 7 muestra la experimentación realizada donde se compara el algoritmo PSP-CPA y el algoritmo NO-ACO, esta consiste en la unión de las fronteras no superadas generadas por dichos algoritmos y contabilizar el número de soluciones que aportan al nuevo NS. Las columnas de la Tabla 7 se pueden describir de izquierda a derecha como el nombre de la instancia, el nombre del algoritmo utilizado, el número de soluciones que aporta (sus soluciones NS), el número de soluciones que pertenecen a Min(NS,NW,NF) y por último el tiempo promedio por ejecución del algoritmo (en segundos), además al término de la construcción de la tabla se generan los totales.

El proceso para realizar la experimentación consiste en unir los frentes NS que generan los algoritmos a evaluar agregando un identificador de procedencia a las soluciones tratadas, inmediatamente se filtran las soluciones generando nuevos frentes NS, NW y NF. Como resultado importante se tiene que el algoritmo que aporta mayor número de soluciones en el nuevo frente NS, NW y NF. La Figura 17 ilustra este proceso.

Instancia	Algoritmo	$\lvert NS(O_1 \cup O_2)\rvert$	|Min(NS,NW,NF)|	Tiempo promedio (S)
o9p100_00	PSP-CPA	7	1.6	45.91
	NO-ACO	1.4	1	31.10
o9p100_01	PSP-CPA	12	2.8	44.85
	NO-ACO	1.2	0.6	32.61
o9p100_02	PSP-CPA	13.6	0.8	36.54
	NO-ACO	1.4	0.8	40.63
o9p100_03	PSP-CPA	14.8	1.4	28.29
	NO-ACO	2.4	1.8	37.82
o9p100_04	PSP-CPA	6.4	1.4	48.20
	NO-ACO	1.6	1.2	38.41
o9p100_05	PSP-CPA	6.6	1.6	52.35
	NO-ACO	2	1.4	41.61
o9p100_06	PSP-CPA	9.2	1.6	43.61
	NO-ACO	1.4	0.8	47.18
o9p100_07	PSP-CPA	11.4	1.2	44.77
	NO-ACO	1	0.8	38.26
o9p100_08	PSP-CPA	10.2	1.4	44.13
	NO-ACO	2.4	1.2	38.66

o9p100_09	PSP-CPA	9.8	1.2	44.17
	NO-ACO	1.2	1	32.77
o9p100_10	PSP-CPA	18	1.4	31.15
	NO-ACO	1.2	1	32.29
o9p100_11	PSP-CPA	40.6	0.2	14.34
	NO-ACO	2	0.6	34.62
o9p100_12	PSP-CPA	11.8	0	28.87
	NO-ACO	1.6	1	39.75
o9p100_13	PSP-CPA	7.6	0.8	46.55
	NO-ACO	1.4	1	34.87
o9p100_14	PSP-CPA	15.4	1.4	35.32
	NO-ACO	1.6	1.2	38.53
o9p100_15	PSP-CPA	11	1.8	37.01
	NO-ACO	1.6	0.8	33.86
o9p100_16	PSP-CPA	14	0.2	29.49
	NO-ACO	1.4	0.8	35.23
o9p100_17	PSP-CPA	24.8	0.2	29.49
	NO-ACO	1.4	1	37.00
o9p100_18	PSP-CPA	9.8	1	48.24
	NO-ACO	1	0.4	26.60
o9p100_19	PSP-CPA	21	0.6	24.61
	NO-ACO	1.6	0.6	42.81
o9p100_20	PSP-CPA	11.8	0.6	40.81
	NO-ACO	1.4	1.2	34.96
o9p100_21	PSP-CPA	12.4	1.2	34.51
	NO-ACO	1.4	1	37.00
o9p100_22	PSP-CPA	15	1.2	32.60
	NO-ACO	1.4	1	29.86
o9p100_23	PSP-CPA	23.4	0.8	25.67
	NO-ACO	1.6	0	55.19
o9p100_24	PSP-CPA	9.4	1	27.36
	NO-ACO	1.6	1	27.36
o9p100_25	PSP-CPA	16.2	0.6	26.48
	NO-ACO	1.6	0.6	43.69
o9p100_26	PSP-CPA	6.6	1.4	35.19
	NO-ACO	1.4	0.6	38.66
o9p100_27	PSP-CPA	14.4	1.6	37.70
	NO-ACO	1.2	0.6	39.40
o9p100_28	PSP-CPA	5.2	1.8	43.77
	NO-ACO	1	0.8	35.59
o9p100_29	PSP-CPA	5.6	1	44.45
	NO-ACO	2.4	1.2	36.20
Totales	PSP-CPA	13.16	1.14	37.50
	NO-ACO	1.52	0.90	37.01

Tabla 7. Resultados experimentales (Promedios de cinco ejecuciones), O_1 y O_2 son las soluciones generadas por PSP-CPA y NO-ACO respectivamente.

La Tabla 7 carece por si sola de argumentos suficientes para marcar una superioridad entre algoritmos aun cuando NO-ACO ofrece un menor número de soluciones promedio que minimizan lexicográficamente el vector (NS, NW, NF).

Se decidió aplicar la prueba de los rangos con signo de Wilcoxon (Wilcoxon, Katti, & Wilcox, 1963) con el fin de determinar si las muestras tienen diferencias significativas entre sí. Se utilizó el software estadístico R (R Development Core Team, 1997) para calcular la prueba, los resultados muestran el rechazo de la hipótesis nula, indicando que existe una diferencia significativa entre las muestras y que el resultado no fue definido por el azar.

Se propuso experimentar con las cualidades del algoritmo de procesamiento celular y se incrementó el número de PCells subdividiendo el número de población para mantener características equitativas. La Tabla 8 muestra la configuración presentada.

