CENTRO DE INVESTIGACIÓN EN
MATEMÁTICAS, A.C.

TESIS DE MAESTRÍA

Optimización de Estructuras de Barras Utilizando Metaheurísticas

Presenta:
María Gracia MORENO VILLAMIZAR

Director de Tesis:
Dr. Carlos SEGURA
Dr. Salvador BOTELLO

Para obtener el grado de:

MAESTRO EN CIENCIAS CON ESPECIALIDAD DE COMPUTACIÓN Y MATEMÁTICAS INDUSTRIALES

Área de Ciencias de la Computación

Noviembre, 2019

Dedicado a
mi madre y a mi patria Venezuela
a las cuales llevo en mi corazón

Agradecimientos

A mi madre, que desde la distancia me dio apoyo y amor incondicional. Gracias mami.

A mis tutores, el Dr. Carlos Segura y el Dr. Salvador Botello por el tiempo dedicado, por sus ideas, por compartir su conocimiento sin reparar en ello y por su paciencia durante los últimos meses, muchas gracias.

Al Dr. Arturo Hernández, quién fue mi tutor desde que llegué a CIMAT, gracias igualmente por su tiempo y por todo lo aprendido.

A Joel Chacón, quién me contagió de su emoción por el tema, quien dedicó horas a explicarme en su cubo, y fue generoso con sus ideas y conocimiento. Gracias Joel.

A la hermana mayor que me dió el CIMAT Natalia Cardona, quien siempre estuvo en los días buenos, en los días malos y en los muy malos. Gracias Nati, no sé que hubiese hecho sin ti.

A Emilio Romero, Edith López y Favia Avilés, gracias por creer en mi incluso cuando yo no lo hacia, gracias por ser mi familia más que mis amigos.

A mi compañeros de cubo D712 Daniela Zavala, Erick Alvarez, Bernardo Martínez y Nancy Mejía, hicieron que CIMAT fuese un lugar al que siempre quisiera llegar. Gracias por toda su ayuda.

A Andrea Carrasquel, Reyna Pernía, Pierina Villegas, Mariana Estrada, Oriana Zambrano, Freddy Contreras, Carlos Moreno, Dayana Toro, Daniel Caicedo, me transmitieron fortaleza y certezas cuando lo necesité. Muchísimas Gracias.

Y por último y no menos importante al Dr. Jean-Bernard Hayet, quien fue el primer contacto con CIMAT, al propio CIMAT, a CONACYT y a México, por la oportunidad de aprender, crecer y vivir esta experiencia.

Resumen

Optimización de Estructuras de Barras Utilizando Metaheurísticas

María Gracia Moreno Villamizar

El problema de optimización de estructuras de barras es comúnmente encontrado en aplicaciones de ingeniería civil. Consiste en dada una estructura de barras tridimensional, seleccionar de un catálogo, las secciones (conjunto de características tales como: altura, base, resistencia del concreto, resistencia del acero y número de varillas) que serán asignadas a cada una de las barras que forman la estructura.

Para el enfoque *mono-objetivo* se busca que el peso total de la estructura sea minimizado, al tiempo que satisfaga un conjunto de restricciones de diseño, como la eficiencia y el desplazamiento, para esto se propone un algoritmo memético junto con un operador de cruce RCBX —Random Connected Bars Crossover— y un operador de reemplazamiento llamado BNP —Best Not Penalized—. Se propone, además, un enfoque *multi-objetivo* del problema, donde el desplazamiento y el peso propio serán los dos objetivos a minimizar junto a las restricciones antes mencionadas. Para abordar este nuevo planteamiento se utilizó el algoritmo NSGA II y el VSD-MOEA pasando luego a la implementación de una búsqueda local (LS) en ambos, siendo las propuestas con mejores resultados el NSGA II-LS y el VSD-MOEA-LS.

Tanto en el caso mono-objetivo como en el multi-objetivo se analiza el comportamiento de las estrategias al incluir operadores de control de la diversidad en el espacio de las variables. Otra contribución importante consistió en el diseño de un operador de cruce que posee la capacidad de heredar las características más importantes de las soluciones padres. Este operador fue crucial para obtener un algoritmo mono-objetivo lo suficientemente robusto como para encontrar soluciones factibles y de bajo costo para las dimensiones del problema, a su vez, los resultados obtenidos para la propuesta multi-objetivo muestran la relevancia de la hibridación en dichos esquemas.

Se propusieron 6 instancias/estructuras que cuentan con distintas dimensiones (cantidad de barras) y sometidas a distintas cargas que permitieron evaluar los aportes hechos a la resolución del problema.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Introducción

Existen diversos problemas relacionados con optimización de estructuras que surgen con frecuencia en aplicaciones de ingeniería civil. La variante tratada en esta investigación consiste en, dada una estructura de barras tridimensional, seleccionar de un catálogo de *concreto* dado, las secciones (conjunto de características tales como: altura, base, resistencia del concreto, resistencia del acero y número de varillas) que serán asignadas a cada una de las barras que conforman la estructura, de tal manera que el peso total (o costo) sea minimizado, al tiempo que satisfaga un conjunto de restricciones de diseño, como la eficiencia y el desplazamiento. Matemáticamente, este problema puede formularse en términos de *optimización combinatoria* (OC), y puede plantearse de manera mono-objetivo o multi-objetivo.

Debido a la importancia práctica y teórica de los problemas de OC, en la literatura se pueden encontrar muchos algoritmos para abordarlos. Estos algoritmos se pueden clasificar en un primer nivel como algoritmos exactos o aproximados. Los algoritmos exactos siempre encuentran una solución óptima del problema (si esta existe), mientras que en los algoritmos aproximados se sacrifica la garantía de encontrar soluciones óptimas para obtener buenas soluciones en un tiempo más reducido [3].

Entre los métodos aproximados están las Metaheurísticas, tales como los Algoritmos Evolutivos, que intentan lograr un equilibrio dinámico entre la exploración del

espacio de búsqueda (es decir, la fase de "diversificación" o localización de regiones promisorias) y la explotación de la experiencia de búsqueda acumulada (es decir, la fase de "intensificación" o refinamiento de soluciones). Este equilibrio permite identificar rápidamente regiones del espacio de búsqueda que contienen soluciones de alta calidad, así como omitir regiones que ya fueron exploradas o que están lejos de ser óptimas globales, y dedicar cierto esfuerzo computacional a realizar refinaciones de las soluciones encontradas [27]. Estas metaheurísticas han presentado una muy buena actuación en resolución de problemas de naturaleza combinatoria tales como el problema del vendedor ambulante (TSP), el problema de asignación cuadrática (QAP), y el problema de asignación de frecuencias (FAP). De hecho, en la mayor parte de problemas combinatorios NP-duros, las mejores soluciones conocidas para instancias grandes se han encontrado usando metaheurísticas.

Teniendo esto en cuenta y la complejidad del problema a tratar, en esta tesis se han adaptado y analizado variantes mono-objetivo y multi-objetivo de metaheurísticas a problemas de optimización de estructuras. En concreto, para el caso mono-objetivo se propusieron variantes de un Algoritmo Memético (AM), mientras que para el caso multi-objetivo se usaron variantes del NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) y del VSD-MOEA (*Multi-Objective Evolutionary Algorithm with Explicit Variable Space Diversity*). Tanto en el caso mono-objetivo como en el multi-objetivo se analiza el comportamiento de las estrategias al incluir operadores de control de la diversidad en el espacio de las variables. Otra contribución importante consistió en el diseño de un operador de cruce que posee la capacidad de heredar las características más importantes de las soluciones padres. Este operador fue crucial para obtener un algoritmo mono-objetivo lo suficientemente robusto como para encontrar soluciones factibles y de bajo costo para las dimensiones del problema (cantidad de barras de la estructura) que se consideraron en esta tesis, en prácticamente todas las ejecuciones, es decir, con independencia de la semilla utilizada en el generador aleatorio. En el caso multi-objetivo también se realizó diseño y análisis de algoritmos meméticos, siendo la incorporación del control de diversidad más compleja y problemática que la llevada a cabo en el caso mono-objetivo.

El principal propósito de esta investigación es desarrollar algoritmos poblacionales, y más específicamente algoritmos meméticos (AM), que resuelvan el problema de selección de secciones, así como analizar la importancia que tiene la inclusión de operadores de control de diversidad y operadores de cruce específicamente diseñados para el problema en cuestión, en el rendimiento de los optimizadores desarrollados.

1.2. Motivación

El proceso de adaptar un algoritmo evolutivo a un nuevo problema es un proceso complejo en el que habitualmente hay que rediseñar varias de sus múltiples componentes. Experimentalmente se ha visto que tres aspectos de gran relevancia para obtener soluciones de alta calidad al utilizar metaheurísticas poblacionales son la incorporación de esquemas de control de diversidad apropiados, el desarrollo de operadores de cruce específicamente diseñados para el problema, y la integración de mecanismos de intensificación. En el caso de la utilización de esquemas de control de diversidad sobre el espacio de las variables en los algoritmos evolutivos, se ha visto que han sido de gran ayuda en la mejora de las soluciones encontradas para ciertos problemas de *optimización combinatoria* (OC), entre los que se encuentran el problema de asignación de frecuencias (FAP) [41], el problemas de ruteo de vehículos [22], el problema de coloreado de grafos [42] y el problema de particionado de grafos [36]. Referente a los operadores de cruce se ha visto que una parte importante del poder exploratorio de un algoritmo evolutivo radica en la capacidad que tiene para crear nuevas buenas soluciones a partir de la combinación de soluciones previas existentes con rendimientos prometedores [16], aspecto que sólo ocurre cuando el operador de cruce es capaz de combinar características relevantes de las soluciones que se cruzan. Finalmente, en lo referente al proceso de intensificación es habitual realizar la hibridación de algoritmos poblacionales con metaheurística basadas en una solución, como la búsqueda local [47], con el fin de que la parte poblacional puedan identificar rápidamente regiones con buenas soluciones, pero que a la vez con la parte de intensificación se pueda obtener lo mejor de la región en cuestión.

En el caso del problema tratado en esta tesis, la revisión de la literatura relacionada muestra que estos aspectos no han sido ampliamente estudiados. La hipótesis principal de estudio es que para obtener optimizadores robustos capaces de resolver el problema mono-objetivo y multi-objetivo tratado en esta tesis se hace necesario considerar los tres aspectos anteriormente mencionados. Teniendo esto en cuenta, la principal motivación de esta tesis ha sido crear optimizadores heurísticos capaces de resolver el problema en cuestión, así como, comprender la importancia de cada uno de los tres aspectos anteriormente mencionados.

1.3. Descripción del problema

El concreto es un material similar a la piedra, los factores que hacen que el concreto sea un material de construcción universal son bastante relevantes desde los tiempos más primitivos hasta el presente, empezando con los morteros de cal desde los 12000 hasta los 6000 A.C. en Creta, Chipre y el Medio Oriente. La facilidad con la que puede depositarse y rellenarse formas y moldes es uno de los factores más resaltantes. Es resistente al fuego y a cualquier tipo de clima, lo cual es una evidente ventaja. Su resistencia a la compresión, como la de las piedras naturales, es alta, lo que lo hace adecuado para barras sujetas principalmente a compresión, como columnas.

Por otro lado, similar a las piedras naturales, el concreto es un material relativamente frágil cuya resistencia a la tracción es baja en comparación con su resistencia a la compresión. Esto evita su uso económico como único material de construcción en elementos estructurales que están sujetos a esfuerzo, ya sea completamente o en parte como sus secciones transversales (vigas). Para compensar esta limitación, se descubrió en la segunda mitad del siglo XIX que, se podía usar acero con su alta resistencia a la tracción para reforzar el concreto, principalmente en aquellos lugares donde su baja resistencia a la tracción limitaría la capacidad de carga del elemento. Las diferentes configuraciones de formas y tamaños de la sección transversal de las barras de concreto da pie a tener un catálogo de secciones donde se unen todas las posibles barras del material específico [28]. La elección del tipo de barras a utilizar presenta una fuente

de oportunidades ya que tiene un impacto considerable sobre el costo económico de la estructura, aunque también impacta sobre la seguridad estructural y la funcionalidad de la estructura resultante [30].

En el contexto de esta tesis, el diseño de la estructura en el caso ***mono-objetivo*** se convierte en un problema de optimización con restricciones que consiste en encontrar la sección transversal de cada elemento (tipo de barra) [30], seleccionando la sección a partir de un conjunto discreto, al que se denomina catálogo, de tal manera que la *eficiencia* se encuentre en el rango de 0 a 1 y el *desplazamiento* en el rango de 0 a 10, logrando así un valor permitido y donde el objetivo será minimizar el *peso* total de la estructura. Llamando $x(e)$ a la entrada de catálogo seleccionada para la barra e de la estructura, el problema se convierte en encontrar la solución que minimiza la siguiente función:

$$f(x) = \sum_{e} p_{x(e)} + \lambda_1(E(x) - 1) + \lambda_2(D(x) - 10) \tag{1.1}$$

Donde $p_{x(e)}$ es el peso específico de cada barra, $E(x)$ es la *máxima eficiencia de la estructura*, $D(x)$ es el *máximo desplazamiento de la estructura* y λ_i es un factor de penalización que se calcula en función de los valores $E(x)$ o $D(x)$ tal y como se expresa en las ecuaciones 1.2 y 1.3 y que permiten penalizar a las soluciones en las que los valores de E o D excedan los rangos establecidos, es decir, penalizar a las estructuras que no cumplan las restricciones.

$$\lambda_1 = \begin{cases} 0, & \text{si } E <= 1 \\ 1e12, & \text{otherwise} \end{cases} \tag{1.2}$$

$$\lambda_2 = \begin{cases} 0, & \text{si } D <= 10 \\ 1e12, & \text{otherwise} \end{cases} \tag{1.3}$$

Para el caso ***multi-objetivo***, se consideran como objetivos tanto el *peso de la estructura* $\sum_e p_{x(e)}$ como el *desplazamiento* D (Ver ec. 1.4 y 1.5), sumando a ambos objetivos los factores de penalización λ_i, con el propósito de que el frente de Pareto esté formado exclusivamente por soluciones factibles.

$$f_1(x) = \sum_{e}^{B} p_{x(e)} + \lambda_1(E(x) - 1) + \lambda_2(D(x) - 10) \tag{1.4}$$

$$f_2(x) = D(x) + \lambda_1(E(x) - 1) + \lambda_2(D(x) - 10) \tag{1.5}$$

En lo relativo a la *eficiencia* E si un elemento está trabajando a su capacidad máxima, tendrá un valor de eficiencia igual a uno, si está sujeto a una capacidad menor que la máxima, su eficiencia será menor que uno, y si está sobrecargado, su eficiencia será mayor que uno [30]. El cálculo de la eficiencia del concreto se lleva a cabo mediante un análisis local en los componentes de la barra y uno global de la sección completa considerando las combinaciones de esfuerzos de flexión-corte y flexión-axial. En el caso de Flexión, se consideran los momentos en dos direcciones, para cada uno de ellos se calcula el esfuerzo crítico de Euler para evaluar el pandeo y la torsión, el esfuerzo de torsión lateral crítico, las áreas efectivas de cada componente y los módulos de sección (basados en los tres valores previos). El momento nominal será el producto del módulo de sección debido al esfuerzo crítico de torsión lateral. La cizalladura se evalúa en una de las siguientes condiciones: pandeo elástico, pandeo inelástico o fluencia, dependiendo de la delgadez del alma. La carga de compresión nominal se calcula en función de los esfuerzos críticos de Euler y la esbeltez del elemento. La eficiencia final del elemento es el valor máximo de las combinaciones de esfuerzos antes mencionados.

El diseño e implementación de la simulación de la estructura para el cálculo de las eficiencias y desplazamientos no fue un objetivo de esta tesis, pues dicho método había sido desarrollado anteriormente [4].

1.4. Objetivos

El objetivo general de este proyecto es diseñar, adaptar y analizar algoritmos evolutivos mono-objetivo y multi-objetivo para el problema de optimización de estructuras, considerando la importancia de administrar la diversidad de forma explícita, de incorporar operadores de cruce adaptados al contexto y de incluir hibridación con

metaheurísticas de trayectoria.

Los objetivos específicos son:

- Determinar una codificación apropiada para los individuos a la hora de utilizar algoritmos evolutivos.
- Diseñar un operador de cruce adaptado al problema de optimización de estructuras de barras que tenga en cuenta cuáles son las propiedades importantes que se deben heredar.
- Implementar un algoritmo híbrido mono-objetivo que integre el operador diseñado y control de diversidad.
- Analizar los resultados del algoritmo mono-objetivo implementado y la importancia de los tres aspectos mencionados.
- Incluir el problema de optimización de estructura de barras en dos algoritmos evolutivos multi-objetivo que ya se encontraban desarrollados al empezar esta tesis.
- Extender los algoritmos multi-objetivo anteriores para desarrollar esquemas híbridos utilizando funciones de escalarización que permitan realizar búsquedas locales.
- Analizar los resultados de los algoritmos multi-objetivo implementados, contrastando la importancia de las diferentes componentes.

1.5. Organización de la tesis

El presente documento consta de seis capítulos donde se describe el trabajo realizado, los resultados obtenidos y las conclusiones. A continuación, se explica de forma breve el contenido de los siguientes capítulos.

El Capítulo 2 introduce al análisis de estructuras de barras así como el software MECA, en el cual se apoya la función de evaluación de todos los algoritmos diseñados/adaptados para dar solución al problema.

En el Capítulo 3 se hace una revisión del estado del arte que incluye algoritmos evolutivos mono-objetivo y multi-objetivo. Se enfoca principalmente en problemas de optimización combinatoria, ofreciéndose una recopilación de trabajos y conceptos relacionados estrechamente con los métodos propuestos en la tesis.

El Capítulo 4 está dedicado a la primera propuesta algorítmica de esta tesis. Se explica el esquema mono-objetivo general planteado y se definen cada una de las componentes a detalle. Se hace una validación experimental a través de pruebas estadísticas de los resultados arrojados por el algoritmo para ciertas instancias planteadas y se analizan dichos resultados.

En el Capítulo 5 se presenta la adaptación de dos algoritmos multi-objetivo presentes en la literatura al problema de optimización de estructuras de barras. En el mismo se incluyen comparativas entre diferentes esquemas, a través de una validación experimental basada en métricas y gráficas como el hipervolumen y las superficies de cubrimiento aplicadas a los resultados obtenidos con cada variante propuesta.

Por último en el Capítulo 6 se presentan las conclusiones y el trabajo futuro con base en los resultados obtenidos.

Capítulo 2

Estructuras de Barras

2.1. Descripción

La misión básica del diseño estructural es tratar con precisión la contradicción entre la seguridad estructural y la económica. Elegir un equilibrio razonable entre las contradicciones es fundamental para cumplir con el requisito estructural planificado y tener el costo mínimo.

La elección óptima de los elementos estructurales toma de esta manera gran importancia, razón por la que se desarrolló con anterioridad un software en lenguaje c++ llamado MECA [4]. Es un programa que calcula mediante el método matricial de rigideces las acciones mecánicas en cada barra de una estructura particular y el grado de eficiencia de la estructura (con base en las características geométricas —secciones— elegidas por un usuario o por un proceso de optimización).

2.2. Historia

¿Quién escribió por primera vez una matriz de rigidez o flexibilidad? La pregunta fue planteada en un artículo de 1995. La suposición hecha fue la siguiente "alguien que trabajó en la industria aeronáutica de Gran Bretaña o Alemania, a finales de la década de 1920 o principios de la de 1930". Desde entonces, los escritores han examinado informes y publicaciones de esa época. Estos trazan los orígenes del Análisis estruc-

tural matricial (MSA) hasta el grupo de aeroelasticidad del Laboratorio Nacional de Física (NPL) en Teddington, UK.

Se pudiese resumir entonces los principales pasos de la evolución de MSA al resaltar las contribuciones fundamentales de cuatro personas: Collar, Duncan, Argyris y Turner. Estas contribuciones se agrupan en tres hitos [18]:

Creación: A partir de 1930, Collar y Duncan formularon la aeroelasticidad discreta en forma de matriz. Los primeros dos artículos de revistas sobre el tema aparecieron en 1934-1935 [17, 13] y el primer libro que también incluyó de co-autor fue Frazer, en 1938 [20]. La representación y terminología para sistemas dinámicos discretos es esencialmente la que se usa hoy.

Unificación: En una serie de artículos de revistas que aparecieron en 1954 y 1955, Argyris presentó un informe formal sobre la unificación de métodos de fuerza y desplazamientos utilizando teoremas de energía dual, aunque estudió las aplicaciones de la dualidad resultaron efímeras, sin embargo este trabajo sistematizó el concepto de ensamblaje de Ecuaciones del sistema estructural a partir de componentes elementales.

Iniciación de FEM: En 1959, Turner propuso el Método de rigidez directa (DSM) como una implementación computacional eficiente y general del método de elementos finitos, entonces embrionario y, aún sin nombre.

Con el nacimiento de los microcomputadores, el uso de métodos matriciales alcanzó un extraordinario desarrollo debido a la posibilidad de efectuar cálculos a grandes velocidades.

2.3. Conceptos relevantes

Ley de Maxwell Esta es la llamada ley de las deflexiones recíprocas, y fue desarrollada por James Clerck Maxwell en 1864. Se considera esta ley de Maxwell un caso particular de la ley de Betti. Dicha ley se enuncia así: “Para una estructura linealmente elástica, la deflexión en un punto i debida a una carga unitaria aplicada en un punto j es igual a la deflexión en j debida a una carga unitaria en i”.

Equilibrio Todas las sumas algebraicas de los componentes en cada eje coordenado de todos los vectores fuerza que actúan sobre el cuerpo libre deben ser 0.

Compatibilidad Referido a la continuidad de los desplazamientos a lo largo de la estructura.

Análisis Estructural Es la determinación de desplazamientos y fuerzas internas de una estructura bajo cualquier condición de carga.

Rigidez Fuerza necesaria para mantener un elemento en una unidad de desplazamiento. Tiene unidades de fuerza por longitud.

Viga Una viga es un elemento prismático cuyas dimensiones en la sección transversal son considerablemente menores que en la longitudinal. Usualmente su eje longitudinal se encuentra en posición horizontal y las cargas son perpendiculares a este. Trabaja mecánicamente a flexión, sin embargo, su carga axial es tan pequeña que es posible despreciarla.

Armadura Una armadura es una estructura constituida por un conjunto de barras que suponemos articuladas en sus extremos y la carga que pueden resistir es la integrada por fuerzas concentradas en los nodos o articulaciones. Las barras de las armaduras solo resisten cargas axiales, en tensión o compresión.

Marco Un marco es aquel elemento estructural que podrá soportar tanto cargas axiales como cortantes y momentos flexionantes.

Grados de libertad Son los desplazamientos independientes (traslaciones y rotaciones) de los nodos que son necesarios para especificar la forma deformada de una estructura, cuando se vaya a sujetar a una carga arbitraria.

Sistemas de coordenadas Tanto la estructura como cada uno de sus elementos, se estudian respecto a un sistema de coordenadas ortogonales, cartesianas y de mano derecha. En el análisis matricial se consideran dos sistemas de coordenadas: locales y globales.

Coordenadas globales Son llamadas también coordenadas estructurales o de la estructura. Se denomina así debido a que respecto a estas se refieren todos los datos de la estructura en su conjunto, tales como la posición de los nudos, las cargas que actúan sobre ellos, sus desplazamientos y las reacciones de los apoyos.

Coordenadas locales Son llamadas también coordenadas particulares o del elemento. Se denominan así debido a que respecto a éstas se referencia todas las propiedades de los elementos, como las dimensiones y momentos de inercia, al igual que las cargas aplicadas sobre los mismos y las fuerzas internas a que se ven sometidos. Se definen colocando el eje x a lo largo del eje centroidal del elemento, colocando el origen del mismo en el nodo inicial. Los demás ejes (y, z) se definen teniendo en cuenta la ortogonalidad de los mismos. Con estas coordenadas queda definida la orientación del elemento estructural.

Transformación de coordenadas Cuando los miembros de una estructura están orientados en direcciones diferentes es necesario transformar las relaciones de rigidez de cada miembro, del sistema de coordenadas locales del mismo, hacia un sistema común de coordenadas globales. Luego se combinan las relaciones de rigidez de los miembros así obtenidas, a fin de establecer las relaciones de rigidez para la estructura completa.

2.4. Análisis Estructural a través del método matricial de rigideces

Los Métodos matriciales consisten en reemplazar la estructura continua real por un modelo matemático de elementos estructurales finitos, cuyas propiedades pueden expresarse en forma matricial. El proceso de análisis se puede considerar como:

I. Acción sobre la estructura.

II. Acción sobre los elementos.

III. Respuesta de los elementos.

IV. Respuesta de la estructura.

Los métodos matriciales tienen dos grandes variantes: el método de la flexibilidad en el cual las incógnitas son las fuerzas y el método de la rigidez en el cual las incógnitas son los desplazamientos [34].

$$\{\mathrm{P}\} = [\mathrm{K}].\{\mathrm{d}\} \qquad\qquad \{\mathrm{d}\} = [C].\{P\}$$

Método de los desplazamientos o rigidez Método de las fuerzas o flexibilidad

(2.1)

donde P =representa las cargas, K =la rigidez de la estructura, d =los desplazamientos y C= flexibilidad de la estructura. Para poder encontrar la relación entre fuerzas y desplazamientos de cualquier sistema estructural es necesario utilizar las propiedades elásticas y geométricas del material dado, así como los conceptos de **equilibrio** y **compatibilidad**.

