Hiperheurísticas para resolver el problema de empacado irregular de material en dos dimensiones

T E S I S

Maestría en Ciencias en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Eunice López Camacho

Mayo 2007

Hiperheurísticas para resolver el problema de empacado irregular de material en dos dimensiones

TESIS

Maestría en Ciencias en

Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Eunice López Camacho

Mayo 2007

Instituto Tecnológico y de Estudios Superiores de Monterrey

División de Graduados en Tecnologías de Información y Electrónica

Los miembros del comité de tesis recomendamos que la presente tesis de Eunice López Camacho sea aceptada como requisito parcial para obtener el grado académico de Maestro en Ciencias con especialidad en:

Sistemas Inteligentes

Comité de tesis:

Dr. Hugo Terashima Marín
Asesor de la tesis

Dr. Manuel Valenzuela Rendón
Sinodal

Dr. Eduardo Uresti Charre
Sinodal

Dr. Graciano Dieck Assad
Director del Programa de Graduados en Tecnologías de Información y Electrónica

Mayo de 2007

Hiperheurísticas para resolver el problema de empacado irregular de material en dos dimensiones

Por

Eunice López Camacho

TESIS

Presentada a la División de Tecnologías de Información y Electrónica

Este trabajo es requisito parcial para obtener el grado académico de Maestro en Ciencias con especialidad en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Monterrey, N.L. Mayo de 2007

A mi esposo Arturo
A mis hijos Arturo y Pablo
A mis padres Salomón y Felícitas

Reconocimientos

Deseo externar un sincero agradecimiento a las personas que de alguna forma colaboraron en el desarrollo de esta tesis.

Al Dr. Hugo Terashima Marín, por su excelente guía y dedicación durante el desarrollo de la presente investigación.

Al Dr. Eduardo Uresti Charre por su valiosa retroalimentación para este trabajo.

Al Dr. Manuel Valenzuela Rendón por sus adecuados comentarios y correcciones para esta tesis.

A mis familiares, en especial a mi esposo Arturo por todo su amor y paciencia. A mis padres y hermanos por su apoyo incondicional.

A mis amigos y compañeros de la maestría especialmente a Claudia Janneth Farías y Gisela Medina por su apoyo y consejos.

Eunice López Camacho

Instituto Tecnológico y de Estudios Superiores de Monterrey
Mayo 2007

Hiperheurísticas para resolver el problema de empacado irregular de material en dos dimensiones

Eunice López Camacho, M.C.
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2007

Asesor de la tesis: Dr. Hugo Terashima Marín

El problema de corte y empacado de materiales en dos dimensiones tiene gran relevancia práctica; por ejemplo, en la industria textil, de lámina metálica, madera, piel, plástico y papel [1]. Es por esto que ha sido ampliamente estudiado en los últimos años, sobre todo en lo que se refiere al corte de piezas rectangulares. El corte de figuras irregulares es más complejo por lo que es posible abordarlo desde muy distintos enfoques. La presente investigación aplica al caso de los polígonos irregulares, un método para generar hiperheurísticas basado en un algoritmo genético (AG) que ha probado ser eficaz para el problema de corte de figuras rectangulares. Este método es llamado GHH-2D y fue descrito por Terashima-Marín [28] y Farías [14].

Al generalizar el modelo GHH-2D a polígonos irregulares se incrementa la complejidad geométrica; por ejemplo: el cálculo de áreas, intersecciones, desplazamientos y rotaciones. El AG usa una representación de longitud variable que evoluciona combinaciones de reglas del tipo condición-acción. Las condiciones de cada regla representan posibles estados de un problema mediante un vector de 8 números reales. Esta representación toma en cuenta el porcentaje de piezas que faltan por acomodar, la altura, el área y la rectangularidad de las piezas que faltan por acomodar. Gomes y Oliveira [19] definen rectangularidad con base en la diferencia entre el área de la pieza y el área del rectángulo que la contiene. Las acciones de cada regla indican heurísticas simples que encuentran una solución al problema de corte y empacado de material en dos dimensiones. La función primordial del AG es la de seleccionar diferentes estados del problema y asociar cada uno de ellos con alguna de las cuarenta acciones disponibles.

Se realizaron experimentos donde el AG se evolucionó durante 500 ciclos o generaciones. El mejor individuo del último ciclo constituye la hiperheurística generada la cual relaciona distintos estados del problema con heurísticas simples convenientes, de tal modo que al resolver una instancia utilizando las reglas de la hiperheurística se

obtienen, en promedio, mejores resultados que empleando cualquiera de las heurísticas simples de manera exclusiva. El modelo desarrollado considera instancias con polígonos convexos de tres a ocho lados que deberán acomodarse en el menor número de objetos rectangulares idénticos. Para probar el modelo propuesto se generaron aleatoriamente 540 instancias, además de una instancia documentada en la literatura que incluye sólo figuras convexas. Se realizaron cuatro experimentos donde las instancias se dividen en conjuntos de entrenamiento y prueba. En cada experimento, se desarrolló una hiperheurística a partir de los problemas de entrenamiento y posteriormente se utilizó la hiperheurística generada para resolver las instancias de prueba. Se obtuvieron muy aceptables resultados, pues la hiperheurística desarrollada resolvió las instancias de prueba utilizando en promedio 1.38 objetos menos que el promedio de las 40 heurísticas simples consideradas.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

El problema de acomodar las piezas a cortar en objetos más grandes de materia prima para minimizar el desperdicio se conoce como el Problema de Corte y Empacado de Material de dos dimensiones. En general, el problema de corte y empacado de materiales en dos dimensiones tiene gran relevancia práctica; por ejemplo, en la industria textil, de lámina metálica, madera, piel, plástico y papel. La solución a este problema también tiene aplicaciones en desarrollo urbano, distribución de espacios en instalaciones y disposición de circuitos eléctricos [1].

El caso donde las piezas son rectangulares ha sido el más estudiado. Sin embargo, existen muchos entornos donde es necesario el corte o el empacado de figuras irregulares. Por ejemplo, en el corte de perfiles a partir de láminas metálicas, es necesario manejar arcos, concavidades y orificios [6]. Otro ejemplo se da en la manufactura de zapatos de piel, donde el costo de la materia prima es uno de los más importantes y las piezas a cortar son irregulares y no convexas. Aquí, además, se tienen restricciones tales como distintos grados de calidad en la piel y orientación en que las piezas deben ser cortadas [8]. En la industria textil, además de manejar irregularidad en las figuras a cortar, es importante tomar en cuenta el grabado en la tela que hace que muchas veces no se permitan giros en la pieza a cortar (o bien, se permiten 180º) [10].

Consideraciones prácticas hacen que las distintas industrias trabajen con distintas restricciones y distintos objetivos. En ciertos casos, aunque el objetivo principal sea minimizar el desperdicio, si se tiene un área grande de desperdicio, esta área puede reutilizarse. El tiempo empleado en cortar el material también podría ser algo de consideración [10].

En el problema de corte y empacado irregular de material de dos dimensiones se ha usado distinta terminología: en la industria de la construcción de barcos el término es 'anidamiento de partes' (*parts nesting*) mientras que en la industria textil se conoce como 'marcador de distribución' (*marker layout*) [10].

En comparación con el caso donde las figuras son solo rectángulos, existe poca teoría y técnicas en el caso de las figuras irregulares. Este problema de optimización, cuando las hojas y/o las piezas son polígonos irregulares, corresponde a la clase NP-duro [16]. Esto significa que todo problema NP puede reducirse a él y que no existe

un algoritmo polinómicamente acotado para encontrar su solución óptima. Es por esto que en la mayor parte de los casos se utilizan técnicas heurísticas con manejo eficiente de la geometría computacional para determinar la relación de piezas dentro de cada objeto y la posición que debe tener cada pieza dentro de cada objeto.

Una heurística (también llamada algoritmo heurístico o algoritmo de aproximación) es un algoritmo que no garantiza la mejor solución pero sí una cercana a la óptima en tiempo polinómicamente acotado. Una heurística es una regla práctica, casi siempre una idea que parece lógica, aunque no pueda demostrarse su bondad [2]. Existe un espectro amplio, desde heurísticas muy simples hasta heurísticas muy complejas que pueden ser muy difíciles de implementar. Para muchas aplicaciones prácticas lo que se requiere no es la solución óptima. A veces, ni siquiera es necesario que la solución está muy cerca de la óptima, sino una solución suficientemente buena, pero rápida y económica.

Por otra parte, algunas heurísticas encuentran una solución de buena calidad para un rango muy reducido de problemas y se comportan mal en el resto de los problemas. Actualmente existe una escuela de pensamiento en tecnología de búsqueda que sostiene que la tendencia en los próximos años es aumentar el nivel de generalidad en que las heurísticas y sistemas de optimización pueden operar [5].

1.1. Definición del problema

Dado un conjunto $L = (a_1, a_2, \ldots, a_n)$ de elementos a cortar, cada uno con tamaño $s(a_i) \in (0, A_0]$, y un conjunto de hojas (objetos) de tamaño A_0, el problema de corte y empacado de material consiste en encontrar patrones de corte dentro de los objetos de tal manera que se tenga un número mínimo de objetos necesarios para proveer todas las piezas pequeñas, con el fin de lograr el menor desperdicio de material posible. Un patrón de corte o solución posible es un acomodo de piezas tal que no existan empalmes y que ninguna pieza exceda los límites del objeto.

Debido a la gran diversidad de problemas y aplicaciones, Dyckhoff [13] en 1990 propuso una clasificación sistemática de problemas de corte y empacado. Su investigación integra 96 tipos de problemas de corte y empacado de material clasificados de acuerdo a cuatro características principales:

1. Dimensionalidad: Una (1), Dos (2), Tres (3) o n (N).

2. Tipo de asignación: Todos los objetos y una selección de piezas (B) o Una selección de objetos y todas las piezas (V).

3. Tipo de objetos: Un objeto (O), Figuras idénticas (I) o Diferentes figuras (D).

4. Tipo de piezas a cortar: Pocas piezas (de diferentes figuras) (P), Muchas piezas de muchas figuras diferentes (M), Muchas piezas de pocas figuras (R) o Figuras congruentes (C).

La complejidad del problema hace que el enfoque de esta investigación sea sólo hacia el problema de dos dimensiones (2), con material suficiente para satisfacer la demanda del total de piezas a recortar (V). Los objetos de donde se recortarán las piezas serán idénticos en forma rectangular (I) y las piezas a recortar serán polígonos irregulares convexos. Se tratarán casos con distintos números de piezas que incluyen mucha variedad de figuras distintas (M), ya que prácticamente todas las figuras a cortar serán diferentes. Es decir, de acuerdo a esta clasificación, la investigación se limita al problema **2VIM**.

En 2006, Wäscher [30] advierte sobre algunas deficiencias que a través de los años han aparecido con la clasificación de Dyckhoff [13] al tratar de manejar algunos desarrollos recientes, por lo que desarrolló y publicó una nueva clasificación de los problemas de corte y empacado. Aquí se utilizan los siguientes cinco criterios de clasificación:

1. Dimensionalidad: Una (1), Dos (2), Tres (3) o n (N). Este criterio se mantiene igual al de Dyckhoff [13].

2. Tipo de asignación: Todos los objetos y una selección de piezas (*Maximización de la salida*) o una selección de objetos y todas las piezas (*Minimización de la entrada*). Este criterio sólo cambia de nombre respecto a la clasificación de Dyckhoff [13].

3. Arreglo de piezas: Todas las piezas son idénticas en forma y tamaño (*piezas idénticas*), las piezas pueden clasificarse en relativamente pocos grupos de piezas idénticas en relación con el número total de piezas (*arreglo débilmente heterogéneo*) o sólo pocas piezas son idénticas en forma y tamaño unas de otras (*arreglo fuertemente heterogéneo*).

4. Arreglo de objetos: Un solo objeto. El cual a su vez puede clasificarse de acuerdo con la determinación de sus dimensiones:

   a) Todas sus dimensiones fijas.

   b) Una o más de sus dimensiones variables.

   O bien, varios objetos. Sólo se consideran dimensiones fijas y a su vez pueden clasificarse en :

   a) Objetos idénticos.

   b) Arreglo débilmente heterogéneo.

c) Arreglo fuertemente heterogéneo.

5. Forma de las piezas: Regulares (rectángulos, círculos, cajas, cilindros, esferas, etc.) e Irregulares.

De acuerdo a estos criterios de clasificación, la investigación se limita al problema de Dos dimensiones, Minimización de la entrada, Arreglo fuertemente heterogéneo de piezas, Objetos idénticos y Piezas Irregulares.

De acuerdo a Wäscher [30], al combinar las características de tipo de asignación y arreglo de piezas, surgen tipos *básicos* de problemas. El problema en que se enfoca la investigación, con características de minimización de la entrada y con arreglo fuertemente heterogéneo de piezas, corresponde al problema básico llamado **Problema de Empacado** (*Bin Packing Problem*). Además, de los tipos básicos de problemas se desprenden los tipos *intermedios* de problemas al agregar la característica de arreglo de objetos. Dado que el presente trabajo se enfoca al caso de objetos idénticos, le corresponde el tipo intermedio de problema llamado **Problema de Empacado de Objetos de un Solo Tamaño (SBSBPP)** (*Single Bin Size Bin Packing Problem*). Finalmente, para llegar al tipo *refinado* de problemas, se agregan los dos criterios restantes (dimensionalidad y forma de las piezas) al nombre del tipo intermedio de problema. Así, bajo la clasificación de Wäscher, la investigación se centra en el problema **SBSBPP irregular de 2 dimensiones**.

Las figuras irregulares a tratar son polígonos irregulares convexos que tendrán mínimo 3 y máximo 8 lados. Podrán rotar en su intento por acomodarse en algún objeto. La figura 1.1 ilustra el problema a tratar donde las piezas a cortar ocuparon dos objetos.

El problema de corte y empacado irregular de material en dos dimensiones ha sido atacado con muy diversos enfoques. Algunos trabajos proponen algoritmos de selección y/o acomodo de piezas mientras que otros desarrollan algoritmos de búsqueda local para la mejora de una solución encontrada:

- K. Dowsland y W. Dowsland [10] hacen una recopilación de enfoques adoptados en el problema. Aquí consideran figuras idénticas o muy parecidas y un solo objeto de longitud indefinida.

- Dowsland et al. [12] implementan una versión rápida y eficiente del algoritmo de acomodo llamado Fondo-Izquierda para polígonos.

- Dowsland et al. [11] describen una heurística de mejora que consiste en pequeñas repeticiones de la política de acomodo Fondo-Izquierda.

- Bennell y Dowsland [3] proponen una estrategia de búsqueda en ciertas vecindades para poder manejar el espacio de búsqueda infinito que presenta este tipo de

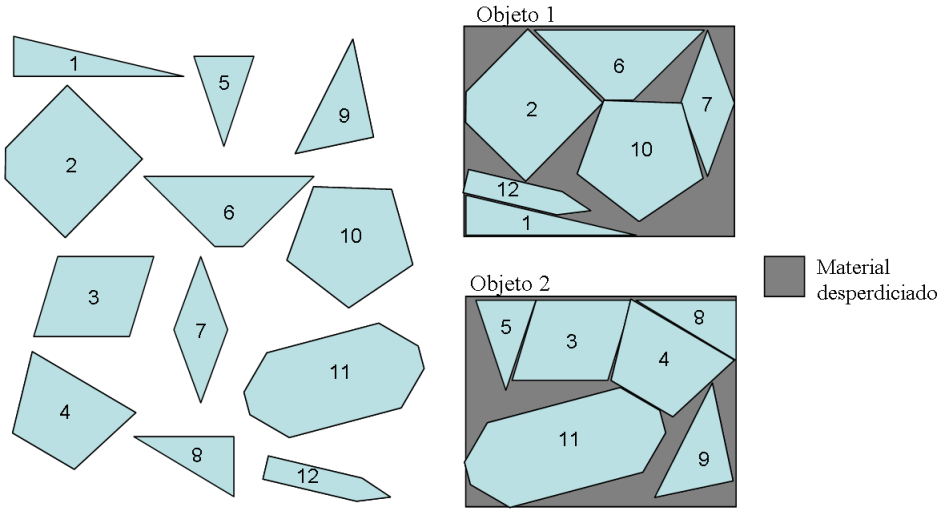

Figura 1.1: Esquema general del problema de corte y empacado irregular de material en dos dimensiones.

problemas. Para esto utilizan una búsqueda tabú híbrida que utiliza dos distintas rutinas de optimización.

- Burke et al. [6] propone un algoritmo de acomodo de piezas que permite acomodar piezas con arcos y orificios.
- Oliveira et al. [26] describen un algoritmo donde el arreglo de piezas se construye sucesivamente añadiendo piezas a la solución parcial. Aquí atacan el problema con un solo objeto de una dimensión variable. Prueban 126 variantes de un algoritmo al utilizar distintas funciones objetivos y criterios de selección de piezas.
- Hifi y M'Hallah [22] proponen un algoritmo completo para la solución del problema de corte y empacado irregular de material, que incluye una política de acomodo y resuelve mediante un algoritmo genético el ordenamiento de piezas.
- Hu-yao y Yuan-jun [24] desarrollan una heurística de acomodo que combina el principio de *No Fit Polygon* con el criterio de minimizar el centro de gravedad del conjunto de piezas conforme éstas se van colocando.

1.2. Motivación

El problema de corte y empacado de material en dos dimensiones tiene gran relevancia práctica cuando las figuras a cortar son irregulares. En trabajos recientes descritos por Terashima-Marín [28] y Farías [14] se refieren al problema de corte y empacado de material en dos dimensiones para el caso de figuras rectangulares. En estos trabajos,

se utiliza un algoritmo genético para desarrollar una hiperheurística cuyo desempeño es muy satisfactorio en relación a heurísticas simples. Dado el buen desempeño de la hiperheurística desarrollada para el problema de figuras rectangulares y que éste es un enfoque que aún no se había explorado para el caso de las figuras irregulares, se decidió que la presente investigación abordara el problema de corte y empacado irregular de material en dos dimensiones bajo ese enfoque.

1.3. Objetivos

El objetivo general de este trabajo es adecuar el método de generación de hiperheurísticas llamado GHH-2D por Terashima-Marín [28] y Farías [14] al Problema de Corte y Empacado Irregular de Material en dos dimensiones y confirmar, a través de experimentos, su buen funcionamiento en instancias con polígonos irregulares convexos. La hiperheurística GHH-2D está basada en un algoritmo genético que usa una representación de longitud variable que evoluciona combinaciones de reglas del tipo condición-acción. La hiperheurística GHH-2D ha probado ser eficaz para el problema de corte de figuras rectangulares. Según Burke et al. [5], una hiperheurística define una heurística de alto nivel que controla heurísticas de bajo nivel. La hiperheurística debe decidir cuándo y dónde aplicar cada heurística de bajo nivel, dependiendo del estado del problema.

Los objetivos particulares a cumplir en esta investigación son los siguientes:

- Seleccionar y generar heurísticas simples para el Problema de Corte y Empacado Irregular de Material en dos dimensiones.
- Adaptar la hiperheurística basada en un algoritmo genético (GHH-2D) al caso de los polígonos irregulares convexos, incluyendo tanto heurísticas de selección como de acomodo o posicionamiento.
- Evaluar el desempeño a través de problemas generados aleatoriamente para los cuales se conozca o no la solución óptima.
- Comparar los resultados de la hiperheurística respecto al desempeño de las heurísticas simples en las mismas instancias.

1.4. Alcances y suposiciones

Los alcances que tiene el presente trabajo de investigación son los siguientes:

- Desarrollar un Algoritmo Genético con cromosomas de longitud variable, capaz de evolucionar hiperheurísticas generales para resolver diferentes instancias del Problema de Corte y Empacado Irregular de Material en dos dimensiones. Específicamente el problema 2VIM (según la clasificación propuesta por Dyckhoff [13]) que equivale al problema SBSBPP irregular de 2 dimensiones (según la clasificación propuesta por Wäscher [30]).
- Utilizar piezas con mínimo 3 y máximo 8 lados.
- Utilizar piezas que sean polígonos convexos.
- Utilizar rotación en las piezas. El ángulo de rotación lo define la heurística de acomodo y se mide en el sentido contrario a las manecillas del reloj.
- Investigar y generar diferentes instancias del problema 2VIM para probar la efectividad del modelo propuesto.
- Evaluar de manera objetiva el desempeño del enfoque propuesto.