Número de PCells	**2**
Soluciones por PCells	100
Comunicación	PathRelinking

Tabla 8. Características de PSP-CPA

En resultados mostrados en la Tabla 9 se observa capacidad de PSP-CPA para disminuir los tiempos de ejecución sin pérdida de calidad gracias a las estrategias implementadas en el algoritmo para el manejo de PCells.

Se decidió continuar subdividiendo la población en un mayor número de PCells con fines experimentales, la Tabla 10 muestra la configuración utilizada y la Tabla 11 ilustra los resultados correspondientes, en esta última se observa que existe una perdida en calidad de soluciones. Al igual que la experimentación anterior la prueba de los rangos con signo de Wilcoxon muestra diferencias significativas entre ambas muestras.

Instancia	Algoritmo	$\lvert NS(O_1 \cup O_2)\rvert$	|Min(NS,NW,NF)|	Tiempo promedio (S)
o9p100_00	PSP-CPA	4.8	2.2	9.78
	NO-ACO	1.4	0.8	29.33
o9p100_01	PSP-CPA	5	1.8	9.16
	NO-ACO	1.4	0.8	30.35
o9p100_02	PSP-CPA	12	1.6	8.32
	NO-ACO	1.4	1	38.69
o9p100_03	PSP-CPA	9	1.2	7.54
	NO-ACO	2.4	1.4	40.39
o9p100_04	PSP-CPA	5.4	0.6	11.07
	NO-ACO	1.6	1.2	36.23
o9p100_05	PSP-CPA	7.2	0.6	11.35
	NO-ACO	2	1	37.42
o9p100_06	PSP-CPA	7.8	0.8	8.94
	NO-ACO	1.6	0.8	42.22
o9p100_07	PSP-CPA	7.6	1	9.79
	NO-ACO	1.2	0.8	36.54
o9p100_08	PSP-CPA	7	1.2	9.86
	NO-ACO	2.4	1.4	39.93
o9p100_09	PSP-CPA	9.2	1.2	10.94
	NO-ACO	1.2	0.6	32.98
o9p100_10	PSP-CPA	14.2	0.4	7.29
	NO-ACO	1.2	0.8	31.75
o9p100_11	PSP-CPA	27.4	0.4	4.82
	NO-ACO	2	0.6	34.43
o9p100_12	PSP-CPA	8.2	1	8.89
	NO-ACO	1.6	1.2	37.56
o9p100_13	PSP-CPA	9.4	1.2	9.27
	NO-ACO	1.4	1.4	32.18
o9p100_14	PSP-CPA	12.4	1	8.01
	NO-ACO	1.4	1.2	37.05
o9p100_15	PSP-CPA	7.8	0.6	9.08
	NO-ACO	1.6	0.2	32.87
o9p100_16	PSP-CPA	9.8	0.8	7.64
	NO-ACO	1.4	0.6	34.59
o9p100_17	PSP-CPA	18	0.8	7.64
	NO-ACO	1.4	1.4	35.61
o9p100_18	PSP-CPA	6.4	2.4	11.27
	NO-ACO	1	1	26.16
o9p100_19	PSP-CPA	23	1.2	5.81
	NO-ACO	1.6	0.8	41.47
o9p100_20	PSP-CPA	11.2	1.4	9.31
	NO-ACO	1.4	1.2	33.88
o9p100_21	PSP-CPA	10.2	2	7.93

	NO-ACO	1.4	1.4	35.61
o9p100_22	PSP-CPA	8.4	2.8	8.41
	NO-ACO	1.4	1	29.92
o9p100_23	PSP-CPA	19.2	1	6.79
	NO-ACO	1.8	0.8	52.35
o9p100_24	PSP-CPA	8.2	1	29.04
	NO-ACO	1.6	1	29.04
o9p100_25	PSP-CPA	10.4	0.4	7.62
	NO-ACO	2	0.8	47.05
o9p100_26	PSP-CPA	10.8	0.6	8.12
	NO-ACO	1.4	1	41.48
o9p100_27	PSP-CPA	9.6	1.6	9.52
	NO-ACO	1.4	1.2	42.09
o9p100_28	PSP-CPA	6.8	1.2	10.47
	NO-ACO	1.2	0.8	38.29
o9p100_29	PSP-CPA	5.4	0.8	10.62
	NO-ACO	2.6	1.6	37.12
Totales	PSP-CPA	10.39	1.19	8.88
	NO-ACO	1.58	0.99	36.40

Tabla 9. Resultados experimentales (Promedios de cinco ejecuciones), O_1 y O_2 son las soluciones generadas por PSP-CPA y NO-ACO respectivamente.

Al observar los resultados se propuso incrementar el número de soluciones de ambos algoritmos con el fin de ver su comportamiento en mayor tiempo de ejecución.

Número de PCells	**4**
Soluciones por PCells	50
Comunicación	PathRelinking

Tabla 10. Características de PSP-CPA

Instancia	Algoritmo	$\lvert NS(O_1 \cup O_2)\rvert$	|Min(NS,NW,NF)|	Tiempo promedio (S)
o9p100_00	PSP-CPA	6.6	1.4	3.38
	NO-ACO	1.4	1.4	30.88
o9p100_01	PSP-CPA	6.4	1.6	2.92
	NO-ACO	1.4	1.2	30.65
o9p100_02	PSP-CPA	9.4	1	2.71
	NO-ACO	1.4	1	38.68