El *Método de rigidez* consiste en restringir los desplazamientos de la estructura de tal manera que cada elemento de esta pueda analizarse de forma individual. La restricción de los desplazamientos viola las ecuaciones de equilibrio, el cual debe ser restablecido (la solución determinará entonces los desplazamientos necesarios para restablecerlo). Los desplazamientos restringidos serán las incógnitas en el sistema de ecuaciones, buscando establecer a través del equilibrio y compatibilidad la relación que hay entre las cargas y los desplazamientos que estas generan en la estructura. A partir de esta relación se pueden conocer los desplazamientos en los nodos de la estructura, y a partir de estos los elementos mecánicos en cada una de las barras que forman las estructuras. Este es el método utilizado en el software MECA [4].

Para construir la matriz de rigidez, se enfoca primero en la rigidez de un elemento estructural que se entiende comúnmente como la **fuerza requerida** para producir un **desplazamiento unitario**. Se tendrán tantos tipos de rigideces como elementos mecánicos, es decir, rigidez axial al corte, a la flexión y a la torsión. Y partiendo de los principios de **resistencia de materiales**, **análisis estructural** y **estática** sabemos que para cada barra sometida a una carga se obtiene la matriz de rigidez en coordenadas locales, vale acotar que los coeficientes de rigidez se denotan como k_{ij} , donde j es el grado de libertad que se hace igual a 1, e i es el grado de libertad en donde se induce una fuerza o momento de acuerdo al desplazamiento unitario de j, y además de acuerdo con el teorema de Maxwell, se tiene que $k_{ij} = k_{ji}$, por lo que la matriz será simétrica:

$$\begin{bmatrix} x'_1 \\ y'_1 \\ z'_1 \\ M'_{x_1} \\ M'_{y_1} \\ M'_{z_1} \\ x'_2 \\ y'_2 \\ z'_2 \\ M'_{x_2} \\ M'_{y_2} \\ M'_{z_2} \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_{zz}}{L^3} & 0 & 0 & 0 & \frac{6EI_{zz}}{L^2} & 0 & -\frac{12EI_{zz}}{L^3} & 0 & 0 & 0 & \frac{6EI_{zz}}{L^2} \\ 0 & 0 & \frac{12EI_{yy}}{L^3} & 0 & -\frac{6EI_{yy}}{L^2} & 0 & 0 & 0 & -\frac{12EI_{zz}}{L^3} & 0 & -\frac{6EI_{yy}}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_{yy}}{L^2} & 0 & \frac{4EI_{yy}}{L} & 0 & 0 & 0 & \frac{6EI_{yy}}{L^2} & 0 & \frac{2EI_{yy}}{L} & 0 \\ 0 & \frac{6EI_{zz}}{L^2} & 0 & 0 & 0 & \frac{4EI_{zz}}{L} & 0 & -\frac{6EI_{zz}}{L^2} & 0 & 0 & 0 & \frac{2EI_{zz}}{L} \\ -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_{zz}}{L^3} & 0 & 0 & 0 & -\frac{6EI_{zz}}{L^2} & 0 & \frac{12EI_{zz}}{L^3} & 0 & 0 & 0 & -\frac{6EI_{zz}}{L^2} \\ 0 & 0 & -\frac{12EI_{yy}}{L^3} & 0 & \frac{6EI_{yy}}{L^2} & 0 & 0 & 0 & \frac{12EI_{yy}}{L^3} & 0 & \frac{6EI_{yy}}{L^2} & 0 \\ 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_{yy}}{L^2} & 0 & \frac{2EI_{yy}}{L} & 0 & 0 & 0 & \frac{6EI_{yy}}{L^2} & 0 & \frac{4EI_{yy}}{L} & 0 \\ 0 & \frac{6EI_{zz}}{L^2} & 0 & 0 & 0 & \frac{2EI_{zz}}{L} & 0 & -\frac{6EI_{zz}}{L^2} & 0 & 0 & 0 & \frac{4EI_{zz}}{L} \end{bmatrix} \begin{bmatrix} u'_1 \\ v'_1 \\ w'_1 \\ \theta'_{x_1} \\ \theta'_{y_1} \\ \theta'_{z_1} \\ u'_2 \\ v'_2 \\ w'_2 \\ \theta'_{x_2} \\ \theta'_{y_2} \\ \theta'_{z_2} \end{bmatrix}$$

$x'_i =$ Fuerza en dirección X local de la barra

$y'_i =$ Fuerza en dirección Y local de la barra

$z'_i =$ Fuerza en dirección Z local de la barra

$M'_{x_i} =$ Momento alrededor del eje X local de la barra

$M'_{y_i} =$ Momento alrededor del eje Y local de la barra

$M'_{z_i} =$ Momento alrededor del eje Z local de la barra

$I_{yy} =$ Momento de inercia de secc. transv. respecto al eje Y local

$I_{zz} =$ Momento de inercia de secc. transv. respecto al eje Z local

$u'_i =$ Desplazamiento en dirección X local de la barra

$v'_i =$ Desplazamiento en dirección Y local de la barra

$w'_i =$ Desplazamiento en dirección Z local de la barra

$\theta'_{x_i} =$ Rotación alrededor el eje X local de la barra

$\theta'_{y_i} =$ Rotación alrededor el eje Y local de la barra

$\theta'_{z_i} =$ Rotación alrededor el eje Z local de la barra

$E =$ Módulo de elasticidad de la barra

$L =$ Longitud de la barra

$A =$ Área de sección transversal de la barra

Para realizar un análisis a través del método programable de rigideces, se puede partir de un marco, que es una combinación entre los elementos armadura y los elementos viga. La forma de analizarlos dependerá del problema mismo, ya que al igual que en las armaduras tendremos distintas formulaciones para las matrices de rigidez,

una para elementos empotrados - empotrados y otra para empotrados - articulados.

Imaginando un marco en 3 dimensiones o un marco en el espacio, el cual es considerado como el elemento que al igual que un marco bidimensional es capaz de soportar tanto esfuerzos axiales, como cortantes, y momentos flexionantes; se tiene que, las cargas externas pueden ser aplicadas en dirección arbitraria. Existirán 6 grados de libertad por nodo referentes al desplazamiento posible en dirección global X, Y o Z y una rotación por cada eje global X,Y o Z [51].

Considerando a cada barra en el espacio para obtener la matriz de rigidez correspondiente, es importante señalar que debido a que la sección transversal de la barra puede tener una dirección arbitraria, se planteará la matriz antes referida para coordenadas locales y que además contemple la propiedad geométrica utilizada para marcos planos y el momento de inercia de la sección (respecto a dos ejes para obtener su rigidez, el eje Y y el eje Z). Posteriormente se involucra una matriz de rotación que cambiará el sistema de referencia a uno global.

La obtención de la matriz de rotación [R] se basa en términos de un punto auxiliar que servirá para dar dirección a la barra. El punto adicional deberá estar en el plano donde algún eje de la sección transversal de la barra coincida. La importancia de este método radica en que en algunas estructuras es complicado obtener ángulos de rotación que son necesarios en otros procedimientos, por tanto, se hace práctico utilizar un punto adicional que actúe en dicho plano [51].

El procedimiento comienza obteniendo la longitud de la barra que se desea rotar:

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{2.2}$$

Donde:

$L =$ Longitud de la barra $\quad x_i =$ Coordenada en eje X

$y_i =$ Coordenada en eje Y $\quad z_i =$ Coordenada en eje Z

Después se obtendrá un vector unitario $\hat{v}_z$ que se encuentre en la misma dirección que el eje longitudinal de la barra:

$$\hat{v}_x = \frac{(x_2 - x_1)}{L}\hat{i} + \frac{(y_2 - x_1)}{L}\hat{j} + \frac{(z_2 - z_1)}{L}\hat{k} \tag{2.3}$$

Donde:

$\hat{v}_x =$ Vector unitario con la misma dirección que el eje longitudinal de la barra.

$L =$ Longitud de la barra $\quad x_i =$ Coordenada en eje X

$y_i =$ Coordenada en eje $Y \quad z_i =$ Coordenada en eje Z

Ahora se tomará el punto adicional; es importante que este punto se encuentre en el mismo plano que el eje longitudinal de la barra, y se obtendrá un vector v_3 del punto 1 al punto 3 de la figura mostrada, que servirá para obtener un vector unitario v_z de la barra:

$$v_3 = (x_3 - x_1)\, i + (y_3 - y_1)\, j + (z_3 - z_1)\, k \tag{2.4}$$

Donde:

$\hat{v}_3 =$ Vector unitario con la misma dirección que el eje longitudinal de la barra.

$L =$ Longitud de la barra $\quad x_i =$ Coordenada en eje X

$y_i =$ Coordenada en eje $Y \quad z_i =$ Coordenada en eje Z

Se deberá calcular un producto cruz entre el vector v_3 y el vector unitario v_x para obtener el vector unitario v_z :

$$\hat{v}_z = v_3 \times \hat{v}_x \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ v_{3x} & v_{3y} & v_{3z} \\ v_{x_x} & v_{x_y} & v_{x_z} \end{vmatrix} \tag{2.5}$$

Se normaliza el resultado del producto cruz:

$$\hat{v}_z = v_{z_x}\hat{i} + v_{z_y}\hat{j} + v_{z_z}\hat{k} \tag{2.6}$$

Ahora por medio de otro producto cruz se deberá calcular un vector que sea perpendicular al vector unitario v_x y al vector unitario v_z , al cual llamaremos v_y:

$$\hat{v}_y = \hat{v}_x \times \hat{v}_z = v_{y_x}\hat{i} + v_{y_y}\hat{j} + v_{y_z}\hat{k} \tag{2.7}$$

Se organizan los vectores obtenidos en una matriz que relacione las coordenadas globales y las coordenadas locales:

$$\begin{bmatrix} \hat{v}_x \\ \hat{v}_y \\ \hat{v}_z \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = \begin{bmatrix} v_{xx} & v_{xy} & v_{xz} \\ v_{yx} & v_{yy} & v_{yz} \\ v_{zx} & v_{zy} & v_{zz} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{2.8}$$

Obteniéndose las siguientes relaciones:

$$\begin{aligned} X_i' &= RX_i \\ M_i' &= RM_i \end{aligned} \tag{2.9}$$

Las cuales nos permiten obtener los vectores fuerza del sistema local a partir del vector fuerza del sistema global, y lo mismo sucede para los momentos. Siendo R la matriz de rotación. Sin embargo, para nuestro caso de estudio, nos interesa encontrar los vectores en sistema global a partir del desplazamiento obtenido en los nodos, por lo que se hará el siguiente procedimiento:

$$\begin{bmatrix} X_1' \\ M_1' \\ X_2' \\ M_2' \end{bmatrix} = [k]_{12\times 12} \begin{bmatrix} U_1' \\ \theta_1' \\ U_2' \\ \theta_2' \end{bmatrix} \tag{2.10}$$

Donde:

$X_i' =$ Conjunto de fuerzas en sistema local de un nodo de la barra

$M_i' =$ Conjunto de momentos en sistema local de un nodo de la barra

$U_i' =$ Conjunto de desplazamientos en sistema local de la barra

$\theta_i' =$ Conjunto de rotaciones en sistema local de la barra

$[k]_{12\times 12} =$ Matriz de rigidez local de la barra

Sustituimos las Ecuaciones 2.9 en la Ecuación 2.10:

$$\begin{bmatrix} R & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & R \end{bmatrix} \begin{bmatrix} X_1 \\ M_1 \\ X_2 \\ M_2 \end{bmatrix} = [k]_{12\times12} \begin{bmatrix} R & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & R \end{bmatrix} \begin{bmatrix} U_1 \\ \theta_1 \\ U_2 \\ \theta_2 \end{bmatrix} \tag{2.11}$$

Multiplicamos por R^T, realizamos las operaciones correspondientes y obtenemos el sistema en coordenadas globales:

$$\begin{bmatrix} X_1 \\ M_1 \\ X_2 \\ M_2 \end{bmatrix} = \begin{bmatrix} R^T & 0 & 0 & 0 \\ 0 & R^T & 0 & 0 \\ 0 & 0 & R^T & 0 \\ 0 & 0 & 0 & R^T \end{bmatrix}_{12\times12} [k]_{12\times12} \begin{bmatrix} R & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & R \end{bmatrix}_{12\times12} \begin{bmatrix} U_1 \\ \theta_1 \\ U_2 \\ \theta_2 \end{bmatrix} \tag{2.12}$$

Donde:

$X_i =$ Conjunto de fuerzas en sistema global de un nodo de la barra

$M_i =$ Conjunto de momentos en sistema global de un nodo de la barra

$U_i =$ Conjunto de desplazamientos en sistema global de la barra

$\theta_i =$ Conjunto de rotaciones en sistema global de la barra

2.5. Resumen

El método matricial requiere asignar a cada barra elástica de la estructura una matriz de rigidez, llamada matriz de rigidez local que dependerá de sus condiciones de enlace extremo (articulación, nudo rígido,...), la forma de sección transversal de la barra recta y las constantes elásticas del material de la barra (módulo de elasticidad longitudinal, módulo de elasticidad transversal y módulo de Corte). A partir del conjunto de matrices elementales y mediante un algoritmo conocido como acoplamiento que tiene en cuenta la conectividad de unas barras con otras se obtiene

una matriz de rigidez global, que relaciona los desplazamientos de los nudos con las fuerzas equivalentes sobre los mismos.

Igualmente a partir de las fuerzas aplicadas sobre cada barra se construye el llamado vector de fuerzas nodales equivalentes que dependen de las acciones exteriores (cargas) sobre la estructura. Junto con estas fuerzas anteriores deben considerarse las posibles reacciones sobre la estructura en sus apoyos o enlaces exteriores (cuyos valores son incógnitas). Finalmente se construye un sistema lineal de ecuaciones, para los desplazamientos y las reacciones de la estructura (todas son incógnitas). El número de reacciones incógnita y desplazamientos incógnita depende del número de nodos (N): es igual a $3N$ para problemas bidimensionales, e igual a $6N$ para un problema tridimensional. Este sistema siempre puede ser dividido en dos subsistemas de ecuaciones desacoplados que cumplen:

- Subsistema 1. Que agrupa todas las ecuaciones lineales del sistema original que sólo contienen **desplazamientos** como incógnita.

- Subsistema 2. Que agrupa al resto de ecuaciones, y que una vez resuelto el subsistema 1 y substituido sus valores en el subsistema 2 permite encontrar los valores de las **reacciones** incógnitas.

Una vez resuelto el subsistema 1 que da los desplazamientos, se substituye el valor de estos en el subsistema 2 que es trivial de resolver. Finalmente a partir de las reacciones, fuerzas nodales equivalentes y desplazamientos se encuentran los esfuerzos en las uniones de las barras a partir de los cuales pueden conocerse los esfuerzos en cualquier punto de la estructura y por tanto sus tensiones máximas, que permiten dimensionar adecuadamente todas las secciones de la estructura.

El cálculo estructural reporta las fuerzas actuantes en cada barra en coordenadas locales a cada elemento. Posteriormente cada elemento pasa por una rutina de cálculo de eficiencias, esto es, el porcentaje de uso de la capacidad del elemento. Si un elemento está trabajando a su máxima capacidad tendrá un valor de eficiencia igual a uno, si está sometido a una solicitación menor que el máximo, su eficiencia será menor a uno, y si está sobre esforzado su eficiencia será mayor a uno.

Estos valores de eficiencia, junto con el desplazamiento son los datos que adquieren importancia en un proceso de optimización. Para estructuras de concreto, la eficiencia final será el máximo entre la combinación de esfuerzos de Flexión con Tensión o Compresión (según sea el caso) y Cortante. La resistencia de los elementos de concreto se calcula para uno de los cuatro casos siguientes, Flexión pura, Compresión Pura, Flexo-Tensión y Flexo-Compresión, dependiendo de las acciones mecánicas actuantes y considerando la revisión por esbeltez. Para Tensión pura, la resistencia nominal será únicamente la proporcionada por el refuerzo de acero. Para Compresión pura, la resistencia nominal (P_0) será la suma de la resistencia del concreto más la resistencia proporcionada por el acero. Para Flexo- Compresión y Flexo-Tensión se calcula en forma individual la carga axial nominal resistente en cada eje (P_{nx} y P_{ny}), la resistencia nominal será obtenida mediante la aplicación de la fórmula de Bressler con P_{nx}, P_{ny} y P_0 como argumentos. La eficiencia en cualquier caso, será la relación entre la fuerza resistente y la fuerza actuante. Para este caso se utilizó la norma del American Concrete Institute (ACI, 2002) [30].

Capítulo 3

Optimización

Una vez que se ha modelado un problema, pueden surgir diversos problemas de optimización asociados que consideren dicho modelo. Por ejemplo, si disponemos de un modelo que permite saber si una estructura es capaz de aguantar o no unas determinadas fuerzas, nos podemos preguntar cuál es la estructura de menor peso que es capaz de aguantar dicha fuerza. Por ello, se usan diversos procedimientos de optimización que dependiendo de sus características generan soluciones que podrían ser óptimas o sub-óptimas. Es importante también tomar en cuenta que, al combinar modelación y optimización, a la hora de optimizar se intenta encontrar una solución para un modelo del problema y no para el problema de la vida real formulado originalmente. Por lo tanto, los rendimientos de la solución obtenidos son indicativos cuando el modelo es preciso. Por otro lado, cuanto más precisos son los modelos, más coste computacional suelen conllevar, lo que puede dificultar la aplicación de técnicas de optimización. Es por ello que en muchos casos debemos llegar a un balance entre la precisión del modelo, y la cantidad de recursos computacionales que podemos utilizar para la resolución del problema.

Un problema de optimización puede definirse a partir de la pareja (S, f), donde S representa el conjunto de soluciones factibles[1], y $f : S \to R$, es la función objetivo[2] a

[1]En algunos ámbitos a la solución también se le conoce como una configuración o un estado y al conjunto S se le denomina espacio de búsqueda o espacio de estados.

[2]También se le denomina función de costo cuando es a minimizar, o de utilidad o función fitness cuando es a maximizar.

optimizar. La función objetivo asigna a cada solución $s \in S$ del espacio de búsqueda un número real que cuantifica el valor de dicha solución. La función objetivo f permite definir una relación de orden entre cualquier par de soluciones en el espacio de búsqueda [46].

3.1. Definiciones

3.1.1. Óptimo Global

Se dice que la solución $s^* \in S$ es un óptimo global si tiene una mejor o igual función objetivo que todas las soluciones del espacio de búsqueda. Considerando un problema de minimización, s^* es un óptimo global si $\forall s \in S$, $f(s^*) \leq f(s)$. Idealmente, el objetivo a la hora de resolver un problema de optimización global es encontrar un óptimo global s^*. Sin embargo, dado que S puede ser inmenso, en muchos casos no será viable encontrar un óptimo global y el propósito será encontrar una solución cuyo valor objetivo se acerque lo más posible a la del óptimo global. También es importante mencionar que pueden existir muchas soluciones óptimas globales para un problema dado. En algunas variantes de optimización, como en optimización multi-modal, se pretende encontrar todos esos óptimos globales o incluso óptimos locales de gran calidad.

3.1.2. Métodos de Optimización Exactos

Los métodos exactos obtienen soluciones óptimas y garantizan su óptimalidad. Listamos a continuación los métodos más relevantes de este inciso:

- La programación dinámica se basa en la división recursiva de un problema en subproblemas más simples. Este método de optimización por etapas es el resultado de una secuencia de decisiones parciales. Esta técnica se basa en utilizar una tabla para almacenar las soluciones de subproblemas y evitar recalcular sus soluciones múltiples veces.

- El algoritmo de ramificación y unión y $A*$ se basan en una enumeración implícita de todas las soluciones del problema de optimización considerado. El espacio de búsqueda se explora construyendo dinámicamente un árbol cuyo nodo raíz representa el problema que se está resolviendo y todo su espacio de búsqueda asociado. Los nodos hoja son las posibles soluciones y los nodos internos son subproblemas del espacio total de la solución. La poda del árbol de búsqueda se basa en una función delimitadora que poda los subárboles que no contienen ninguna solución óptima.

Los métodos exactos se pueden aplicar a casos pequeños de ciertos problemas difíciles [46]. Desafortunadamente, cuando el tamaño de las instancias aumenta, el tiempo de ejecución para una búsqueda exhaustiva pronto se vuelve excesivamente grande, incluso para instancias de tamaño bastante pequeño [49] lo que está relacionado directamente con la complejidad del algoritmo y el tamaño del espacio de búsqueda.

3.1.3. Métodos de Optimización Aproximados

Los métodos aproximados (o heurísticos) funcionan de una forma bastante diferente. En la mayor parte de casos reciben como entrada un parámetro que está relacionado con la cantidad de cómputo que van a poder utilizar, como puede ser el límite de tiempo durante el que se va a realizar el proceso de optimización. Tras alcanzar dicho criterio de parada, devuelven la mejor solución que hayan encontrado. En la práctica se ha visto que cuando son configurados de forma adecuada son capaces de generar soluciones de calidad en tiempos razonables, aunque no hay garantía de encontrar una solución óptima global.

En la clase de métodos aproximados, se pueden distinguir dos subclases de algoritmos: **algoritmos de aproximación** y **algoritmos heurísticos**. Los algoritmos de aproximación proporcionan una calidad de solución demostrable, así como límites de tiempo de ejecución demostrables. Sin embargo, en muchos casos, el nivel de calidad asegurado no es suficiente o la cantidad de tiempo requerida no es admisible. Por otro

lado, las heurísticas no suelen asegurar nada en torno a la calidad, pero en la práctica suelen encontrar soluciones de alta calidad en tiempos razonables. De hecho, hay una gran cantidad de problemas NP-hard, así como problemas con espacios de búsqueda inmensos, en los que las mejores soluciones han sido encontradas mediante métodos heurísticos.

De esta forma, entre las ventajas que se le achacan a los algoritmos heurísticos cabe destacar que suelen encontrar soluciones buenas para problemas de gran tamaño donde otras técnicas no son aplicables y que permiten obtener un rendimiento aceptable a costos admisibles en una amplia gama de problemas. Los algoritmos heurísticos se pueden clasificar a su vez en dos familias: heurísticas específicas y metaheurísticas.

Heurísticas específicas

Las heurísticas específicas se adaptan y diseñan para resolver un problema y/o instancia específica y tienen la particularidad que no son extrapolables a otros problemas diferentes.

Metaheurísticas

Son algoritmos de propósito general que pueden aplicarse para resolver diversos problemas de optimización. Así, pueden verse como metodologías generales de nivel superior que pueden usarse como una estrategia para controlar e integrar un conjunto de heurísticas subyacentes con el fin último de resolver problemas de optimización específicos. Por tanto, podemos visualizar a las metaheurísticas como estrategias que definen un conjunto de componentes que hay que adaptar al problema en cuestión que se va a resolver. Dichas componentes que hay que desarrollar pueden ser heurísticas específicas o componentes de otra clase. Por ejemplo, a la hora de aplicar los algoritmos evolutivos, en general hay que definir una componente para cruzar dos o más soluciones candidatas, de forma que se genere un nuevo individuo que mezcle buenas componentes de estos. Habitualmente, encontrar la forma óptima de mezclarlos no es posible, pero se pueden desarrollar componentes que traten de hacer buenas combinaciones.

Las metaheurísticas han recibido cada vez más popularidad en los últimos 20 años y su uso actual en muchas aplicaciones muestra su eficiencia y eficacia para resolver problemas grandes y complejos. Algunas de las áreas donde las metaheurísticas han sido exitosas son las siguientes [46]:

- Diseño de ingeniería: optimización de topología y optimización estructural, aerodinámica, dinámica de fluidos, telecomunicaciones y robótica.
- Aprendizaje automático y minería de datos en bioinformática y biología computacional y finanzas.
- Modelado, simulación e identificación de sistemas en química, física y biología.
- Control, señalización y procesamiento de imágenes.
- Logística: planificación en problemas de enrutamiento, planificación de robots, problemas de programación y producción, transporte, y gestión de la cadena de suministro.

En 1950, Claude Shannon, un ávido jugador de ajedrez, publicó un artículo proponiendo ideas para programar una computadora que jugara ajedrez. En su artículo, Shannon distinguió entre lo que llamó estrategias de "tipo A" y "tipo B". Las estrategias de tipo A examinan todas las combinaciones posibles de movimientos, mientras que las estrategias de tipo B utilizan conocimientos especializados de ajedrez para centrarse en las líneas de juego que se consideran más productivas. Las estrategias de tipo B referían de lo que Allen Newell y Herbert Simon más tarde en 1960 llamaron *heurística* [29]. En el caso de las *metaheurísticas*, empezaron a ganar popularidad durante los años 60s, y actualmente podemos encontrar dos tipos principales de metaheurísticas: las metaheurísticas de trayectoria o basadas en una solución, y las metaheurísticas poblacionales o basadas en múltiples soluciones. En las próximas secciones se discuten ambas, ya que en esta tesis se hace uso de estos tipos de estrategias.

3.1.4. Optimización con Restricciones: Problemas y Optimizadores

Los problemas con restricciones (Ver Figura 3-1) son aquellos en los que no todas las soluciones presentes en el conjunto S son soluciones válidas. En este contexto, a las soluciones válidas se les llama soluciones factibles, mientras que el resto conforman las soluciones no factibles. En los casos más simples, no hay función objetivo, y por tanto el problema consiste en encontrar una solución factible. Sin embargo, también existen casos en los que hay una función objetivo, además de un conjunto de restricciones que se deben cumplir. En este tipo de problemas se tiene que buscar la mejor solución en S — en lo referente a la función objetivo — que además cumpla todo el conjunto de restricciones. Este tema tiene una gran relevancia práctica porque muchos de los problemas del mundo real tienen restricciones y es muy desafiante (tanto en la teoría como en la práctica) lidiar de forma simultánea con las restricciones y la función objetivo.

Atendiendo al caso de optimización con metaheurísticas también es un tema muy importante, ya que en la mayor parte de casos, las metaheurísticas se define para problemas de optimización global sin restricciones, y es por ello que hay que adaptarlas para el caso con restricciones. A continuación se presentan algunas de las estrategias de manejo de restricciones que se han integrado con las metaheurísticas para abordar este tipo de problemas.