Las siguientes son las suposiciones bajo las cuales estuvo apoyado el desarrollo de la tesis:

- El código disponible que efectúa el algoritmo genético *messy* es correcto.
- El código disponible que efectúa las heurísticas simples de selección es correcto.

El código disponible fue utilizado en la investigación de Farías [14]. Además, la plataforma de desarrollo es el lenguaje Java.

1.5. Hipótesis

De acuerdo con Farías [14] y Terashima-Marín [28], la hiperheurística GHH-2D encuentra una combinación de heurísticas que igualan o mejoran el desempeño de cualquier heurística simple para el problema de empacado de piezas rectangulares. Por lo tanto, se espera que extender la misma estrategia de generación de hiperheurísticas para el caso de polígonos irregulares también provea buenos resultados.

Por lo tanto, se establecen las siguientes preguntas de investigación:

1. ¿Es posible desarrollar hiperheurísticas generales que proporcionen mejores resultados que las heurísticas simples para diferentes instancias del Problema de Corte y Empacado Irregular de Material en dos dimensiones?

2. Si es el caso, ¿cuál es un conjunto apropiado de heurísticas simples que sirven para desarrollar buenas hiperheurísticas generales al Problema de Corte y Empacado Irregular de Material en dos dimensiones?

3. ¿Cuál es la representación más adecuada del problema en el algoritmo genético?

4. ¿Cuáles parámetros del algoritmo genético funcionan mejor?

5. ¿El uso de la hiperheurística GHH-2D es una buena técnica para obtener una combinación de heurísticas que encuentren una buena solución?

1.6. Contribución

El aporte del presente trabajo de investigación consiste en el desarrollo de una herramienta para la creación de hiperheurísticas generales para resolver de forma eficiente una amplia variedad de instancias del problema de corte y empacado irregular de material en dos dimensiones, específicamente, el problema tipo 2VIM. Se espera que la hiperheurística general supere las limitaciones que las heurísticas simples presentan al ser utlizadas de forma individual.

Con base en las preguntas de investigación se busca aportar una contribución en los siguientes puntos:

- Implementación de nuevas variantes de heurísticas simples de acomodo para resolver instancias con piezas de forma irregular. Estas heurísticas simples tendrán la característica de ser rápidas en su ejecución.

- Encontrar la existencia de una combinación de heurísticas simples capaces de resolver de forma eficaz diferentes instancias del problema de corte y empacado irregular de material en dos dimensiones.

- La factibilidad del desarrollo de hiperheurísticas generales para la resolución de diversas instancias del problema de corte y empacado irregular de material.

- La efectividad del empleo de un algoritmo genético con cromosomas de longitud variable para encontrar las asociaciones entre las características de una instancia y determinadas heurísticas simples.

1.7. Organización de la tesis

La organización de esta tesis se presenta de la siguiente forma:

El capítulo 2 muestra los antecedentes vinculados con los problemas de optimización de corte y empacado de material, algoritmos genéticos, heurísticas e hiperheurísticas.

El capítulo 3 expone el modelo de solución propuesto, profundizando en temas específicos como la representación del estado del problema, las heurísticas simples utilizadas y el funcionamiento global del algoritmo genético.

El capítulo 4 explica la metodología de experimentación en donde se describen los problemas utilizados y los diferentes tipos de experimentos realizados.

El capítulo 5 muestra un análisis detallado de los resultados obtenidos en los diferentes tipos de experimentos desarrollados a lo largo de la investigación.

Finalmente, el capítulo 6 presenta las conclusiones y algunas sugerencias para trabajos futuros.

Capítulo 2

Antecedentes

Los problemas de corte y empacado de material irregular de 2-D son problemas combinatorios de muy alta complejidad, y no existe un algoritmo polinómicamente acotado para encontrar la solución óptima. Existen muchas aproximaciones que se utilizan cuando se quieren resolver estos problemas.

La Clase NP es el conjunto formado por todos los problemas de decisión para los cuales existe un algoritmo **N**o determinístico acotado **P**olinomialmente. Para estos problemas existe al menos un algoritmo para verificar si una solución propuesta cualquiera es efectivamente una solución válida al problema. Si la solución propuesta efectivamente se trata de una solución factible, entonces el algoritmo se efectúan en un tiempo que depende polinomialmente respecto al tamaño de la entrada. De este modo, para los problemas de la clase NP, es posible crear un algoritmo que *invente* una solución propuesta, la verifique y que al hacer esto un número grande de veces, logre tener una solución buena [2]. Un algoritmo genético es una forma de proponer (o *inventar*) soluciones de una manera ordenada y encaminada a proponer cada vez mejores soluciones.

Un problema es NP-duro si todo problema de la clase NP puede reducirse a él. Un problema NP-duro no necesariamente es problema NP. Si un problema NP-duro también está en NP, entonces es NP-completo. De acuerdo a Garey [16], el problema de empacado irregular de material en dos dimensiones es NP-duro. Esto significa que es un problema al menos tan difícil como los problemas NP.

Existen muchas aproximaciones (heurísticas) que se utilizan cuando se quieren resolver estos problemas. Cada una tiene sus fortalezas y limitaciones. Resulta natural pensar en la idea de combinar estas heurísticas simples para explotar sus fortalezas aplicando diferentes heurísticas en diferentes problemas o en diferentes estados del mismo problema. El estado de un problema se define como las características que tiene en esa etapa del proceso de solución, por ejemplo: las piezas que faltan por acomodar y las características de esas piezas.

Una hiperheurística puede verse como un conjunto de heurísticas simples además de un criterio que indica cuándo aplicar cada una.

Este capítulo detalla los principales antecedentes teóricos en que se soporta el

modelo propuesto de esta investigación. Esto incluye las heurísticas simples más importantes que han sido utilizadas en la solución del problema de corte y empacado de material. También, se describen los principales conceptos relacionados con un algoritmo genético, específicamente con el llamado algoritmo genético *messy*. Finalmente, se explica detalladamente el modelo GHH-2D, que se utiliza para construir hiperheurísticas con base en un algoritmo genético. El modelo GHH-2D constituye el principal trabajo de investigación relacionado con el modelo que se propone en la presente tesis.

2.1. Heurísticas simples para el problema de empacado de figuras irregulares

Varios enfoques se han adoptado para el problema de corte y empacado de figuras irregulares. Por ejemplo, es posible aproximar las partes usando el rectángulo que las limita y resolver el problema de corte y empacado de figuras rectangulares. Según Anand et al. [1], esto deriva, obviamente, en desperdicio de material. Wu et al. [31] discuten un enfoque sobre el anidamiento de piezas irregulares en rectángulos donde se permiten rotaciones. También ha sido común permitir configuraciones con empalmes y penalizar éstos en la función de evaluación. Esto tiene la limitación de permitir soluciones no factibles de acuerdo con Bennell y Dowsland [3].

Es típico que los enfoques para el corte y empacado de materiales presenten las siguientes fases:

1. **Heurísticas de selección.** Encontrar un ordenamiento óptimo de las piezas a cortar junto con el criterio para elegir el objeto donde se irán insertando las piezas.

2. **Heurísticas de acomodo.** Una vez que se tiene seleccionada la pieza y el objeto donde ésta entrará, la una heurística de acomodo indica la manera en que la pieza se acomodará dentro del objeto. Dado un mismo objeto y pieza dos heurísticas de acomodo distintas pueden llegar a conclusiones distintas sobre si la pieza puede o no ser acomodada en el objeto o sobre las coordenadas finales en que será instalada en el objeto.

3. **Heurísticas de búsqueda local.** Algunos enfoques consideran esta tercera fase. Una vez que tienen una solución generada por las dos primeras fases, realizan una búsqueda local alrededor de la solución encontrada para tratar de mejorarla.

En las siguientes secciones se describen las principales heurísticas existentes de cada una de las fases descritas.

2.1.1. Heurísticas de selección

La mayoría de las heurísticas de selección existentes supone que el número total de piezas que deberán ser cortadas es conocido desde el inicio, así como también las dimensiones de cada una de las piezas.

El orden en que las piezas serán seleccionadas puede hacerse desde un principio y mantenerse fijo o bien, puede ser dinámico: se elige una pieza y se acomoda y posteriormente se escoge la siguiente pieza de acuerdo con algún criterio. Además de seleccionar la pieza, es necesario algún criterio para seleccionar el objeto donde se colocará, que puede ser alguno de los objetos que ya empezaron a ocuparse, o bien, siempre está la posibilidad de iniciar un objeto nuevo.

A continuación se describen las heurísticas de selección encontradas en la literatura, las cuales también son descritas por Farías [14].

- **Heurística Next Fit (NF)**.- En esta heurística los piezas no se ordenan, sino que se toman en el orden en que están dadas en la representación del problema. Se abre un objeto y se colocan ahí las piezas en el orden en que aparecen en la lista. Si una pieza no puede acomodarse, se cierra ese objeto definitivamente y se abre uno nuevo donde se continúa el proceso de seguir colocando piezas. Esta heurística es bastante simple y tiene la ventaja de que, en ciertas implementaciones, permite que los objetos que ya fueron cerrados puedan empezar a cortarse inmediatamente, sin mantenerlos esperando con la esperanza de ser llenados nuevamente.

- **Heurística Next Fit Decreasing (NFD)**.- Este algoritmo es una variante de la heurística Next Fit. Sin embargo, aquí primero las piezas son ordenadas de mayor a menor área.

- **Heurística First Fit (FF)**.- Este algoritmo no tiene la restricción de acomodar las piezas únicamente en el último objeto abierto. En esta heurística se consideran los objetos parcialmente llenos como posibles destinos para que la pieza sea acomodada dentro de cualquiera de ellos. De esta manera todos los objetos que no han sido llenados totalmente permanecen abiertos con la esperanza de poder encontrar una pieza que minimice su desperdicio. Al igual que en la heurística Next Fit las piezas no se ordenan previamente. Con cada pieza a colocar se verifican los objetos en el orden en que éstos fueron abiertos hasta encontrar uno en donde quepa la pieza y si es necesario se abre un objeto nuevo.

- **Heurística First Fit Decreasing (FFD)**.- Este algoritmo es una variante del First Fit, en el cual primero las piezas son ordenadas de forma no creciente de acuerdo al área. La lista de piezas puede ordenarse en tiempo $\Theta(n \log n)$ (n es el

tamaño de la lista de piezas). El proceso de buscar un nuevo objeto para cada pieza se realiza a lo mucho $m(m-1)/2$ veces (m es el tamaño de la lista de objetos y $m \leq n$). El resto de las instrucciones se ejecutan n veces, de forma tal que la complejidad de este algoritmo en tiempo es de $\Theta(n^2)$, donde n es el número de piezas por acomodar. Aunque este algoritmo tiende a producir mejores soluciones que Next Fit, no siempre encuentra el número óptimo de objetos.

- **Heurística First Fit Increasing (FFI)**.- Esta heurística es una variante del First Fit Decreasing, en el cual primero las piezas son ordenadas de menor a mayor área.

- **Heurística Filler + FFD**.- Esta heurística es una combinación de la heurística Filler y la First Fit Decreasing. Dadas las piezas previamente ordenadas decrecientemente por área, el algoritmo Filler trata de acomodar la mayor cantidad de piezas posible en cualquiera de los objetos previamente abiertos, sin abrir ningún objeto nuevo. El proceso termina si se ha acomodado por lo menos una pieza, en caso contrario, se ejecuta el algoritmo First Fit Decreasing.

  Puede observarse que si solamente se utiliza la heurística First Fit Decreasing para acomodar todas las piezas de un problema, el resultado es el mismo que si se utilizara Filler + FFD como única heurística de selección para toda las piezas del problema.

- **Heurística Best Fit (BF)**.- Este algoritmo selecciona el objeto en el cual la pieza que se quiere acomodar proporcione el menor desperdicio posible, es decir el objeto en el cual la diferencia entre el área disponible del objeto y el área de la pieza sea menor, en caso de algún empate siempre se selecciona al objeto que fue primeramente abierto. Si la pieza no puede ser acomodada en ninguno de los objetos disponibles se abre un objeto nuevo. Al igual que en la heurística NF y FF las piezas no se ordenan previamente.

- **Heurística Best Fit Decreasing (BFD)**.- Se aplica la heurística Best Fit a las piezas previamente ordenadas de manera decreciente con respecto a su área.

- **Heurística Worst Fit (WF)**.- Heurística que trabaja de forma inversa a la Best Fit, ya que propone acomodar la pieza a_i en el objeto que proporcione el mayor desperdicio posible. En caso de empate se opta por el objeto que fue abierto primero y al igual que en el resto de los algoritmos, si la pieza no puede ser acomodada en ningún objeto disponible se abre uno nuevo.

- **Heurística de Djang y Finch (DJD)**.- Esta heurística llena objeto por objeto de la siguiente forma: Primero todas las piezas que se desean acomodar son ordenadas por área en orden decreciente y son colocadas en el contenedor hasta que

este ha sido llenado al menos a 1/3 de su capacidad. Posteriormente se inicializa un desperdicio w permitido, el cual inicialmente tiene un valor de cero ($w = 0$). De esta manera la heurística busca un objeto en el cual se acomode una pieza con un desperdicio w, si no lo encuentra entonces trata de acomodar dos piezas que dentro del objeto con un desperdicio w. Si esto no es posible, entonces busca la combinación de tres piezas que llenen al objeto obteniendo un desperdicio w. Si esto tampoco es posible, entonces se incrementa la variable $w = w + \delta$ y se repite el ciclo. El incremento mínimo δ puede ser de una unidad cuadrada de material tal como la proponen Djang y Finch. La tiempo de ejecución de este algoritmo es bastante superior al del resto de las heurísticas. El aumentar δ es una manera de reducir el tiempo de ejecución del algoritmo.

En las heurísticas anteriores, cuando se requiere que las piezas se ordenen, este orden siempre se hace respecto al área. Sin embargo, existen otros criterios respecto los cuales pueden ordenarse las piezas como son:

- De mayor a menor longitud.
- De mayor a menor anchura.
- De menor a mayor rectangularidad. La rectangularidad es la diferencia entre el área de la pieza y el área del rectángulo que la contiene (Gomes y Oliveira, [19]).
- De mayor a menor irregularidad. La irregularidad es medida como la diferencia entre el área de la pieza y el área de su respectivo casco convexo (Gomes y Oliveira, [19]). Este criterio para ordenar o clasificar piezas tiene sentido cuando se consideran piezas cóncavas.
- Área total (área $\times$ número de piezas a cortar) (Gomes y Oliveira, [19] y Oliveira et al. [26]). Cuando el tipo de problema incluye muchas piezas de cada tipo de pieza a cortar.
- Hifi y M'Hallah [22] proponen un algoritmo genético donde los individuos son distintas permutaciones de piezas. Dado que Hifi y M'Hallah [22] trabajan el problema donde se tiene un solo objeto (O) y se minimiza la longitud empleada en recortar todas las piezas, su función de evaluación es dicha longitud utilizada.

Algunas razones para estos criterios de ordenamiento se describen por Gomes y Oliveira [19]:

a) Es más fácil acomodar las piezas más grandes primero y los huecos que éstas dejen pueden ser llenados por piezas más pequeñas después.

b) La irregularidad juega un papel importante en la eficiencia de los algoritmos de corte y empacado de material, por lo que se prefiere acomodar primero las piezas más irregulares.

2.1.2. Heurísticas para acomodo

Una heurística de acomodo o posicionamiento convierte una secuencia de piezas en una distribución posible dentro de los objetos. En esta sección se describen las heurísticas de acomodo encontradas en la literatura que han sido aplicadas al problema de corte y empacado de figuras irregulares. Todas ellas son heurísticas de *una sola pasada*, ya que una vez colocada una pieza, ésta se considera fija y no se vuelve a revisar su lugar.

- **Heurística Fondo-Izquierda**.- Tal vez sea la heurística de acomodo más conocida. Cada pieza inicia en la esquina superior derecha y se desliza dentro del objeto la posición más al fondo posible y posteriormente se desliza lo más a la izquierda que es posible sin empalmarse con ninguna otra pieza ya colocada. Los movimientos hacia el fondo e izquierda se repiten sucesivamente hasta que ningún movimiento extra es posible (Ver figura 2.1). Si la posición final no excede los límites del objeto, la nueva pieza se fija en esa posición. Esta heurística no permite a una pieza *s*altar otra pieza ya colocada, incluso si existe un espacio suficientemente grande para colocarse. De este modo, el buen desempeño de esta heurística depende en gran medida del orden inicial de las piezas (Dowsland et al. [11, 12]). La ventaja de este procedimiento es su velocidad y simplicidad.

- **Heurística de Enfoque constructivo**.- Descrito por Hifi y M'Hallah [22] para el problema donde se tiene un solo objeto (O) y se minimiza la longitud empleada en recortar todas las piezas. Este enfoque utiliza operadores geométricos simples, evitando el cálculo de cascos convexos.

  El enfoque constructivo empieza colocando la primera pieza al fondo y a la izquierda del objeto. La siguiente pieza se coloca en una de cinco posibles posiciones: $(\bar{x}, 0), (0, \bar{y}), (\underline{x}, \bar{y}), (\bar{x}, \bar{y})$ y $(\bar{x}, \underline{y})$, donde $\bar{x}$, $\underline{x}$, $\bar{y}$, y $\underline{y}$ son las coordenadas mínimas y máximas en x y y respectivamente de la pieza ya colocada (Ver figura 2.2). Dado que algunas posiciones podrían coincidir, cada posición aparecen una sola vez en la lista. Para cada posición en la lista, la siguiente pieza a colocar se desplaza vertical y horizontalmente lo más al fondo y a la izquierda posible. La figura 2.3 muestra con un ejemplo cómo se prueba una de las posiciones en la lista colocando la figura en dicha posición como punto de referencia y posteriormente desplazándola al fondo e izquierda. Sólo las posiciones que no causen empalmes ni excedan las dimensiones del objeto se mantienen como posiciones candidatas. Finalmente, se elige la posición que coloque la pieza lo más al fondo

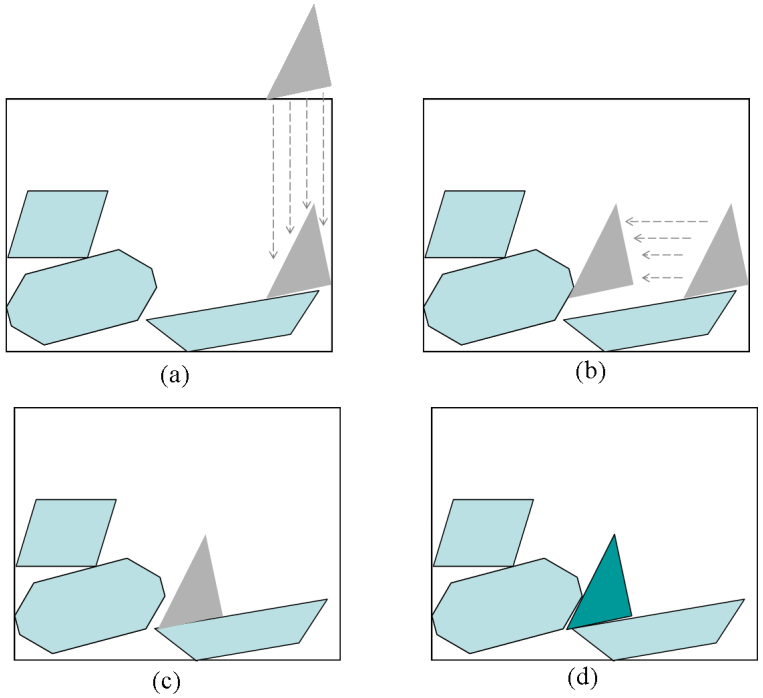

Figura 2.1: Procedimiento de la heurística fondo-izquierda.

y a la izquierda (excepto en ciertos casos, por ejemplo, si se forma un hoyo. Estos casos excepcionales los describen Hifi y M'Hallah [21]).