o9p100_03	PSP-CPA	9.4	0.4	2.30
	NO-ACO	2.4	1.6	37.14
o9p100_04	PSP-CPA	3.2	0.8	3.09
	NO-ACO	1.6	1.2	33.13
o9p100_05	PSP-CPA	4.6	1	3.39
	NO-ACO	2	1.2	34.16
o9p100_06	PSP-CPA	6.8	0.4	2.73
	NO-ACO	1.4	1	38.46
o9p100_07	PSP-CPA	6.4	1	2.92
	NO-ACO	1.2	1	33.50
o9p100_08	PSP-CPA	5.4	0.6	2.82
	NO-ACO	2.4	1.2	36.74
o9p100_09	PSP-CPA	9.4	1.2	3.14
	NO-ACO	1.4	1.2	30.32
o9p100_10	PSP-CPA	17.6	1	2.31
	NO-ACO	1.2	1	31.84
o9p100_11	PSP-CPA	23.2	0.6	2.00
	NO-ACO	2	1.2	34.54
o9p100_12	PSP-CPA	9.4	1	2.61
	NO-ACO	1.6	1.4	37.52
o9p100_13	PSP-CPA	11.2	1.6	2.94
	NO-ACO	1.4	1.4	32.19
o9p100_14	PSP-CPA	13.8	2	3.01
	NO-ACO	1.6	1.4	36.81
o9p100_15	PSP-CPA	8.4	0.6	2.80
	NO-ACO	1.6	1.2	32.91
o9p100_16	PSP-CPA	4	2	2.63
	NO-ACO	1.4	1.4	34.46
o9p100_17	PSP-CPA	14.4	2	2.63
	NO-ACO	1.4	1.2	35.68
o9p100_18	PSP-CPA	4.8	2.2	3.55
	NO-ACO	1	1	26.07
o9p100_19	PSP-CPA	12.4	1	2.07
	NO-ACO	1.6	1.2	41.33
o9p100_20	PSP-CPA	12.6	1	2.79
	NO-ACO	1.4	1	34.07
o9p100_21	PSP-CPA	6	0.6	2.56
	NO-ACO	1.4	1.2	35.68
o9p100_22	PSP-CPA	7.6	1	2.78
	NO-ACO	1.4	1.2	30.04
o9p100_23	PSP-CPA	20.6	0.4	2.35
	NO-ACO	1.8	0.8	52.02
o9p100_24	PSP-CPA	7.4	1.6	26.42
	NO-ACO	1.6	1.6	26.42
o9p100_25	PSP-CPA	9.8	1	2.27
	NO-ACO	2	1.2	42.93

o9p100_26	PSP-CPA	11.6	1.2	2.46
	NO-ACO	1.4	1.4	38.09
o9p100_27	PSP-CPA	12.4	2.2	3.19
	NO-ACO	1.4	1.4	38.65
o9p100_28	PSP-CPA	3.8	1	3.07
	NO-ACO	1.2	1	35.24
o9p100_29	PSP-CPA	6.4	0	3.06
	NO-ACO	2.6	1.2	34.14
Totales	PSP-CPA	9.5	1.08666667	2.78
	NO-ACO	1.58666667	1.22	35.05

Tabla 11. Resultados experimentales (Promedios de cinco ejecuciones), O_1 y O_2 son las soluciones generadas por PSP-CPA y NO-ACO respectivamente.

El algoritmo PSP-CPA conserva un estado de competitividad frente a NO-ACO al concluir el experimento especificado en la Tabla 12 con respecto a los resultados mostrados en la Tabla 13, este efecto se produjo aun cuando se duplicaron las características experimentales iniciales (Figura 10). Al igual que las experimentaciones anteriores la prueba de los rangos con signo de Wilcoxon muestra diferencias significativas entre ambas muestras.

En la Tabla 13 se observa que en quince de las treinta instancias de prueba el frente Min(NS,NW,NF) lo conforman soluciones pertenecientes de PSP-CPA mientras que solamente seis instancias existen soluciones exclusivamente de NO-ACO.

Un análisis a nivel de solución nos permite observar que PSP-CPA muestra una tendencia a apoyar los objetivos de mayor peso comparado contra NO-ACO. Una muestra aleatoria se muestra en la Tabla 14; donde la columna 1 contiene el nombre de los algoritmos.

PSP-CPA		**NO-ACO**	
Número de PCells	2	Hormigas	400
Soluciones por PCells	200	Iteraciones	400
Comunicación	PathRelinking		
Generaciones Máximas	400		

Tabla 12. Características de PSP-CPA y NO-ACO

Instancia	Algoritmo	$\lvert NS(O_1 \cup O_2)\rvert$	|Min(NS,NW,NF)|	Tiempo promedio (S)
o9p100_00	PSP-CPA	12	1	103.09
	NO-ACO	2	0	101.82
o9p100_01	PSP-CPA	12	3	106.00
	NO-ACO	1	1	101.46
o9p100_02	PSP-CPA	19	2	113.38
	NO-ACO	1	0	121.22
o9p100_03	PSP-CPA	29	1	122.86
	NO-ACO	1	0	113.82
o9p100_04	PSP-CPA	5	3	101.66
	NO-ACO	2	1	107.99
o9p100_05	PSP-CPA	28	1	154.55
	NO-ACO	1	0	110.32
o9p100_06	PSP-CPA	20	3	137.88
	NO-ACO	3	0	119.24
o9p100_07	PSP-CPA	18	1	170.20
	NO-ACO	1	1	100.91
o9p100_08	PSP-CPA	15	1	130.13
	NO-ACO	2	0	109.67
o9p100_09	PSP-CPA	10	1	95.61
	NO-ACO	1	0	100.55
o9p100_10	PSP-CPA	36	0	111.74
	NO-ACO	1	1	102.32
o9p100_11	PSP-CPA	85	0	56.15
	NO-ACO	1	1	108.01
o9p100_12	PSP-CPA	13	2	90.70
	NO-ACO	1	0	111.98
o9p100_13	PSP-CPA	19	2	94.18
	NO-ACO	1	0	104.11
o9p100_14	PSP-CPA	32	0	123.68
	NO-ACO	1	1	111.75
o9p100_15	PSP-CPA	6	1	107.28
	NO-ACO	2	0	64.72
o9p100_16	PSP-CPA	7	1	117.61
	NO-ACO	3	2	107.00
o9p100_17	PSP-CPA	26	1	117.61
	NO-ACO	2	1	112.86
o9p100_18	PSP-CPA	12	1	194.32
	NO-ACO	2	0	88.25
o9p100_19	PSP-CPA	28	1	102.20
	NO-ACO	1	1	120.83
o9p100_20	PSP-CPA	39	2	128.93
	NO-ACO	1	0	106.41
o9p100_21	PSP-CPA	15	0	81.17