	Función Objetivo	
Restricciones	SI	NO
SI	Problema de optimización con restricciones	Problema de satisfacción de restricciones
NO	Problema de optimización sin restricciones	No hay problema

Figura 3-1: Restricciones vs Función Objetivo

Se suele distinguir entre formas directa o indirectas de lidiar con las restricciones. Las formas directas son dependientes del problema e incluyen:

- Representaciones de soluciones especializadas, junto con inicializaciones y operadores de reproducción que reducen el espacio de búsqueda para asegurarse que todas las soluciones candidatas generadas por la metaheurística son factibles.

- Mecanismos de reparación capaces de transformar una solución no factible en una solución factible.

- Funciones decodificadoras: son una clase de mapeos desde el espacio genotipo S' a las regiones factibles F del espacio solución S. Idealmente tienen las siguientes propiedades:

  - Cada $s' \in S'$ debe asignarse a una sola solución $s \in F$.
  - Cada solución $s \in F$ debe tener al menos una representación $s' \in S'$.
  - Cada $s \in F$ debe tener la misma cantidad de representaciones en S' (no es necesario que sea 1).

  Dichas funciones decodificadoras proporcionan una forma relativamente simple de usar EA en algunos tipos de problemas como en el Job Shop, pero no están exentos de inconvenientes. Estos se centran en el hecho de que las funciones decodificadoras generalmente introducen mucha redundancia en el espacio del genotipo original cuando el mapeo es de muchos a uno, lo que significa que se pueden mapear varios genotipos radicalmente diferentes en el mismo fenotipo. Esto puede provocar problemáticas similares a las de convergencia prematura.

En el caso de mecanismos indirectos, lo más habitual es definir nuevas funciones de penalización que integran a la función objetivo penalizándola con base en las restricciones incumplidas. Por ello, al usar este tipo de estrategias las soluciones infactibles son consideradas durante el proceso de optimización, sin embargo la función de penalización incluirá mecanismos para penalizar a dicho tipo de soluciones. Existen múltiples formas de integrar dichas penalizaciones para generar la función de fitness [15].

Un caso típico consiste en usar la función fitness penalizada siguiente:

$$f(x)_p = f(x) + \lambda c(s) \ ,$$

donde $c(s)$ representa una medición de las restricciones incumplidas y λ es el peso que se da a la penalización. Las funciones de penalización pueden clasificarse de la siguiente manera [44]:

- Funciones de penalización estáticas: Un método simple para penalizar las soluciones inviables es aplicar una penalización constante a aquellas soluciones que violan la factibilidad de cualquier manera. La función de fitness penalizada sería la función objetivo no penalizada más un valor fijo (para un problema de minimización). Una variación de esta función de penalización es agregar una métrica basada en el número de restricciones violadas. Así, la función de penalización para un problema con m restricciones sería la siguiente:

$$f(x)_p = f(x) + \sum_{i=1}^{m} C_i \lambda_i$$

$$\text{donde} \begin{cases} \lambda_i = 1, \text{ si la restricción } i \text{ es violada} \\ \lambda_i = 0, \text{ si la restricción } i \text{ se satisface} \end{cases}$$

  $f(x)_p$ es la función de fitness penalizada, $f(x)$ es la función objetivo no penalizada y C_i es una constante impuesta por violación de la restricción i. Una penalización más sofisticada y en muchos casos más efectiva, consiste en incluir una medida de cuánto se está incumpliendo cada restricción y agregar una penalización que se vuelva más severa cuanto más lejos se encuentre la solución de la factibilidad. Nótese que en cierto modo aquí se está suponiendo que la medición realizada proporciona información sobre la proximidad de la solución a la zona factible. Estas mediciones de infactibilidad pueden ser continuas o discretas, y podrían ser lineales o no lineales. Una formulación general para un problema de minimización es la siguiente:

$$f(x)_p = f(x) + \sum_{i=1}^{m} C_i d_i^k$$

d_i es una medida del incumplimiento de la restricción i aplicada a la solución x, k es un exponente definido por el usuario, usando habitualmente valores de k igual a 1 o 2. C_i debe estimarse basándose en la escala de las diferentes medidas de incumplimiento de restricciones, la dificultad de satisfacer una restricción y la gravedad de una violación de restricción, o determinarse experimentalmente.

- Funciones de penalización dinámicas: La principal deficiencia de las funciones de penalización estáticas es la dificultad de determinar valores adecuados para los coeficientes C_i. Una variación de las funciones de penalización explicadas anteriormente consiste en incorporar un aspecto dinámico que, en general, aumenta los valores de penalización para un determinado nivel conforme transcurre el proceso de optimización. Esto tiene la propiedad de permitir soluciones altamente inviables al principio de la búsqueda (menor penalización), mientras que aumenta continuamente la penalización impuesta para eventualmente preferir siempre soluciones factibles frente a infactibles, aumentando así la probabilidad de alcanzar la región factible.

  Una forma general de un método de penalización que incorpora un aspecto dinámico basado en la duración de la búsqueda, t, es la siguiente para un problema de minimización:

  $$f(x,t)_p = f(x) + \sum_{i=1}^{m} s_i(t) d_k^i$$

  donde $s_i(t)$ es una función monótonamente creciente con t. Las t puede representar número de generaciones o el número de evaluaciones transcurridas.

  Una preocupación común de estas funciones de penalización dinámicas es que suelen localizar soluciones factibles solo hasta el fin de la evolución. Si $s_i(t)$ crece demasiado lento, pueden aparecer soluciones finales no factibles, y si $s_i(t)$ crece demasiado rápido, la búsqueda puede converger a soluciones factibles no óptimas, pues es posible que no busque prácticamente en las fronteras de la región factible, lo que es un aspecto negativo para muchas funciones. Estas

funciones de penalización requieren, en muchos casos, un ajuste específico para cada problema.

- Funciones de penalización adaptativas: Si bien la incorporación de una métrica de incumplimiento por cada restricción (d_i) mezcladas con información sobre la duración de la búsqueda en la función de penalización (t) ha sido generalmente efectiva, estas penalizaciones ignoran cualquier otro aspecto de la búsqueda. Por ejemplo, no se adaptan al éxito en curso (o la falta del mismo) durante el proceso de búsqueda y no pueden guiar la búsqueda a regiones particularmente atractivas o alejarse de regiones poco atractivas en base a lo que ya se ha observado. Una de las propuestas que extiende a las funciones de penalización anteriores consiste en variar la misma en función de la factibilidad o infactibilidad de la mejor solución encontrada durante las últimas generaciones. Esta función de penalización permite un aumento o una disminución de la penalización impuesta durante la evolución dependiendo de como esté transcurriendo el proceso de optimización Una de las formas más típicas de llevar a cabo este proceso consiste en seleccionar dos constantes, β_1 y β_2 ($\beta_1 > \beta_2 > 1$) que modificarán al multiplicador de la función de penalización, para actualizarlo adaptativamente según la factibilidad de la mejor solución en intervalos sucesivos de N_f generaciones. A medida que avanza la búsqueda, el multiplicador de la función de penalización se actualiza cada N_f generaciones en función de si la mejor solución fue factible o no durante este intervalo [1].

3.2. Optimización Mono-objetivo

En ingeniería hay muchos problemas que se formulan como problemas de optimización continua mono-objetivo con restricciones. Para estos casos, el problema puede ser descrito en forma matemática de la siguiente forma:

$$\text{Minimizar } f(x), \text{ tal que } x \in S, S \subseteq E \tag{3.1}$$

sujeto a

$$h_j(x) = 0, (j = 1, 2, ..., J), \tag{3.2}$$

$$g_k(x) \geq 0, (k = 1, 2, ..., K) \tag{3.3}$$

donde

$$x = (x_1, x_2, ...x_n)^T, \text{ para } (i = 1, 2, ..., n) \text{ dado que } x_i \in [x_i^l, x_i^u] \tag{3.4}$$

La ec. 3.4 es el vector diseñado que contiene las n variables de decisión, x, pueden ser continuas, enteras, binarias o cualquier combinación de los tipos anteriores y con limites inferior y superior, además que el conjunto al que pertenecen, S, puede estar sujeto a J restricciones de igualdad (ec. 3.2) y a K restricciones de desigualdad (ec. 3.3).

El espacio de solución E puede ser una representación de un conjunto finito y muy grande haciendo de la ec. 3.1 un problema de optimización combinatoria si $E = \mathbb{N}^n$ o un problema de optimización continua si $E = \mathbb{R}^n$. Además la función objetivo, puede no ser necesariamente lineal, continua o incluso convexa. El conjunto de factibilidad, S, puede no ser necesariamente convexo e incluso puede incluir subconjuntos desconectados.

En resumen, cuando el conjunto S es un conjunto discreto, el problema de optimización cae en la categoría de problemas de optimización discreta (también conocida como optimización combinatoria), mientras que si se trata de un conjunto continuo, se denomina un problema de optimización continua (también conocido como optimización global) [38].

En las siguientes secciones se van a describir algunas de las metaheurísticas más utilizadas para lidiar con este tipo de problemas, incluyendo las metaheurísticas basadas en una solución y las metaheurísticas poblacionales.

3.2.1. Metaheurísticas basadas en una sola solución

Las metaheurísticas basadas en una sola solución son aquellas que en cada instante mantiene una única solución activa, a la cual se le aplican un conjunto de transformaciones, para realizar una especie de caminata o trayectoria por el espacio de búsqueda. En muchos casos dichas "caminatas" están basadas en la noción de vecindarios aunque en algunos casos se usan otras nociones como perturbaciones o mutaciones. Las caminatas se realizan mediante procedimientos iterativos que pasan de una solución actual a otra en el espacio de búsqueda. Este tipo de metaheurísticas han mostrado su eficacia para abordar diversos problemas de optimización en diferentes dominios [46] y habitualmente son buenos para encontrar buenas soluciones en una región dada, aunque su poder como buscador global no suele ser tan bueno como el de los algoritmos poblacionales.

En este tipo de metaheurísticas se aplican iterativamente procedimientos de generación y reemplazo de una solución actual. En la fase de generación, se generan un conjunto de soluciones candidatas a partir de la solución actual. Este conjunto se obtiene generalmente mediante transformaciones locales de la solución, aunque puede tener en cuenta también otros aspectos, como una memoria de lo que ya ha sido recorrido. En la fase de reemplazo, se realiza una selección a partir de este conjunto de soluciones candidatas para reemplazar la solución actual, es decir, se selecciona una solución de este conjunto para que sea la nueva solución. En algunas metaheurísticas, incluso se puede mantener la solución actual y volver al proceso de regeneración de soluciones, para buscar soluciones alternativas. Este proceso se itera hasta un criterio de parada determinado.

La generación y las fases de reemplazo pueden ser sin memoria. En este caso, los dos procedimientos se basan únicamente en la solución actual. En otros casos se utiliza información sobre la búsqueda realizada hasta el momento, almacenando cierta información en una memoria. Algunos de los conceptos que se usan reiteradamente en este tipo de técnicas son los siguientes:

- Vecindario: La definición de un vecindario es un paso requerido para el diseño

de una la gran mayoría de metaheurísticas de trayectoria.

> Una solución, digamos $x \in S$, tiene un conjunto asociado de vecinos, digamos $N(x) \subset S$, y que es llamada la vecindad de x. Una solución, $x' \in N(x)$, puede obtenerse directamente de x por un operación que se conoce como movimiento (o transición m), digamos $x' = m(x)$. El mismo puede ser un solo movimiento que consiste en operaciones simples o complejas, o una serie de movimientos juntos que se pueden usar de manera secuencial bien definida y determinista, o de manera aleatoria o pseudoaleatoria.

La estructura del vecindario desempeña un papel crucial en el desempeño de este tipo de metaheurísticas, por lo que suele ser habitual analizar diferentes tipos de vecindarios hasta dar con uno adecuado para el problema en cuestión. Una de las propiedades deseables que deben tener los vecindarios es tener una alta localidad. Esto significa que los vecinos de una solución deben compartir muchas características con la misma, y las características que comparten deben ser aquellas que son significativas para el problema en cuestión.

- Generación de Solución Inicial: Se utilizan dos estrategias principales para generar la solución inicial: el enfoque aleatorio y el enfoque greedy. En el enfoque aleatorio cualquier solución candidata suele tener la misma probabilidad de ser elegida. En algunas ocasiones, algunas soluciones podrían ser más probables que otras, pero estas probabilidades no dependen directamente de una calidad estimada para la solución que se va a generar. En los enfoques greedy, se realiza algún cambio en el proceso de generación de soluciones iniciales, para tratar de muestrear con mayor probabilidad soluciones de mejor calidad. Uno de los problemas de este último tipo es que si el esquema se usa para generar múltiples soluciones iniciales, podría ocurrir que la búsqueda se sesgue hacia una región específica del espacio de búsqueda. El mejor desempeño de uno u otro esquema depende de muchos factores, como la metaheurística a usar, el tiempo que se va a dedicar a la optimización o si se van a incluir o no mecanismos de reinicio.

En general, el enfoque greedy se suele utilizar para acelerar la búsqueda, y no tanto para alcanzar soluciones de mejor calidad. Es habitual que los algoritmos greedy tengan una complejidad reducida, habitualmente siendo polinomial. Al conseguir comenzar con soluciones de mejor calidad, la metaheurística requerirá, en general, menos iteraciones para converger hacia un óptimo local. Algunos algoritmos greedy de aproximación también se pueden usar para obtener una garantía limitada para la solución final.

- Paisaje de Búsqueda: Está definido por la función objetivo junto a un grafo dirigido $G = (S, E)$. En el grafo G el conjunto de vértices S corresponde a las soluciones candidatas del problema, y el conjunto de aristas E unen soluciones que se consideran vecinas.

- Distancia en el espacio de búsqueda: En muchos casos, en el análisis o diseño de este tipo de metaheurísticas se usan nociones de distancia entre soluciones. Algunos ejemplos pueden ser la distancia de Hamming o la distancia Euclidiana.

El ejemplo más sencillo de metaheurística de trayectoria es la búsqueda local por escalada o Hill Climbing. Con el fin de escapar de óptimos locales, se proponen metaheurísticas más complejas como el Recocido Simulado, la Búsqueda Tabú o la Búsqueda Local Iterada. A continuación, se describe la búsqueda local por escalada — esta estrategia se aplica en esta tesis — y el Recocido Simulado, método que había sido aplicado anteriormente para abordar el mismo problema que se utiliza en esta tesis.

Búsqueda por Escalada

El Algoritmo 1 presenta el pseudocódigo de la búsqueda por escalada [46]. No mantiene un árbol de búsqueda o memoria, por lo que se trata de un algoritmo que se puede implementar de forma sencilla y rápida. La entrada del algoritmo será el número máximo de evaluaciones a la función objetivo (línea 1), para cada solución generada (vecinos) tendrá que evaluarse dicha función. Se inicializa una solución aleatoria como punto de partida (línea 3). El algoritmo sólo necesita almacenar la solución candidata

actual y su valor de la función objetivo correspondiente. La búsqueda por escalada consiste en aplicar un ciclo que moverá continuamente la solución actual a alguno de sus vecinos (línea 4), en otras palabras, no se mira hacia el "futuro" más allá del vecino generado (línea 5), si el nuevo vecino no mejora el valor de la función objetivo, se continuará con el ciclo sin actualizar la solución (línea 6). En caso de que mejore el valor de la función objetivo se actualizara la solución actual (línea 7). Al final se obtiene como salida la mejor solución encontrada durante n evaluaciones de función (línea 2, 8).

Algoritmo 1: Búsqueda por escalada

```
1 Entrada: n evaluaciones a la función objetivo
2 Salida   : Mejor Solución encontrada (Óptimo local)
3 s = s_0
4 Respetir hasta n evaluaciones
5 |   Generación de vecino donde s' ∈ N(s)
6 |   Si el vecino no es mejor que la solución actual Continuar
7 |   s = s'
  Fin
8 Retornar s
```

Desde la solución inicial s_0, el algoritmo generará una secuencia $s_1, s_2, ..., s_k$ de soluciones con las siguientes características:

- El tamaño de la secuencia k es desconocido a *priori*
- $s_{i+1} \in N(s_i)$, $\forall i \in [0, k-1]$
- $f(s_{i+1}) < f(s_i)$, $\forall i \in [0, k-1]$

A la hora de realizar una implementación específica, hay que elaborar la estrategia de selección del vecino que determinará la próxima solución actual. Es importante pensar en este paso, para hacerlo de forma eficiente.

Recocido Simulado

La estrategia de búsqueda por escalada tiene como inconveniente que una vez que se llega a un óptimo local, el algoritmo no puede continuar mejorando a la solución

actual. En el caso del problema que se aborda, se desarrolló una tesis de licenciatura que aplicaba la técnica denominada *recocido simulado* como parte del programa MECA [50]. El Recocido Simulado (RS) se basa en los principios del proceso de recocido en mecánica, proceso que requiere calentar y luego enfriar lentamente una sustancia para obtener una estructura cristalina fuerte. La resistencia de la estructura depende de la velocidad de enfriamiento de los metales. Si la temperatura inicial no es lo suficientemente alta o se aplica un enfriamiento rápido, se obtienen imperfecciones. En este caso, el material no alcanzará el equilibrio térmico a cada temperatura. Experimentalmente se sabe que los cristales se hacen más fuertes si se hacen a partir de un enfriamiento lento y cuidadoso. El algoritmo RS realiza un mapeo entre conceptos que surgen en el recocido y en optimización, y su aplicación permite realizar optimizaciones que escapan con cierta probabilidad de óptimos locales.

El RS se basa como la búsqueda local por escalada en la utilización de vecindades. Sin embargo, en este caso, la solución actual se puede mover a un vecino que lo empeore. En concreto, la probabilidad de moverse a un vecino que empeore dependerá de la cantidad de empeoramiento que se produce, y de un valor que va cambiando a lo largo del proceso de optimización: la temperatura. El proceso de optimización comienza con una temperatura alta, lo que se traduce en aceptar con mayor probabilidad movimientos de empeoramiento. Conforme la optimización transcurre, la temperatura se reduce gradualmente, pasando a tener un comportamiento más parecido al de la búsqueda por escalada. El bucle más interno del algoritmo de recocido simulado es bastante similar al de búsqueda por escalada en el sentido que se realiza un muestreo de la vecindad. Sin embargo, en lugar de elegir un vecino que mejore al actual, elige uno de forma aleatoria. Si el movimiento mejora la solución , siempre se acepta. De lo contrario, el algoritmo acepta el movimiento con cierta probabilidad. La probabilidad disminuye exponencialmente con base en el nivel de empeoramiento. Además, la probabilidad también decrece a medida que la "temperatura" T disminuye. Se han demostrado ciertas propiedades deseables a nivel de convergencia. Por ejemplo, si T baja lo suficientemente lento, el algoritmo encontrará un óptimo global con una probabilidad que se aproxima a 1 [37]. Existen múltiples formulas para actualizar la T,

entre ellas están:

$$T_{k+1} = \alpha.T_k \tag{3.5}$$

$$T_{k+1} = T_0/log(k+1) \tag{3.6}$$

Donde k refiere a la iteración actual, ec. 3.5 es propuesta en [12], consiste en actualizar T a través de la multiplicación de un factor $\alpha \in (0, 1)$ con el fin de lograr la disminución de la temperatura en la próxima iteración. La tesis de licenciatura antes mencionada utiliza esta función de actualización para la temperatura. Por su parte la ec. 3.6 es otra propuesta de modelo de actualización dado por [21] y donde T_0 es la temperatura inicial en el algoritmo.

En la ecuación 3.8 se describe la fórmula que se usa para determinar la probabilidad de aceptación, mientras que el Algoritmo 2 describe la plantilla del algoritmo del Recocido Simulado.

$$\Delta E = f(s') - f(s) \tag{3.7}$$

$$P(\Delta E, T) = e^{-\frac{\Delta E}{T}} \tag{3.8}$$

3.2.2. Metaheurísticas Poblacionales

Uno de los inconvenientes que tienen las metaheurísticas de trayectoria, es que habitualmente es difícil que la caminata realizada transforme radicalmente a la solución inicial, con lo que se suelen considerar buenos buscadores para localizar buenas soluciones en regiones específicas, pero no tan buenos para buscar en múltiples regiones. La idea subyacente tras las metaheurísticas poblaciones es disponer en cada momento de una población o conjunto de soluciones candidatas, con el fin de mejorar las propiedades como buscadores globales, es decir, mejor la capacidad de exploración. El algoritmo más popular dentro de las metaheurísticas poblaciones es el algoritmo evolutivo, esquema que se usa para ejemplificar este tipo de estrategias.

Algoritmo 2: Algoritmo Recocido Simulado

Entrada: Criterio de Parada, Función que mapea el comportamiento de la Temperatura
Salida : Mejor Solución encontrada
$s = s_0$
$T = T_{max}$
repetir
  repetir
    Generación de vecino s'
    Calcular ΔE con ecuación 3.7
    Si $\Delta E \leq 0$ **Entonces**
      $s = s'$
    Sino *Aceptar s' con una probabilidad P descrita en la ecuación 3.8*
  hasta que *Se cumpla condición de Equilibrio. Ej: Número de ejecuciones para cada valor de Temperatura*
  T es actualizada evaluando la función que la mapea
hasta que *Se cumpla criterio de parada. Ej: $T < T_{min}$*
Retornar s

Los algoritmos evolutivos se basan en realizar un proceso de optimización con algunas componentes inspirada en la evolución de la especies, y la selección natural (supervivencia del más apto). El algoritmo evolutivo inicia creando un conjunto de soluciones candidatas — habitualmente de forma aleatoria — para conformar una población inicial. A continuación se itera para crear nuevos conjuntos de soluciones candidatas, paso que recibe el nombre de generación. En cada iteración, cada individuo es evaluado y recibe un valor de aptitud, que habitualmente es directamente el dado por la función objetivo, aunque en algunos casos se consideran otros aspectos. Sobre la base de estos valores de aptitud, algunos de los mejores candidatos son elegidos para actuar como base para generar nuevas soluciones candidatas. Esto se hace aplicando operadores de recombinación y/o mutación a los mismos. La recombinación es un operador que se aplica a dos o más candidatos seleccionados (los llamados padres), produciendo uno o más candidatos nuevos (los hijos). La mutación se aplica a los hijos generados y da como resultado nuevos candidatos. Por lo tanto, ejecutar las operaciones de recombinación y mutación en los padres conduce a la creación de un conjunto de nuevos candidatos, que recibe el nombre de descendencia. Finalmente, se

realiza un proceso de selección de reemplazamiento para decidir qué individuos pasa a tener un lugar en la generación futura [15].

Hay muchas teorías y formas de visualizar los roles de los diferentes componentes, siendo habitual considerar lo siguiente:

- Los operadores de variación (recombinación y mutación) crean la diversidad necesaria dentro de la población y facilitan que hayan soluciones candidatas novedosas. Al respecto, hay que hacer notar que los operadores de mutación pueden ser visualizados también como esquemas de intensificación, ya que habitualmente hacen cambios muy pequeños.

- La selección actúa como una fuerza que aumenta la calidad media de las soluciones en la población.

El algoritmo 3 esquematiza el funcionamiento de los algoritmos evolutivos.

Algoritmo 3: Esquema de un algoritmo evolutivo

```
1 Respetir hasta Criterio de parada sea satisfecho
2 |  Selección de padres
3 |  Recombinación de pares de padres
4 |  Mutación de la descendencia
5 |  Selección de individuos para la siguiente generación
  Fin
```

Algunas características importantes de los algoritmos evolutivos son las siguientes:

- Los algoritmos evolutivos se basan en la población, es decir, procesan una colección completa de soluciones candidatas simultáneamente.

- La mayoría de los algoritmos evolutivos utilizan la recombinación, mezclando información de dos o más soluciones candidatas para crear una nueva.

- Las algoritmos evolutivos son estocásticos.

Se han desarrollado múltiples extensiones del algoritmo evolutivo básico anteriormente descrito. Una de las extensiones más habituales consiste en desarrollar algoritmos híbridos, que integren en un evolutivo, una búsqueda local o un algoritmo de

trayectoria. A esta integración se le conoce como Algoritmo Memético de primera generación.

Algoritmo Memético de Primera Generación

La categoría de algoritmos híbridos es muy exitosa en la práctica y forman un área de investigación en rápido crecimiento que se exploró casi desde la aparición de los primeros evolutivos. Hay una serie de factores que motivan la hibridación de algoritmos evolutivos con otras técnicas. Entre estos factores se encuentra que muchos problemas complejos se pueden descomponer en varias partes, para algunos de los cuales es posible que ya se encuentren disponibles métodos exactos o muy buenas heurísticas. En estos casos, tiene sentido utilizar una combinación de los métodos más apropiados para diferentes subproblemas.

Existe una visión generalizada de que, si bien los Algoritmos Evolutivos son muy buenos para identificar rápidamente buenas áreas del espacio de búsqueda (exploración), son menos buenos en la etapa de explotación. Por ello, casi desde los primeros años, se desarrollaron esquemas que incluían búsqueda sistemáticas de la vecindades de las soluciones generadas por el evolutivo, pudiendo integrarse por ejemplo, con un método de búsqueda por escalada. A los métodos que integran un evolutivo con métodos de trayectoria simples, se les llaman algoritmos meméticos de primera generación.

El primer uso del término Algoritmo Memético en el ámbito de metaheurísticas fue llevado a cabo por Moscato [31]. En este primer trabajo, se presentó una metaheurística que combinaba Recocido Simulado (Algoritmo 2) local junto con un juego competitivo y cooperativo entre agentes que incorporaba el uso de un operador de cruza. El termino memético viene de meme [10] de Dawkins, siendo el meme un patrón de información contagioso que se replica por parásitos.