- **Heurística de Enfoque de máxima adyacencia**.- Sugerido por Uday et al. [29]. Esta heurística de acomodo primero selecciona el vértice del objeto donde será acomodada la siguiente pieza (ignorando ciertos 'puntos malos'). El vértice a seleccionar debe ser aquél con menor coordenada en y y en caso de empate, con menor coordenada en x. Una vez seleccionado el vértice, la pieza se acomoda en la rotación que le permita mayor adyacencia a la izquierda y abajo con las piezas ya acomodadas o con la orilla del objeto. Esta heurística también penaliza las posiciones que permiten mucha área sobrante a la izquierda y fondo del objeto.

- **Nueva heurística fondo-izquierda**.- Es un algoritmo propuesto por Burke et al. [6]. Permite el corte de piezas con arcos circulares y orificios. Dado una secuencia inicial de piezas a acomodar, acomoda cada una en el extremo inferior izquierdo de la hoja y al ver si se intersecta con otra pieza ya acomodada (o bien, alguna de las piezas contiene a la otra) elije una intersección entre dos de sus lados y desplaza la nueva pieza la menor distancia posible hacia arriba para evitar la intersección. Una vez que se han evitado todas las intersecciones, se continúa con la siguiente pieza. Si continuos empalmes desplazan la pieza hacia afuera de la parte de arriba de la hoja, la pieza se regresa a la base pero desplazada

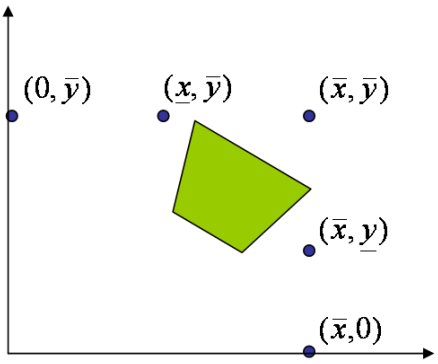

Figura 2.2: Heurística enfoque constructivo. Posiciones que se consideran por cada pieza colocada.

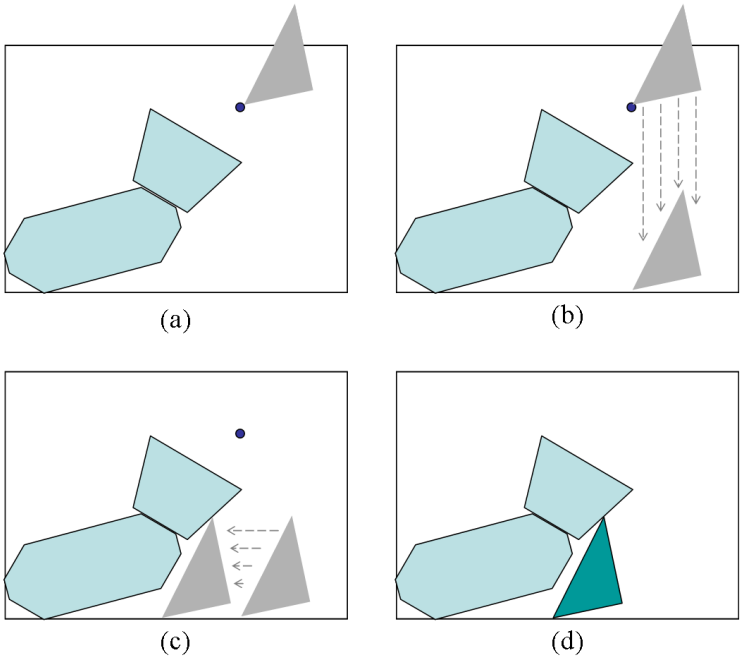

Figura 2.3: Procedimiento de la heurística enfoque constructivo.

a la derecha una cantidad positiva (llamada resolución). Este algoritmo también permite rotaciones de piezas. Burke et al. [6] utilizan este enfoque para el problema donde se tiene un solo objeto (O); sin embargo, sería adaptable para el problema con objetos idénticos (V).

- **Heurística No fit polygon (NFP)**.- Es un algoritmo muy popular. El *No Fit Polygon* (NFP) es la herramienta geométrica fundamental para el acomodo de piezas irregulares en 2D según Hu-yao y Yuan-jun [25]. Se menciona por primera vez en Art (1966). Se tiene un polígono P_1 considerado fijo, y un segundo polígono P_2 *móvil*. Se selecciona un vértice de referencia de P_2 y se desliza el polígono P_2 alrededor de P_1, manteniendo ambos polígonos su orientación inicial. El polígono que describe el vértice de referencia cuando P_2 se mueve alrededor de P_1 se conoce como el NFP y constituye el conjunto de todas las posibles posiciones del vértice de referencia de P_2 para que P_2 toque a P_1 sin traslaparse con él. Una vez construido el NFP, el vértice de referencia de P_2 se coloca en varios puntos del NFP y se elige una posición conveniente, obviamente, evitando los casos donde haya empalmes. Los dos polígonos juntos forman un nuevo polígono fijo y el siguiente polígono a acomodar es usado como polígono móvil.

  Entre los criterios para elegir la posición más conveniente del polígono a acomodar están:

  - Minimizar el área del casco convexo para los dos polígonos (el fijo y el móvil) [7]. Un algoritmo eficiente para calcular el casco convexo es presentado por Grinde y Cavalier [20].
  - Minimizar el área del rectángulo que circunscribe a los dos polígonos [26].
  - Minimizar la longitud del rectángulo que circunscribe a los dos polígonos [26].
  - Maximizar el empalme de los rectángulos que contienen a cada uno de los dos polígonos [26].

  BennellNFP et al. [4] y Hu-yao y Yuan-jun [25] detallan enfoques para generar el NFP.

El resto de las heurísticas para acomodo presentadas aquí también parten del cálculo del NFP.

- **Heurística basada en el principio del más bajo centro de gravedad**.- Descrito por Hu-yao y Yuan-jun [24], también necesita el cálculo del NFP. Primero, se encuentra el centro de gravedad de la pieza a colocar y se convierte en el punto de referencia para calcular el NFP. Varias rotaciones posibles generarían distintos

NFP. De todos ellos, se elige el punto más bajo para colocar la pieza. Según los autores, este enfoque tiende a buscar que las piezas se compacten más hacia abajo dentro del objeto.

- **Heurística de fondo-izquierda codicioso**.- Propuesta por Gomes y Oliveira [19]. Dado un objeto P y una serie de piezas P_i, $i = 1, \ldots, m$ ya colocadas en el objeto, la siguiente pieza P_k se coloca siguiendo la estrategia fondo-izquierda. Es decir, en la coordenada (x_k, y_k) con menor x y para igual x en el punto con menor y, tal que la pieza P_k quede totalmente dentro del objeto y no se empalme con ninguna pieza ya colocada. Para lograr esto, se usa No fit polygon (NFP) y el Inner-fit rectangle (IFR). El IFR se deriva del NFP y es el rectángulo que representa el conjunto de puntos donde puede colocarse el punto de referencia un polígono dentro de un objeto rectangular para que el polígono quede dentro del objeto. Con esta heurística P_k queda en contacto con los límites del objeto, o bien, en contacto con los límites del objeto y al menos una pieza ya colocada, o bien, con dos o más piezas ya colocadas.

- **Otra heurística de fondo-izquierda con varios NFP**.- Propuesta por Dowsland et al. [12]. Básicamente consiste en calcular el NFP para cada figura ya colocada, generando una lista de los vértices de los NFP como puntos candidatos para colocar la siguiente pieza. La pieza es colocada en el punto más al fondo-izquierda posible donde no haya empalmes y los límites de la figura no excedan los del objeto.

2.1.3. Heurísticas para búsqueda local

También llamadas heurísticas de mejora. Una vez que se ha obtenido una solución por alguna(s) heurística(s) de selección y de acomodo, es común que se hagan mejoras a la solución mediante un mecanismo de búsqueda local.

Algunas heurísticas aquí son:

- **Algoritmo del alpinista**.- Descrito por Burke et al. [6]. Se aplica cierto operador a la solución actual para hallar un *vecino*. Si puede hallarse un vecino que sea una mejor solución, éste reemplaza la solución anterior y la búsqueda continua.

- **Búsqueda Tabú**.- También descrito por Burke et al. [6]. El proceso genera un número fijo de vecinos (aplicando algún operador) y se mueve hacia la mejor solución en este subconjunto de soluciones. La mejor solución hasta entonces es utilizada para generar el siguiente conjunto de vecinos y el ciclo continúa. La lista tabú es una lista de longitud definida que evita revisar soluciones ya vistas.

Los operadores usados en las dos heurísticas anteriores consisten en remover aleatoriamente de 1 a N elementos e insertarlos en una posición arbitraria dentro de la secuencia inicial para generar nuevamente una solución por medio de la heurística de acomodo.

- **Búsqueda de intercambio**.- Propuesta por Gomes y Oliveira [19]. La heurística es responsable de mejorar la solución dentro del espacio de búsqueda. El algoritmo se mueve a otra solución del problema intercambiando dos piezas en la secuencia. El tamaño de la vecindad se controla al permitir intercambios de piezas que estén a una distancia máxima Δ. La búsqueda termina cuando es imposible moverse a una mejor solución.

2.2. Algoritmo genético

Un algoritmo genético es un algoritmo de optimización ciega que está inspirado en la naturaleza, específicamente en el fenómeno de la evolución natural. Los algoritmos genéticos pertenecen al grupo de métodos que se basan en simular el proceso evolutivo y que en conjunto se conocen como Computación Evolutiva.

La optimización ciega busca los parámetros o argumentos que maximizan (o minimizan) una función objetivo. El mecanismo puede verse como una *caja negra* donde se introducen parámetros y se devuelve un valor de evaluación de la función objetivo. No se asume ningún conocimiento sobre derivadas parciales, continuidad, número de dimensiones o número de extremos locales de la función objetivo.

Una descripción muy detallada de lo que es un algoritmo genético la hacen Holland [23] y Goldberg [17]. En un algoritmo genético, las soluciones propuestas a un problema o instancia se codifican en un vector de alfabeto reducido (generalmente binario). Estos vectores se conocen como individuos o cromosomas. Los elementos del cromosoma se conocen como bits o genes. Inicialmente se construye una población de estos individuos de manera aleatoria. De cada individuo se obtiene una evaluación también llamada aptitud. La aptitud juega el papel de la selección natural. Los individuos con mayor aptitud tendrán más posibilidades de sobrevivir, y por lo tanto, cruzarse y tener descendencia, por lo que parte de sus cromosomas estarán en las futuras generaciones. Además, al igual que en el proceso de evolución, algunos individuos sufrirán mutaciones, que son pequeñas variaciones en su cromosoma y que en algunas ocasiones pueden constituir una mejora. En resumen, un algoritmo genético evoluciona a los individuos de generación en generación a través de operadores genéticos de selección, cruce y mutación.

El ciclo que se realiza en un algoritmo genético simple se ilustra en la figura 2.4 y básicamente consta de los siguientes pasos:

1. Se genera una población inicial de manera aleatoria[1].

2. Selección de los mejores individuos de acuerdo a su aptitud formando una 'lista de selección'. Un individuo puede tener ninguna, una o varias copias en dicha lista en función de su aptitud.

3. Cruce de parejas permitiendo la recombinación de material genético.

4. Mutación de algunos individuos elegidos aleatoriamente.

   Al finalizar la selección, cruce y mutación se genera una nueva población y se regresa al paso 2, evolucionando la población de generación en generación hasta que se cumple con un criterio de terminación.

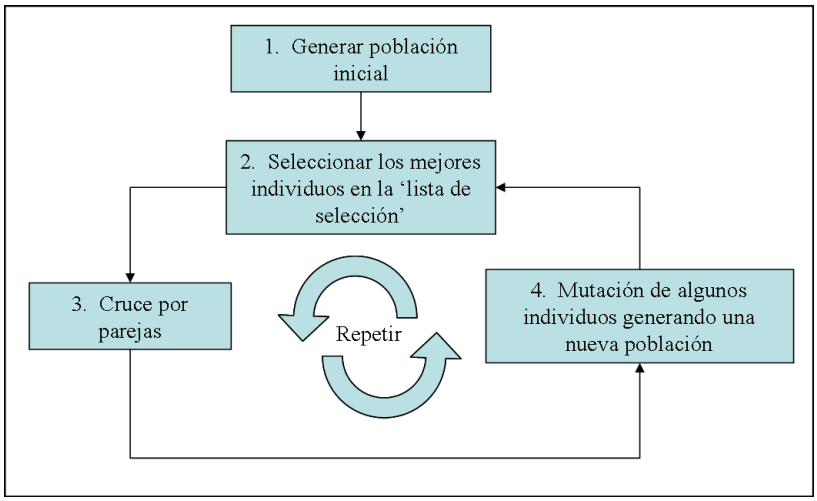

Figura 2.4: Ciclo básico de un algoritmo genético simple.

Los principales elementos y la terminología utilizada en un algoritmo genético básico se describen a continuación.

Población. Es el conjunto de individuos que representan soluciones propuestas en una generación dada. Entre mayor sea la población, se tendrá mayor número de soluciones propuestas por lo que se tendrá mayor oportunidad de tener una solución bastante bueno y/o cercana a la óptima. Sin embargo, el tamaño de la población impacta directamente en el trabajo computacional. Hay poblaciones que se renuevan casi totalmente de generación en generación (algoritmo generacional) mientras que hay algoritmos genéticos donde realmente no existe el concepto de generación pues los individuos permanecen más o menos tiempo en la población, lo que se conoce como estado estable

[1]Aunque por lo regular la población inicial es generada aleatoriamente, existe la posibilidad de generar individuos con ciertas características especiales.

o no generacional. En lugar de generaciones, en este caso puede usarse el término de ciclos.

Aptitud. Es el resultado de la evaluación de un individuo que se obtiene a través de la función de evaluación, la cual debe estar directamente relacionada con el problema a optimizar. Se considera que el tiempo de ejecución de la función de evaluación es el principal en un algoritmo genético.

Selección. Es el proceso mediante el cual los individuos con mayor aptitud son elegidos para pasar a la 'lista de selección'. Los individuos de esta lista se reproducen y sus hijos formarán la próxima generación. Hay varias maneras de construir la lista de selección, pero la idea básica es que mayor aptitud represente mayor posibilidad de ser seleccionado. Una manera natural de hacer la selección es darle a cada individuo una probabilidad para ser seleccionado proporcional a su aptitud. Una forma de selección proporcional con reemplazo es llamada **selección de rueda de ruleta**. Este procedimiento está influenciado por la escala en que esté dada la aptitud. Por ejemplo, si la aptitud de todos los individuos se incrementara en un número fijo, la probabilidad proporcional de ser seleccionado de acuerdo a su aptitud disminuiría para los individuos mejores al promedio mientras que aumentaría para los individuos de menor aptitud. Una segunda manera de efectuar la selección es otorgar a cada individuo la oportunidad de ser elegido proporcional al orden que ocupa en el la lista ordenada de mayor a menor aptitud (**selección por ranqueo**). Por otra parte, una de las maneras más populares de seleccionar a los mejores individuos es la llamada **selección por torneo**, en la cual primero se enlistan los individuos de manera aleatoria. Luego, compite cada posible grupo de m individuos consecutivos. El individuo con mayor aptitud en cada torneo obtiene una copia de él en la lista de selección. Tanto la selección por ranqueo como la selección por torneo son independientes de la escala en que esté dada la aptitud.

Cruce. Una vez seleccionados, los individuos tienen cierta probabilidad de reproducirse, mientras que también pueden pasar iguales a la siguiente generación. Generalmente el cruce se da entre dos individuos y produce dos hijos que son una combinación del material genético de sus padres. Entre los tipos de cruce más utilizados están 'cruce de un punto' y 'cruce de dos puntos'. En el cruce de un punto se selecciona aleatoriamente un punto en la longitud del cromosoma y el primer hijo se compone del material del primer padre desde el inicio hasta el punto de cruce y el resto lo toma del segundo padre, mientras que el segundo hijo es una copia del segundo padre desde el inicio hasta el punto de cruce y el resto lo toma del primer padre. El cruce de dos puntos es análogo: se seleccionan aleatoriamente dos puntos de cruce distintos. Uno de los hijos es igual al primer padre salvo por la parte entre los dos puntos de cruce, los cuales se toman del segundo padre y viceversa para el segundo hijo.

Mutación. Es usual que los individuos en un algoritmo genético sufran de mutación con alguna probabilidad pequeña. En un individuo se selecciona al azar uno de sus genes y se cambia por otro elemento posible de su codificación.

2.2.1. Algoritmo genético *messy*

Un algoritmo genético *messy* o desordenado (Goldberg et al. [18], Deb y Goldberg [9]), tiene cromosomas de distinta longitud y está sobreespecificado (con más de un valor para alguno de sus genes) o subespecificado (con algunos genes con valor ausente). De manera similar al proceso de evolución, cadenas pequeñas pueden combinarse para formar cadenas más largas y complejas que sean mejores propuestas de solución para el problema.

Un Algoritmo Genético *messy* posee las siguientes características que lo diferencian de un AG simple:

- Utiliza cromosomas de longitud variable que pueden estar subespecificadas y/o sobreespecificadas. Por ejemplo, en el caso del cromosoma de 3 bits: ((2 1) (3 0) (2 0)) existe sobreespecificación pues al gen 2 le corresponde el valor 1 y también el valor 0, mientras que también existe subespecificación pues el gen 1 está ausente. Así, ((1 0) (3 1) (2 0) (3 0)) y ((1 0) (3 1)) son codificaciones válidas para un cromosoma de longitud 3. En el primer caso existe sobreespecificación, mientras que en el segundo caso se presenta una subespecificación.

- Utiliza reglas de precedencia interna en las cadenas. La regla *primero que llega, primero que sale* es asociada al escaneo de cromosomas de izquierda a derecha como una forma de superar la sobreespecificación. De este modo, en el cromosoma ((2 1) (3 0) (2 0)), el gen 2 tomará el valor de 1 pues es el primer valor encontrado de izquierda a derecha.

- La subespecificación se maneja con el uso de formatos competetivos (*competitive templates*). Un formato competitivo es una especificación única para cada gen de un cromosoma, de modo que cuando el algoritmo se encuentra con un cromosoma subespecificado toma del formato los elementos correspondientes a los genes que falten de especificar.

- Utiliza los operadores de corte (*cut*) y unión (*splice*) en lugar del operador de cruce tradicional, los cuales trabajan en forma conjunta. El operador de corte divide en dos a un cromosoma con una probabilidad que está en función de la longitud del cromosoma (a mayor longitud, mayor probabilidad de corte). El operador de unión junta dos cadenas con cierta probabilidad fija. Un ejemplo del

uso del operador de unión es: ((2 0) (5 0)) y ((3 1) (6 0) (5 1)) → ((2 0) (5 0) (3 1) (6 0) (5 1)).

- La división del proceso evolutivo ocurre en dos fases: primordial y yuxtaposicional. La fase primordial inicializa la población que contenga todas las posibles cadenas de longitud determinada. Esta longitud es seleccionada de tal manera que abarque todos los posibles engaños de dicha longitud. Es decir, utiliza la inicialización parcial de las cadenas para evitar el engaño producido por la no-linealidad. En esta fase se ejecutan varias generaciones donde sólo se aplica la reproducción sin ningún otro operador genético y usualmente el tamaño de la población se reduce intencionalmente en determinados tiempos. La fase yuxtaposicional aplica la reproducción, corte, unión y otros operadores genéticos.
- Utiliza una población de tamaño variable, apropiada al tipo de fase evolutiva (primordial o yuxtaposicional).

2.3. Trabajo relacionado: Hiperheurística basada en un algoritmo genético (GHH-2D)

Es el trabajo principal en el cual está basada esta investigación y su desarrollo lo describen Terashima, H. [28] y Farías, C. [14]. A su vez, la hiperheurística GHH-2D toma características de la investigación de Ross et al. [27] cuyo enfoque fue en resolver el problema de corte y empacado de material en una dimensión usando un algoritmo genético (AG) con individuos de longitud variable.