	NO-ACO	2	1	112.86
o9p100_22	PSP-CPA	13	1	138.56
	NO-ACO	1	1	98.59
o9p100_23	PSP-CPA	38	0	93.98
	NO-ACO	1	1	138.26
o9p100_24	PSP-CPA	3	1	89.22
	NO-ACO	3	1	89.22
o9p100_25	PSP-CPA	21	1	103.62
	NO-ACO	1	0	128.46
o9p100_26	PSP-CPA	25	1	139.05
	NO-ACO	1	0	121.80
o9p100_27	PSP-CPA	15	5	137.18
	NO-ACO	1	1	119.88
o9p100_28	PSP-CPA	7	1	102.28
	NO-ACO	1	1	111.18
o9p100_29	PSP-CPA	8	2	134.99
	NO-ACO	0	0	111.09
Totales	PSP-CPA	20.5333333	1.3	118.48
	NO-ACO	1.4	0.53333333	108.47

Tabla 13. Resultados experimentales (Promedios de cinco ejecuciones), O_1 y O_2 son las soluciones generadas por PSP-CPA y NO-ACO respectivamente.

o9p100_00					
N Obj	1	2	3	4	5
Pesos	10	13	10	12	7
PSP-CPA	1277520	1051245	1581440	1007195	1684115
NO-ACO	1335340	1042010	1606335	989950	1721230
N Obj	6	7	8	9	
Pesos	13	10	7	18	
PSP-CPA	1327705	2107370	1498380	1767615	
NO-ACO	1338425	2084490	1496610	1743455	

Tabla 14. Muestra de soluciones tomadas aleatoriamente de la experimentación de la Tabla 12.

El incremento progresivo del número de células entre Tabla 7 , Tabla 9 y Tabla 11 muestra disminución del tiempo de ejecución, sin embargo, el abuso de la subdivisión de la población trae consigo repercusiones negativas a la calidad de la población a partir de cierto punto. Un resumen puede encontrase en la Tabla 15 donde se muestra los resultados de las experimentaciones.

Algoritmo	$\|NS(O_1 \cup O_2)\|$	|Min(NS,NW,NF)|	Tiempo promedio (S)
PSP-CPA 1PCell	13.16	1.14	37.50
NO-ACO	1.52	0.90	37.01
PSP-CPA 2PCell	10.39	1.19	8.88
NO-ACO	1.58	0.99	36.40
PSP-CPA 4PCell	9.5	1.08666667	2.78
NO-ACO	1.58666667	1.22	35.05

Tabla 15. Resumen de resultados de la experimentación (cada par de renglones representa un experimento independiente).

Un factor de importancia es la capacidad de PSP-CPA de manejar un mayor número de soluciones en un mismo plazo de tiempo comparado contra NO-ACO, lo que permite realizar experimentaciones restringidas por tiempo y no por número de soluciones.

Se experimentó además duplicando los recursos disponibles para ambos algoritmos mostrando los totales en la Tabla 16, se demostró que el algoritmo PSP-CPA presenta un mayor número de soluciones consideradas mejor compromiso.

Algoritmo	$\|NS(O_1 \cup O_2)\|$	|Min(NS,NW,NF)|	Tiempo promedio (S)
PSP-CPA	20.5333333	1.3	118.48
NO-ACO	1.4	0.53333333	108.47

Tabla 16. Totales de experimentación.

5.4 Experimento 3: MSS contra PSP-CPA

Además de experimentar contra NO-ACO se procedió a experimentar contra MSS (Martínez, 2015), el procedimiento para realizar la experimentación difiere al Experimento 2 debido a la naturaleza de los resultados disponibles, se usó la instancia 09p100_00 solamente, las características de los algoritmos se observan en la Tabla 17 incluye las características experimentales de ambos algoritmos y en la Tabla 18 se muestran los resultados obtenidos.

Faltan argumentos suficientes para determinar un algoritmo como vencedor debido a la falta de casos de prueba presentados, pero aun así, es importante notar que producen resultados de manera similar.

PSP-CPA		MSS	
Número de PCells	4	Número de evaluaciones	700000
Soluciones por PCells	200	Tamaño de P,RS1,RS2,ext	100,30,50,50
Comunicación	PathRelinking	Cruza	Uniforme
Generaciones Máximas	200		

Tabla 17. Configuración de la experimentación.

Instancia	Algoritmo	$\lvert NS(O_1 \cup O_2)\rvert$	|Min(NS,NW,NF)|
o9p100_00	PSP-CPA	84.5	2.5
	MSS	74	2.4

Tabla 18. Resultados experimentales de MSS contra PSP-CPA

Capítulo 6. Conclusiones y recomendaciones

6.1 Conclusiones

Este documento contiene la propuesta de utilizar un método poblacional para dar solución a la cartera de proyectos, en específico un algoritmo de procesamiento celular. La principal ventaja de este método es que permite la exploración de diferentes regiones cuando la búsqueda se estanca en un estado de no mejora.