El contenido del meme es análogo a los cromosomas de los genes. Dawkins, quien utilizó la palabra en su libro The Selfish Gene, define al meme como una simple unidad de información intelectual o cultural que sobrevive el tiempo suficiente para ser reconocida y poder ser traspasada. Algunos ejemplos de memes son canciones,

Algoritmo 4: Algoritmo Memético de Primera Generación

```
1  Entrada: Criterio de parada: Generaciones
2  Salida  : Mejor individuo de la población actual
3  Inicializar población
4  Evaluar cada solución
5  repetir
6  |  Selección de Padres
7  |  Cruza para producir la descendencia
8  |  Mutación
9  |  Mejorar descendencia a través de Búsqueda Local
10 |  Evaluar descendencia
11 |  Seleccionar individuos para la siguiente generación
   |  (Reemplazamiento)
12 |  GeneraciónActual++
   hasta que GeneraciónActual <Generaciones
13 Retornar Mejor individuo de la población actual
```

ideas, frases o hasta forma de construir cosas. Así como los genes se propagan a sí mismos en el acervo genético a través de espermatozoides u óvulos, los memes se propagan saltando de cerebro a cerebro a través de un proceso que, en sentido amplio, puede llamarse imitación.

El algoritmo 4 describe una plantilla muy básica de un Algoritmo Memético. Nótese que en el mismo no hay transmisión de memes, sino que el único cambio es la incorporación de un mecanismo de mejora individual. Por ello, se le suelen llamar algoritmos meméticos de primera generación, frente a los algoritmos más recientes que sí que incluyen transmisión de memes.

3.3. Optimización Multi-objetivo

Hay muchas aplicaciones en la vida real que involucran la optimización de múltiples objetivos. En lo referente a esta investigación se pretende diseñar estructuras y cada posible solución puede ser evaluada con base en múltiples características que describen que tan adecuada es. Por ello, el problema de diseño puede ser visualizado como un proceso de optimización multi-objetivo.

Un problema de optimización multi-objetivo con restricciones (MOPA) se define de la siguiente forma:

$$\begin{aligned} \text{Minimizar } \mathbf{f}(\mathbf{x}) &:= (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})) && (3.9)\\ \text{Sujeto a } g_i(\mathbf{x}) &\geq 0 \ \forall i \in \{1, 2, \ldots, o\} && (3.10)\\ h_j(\mathbf{x}) &= 0 \ \forall j \in \{1, 2, \ldots, r\} && (3.11)\\ x_k \in [x_k^l, x_k^u] &\quad \forall k \in \{1, 2, \ldots, n\} && (3.12) \end{aligned}$$

Donde $X \subset R^n$ es el espacio de búsqueda y $Z \subset R^m$ es el espacio objetivo. Cada vector de decisión $x \in X$ se relaciona con un vector objetivo $\mathbf{f}$, donde X está restringida con restricciones de desigualdad (3.10), restricciones de igualdad (3.11) y variables con dominio limitado (3.12).

3.3.1. Conceptos Básicos

Dominancia

Una de las dificultades de optimización multi-objetivo es que comparar la calidad de dos soluciones $\mathbf{x}, \mathbf{y} \in X$ es más complicado que en optimización mono-objetivo. Para realizar la comparativa habitualmente se usa la *Relación de dominancia de Pareto*:

> Una solución $\mathbf{x}$ se dice que *domina* a una solución $\mathbf{y}$, denotado por $\mathbf{x} \prec \mathbf{y}$ o $\mathbf{f(x)} \prec \mathbf{f(y)}$, si y solo si $\mathbf{x}$ es al menos tan bueno como $\mathbf{y}$ en todos los objetivos ($\forall i \in \{1, \ldots, m\}\ f_i(x) \leq f_i(y)$) y mejor en al menos un objetivo ($\exists j \in \{1, \ldots, m\},\ f_j(x) < f_j(y)$).

Esta relación binaria induce a un *orden parcial estricto* sobre X, donde pueden ocurrir 3 escenarios: $\mathbf{x} \prec \mathbf{y}$, $\mathbf{y} \prec \mathbf{x}$ o ninguno de los dos (Ver figura 3-2).

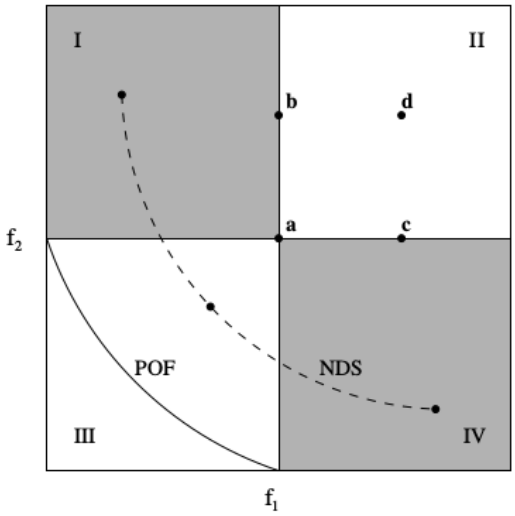

Figura 3-2: Ejemplos de relaciones de dominio. Sostiene que a$\preceq$a, a$\preceq$b, a$\preceq$c, a$\preceq$d, a$\prec$b, a$\prec$c y a$\prec$d. Además las soluciones que están dentro de los cuadrantes I y IV son incomparables la una con la otra [23].

Dado el conjunto A, el conjunto de soluciones no dominadas, son aquellas que no son dominadas por cualquier otro miembro del conjunto X, siendo $A \subseteq X$, es conocido como el conjunto No-Dominado (NDS por sus siglas en inglés), las soluciones en este conjunto son incomparables y está dado por:

$$\text{NDS(A)} := \{\mathbf{a} \in A : \nexists a' \in A, a' \prec a\} \tag{3.13}$$

La solución óptima a un problema multi-objetivo consiste en el conjunto de vectores de decisión en X que son no-dominados y que no se puede mejorar en ningún objetivo sin empeorar al menos otro objetivo. Este conjunto se conoce como el conjunto óptimo de Pareto (POS) y se define formalmente como:

$$\text{POS} := \{\text{x}^* \in \mathcal{X} : \nexists \text{x} \in \mathcal{X} \subset \mathbb{R}^n, \text{x} \prec \text{x}^*\} \tag{3.14}$$

Y a las imágenes de este conjunto se le conoce como frente de Pareto (POF).

$$\text{POF} := \{\mathbf{f(x^*)} \in Z \subset R^m : \mathbf{x^*} \in POS\} \tag{3.15}$$

Otra forma común de comparar soluciones es usando la relación de dominancia débil de Pareto:

> Una solución $\mathbf{x}$ se dice que *domina débilmente* a una solución $\mathbf{y}$, denotado por $\mathbf{x} \preceq \mathbf{y}$ o $\mathbf{f(x)} \preceq \mathbf{f(y)}$, si y solo si $\mathbf{x}$ es al menos tan bueno como $\mathbf{y}$ en todos los objetivos $(\forall i \in \{1, \ldots, m\}\ f_i(x) \leq f_i(y))$.

Métricas

Un tema importante en optimización multi-objetivo es la comparación cuantitativa del rendimiento de diferentes algoritmos. En el caso de los MOEAs, el resultado de los mismos suele ser una aproximación del frente de Pareto al que llamaremos *conjunto aproximado*, y por lo tanto, surge la pregunta de cómo evaluar la calidad de dichos conjuntos [54]. Las métricas o indicadores de rendimiento evalúan la calidad de dicha aproximación.

> Un indicador de rendimiento k-ario I es una función $I : Z^k \rightarrow R$, el cual asigna a cada vector $(A_1, A_2, \ldots, A_k)$ de k conjuntos de aproximaciones un valor real $I(A_1, A_2, \ldots, A_k)$.

Los *indicadores de rendimiento unarios* (cuando k=1) son de los más utilizados, y asignan a cada aproximación un valor real. Las comparaciones entre los conjuntos de aproximación $A, B \subset Z$ son posibles utilizando la relación de dominancia débil de Pareto. Se dice que un indicador de rendimiento unario I es débilmente compatible con Pareto, si cumple:

$$\forall A, B \subset Z : A \preceq B \Rightarrow I(A) \geq I(B) \tag{3.16}$$

donde A domina débilmente a B. Y es compatible con Pareto, si satisface:

$$\forall A, B \subset Z : (A \preceq B \wedge A \neq B) \Rightarrow I(A) > I(B) \tag{3.17}$$

También se usan mucho los indicadores binarios. Reciben dos conjuntos, el valor devuelto expresa una relación sobre la calidad de ambos conjuntos, por ejemplo, la aportación del primero en relación al segundo con respecto a cierta característica. Es habitual que cuando se usan indicadores binarios, uno de los conjuntos sea de muy buena calidad como referencia, para así poder cuantificar las diferencias entre un conjunto de muy alta calidad y la aproximación encontrada. Se enlistan brevemente algunos de los indicadores de desempeño más importantes reportados en la literatura. A menos que se indique lo contrario, se usarán los símbolos A, B para los conjuntos de aproximaciones a evaluar y R para el conjunto de referencia.

1) Hipervolumen

   El único indicador de rendimiento unario que se sabe es compatible con la relación de Pareto es el hipervolumen [55], también conocido como la métrica S. Este indicador determina el tamaño de la porción del espacio objetivo que está dominada por las soluciones de un conjunto A, colectivamente y delimitada por un punto de referencia $z \in R^m$:

$$HV(A; \mathbf{z}) = \Lambda \left(\bigcup_{\mathbf{a} \in A} \{\mathbf{x} | \mathbf{a} < \mathbf{x} < \mathbf{z}\} \right) \tag{3.18}$$

   donde Λ denota la *medida de Lebesgue*[3] y $\mathbf{z}$ debe estar dominados por todos los elementos en A. Se prefieren los valores más altos de este indicador. y en general tiene cierta tendencia a favorecer a aquellas aproximaciones que tienen más puntos en la región de la rodilla. Una muy buena propiedad matemática del hipervolumen es que su maximización es equivalente a alcanzar el conjunto óptimo de Pareto [19]. Por estas razones, el hipervolumen es uno de los indicadores de rendimiento

[3]Es la forma estándar de asignar una longitud, área o volumen a los subconjuntos de un espacio euclídeo

más utilizados para comparar esquemas multi-objetivo. El principal inconveniente de utilizar el hipervolumen es su alto costo computacional, pues este costo crece exponencialmente con el número de objetivos [5], aunque eso no es un inconveniente en esta tesis, pues se realizan optimización solo con dos objetivos (Ver imagen 3-3).

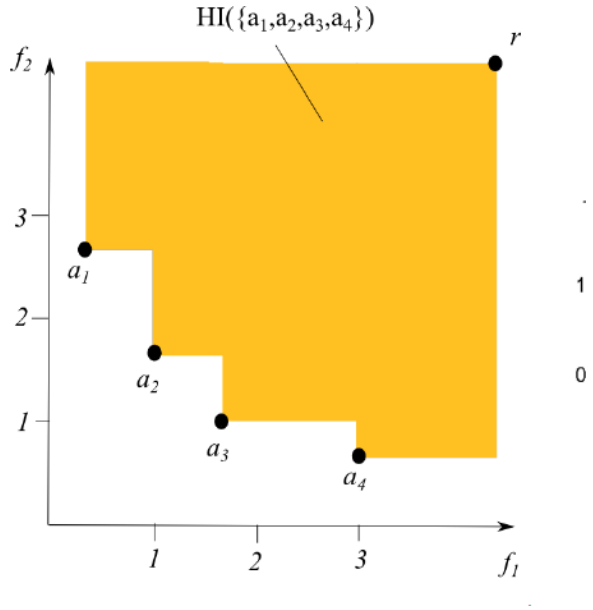

Figura 3-3: Indicador de hipervolumen para un problema de 2 objetivos de un frente integrado por a_1, a_2, a_3, a_4 con un punto de referencia r .

2) Distancia generacional invertida (IGD)

La distancia generacional invertida es una métrica binaria que usa un conjunto de referencia. IGD es la distancia promedio de cada punto de referencia a su solución más cercana. Cuando se utiliza un conjunto de puntos de referencia bien distribuidos en todo el frente de Pareto, un valor pequeño del indicador IGD sugiere una buena convergencia de las soluciones al frente de Pareto así como una buena distribución en todo el frente de Pareto. El IGD se define de la siguiente forma [25]:

$$IGD(A) = \frac{1}{|Z|}\left(\sum_{j=1}^{|Z|} \hat{d}_j^p\right)^{1/p} \tag{3.19}$$

donde $\hat{d}_j$ es la distancia euclidiana desde z_j (referencia) hasta su vector objetivo

más cercano en A. El término "*distancia generacional invertida* (IGD)" se utilizó por primera vez en 2004 [8, 43]. Este indicador de desempeño mide tanto la convergencia como la diversidad. Desafortunadamente, no es compatible con Pareto.

3) Distancia generacional invertida (IGD+)

La distancia *generacional invertida modificada* (IGD+) propuesta por Ishibuchi en [25] es una métrica débilmente compatible con Pareto. IGD+ puede ser condensado como: Si una solución es dominada por un punto de referencia, entonces se utiliza la distancia Euclídeana sin ninguna modificación. Sin embargo, si la solución y el punto de referencia son no dominados entre sí, entonces se calcula la mínima distancia desde el punto de referencia a la región dominada por la solución. La forma del IGD+ es similar al IGD propuesto previamente, a diferencia que la distancia Euclídea es modificada, de la siguiente manera:

Cálculo de distancia modificada para problemas de minimización

$$d^{+}(z,a) = \sqrt{d_1^{+^2} + \cdots + d_m^{+^2}} = \sqrt{\left(\text{máx}\left\{a_1 - z_i, 0\right\}\right)^2 + \cdots + \left(\text{máx}\left\{a_m - z_m, 0\right\}\right)^2} \tag{3.20}$$

Cálculo de distancia modificada para problemas de maximización

$$d^{+}(z,a) = \sqrt{d_1^{+^2} + \cdots + d_m^{+^2}} = \sqrt{\left(\text{máx}\left\{z_1 - a_1, 0\right\}\right)^2 + \cdots + \left(\text{máx}\left\{z_m - a_m, 0\right\}\right)^2} \tag{3.21}$$

4) Superficies de cubrimiento

Las formas descritas hasta ahora permiten comparar numéricamente diferentes métodos. Sin embargo, podría ser útil también disponer de ciertas gráficas que nos muestren aspectos relevantes de las aproximaciones encontradas. Una posibilidad es hacer uso de las superficies de cubrimiento. Dado un conjunto de n ejecuciones, una superficie de cubrimiento al x %, es una superficie que sirve para separar la región del espacio objetivo que fue dominada en al menos el x % de las

ejecuciones, de aquellas regiones que no fueron dominadas en dicho porcentaje de ejecuciones [26].

Nótese que una alternativa es simplemente mostrar las aproximaciones encontradas por las diferentes ejecuciones. A modo de ejemplo, la fig. 3-4 muestra como se visualizaría el resultado de 3 ejecuciones. Si se realizan muchas ejecuciones, estas gráficas quedarían muy saturadas y obtener información sobre la distribución general sería muy complicado, razón por la que el uso de superficies de cubrimiento es mucho más común.

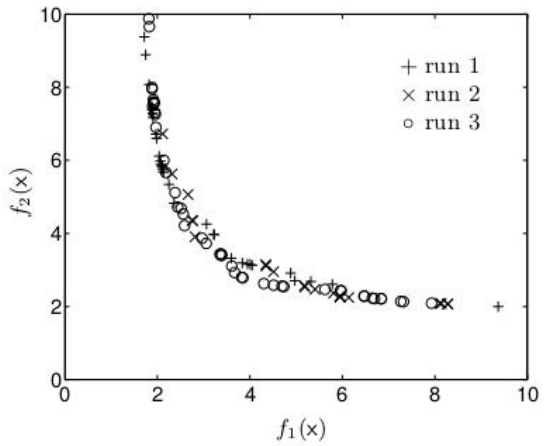

Figura 3-4: Tres diferentes ejecuciones de un algoritmo.

Observando la figura 3-5, se puede evidenciar claramente 3 zonas. La primera, localizada en la parte inferior hacia la izquierda, es la zona que nunca fue alcanzada por los vectores objetivo. La segunda, localizada en la parte superior hacia la derecha de las superficies de cubrimiento está compuesta por los vectores objetivo que se alcanzaron en todas las ejecuciones. Y finalmente, la tercera y última zona es la ubicada entre los limites de las dos zonas anteriormente descritas, y está compuesta por las regiones alcanzadas en ciertas ejecuciones y en otras no.

Esta forma de ver la superposición de múltiples superficies de cubrimiento proporciona una base sobre la cual se puede construir una noción cuantitativa de rendimiento "típico". Ahora se puede buscar identificar la familia de vectores objetivos que es probable que se alcancen, cada uno por su cuenta, en exactamente

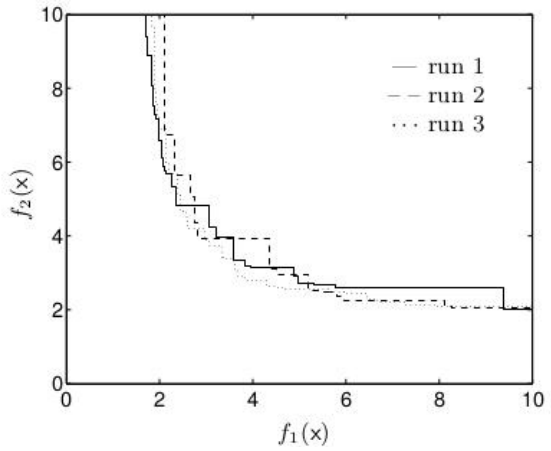

Figura 3-5: Superposición de superficies de cubrimiento

el x % de las corridas (lo que se llamará la superficies de cubrimiento del x % del optimizador).

Las estimaciones para las superficies de logro del 25 % y del 75 % podrían producirse exactamente de la misma manera estimando los cuartiles inferior y superior en lugar de la mediana.

3.3.2. Algoritmos Evolutivos Multi-objetivo

Los Algoritmos Evolutivos (EA) diseñados para problemas de optimización multi-objetivo se denominan Algoritmos Evolutivos Multi-objetivo (MOEA). La característica más distintiva de los EAs es que un grupo de individuos evolucionan juntos para buscar buenas soluciones a un problema. El propósito de los optimizadores multi-objetivo es obtener buenas aproximaciones del frente de Pareto. Por ello, si se puede modificar el algoritmo evolutivo para que la población no converja en un punto sino que intenten residir en diferentes puntos cercanos al frente de Pareto, los EA podrían proporcionar a los diseñadores soluciones no dominadas en una sola ejecución. Esta es precisamente el principio que guía el diseño de la mayor parte de MOEAs.

La principal diferencia entre los algoritmos evolutivos mono-objetivo y multi-objetivo (MOEA) se encuentra en la técnica de asignación del valor de aptitud

y en el proceso de selección de supervivientes En algoritmos evolutivos multi-objetivo el diseño de estas componentes debe conducir al algoritmo hacia el frente óptimo de Pareto, mientras mantiene una buena distribución de las soluciones. Este propósito se puede intentar conseguir de múltiples formas, por lo que han surgido múltiples tipos de MOEAs. La clasificación propuesta en [33] identifica a los algoritmos basados en dominancia, los basados en indicadores y los basados en descomposición. En esta tesis, sólo se usan los basados en dominancia. Sin embargo, a lo largo del texto se usan varios de los conceptos que se usan en los otros tipos, así que a continuación se describen los tres tipos de métodos.

Algoritmos basados en dominancia

Este grupo de algoritmos utiliza la relación de dominancia en diferentes componentes del MOEA, principalmente en el proceso de asignación de aptitud, valor que influye habitualmente sobre el proceso de selección de padres y sobre el proceso de supervivencia. Nótese que la relación de dominancia por sí misma no implica mantenimiento de diversidad, así podríamos encontrar una situación en las que todas las soluciones de la población son no dominadas, pero cubren un espacio muy pequeño del frente. Para tratar de no solo converger al frente, sino también hacerlo de una forma diversa se incluyen técnicas adicionales como estimadores de densidad del espacio objetivo. A continuación se describen de forma breve los dos MOEAs que se han utilizado en esta tesis:

- **Non-dominated Sorting Genetic Algorithm II: NSGA-II [11]**

  El NSGA II es un método basado en dominancia al que se le achacan 3 características principales: incluye un procedimiento para ordenar a las soluciones no-dominadas en frentes de dominancias, usa un procedimiento de estimación de densidad en el espacio objetivo y aplica un operador de comparación basado tanto en calidad como en densidad. Se detallan los operadores que conforman al algoritmo a continuación:

a) Selección de Padres: Utiliza un torneo binario basado en un operador que integra la ordenación las soluciones por el algoritmo de Non-dominated sort y la distancia de crowding calculada para la población actual. Este operador se denota como comparador de crowding $\prec_n$.

b) Utiliza operadores de cruza y mutación para generar la descendencia. El algoritmo es genérico en ese aspecto, pero la versión estándar para optimización continua usa el operador de cruce SBX y el operador de mutación polinomial.

c) Reemplazamiento: Utiliza un esquema elitista basándose en $\prec_n$, es decir, selecciona a los N mejores individuos con base en dicho operador.

- **A Dominance-Based Multi-Objective Evolutionary Algorithm with Explicit Variable Space Diversity Management: VSD-MOEA [6]**

  Recientemente, se diseño el Algoritmo VSD-MOEA, que también entra en la clase de algoritmos basados en dominancia. Hay dos contribuciones principales en este algoritmo: primero, se propone un nuevo estimador de densidad en el espacio objetivo, segundo, se diseña una nueva fase de reemplazo que toma en cuenta la diversidad en el espacio de las variables. La fase de reemplazo no solo considera la diversidad en el espacio de las variables sino que también toma en cuenta el nuevo estimador de densidad del espacio objetivo. El principio fundamental tras el diseño del nuevo operador de reemplazamiento es utilizar el criterio de parada y las generaciones transcurridas con el objetivo de avanzar paulatinamente desde fases de exploración a fases de intensificación. Se enlista a continuación los operadores utilizados por el esquema VSD-MOEA:

  a) Selección de Padres: Utiliza un torneo binario que se basa en la ordenación por dominancia de las soluciones, usándose selección aleatoria en caso de empate.

  b) Utiliza operadores de cruza y mutación estándar para generar la descendencia. En particular, no imponen ninguna restricción adicional sobre

esta fase.

c) Reemplazamiento: este operador incluye penalizaciones por cercanía en el espacio de las variables, así como un estimador de densidad en el espacio de los objetivos, procurando así diversidad en ambos espacios. Desarrollaremos este punto en el capítulo 5.

Algoritmos basados en indicadores

El principio que utilizan los algoritmos evolutivos basados en indicadores es sustituir (o a veces complementar) el uso de la relación de dominancia, por la aplicación de indicadores. Se puede usar en diferentes fases, siendo habitual aplicarlo en la asignación de aptitud y selección de supervivencia. Dado que muchos indicadores contemplan tanto la parte relativa a convergencia como la parte relativa a la diversidad, la principal ventaja es que no es necesario incorporar métodos adicionales para preservar la diversidad en el espacio objetivo. Además, los indicadores se pueden diseñar para expresar las preferencias del tomador de decisiones. Los indicadores más utilizados incluyen el indicador $I_{\epsilon+}$ y el hipervolumen. A continuación se describen de forma breve dos de los métodos basados en indicadores más populares:

- **Indicator-Based Evolutionary Algorithm: IBEA [56]**

  La idea principal de este algoritmo es definir el objetivo de la optimización en términos de un indicador binario y luego usar directamente esta medida en el proceso de selección. Dado que el cálculo de los indicadores es un proceso costoso, se propone una simplificación para calcular la contribución a la métrica por parte de cada individuo con base en contribuciones parciales por pares. El algoritmo no se asoció a una métrica en particular, sino que puede ser usado con diferentes métricas. Algunas de las características principales son las siguientes:

  a) Selección de Padres: Utiliza torneo binario. El valor de fitness es dado por una función exponencial, calculada con base en la contribución de

cada individuo a la función indicadora.

b) Utiliza operadores de cruza y mutación para generar la descendencia.

c) Reemplazamiento: Utiliza un esquema elitista $\mu+\lambda$ el cual iterativamente va eliminando el peor individuo con base en su contribución a la función indicadora, actualizando el valor de fitness de los individuos cada vez que se elimina uno.

- **Multiobjective Selection Based on Dominated Hypervolum: SMS-EMOA [2]**

  Este es un algoritmo *steady-state* que se fundamenta en dos pilares: en el la ordenación de no-dominados que se utiliza como criterio inicial de clasificación y en el hipervolumen que se aplica como criterio de selección para discernir entre individuos pertenecientes al peor/último frente y que deben ser eliminados. Algunas de sus características principales:

  a) Selección de Padres: se basa en selección aleatoria.

  b) Utiliza operadores de cruza y mutación para generar 1 individuo que se convertirá en miembro de la siguiente generación.

  c) Reemplazamiento: Esquema elitista. Se elimina la solución con la peor contribución de hipervolumen del último frente no dominado.

Algoritmos basados en descomposición

Los algoritmos evolutivos multi-objetivo basados en descomposición se hicieron populares con la propuesta hecha por Zhang y Li (**MOEA/D**) [52]. Estos algoritmos tienen enfoques de agregación en los que un MOP se descompone en una serie de problemas de optimización mono-objetivo. La meta para cada uno de estos problemas de optimización, también llamados sub-problemas, es tener una agregación ponderada (lineal o no lineal) de los objetivos individuales. Las relaciones de vecindad entre estos sub-problemas se definen en función de las distancias entre sus vectores de ponderación de agregación. El sub-problema i es vecino del sub-problema j si el vector de peso i está cerca del vector de peso j. Cada sub-problema

se optimiza en el MOEA/D mediante el uso de información principalmente de sus sub-problemas vecinos.