Los métodos metaheurísticos usualmente trabajan directamente sobre el problema. Las hiperheurísticas trabajan en el proceso de elegir la heurística más adecuada para solucionar el problema. La idea de GHH-2D es descubrir la combinación de heurísticas de bajo nivel que presente un buen desempeño en un rango amplio de problemas.

El GHH-2D tiene el objetivo de encontrar una combinación de heurísticas de bajo nivel (de selección y acomodo) que resuelvan eficientemente una amplia variedad de instancias del problema de empacado irregular de material en 2-D. Para una evaluación imparcial de la hiperheurística generada, todas las instancias disponibles se dividen en dos conjuntos: el de entrenamiento y el de prueba.

El proceso general del modelo de solución consta de cuatro etapas (Ver figura 2.5):

1. Resolver todas los problemas, tanto de entrenamiento como de prueba, con cada una de las heurísticas simples. En esta fase se obtiene una aptitud de cada heurística simple que se calcula a través de las siguientes expresiones:

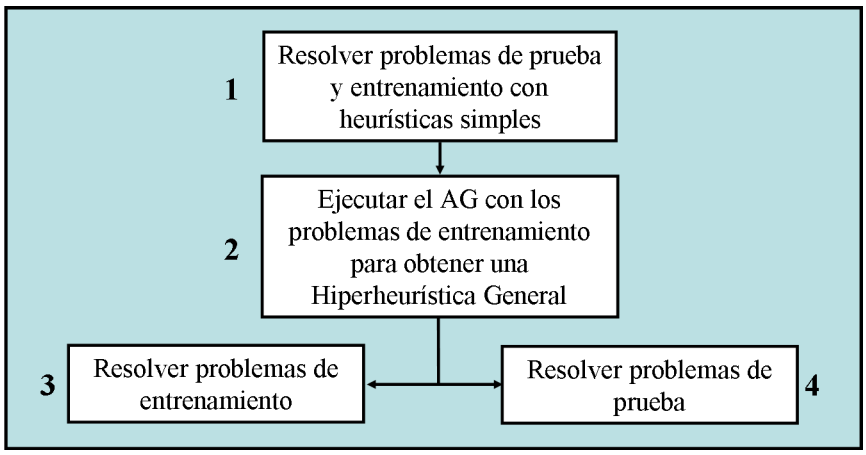

Figura 2.5: Modelo de solución del algoritmo GHH-2D.

$$P_u = \frac{\sum_{j=1}^{n} Ap_j}{A_o} \tag{2.1}$$

donde P_u representa el porcentaje de utilización de un objeto, Ap el área de cada pieza, A_o el área del objeto y n el número de piezas dentro del objeto.

Una vez que se ha calculado el porcentaje de utilización de cada objeto, cada heurística simple es evaluada mediante:

$$Apt = \frac{\sum_{u=1}^{No} P_u^2}{N_o} \tag{2.2}$$

donde N_o es el número total de objetos usados y P_u es el porcentaje de utilización para cada objeto u.

2. Solamente con los problemas de entrenamiento, correr el algoritmo genético.

   El algoritmo genético (AG) inicialmente crea una población aleatoria de individuos, que representan hiperheurísticas. Cada uno es evaluado y de acuerdo a la aptitud obtenida es seleccionado o no para dar origen a nuevos individuos mediante el proceso de cruce. El AG se ejecuta durante un número determinado de ciclos (o generaciones) al final de los cuales la población ha evolucionado y el mejor individuo de la población tiene una aptitud suficientemente buena.

   El proceso de entrenamiento del AG será explicado a detalle posteriormente, en la sección 2.3.1.

3. Una vez que el AG ha producido una hiperheurística con el conjunto de entrenamiento, cada problema de entrenamiento se resuelve con la hiperheurística.

4. Finalmente, con la hiperheurística generada también se resuelve el conjunto de problemas de prueba y los resultados se comparan con el desempeño de las heurísticas individuales y con el desempeño de la hiperheurística con el conjunto de entrenamiento.

La fase 3 y 4 pueden realizarse simultáneamente en cuanto a trabajo computacional se refiere y por otra parte, la fase 3 no es estrictamente necesaria ya que evaluar la hiperheurística generada con los mismos problemas de entrenamiento puede representar un sesgo al medir su desempeño.

2.3.1. Algoritmo genético propuesto por GHH-2D

El Algoritmo Genético (AG) utilizado es de estado estable. El proceso completo se resume en la figura 2.6 y es como sigue:

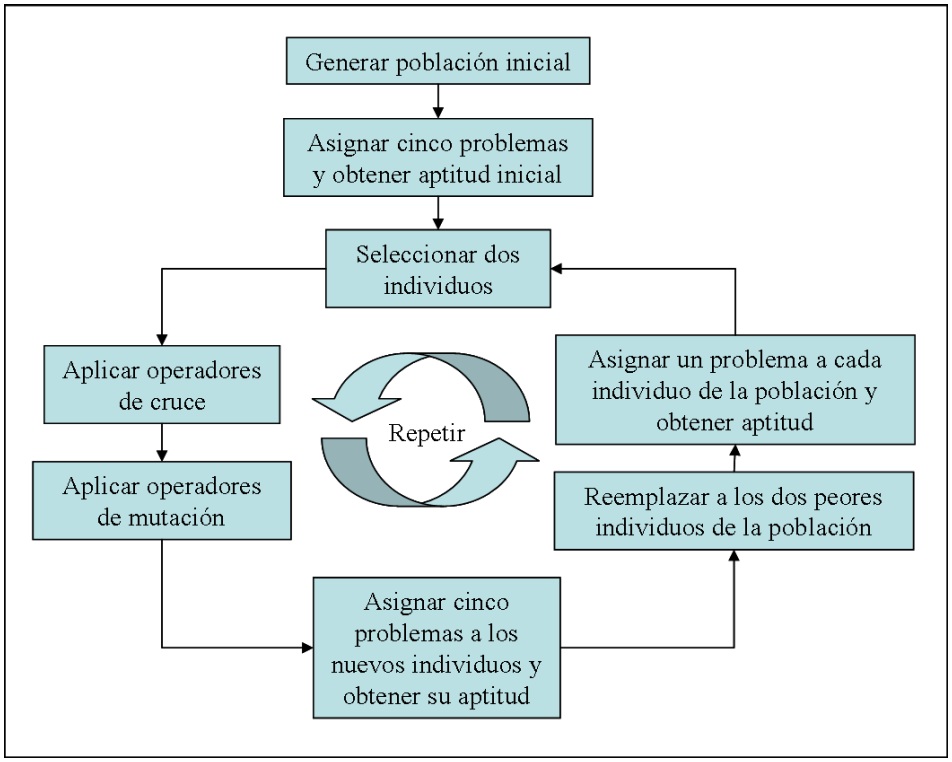

Figura 2.6: Proceso general del algoritmo genético *messy* en el modelo GHH-2D.

Generar población inicial. Inicialmente se generará una población inicial de forma aleatoria, a partir de la cual se llevará a cabo el proceso de evolución de los individuos (o hiperheurísticas).

Asignar cinco problemas. A cada individuo se asignan cinco individuos para que los resuelva y pueda obtenerse una aptitud inicial. Dado un problema P, su estado se representa con un conjunto de valores que describen el problema de una manera simplificada. En el caso del modelo GHH-2D (Farías [14]), el problema está definido por cinco números reales:

1. *Huge*. Este valor determina el porcentaje de piezas cuya área es mayor a un medio del área del objeto que faltan por acomodar.

2. *Large*. Este número representa la proporción de piezas por acomodar, con área mayor a un tercio y menor o igual a un medio del área total del objeto.

3. *Medium*. Simboliza la porción de piezas que faltan por acomodar cuya área es mayor a un cuarto y menor o igual a un tercio del área del objeto.

4. *Small*. Porcentaje de piezas que faltan por acomodar con área menor a un cuarto del área total del objeto.

5. *Remain*. Porcentaje del número total de piezas que faltan por acomodar. Este valor nos representa el grado de avance en el proceso global de solución.

El problema P es asociado con el punto más cercano en el cromosoma que indica la heurística de selección y de acomodo que será aplicada. La distancia entre el problema P y cada bloque del cromosoma se calcula como distancia euclidiana. Al aplicar la acción indicada por el bloque más cercano, se transforma el problema en uno nuevo P'.

Un cromosoma del AG es una secuencia de puntos dentro del espacio de estados en su representación simplificada. Los cromosomas son de longitud variable. Cada punto de la secuencia tiene una etiqueta que corresponde a una heurística. Es decir, cada par punto-heurística puede ser visto como una regla condición-acción que relaciona el punto del espacio de estados con la heurística (de selección y acomodo) que debe aplicarse. De este modo, el cromosoma constituye una receta para resolver un problema aplicando un algoritmo simple:

Mientras el problema no sea resuelto:

1. Calcular el estado actual del problema.

2. Encontrar en el cromosoma el punto más cercano.

3. Aplicar la heurística relacionada con el punto.

4. Actualizar el estado.

Obtener aptitud inicial. La aptitud de cada individuo se calcula como la distancia entre la calidad de la solución obtenida por el cromosoma y la calidad de la mejor solución obtenida por una heurística simple. En la primera generación se le asignan 5 problemas a cada individuo y su aptitud está dada por:

$$Apt(HH) = \frac{\sum_{k=1}^{5} P_k \left(Apt_k - MHS_k\right)}{\sum_{k=1}^{5} P_k} \tag{2.3}$$

donde P_k es el número de veces que la instancia k ha sido resuelta previamente. MHS_k es la mejor aptitud obtenida por las heurísticas simples para la instancia k; y Apt_k es la aptitud obtenida por la hiperheurística para la instancia k mediante la ecuación 2.2.

Seleccionar dos individuos. Con base en su aptitud, de la población se seleccionan dos individuos utilizando el método de selección por torneo de tamaño 2. Este método es un tipo de selección por ranqueo, es decir, se basa en el orden relativo de los individuos respecto a su aptitud. Actualmente es muy usado pues ha demostrado ser muy eficiente. La selección por torneo no necesita escalamiento y es fácil de implementar, además de mantener un buen equilibrio en la presión selectiva. La selección de torneo de tamaño dos se realiza de la siguiente forma:

1. Se revuelve la población aleatoriamente para evitar la posibilidad de que los individuos compitan siempre contra los mismos.

2. Cada individuo participa en dos torneos, uno contra el individuo anterior a él y el otro contra el individuo posterior a él de acuerdo al orden en que quedaron en el paso 1.

3. En cada torneo, al mejor de los individuos se le asigna una copia en la lista de selección. Finalmente, la lista de selección tiene dos réplicas del mejor individuo, cero réplicas del peor individuo y en valor esperado, tiene una réplica del individuo cuya aptitud es la mediana.

Aplicar operadores de cruce. El algoritmo genético propuesto por GHH-2D propone utilizar dos operadores de cruce:

1. Cruce simple:
   Este operador trabaja a nivel de bloques y es muy similar al cruce de un punto. Este operador intercambia el 10 % de bloques entre padres, es decir el hijo 1 estará formado por el 90 % del padre 1 y el 10 % del padre 2, de la misma forma el hijo 2 tendra el 90 % del padre 2 y el 10 % del padre 1. Los bloques que representen el 10 % de cada cromosoma serán elegidos aleatoriamente.

2. Cruce de dos puntos:
   Este operador de cruce es bastante parecido al cruce normal de dos puntos, para los AG de cromosomas de longitud fija donde el cruce de dos puntos utiliza los mismos dos puntos para ambos padres, con lo cual se mantiene la misma longitud de los cromosomas hijos. Para este caso donde se trabaja con cromosomas de longitud variable, cada padre elige los puntos de intercambio de manera independiente, con la única condición de que los puntos dentro de cada bloque deberán caer en la misma posición para ambos padres. Lo cual asegura que al final del proceso los bloques estarán completos.

Aplicar operadores de mutación. El algoritmo genético propuesto por GHH-2D contempla los siguientes tres esquemas de mutación:

1. Agregar bloque:
   Este operador de mutación se encarga de agregar un nuevo bloque al individuo elegido, para lo cual se genera un nuevo bloque de forma aleatoria y es agregado al cromosoma.

2. Remover bloque:
   Simplemente elige un bloque aleatoriamente y lo elimina del cromosoma.

3. Cambiar bloque:
   El tercer operador de mutación selecciona un bloque de forma aleatoria y dentro de este bloque elige una de las localidades y genera un nuevo valor válido para reemplazarlo.

Asignar 5 problemas a los dos nuevos individuos y obtener aptitud. Con el fin de evaluar la aptitud de los nuevos individuos creados a partir de los operadores de cruce y mutación, se les asignan cinco problemas del conjunto de entrenamiento de forma aleatoria a cada uno. La aptitud se mide mediante la ecuación 2.3.

Reemplazar a los dos peores individuos de la población. Una vez que se ha evaluado la aptitud de los nuevos individuos estos son introducidos dentro de la población ocupando el espacio de los dos peores individuos de la población, siempre y cuando la aptitud obtenida por los nuevos sea mejor que la de los peores.

Asignar un problema a cada uno de los individuos de la población y calcular una nueva aptitud. Después de cada ciclo l, un nuevo problema es asignado a cada individuo m dentro de la población y su aptitud es recalculada:

$$Apt_m^l = \frac{Apt_m^{l-1} mp_m + Apt(m)}{mp + 1} \tag{2.4}$$

donde Apt_m^{l-1} es la aptitud del individuo m en el ciclo $l-1$; mp_m es la cantidad de problemas que han sido resueltos por el individuo m y $Apt(m)$ es la aptitud obtenida por el individuo m para el nuevo problema.

Repetir el ciclo. El ciclo se repite iterativamente a partir de la selección de dos individuos mediante el método de torneo. De este modo, el AG evoluciona un conjunto de cromosomas para encontrar el que contenga las reglas que mejor solucionen cualquier estado intermedio en el proceso de solución de una instancia o problema. El mejor individuo generado por el AG en su último ciclo constituye la hiperheurística general creada por el modelo.

2.3.2. Resultados obtenidos por GHH-2D

En el trabajo realizado por Farías [14], se diseñan y realizan dos experimentos que generan una hiperheurística cada uno. Además, se realizan dos experimentos adicionales donde las mismas hiperheurísticas generadas son probadas en distintos grupos de problemas de figuras rectangulares. En total, las dos heurísticas son empleadas para resolver cuatro grupos de problemas. Los grupos de instancias fueron conformados por problemas de la literatura y por problemas generados aleatoriamente.

Se resolvieron todos los problemas con todas las heurísticas simples y también con las hiperheurísticas generadas, y se comparó el desempeño de las hiperheurísticas con el de la mejor heurística simple para cada una de las instancias. Las dos hiperheurísticas tuvieron un desempeño muy similar. En promedio, las hiperheurísticas generadas resuelven el 2.99 % de las instancias utilizando un objeto menos que la mejor heurística simple. En la mayoría de las instancias (79.92 %) el mismo número de objetos empleado por la hiperheurística es el mismo que el empleado por la mejor heurística simple. En el resto de las instancias, la hiperheurística requirió un objeto más que la solución de la mejor heurística simple.

2.4. Resumen

En este capítulo se expusieron los principales conceptos teóricos vinculados con la presente investigación. Se definió el problema de optimización de corte y empacado en dos dimensiones y las heurísticas principales que existen para abordarlo documentadas en la literatura. Se explicaron y definieron los conceptos relacionados con los algoritmos genéticos, así como también una variante específica conocida como algoritmo genético *messy*. Finalmente, se describió un trabajo estrechamente relacionado con la presente investigación.

Capítulo 3

Modelo de Solución

En este capítulo se describe el algoritmo propuesto para la generación de la hiperheurística, el cual está basado en el algoritmo GHH-2D descrito en el capítulo anterior (Terashima, H. [28] y Farías, C. [14]). Sin embargo, para adaptarlo al contexto de los polígonos convexos irregulares fueron necesarias importantes adecuaciones que se describen en las siguientes secciones. La introducción de la irregularidad impacta en las heurísticas simples y la representación de las instancias, de las acciones y de los estados del problema.

3.1. Modelo de solución propuesto

El proceso general propuesto en la presente investigación consta de tres etapas como lo muestra la figura 3.1.

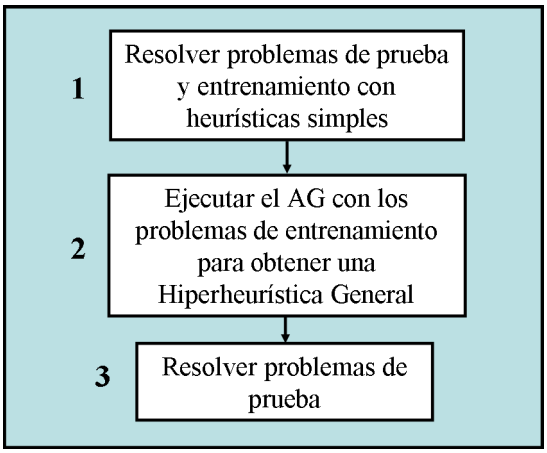

Figura 3.1: Modelo de solución propuesto.

La principal diferencia con el modelo del algoritmo GHH-2D es que se elimina la fase de prueba con las instancias de entrenamiento. La razón, fue explicada anteriormente y se relaciona con el sesgo que produce el obtener una evaluación de la

hiperheurística a partir de los mismos problemas de entrenamiento con los que fue generada.

3.2. Representación en el algoritmo genético

Un cromosoma del AG está compuesto de bloques, y cada bloque i contiene nueve números. Los primeros ocho caracterizan el estado del problema. El noveno número es un entero del 0 al 39 que indica la combinación de heurísticas de selección y acomodo que está asociada con esa instancia del problema en particular, como puede observarse en la figura 3.2.

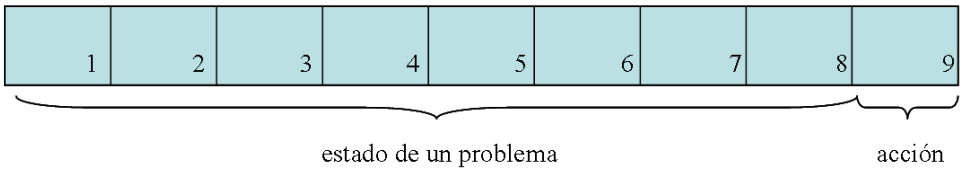

Figura 3.2: Representación de los bloques pertenecientes a los cromosomas en el AG.

Una descripción detallada de la representación del estado del problema en ocho valores se encuentra en la sección 3.3 y una explicación de las acciones se hace en la sección 3.4.

3.3. Representación de los estados

La representación del estado de un problema se establece mediante un vector de ocho números reales que se refieren a los siguentes aspectos:

- Rectangularidad. Definimos rectangularidad como el cociente del área de la pieza entre el área del rectángulo de lados horizontales y verticales de menor área que contiene a la pieza. Obsérvese que un rectángulo horizontal (o vertical) tiene rectangularidad igual a 1 y un triángulo rectángulo cuya base es horizontal (o vertical) tiene rectangularidad de 0.5. Por otra parte, un triángulo con un ángulo obtuso tiene rectangularidad menor a 0.5. Se mide en porcentaje promedio de rectangularidad de las de las piezas que faltan por acomodar.
- Área. Se mide en porcentaje de piezas que faltan por acomodar y que guardan cierta proporción respecto al área total del objeto.

- Altura. Se mide en porcentaje de piezas que faltan por acomodar y que guardan cierta proporción respecto la altura total del objeto.
- Piezas restantes. Se mide en porcentaje y este aspecto nos representa el grado de avance en el proceso global de solución.