Al analizar los métodos del estado del arte se determinó que utilizar alguna variación del NSGA-II sería lo más adecuado para ser parte de nuestro algoritmo de procesamiento celular dada su simplicidad y eficiencia.

Las pruebas experimentales señalan que el uso de métodos de comunicación y detección de estancamiento impactan en gran medida los tiempos de ejecución del algoritmo por lo tanto ahorran considerables cantidades de procesamiento sin perder calidad de soluciones, por el contrario, mejorándolas. Además se determinó que la subdivisión de la población ayuda en evitar el estancamiento del algoritmo.

La comparación del algoritmo propuesto con NO-ACO demostró que el trabajo actual es un software competitivo con respecto al problema planteado.

6.2 Trabajo Futuro

- Desarrollar una métrica de desempeño independiente de las preferencias propuestas por (Fernández, López, & Coello, 2010) (Habilitaría posible experimentación en una zona neutral entre algoritmos).
- Considerar el uso de células de procesamiento con y sin preferencias trabajando simultáneamente (Con el fin de no perder la perspectiva del problema original en la búsqueda).
- Considerar el uso de células con diferentes niveles de preferencia (permitiría uso de células orientadas a la exploración y células orientadas a la explotación).
- Incluir preferencias adaptativas de forma que el algoritmo busque inicialmente un frente sin preferencias e intensifique a través del tiempo a través de preferencias.
- Emplear técnicas de cómputo paralelo para su ejecución en clúster.

A. Anexo: NSGA-II

NSGA-II es un algoritmo popular en la literatura propuesto por (Deb, Pratap, Agarwal, & Meyarivan, 2002), NSGA-II es un algoritmo genético basado en la no dominancia. La población se inicializa al igual que en un genético, entonces la población se ordena en frentes a través del algoritmo FastNonDominatedSort donde en el primer frente se encuentran todas las soluciones no dominadas, en el segundo frente se encuentran las soluciones solo dominadas por el primer frente, así hasta ordenar la población completa. El pseudocódigo Algoritmo 5 muestra el FastNonDominatedSort.

Algoritmo 5. Pseudocódigo de FastNonDominatedSort

```
1   fast-non-dominated-sort(P)
2   Inicio
3     Para cada p ∈ P
4       S_p = ∅
5       n_p = 0
6       Para cada q ∈ P
7         Si (p ≺ q) Entonces                    Si p domina a q
8           S_p = S_p ∪ {q}                      Soluciones dominadas por p
9         De otro modo Si (q ≺ p)
          Entonces
10          n_p = n_p + 1
11      Si n_p = 0 Entonces                      Nadie domina a p
12        p_rank = 1
13        F_1 = F_1 ∪ {p}
14    i = 1
15    Mientras F_i ≠ ∅
16      Q = ∅
17      Para cada p ∈ F_i
18        Para cada q ∈ S_p
19          n_q = n_q − 1
20          Si n_q = 0 Entonces
21            q_rank = i + 1
22            Q = Q ∪ {q}
23      i = i + 1
24      F_i = Q
25  Fin
```

Después del ordenamiento en frentes se asigna un fitness a cada individuo de la población dependiente del frente en el que se encuentre y su *Crowding Distance* Algoritmo 6.

El *Crowding Distance* es una métrica de distancia entre soluciones propuesta por el mismo Deb, la *Crowding Distance* de un punto determinado se mide a partir de la distancia entre las dos soluciones más cercanas; La Figura 18 ilustra el concepto. Para lograr el cometido se requiere la ordenación de las soluciones por cada uno de los objetivos, después, se calcula la distancia acumulada de todos las soluciones con sus inmediatas a través de todos los objetivos. Este proceso se describe en el Algoritmo 6.

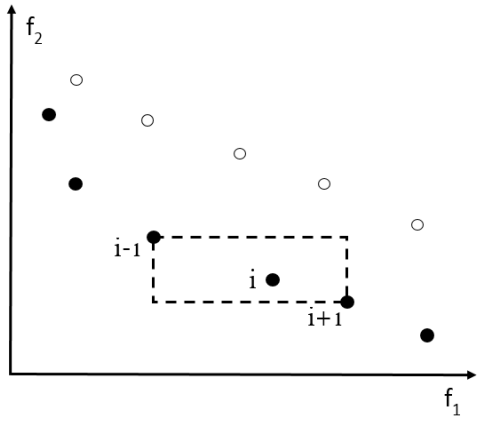

Figura 18. Ejemplo del cálculo de Crowding Distance

El algoritmo principal mostrado en el pseudocódigo Algoritmo 7 se comporta similar a un genético regular, la principal diferencia es la incorporación del *FastNonDominatedSort* y *CrowdingDistanceAssigment*; ambas estrategias incorporan un proceso de elitismo y se apoya la evolución de la población preservando la diversidad.

Algoritmo 6. Pseudocódigo del cálculo Crowding Distance

```
1   Crowding-distance-assignment(I)
2   Inicio
3     l = |I|
4     Para cada i, asignar I[i]_distance = 0        Inicializar estructuras
5     Para cada objetivo m
6       I = Ordenar(I, m)                           Ordenamiento por objetivo
7       I[1]_distance = I[l]_distance = ∞           Soluciones extremas
8       Para i = 2 Hasta l − 1
```

9	$I[i]_{distance} = I[i]_{distance} + (I[i+1].m - I[i-1].m)/(f_m^{max} - f_m^{min})$