En una versión simple de MOEA/D, cada sub-problema individual mantiene una solución en su memoria, que podría ser la mejor solución encontrada hasta ahora. Para cada cada uno de estos, el algoritmo genera una nueva solución al aplicar operadores genéticos y actualizará su memoria si la nueva solución es mejor que la anterior. Un sub-problema también pasa su solución recién generada a algunos (o todos) sus vecinos, que actualizarán sus soluciones actuales si la recibida es mejor. Una ventaja importante de MOEA/D es que se puede utilizar una búsqueda local a través de la propia *función de escalarización* que se usó para establecer el sub-problema [53].

Las funciones de escalarización transforman al MOP original (ver ec. 3.9) en un problema mono-objetivo usando un conjunto de pesos $\mathbf{w} \in R^m$. Al transformarlo el nuevo problema queda como [23]:

$$\begin{aligned} &\text{Minimizar } u\left(\mathbf{f}(\mathbf{x}); \mathbf{w}\right) \\ &\text{sujeto a } \mathbf{x} \in \mathcal{X} \end{aligned} \tag{3.22}$$

Cada componente de $\mathbf{w}$ debe satisfacer $w_i > 0$, y aunque no hay ninguna razón particular más allá de lograr la uniformidad entre las soluciones, también asumimos que $\sum_i w_i = 1$. Si resolvemos múltiples problemas de ese tipo con diferentes pesos, es posible que podamos conseguir soluciones que estén distribuidas a lo largo del frente de Pareto.

Se presentan algunas de las funciones de escalarización (definición y características correspondientes) en la Tabla 3.1.

Una mérito de los algoritmos basados en funciones de escalarización es que pueden hacer uso de cualquier algoritmo conocido para resolver los problemas de manera efectiva, y hacer uso de su eficacia en el contexto de objetivos múltiples [47].

Acrónimo	Nombre Completo	Minimizar $u(y; \mathbf{w}) :=$	Soporte	Parámetro del modelo
WS	Weighted Sum	$\sum_i w_i y_i$	x $\prec$	-
CHE	Chebyshev Function	máx$_i\{w_i \lvert y_i \rvert\}$	x,c,l $\preceq$	-
ASF	Achievement Scalarizing Function	máx$\{\frac{y_i}{w_i}\}$	x,c,l $\preceq$ $\lvert\lvert$	-

Tabla 3.1: Algunas funciones de escalarización y sus características. Formas del frente de pareto son abreviadas en x (convexo) c (cóncavo) o l (lineal). $\prec$ ($\preceq$) denota compatibilidad con (débil) óptimalidad. de Pareto. $\lvert\lvert$ significa que el vector de objetivos óptimos $\mathbf{y}*$ es casi paralelo al vector de peso $\mathbf{w}$ [23].

3.3.3. Búsquedas Locales en Optimización Multi-objetivo

Al igual que en el caso de optimización mono-objetivo, en optimización multi-objetivo se han desarrollado EAs híbridos que integran un paso de optimización local. En el caso de utilizar MOEAs basados en descomposición es muy sencillo integrar búsquedas locales, pues se dispone de funciones mono-objetivo a optimizar a través de las funciones de escalarización, como se mencionó anteriormente. En los otros tipos, no se incluyen dichas funciones de escalarización, por lo que integrar una búsqueda local no es tan directo. Sin embargo, una posibilidad es tener una etapa separada en la que se generen un conjunto de funciones de escalarización y se optimicen con base en los individuos presentes en la población tal y como se muestra en la figura 3-6

Otra forma para adaptar el algoritmo básico de búsqueda local al caso multi-objetivo consiste en tener en cuenta la relación de dominancia de pareto. El proceso de hibridación consiste en generar un vecindario para cada solución del frente de pareto y a continuación las nuevas soluciones no-dominadas (vecinos) son insertadas en la población, mientras que las soluciones que se dominaron son eliminadas [47].

En esta tesis, se pudo detectar que para el caso multi-objetivo era muy importante integrar búsquedas locales y se decidió utilizar el primer tipo de estrategias discutidas, es decir, generar un conjunto de funciones de escalarización y tratar de optimizarlas a partir de los individuos de la población.

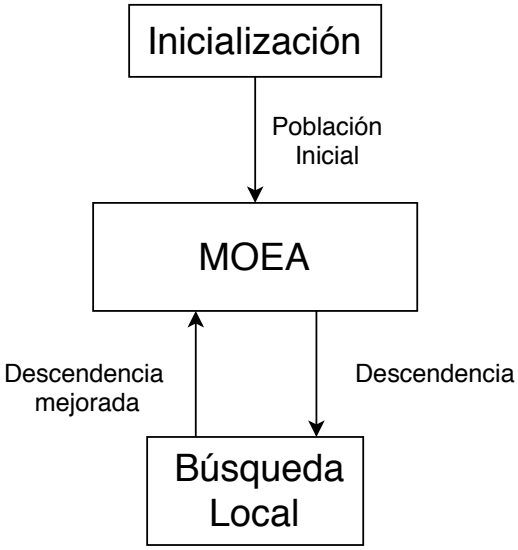

Figura 3-6: Forma genérica de los algoritmos genéticos que integran búsqueda local en la optimización multi-objetivo.

Capítulo 4

Diseño de Algoritmo Memético para Optimización Mono-objetivo

4.1. Motivación

Hasta la fecha, a pesar de que se han trabajado variantes de problemas cercanos al utilizado en esta tesis, no existe un *benchmark* establecido que permita realizar comparativas justas entre diferentes algoritmos. Es por ello que a partir de la revisión del estado del arte no se pudo establecer un algoritmo heurístico como el método a superar y frente al cual comparar los desarrollos realizados. Dado que en general los Algoritmos Meméticos (Ver sección 3.2.2) han ofrecido resultados de muy alta calidad en optimización global y más específicamente en problemas de naturaleza combinatoria, se decidió desarrollar algoritmos meméticos para la resolución del problema tratado en esta tesis. En las siguientes secciones se especifican cada una de las componentes que hubo que seleccionar y/o diseñar para disponer de dicho algoritmo memético.

4.2. Representación del individuo

Palmer en [32] habla sobre los puntos claves para una buena representación de la solución y enlista los siguientes:

- Una codificación debe ser capaz de representar todos los fenotipos posibles.
- Una codificación debe ser carente de sesgos en el sentido de que todos los individuos deben estar igualmente representados en el conjunto de todos los genotipos posibles.
- Una codificación no debería codificar soluciones infactibles.
- La decodificación debería ser fácil.
- Una codificación debe poseer localidad (o sea, cambios pequeños en el genotipo deberían producir cambios pequeños en el fenotipo)

Para el problema de estructura de barras se eligió trabajar al individuo como un vector de números enteros los cuales están en el rango del tamaño del catálogo dado, el vector será de dimensión d (cantidad de barras que conforman la estructura). A continuación se muestra un ejemplo de individuo **s** para una estructura de 8 barras, cuyo catálogo tiene 59 secciones.

s	*59*	*2*	*6*	*1*	*27*	*31*	*42*	*16*

4.3. Función de Fitness

Como se ha comentado, la variante mono-objetivo tratada en esta tesis, se puede formular como un problema de minimización con restricciones. Uno de los primeros pasos a la hora de tratar un problema de optimización con restricciones consiste en diseñar la función de fitness. En la sección 3.1.4 se discutieron algunas de las formas que se han ideado para realizar el manejo de restricciones en algoritmos evolutivos. En esta tesis, se utilizó una *función de penalización estática*, por lo que tras realizar la conversión se puede aplicar cualquier método para optimización mono-objetivo sin restricciones.

En concreto, y dado que el objetivo principal a minimizar es el *peso propio* de la estructura de barras, la función de fitness es igual a dicho peso más una penalización

que se aplica a aquellas estructuras que no cumplan las restricciones de *desplazamiento* y *eficiencia* que ya se discutieron al describir el problema. Dichos valores cuentan con un rango de factibilidad igual a $[1, 10]$ para el caso del desplazamiento y a $[0, 1]$ para el caso de la eficiencia. Si en una solución específica, los valores de eficiencia o factibilidad se salen de sus rangos correspondientes, se calcula la distancia a los valores límites de factibilidad y dicha distancia se multiplica por un valor muy grande para calcular la penalización (Ver algoritmo 5), obteniéndose así la aptitud de la solución evaluada por la función.

Algoritmo 5: Función Fitness

1 **Entrada**: *Individuo o solución*
2 **Salida** : *Valor de Fitness*
3 Evaluación de la solución a través de MECA [4] $\rightarrow$ arroja valores de **peso propio**, **desplazamiento** y **eficiencia**.
4 dd = desplazamiento - 10.0 (Desplazamientos mayores a 10cm será penalizado)
5 de = eficiencia - 1.0 (Eficiencias mayores a 1 será penalizado)
6 peso = Peso propio de la estructura.
7 penaliza=$1x10^{12}$
8 if (de >0) penalización1=de $*$ penaliza
9 if (dd >0) penalización2=dd $*$ penaliza
10 Fitness = peso + penalización1 + penalización2
11 **Retornar** *Valor de Fitness*

4.4. Operador de Cruce

Una parte importante del poder exploratorio de un algoritmo evolutivo se atribuye al operador de recombinación y a su capacidad para generar nuevas buenas soluciones a partir de otras soluciones que ya tienen rendimientos prometedores [16]. En muchos casos, se utilizan los algoritmos evolutivos sin realizar adaptaciones específicas de los operadores de cruce al problema en cuestión que se está tratando. Entre los operadores más habituales se encuentra el operador de cruce uniforme, el cual dado dos padres, genera nuevos hijos tomando para cada variable el valor de uno de los dos padres con igual probabilidad. Este operador puede llegar a ser muy disruptivo pues dependiendo

de los dependencias existentes entre las variables puede resultar en la ruptura de bloques constructivos útiles.

Con el fin de desarrollar un operador de cruce adaptado al problema de optimización de estructuras, nos inspiramos en varios esquemas que han tenido mucho éxito en el ámbito de las telecomunicaciones. En el caso del Problema de Posicionamiento de Antenas (APP) se desarrolló un operador que hereda de forma conjunta, es decir, a partir del mismo padre, estaciones base (BS) que se encuentran en un radio específico con el propósito de diseñar un operador menos destructivo que permita pasar regiones completas de padres a hijos [40]. Algo similar ocurre en el Problema de Asignación de Frecuencia (FAP) donde uno de los operadores de recombinación diseñados se basa en realizar una descomposición del área de servicio (terreno) en dos mitades a lo largo de una de las dimensiones (vertical u horizontal). Para cada mitad, se evalúa la función objetivo de los individuos padres, y la descendencia hereda la configuración para cada una de las sub-áreas donde el padre fue más adecuado [48]. Los operadores de cruce adaptados al contexto del problema generan nuevos individuos que reciben características significativas de ambos padres, pero primero se debe analizar para el problema en cuestión cuáles son esas características que se deben heredar.

Para el *Problema de Optimización de Estructura de Barras*, las soluciones candidatas se representan con un vector con tantas posiciones como barras haya en la estructura. Cada posición del vector está asociado a una barra y el valor que se almacena en cada posición indica el número de sección a utilizar del catálogo que se esté considerando en el proceso de optimización. Se busca entonces el diseño de una recombinación u operador de cruce que pueda crear individuos con herencia de características parentales relevantes. La naturaleza del problema es sobre una estructura constituida por barras que están conectadas entre sí, lo que puede representarse a tráves de un grafo, donde las barras son los vértices y las aristas representan conexiones entre barras. Dado que en problemas de telecomunicaciones se ha visto que se obtiene mejor rendimiento al heredar de forma conjunta los valores asociados a antenas que están cercanas, para este problema se hipotetizó que tendría sentido heredar de forma conjunta barras que estén conectadas en el grafo. Por ello, en el operador de cruce se

obtiene en primer lugar un subgrafo inducido del mismo que sea conexo, y al recombinar dos padres se heredan de un padre todas las barras asociadas a los vértices que pertenezcan a dicho subgrafo y del otro padre las restantes. Hay múltiples formas de generar grafos inducidos que sean conexos. En el caso propuesto (ver Algoritmo 6: **Random Connected Bars Crossover (RCBX)**), se tiene como entrada el grafo que representa a la estructura (línea 1). En primer lugar se selecciona el tamaño del grafo inducido deseado de forma aleatoria (línea 2) , también se elige una barra de forma aleatoria (línea 3) y se inserta en el grafo inducido (línea 6), siendo marcada como barra incluida (línea 8). A continuación, se insertan en el conjunto *Connected_bars* todas las barras que estén conectadas a la misma, pasando a ser barras seleccionables (línea 10). Este proceso se repite (línea 11), pero en cada iteración en lugar de elegir cualquier barra, se elige una que no haya sido insertada aún en el grafo inducido y que esté conectada a alguna de las barras previamente seleccionadas (línea 12). El proceso finaliza cuando el grafo inducido alcanza el número de barras deseadas (*DesiredSize*). Estas barras y su complemento serán cruzadas entre los padres para generar la descendencia.

4.5. Búsqueda Local

Los algoritmos meméticos son híbridos que incorporan procedimientos heurísticos de búsqueda local para incrementar el desempeño del algoritmo. Específicamente se utilizó una búsqueda por escalada o Hill Climbing (Ver Algoritmo 1 en sección 3.2.1), para esto fue necesario definir una noción de vecindad, un criterio de parada y un criterio de selección para la aceptación del vecino.

Dado que la representación del individuo es un vector de números enteros, la vecindad serán todas las soluciones a las que se pueda llegar cambiando el valor de una entrada del vector. Se estableció como criterio de parada 75 evaluaciones a la función objetivo y para la aceptación del vecino solo se tomará soluciones que mejoren en valor de fitness a la solución actual y la vecindad se explorará en orden aleatorio.

Se muestra a continuación un ejemplo de vecindad para una solución actual **s**,

Algoritmo 6: Random Connected Bars Crossover (RCBX)

Entrada: *Graph:* Grafo con todas las barras de la estructura.
Salida : *New Graph:* Grafo con barras aleatorias.
$DesiredSize = random \in (0,$ Cantidad de barras en $Graph)$
$Bar\_selected = random$ del conjunto de barras de $Graph$
repetir
    $New\ Graph = New\ Graph \cup Bar\_selected$
    $Count\_bars++$
    Visited[$Bar\_selected$]=TRUE
    Respetir para *Barras conectadas con Bar_selected en Graph*
        $Connected\_bars.insert$(Barra en $Graph$ conectada con $Bar\_selected$)
    Fin
    repetir
        $Bar\_selected = random$ del conjunto de $Connected\_bars$
    hasta que *Visited[Bar_selected]==FALSE*
hasta que $Count\_bars == DesiredSize$
Retornar *New Graph*

los vectores que se encuentran en la parte inferior de **s** serán algunos de los posibles vecinos ($s' \in N(s)$) generados, observando que el cambio de un solo valor en alguna entrada del individuo actual bastará para generar un vecino.

s	*59*	*2*	*6*	*1*	*27*	*31*	*42*	*16*
s'	*59*	*22*	*6*	*1*	*27*	*31*	*42*	*16*
s'	*59*	*2*	*6*	*1*	*20*	*31*	*42*	*16*
s'	*3*	*2*	*6*	*1*	*27*	*31*	*42*	*16*
s'	*59*	*2*	*6*	*1*	*27*	*31*	*24*	*16*

Este tipo de esquemas son adecuados para hallar buenas soluciones en regiones especificas de la zona de búsqueda.

4.6. Mutación

En los algoritmos basados en poblaciones es habitual tener un operador de mutación. Para estas propuestas *mono-objetivo* se implementó un operador de mutación uniforme por variable.

4.7. Mecanismo de diversidad

La diversidad de la población es, sin duda, una cuestión clave en el rendimiento de los algoritmos evolutivos. Mantener una diversidad alta es importante para evitar la convergencia prematura y escapar de óptimo locales [27]. Sin embargo, mantener una diversidad excesivamente alta, provoca problemas de divergencia consistentes en que los individuos de la población están continuamente moviéndose de unas regiones del espacio de búsqueda a otras, y aunque se puedan identificar en algunos momentos ciertas regiones promisorias, el algoritmo no consigue intensificar en dichas regiones para obtener soluciones de alta calidad. El problema contrario, consistente en que todos los individuos se localicen rápidamente en unos pocas regiones perdiéndose el poder exploratorio se conoce como convergencia prematura. En la literatura es habitual usar medidas de diversidad para analizar el comportamiento de los algoritmos evolutivos y rediseñar sus componentes con base en las problemáticas que se puedan ir detectando. Una corriente más actual consiste en diseñar los algoritmos evolutivos para que de forma explícita tomen en cuenta la diversidad en el propio algoritmo con el fin de guiar la búsqueda. Esta alternativa es la que se decidió utilizar en esta tesis.

Actualmente, podemos encontrar diversos métodos que permiten controlar la diversidad de la población tanto de forma implícita como explícita cambiando diferentes componentes de los algoritmos evolutivos. Así encontramos esquemas basados en operadores de selección, en modelos de población, en operadores de cruce/mutación, en la función de fitness, y los basados en *operadores de reemplazamiento* [39]. Los basados en operadores de reemplazamiento han tenido gran éxito a la hora de abordar problemas de optimización combinatoria. El principio básico de estos métodos consiste

en preservar en cada generación individuos diversos, con el fin de que el operador de cruce no pierda su poder exploratorio [41]. Lo anterior se puede implementar de múltiples formas, y un principio que ha tenido gran éxito es el de promover mayor diversidad en las generaciones iniciales, y menor diversidad en las generaciones finales. Este es precisamente el principio del esquema de reemplazamiento que se consideró para esta tesis. Este esquema fue propuesto en [36] y recibe el nombre de BNP (*Mejor Not Penalized*) y está descrito en el Algoritmo 7. Una de las principales estrategias del BNP es evitar la selección de individuos demasiado cercanos. Específicamente, el enfoque intenta evitar la selección de pares de individuos que están más cerca que un valor D. En primer lugar une padres e hijos (línea 4) en el conjunto *Elegible*. Dado que en las fases iniciales es importante explorar, mientras que en las últimas fases el proceso debería intensificarse, el valor D varía durante la ejecución. Particularmente la variable D se inicializa con base al contenido de la primera población (ec. 4.1) y se reduce de manera lineal. Esto lo hace a través de una función lineal de manejo de diversidad que utiliza el parámetro $BNPFactor \in [0, 1]$ que es una escala para la pendiente de la misma (línea 8).

$$D_{ini} = distInitFactor \times M \tag{4.1}$$

donde M es la distancia media en la primera población y $distInitFactor \in [0, 1]$. En cada ejecución del operador se seleccionan N individuos de la población anterior y descendientes para sobrevivir. BNP selecciona iterativamente el mejor individuo que no está penalizado. Luego, todos los individuos restantes con una distancia máxima de D a los previamente seleccionados son penalizados (línea 12,13,14). Si no es posible encontrar un individuo no penalizado se toma el individuo con una distancia mayor con respecto a los individuos seleccionados actualmente. Este proceso se repetirá hasta obtener el número de individuos necesarios para pasar a la siguiente generación.

Hay que tener en cuenta que los principios de BNP son similares a los que orientaron el diseño de la estrategia RMDDC (Replacement with Multi-objective based Dynamic Diversity Control strategy) [41], pero BNP es más sencillo de implementar

y ha ofrecido resultados similares, por lo que se decidió utilizar esta estrategia.

Un hecho importante que debe tenerse en cuenta es que el operador sobreviviente requiere una función de distancia entre individuos. En este caso, se utiliza la distancia euclidiana.

4.8. Propuestas Finales

Tomando en cuenta lo expuesto previamente, se generaron un conjunto de propuestas combinando los operadores diseñados pero utilizando también operadores estándar más sencillos con el fin de validar los beneficios de los diseños realizados. En las siguientes secciones de describen las diferentes propuestas que se validaron experimentalmente. El código de estas propuestas está disponible en:

https://github.com/mgmoreno13/OptimizationBarStructures/

4.8.1. Algoritmo Memético + Cruza Uniforme

El algoritmo memético presentado anteriormente (Ver Algoritmo 4) se adaptó a nuestro problema generando así el Algoritmo 8. En este caso se hace uso del operador de cruza uniforme (línea 7) que tal y como se comentó se espera que tenga un comportamiento disruptivo significativo. Este operador fue propuesto originalmente por Syswerda [45] y se usa con $Probabilidad = 0{,}5$, es decir, que la probabilidad de heredar valores de cualquiera de los dos padres es la misma. El algoritmo además no cuenta con operador de reemplazamiento basado en diversidad sino aplica un reemplazamiento elitista donde conserva al mejor de los padres junto a la descendencia. Se implementa además una selección de padres por torneo binario basado en el valor de fitness de las soluciones (línea 6) y un operador de mutación que muta cada variable de forma uniforme (línea 8) con cierta probabilidad. Consideramos a esta primera propuesta como la más sencilla de las que se presentarán en este capítulo.

Finalmente, por ser un algoritmo híbrido, además cuenta con una búsqueda local (línea 9). En este caso es el algoritmo Hill Climbing que se encuentra previamente especificado en la sección 4.5.

Algoritmo 7: BNP

Entrada: $Population, Offspring, Gen.$
Salida : Nueva Población
$Penalized = \oslash$
$Eligible =$ Population $\cup$ Offspring
$Mejor =$ Individuo con menor valor de fitness en $Eligible$
$NewPop =$ Mejor
$Eligible = Eligible \backslash$ Mejor
Actualizar D tomando en cuenta: la generación actual del algoritmo (Gen), las generaciones totales ($GenTotal$) y valor incial de $D(D_{ini})$

$$D = D_{ini} * (1 - \frac{Gen}{BNPFactor * GenTotal})$$

Respetir hasta $|NewPop| < Size\ of\ Population$
    Respetir para $I \in Eligible$
        $I.d =$ distancia al individuo I más cercano en $NewPop$
        Si $I.d < D$ **Entonces**
            $Penalized.insert(I)$
            $Eligible = Eligible \backslash I$
    Fin

    Respetir para $I \in Penalized$
        $I.d =$ distance to the closest individual of I in $NewPop$
    Fin
    si $Eligible\ is\ \emptyset$ **entonces**
        $Selected = I$ with largest I.d in Penalized
        $Penalized = Penalized \backslash Selected$
    en otro caso
        $Selected = I$ with lowest $I.cost$ in $Eligible$
        $Eligible = Eligible \backslash Selected$
    fin
    $NewPop = NewPop \cup Selected$
Fin
Population $= NewPop$
Retornar $Population$

Algoritmo 8: Algoritmo Memético + C. Uniforme

```
1  Entrada: Criterio de parada: Generaciones
2  Salida  : Mejor individuo de la población actual
3  Inicializar población
4  Evaluación de Individuos en la población
5  repetir
6      Selección de Padres a través de Torneo Binario
7      Cruza Uniforme
8      Mutación Uniforme
9      Mejorar descendencia -a través de- Hill Climbing (Ver Alg. 1)
10     Evaluar descendencia
11     Operador de reemplazamiento: la población estará conformada por
       el mejor individuo de la generación previa junto a los Individuos de
       la descendencia
12     GeneraciónActual++
   hasta que GeneraciónActual <Generaciones
13 Retornar Mejor individuo de la población actual
```

4.8.2. Algoritmo Memético + BNP + Cruza Uniforme

La segunda propuesta (Algoritmo 9) consiste en incorporar el operador de reemplazamiento *Best Not Penalized-BNP* para incorporar el manejo explícito de diversidad discutido en la sección 4.7. Se mantiene el operador de selección por Torneo Binario (línea 6), nuevamente se maneja el operador de cruza Uniforme (línea 7), la búsqueda local es llevada acabo por el mismo Algoritmo Hill-Climbing ya discutido (línea 9) y ahora se tiene al operador BNP (línea 11) para realizar el reemplazamiento.

4.8.3. Algoritmo Memético + Cruza RCBX

La tercera propuesta (Algoritmo 10) ensambla el algoritmo original presentado en la sección 3.2.2 junto con un conjunto de operadores entre los cuales se tienen: operador de selección por torneo binario (línea 6), mutación uniforme (línea 8), búsqueda local por Hill Climbing (línea 9) y lo novedoso de esta propuesta es la incorporación del operador de cruza diseñado especialmente para este problema (línea 7). Este operador es el RCBX que se discutió en la sección 4.4.

Algoritmo 9: Algoritmo Memético + BNP + Cruza Uniforme

Entrada: Criterio de parada: *Generaciones*
Salida : Mejor individuo de la población actual
Inicializar población
Evaluar cada solución
repetir
    Selección de Padres a través de Torneo Binario
    Cruza **Uniforme**
    Mutación
    Mejorar descendencia -a través de- **Hill Climbing** (Ver Alg. 1)
    Evaluar descendencia
    Operador de reemplazamiento **BNP** (Ver Alg. 7)
    GeneraciónActual++
hasta que *GeneraciónActual* <*Generaciones*
Retornar *Mejor individuo de la población actual*

Algoritmo 10: Algoritmo Memético + Cruza RCBX

Entrada: Criterio de parada: *Generaciones*
Salida : Mejor individuo de la población actual
Inicializar población
Evaluar cada solución
repetir
    Selección de Padres a través de Torneo Binario
    Cruza **RCBX** (Ver Alg. 6)
    Mutación
    Mejorar descendencia -a través de- **Hill Climbing** (Ver Alg. 1)
    Evaluar descendencia
    GeneraciónActual++
hasta que *GeneraciónActual* <*Generaciones*
Retornar *Mejor individuo de la población actual*

4.8.4. Algoritmo Memético + BNP + Cruza RCBX

Finalmente, la última propuesta (Algoritmo 11) de este capítulo integra todos los componentes que se diseñaron. Así, cuenta con operador de selección por torneo binario (línea 6), operador de cruza RCBX (línea 7), mutación uniforme (línea 8), búsqueda local Hill Climbing (línea 9) y reemplazamiento BNP (línea 11).