Se realizó una prueba preliminar con un vector de 8 números reales. La representación que se probó fue el bloque compuesto por:

1. Rectangularidad alta. Porcentaje de piezas que tienen un valor de rectangularidad en el intervalo (0,9, 1].
2. Rectangularidad media. Porcentaje de piezas que tienen un valor de rectangularidad en el intervalo (0,5, 0,9].
3. Rectangularidad baja. Porcentaje de piezas que tienen un valor de rectangularidad en el intervalo (0, 0,5].
4. Piezas muy grandes. Porcentaje de piezas cuya área está en el intervalo (1/2 Área del Objeto, Área del Objeto].
5. Piezas grandes. Porcentaje de piezas cuya área está en el intervalo (1/3 Área del Objeto, 1/2 Área del Objeto].
6. Piezas medianas. Porcentaje de piezas cuya área está en el intervalo (1/4 Área del Objeto, 1/3 Área del Objeto].
7. Piezas pequeñas. Porcentaje de piezas cuya área está en el intervalo (0, 1/4 Área del Objeto].
8. Piezas restantes. Porcentaje del número total de piezas que faltan por acomodar.

En esta codificación, los primeros tres valores de la representación del problema están directamente relacionados con el manejo de piezas irregulares, mientras que los siguientes cuatro están relacionados con su área.

Finalmente se optó por la representación que se describe en el cuadro 3.1, la cual es una representación que ofrece más información utilizando los mismos 8 números reales, pues incluye porcentajes de piezas que tienen que ver además de la rectangularidad y el área, también con la altura de las piezas.

Los valores para definir los intervalos de área y altura para clasificar a las piezas se eligieron pensando en que existiera suficiente número de piezas en cada categoría en muchas de las instancias con las que se trabajaría en la experimentación.

La representación del estado de una instancia se utiliza para caracterizar un problema en las fases 2 y 3 del modelo de solución (ver figura 3.1) y decidir qué acción aplicar

Cuadro 3.1: Representación del estado del problema

Categoría	**Descripción**
Rectangularidad alta	Es el porcentaje de piezas que tiene un valor de rectangularidad mayor a 0.9.
Rectangularidad media	Es el porcentaje de piezas cuyo valor de rectangularidad es mayor a 0.5 pero menor o igual a 0.9.
Rectangularidad baja	Es el porcentaje de piezas cuyo valor de rectangularidad es menor o igual a 0.5.
Piezas grandes	Porcentaje de piezas por acomodar con área mayor a 1/4 del área del objeto.
Piezas pequeñas	Simboliza el porcentaje de piezas que faltan por acomodar cuya área es mayor a 1/10 y menor o igual a 1/4 del área del objeto.
Piezas altas	Porcentaje de piezas por acomodar con altura mayor a 1/2 de la altura del objeto.
Piezas cortas	Porcentaje de piezas por acomodar con altura mayor a 1/4 y menor o igual a 1/2 de la altura del objeto.
Piezas restantes	Porcentaje del número total de piezas que faltan por acomodar.

al ubicar el bloque del cromosoma más cercano al estado del problema. La distancia entre el vector que representa el estado del problema y cada bloque del cromosoma se calcula como distancia euclidiana.

3.4. Representación de las acciones

Diez heurísticas de selección con cada una de cuatro heurísticas de acomodo forman 40 posibles heurísticas simples para solucionar el problema de empacado de material en dos dimensiones, que se enlistan en el cuadro 3.2.

La descripción detallada de las heurísticas se encuentra en la sección 3.7.

3.5. Función objetivo

En la primera generación se le asignan 5 problemas a cada individuo (hiperheurística) y su aptitud está dada por:

Cuadro 3.2: Representación de acciones.

Accion	Heurística de Selección	Heurística de Acomodo
0	First Fit (FF)	Fondo Izquierda. (FI).
1		Enfoque Constructivo. (EC).
2		Enfoque Constructivo de Mínima Área (ECMA).
3		Máxima Adyacencia. (MA).
4	First Fit Decreasing (FFD)	Fondo Izquierda. (FI).
5		Enfoque Constructivo. (EC).
6		Enfoque Constructivo de Mínima Área (ECMA).
7		Máxima Adyacencia. (MA).
8	First Fit Increasing (FFI)	Fondo Izquierda. (FI).
9		Enfoque Constructivo. (EC).
10		Enfoque Constructivo de Mínima Área (ECMA).
11		Máxima Adyacencia. (MA).
12	Filler + FFD	Fondo Izquierda. (FI).
13		Enfoque Constructivo. (EC).
14		Enfoque Constructivo de Mínima Área (ECMA).
15		Máxima Adyacencia. (MA).
16	Next Fit (NF)	Fondo Izquierda. (FI).
17		Enfoque Constructivo. (EC).
18		Enfoque Constructivo de Mínima Área (ECMA).
19		Máxima Adyacencia. (MA).
20	Next Fit Decreasing (NFD).	Fondo Izquierda. (FI).
21		Enfoque Constructivo. (EC).
22		Enfoque Constructivo de Mínima Área (ECMA).
23		Máxima Adyacencia. (MA).
24	Best Fit (BF)	Fondo Izquierda. (FI).
25		Enfoque Constructivo. (EC).
26		Enfoque Constructivo de Mínima Área (ECMA).
27		Máxima Adyacencia. (MA).
28	Best Fit Decreasing (BFD)	Fondo Izquierda. (FI).
29		Enfoque Constructivo. (EC).
30		Enfoque Constructivo de Mínima Área (ECMA).
31		Máxima Adyacencia. (MA).
32	Worst Fit (WF)	Fondo Izquierda. (FI).
33		Enfoque Constructivo. (EC).
34		Enfoque Constructivo de Mínima Área (ECMA).
35		Máxima Adyacencia. (MA).
36	Djang and Finch (DJD)	Fondo Izquierda. (FI).
37		Enfoque Constructivo. (EC).
38		Enfoque Constructivo de Mínima Área (ECMA).
39		Máxima Adyacencia. (MA).

$$Apt(HH) = \frac{\sum_{k=1}^{5} P_k \left(Apt_k - MHS_k\right)}{\sum_{k=1}^{5} P_k} \tag{3.1}$$

donde P_k es el número de veces que la instancia k ha sido resuelta previamente. MHS_k es la mejor aptitud obtenida por las heurísticas simples para la instancia k; y Apt_k es la aptitud obtenida por la hiperheurística para la instancia k mediante la ecuación 2.2.

En los siguientes ciclos, se le asigna una instancia más a cada individuo y su aptitud se actualiza mediante la expresión:

$$Apt_m^l = \frac{Apt_m^{l-1} mp_m + Apt(m)}{mp + 1} \tag{3.2}$$

donde Apt_m^{l-1} es la aptitud del individuo m en el ciclo $l-1$; mp_m es la cantidad de problemas que han sido resueltos por el individuo m y $Apt(m)$ es la aptitud obtenida por el individuo m para el nuevo problema.

La aptitud de cada individuo representa la distancia entre la calidad de la solución obtenida por el cromosoma y la calidad de la mejor solución obtenida por las heurísticas simples. El objetivo es ir mejorando la aptitud de los individuos ciclo tras ciclo por lo que mayor aptitud de un individuo representa una mayor probabilidad de ser seleccionado para reproducirse (Ver sección 2.3.1).

3.6. Representación de las instancias

La representación de las instancias de entrada para el algoritmo propuesto consiste en un renglón donde se especifica el número de piezas; un segundo renglón donde se indican las dimensiones de los objetos. Del tercer renglón en adelante se representan las piezas a través del número de vértices del que constan y sus coordenadas (Ver figura 3.3). Las coordenadas están en relación a un punto de referencia, el cual se define como la esquina inferior izquierda del rectángulo que contiene a la pieza. El rectángulo que contiene a la pieza debe tener sólo lados horizontales y verticales (Ver figura 3.4).

3.7. Heurísticas simples implementadas

Las heurísticas simples representan las acciones en cada bloque del AG del modelo, es decir el último elemento de cada bloque. Cada heurística simple se conforma de una heurística de selección y una heurística de acomodo, que al aplicarlas consecutivamente representan un algoritmo para solucionar un problema.

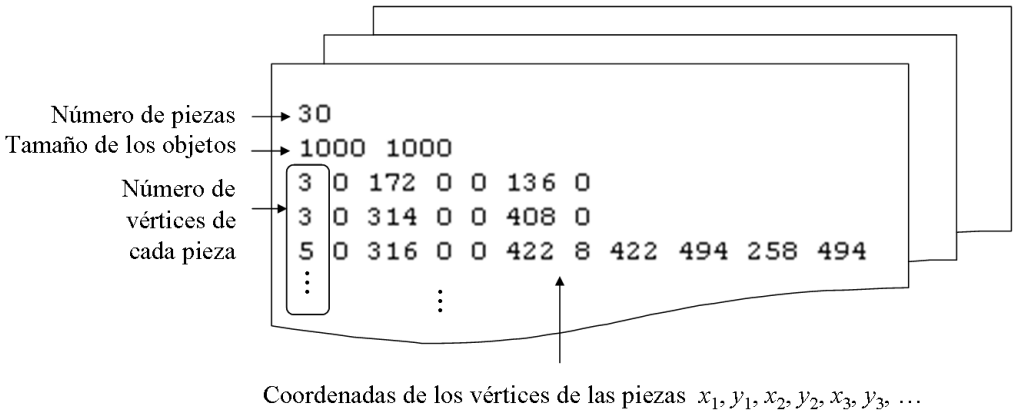

Figura 3.3: Representación de una instancia en un archivo de texto.

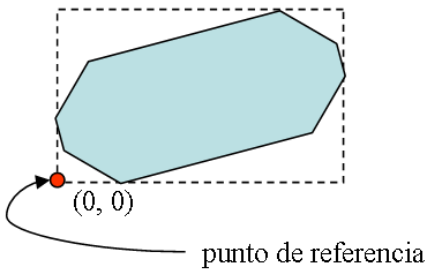

Figura 3.4: Punto de referencia para indicar las coordenadas de cada pieza en la representación de las instancias.

3.7.1. Heurísticas de selección implementadas

Las heurísticas de selección implementadas en el algoritmo propuesto son aquellas que se describieron en el capítulo 2. Aquí se enlistan nuevamente. Algunas de ellas no fueron implementadas tal cual se encuentran en la literatura, por lo que se describen las modificaciones que se efectuaron.

1. **Heurística Next Fit (NF)**. Las piezas se toman en el orden en que están dadas en la representación del problema, el cual es aleatorio. La variante a esta heurística que se implementó en el presente trabajo es: La primera pieza trata de acomodarse en el último objeto abierto, si cabe, ahí mismo se intentan acomodar *todas* las piezas que quepan, de modo que si una pieza no cabe, se hace a un lado y se sigue intentando acomodar las demás. Si la primera pieza no cabe en el último objeto abierto, se clausura ese objeto, se abre un objeto nuevo y nuevamente, se intentan acomodar ahí todas las piezas que quepan.

2. **Heurística Next Fit Decreasing (NFD)**.

3. **Heurística First Fit (FF)**. En la implementación del presente trabajo se tiene una variante similar a la de la heurística NF: Se busca el objeto donde la primera pieza de la lista se acomodará (puede ser un objeto ya abierto o uno nuevo). En él se intentarán acomodar *todas* las piezas de la lista; de modo que si una pieza de la lista no cabe, se hace a un lado y se sigue intentando acomodar las demás.

4. **Heurística First Fit Decreasing (FFD)**.

5. **Heurística First Fit Increasing (FFI)**.

6. **Heurística Filler + FFD**.

7. **Heurística Best Fit (BF)**.

8. **Heurística Best Fit Decreasing (BFD)**.

9. **Heurística Worst Fit (WF)**.

10. **Heurística de Djang y Finch (DJD)**. Djang y Finch proponen un incremento $\delta = 1$ en el desperdicio permitido en cada iteración. No obstante, en esta implementación se fijó $\delta = 2000$ por dos razones:

    - Las áreas de las piezas manejadas tienen un órden de magnitud de varias decenas de miles de unidades cuadradas.
    - Se probaron varios incrementos δ observándose que incrementos muy pequeños hacen mínima –o nula– la ganancia en la calidad de las soluciones y sí restan mucha eficiencia al trabajo computacional.

3.7.2. Heurísticas de acomodo implementadas

No todas las heurísticas de acomodo documentadas en la literatura y descritas en el capítulo anterior son implementadas en el presente trabajo. Básicamente por razones de complejidad, ya que se desea lograr que la hiperheurística combine fortalezas de heurísticas relativamente simples para lograr soluciones buenas para una gran variedad de instancias.

Las heurísticas de acomodo utilizadas en esta investigación son:

1. **Heurística Fondo-Izquierda**. Se implementó tal como se describe en el capítulo anterior.

2. **Heurística de Enfoque Constructivo**. A esta heurística se aplicó una variante: además de las 5 posibles posiciones para cada pieza ya colocada que indican Hifi y M'Hallah [22], se incluyen las cuatro esquinas del objeto. Un ejemplo de los puntos a considerar con dos piezas ya colocadas puede verse en la figura 3.5. Al utilizar todas las esquinas como puntos de partida para deslizarse al fondo y a la izquierda es posible accesar a ciertos huecos entre piezas, que no podrían accesarse con el resto de las posiciones consideradas.

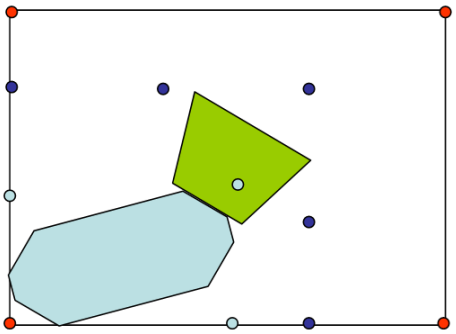

Figura 3.5: Puntos a considerar en la heurística de enfoque constructivo. Un ejemplo con dos piezas ya colocadas.

3. **Heurística de enfoque constructivo de mínima área fondo-izquierda**. Adaptación propia de la heurística de enfoque constructivo. La diferencia con la heurística anterior radica en el criterio utilizado para elegir la mejor posición de la lista. En este caso se elige la posición que forma el rectángulo de menor área que contiene todas las piezas y que puede colocarse en la esquina inferior izquierda del objeto. Dicho rectángulo se ilustra en la figura 3.6 y su área se calcula con el producto de la máxima coordenada horizontal y la máxima coordenada vertical de todas las piezas ya colocadas más la pieza a colocar en la posición propuesta. La figura 3.7 muestra un ejemplo donde se ilustra el rectángulo de mínima área fondo-izquierda para dos distintas posiciones propuestas para una pieza. La razón para implementar este último criterio se basa en la idea de elegir un punto que, en general, deje todas las piezas lo más al fondo e izquierda posible y no sólo la última pieza.

4. **Heurística de enfoque constructivo de máxima adyacencia**. La idea detrás de la implementación de esta heurística se basa en el enfoque sugerido por Uday et al. [29]. Pero en esta investigación, cuando la primera pieza va a colocarse solamente se exploran los cuatro vértices del objeto. Para las siguientes piezas, los *puntos posibles* donde la siguiente pieza podría ser colocada son los mismos

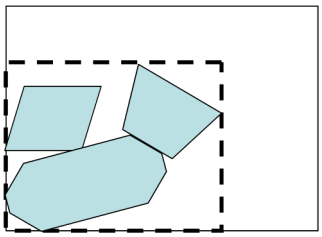

Figura 3.6: Rectángulo de menor área en la esquina inferior izquierda del objeto que contiene todas las piezas.

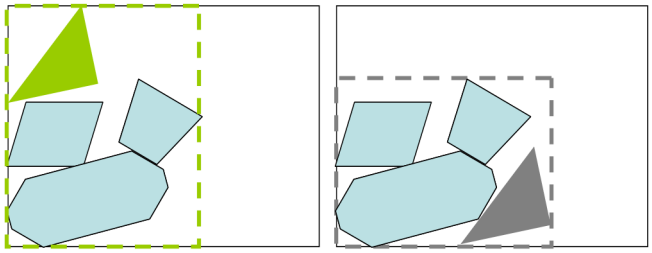

Figura 3.7: Rectángulo de menor área en la esquina inferior izquierda del objeto para dos posiciones propuestas para una pieza.

que para las heurísticas de enfoque constructivo. Cada punto de la lista se evalúa dos veces: Primero se coloca la pieza en el punto y se calcula la longitud total de su perímetro en que está adyacente con las piezas ya colocadas y con las aristas del objeto. Después, se desliza lo más al fondo e izquierda posible y vuelve a calcularse la adyacencia. La posición con mayor adyacencia de todas las evaluadas se convierte en la posición de la pieza nueva.

Programación de operadores geométricos

Para implementar las heurísticas de acomodo descritas fue necesario programar algunos operadores geométricos para realizar determinados cálculos. Se describen a continuación, donde P_1 es Pieza 1 con v_1 vértices y P_2 es Pieza 2 con v_2 vértices.

1. Calcular qué tanto pueden aproximarse P_1 hacia P_2 horizontalmente hacia la izquierda (o verticalmente hacia abajo) para quedar adyacentes y no empalmarse. Esto se hace a través del siguiente algoritmo:

   a) Para cada vértice de P_1, calcular la distancia horizontal hacia la izquierda (o vertical hacia abajo) hacia todos los lados de P_2.

b) Para cada vértice de P_2, calcular la distancia horizontal hacia la izquierda (o vertical hacia abajo) hacia todos los lados de P_1.

c) El máximo desplazamiento será la menor de las distancias calculadas.

d) En caso de que P_1 no pueda accesar a P_2, el desplazamiento será ∞ teóricamente.

Este algoritmo se ejecuta en un tiempo de orden polinomial respecto producto $v_1 \times v_2$ y es la base para calcular el máximo desplazamiento horizontal (o vertical) de una pieza dentro de un objeto sin tener ningún empalme con ninguna pieza dentro del objeto o con los los mismos límites del objeto.

2. Calcular el perímetro que dos piezas son adyacentes entre sí.

   a) Para cada lado de P_1, se calcula el segmento que coincide con cada uno de los lados de P_2.

   b) El perímetro P_1 y P_2 son adyacentes entre sí es la suma de los cáculos anteriores.

   Este operador geométrico es básico en la heurística de máxima adyacencia.

3. Decidir si dos piezas dadas P_1 y P_2 se intersectan o no. El algoritmo implementado para esto contiene los siguientes pasos:

   a) Para cada lado de P_1, evaluar si se intersecta con cada lado de P_2.

   b) Esta decisión termina en cuanto se encuentra un lado de una pieza que intersecta con el lado de la otra pieza.

4. Decidir si, dadas dos piezas P_1 y P_2, una está dentro de la otra. Los siguientes pasos componen el algoritmo implementado:

   a) De cada vértice de P_1 se construye un rayo horizontal que sale del vértice indefinidamente hacia la derecha.

   b) Si alguno de los rayos construidos corta sólo una vez a P_2, entonces se decide que P_1 está dentro de P_2. Si el rayo pasa exactamente por un extremo de uno de los lados no se considera un corte. En caso de tener figuras cóncavas la regla de decisión sería: Si alguno de los rayos construidos corta un número *impar* de veces a P_2, entonces se decide que P_1 está dentro de P_2.

5. Decidir si, dadas dos piezas P_1 y P_2, éstas son iguales. Debido a la representación de las piezas, éstas podrían ser iguales pero tener sus coordenadas codificadas en distinto orden, por lo que se implementó el siguiente algoritmo:

a) Si $v_1 \neq v_2$ se decide que P_1 y P_2 no son iguales.

b) Si alguno de los extremos derecho, izquierdo, inferior o superior de ambas piezas no coincide, se decide que P_1 y P_2 no son iguales.

c) Para cada vértice de P_1 se cuenta cuántas veces está incluido en los lados de P_2. Si cada vértice de P_1 está exactamente en dos de los lados de P_2 se concluye que cada vértice de P_1 coincide con cada vértice de P_2 y por lo tanto se decide que las piezas son iguales.

Para verificar que una posición sea válida para una pieza, es necesario validar que no intersecta, ni está dentro, ni coincide exactamente con ninguna otra pieza dentro del objeto, para lo cual, los últimos tres operadores geométricos son útiles.

Rotación

Cuando se utiliza alguna de las siguientes heurísticas de acomodo:

- Enfoque constructivo
- Enfoque constructivo de mínima área fondo-izquierda
- Enfoque constructivo de máxima adyacencia

El algoritmo de cada heurística prueba cada pieza en varias rotaciones en cada posición posible de la lista de posiciones.