Algoritmo 7. Pseudocódigo de NSGA-II

1	**NSGA-II (P)**
2	**Inicio**
3	Inicializar P_t y Q_t
4	**Mientras** Condición de parada no se cumpla
5	$R_t = P_t \cup Q_t$
6	$\mathcal{F}$=fast-non-dominated-sort(R_t)
7	$P_{t+1} = 0$ y $i = 1$
8	**Mientras** $\lvert P_{t+1}\rvert + \lvert\mathcal{F}_i\rvert \leq N$
9	crowding-distance-assigment($\mathcal{F}_i$)
10	$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
11	i=i+1
12	**Ordenar por Frentes y Crowding**
13	$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1:(N - P_{t+1})]$
14	$Q_{t+1} = crear\ poblacion\ apartir\ de\ P_{t+1}$
15	$t = t + 1$
16	**Fin**

B. Anexo: JMetal

Introducción a JMetal

Jmetal es un *framework* diseñado para realizar optimización multi-objetivo a través de metaheurísticas implementadas en su biblioteca. El propósito del software es ayudar a los investigadores a concentrar su tiempo en la aportación científica que realizan y no en tiempo de codificación para implementar las metaheurísticas de uso popular.

JMetal está implementado en el lenguaje de programación Java y hace un fuerte uso del paradigma de programación orientado a objetos y patrones de diseño, el software trata de sintetizar en un solo framework las necesidades del investigador relacionadas con la codificación de problemas de optimización multi-objetivo.

JMetal incluye diversas metaheurísticas, métodos de selección, recombinación y mutación, además representación de variables booleanas, continuas y discretas. Como parte de sus atributos incluye módulos orientados a la experimentación con métricas de desempeño y configuración de experimentos, Además JMetal incluye problemas base (benchmarks) del estado del arte los cuales ayudan en la validación de las heurísticas.

A la fecha de escritura de este trabajo JMetal se encuentra en su versión 4.5 con una versión 5.0 en su fase beta, la versión 5.0 trae consigo una reestructuración del núcleo reduciendo el número de clases base de siete a cinco y simplificado la codificación de variables.

Estructura de JMetal

La Figura 19 muestra la estructura general de JMetal de la versión 4.5 a través de un diagrama de clases UML. El núcleo de JMetal se conforma de las clases *Algorithm, Problem,*

Solution, SolutionType, Variable y Operator, cada una con una función específica, representando cada aspecto de un algoritmo metaheurístico.

Los paquetes de clases contenidos en JMetal se distribuyen en ocho. Una descripción de cada paquete se incluye en la Tabla 19.

Nombre del paquete	**Descripción del paquete**
core	El paquete contiene las clases base necesarias para utilizar JMetal (Algorithm, Problem, Solution, SolutionType, Variable y Operator)
problems	Aquí se incluyen las clases relacionadas con todos los problemas de optimización incluidos en JMetal
metaheuristics	Este paquete incluye los algoritmos metaheurísticos utilizados en JMetal, asi como las configuraciones de los mismos.
operators	El paquete contiene todos los operadores que se pueden aplicar a las soluciones desde selección, cruza, muta e incluso búsquedas locales.
encondings	El paquete maneja los tipos de variables y soluciones necesarios para representar correctamente un set de soluciones.
util	Este paquete incluye utilidades necesarias para la ejecución como generadores pseudoaleatorios, lectura de archivos, extracción de frentes, comparadores de solución, entre otros.
qualityIndicator	El paquete incluye métodos desarrollados para medir la calidad de los resultados, incluye las métricas más populares en el estado del arte.
experiments	El paquete incluye clases diseñadas para realizar experimentaciones y controlar las mismas.

Tabla 19. Descripción de los paquetes de JMetal.