Algoritmo 11: Algoritmo Memético + BNP + Cruza RCBX

```
1  Entrada: Criterio de parada: Generaciones
2  Salida   : Mejor individuo de la población actual
3  Inicializar población
4  Evaluar cada solución
5  repetir
6  |  Selección de Padres a través de Torneo Binario
7  |  Cruza RCBX (Ver Alg. 6 )
8  |  Mutación
9  |  Mejorar descendencia -a través de- Hill Climbing (Ver Alg. 1)
10 |  Evaluar descendencia
11 |  Operador de reemplazamiento BNP (Ver Alg. 7)
12 |  GeneraciónActual++
   hasta que GeneraciónActual <Generaciones
13 Retornar Mejor individuo de la población actual
```

4.9. Validación Experimental

En esta sección, se compara el desempeño de las diferentes propuestas con el fin de validar el rendimiento e importancia de cada una de las componentes diseñadas. Analizamos las propuestas aplicándolas a seis instancias. Las instancias seleccionadas son 16_bar, 40_bar, 80_bar, 120_bar, 80_bar_hor y 120_bar_hor. La etiqueta utilizando el siguiente formato [número de barras que conforman la estructura]**_bar_**. Por defecto la *carga* de la estructura será *vertical* pero si aparece **hor** al final, la estructura tendrá *carga horizontal*. Las instancias tienen diferentes niveles. Se pueden apreciar las mismas en la Figura 4-1.

El software evaluador para las instancias es el MECA [4], el cual genera los tres valores fundamentales requeridos para calcular la función de fitness: el peso propio

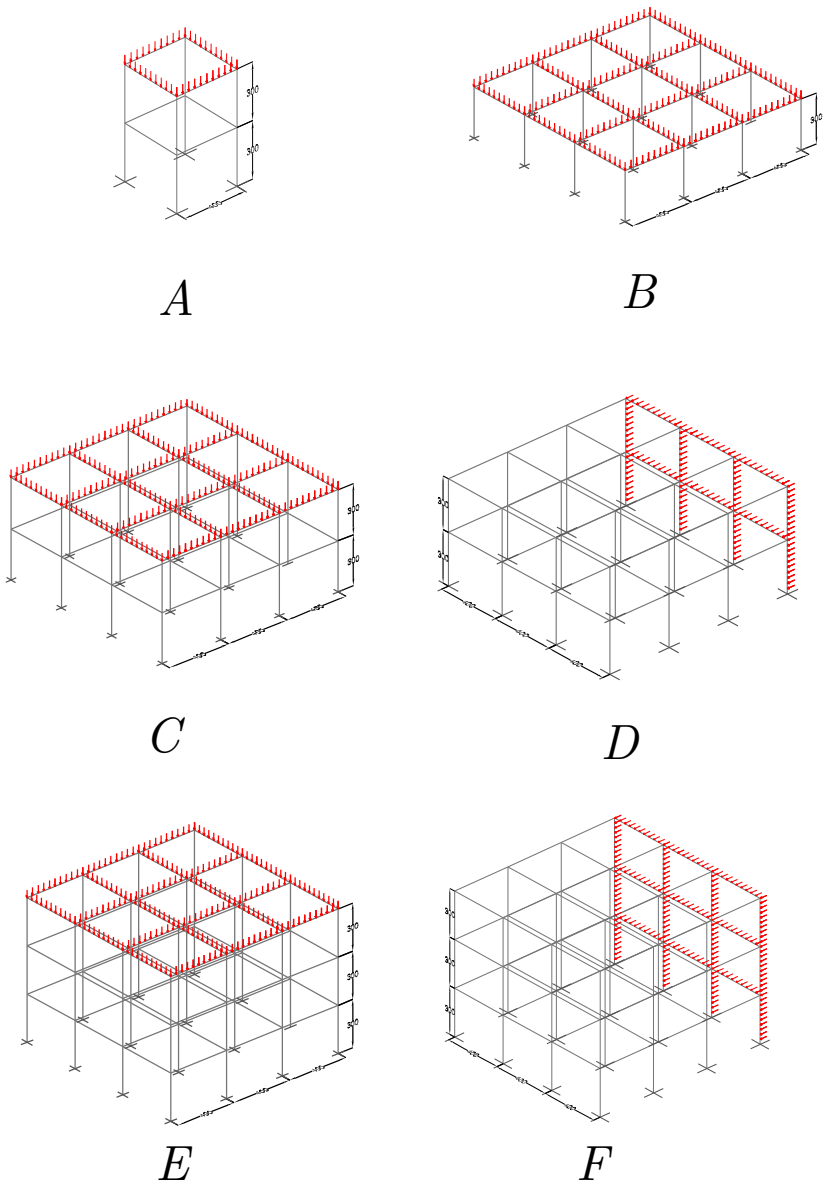

Figura 4-1: Instancias ***A***: 16_bar, ***B***: 40_bar, ***C***: 80_bar, ***D***: 80_bar_hor, ***E***:120_bar, ***F***: 120_bar_hor

de la estructura que buscamos minimizar, la eficiencia y el desplazamiento que serán las restricciones del problema.

Dado que este estudio involucra el uso de algoritmos estocásticos, cada ejecución se repitió 30 veces con distintas semillas y las comparativas se sustentan en el uso de un conjunto de test estadísticos. Concretamente dados dos conjuntos de 30 ejecuciones con una misma instancia obtenidos por dos esquemas de optimización diferentes, se realizan los siguientes tests asumiendo un nivel de significancia del 5 %. Primero, se realizó una *prueba de Shapiro-Wilk* para verificar si los valores de los resultados siguieron una distribución gaussiana. Si es así, se utilizó la *prueba de Levene* para verificar la homogeneidad de las varianzas. Si las muestras tenían la misma varianza, se aplica la prueba ANOVA. Si no, se aplica la *prueba de Welch*. Finalmente, para distribuciones no gaussianas, se utiliza la prueba no paramétrica de *Kruskal-Wallis*. En este trabajo, la frase "el algoritmo A es superior al algoritmo B" significa que las diferencias entre ellos fueron estadísticamente significativas, y que la media y la mediana de la función de fitness obtenidas por A son más bajas que la media y la mediana logradas por B. A su vez, mediante este conjunto de pruebas se calcula el *score* para todos los esquema propuestos dada una instancia, se aplica los tests par a par, y se suma un punto positivo al algoritmo que queda como ganador, uno negativo al que queda como perdedor y no sumará ningún punto si son iguales.

Nuestras propuestas implican el establecimiento de algunos parámetros: el tamaño de la población se estableció en 50, la probabilidad de cruce (pc) en 0.8, la probabilidad de mutación (pm) en $1/numero\_de\_barras$, el número de iteraciones en la búsqueda local se estableció en 75 y el criterio de parada global en $1,000$ generaciones. Estos son valores bastante típicos y cabe destacar que no se hizo un gran esfuerzo por buscar otros valores más adecuados, por lo que es posible que con técnicas de ajuste de parámetros se puedan mejorar aún más los resultados. Sin embargo, los resultados obtenidos con estos valores estándar ya fueron suficientemente convincentes para alcanzar conclusiones satisfactorias, por lo que se decidió no usar recursos computacionales adicionales para realizar dichos ajustes adicionales.

Nuestra validación experimental fue diseñada con el fin de validar la importancia

de realizar gestión explícita de diversidad y de incluir operadores de cruce específico, por lo que incluimos secciones para cada una de estas partes. En primer lugar nos centraremos en validar mediante un análisis de diversidad y rendimiento el esquema BNP, y en segundo lugar, se ofrece una comparación entre el método de cruce diseñado y el cruce uniforme.

4.9.1. Análisis de Diversidad

Una de las contribuciones que se hace en las propuestas es que se considera explícitamente la diversidad, aspecto que no se considera en los artículos relacionados. Nuestras propuestas (Algoritmos 9 y 11) dependen de un parámetro, $BNPFactor \in [0, 1]$. Se realizaron algunos experimentos preliminares, a partir de los cuales se estableció el valor en 0,9.

Para entender adecuadamente las razones detrás de la superioridad de la técnica BNP es importante analizar la evolución y actuación tanto de la diversidad como el valor de fitness de las soluciones que integran la población [35].

En la tabla 4.1 se tienen los resultados de los test estadísticos para las instancias 16_*bar*, 40_*bar* y 80_*bar*, allí se refleja el *score* para cada uno de los métodos propuestos. La instancia de 16_*bar* cuenta con un *score* igualitario en todas las propuestas, es algo esperado por ser una instancia pequeña, sin embargo en las instancias de 40_*bar* y 80_*bar* se nota como los esquemas que cuentan con +BNP obtienen *score* más altos.

En la tabla 4.2 se encuentran las instancias más complejas de 120_*bar*, 80_*bar*_*hor* y 120_*bar*_*hor*. Para este caso los *score* nuevamente ponderan a los esquemas que implementan +BNP por encima de los que no.

En el mismo sentido, se tienen los valores de fitness encontrados para cada propuesta (Ver tablas 4.3, 4.4, 4.5 y 4.6) y se verifica lo observado en las tablas 4.1 y 4.2. Los métodos Algoritmo Memético+BNP+RCBX y Algoritmo Memético+BNP+C.Uniforme consiguen para las instancias un *promedio* de fitness y el *mejor* valor que superan a los encontrados en los métodos sin BNP.

Al graficar el cambio del fitness y diversidad en el tiempo (Ver figuras 4-3 y 4-2

izquierda y derecha respectivamente) se observa en los métodos que no cuentan con BNP tales como $Memtico + C.Uniforme$, $Memtico + RCBX$ que la divergencia en la diversidad es notoria, el operador de reemplazamiento de estos métodos se basa en pasar a la siguiente generación a la descendencia y al mejor de la generación previa, estos esquemas específicamente no son lo suficientemente elitistas. Por otro lado, las implementaciones que incluyen BNP muestran un control de forma similar a la esperada, un descenso prolongado a través de las generaciones y aunque no consigue un descenso lineal logra el objetivo de manejar la diversidad en la población a nuestro favor, esto mismo abre la puerta a desarrollar otros sistemas que aporten en el control de la diversidad , este podría ser adaptable según el comportamiento que la ejecución esté tomando. Ese trabajo está más allá del alcance de este documento.

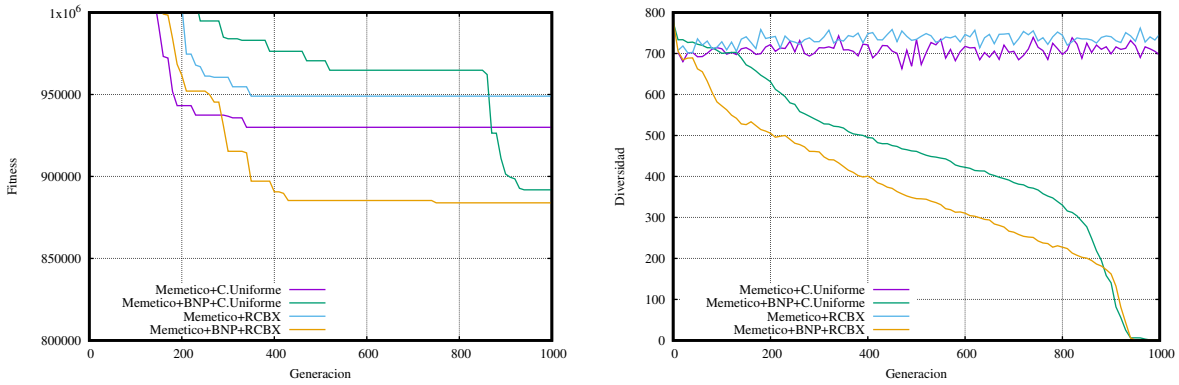

Figura 4-2: Evolución del fitness y diversidad para la instancia 80_bar

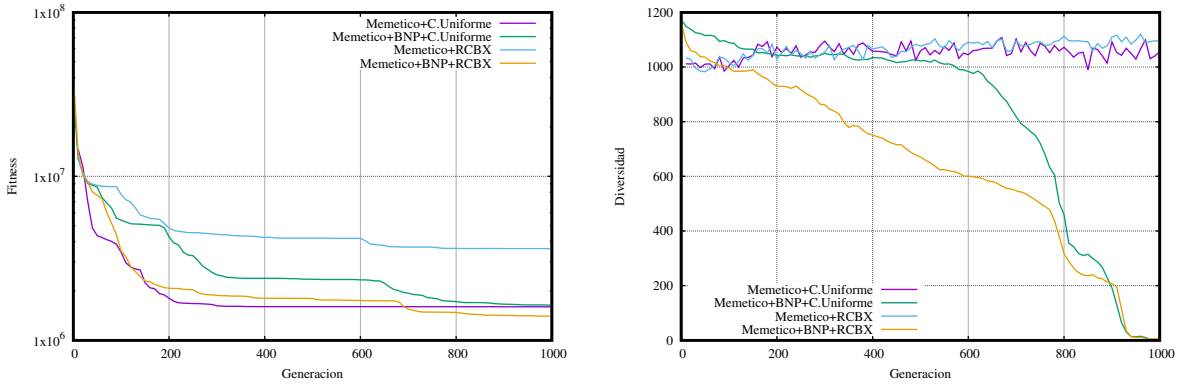

Figura 4-3: Evolución del fitness y diversidad para la instancia 120_bar

Tabla 4.1: Resultados de los test estadísticos para las instancias 16_bar, 40_bar, 80_bar

	16_bar				40_bar				80_bar			
Algoritmo	↑	↓	↔	**Score**	↑	↓	↔	**Score**	↑	↓	↔	**Score**
Memético+C.Uniforme	0	0	3	0	0	2	1	-2	1	2	0	-1
Memético+BNP+C.Uniforme	0	0	3	0	2	1	0	1	2	1	0	1
Memético+RCBX	0	0	3	0	0	2	1	-2	0	3	0	-3
Memético+BNP+RCBX	0	0	3	0	3	0	0	3	3	0	0	3

Tabla 4.2: Resultados de los test estadísticos para las instancias 120_bar, 80_bar_hor, 120_bar_hor

	120_bar				80_bar_hor				120_bar_hor			
Algoritmo	↑	↓	↔	**Score**	↑	↓	↔	**Score**	↑	↓	↔	**Score**
Memético+C.Uniforme	1	2	0	-1	1	2	0	-1	0	2	0	-2
Memético+BNP+C.Uniforme	2	1	0	1	2	1	0	1	2	0	1	2
Memético+RCBX	0	3	0	-3	0	3	0	-3	0	2	1	-2
Memético+BNP+RCBX	3	0	0	3	3	0	0	3	2	0	1	2

4.9.2. Análisis Operadores de Cruce

Esta sección se enfoca en estudiar la importancia de incluir en el algoritmo memético el operador RCBX diseñado de forma específica para el problema de optimización de estructura. Para realizar la validación se comparan los algoritmos que incluyen al operador RCBX con los que incluyen el operador de cruce uniforme.

Una vez más fijándonos en la tabla 4.1 que contiene los resultados de los test estadísticos para las instancias 16_*bar*, 40_*bar* y 80_*bar*, allí se refleja el *score* para cada uno de los métodos propuestos. Se destaca el método Memético+BNP+RCBX en todas las instancias, este método obtiene *scores* por encima de aquellos que tienen el Operador Uniforme .

Por su parte, en la tabla 4.2 se encuentran las instancias más complejas: 120_*bar*, 80_*bar*_*hor* y 120_*bar*_*hor*. El método Memético+BNP+RCBX obtiene nuevamente para estas instancias el *score* más alto. Sin embargo es importante notar que el esquema con RCBX que no implementa BNP queda rezagado en sus *scores*.

Observando las Tablas 4.3, 4.4, 4.5 y 4.6 que reúnen el *promedio*, el *mejor*, el *peor* y la *desviación estándar* de los valores obtenidos de fitness en todas las instancias tratadas, se pueden encontrar que el método Memético+BNP+RCBX consigue va-

lores *promedio* de fitness que supera a los métodos que no tienen BNP e incluso a aquellos que si tienen BNP pero no implementan el RCBX, igualmente pasa con el *mejor* valor de fitness . Además, este método que une **BNP+RCBX**, consigue valores de *desviación estándar* más bajos en comparación al resto, lo que significa que cuenta con mayor estabilidad.

Tabla 4.3: Valores obtenidos por los métodos probados para Memético+C.Uniforme

Instancia	Promedio	Mediana	Mejor	Peor	Desviación Estándar
16_bar	**157440**	157440	157440	157440	0
40_bar	437104	435360	432240	441600	3263.14
80_bar	986400	960480	959520	1.0392e+06	37337.3
120_bar	1.70335e+06	1.67396e+06	1.55919e+06	2.0994e+06	153356
80_bar_hor	1.00904e+06	1.0416e+06	924720	1.0608e+06	60136.3
120_bar_hor	1.91892e+06	2.0022e+06	1.73436e+06	2.07352e+06	150490

Tabla 4.4: Valores obtenidos por los métodos probados para Memético+BNP+C.Uniforme

Instancia	Promedio	Mediana	Mejor	Peor	Desviación Estándar
16_bar	**157440**	157440	157440	157440	0
40_bar	433775	434446	431520	435360	1637.85
80_bar	896089	908908	863760	915600	23023
120_bar	1.60729e+06	1.64192e+06	1.53773e+06	1.6422e+06	49181.1
80_bar_hor	934720	940080	904800	959280	22562
120_bar_hor	1.67115e+06	1.65396e+06	1.50972e+06	1.83996e+06	129704

Tabla 4.5: Valores obtenidos por los métodos probados para Memético+RCBX

Instancia	Promedio	Mediana	Mejor	Peor	Desviación Estándar
16_bar	**157440**	157440	157440	157440	0
40_bar	435352	435120	433200	441600	1312.32
80_bar	1.00576e+06	999600	999360	1.0308e+06	12520.4
120_bar	2.27091e+06	2.59822e+06	1.72692e+06	2.77759e+06	451514
80_bar_hor	1.05368e+06	1.0752e+06	969600	1.0824e+06	46490.2
120_bar_hor	2.11156e+06	1.89304e+06	1.7898e+06	3.44951e+06	612442

Se puede ver entonces que es la combinación $BNP+RCBX$ (Tabla 4.6) que logra alcanzar los mejores valores de fitness. Esto confirma la importancia del operador de cruce RCBX en los esquemas que incluyen BNP. Se puede interpretar basado en las pruebas que el operador RCBX provee individuos potenciales, lo cual permitirá junto a BNP la evaluación de buenos individuos a lo largo de todo el área de búsqueda.

Tabla 4.6: Valores obtenidos por los métodos probados para Memético+BNP+RCBX

Instancia	Promedio	Mediana	Mejor	Peor	Desviación Estándar
16_bar	**157440**	157440	157440	157440	0
40_bar	**428623**	428160	428160	429120	479.715
80_bar	**868720**	860880	860880	877680	8381.31
120_bar	**1.48764e+06**	1.52628e+06	1.44348e+06	1.52628e+06	41307.9
80_bar_hor	**901400**	886320	886320	921120	17244.6
120_bar_hor	**1.64632e+06**	1.65612e+06	1.57284e+06	1.72463e+06	41955.7

4.10. Resumen

Las propuestas hechas en este capítulo se enfoca en encontrar una metaheurística que pueda obtener buenas soluciones para el problema de optimización de estructuras de barras. Al ser un problema combinatorio, el campo de investigación es muy amplio, pero teniendo en cuenta los resultados obtenidos en problemas relacionados se decidió trabajar con algoritmos poblacionales híbridos, y más concretamente con el algoritmo memético. A partir de la inclusión o no de dos componentes novedosos, se desarrollaron 4 métodos de optimización. Estas dos componentes consisten en un operador de cruce que se diseñó de forma específica para el problema de optimización de estructuras y un operador de reemplazamiento que permite controlar de forma explícita la diversidad de la población.

La validación experimental muestra que el algoritmo más básico, el cual no incluye operador de reemplazo BNP ni cruce específico fue capaz de lidiar correctamente con instancias pequeñas. Sin embargo, no escala adecuadamente para instancias más grandes, presentando un rendimiento pobre para dichos casos. La inclusión de control de diversidad explícito sin cruce específico permitió superar los resultados obtenidos con la propuesta simple. Sin embargo, no se consiguió aun un esquema demasiado estable, pues tenía una desviación estándar bastante grande, y en algunas ejecuciones obtenía resultados bastante pobres. Finalmente, la combinación del cruce específico más el control explícito de diversidad permitió obtener resultados de mejor calidad y además de una forma estable. Se pudo concluir por tanto que, como en otros problemas del área de optimización combinatoria, para obtener resultados de alta calidad es necesario incluir tanto un control explícito de diversidad, como un operador de cruce

adaptado al problema de optimización tratado.

Se podrían explorar varias líneas de trabajo futuro. Primero, ahondar más sobre la búsqueda local en el MA para usar de manera más eficiente las evaluaciones y poder llegar así a resultados de alta calidad usando menos recursos computacionales. En este sentido, una de las líneas que nos gustaría explorar sería fijar de forma dinámica el criterio de parada aplicado a la búsqueda local. Por otro lado, incluir alguna estrategia para controlar los parámetros internos del algoritmo podría llevar a mejores resultados [14]. Por último, se pudo ver que la forma en que descendía la diversidad no era lineal en algunas instancias. Sería interesante analizar otras formas de calcular la diversidad deseada en cada instante, considerando incluso algunos esquemas adaptativos.

Capítulo 5

Adaptación de Algoritmos multi-objetivo

5.1. Motivación

Con base en la caracterización de la estructura (instancia) que se hace a través de MECA [4], se procuró en el enfoque mono-objetivo del Capítulo 4 encontrar la combinación óptima de secciones en la estructura para obtener el mínimo *peso propio* posible. Sin embargo, como ya se ha referido antes, la selección de dicha estructura también pasa por cumplir con un rango de valores tanto de *desplazamiento* como *eficiencia*. Específicamente para el desplazamiento, no solo bastará con cumplir este rango sino que además puede ser un objetivo relevante a la hora de tomar la decisión de construcción. Una estructura con menor valor de desplazamiento aunado al menor valor de peso propio posible sería una muy buena elección de solución.

Al formularse un problema multi-objetivo, se dio paso a una revisión del estado del arte específicamente sobre algoritmos evolutivos multi-objetivo (MOEA) y sobre hibridación de los mismos, razón por la cual se presenta anteriormente la sección 3.3.2. Por un lado se pudo concluir que existen algoritmos muy sencillos de implementar, como el NSGA-II, que han reportado buenos resultados en diversos problemas, por lo que se siguen usando hoy en día. Y a la vez, existen algoritmos más complejos, como el VSD-MOEA que para algunos problemas de gran complejidad han resulta-

do mucho más eficientes. Con base en esto se decidió usar ambos algoritmos en esta tesis. Guiándonos esto, a realizar una adaptación de algoritmos encontrados con un buen desempeño tal como el NSGA-II y el VSD-MOEA para el problema planteado. Observando a su vez, los resultados obtenidos con el problema mono-objetivo se intenta además integrar algunas de las ideas que ofrecieron mejoras importantes para el caso mono-objetivo. Es así como se llega a la hibridación de los MOEAs, buscando sumar una búsqueda local (*LS* por sus siglas en inglés) a los algoritmos encontrados y por lo tanto proponer ahora como pasar del problema multi-objetivo formulado a uno mono-objetivo, decidiéndose implementar una función de escalarización que permitirá entonces cumplir con esto.

5.2. Función de Escalarización

En la sección 3.3.3 se hace una introducción sobre lo que involucra implementar una Búsqueda Local en problemas de optimización multi-objetivo. Una de las formas de aplicar búsqueda local en optimización multi-objetivo es a través de la transformación del problema multi-objetivo en múltiples mono-objetivo. Esta fue la estrategia seguida por lo que hubo que recurrir a las funciones de escalarización para realizar dicha transformación.

Entre las funciones expuestas en la Tabla 3.1 se encuentra una forma que apenas se ha estudiado en los MOEAs es la "*Achievement Scalarizing Function*" (ASF) [24], definida como[1] :

$$u_{asf}(\mathbf{y};\mathbf{w}) = \max_{i\in\{1,\cdots,m\}} \left\{ \frac{y_i}{w_i} \right\} \tag{5.1}$$

donde $\mathbf{w}$ es un vector de pesos tal que $w_i \geq 0$ para todo $i \in \{1,\cdots,m\}$ y $\sum_{i=1}^{m} w_i = 1$. Por su parte, $\mathbf{y}$ es el vector de objetivos.

ASF es muy similar a "*Chebyshev Function*" (CHE) con respecto a su expresión matemática (ver Tabla 3.1). La principal diferencia es que en CHE el componente

[1]Se utilizó una nomenclatura diferente con el fin de tener uniformidad con las ecuaciones ya mencionadas anteriormente en esta investigación

de peso se multiplica, mientras que en ASF se divide. Sus líneas de contorno se muestran en la Figura 5-1. Es interesante observar que para CHE las líneas de contorno son incongruentes con respecto al vector de peso. Como resultado, el punto óptimo (representado como un punto negro) está muy lejos de la intersección entre el vector de peso y el frente óptimo de Pareto.

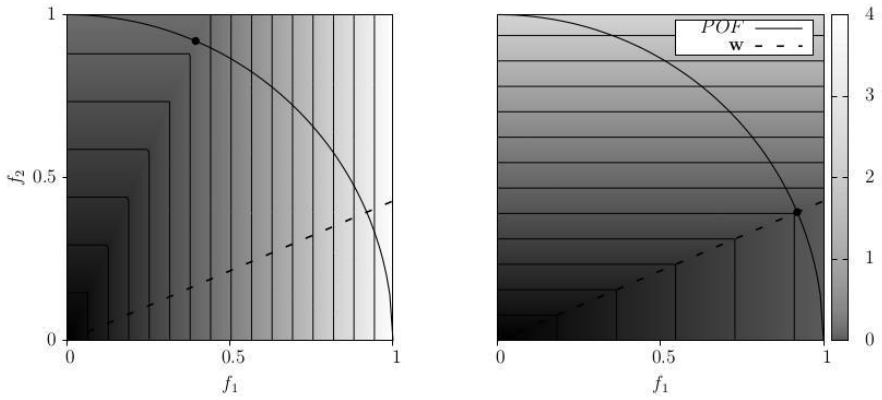

Figura 5-1: Líneas de contorno CHE (izquierda) y ASF (derecha) para los pesos $\mathbf{w}$=(0.7, 0.3).