La rotación de una pieza un ángulo θ se consigue mediante el siguiente algoritmo, cuya entrada son las coordenadas de los vértices de la pieza y el ángulo de giro deseado. La salida del algoritmo son las coordenadas de los vértices de la pieza ya girada.

1. Guardar en variables temporales la coordenadas horizontal y vértical mínimas de la pieza. Estas coordenadas mínimas serán las mismas después de rotar la pieza.

$$\begin{bmatrix} x_{\text{mín}} \\ y_{\text{mín}} \end{bmatrix} = \begin{bmatrix} \text{mín}\,(x_1, x_2, \ldots, x_n) \\ \text{mín}\,(y_1, y_2, \ldots, y_n) \end{bmatrix} \tag{3.3}$$

2. Realizar una transformación en el vector de coordenadas $\begin{bmatrix} x_i & y_i \end{bmatrix}^t$ de cada vértice de la pieza a rotar:

$$\begin{bmatrix} x_i^* \\ y_i^* \end{bmatrix} = \begin{bmatrix} x_i & y_i \end{bmatrix} \begin{bmatrix} \cos\theta & \text{sen}\theta \\ -\text{sen}\theta & \cos\theta \end{bmatrix} \tag{3.4}$$

Al hacer esto, las coordenadas de la pieza girada rotan también respecto al origen de su sistema de coordenadas como se muestra en la figura 3.8.

3. Obtener las coordenadas mínimas en x y y de la pieza rotada:

$$\begin{bmatrix} x^*_{\text{mín}} \\ y^*_{\text{mín}} \end{bmatrix} = \begin{bmatrix} \text{mín}\,(x^*_1, x^*_2, \ldots, x^*_n) \\ \text{mín}\,(y^*_1, y^*_2, \ldots, y^*_n) \end{bmatrix} \tag{3.5}$$

4. Obtener el vector de desplazamiento:

$$\text{Vector\_desplazamiento} = \begin{bmatrix} x_{\text{mín}} \\ y_{\text{mín}} \end{bmatrix} - \begin{bmatrix} x^*_{\text{mín}} \\ y^*_{\text{mín}} \end{bmatrix} \tag{3.6}$$

5. Actualizar cada par de coordenadas de la pieza girada como sigue:

$$\begin{bmatrix} x^*_i \\ y^*_i \end{bmatrix} \leftarrow \begin{bmatrix} x^*_i \\ y^*_i \end{bmatrix} + \text{Vector\_desplazamiento} \tag{3.7}$$

   Al realizar esto, la pieza girada se coloca en con las mismas coordenadas mínimas en x y en y que la pieza tenía antes del giro (Ver figura 3.9). Esto evita que piezas giradas puedieran tener coordenadas negativas o muy fuera de los límites del objeto.

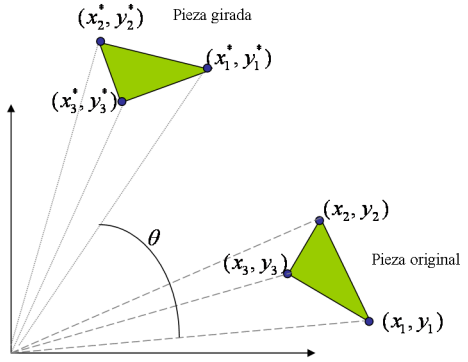

Figura 3.8: Rotación de piezas un ángulo θ.

Heurísticas de búsqueda local

El presente trabajo no implementa heurísticas de búsqueda local. Puede considerarse que el algoritmo genético que evoluciona la hiperheurística realiza una búsqueda local entre las hiperheurísticas posibles al realizar la mutación y el cruce de individuos con buena aptitud. Además, podría pensarse que hiperheurísticas vecinas (es decir, hiperheurísticas muy parecidas entre sí) dan lugar a soluciones vecinas (también muy parecidas entre sí).

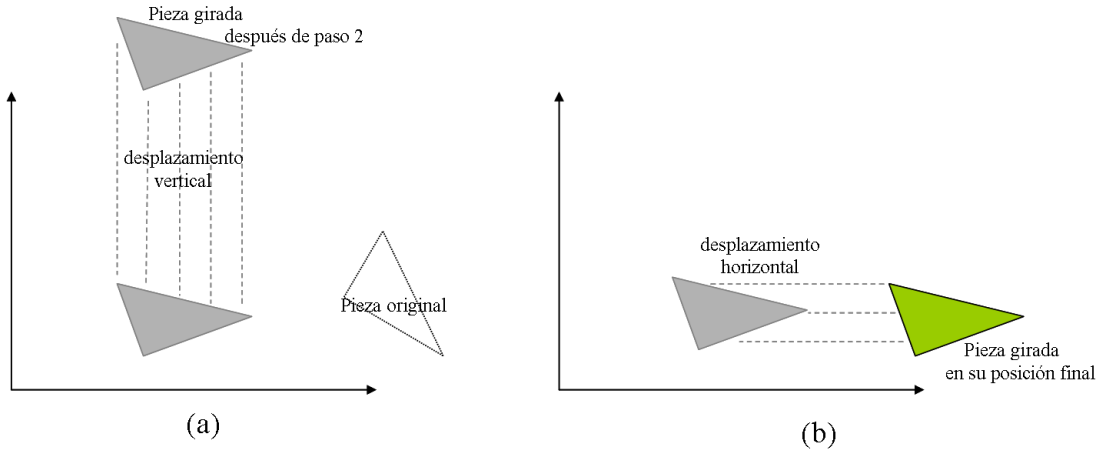

Figura 3.9: Posicionamiento de la pieza girada para mantener las coordenadas mínimas (en x y en y) que la pieza tenía antes del giro.

3.8. Resumen

En este capítulo se describió el procedimiento propuesto para generar hiperheurísticas a través de un algoritmo genético. Se explicó la manera en que las instancias, las acciones y los estados del problema fueron representados en el modelo de solución. Las hiperheurísticas generadas son combinaciones de heurísticas simples que resuelven el problema de corte y empacado de material en dos dimensiones. Se describieron las heurísticas simples implementadas que conforman las hiperheurísticas.

Capítulo 4

Metodología de Experimentación

En este capítulo se describen la generación de instancias que fue necesaria para probar el modelo de solución propuesto, los problemas seleccionados para llevar a cabo la experimentación y además se detallan las características de los diferentes experimentos realizados.

4.1. Generador de instancias

Para poder probar el algoritmo propuesto fue necesario diseñar y programar un algoritmo capaz de generar de manera aleatoria instancias del problema de empacado irregular de material de dos dimensiones, específicamente del problema SBSBPP irregular de 2 dimensiones. Las piezas generadas en cada problema son polígonos irregulares convexos que tendrán mínimo 3 y máximo 8 lados.

El algoritmo fue implementado en el lenguaje de programación Java. El generador de instancias consiste en un algoritmo que, dados ciertos parámetros, genera una instancia del problema de empacado irregular de material de dos dimensiones con solución óptima conocida donde el desperdicio de material es nulo, es decir, las piezas generadas pueden colocarse de manera exacta en el número de objetos de la solución óptima. El patrón de piezas en la solución óptima puede obtenerse realizando cortes en una orilla o otra orilla del objeto completo o ya dividido; aunque los cortes pueden ser con cualquier inclinación.

Con la finalidad de que las instancias contengan en menor o mayor medida algunos rectángulos, el algoritmo empieza generando solamente rectángulos y a partir de cierto número de rectángulos creados, genera piezas irregulares. En este sentido, puede considerarse un algoritmo híbrido. El número de rectángulos que se crean inicialmente, es definido mediante un parámetro de entrada del algoritmo. Debe considerarse que el objeto en sí mismo es un rectángulo, de tal modo que si el parámetro de rectángulos iniciales es 0, significa que el algoritmo no creará intencionalmente ningún rectángulo adicional. Si, por ejemplo, el parámetro de rectángulos iniciales es 1, esto indica que se creará un rectángulo adicional dividiendo el objeto en dos rectángulos. En general, si el

parámetro de rectángulos iniciales es k, el algoritmo dividirá el objeto en rectángulos hasta formar $k+1$ rectángulos y a partir de ese momento, en cada ocasión dividirá la pieza mayor en dos (con algún corte que puede ser generalmente inclinado) hasta alcanzar el número de piezas deseada en cada objeto.

El número final de rectángulos en el problema generado será algo o mucho menor que el número de rectángulos creados inicialmente ya que en la etapa donde se generan las piezas irregulares, algunos –o todos– los rectángulos creados serán divididos en figuras irregulares hasta completar el número deseado de piezas en el objeto.

Los parámetros del algoritmo que pueden elegirse para crear una instancia son:

- Número de objetos.
- Dimensiones de los objetos.
- Número de piezas en cada objeto.
- Lado mínimo que tendrá cada pieza.
- Razón $(lado\ mayor)/(lado\ menor)$ máxima permitida en cualquier pieza.
- Número inicial de rectángulos que se crearán.

Las coordenadas de las piezas creadas siempre son números pares con la finalidad de poder obtener siempre un punto medio con coordenadas enteras de cualquier arista de cualquier pieza (esta característica de las coordenadas es necesaria en la función implementada que verifica si una pieza está dentro de otra, función que es utilizada en algunas heurísticas de acomodo).

El procedimiento para crear problemas de empacado irregular con solución óptima conocida consiste en dividir sucesivamente al objeto de acuerdo al siguiente algoritmo (Ver figura 4.1):

1. Seleccionar la pieza de mayor área de la lista. Al iniciar el algoritmo, la pieza mayor será el objeto mismo.

2. Si el número de piezas es menor al número inicial de rectángulos que se desea crear, se divide la pieza de mayor área (que debe ser un rectángulo) en dos nuevos rectángulos de la siguiente forma (Ver figura 4.2):

   a) Seleccionar una orientación del rectángulo de forma aleatoria (horizontal o vertical).

   b) Seleccionar un punto aleatoriamente en los lados con la orientación seleccionada anteriormente. La selección se hace de tal manera que el punto seleccionado no esté más cerca de uno de los vértices que la distancia permitida como lado mínimo de una pieza.

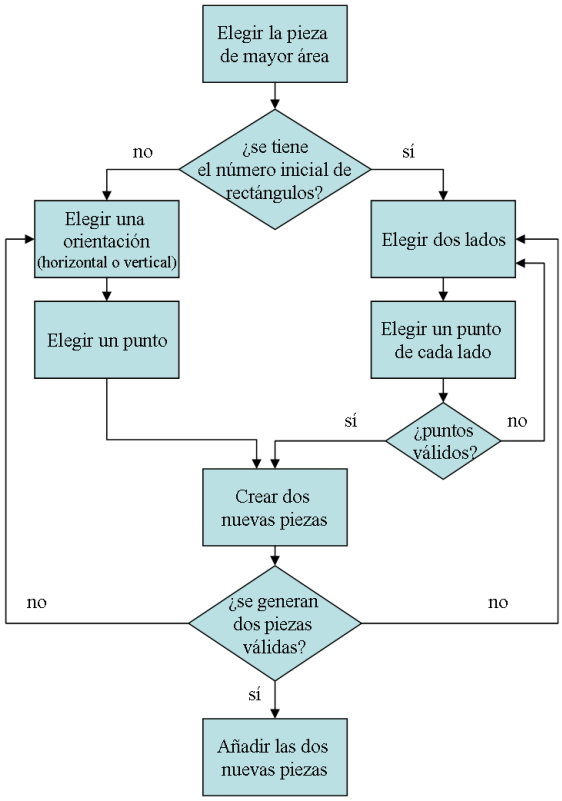

Figura 4.1: Proceso general del generador de instancias. Este proceso se repite hasta alcanzar el número de piezas deseado.

c) Construir una línea perpendicular a la orientación seleccionada a partir del punto.

d) Determinar los puntos de intersección con los bordes del rectángulo seleccionado.

e) Crear dos rectángulos nuevos.

Si el número de piezas ya alcanzó al número inicial de rectángulos que se desea crear, se divide la pieza de mayor área en dos de la siguiente forma (Ver figura 4.3):

a) Seleccionar dos lados de la pieza de forma aleatoria.

b) Seleccionar un punto aleatoriamente de cada lado seleccionado, de tal manera que el punto no esté más cerca de uno de los vértices que la distancia permitida como lado mínimo de una pieza. El punto seleccionado de cada lado puede ser también uno de sus vértices.

c) Verificar que los dos puntos seleccionados sean válidos. Esto significa:

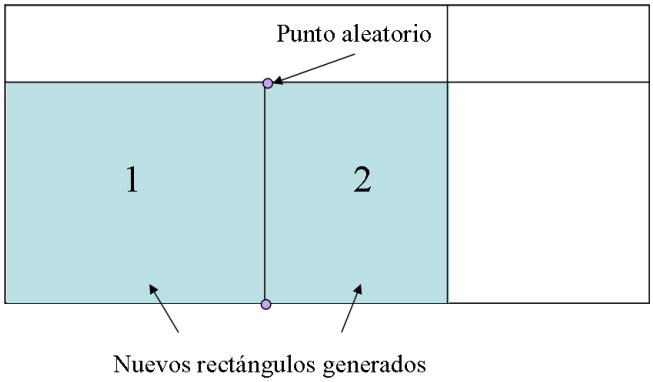

Figura 4.2: División de un rectángulo en dos.

- Que al cortar en los dos puntos seleccionados no se esté generando una pieza de más de 8 lados. Esto ocurre cuando la pieza a dividir tiene 8 vértices, se seleccionan dos lados consecutivos y ninguno de los puntos seleccionados es un vértice.
- Que los dos puntos seleccionados no estén sobre una misma arista de la pieza. Esto puede ocurrir cuando al menos uno de los puntos seleccionados es un vértice.
  Si se da alguno de estos casos, rehacer pasos *a*), *b*), y *c*).

d) Construir una línea entre los dos puntos seleccionados.

e) Determinar los puntos de intersección con los bordes de la pieza seleccionada.

f) Crear dos piezas nuevas.

3. Verificar que las piezas seleccionadas sean válidas. Esto significa calcular la razón (lado mayor)/(lado menor) de cada una de las dos nuevas piezas. Si la razón de alguna de las piezas creadas es mayor a la establecida, repetir a partir del paso 2.

4. Si las piezas cumplen con las especificaciones:

   a) Eliminar la pieza de mayor área de la lista.

   b) Agregar las dos nuevas piezas creadas a la lista.

5. Repetir hasta que se haya generado el número de piezas requeridas.

Finalmente, cuando se tienen todas las piezas generadas de cada objeto, a éstas se les asigna un orden aleatorio y la instancia es almacenada en un archivo de texto donde se indican el número de piezas, las dimensiones de los objetos y las coordenadas de las piezas (Ver figura 3.3). Las coordenadas de las piezas que se escriben en el archivo no tienen relación con su ubicación en la solución óptima pues se escriben las coordenadas

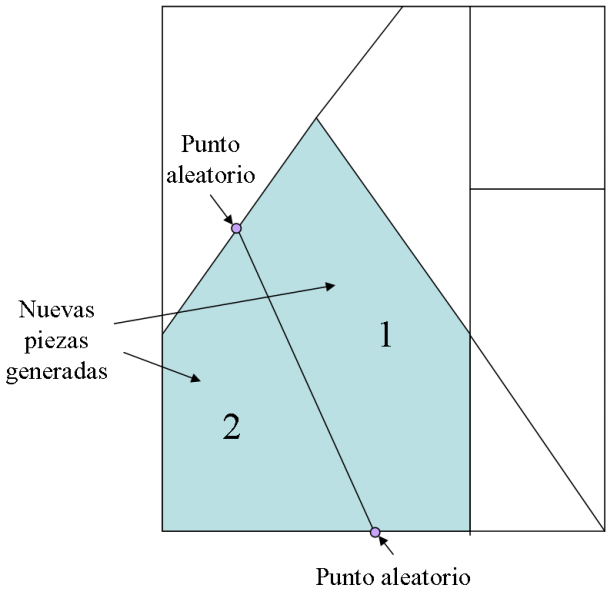

Figura 4.3: División de una pieza en dos.

de la pieza posicionada en la esquina inferior izquierda de un objeto (Ver figura 3.4). Así, para cada pieza generada, la mínima coordenada horizontal y la mínima coordenada vertical tendrán valor de 0. La forma en que cada instancia es almacenada en un archivo de texto coincide con la forma en que el modelo propuesto exite la representación de las instancias de acuerdo a la sección 3.6.

La figura 4.4 muestra un ejemplo de una instancia creada con el generador de un objeto con 10 piezas. En la figura puede observarse que se crearon algunos rectángulos primero, antes de empezar a dividir la pieza en figuras irregulares. De los rectángulos que se crearon, la instancia conservó dos.

4.2. Conformación del conjunto de instancias

De entre los problemas encontrados en la literatura, la instancia llamada *Fu* es un problema artificial de 12 polígonos convexos creado por Fujita et al. [15]. Para nuestra implementación se escaló por un factor de 10 y se le asignaron dimensiones de 300 $\times$ 300 a los objetos. Este fue el único problema *benchmark* encontrado que contenía exclusivamente polígonos convexos. Su solución óptima no es conocida.

Con el programa generador de problemas descrito en la sección anterior se generaron 540 problemas divididos en 18 tipos principales. De cada tipo de problemas se generaron 30 instancias con las siguientes características (Ver cuadro 4.2). En todos los problemas generados, cuando el óptimo es conocido, éste se logra sólo si se tiene

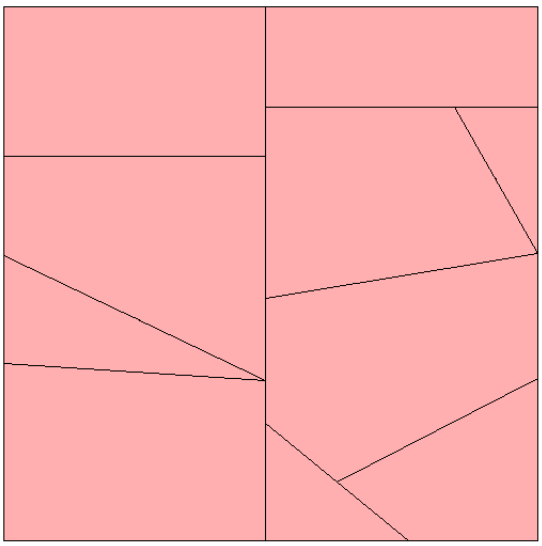

Figura 4.4: Ejemplo de una instancia generada de un objeto con 10 piezas.

desperdicio nulo en todos los objetos.

Los problemas tipo G son los únicos problemas generados aleatoriamente cuyo óptimo es desconocido, ya que fueron producidos originalmente con 15 objetos que contienen 3 piezas cada uno. Es decir, 45 piezas en total con tamaño promedio por pieza de 1/3 de objeto. Luego fueron eliminadas 9 de esas piezas elegidas al azar para dejar un total de 36 piezas. Así que el óptimo de cada instancia tipo G queda desconocido, pero se conoce su límite superior que es de 15 objetos.

En la mayoría de los problemas generados, el óptimo se consigue con el mismo número de piezas en cada objeto, a excepción de los problemas tipo C, K, P y R. En estos problemas, el tamaño de las piezas es más variable. Por ejemplo, los problemas tipo C se generaron dividiendo 3 objetos en 4 piezas cada uno y dividiendo otros 3 objetos en 8 piezas cada uno. Los problemas tipo K se generaron dividiendo 3 objetos en 3 piezas cada uno y dividiendo otros 3 objetos en 15 piezas cada uno, es decir, este tipo de problemas tiene piezas muy grandes (de tamaño promedio 1/3 de objeto) al mismo tiempo que también tiene piezas pequeñas (de tamaño promedio 1/15 de objeto). Los problemas tipo P se generaron dividiendo 2 objetos en 2 piezas cada uno, 2 objetos en 5 piezas cada uno, 2 objetos en 9 piezas cada uno y finalmente dividiendo otros 2 objetos en 12 piezas cada uno. Los problemas tipo R se generaron dividiendo 9 objetos en 2, 3, ..., 10 piezas respectivamente.

Las características de los problemas que son un indicador de irregularidad se encuentran descritas en el cuadro 4.2.