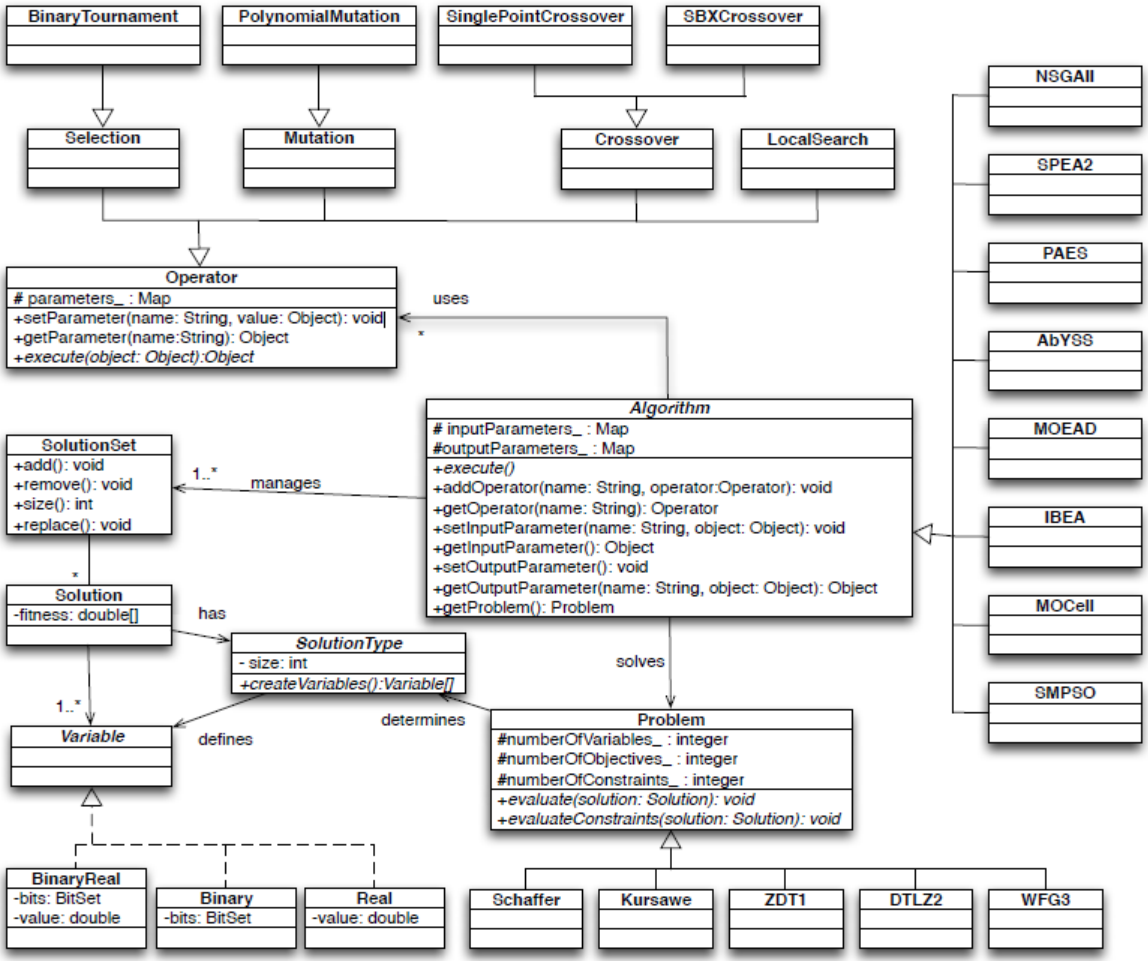

Figura 19. Diagrama de Clases de JMetal 4.5

Caso de estudio

Consideremos el siguiente caso de estudio; deseamos implementar un algoritmo NSGA-II para el problema DTLZ1 (Deb, Thiele, Laumanns, & Zitzler, 2002). Para el desarrollo de la heurística se utiliza la clase NSGAII_main.java que se encuentra en el paquete *metaheuristics*. Una versión reducida del código fuente se muestra en el Algoritmo 8.Lo más destacable del algoritmo se observa en las líneas 18 y 19, donde se declara el tipo de problema y el algoritmo que lo resolverá, para agregar un problema diferente se pueden sobrescribir los métodos *evalute* y *evaluateconstraints* en una clase nueva que herede de *Problem*. Además en las líneas 27, 31 y 34 se definen las operaciones de selección, mutación y recombinación que se aplicarán a las soluciones.

Es importante considerar el tipo de codificación de las variables que se utilizan dado que definen el tipo de operaciones que pueden aplicarse a las soluciones. Por ejemplo, si tenemos una representación *Binary* (arreglo de bits) no podemos utilizar una operación SBXCrossover porque está diseñada para representaciones del tipo *Real*.

La ejecución del Algoritmo 8 producirá dos archivos ubicados en la raíz del proyecto llamados VAR y FUN. VAR incluye todas las soluciones a nivel de variables que se encontraban en el Frente de Pareto encontrado al finalizar la ejecución, FUN incluye las mismas soluciones pero con sus valores objetivo.

Algoritmo 8 Clase de configuración del NSGA-II de JMETAL

```
1 public class NSGAII_main {
2   public static Logger      logger_ ;      // Logger object
3   public static FileHandler fileHandler_ ; // FileHandler object
4
5   public static void main(String [] args) {
6     Problem   problem   ; // The problem to solve
7     Algorithm algorithm ; // The algorithm to use
8     Operator  crossover ; // Crossover operator
9     Operator  mutation  ; // Mutation operator
10    Operator  selection ; // Selection operator
11    HashMap  parameters ; // Operator parameters
12    QualityIndicator indicators ; // Object to get quality indicators
13    // Logger object and file to store log messages
14    logger_      = Configuration.logger_ ;
15    fileHandler_ = new FileHandler("NSGAII_main.log");
16    logger_.addHandler(fileHandler_) ;
17    indicators = null ;
18    problem = new DTLZ1("Real");
19    algorithm = new NSGAII(problem);
20    // Algorithm parameters
21    algorithm.setInputParameter("populationSize",100);
22    algorithm.setInputParameter("maxEvaluations",25000);
23    // Mutation and Crossover for Real codification
24    parameters = new HashMap() ;
25    parameters.put("probability", 0.9) ;
26    parameters.put("distributionIndex", 20.0) ;
27    crossover = CrossoverFactory.getCrossoverOperator("SBXCrossover", parameters);
28    parameters = new HashMap() ;
29    parameters.put("probability", 1.0/problem.getNumberOfVariables()) ;
30    parameters.put("distributionIndex", 20.0) ;
31    mutation = MutationFactory.getMutationOperator("PolynomialMutation",
parameters);
```

```
32    // Selection Operator
33    parameters = null ;
34    selection = SelectionFactory.getSelectionOperator("BinaryTournament2",
parameters) ;
35    // Add the operators to the algorithm
36    algorithm.addOperator("crossover",crossover);
37    algorithm.addOperator("mutation",mutation);
38    algorithm.addOperator("selection",selection);
39    // Add the indicator object to the algorithm
40    algorithm.setInputParameter("indicators", indicators) ;
41    // Execute the Algorithm
42    long initTime = System.currentTimeMillis();
43    SolutionSet population = algorithm.execute();
44    long estimatedTime = System.currentTimeMillis() - initTime;
45    population.printVariablesToFile("VAR");
46    population.printObjectivesToFile("FUN");
47  } //main
48 } // NSGAII_main
```

Referencias

Alba, E., & Dorronsoro, B. (2009). *Cellular genetic algorithms.* Springer Science & Business Media.

Alba, E., & Troya, J. M. (1999). A survey of parallel distributed genetic algorithms. *Complexity*, 31-52.

Alba, E., Luna, F., & Nebro, A. (2004). Advances in parallel heterogeneous genetic algorithms for continuous optimization. *International Journal of Applied Mathematics and Computer Science*, 317-334.

Andersson, J. (2000). *A survey of multiobjective optimization in engineering design.* Linkoping, Sweden.