Esto provoca que cuando se eligen pesos generados de forma aleatoria uniforme, habitualmente se consigan mejores resultados con la función de escalarización ASF, razón por la cual es la variante implementada en la búsqueda local (LS) Multi-Objetivo de esta investigación.

5.3. Normalización de Objetivos

Uno de los temas relevantes al que nos enfrentamos a la hora de implementar una función de escalarización es la normalización de los objetivos que entrarán a dicha función. Utilizando la normalización se puede tratar con objetivos de escala dispareja como es el caso del *desplazamiento* y el *peso propio* de la estructura. Es muy útil para aumentar la uniformidad de la solución [52] (evitando el *efecto de escalado*). Se ha hecho un gran esfuerzo en el tema de la normalización de objetivos en la computación evolutiva como por ejemplo en [9] y se han propuesto diversas estrategias de gran complejidad. Nosotros usamos una estrategia sencilla basada en dos casos extremos,

el mejor y el peor valor de los objetivos. Estos proporcionan información importante acerca del rango de valores en el que se encuentran las soluciones. Para lo anterior se hace indispensable disponer de (o aproximar) una solución ideal positiva y negativa [7], que se definen como sigue:

La *solución ideal positiva* se compone de todos los mejores valores objetivos alcanzables; se denota como $z^+ = z_1^+, z_2^+, ..., z_q^+$, donde z_q^+ es el mejor valor para el q-ésimo objetivo sin considerar otros objetivos.

La *solución ideal negativa* se compone de todos los peores valores objetivos alcanzables; esto se denota como $z^- = z_1^-, z_2^-, ..., z_q^-$, donde z_q^- es el peor valor para el objetivo sin considerar otros objetivos.

Se utilizó valores estáticos partiendo de los mejores y peores puntos encontrados a lo largo de las ejecuciones hechas sin normalización. La implementación se hizo utilizando la ecuación 5.2, así se consiguió mapear el valor de los objetivos al intervalo [0,1].

$$\hat{y}_i = \frac{y_i - z_i^+}{z_i^- - z_i^+} \tag{5.2}$$

Donde y_i es el vector objetivo y $\hat{y}_i$ es el vector objetivo ya normalizado, z_i^+ y z_i^- son definidos previamente e $i \in 1, ..., m$, siendo m la cantidad de objetivos.

Para el problema de estructuras de barras $m = 2$, la *solución ideal positiva* será $y^+ = [800000, 0]$ y la *solución ideal negativa* será $y^+ = [2500000, 4]$, correspondiendo a [peso,desplazamiento] para la instancia 80_bar, utilizada dicha instancia para probar las propuestas de este capítulo.

5.4. Non-dominated Sorting Genetic Algorithm II NSGA-II

El NSGA-II es un algoritmo basado en dominancia que utiliza el esquema de ordenación por frentes. Tanto en el proceso de selección de padres como en el proceso de reemplazamiento su primer elemento para discernir es el número de frente en que

el individuo quedó asignado. Una de sus principales características distintivas es la forma en que discierne entre individuos que quedaron asignados al mismo frente. Lo que se traduce en utilizar una estrategia para mantener la diversidad en el espacio objetivo que además no necesita de un parámetro predefinido por el usuario. Para describir este aporte, se define el *Estimador de densidad* basado en la distancia de crowding y el *Operador de comparación de crowded*.

El cálculo de la distancia de crowding (Ver algoritmo 13) requiere ordenar la población de acuerdo con cada valor de función objetivo en orden de magnitud ascendente. A partir de entonces, para cada función objetivo, a las soluciones límites (soluciones con mejores valores de función) se les asigna un valor de distancia infinita (Ver línea 8). A todas las demás soluciones intermedias se les asigna un valor de distancia igual a la diferencia normalizada (Ver línea 9) absoluta en los valores de función de dos soluciones adyacentes. Este cálculo continúa con todas las funciones objetivos. El valor global de distancia de crowding se calcula como la suma de los valores de distancia individuales correspondientes a cada objetivo. Cada función objetivo se normaliza antes de calcular la distancia de crowding.

Luego de que a todas las soluciones se les haya asignado la distancia de crowding, ya se puede comparar dos soluciones por su proximidad a otras soluciones. Esto es exactamente en lo que se basa el proceso de selección del NSGA II a través de *Crowded-Comparison Operator* $\prec_n$ (Ver algoritmo 12), este guía el proceso de selección en las diversas etapas del algoritmo hacia un conjunto de soluciones no dominadas, uniformemente extendidas.

Suponiendo que cada individuo en la población tiene dos atributos: 1) ordenación según dominancia. 2) distancia de crowding. Entre dos soluciones con diferentes ordenaciones por dominancia (Ver línea 2), preferimos la solución con el rango más bajo (mejor). De otra manera, si ambas soluciones pertenecen al mismo frente, entonces preferimos la solución que se encuentra en una región menos concurrida (Ver línea 4).

Dada la popularidad de NSGA-II, uno de los primeros pasos que dimos fue implementar el NSGA-II sin realizar ninguna modificación al mismo, y probar su rendimiento. Sin embargo, no dio muy buenos resultados, por lo que se extendió para

Algoritmo 12: $\prec_n$

Entrada: Individuos a comparar i y j
$i \prec_n j$ si (i_{rankeo}<j_{rankeo})
o (i_{rankeo}=j_{rankeo})
y ($i_{distancia}$>$j_{distancia}$)

Algoritmo 13: Asignación-distancia-crowding

Entrada: I (Individuos)
Salida: Distancia de crowding asignada a todos los individuos I
$l = |I|$
Respetir hasta $i \in l$
  $I[i].distancia = 0$
Fin
Respetir hasta $k \in$ *número de objetivos*
  I =sort(I,k)
  I[1].distancia=$I[l]$.distancia= ∞
  Respetir para *i=2 hasta i=l-1*
    $I[i]$.distancia=$I[i]$.distancia + (I[i+1].k-I[i-1].k)/(f_{max}-f_{min})
  Fin
Fin
Retornar I

incorporar una búsqueda local, tal y como se describe en la sección 5.6.

5.5. A Dominance-Based Multi-Objective Evolutionary Algorithm with Explicit Variable Space Diversity Management VSD-MOEA

El algoritmo VSD-MOEA cuenta con 2 contribuciones principales que fueron la motivación para utilizarlo en esta investigación. El primero es un estimador de densidad en el espacio objetivo y el segundo es una fase de reemplazamiento que no solo toma en cuenta la diversidad en el espacio de las variables sino información de ambos espacios: objetivo y variables. Para esto el estimador de densidad es usado por la fase de reemplazamiento del algoritmo [6].

Dentro de la fase de reemplazamiento (Ver algoritmo 14) se decide para cada ge-

neración cuales son los sobrevivientes entre los miembros de la población previa y la descendencia. Esta fase opera de la siguiente manera: se une la población previa y la descendencia en un solo conjunto (Ver línea 3). Se seleccionan la cantidad de individuos que se haya estipulado para la población (en un proceso iterativo) (Ver línea 7). Con el fin de tomar en cuenta la diversidad en el espacio de las variables se calcula la distancia al superviviente más cercano (DCS - Distance to Closest Survivor) para cada individuo en cada paso. DCS se calcula para cada individuo como $\text{mín}_{s \in S}\, \text{Distance}(I, s)$. donde S es el conjunto múltiple que contiene el conjunto de supervivientes actuales $s \in S$. Considerando la distancia euclidiana normalizada, y a los individuos A y B se tiene que:

$$\text{Distance}(A, B) = \left(\frac{1}{n} \sum_{i=1}^{n} \left(\frac{A_i - B_i}{x_i^{(u)} - x_i^{(l)}} \right)^2 \right)^{1/2} \tag{5.3}$$

recordando que n será la dimensión del espacio de las variables.

Nótese que individuos con un valor alto de DCS contribuyen significativamente a promover la exploración. Se penalizarán los individuos cuya contribución a la diversidad (DCS) en el espacio de las variables sea menor a un umbral (línea 9). Para calcular este umbral se utiliza una función lineal que dependerá de la generación en transcurso, por lo que se irá actualizando a medida que avance el algoritmo (línea 6), se promueve al principio más exploración e ira disminuyendo siguiendo a la función. Si todos los individuos candidatos son penalizados por contribuir poco a la diversidad en el espacio de las variables, se elegirá aquel individuo penalizado cuya contribución sea mayor en el espacio de las variables (línea 10). En caso contrario (no todos los individuos candidatos son penalizados), se seleccionará al individuo candidato no penalizado con mayor contribución en el espacio de los objetivos (línea 14).

Por su parte, el *estimador de densidad* del espacio objetivo (Ver algoritmo 15), realiza la selección del individuo candidato con mejor contribución a la calidad en el espacio de los objetivos. La calidad de cada individuo en el espacio de los objetivos es considerado tanto en la convergencia como la diversidad en dicho espacio. Así, por medio de un procedimiento de ordenación por dominancia, se selecciona al individuo

candidato o disponible (no penalizado), que pertenece al menor frente (F). Primero se inicializan los conjuntos, FP contiene los sobrevivientes que están en F (Ver línea 3) y FR contiene los restantes individuos no penalizados que están en F (Ver línea 4). Se incluye un paso para seleccionar las soluciones extremas o las mejores soluciones (mejores soluciones para cada objetivo) (Ver línea 6). Se calcula la contribución a la diversidad del espacio-objetivo para cada individuo en FR (Ver línea 10). Esta contribución se calcula tomando en cuenta los sobrevivientes actuales del frente (FP) y está definida por la *Distancia de Mejoría* (ID) (Ver línea 11). La ID de un individuo A con respecto a un individuo B se calcula teniendo en cuenta solo los objetivos donde A es mejor.

$$ID(A,B) = \left(\sum_{i=1}^{M} \left(\text{máx}\left(0, B_i - A_i\right)\right)^2 \right)^{1/2} \tag{5.4}$$

La contribución para cada miembro de FR(I) es calculado con $\text{mín}_{s \in FP}\, ID(I,s)$. Entonces, el individuo con mayor contribución es seleccionado como siguiente sobreviviente.

5.6. Propuestas Finales

En nuestra validación experimental se analizan los algoritmos NSGA-II y VSD-MOEA descritos anteriormente. Sin embargo, como se verá los resultados obtenidos no fueron muy prometedores. Teniendo en cuenta que en el caso mono-objetivo la inclusión de búsqueda local era muy importante para el rendimiento adecuado de los optimizadores, se decidió extender tanto el NSGA-II como el VSD-MOEA para incluir búsqueda local en los mismos. Con esto se pudo generar nuestras propuestas finales, que presentamos a continuación. El código de estas propuestas está disponible en:

https://github.com/mgmoreno13/OptimizationBarStructuresMultiobjective

Algoritmo 14: Fase de Reemplazamiento VSD-MOEA

Entrada: P_t (Población de la generación actual), Q_t (Descendencia de la generación actual)
Salida: P_{t+1}
$R_t = P_t \cup Q_t$
$P_{t+1} = \emptyset$
$Penalized = \emptyset$
$D_t = D_I - D_I * \frac{G_{Elapsed}}{0{,}8 * G_{End}}$
Respetir hasta $|P_{t+1}| \leq N$
    Calcular DCS de individuos en R_t, usando P_{t+1} como conjunto de referencia
    Mover individuos en R_t con $DCS < D_t$ para $Penalized$
    Si R_t *está vacio* **Entonces**
        Calcular DCS de individuos en $Penalized$, usando P_{t+1} como conjunto de referencia
        Mover los individuos $Penalized$ con los más grandes DCS para R_t
    Identificar el primer frente (F) en $R_t \cup P_{t+1}$ con un individuo $I \in R_t$
    Usar el estimador de densidad (Algoritmo 15) para seleccionar un nuevo sobreviviente desde F y moverlo a P_{t+1}
Fin
Retornar P_{t+1}

5.6.1. Non-dominated Sorting Genetic Algorithm II (NSGA-II) + Búsqueda Local

Para adaptar la versión original de NSGA-II se realizó lo siguiente:

1) Incluir el operador de cruce RCBX.

2) Incluir el operador de mutación uniforme.

3) Incluir un proceso para realizar búsqueda local.

El operador de cruce (Ver sección 4.4) y mutación (Ver sección 4.6) ya fueron descritos, y para aplicarlo en el NSGA-II no hubo que realizar ninguna modificación adicional. Sin embargo, para la búsqueda local, al disponer de dos objetivos simultáneos si hubo que realizar cambios. Se utilizó una función de escalarización nombrada ASF (Ver sección 5.2). Es necesario la asignación de pesos a cada uno de los individuos a los que se les aplicará la búsqueda por escalada. La asignación se hace a través

Algoritmo 15: Estimador de densidad

```
1  Entrada: P_{t+1} (Sobrevivientes), R_t (Candidatos), F (Frente Actual)
2  Salida: I ∈ R_t
3  FP = P_{t+1} ∩ F
4  FR = R_t ∩ F
5  Respetir hasta k ∈ número de objetivos
6  |  Seleccionar el mejor individuo I ∈ F en objetivo k.
7  |  Si I ∈ FR Entonces
8  |  |  Retornar I
   Fin
9  MaxID = 0
10 Respetir hasta Ic ∈ FR
11 |  Improvement = mín_{s∈FP} ID(Ic, s)
12 |  Si Improvement > MaxID Entonces
13 |  |  MaxID = Improvement
14 |  |  I = Ic
   Fin
15 Retornar I
```

de la partición del intervalo [0,1] en i particiones siendo i el número de individuos que conforman la población. Una vez ordenados los individuos por el primer objetivo de forma ascendente son asignados uno a uno los vectores de peso calculados.

Con el objetivo de facilitar la reproducibilidad, se ofrece el pseudocódigo completo dando una explicación del mismo (Ver Algoritmo 16). Nuestra implementación tiene como criterio de parada el número de generaciones (línea 1) y hace una clasificación basada en dominancia de los frentes que conforman los padres y la descendencia (línea 3). Se llena la nueva población partiendo del frente con menor rango mientras no se sobrepase el tamaño pre-establecido (líneas 5, 6, 7, 8). Se ordena en función del valor de la distancia de crowding las soluciones del frente que sobrepasó el tamaño de la población (línea 9), para terminar de llenar la población con aquellas soluciones que tengan mayor distancia de crowding (línea 10). Una vez obtenida la nueva población se aplicara los operadores evolutivos (línea 11) mencionados anteriormente, obteniendo así a los hijos —descendencia— para aplicar a cada una de estas soluciones Q_{t+1} una Búsqueda Local (línea 12).

Algoritmo 16: NSGA-II + Búsqueda Local

Respetir hasta $t >$*Generaciones*
  $R_t = P_t \cup Q_t$
  F=fast-non-dominated-sort(R_t) donde $F = (F_1, F_2, \dots)$
  // todos los frentes dominados de R_t
  $P_{t+1} = \emptyset$ and $i = 1$
  // Hasta que la población este llena
  Respetir hasta $|P_{t+1}| + |F_i| \leq N$
    crowding-distance-assigmnet(F_i) // (Ver algoritmo 13)
    $P_{t+1} = P_{t+1} \cup F_i$ // incluye el i-ésimo frente no-dominado en la población de padres
    $i = i + 1$ // Revisa el próximo frente para inclusión
  Fin
  Sort($F_i, \prec_n$)
  $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$
  $Q_{t+1} = make - new - pop(P_{t+1})$ // Utiliza operadores de selección (Ver Alg. 12), cruce RCBX (Ver Alg. 6) y mutación para crear la nueva población
  Búsqueda Local sobre hijos —descendencia— Q_{t+1}.
  $t = t + 1$
Fin

5.6.2. A Dominance-Based Multi-Objective Evolutionary Algorithm with Explicit Variable Space Diversity Management (VSD-MOEA) + Búsqueda Local

Al igual que la sección anterior para adaptar la versión original de VSD-MOEA se realizó lo siguiente:

1) Incluir el operador de cruce RCBX.

2) Incluir el operador de mutación uniforme.

3) Incluir un proceso para realizar búsqueda local.

El operador de cruce (Ver sección 4.4) y mutación (Ver sección 4.6) ya fueron descritos, y para aplicarlo en el VSD-MOEA no hubo que realizar ninguna modificación adicional. Sin embargo, para la búsqueda local, al disponer de dos objetivos simultáneos si hubo que realizar cambios. Se utilizó una función de escalarización nombrada ASF (Ver sección 5.2). Es necesario la asignación de pesos a cada uno de los individuos a los que se les aplicará la búsqueda por escalada. La asignación se hace a través de la partición del intervalo [0,1] en i particiones siendo i será el número de individuos que conforman la población. Una vez ordenados los individuos por el primer objetivo de forma ascendente son asignados uno a uno los vectores de peso calculados.

Con el objetivo de facilitar la reproducibilidad del método, se ofrece el pseudocódigo completo dando una explicación del mismo (Ver Algoritmo 17). El algoritmo VSD-MOEA tiene como criterio de parada el número de generaciones (línea 4), utiliza un operador de selección por torneo binario que se basa en ordenación por dominancia de las soluciones e incluye un factor aleatorio por si se genera un posible empate (línea 5). Aplica el operador de cruce RCBX (línea 6). Obteniendo así a los hijos —descendencia— para aplicar a cada una de estas soluciones una Búsqueda Local (línea 8). Por último se aplica el operador de reemplazamiento (línea 9 - Ver sección 5.5) y así se obtienen a los individuos que pasarán a la siguiente generación.

Algoritmo 17: VSD-MOEA + Búsqueda Local

Inicialización:Genera una población inicial P_0 con N individuos.
Evaluación: Evaluar todos los individuos de la población.
Asignar $t = 0$
Respetir hasta $t >$*Generaciones*
    Selección: Llena el grupo de apareamiento realizando una selección basada en torneo binario sobre P_t (basado en ordenación).
    Cruce: Aplicar cruzamiento RCBX (Ver Alg. 6) para crear a la población de hijos o descendientes Q_t.
    Evaluación: Evaluar todos los individuos en Q_t.
    Búsqueda Local sobre la descendencia/hijos.
    Selección de Sobrevivientes: Generar P_{t+1} aplicando la ***fase de reemplazamiento*** (Algoritmo 14), usando P_t y Q_t como entrada.
    $t = t + 1$
Fin

5.7. Validación Experimental

En esta sección, se compara el desempeño de las propuestas y sus implementaciones originales (sin búsqueda local). Analizamos las propuestas aplicándolas a una instancia, específicamente la de 80_bar (Ver figura 4-1 C) que consta de 2 niveles, recordando el formato: [número de barras que conforman la estructura]**_bar_**[por defecto, la *carga* de la estructura será *vertical*].

El software evaluador sigue siendo el MECA [4] al igual que en el aporte Mono-Objetivo, genera 3 valores fundamentales para la función de fitness, el peso propio de la estructura junto con el desplazamiento que buscamos minimizar y la eficiencia que será restricción del problema. Y dado que los MOEAs se enmarcan entre los algoritmos estocásticos, cada ejecución de los algoritmos se repitió 30 veces con distintas semillas.

Nuestras propuestas implican el establecimiento de algunas parámetros, el tamaño de la población se estableció en 100, la probabilidad de cruce (pc) se estableció en 0.8, la probabilidad de mutación (pm) se estableció en $1/numero\_de\_barras$ y el número de vecinos en la búsqueda local se estableció en 75. Los resultados obtenidos con estos valores son prometedores. Todas las ejecuciones se realizaron con el criterio

de parada establecido en 5000 generaciones del algoritmo, esto debido a que pruebas previas con menor número de generaciones mostraron pobres resultados.

Debido a que nuestros resultados ahora son conjuntos de soluciones, se debe pensar en la evaluación de su calidad. Previamente se ha hecho una introducción a las métricas que servirán para dicho fin (Ver sección 3.3.1). Principalmente el análisis experimental se lleva a cabo considerando el indicador de hipervolumen (HV). El HV mide el tamaño del espacio objetivo dominado por el conjunto de soluciones no dominadas obtenidas por el optimizador dado un punto de referencia. Para lograr una comparación justa se toma en cuenta los valores objetivos normalizados de las soluciones. De esta manera, cuanto más aproximada sea esta métrica a la unidad, es mayor la convergencia de las soluciones al frente de Pareto. Por otro lado esto no significa que un HV igual a la unidad sea alcanzable.

Con el objetivo de comparar los resultados de HV, se hace un análisis similar al utilizado en la sección 4.9, asumiendo un nivel de significancia del 5 %. Primero, se realizó una *prueba de Shapiro-Wilk* para verificar si los valores de los resultados siguieron una distribución gaussiana. Si es así, se utilizó la *prueba de Levene* para verificar la homogeneidad de las varianzas. Si las muestras tenían la misma varianza, se realizó una prueba ANOVA; Si no, se realizó una *prueba de Welch*. Para distribuciones no gaussianas, la prueba no paramétrica *Kruskal-Wallis* fue utilizada para probar si las muestras se extraen de la misma distribución. En este trabajo, la frase "algoritmo A es superior al algoritmo B" significa que las diferencias entre ellos son estadísticamente significativas, y que la media y la mediana de HV obtenidas por A son más altas que la media y la mediana de HV logradas por B. A su vez, mediante este conjunto de pruebas se calcula el *score* para todos los esquema propuestos dada la instancia, se aplica los tests par a par, y se suma un punto positivo al algoritmo que queda como ganador, uno negativo al que queda como perdedor y no sumará ningún punto si son iguales.

En la Tabla 5.1 se muestra el hipervolumen normalizado y a partir de aquí se analizarán los resultados obtenidos en nuestras propuestas. La tabla refleja los valores de hipervolumen mínimo y máximos, media y desviación estándar para cada uno de

los métodos.

5.7.1. NSGA - II

Buscando utilizar un esquema clásico MOEA que procurara buenos resultados para un problema como el planteado, se implementa y adapta el NSGA II. Al graficar la superficie de cubrimiento con base a los 30 frentes encontrados en las ejecuciones (Ver fig. 5-2) se puede claramente observar que ninguna de las soluciones dentro del frente se acerca al *mínimo peso* encontrado con la propuesta mono-objetivo (Ver Tabla 4.6). Dicho valor está graficado en corte con el eje de *Peso* para poder compararlo con lo obtenido en la superficie de cubrimiento.

Buscando cuantificar las aproximaciones al frente de Pareto obtenidas, se utilizó como métrica el hipervolumen de allí que se obtuviera a partir de las 30 muestras una *media* de 0,624 (Tabla 5.1). Aunado a esto en la Tabla 5.2 se tienen los resultados de los test estadísticos, allí se refleja el *score* para cada uno de los métodos propuestos. El NSGA II en comparación con las demás propuestas obtiene el score más bajo. Llevando esto a plantear una propuesta que contara con aportes en diversidad.

Tratando entonces alcanzar o acercar el conjunto/frente a la solución encontrada en el mono-objetivo se analizó los conjuntos de individuos y la diversidad en el espacio de las variables (Ver fig. 5-4). Observando la diversidad a través de las generaciones del NSGA II, si bien se nota un brusco descenso de la misma en el espacio de las variables antes de las 1000 generaciones, se mantiene en un nivel que permite la exploración de la zona de búsqueda. Con esto se puede inferir entonces que la diversidad promovida en el espacio objetivo[2] influye directamente en promover la diversidad en el espacio de las variables. Esto permite mantener por el resto de las generaciones un nivel significativo de diversidad, que además evita la convergencia prematura del conjunto solución.

[2]Específicamente a través de la asignación de la *distancia de crowding.* a cada individuo y la ordenación según dominancia de los mismos

Alg.	Mín	Max	Mean	S. Deviation
NSGA II	0.593	0.655	0.624	0.015
NSGA II-LS	0.665	0.700	**0.685**	0.006
VSD-MOEA	0.619	0.650	0.638	0.007
VSD-MOEA-LS	0.653	0.678	0.667	0.005

Tabla 5.1: Valores de Hipervolumen obtenidos por los 4 métodos propuestos para la instancia 80_bar

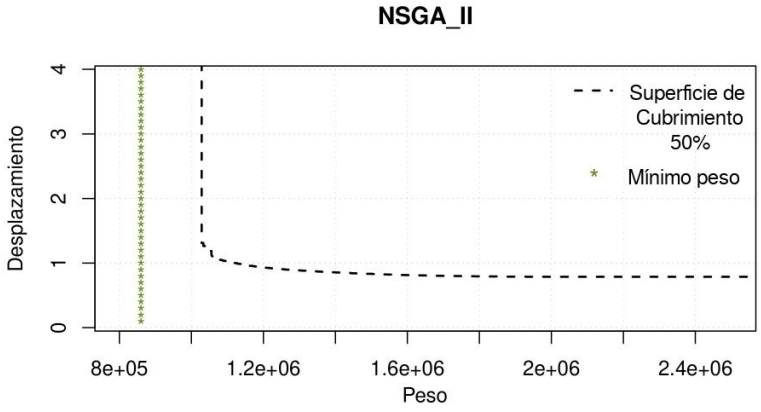

Figura 5-2: Superficie de Cubrimiento percentil al 50 % para NSGA II

	80_bar			
Algoritmo	↑	↓	↔	**Score**
NSGA II	0	3	0	-3
NSGA II - LS	3	0	0	3
VSD-MOEA	1	2	0	-1
VSD-MOEA - LS	2	1	0	1

Tabla 5.2: Resultados de los test estadísticos para las propuestas multi-objetivo

5.7.2. Gestión de diversidad a través del VSD-MOEA

Los resultados obtenidos con el NSGA II motivaron a buscar otro algoritmo en el estado del arte y es la razón por la que se decide adaptar un algoritmo específicamente que pueda implementar un control de diversidad, no solo en el espacio objetivo (tal como lo hace el NSGA II con la distancia de crowding y como lo hace VSD-MOEA que cuenta con *estimador de densidad del espacio objetivo* (Ver sección 5.5)) sino en el espacio de las variables, el valor DCS[3] asignado a cada una de las soluciones con el mejor rango de dominancia permite realizar un reemplazamiento (Ver algoritmo 14) en el VSD-MOEA que procura administrar la exploración en el conjunto de las soluciones (análogo al BNP[4] implementado en mono-objetivo).