En general, puede considerarse que el problema es más irregular si contiene piezas cuyo lado mínimo es pequeño y la razón (*lado mayor*)/(*lado menor*) de las piezas es mayor, y esto produce piezas con índice de rectangularidad más pequeño. La rectan-

Cuadro 4.1: Descripción de los problemas considerados en la experimentación.

Tipo	Objetos	Piezas	Cantidad de Problemas	Óptimo (número de objetos)
Fu	300×300	12	1	desconocido
Tipo A	1000×1000	30	30	3
Tipo B	1000×1000	30	30	10
Tipo C	1000×1000	36	30	6
Tipo D	1000×1000	60	30	3
Tipo E	1000×1000	60	30	3
Tipo F	1000×1000	30	30	2
Tipo G	1000×1000	36	30	desconocido
Tipo H	1000×1000	36	30	12
Tipo I	1000×1000	60	30	3
Tipo J	1000×1000	60	30	4
Tipo K	1000×1000	54	30	6
Tipo L	1000×1000	30	30	3
Tipo M	1000×1000	40	30	5
Tipo N	1000×1000	60	30	2
Tipo O	1000×1000	28	30	7
Tipo P	1000×1000	56	30	8
Tipo Q	1000×1000	60	30	15
Tipo R	1000×1000	54	30	9
Total			541	

gularidad es la proporción entre el área de la pieza y el área del rectángulo horizontal que la contiene. La combinación de lado mínimo de piezas pequeño y razón máxima permitida grande, que se da en los problemas Tipo L, M y O ocasiona que se generen muchas piezas delgadas y largas, muchas de las cuales van de un extremo a otro del objeto. Entre mayor sea el número de rectángulos iniciales, la instancia tiende a tener algunos rectángulos o bien, piezas con lados horizontales y/o verticales. Es de esperarse que esto la convierta en una instancia más fácil de resolver pues sus lados tienen más posibilidad de empatar con las orillas del objeto o con otra pieza ya colocada que también contenga lados horizontales y/o verticales.

Dentro de los problemas generados, las 30 intancias de tipo I son totalmente regulares (sólo rectángulos). Fueron formadas dividiendo 3 objetos en 20 rectángulos cada uno (19 rectángulos extras al rectángulo inicial que es el objeto mismo).

Cuadro 4.2: Características de irregularidad de los probremas generados.

Tipo	Lado mínimo de las piezas	Razón *(lado mayor)*/*(lado menor)* máxima de las piezas	Rectángulos iniciales por objeto
Tipo A	100	3	3
Tipo B	100	3	1
Tipo C	100	3	1
Tipo D	100	3	3
Tipo E	50	10	1
Tipo F	50	5	3
Tipo G	100	3	1
Tipo H	100	3	1
Tipo I	100	3	19
Tipo J	100	3	9
Tipo K	100	4	1
Tipo L	10	10	1
Tipo M	10	10	1
Tipo N	30	5	8
Tipo O	10	10	0
Tipo P	25	5	0
Tipo Q	200	2	1
Tipo R	80	4	1

4.3. Descripción de los experimentos

Para validar la bondad del modelo propuesto se diseñaron los experimentos descritos en el cuadro 4.3.

- **Experimento I**.- El experimento I utiliza como problemas de entrenamiento la instancia *Fu* además de 270 instancias generadas: los problemas tipo A, B, C, D, E, F, G, H e I. Estos problemas de entrenamiento serán resueltos múltiples veces por las distintas generaciones de individuos del algoritmo genético. El desempeño que tengan los individuos (hiperheurísticas) les hará acreedores a una medición de su aptitud que condicionará su permanencia en la población. La población evolucionará un determinado número de generaciones. Finalmente, el mejor individuo de la última generación, que llamaremos HH1, será utilizado para resolver los 270 problemas de prueba constituido por las 30 instancias de cada tipo J, K, L, M, N, O, P, Q y R. Los resultados obtenidos por HH1 en los problemas de prueba serán comparados con el desempeño de las 40 heurísticas simples en la resolución de los mismos 270 problemas de prueba.

  Para el experimento I se realizaron 2 réplicas, para validar la consistencia y ro-

Cuadro 4.3: Descripción de los experimentos realizados.

Experimento	Problemas de Entrenamiento	Problemas de Prueba	Hiperheurística Generada
I	Instancia *Fu* y problemas Tipo A a I (271 problemas)	Problemas Tipo J a R (270 problemas)	HH1a HH1b
II	Problemas Tipo J a R (270 problemas)	Instancia *Fu* y problemas Tipo A a I (271 problemas)	HH2
III	Instancia *Fu* y 15 primeros problemas de cada Tipo A a R (271 problemas)	Últimos 15 problemas de cada Tipo A a R (270 problemas)	HH3
IV	Últimos 15 problemas de cada Tipo A a R (270 problemas)	Instancia *Fu* y 15 primeros problemas de cada Tipo A a R (271 problemas)	HH4

bustez de los resultados.

- **Experimento II**.- Este experimento sólo intercambia los problemas de entrenamiento y de prueba respecto al experimento I.

- **Experimento III**.- El experimento III utiliza como problemas de entrenamiento la instancia *Fu* además de 270 instancias generadas: 15 problemas de cada tipo A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q y R. Como problemas de prueba utiliza los 15 problemas generados de cada tipo que no fueron utilizados como problemas de entrenamiento. La elección de los 15 problemas de cada tipo que serían utilizados como entrenamiento y prueba fue de manera sistemática: los primeros 15 problemas generados de cada tipo se utilizaron como entrenamiento y los últimos 15 como prueba. Dado que la generación de instancias es aleatoria y puede suponerse independencia en cuanto a las características de dos instancias creadas consecutivamente, se espera que la selección sistemática ofrezca los mismos resultados que la selección totalmente aleatoria de instancias.

- **Experimento IV**.- Por último, y similar al caso del experimento II, el experimento IV intercambia los problemas de entrenamiento y de prueba respecto al experimento III.

4.4. Parámetros del modelo de solución

En esta sección se describen algunos parámetros que se probaron antes de llegar a los parámetros definitivos que se utilizaron en los cuatro experimentos diseñados.

4.4.1. Elección de la estrategia de rotación de piezas

Por otra parte, antes de realizar los experimentos aquí reportados, se hicieron pruebas preliminares considerando los siguientes dos tipos de rotación. Ambos tipos de rotación están implementados.

- **Rotación en múltiplos de 90°**. Cada vez que se va a colocar una pieza en un objeto se considera una lista con los ángulos 0°, 90°, 180° y 270°.
- **Rotación en múltiplos de 5°**. Cada vez que se va a colocar una pieza en un objeto se construye una lista con ángulos en el intervalo $[0, 360)$. Esta lista incluye necesariamente los ángulos 0°, 90°, 180° y 270° y se construye mediante el siguiente algoritmo.
  Para *cada* lado de la pieza a colocar y *cada* lado de *cada* pieza ya colocada:

  1. Se obtiene la diferencia de sus ángulos de inclinación.
  2. Si el ángulo obtenido es negativo se le encuentra su ángulo positivo equivalente sumándole 360°.
  3. El ángulo se redondea al múltiplo de 5° más cercano.
  4. Se encuentra el ángulo que está a 180° de diferencia del ángulo original en el intervalo $[0, 360)$. Por ejemplo, si se tiene el ángulo 35° también se considera el ángulo 215° $(35° + 180°)$; si se tiene el ángulo 250° también se considera el ángulo 70° $(250° - 180°)$.
  5. Ambos ángulos se incluyen en la lista. Si un ángulo ya está en la lista, se incluye una sola vez.

  La lista resultante contiene los ángulos 0°, 90°, 180° y 270° además de otros ángulos múltiplos de 5°. No necesariamente incluye todos los ángulos múltiplos de 5° en el intervalo $[0, 360)$. La longitud de esta lista de ángulos tiende a ser mayor entre más piezas haya ya colocadas en el objeto.

Ya sea con la lista de ángulos múltiplos de 5° o múltiplos de 90°, la rotación de piezas se incorpora en tres de las heurísticas de acomodo implementadas: Enfoque Constructivo, Enfoque constructivo de mínima área fondo-izquierda y Enfoque Constructivo de máxima adyacencia. Cada vez que se va a colocar una pieza en un objeto, se prueba en todas las rotaciones de la lista y se elige la rotación en donde el criterio de

la heurística se optimice. Por ejemplo, si se está acomodando la pieza con el Enfoque Constructivo de máxima adyacencia, se elige la inclinación de la pieza donde logre más adyacencia con el resto de las piezas del objeto y/o los límites el objeto mismo.

En la heurística Fondo-Izquierda no es posible implementar rotación, ya que la heurística en sí no tiene un criterio para determinar qué ángulo de giro se eligiría para cada pieza.

Contra lo que la intuición pudiera sugerir, en pruebas preliminares sólo se observó una mejoría muy marginal al usar el segundo tipo de rotación, comparado con la rotación en múltiplos de 90°. Sin embargo, el costo computacional sí aumentó considerablemente. Es por esto que, los experimentos descritos en el cuadro 4.3 se realizan usando la rotación en múltiplos de 90°. Además, es necesario tener presente que el objetivo de este trabajo es desarrollar una hiperheurística que tenga un mejor desempeño *relativo* al desempeño de las heurísticas simples. La mejora relativa hiperheurística *vs.* heurísticas simples debe poder lograrse con cualquier estrategia de rotación, o incluso con la restricción de que las piezas no roten.

4.4.2. Elección de los parámetros del algoritmo genético

Se realizaron algunas pruebas preliminares para obtener parámetros del algoritmo genético (AG) que dieran como resultado una hiperheurística con buen desempeño.

- Tamaño de la población. Se refiere al número de individuos (hiperheurísticas) presentes en cada generación del AG. Entre mayor sea el tamaño de la población, es de esperarse que el mejor individuo tenga una mayor aptitud. El tiempo de ejecución del AG es proporcional al tamaño de la población. En la presente investigación se encontró que una población de 100 individuos es un buen número pues se obtiene una solución de buena calidad sin hacer demasiado lento el trabajo computacional. El resultado obtenido con 100 individuos fue ligeramente mejor que el resultado obtenido con 50 individuos en experimentos preliminares.

- Probabilidad de cruce y mutación. Debido a que el AG utilizado es de estado estable, es decir con traslape generacional donde algunos individuos viven durante varias generaciones; 1.0 es el mejor valor para el parámetro de probabilidad de cruce. Los dos operados descritos en la sección 2.3.1 son aplicados con la misma probabilidad. Por otra parte, experimentos preliminares mostraron que una probabilidad de mutación 0.05 ofrece buenos resultados, pues no es suficientemente grande para evitar que el AG converja. En la sección 2.3.1 se describen los tres diferentes operadores de mutación implementados en el AG: agregar bloque, remover bloque y cambiar bloque. Debido a que los dos primeros operadores de mutación son algo más disruptivos que el tercero, cada uno posee un 25 % de prob-

Cuadro 4.4: Parámetros utilizados en el algoritmo genético de los experimentos.

Parámetro	Descripción	Valor
N	Tamaño de la Población	100
X	Probabilidad de Cruce	1.0
μ	Probabilidad de Mutación	0.05
G	Número de Generaciones	500

abilidad de ocurrir, mientras que el tercer operador tiene un 50 % de probabilidad de ocurrir.

- Tamaño de los individuos. Dados los operadores de cruce y mutación aplicados, la longitud de los individuos varía conforme al número de bloques que contengan. Para controlar que un cromosoma pueda llegar a ser muy pequeño, se estableció lo siguiente: cada vez que se selecciona para cruce un individuo de 2 o menos bloques, éste es reemplazado por un individuo totalmente nuevo creado al azar (con longitud entre 5 y 10 inclusive). Por otra parte, es posible controlar que el tamaño de los individuos no crezca demasiado sólo con asegurarse que nunca tendrán bloques repetidos. Esto se logra evitando agregar un bloque que ya esté en el cromosoma cuando se efectúa el operador de mutación que agrega bloques o cuando se efectúa cualquiera de los operadores de cruce.

- Número de generaciones. El número de ciclos que es necesario correr el AG hasta que se detenga debe ser suficiente para que la población *evolucionen* generando un individuo en la última generación que constituya una buena hiperheurística. Se observó que 500 es un número de generaciones más que suficiente para lograr el objetivo.

El cuadro 4.4.2 resume los parámetros utilizados.

Para la generación de los N individuos de la población inicial, se utilizaron números aleatorios:

1. La longitud de cada individuo fue de 5 a 10 bloques con distribución uniforme.

2. Cada bloque fue creado a través de 9 números aleatorios:

   a) Los primeros 8 elementos del bloque representan proporciones que van de 0 a 1 y se generan con distribución normal con media 0.5 y desviación estándar igual a 0.5, descartando valores que cayeran fuera del rango de -3 a 3. Esta distribución produce aproximadamente un 68.3 % de números entre 0 y 1. El resto de los números es mayor a 1 o menor a 0 lo que produce un 95.3 % de bloques con al menos un elemento fuera del rango de 0 a 1. Para el estado de un problema en particular, los primeros ocho números se encuentran dentro

del rango de 0.0 a 1.0, de tal manera que el estado actual del problema es un punto dentro de un hipercubo unitario de ocho dimensiones. Es decir, se crean bloques que no representarían estados reales de un problema, pero que sí pueden llegar a ser el bloque con menor distancia a una determinada instancia del problema. Con base al trabajo realizado por Ross [27], los puntos definidos en los cromosomas pueden estar fuera del rango de 0 a 1 con la finalidad de ampliar el espacio en donde los puntos puedan ocurrir. Si consideramos los estados del problema que representan puntos muy cercanos a las esquinas del hipercubo, resulta bastante probable que los bloque que representan puntos que caen fuera del hipercubo se encuentren más cercanos a estos estados del problema.

b) El noveno elemento del bloque lo constituyó un número aleatorio entero entre 0 y 39, con distribución uniforme, que representa la acción a seguir.

4.5. Resumen

En este capítulo se explicó el procedimiento diseñado e implementado para generar instancias aleatoriamente. Las instancias creadas, además de la única instancia de polígonos convexos encontrada en la literatura, fueron clasificadas como problemas de entrenamiento y prueba en los experimentos diseñados para probar el modelo de solución propuesto. Además se detallan las características de los cuatro diferentes experimentos realizados.

Capítulo 5

Análisis de Resultados

En el presente capítulo se describen los resultados obtenidos en cada uno de los experimentos realizados. Sólo del experimento I se realizaron dos réplicas y del resto de los experimentos, una sóla réplica. Las hiperheurísticas generadas y su desempeño en instancias de prueba es reportada en este capítulo.

Cada uno de los experimentos utilizan los mismos 541 problemas disponibles, clasificándolos de manera diferente en instancias de entrenamiento o prueba. La fase 1 de los experimentos, que consiste en resolver todas las instancias (entrenamiento y prueba) con cada una de las 40 heurísticas simples, se describe en la siguiente sección.

Las fases 2 y 3 se describen en los 4 experimentos. La fase 2 consiste en generar la hiperheurística por medio del AG. La fase 3 evalúa el desempeño de la hiperheurística comparándola con el desempeño de las diez heurísticas simples de selección. Se utilizan dos medidas de desempeño: 1) el número de objetos extras que requieren para resolver las instancias de prueba respecto al número de objetos que requiere la mejor de las 40 heurísticas simples y 2) el número de objetos extras que requieren para resolver las instancias de prueba respecto al número de objetos que requieren en promedio las 40 heurísticas simples.

5.1. Resolver problemas de prueba y entrenamiento con las heurísticas simples

Tanto los problemas entrenamiento como los de prueba (541 en total) fueron resueltos con cada una de las 40 heurísticas simples. Los problemas de entrenamiento se resuelven con las heurísticas simples para identificar la mejor heurística y su aptitud correspondiente (medida de acuerdo a la ecuación 2.2). La aptitud obtenida por la mejor heurística simple al resolver cada problema es luego utilizada en la función de evaluación de cada individuo (hiperheurística) del AG, de acuerdo a las ecuaciones 2.3 y 2.4.

Por otra parte, los problemas de prueba se resuelven con las heurísticas simples para posteriormente comparar su desempeño contra el desempeño observado para la

hiperheurística generada por el AG al entrenar con los problemas de entrenamiento.

Al resolver todas las instancias, se registra la mejor heurística simple y su aptitud, calculada conforme la ecuación 2.2. La aptitud promedio de la mejor heurística simple que resolvió cada uno de los 541 problemas fue de 0.5925. En muchos casos, hubo varias heurísticas simples que resolvieron con el mismo número de objetos el mismo problema. En estos casos se considera como mejor heurística a la que obtuvo mayor aptitud. La heurística 7 fue la mejor en resolver el 35.5 %. De acuerdo con el cuadro 3.2, esta heurística se forma con la heurística de selección *First Fit Decreasing* (FFD) y la heurística de acomodo *Máxima Adyacencia* (MA). En el cuadro 5.1 se muestra cuáles fueron las heurísticas simples que resultaron mejores en algunos de los problemas considerados en los experimentos realizados. De las 40 heurísticas disponibles sólo 18 resultaron la mejor para al menos un problema.

Cuadro 5.1: Mejores heurísticas simples al resolver las instancias de experimentación.

Heurística simple	Número de instancias	Porcentaje
7: FFD - MA	192	35.5 %
15: Filler - MA	113	20.9 %
5: FFD - EC	52	9.6 %
6: FFD - ECMA	49	9.1 %
13: Filler - EC	46	8.5 %
35: Filler - ECMA	35	6.5 %
39: DJF - MA	30	5.5 %
37: DJF - EC	9	1.7 %
26: BF - ECMA	3	0.6 %
4: FFD - FI	2	0.4 %
27: BF - MA	2	0.4 %
36: DJF - FI	2	0.4 %
1: FF - EC	1	0.2 %
3: FF - MA	1	0.2 %
12: Filler - FI	1	0.2 %
21: NFD - EC	1	0.2 %
24: BF - FI	1	0.2 %
38: DJF - ECMA	1	0.2 %
Total	541	100.0 %

Analizando las heurísticas de selección y de acomodo por separado, la heurística de selección *First Fit Decreasing* (FFD) fue la de mejor desempeño al estar involucrada en la mejor heurística simple en poco más de la mitad de las instancias (54.5 %). Cuatro de las diez heurísticas de selección no formaron parte de la mejor heurística

simple para ninguno de los 541 problemas. Por otra parte, la heurística de acomodo *Máxima Adyacencia* (MA) formó parte de la mejor heurística simple en el 62.5 % de las instancias. Los cuadros 5.2 y 5.3 muestran el detalle del desempeño de las heurísticas de selección y acomodo.

Cuadro 5.2: Mejores heurísticas de selección al resolver las instancias de experimentación.

Heurística de selección	Número de instancias	Porcentaje
First Fit Decreasing (FFD)	295	54.5 %
Filler + FFD	195	36.0 %
Djang and Finch (DJF)	42	7.8 %
Best Fit (BF)	6	1.1 %
First Fit (FF)	2	0.4 %
Next Fit Decreasing (NFD)	1	0.2 %
Total	541	100.0 %

Cuadro 5.3: Mejores heurísticas de acomodo al resolver las instancias de experimentación.

Heurística de acomodo	Número de instancias	Porcentaje
Máxima Adyacencia (MA)	338	62.5 %
Enfoque Constructivo (EC)	109	20.1 %
E.C. de Mínima Área (ECMA)	88	16.3 %
Fondo Izquierda (FI)	6	1.1 %
Total	541	100.0 %

Por otra parte, también se desarrolla un análisis con relación al desempeño de las heurísticas simples respecto al óptimo. De los 541 problemas en total, se conoce el óptimo de 510 instancias, en los cuales la solución óptima se logra sólo al acomodar las piezas en objetos con desperdicio nulo. Se resolvieron los 510 problemas con cada una de las 40 heurísticas simples. Se analizó el porcentaje de objetos adicionales al valor óptimo que, en promedio, cada heurística de selección y heurística de acomodo requirió en su solución. Este análisis también permite medir el desempeño de cada heurística individualmente. Los resultados pueden apreciarse en las figuras 5.1 y 5.2. La definición de las siglas de cada heurística puede obtenerse del cuadro 3.2.