Balderas, A. (2012). *Sistema de Apoyo a la Decisión para la Selección.* Madero: Trabajo de Tesis.

Bastiani, M. (Abril de 2013). Solución de problemas de cartera de proyectos públicos a partir de información del ranking de prioridades. *Propuesta de Tesis*. Madero.

Bastiani, S. S., Cruz, R., Fernandez, E., Gómez, C., & Rivera, G. (2015). An Ant Colony Algorithm for Solving the Selection Portfolio Problem, Using a Quality-Assessment Model for Portfolios of Projects Expressed by a Priority Ranking. *Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization*, pp 357-373.

Benayoun, R., De Montgolfier, J., Tergny, J., & Laritchev, O. (1971). Linear programming with multiple objective functions: Step method (STEM). *Mathematical programming*, 366-375.

Blickle, T., & Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 268-308.

Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 141-171.

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

Coello, C. A. (2015). *Introduccion a la Computacion Evolutiva (Notas de Curso).*

Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems.* New York: Kluwer Academic.

Collette, Y., & Siarry, P. (2013). *Multiobjective Optimization: Principles and Case Studies.* Springer Science & Business Media.

De Jong, K. A., & Spears, W. M. (1991). An Analysis of of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. *In Parallel problem solving from nature*, 38-47.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 182-197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002). Scalable multi-objective optimization test problems. *Proceedings of the Congress on Evolutionary Computation*, 825-830.

Delaossa, L., Gamez, J., & Puerta, J. M. (2006). Initial approaches to the application of islands-based parallel EDAs in continuous domains. *Parallel Processing*, 580-587.

Díaz, B. (2007). *Diseño e implementación de algoritmos genéticos celulares para problemas complejos.* Tesis Doctoral,Universidad de Málaga.

Díaz, M. (2006). *La elección de un individuo (Tesis).*

Doerner, K., Gutjahr, W. J., Hartl, R., Strauss, C., & Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of operations research*, 79-99.

Feo, T., & Resende, M. (1995). Greedy Randomized Adaptative Search Procedures. *Journal Global Optimization*, 109-133.

Fernández, C., Gómez, N., & Guerrero, C. M. (2008). Evaluación y clasificación de las técnicas utilizadas por las organizaciones, en las últimas décadas, para seleccionar proyectos. *Métodos cuantitativos para la economía y la empresa*, 67-115.

Fernández, E., López, F., & Coello, C. A. (2010). Increasing selective pressure towards the best compromise in evolutionary multiobjective optimization: The extended NOSGA method. *Information Sciences*, 44–56.

Gandibleux, X., & Freville, A. (2000). Tabu Search Based Procedure for Solving the 0-1 MultiObjective Knapsack Problem: the two objectives case. *Journal of Heuristics*, 361-383.

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

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

Garfinkel, R. S., & Nemhauser, G. L. (1972). *Integer programming.* NewYork: Wiley.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 533-549.

Goldberg, D. (1989). *genetic algorithms in search optimization and machine learning.* Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Ishibuchi, H., Tsukamoto, N., & Nojima, Y. (2008). Evolutionary many-objective optimization: A short review. *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, (págs. 2419-2426).

Jozefowiez, N., Semet, F., & Talbi, E.-G. (2002). *Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem.* Springer Berlin Heidelberg.

Karp, R. M. (1972). Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, 85-103.

Martínez, V. (2015). *Método de solución basado en un algoritmo de búsqueda dispersa multiobjetivo para el problema de la cartera de proyectos.* CD.Madero: Instituto Tecnologico de CD.Madero.

Osyczka, A. (1985). Multicriteria optimization for engineering design. En Design Optimization,. *Academic Press*, 155-183.

R Development Core Team. (1997). *The R Project for Statistical Computing*. Obtenido de https://www.r-project.org/

Resende, M., & González, V. (2003). GRASP: Procedimientos de Búsqueda Miopes Aleatorizados y Adapatativos. *Inteligencia Artificial*, 61-76.

Rivera, G., Cruz, L., Fernandez, E., Gómez, C., & Pérez, F. (2014). Many-Objective Portfolio Optimization of Interdependent Projects with 'a priori' Incorporation of Decision-Maker Preferences. *Appl. Math*, 1517-1531.

Roy, B. (1991). Reading in multiple criteria decision aid, chapter The Outranking Approach and the Foundations of ELECTRE methods, pages 155–183. Springer-Verlag, 1990. *Theory and decision*, 49-73.

Santana, Q. L. (2004). *Un Algoritmo Basado en Evolución Diferencial para Resolver Problemas Multiobjetivo.* Mexico D.F.: Tesis de Maestría CINVESTAV-IPN.

Schrijver, A. (1998). *Theory of linear and integer programming.* John Wiley & Sons.

Serrano, H. V. (2007). *Métodos para reducir evaluaciones en algoritmos.* Mexico D.F: Tesis de Maestría CINVESTAV-IPN.

Terán, V., Fraire, H., Carpio, V., Puga, S., & Martínez, F. (2012). Cellular processing scatter search for minimizing power consumption on wireless communications systems. *HPCS.* Madrid.

Van Veldhuizen, D., & Lamont, G. (1998). Evolutionary computation and convergence to a pareto front. *In Late breaking papers at the genetic programming 1998 conference* (págs. 221-228). Stanford University.

Wilcoxon, F., Katti, S. K., & Wilcox, R. A. (1963). Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. *American Cyanamid Comp*.

Woeginger, G. J. (2003). Exact algorithms for NP-hard problems: A survey. *In Combinatorial Optimization—Eureka, You Shrink!*, 185-207.

Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., & Tsang, E. (2006). Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. *In in Proceedings of the Congress on Evolutionary Computation* (págs. 3234–3241). IEEE Press.