Se analiza su actuación a través del valor de hipervolumen registrado en las ejecuciones del VSD-MOEA, donde se evidencia un mejor valor que el registrado por el NSGA II (Ver Tabla 5.1). Si sacamos la superficie de cubrimiento (Ver fig. 5-3) se aprecia que aún el conjunto solución se encuentra lejos del valor alcanzado por el *mínimo peso* encontrado con la propuesta mono-objetivo.

Además se realizó el análisis de diversidad (Ver fig. 5-4) para poder verificar la administración/manejo de la exploración que es introducida por la naturaleza del algoritmo VSD-MOEA. Si se observa el comportamiento que sigue la diversidad podemos relacionarlo fácilmente con la función que determina el umbral de distancia a lo largo de las generaciones. La diversidad si bien llega a bajar, una vez en cierto nivel no sigue decreciendo pues se sigue manteniendo la diversidad en el espacio objetivo produciendo cierto nivel de diversidad en el espacio de las variables. Por ello, los resultados no difieren tanto al NSGA-II (Ver fig. 5-2) y aunque también en la Tabla 5.2 el VSD-MOEA mejora un poco en el *score*, no es tan relevante como lo que pasó en el caso mono-objetivo al tener control de diversidad en el espacio de las variables.

[3]Distance to Closest Survivor
[4]Reemplazamiento Best Not Penalized

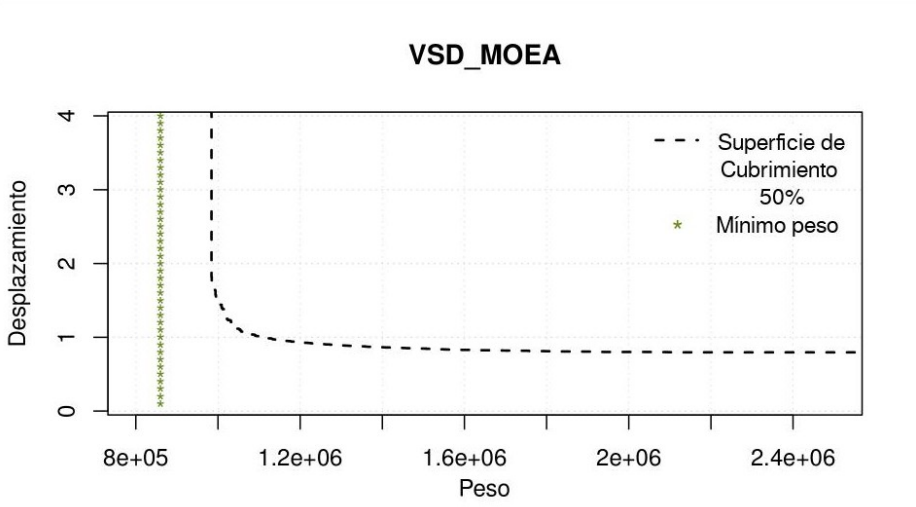

Figura 5-3: Superficie de Cubrimiento percentil al 50 % para VSD-MOEA

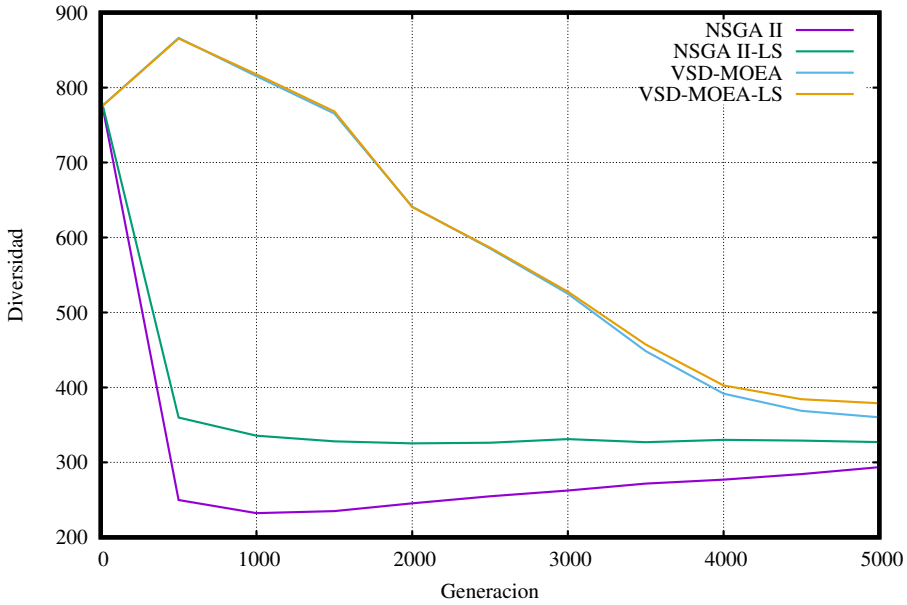

Figura 5-4: Diversidad en el espacio de las variables

5.7.3. Búsqueda Local en MOEAs

Una vez planteadas las adaptaciones del NSGA II y VSD-MOEA, se seleccionó la función de escalarización *Achievement Scalarizing Function*" (ASF) a través de pruebas empíricas para así poder implementar una búsqueda local en ambas propuestas. Se busca obtener una analogía a lo que demostró funcionar en la propuesta mono-objetivo, es decir, la hibridación de algoritmos (Ver algoritmos 16 y 17).

Realizada las ejecuciones para obtener las aproximaciones de frentes de Pareto, se calcula el HV, donde en efecto el valor de HV se mejoró significativamente en comparación con los valores alcanzados por los algoritmos simples, el mejor valor de HV lo alcanzó la implementación NSGA II-LS, sin embargo la implementación VSD-MOEA-LS logró mejorar los valores del NSGA II y VSD-MOEA.

Al hacer el análisis de diversidad (Ver fig. 5-4) de ambas propuestas (VSD-MOEA-LS y el NSGA II-LS), en el VSD-MOEA-LS se puede observar como la gráfica que muestra el comportamiento de la diversidad a lo largo de las generaciones, sigue un descenso continuado producto de la función lineal que permite el control de la diversidad en la población a lo largo de las generaciones (Ver Algoritmo 14 —línea 4—), mientras que el NSGA II-LS baja bruscamente su diversidad en el espacio de las variables antes de las 1000 generaciones pero logra mantener un valor no insignificante de diversidad para el resto de las generaciones.

La diversidad en el espacio de las variables inducido en el VSD-MOEA-LS y guiada por la función lineal antes mencionada no parecer tener relevancia al menos para la instancia trabajada en este capítulo, contrario al aporte de la Búsqueda Local. Esto nos permite inferir que la diversidad promovida en el espacio objetivo por las variantes del NSGA II es suficiente para a su vez promover la diversidad en el espacio de las variables y lograr que sea significativa para la exploración de la zona de búsqueda. Además la intensificación promovida por la Búsqueda Local si logra una gran diferencia en los frentes resultantes.

Observando las superficies de cubrimiento (Ver fig. 5-5), el NSGA II-LS es superior y se muestra las zonas especificas del frente donde puede lograr superar al

VSD-MOEA-LS con mayor probabilidad. La linea roja punteada es el *mínimo peso* encontrado con la propuesta mono-objetivo (Ver Tabla 4.6) y ambas propuestas se acercan en su frente a este valor. El NSGA II-LS incluso supera el valor. Sin embargo, hay que hacer notar que la propuesta multi-objetivo se ejecutó por 5000 generaciones y la mono-objetivo por 1000 generaciones, por lo que habría que realizar más análisis que contemplen los beneficios de cada uno de los planteamiento bajo diferentes condiciones de uso.

Si se observa la Tabla 5.2 los métodos que implementan la búsqueda Local (LS) obtienen el mejor *score* sobre los métodos que no lo hacen. A su vez el NSGA II - LS se destaca por tener el *score* más alto entre nuestras propuestas.

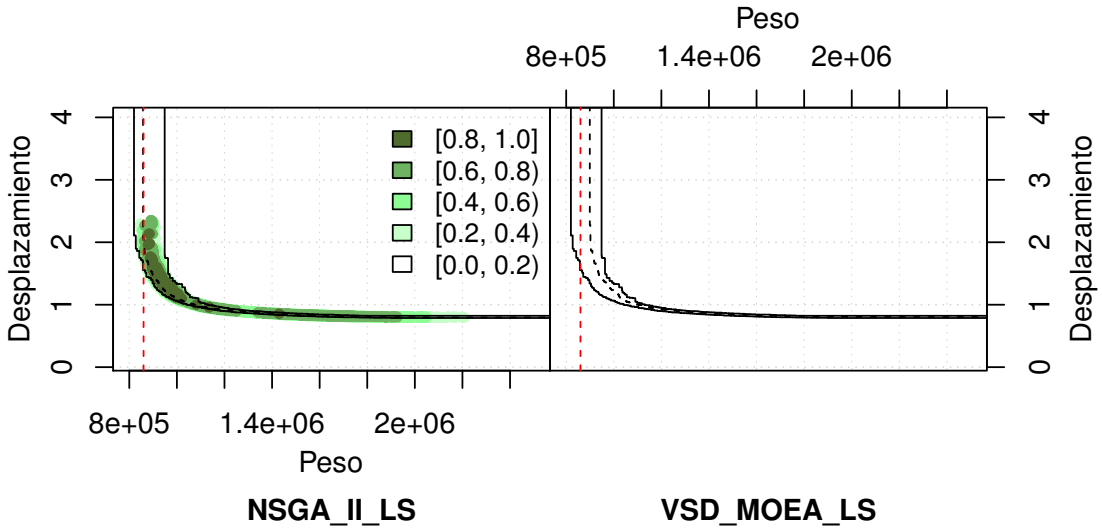

Figura 5-5: Superficies de cubrimiento NSGA II-LS y VSD-MOEA-LS

5.8. Resumen

Las propuestas hechas en este capítulo se enfocan en encontrar un Algoritmo Evolutivo Multi-Objetivo que pueda arrojar un conjunto de soluciones al problema de Estructuras de Barras (minimizando el peso y el desplazamiento de la estructura).

Las principales propuestas hechas se basan en esquemas basados en dominancia, control de diversidad en el espacio objetivo y de las variables, e hibridación de MOEAs a través de la función de escalarización ASF en la Búsqueda Local. Se tienen entonces 4 aportes principales que son NSGA II, VSD-MOEA, NSGA II-LS, VSD-MOEA-LS.

La validación experimental arrojó que el NSGA II-LS podía obtener frentes con mayor valor de HV y llegar a zonas del espacio objetivo que ninguno de los otros algoritmos pudieron. Si bien el control de diversidad en el espacio de las variables no fue determinante para el problema multi-objetivo si queda claro que por la naturaleza del problema hay una relación entre procurar la diversidad del espacio objetivo y el de las variables.

Se podría a futuro trabajar en verificar si esta hipótesis acerca de la naturaleza del problema es cierta en otras instancias y además probar otras funciones que guíen el manejo de diversidad en las variables (específicamente en las variantes del VSD-MOEA). Así también buscar una adaptación de la búsqueda local a lo largo de las generaciones que permita un uso óptimo de las evaluaciones de función.

Capítulo 6

Conclusiones

6.1. Conclusiones

El problema de optimización de estructuras de barras surge específicamente en aplicaciones de ingeniería civil. Para esta investigación se seleccionó una de las variantes de este problema que consiste en dada una estructura de barras tridimensional, seleccionar de un catálogo de concreto dado, las secciones (conjunto de características tales como: altura, base, resistencia del concreto, resistencia del acero y número de varillas) que serán asignadas a cada una de las barras que conforman la estructura, de tal manera que el peso total (o costo) sea minimizado, al tiempo que satisfaga un conjunto de restricciones de diseño, como la eficiencia y el desplazamiento.

Matemáticamente, este problema puede formularse en términos de optimización combinatoria (OC), y puede plantearse no solo de manera mono-objetivo sino también multi-objetivo. Al formular el problema en el enfoque multi-objetivo, el desplazamiento pasará a ser un objetivo a minimizar, a su vez, seguirá siendo junto a la eficiencia restricciones del problema.

La variante del problema de optimización de estructura de barras presentada en esta investigación no ha sido visiblemente desarrollada, es por esto que partimos proponiendo 6 instancias/estructuras que cuentan con distintas dimensiones (cantidad de barras) y sometidas a distintas cargas que permitieron evaluar los aportes hechos a la resolución del problema. Ahora bien, transformando el problema a una visión

matemática nos encontramos frente a un problema combinatorio cuyo espacio de búsqueda está definido por Tamaño_Catálogo$^{\text{Dimensión}}$. Tomando esto en cuenta, nuestra propuesta se basa en una metaheurística que pueda abarcar efectivamente las zonas promisorias de búsqueda.

Se planteó un algoritmo mono-objetivo para la minimización del peso de la estructura, específicamente un *algoritmo memético*. Este cuenta con operador de cruce adaptado al problema (Operador RCBX) y una búsqueda local que se basa en una *Búsqueda por escalada*. Un operador de reemplazamiento (BNP —Best Not Penalized—) permitirá a su vez un control sobre la diversidad en el espacio de las variables de los individuos que conforman la población a lo largo de las generaciones. Se hizo la validación de esta propuesta a través de un análisis estadístico y a su vez se hizo un análisis de diversidad a las diferentes variantes diseñadas. El valor del peso mínimo encontrado para cada estructura evaluada resultó ser promisorio.

Para poder llegar a resultados de alta calidad con instancias pequeñas (16_bar) será suficiente usar el Algoritmo *Memético + C.Uniforme*. Sin embargo, este esquema no escala adecuadamente para instancias más grandes, presentando un rendimiento pobre para dichos casos. Agregar un control de diversidad explícito sin cruce específico permitió mejorar los resultados obtenidos con la propuesta simple. Aunque no se consiguió un esquema estable, pues tenía una desviación estándar bastante grande, que afectó directamente a los resultados de algunas ejecuciones. Finalmente, la combinación del cruce específico más el control explícito de diversidad permitió obtener resultados de mejor calidad y además de una forma estable. Se puede inferir entonces que para obtener resultados de alta calidad es necesario incluir tanto un control explícito de diversidad, como un operador de cruce adaptado al problema de optimización tratado.

También se planteó un algoritmo multi-objetivo que minimizará tanto el *peso* como el *desplazamiento* de la estructura. Se optó por la hibridación de los algoritmos NSGA II y VSD-MOEA. El NSGA II es un esquema clásico en optimización multi-objetivo, se basa en la ordenación de las soluciones a través del *Non-dominated sort* y el calculo de la distancia de crowding por su parte el VSD-MOEA es una propuesta

novedosa que cuenta con control de diversidad en el espacio objetivo y en el espacio de las variables. Se utiliza para ambos algoritmos una función de escalarización para así añadir una búsqueda local que permitiese llegar a un buen conjunto de soluciones. La validación experimental se hace a través de la métrica de hipervolumen, además que se utilizan superficies de cubrimiento para poder comparar los distintos métodos multi-objetivo, los resultados obtenidos muestran la relevancia de la hibridación, y al menos para este enfoque, el control de diversidad en el espacio de las variables no muestra ser tan relevante como lo fue en el mono-objetivo.

Para finalizar se destaca entonces la importancia del operador de cruce RCBX diseñado y utilizado en los métodos mejor calificados cuantitativamente en las validaciones, además el operador de diversidad que permite el control sobre los individuos que son seleccionados para la población a lo largo de las generaciones procurando así la exploración de las zonas más prometedoras del espacio de búsqueda. Y por último, la intensificación a cargo de la búsqueda local, esta es aplicada sobre cada individuo de la población. Los resultados obtenidos en los mono-objetivo y multi-objetivo nos hace pensar en que la linea de investigación si bien ha traído muy buenos resultados solo es una mínima parte de todo el trabajo que queda por realizar.

6.2. Trabajo Futuro

En primer lugar, creemos que consolidar un conjunto de instancias de prueba es necesario, donde se sumen a las estructuras propuestas en esta investigación, otras que tengan mayores dimensiones e incluso de estructuras reconocidas en el ámbito.

Muy relacionado a las instancias de prueba se encuentran los catálogos de secciones, cada catálogo cuenta con un número especifico de secciones de una material y registra los distintos tamaños y características para cada una. Estos catálogos no poseen un orden dentro de ellos, pero se puede inferir que para nuestro problema elegir un criterio para ordenarlos sería determinante.

Por otro lado, al estar el valor de las variables sujetas a un catálogo, determinar la similitud entre una y otra es algo que puede conllevar a la definición de un criterio o métrica para poder cuantificarlo, probar con la distancia Hamming sería otro aporte a esta linea de trabajo.

El manejo de diversidad en el espacio de las variables de las propuestas basadas en el VSD-MOEA se ha llevado a cabo con una función lineal expuesta en el Algoritmo 14 específicamente en la línea 6. Probar otras configuraciones de dicha función podría igualmente llevar a encontrar muy buenos resultados, promoviendo de esta manera una intensificación más temprana y que permita al algoritmo mejorar su actuación.

Nuestros aportes han recurrido a la hibridación de algoritmos y ha resultado ser una técnica satisfactoria, sin embargo la inversión de evaluaciones es alta, por lo que una búsqueda local adaptativa a lo largo de las generaciones podría administrar las evaluaciones a la función fitness de forma más eficiente. Una opción podría ser evaluar el avance con la métrica de hipervolumen y a partir de ahí condicionar la cantidad de evaluaciones de la búsqueda local.

Por último, utilizar MOEAs de otros paradigmas, como los basados en descomposición o en métricas para abordar el problema de optimización de estructura de barras podrían ser de utilidad para futuros aportes al estado del arte.

Bibliografía

[1] James C Bean and Atidel ben Hadj-Alouane. *A dual genetic algorithm for bounded integer programs.* 1993.

[2] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[3] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003.

[4] S. Botello, J. L. Marroquin, A. B. Rionda, and R. Ducoing. MECA Programa para el Análisis Matricial de Estructuras. *Facultad de Ingeniería Civil, Universidad de Guanajuato*, 1997.

[5] Karl Bringmann and Tobias Friedrich. Don't be greedy when calculating hypervolume contributions. In *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, pages 103–112. ACM, 2009.

[6] J. Chacón, C. Segura, and C. A. Coello. VSD-MOEA: A Dominance-Based Multi-Objective Evolutionary Algorithm with Explicit Variable Space Diversity Management. 2019.

[7] Runwei Cheng and Mitsuo Gen. *Genetic algorithms and engineering design.* John Wiley, 1997.

[8] Carlos A Coello and Margarita Reyes Sierra. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *Mexican International Conference on Artificial Intelligence*, pages 688–697. Springer, 2004.

[9] Dragan Cvetkovic and Ian C Parmee. Preferences and their application in evolutionary multiobjective optimization. *IEEE Transactions on evolutionary computation*, 6(1):42–57, 2002.

[10] Richard Dawkinss. *The selfish gene.* Oxford University Press, Oxford, UK, 1976.

[11] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[12] SE Dosso and DW Oldenburg. Magnetotelluric appraisal using simulated annealing. *Geophysical Journal International*, 106(2):379–385, 1991.

[13] W.J. Duncan and A.R. Collar. XV. matrices applied to the motions of damped systems. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 19(125):197–219, 1935.

[14] A. E. Eiben and S. K. Smit. Evolutionary algorithm parameters and methods to tune them. *Autonomous search*, pages 15–36, 2011.

[15] A. E. Eiben and James E. Smith. *Introduction to evolutionary computing*, volume 53. Berlin: springer, 2003.

[16] C. Emmanouilidis, A. Hunter, and J. MacIntyre. A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, pages 309–316 vol.1, July 2000.

[17] C. A. Felippa. Parametric unification of matrix structural analysis: Classical formulation and d-connected mixed elements. *Finite Elem. Anal. Des.*, 21(1-2):45–74, October 1995.

[18] Carlos A. Felippa. A historical outline of matrix structural analysis: a play in three acts. 2001.

[19] Mark Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 519–533. Springer, 2003.

[20] R.A. Frazer, W.J. Duncan, and A.R. Collar. *Elementary Matrices and Some Applications to Dynamics and Differential Equations*. Cambridge University Press Cambridge, 1938.

[21] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[22] O. M. González, C. Segura, S. I. V. Peña, and C. León. A memetic algorithm for the capacitated vehicle routing problem with time windows. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2582–2589, June 2017.

[23] Raquel Hernández Gómez. *Hiper-Heurísticas Paralelas para Optimización Multi-Objetivo*. PhD thesis, Instituto Politécnico Nacional, 2018.

[24] Raquel Hernández Gómez and Carlos A Coello Coello. Improved metaheuristic based on the R2 indicator for many-objective optimization. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 679–686. ACM, 2015.

[25] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 110–125. Springer, 2015.

[26] Joshua Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, pages 552–557. IEEE, 2005.

[27] Bianchi Leonora, Dorigo Marco, Gambardella Luca Maria, and Gutjahr Walter J. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.

[28] A.H. Nilson, D. Darwin, and C.W. Dolan. *Design of concrete structures*. McGraw-Hill Higher Education, 2010.

[29] Nils J Nilsson. *The quest for artificial intelligence*. Cambridge University Press, 2009.

[30] E. Ortega, E. Salazar Solano, S. Botello, M. Tapia, I. Munguía, J. C. Gomez, D. Quiroz, and M. Yañez. Diseño óptimo de estructuras de barras tridimensionales. aplicaciones a vivienda popular ya estructuraciones no convencionales. *XVIII Congreso de Ingeniería Estructural*, 2012.

[31] Moscato Pablo. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts. *Caltech concurrent computation program C3P Report*, 826:1989, 1989.

[32] Charles C Palmer and Aaron Kershenbaum. An approach to a problem in network design using genetic algorithms. *Networks*, 26(3):151–163, 1995.

[33] M Pilát. Evolutionary multiobjective optimization: A short survey of the state-of-the-art. *Proceedings of the Contributed Papers Part I-Mathematics and Computer Sciences, WDS, Prague, Czech*, pages 1–4, 2010.

[34] C. A. Riveros. *Análisis Estructural*. Universidad de Antioquia, 2017.

[35] E. Romero Ruiz and C. Segura. Memetic algorithm with hungarian matching based crossover and diversity preservation. *Computación y Sistemas*, 22(2), 2018.

[36] Emmanuel Romero Ruiz and Carlos Segura. Memetic algorithm with hungarian matching based crossover and diversity preservation. *Computación y Sistemas*, 22, 2018.

[37] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[38] Saïd Salhi. *Heuristic search: The emerging science of problem solving*. Springer, 2017.

[39] C Segura, A Aguirre, S. I. V. Peña, and S. B Rionda. The Importance of Proper Diversity Management in Evolutionary Algorithms for Combinatorial Optimization. *NEO 2015*, pages 121–148, 2017.

[40] C. Segura, Y. González, G. Miranda, and C. León. A multi-objective evolutionary approach for the antenna positioning problem. *In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 1:51–60, 2010.

[41] C. Segura, A. Hernandez, F. Luna, and E Alba. Improving diversity in evolutionary algorithms: New best solutions for frequency assignment. *IEEE Trans. Evol. Comput*, pages 539–553, 2017.

[42] Ines Sghir, Jin-Kao Hao, Ines Ben Jaafar, and Khaled Ghédira. A distributed hybrid algorithm for the graph coloring problem. In *Revised Selected Papers of the 12th International Conference on Artificial Evolution - Volume 9554*, pages 205–218, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[43] Margarita R Sierra and Carlos A Coello. A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms. *Technical Report of CINVESTAV-IPN*, 2004.

[44] Alice Smith, David Coit, Thomas Bäck, David Fogel, and Zbigniew Michalewicz. *Penalty Functions*. University of Pittsburgh in Section C 5.2 of Handbook of Evolutionary Computation, 1998.

[45] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[46] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[47] El-Ghazali Talbi et al. *Hybrid metaheuristics*, volume 166. Springer, 2013.

[48] N. Weicker, G. Szabo, K. Weicker, and P. Widmayer. Evolutionary multiobjective optimization for base station transmitter placement with frequency assignment. *IEEE Transactions on Evolutionary Computation*, 7(2):189–203, 2003.

[49] Gerhard J Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial optimization—eureka, you shrink!*, pages 185–207. Springer, 2003.

[50] E. J. Muttio Zavala. Diseño, análisis y optimización estructural paramétrica de edificaciones. Tesis de Licenciatura, Departamento de Ingeniería Civil División de Ingenierías Universidad de Guanajuato, 2017.

[51] E. J. Muttio Zavala, E. Salazar Solano, and S. Botello Rionda. *Análisis Estructural Estático y Dinámico Armaduras, Vigas y Marcos*. Universidad de Guanajuato, 2017.

[52] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[53] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[54] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. *Performance assessment of multiobjective optimizers: an analysis and review*, volume 7. IEEE Transactions on Evolutionary Computation, April 2003.

[55] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), 1999.

[56] Eckart Zitzler and Simon Künzli. *Indicator-based selection in multiobjective search*. International Conference on Parallel Problem Solving from Nature, 2004.