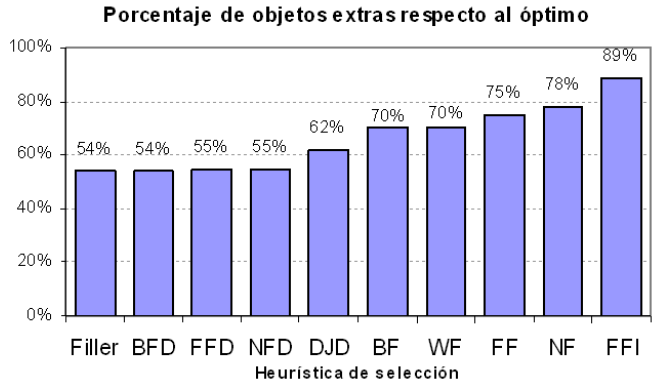

Figura 5.1: Porcentaje promedio de objetos adicionales al óptimo que utiliza cada heurística de selección.

En cuanto a las heurísticas de selección, la mejor es la heurística *Filler* la cual, en promedio necesita 54.1 % de objetos extras al óptimo en todas las instancias analizadas. Por una mínima diferencia, le sigue la heurística *Best Fit Decreasing* quien se excede al óptimo en 54.4 % de objetos en promedio. Por último la heurística *First Fit Increasing* tiene el peor desempeño individual al requerir 89 % de objetos adicionales al óptimo, en promedio.

En cuanto a las heurísticas de acomodo, la mejor es la heurística de *Máxima Adyacencia* la cual, en promedio necesita 54.4 % de objetos extras al óptimo en todas las instancias analizadas. Le sigue la heurística *Enfoque Constructivo* quien se excede al óptimo en 57.6 % de objetos en promedio. Por último, y por mucha diferencia respecto a las otras tres, la heurística *Fondo-Izquierda* tiene el peor desempeño individual al requerir casi el doble (94.9 %) de objetos adicionales al óptimo, en promedio. Esto tiene sentido, pues es la única heurística que no permite rotación de piezas.

Cada tipo de problemas generados tiene distinto grado de irregularidad y de dificultad. Esto se hace patente al observar que el desempeño que tienen las distintas heurísticas en cada tipo de problemas cuyo óptimo es conocido. Las figuras 5.3 y 5.4 muestran cuáles tipos de problemas fueron resueltos utilizando menos objetos adicionales en promedio por las 40 heurísticas simples, tanto en porcentaje como en valor absoluto de los objetos.

Por supuesto, el porcentaje de objetos extras tiene relación directa con el valor del óptimo. Por ejemplo, los problemas tipo F tienen un óptimo de 2 objetos. Si una heurística requiere de 3 objetos, esto significa que requirió un 50 % de objetos extras en constraste con los problemas tipo Q, por ejemplo, donde un objeto extra significa apenas el 6.7 % dado que su valor óptimo es de 15 objetos. Esto explica, en parte, por qué los problemas tipo Q son los que requieren en promedio menor porcentaje de

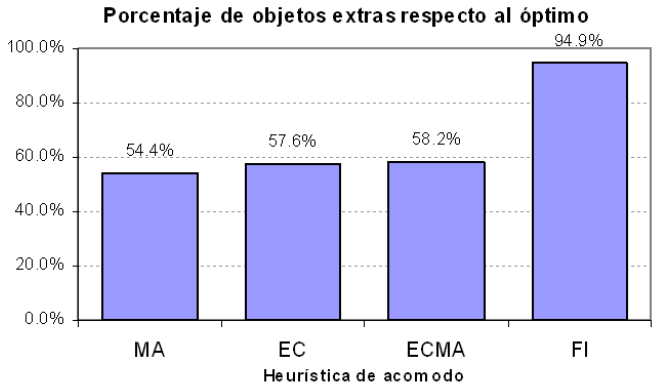

Figura 5.2: Porcentaje promedio de objetos adicionales al óptimo que utiliza cada heurística de acomodo.

objetos adicionales al óptimo. El valor del óptimo para cada tipo de problemas está en el cuadro 4.2.

Los problemas tipo I son regulares, es decir continenen solamente rectángulos, así que son resueltos con relativa facilidad por las heurísticas al requerir sólo 1.1 objetos adicionales en promedio (un 37 % del valor óptimo, que es 3).

Por el contrario, las instancias tipo E son bastante irregulares, al permitir casi ningún lado perpendicular de las piezas (número inicial de rectángulos igual a uno, según cuadro 4.2). Por lo que son resueltos con algo de dificultad por las heurísticas ya que requieren 4.1 objetos adicionales en promedio (un 138 % del valor óptimo, que es 3).

5.2. Experimento I

Este experimento consiste en entrenar el algoritmo genético con la instancia *Fu* además de los primeros 270 problemas generados (instancias tipo A a la I). Posteriormente, evaluar el desempeño de la heurística generada con los problemas de prueba (instancias tipo J a la R).

5.2.1. Ejecutar el AG con los problemas de entrenamiento para obtener una hiperheurística general

Cada una de las 500 generaciones del AG está conformada con 100 individuos, cada uno de los cuales constituye una hiperheurística válida con un valor de aptitud determinado. No obstante, el mejor individuo de la última generación suele tener una

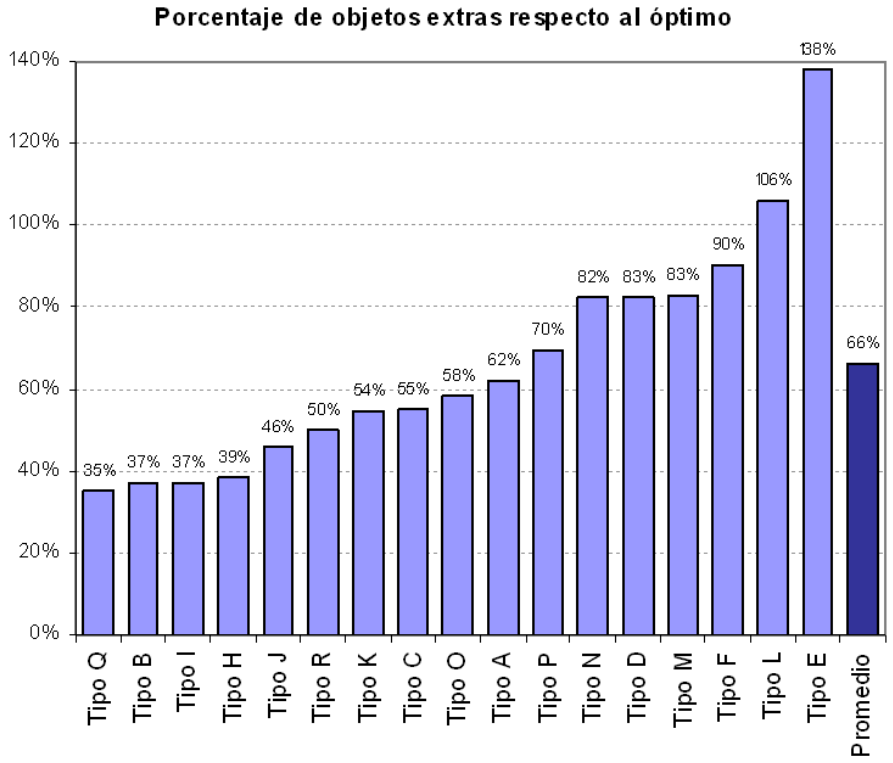

Figura 5.3: Porcentaje promedio de objetos adicionales al óptimo que utilizan las 40 heurísticas simples en cada tipo de problemas generados.

aptitud buena y es quien se considera como la hiperheurística generada por AG.

En el caso del experimento I, las hiperheurísticas generadas por cada una de las dos réplicas realizadas están en los cuadros 5.4 y 5.5.

Las hiperheurísticas están formadas por seis y cinco reglas respectivamente. Las primeras ocho columnas nos representan el estado del problema en donde las primeras tres columnas indican el porcentaje de piezas con alta, media y baja rectangularidad que faltan por acomodar; la cuarta y quinta columna representan el porcentaje de piezas de área grandes y pequeñas respectivamente; las columnas 6 y 7 representan el porcentaje de piezas altas y cortas; mientras que la octava columna indica el porcentaje del total de piezas que faltan por acomodar. En el cuadro 3.1 se encuentra una descripción detallada de las características que forman el estado del problema. La última columna representa la acción que deberá llevarse a cabo si se cumple la máxima cercanía con la condición anterior. La acción es una combinación de una heurística de selección y una de acomodo, la descripción de los diferentes tipos de acciones posibles se encuentra en el cuadro 3.2.

Existen ciertas acciones que se repiten, como la acción 23 (dos veces en la primera

Cuadro 5.4: Conjunto de reglas desarrolladas por el AG. Experimento I, **primera réplica**.

Large Rect.	**Medium Rect.**	**Low Rect.**	**Large**	**Small**	**High**	**Short**	**Remain**	**Action**
-0.75	1.04	0.20	0.50	1.1	0	0.88	1.59	23
0.57	-0.15	1.27	-0.63	0.48	0.32	0.40	0.46	33
-0.13	0.61	0.23	1.13	0.68	0.67	0.70	0.28	15
-0.04	0.36	0.60	-0.34	0.97	-0.26	0.63	-0.03	7
0.14	0.31	0.19	0.35	0.17	0.47	0.37	0.78	23
0.04	1.06	-0.36	-0.12	-0.42	-0.11	0.17	0.26	22

Cuadro 5.5: Conjunto de reglas desarrolladas por el AG. Experimento I, **segunda réplica**.

Large Rect.	**Medium Rect.**	**Low Rect.**	**Large**	**Small**	**High**	**Short**	**Remain**	**Action**
0.09	-0.11	0.13	0.44	1.23	0.26	0.41	0.81	31
0.49	0.28	0.03	0.05	0.27	0.77	1.37	0.48	7
0.54	0.28	0.03	0.05	0.28	0.80	-0.29	0.56	7
1.05	0.73	1.04	1.84	0.47	0.72	1.11	0.39	6
-0.13	1.26	0.26	0.75	0.79	1.24	0.37	0.91	39

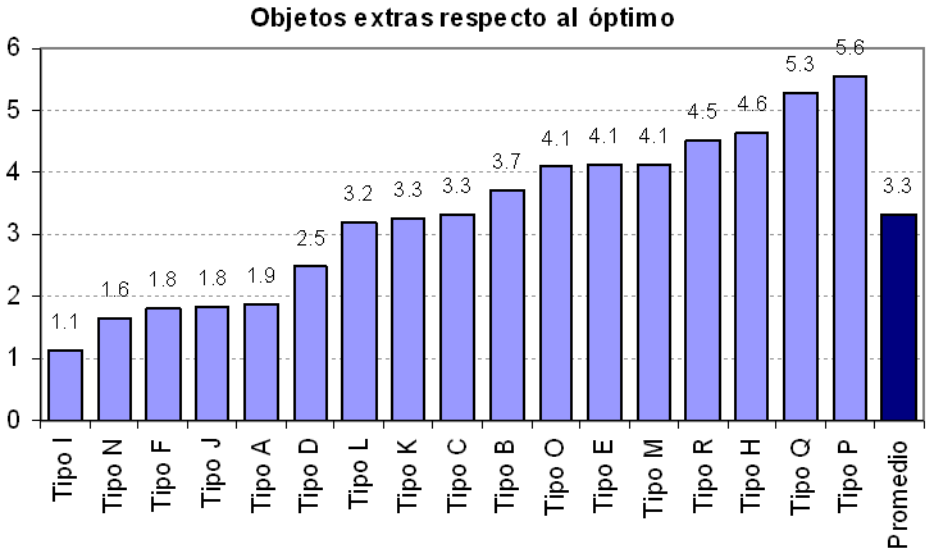

Figura 5.4: Promedio de objetos adicionales al óptimo que utilizan las 40 heurísticas simples en cada tipo de problemas generados.

réplica) y la 7 (dos veces en la hiperheurística de la segunda réplica), sin embargo aunque la acción sea la misma, la condición es diferente en todos los casos. No obstante, en el caso de la segunda réplica, el segundo y tercer bloque que señalan la acción 7 son muy similares.

5.2.2. Resolver problemas de prueba con la hiperheurística generada

En el experimento I los problemas de prueba los contituyen las 270 instancias tipo J a R. Comparando la **mejor** heurística simple, el número de objetos extras utilizados por las heurísticas de selección y la hiperheurística de ambas réplicas en analizarse en el cuadro 5.6. Los valores dentro del cuadro indican el porcentaje de instancias de prueba (del total de 270) en que se requirió un determinado número de objetos extras. Por ejemplo, en la primera réplica del experimento I, la hiperheurística resolvió el 87.0 % de las instancias de prueba con el mismo número de objetos con que lo hizo la mejor de las 40 heurísticas simples. Y en en el resto de los problemas de prueba (13.0 %) la hiperheurística requirió un objeto adicional. La heurística simple de selección *Filler* se utiliza en 4 de las heurísticas simples (una vez por cada heurística de acomodo considerada). Al promediar el desempeño de estas 4 heurísticas simples, se llega a la conclusión de que la heurística de selección *Filler* fue la que mejor desempeño tuvo con

las instancias de prueba resolviendo un 55.6 % de las instancias con el mismo número de objetos con que lo hizo la mejor heurística simple. En el 21.9 %, 11.0 %, 5.6 %, 3.8 % y 1.9 % de las ocasiones requirió 1, 2, 3, 4 y 5 objetos adicionales, respectivamente. Incluso, en el 0.3 % de las instancias de prueba la heurística *Filler* requirió más de 5 objetos adicionales respecto al número de objetos utilizado por la mejor de las 40 heurísticas simples.

Cuadro 5.6: Número de objetos extra en los problemas de prueba respecto al resultado obtenido por la mejor heurística simple. Experimento I, primera y segunda réplicas.

			Heurísticas de selección									
			FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HHa**	**HHb**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
0	87.0	90.4	5.0	54.4	0.5	55.6	3.7	52.7	10.5	55.0	10.5	30.9
1	13.0	9.6	32.4	22.2	20.7	21.9	29.5	23.5	34.4	21.9	34.4	32.3
2			28.1	11.4	20.2	11.0	27.4	11.3	25.4	11.1	25.4	17.3
3			15.6	5.8	18.9	5.6	16.7	6.1	13.1	5.8	13.1	10.1
4			10.5	3.7	14.6	3.8	11.9	3.4	8.3	3.9	8.3	4.3
5			4.0	2.0	9.7	1.9	5.5	2.5	3.8	1.9	3.8	3.5
> 5			4.4	0.4	15.4	0.3	5.3	0.5	4.5	0.3	4.5	1.6

Sorprende que el desempeño de las hiperheurísticas de las dos réplicas es muy similar a pesar de que las hiperheurísticas en sí, no son muy similares, ya que utilizan acciones diferentes. Las hiperheurísticas igualan el desempeño de la mejor heurística simple en la mayoría de los casos (87.0 % y 90.4 % en cada una de las réplicas). Cuando las hiperheurísticas no igualan el resultado de la mejor heurística simple, sólo agregan un objeto más.

Comparando la hiperheurística y las heurísticas de selección con el **promedio** de las heurísticas simples, los resultados de ambas réplicas en analizarse en el cuadro 5.7. Dado que el promedio de objetos utilizado por las 40 heurísticas simples en cada una de las instancias no necesariamente es un número entero, el número de objetos extra reportado en el cuadro 5.7 fue redondeado al entero más próximo (superior o inferior).

En la primera réplica, la hiperheurística generada resuelve las instancias utilizando el mismo número de objetos que el promedio de las heurísticas simples sólo el 4.4 % de las veces. Casi la mitad de las veces (47.0 %), la hiperheurística logra reducir un objeto en la solución comparándola con la heurística simple *promedio*. Mientras que en el 32.6 % de las instancias de prueba fueron resueltas por la hiperheurística con 2 objetos menos que en el promedio de las heurísticas simples. 13.0 % de las instancias tuvieron un ahorro de 3 objetos y 3.0 % de las instancias redujeron en 4 o más el número de objetos que utilizan en promedio las 40 heurísticas simples.

Cuadro 5.7: Número de objetos extra en los problemas de prueba respecto al promedio obtenido por las 40 heurísticas simples. Experimento I, primera y segunda réplicas.

			Heurísticas de selección									
			FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HHa**	**HHb**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
< -3	3.0	4.8		1.2		1.2		0.7		1.2		0.1
-3	13.0	12.2		4.3		4.7		4.3		4.2		0.8
-2	32.6	31.5	0.7	23.5		24.2	0.6	23.0	1.6	24.0	1.6	10.7
-1	47.0	47.4	13.2	42.6	0.9	41.8	8.4	43.1	22.4	42.5	22.4	44.6
0	4.4	4.1	47.8	12.6	24.7	12.7	46.7	12.8	44.4	12.6	44.4	21.7
1			21.7	11.1	29.3	11.2	24.8	10.6	16.6	10.9	16.6	12.9
2			9.1	4.0	22.4	3.8	9.7	4.4	8.2	4.0	8.2	6.0
3			4.6	0.6	13.3	0.4	5.7	0.7	4.9	0.6	4.9	2.4
> 3			2.9	0.2	9.4	0.1	4.1	0.3	1.9	0.1	1.9	0.7

Cuadro 5.8: Número de objetos extras en los problemas de prueba respecto al promedio obtenido por las 40 heurísticas simples. Comparación de primera y segunda réplicas del experimento I.

Objetos extra	Primera réplica	Segunda Réplica
	Número de instancias	
< -3	8	13
-3	35	33
-2	88	85
-1	127	128
0	12	11
Total	270	270

El desempeño de la hiperheurística generada en la segunda réplica del experimento I podría verse como ligeramente superior, como lo indica el cuadro 5.8. Para verificar si hay diferencias estadísticamente significativas entre las heurísticas generadas por ambas réplicas, se realizó la prueba de independencia basada en el estadístico χ^2 con la tabla de contingencia que representa el cuadro 5.8. El resultado no indica que las ligeras diferencias observadas entre una heurística y otra puedan deberse a otra causa distinta del azar ($valor\ p = 0.853$). Es decir, esto es un muy buen indicio de que el método para construir la hiperheurística es suficientemente robusto para generar hiperheurísticas muy parecidas en desempeño —con un mismo conjunto de instancias de entrenamiento— a pesar del componente aleatorio que representa el algoritmo genético.

Cuadro 5.9: Conjunto de reglas desarrolladas por el AG. Experimento II.

Large Rect.	**Medium Rect.**	**Low Rect.**	**Large**	**Small**	**High**	**Short**	**Remain**	**Action**
0.35	-0.01	0.08	0.22	0.34	-0.27	0.57	0.08	15
-0.12	-0.05	1.10	0.92	1.04	0.87	0.12	-0.07	31
1.11	0.71	0.12	0.54	0.70	0.67	0.33	1.30	33
0.35	-0.01	0.08	-0.03	0.58	-0.04	0.71	0.85	15
1.04	-0.01	0.08	0.22	0.34	0.55	0.70	0.22	15
-0.20	0.68	0.71	0.03	0.59	0.80	0.19	0.44	18

5.3. Experimento II

Este experimento invirtió los papeles de instancias de prueba y de entrenamiento respecto al experimento I. Es decir, los problemas tipo J a R fueron instancias de entrenamiento y los 271 instancias restantes fueron los problemas de prueba (Ver cuadro 4.3).

5.3.1. Ejecutar el AG con los problemas de entrenamiento para obtener una hiperheurística general

La hiperheurística generada por el experimento II está indicada en el cuadro 5.9. Tiene seis bloques, tres de los cuales reportan la misma acción (15) sin embargo, corresponden a diferentes estados del problema.

5.3.2. Resolver problemas de prueba con la hiperheurística generada

El cuadro 5.10 muestra desempeño de la hiperheurística generada en el experimento II comparado contra el desempeño de las heurísticas de selección en las instancias de prueba. El desempeño en este cuadro es medido con el número de objetos extra que cada heurística (o hiperheurística) requiere para solucionar la instancia respecto al número de objetos requeridos por la mejor heurística simple.

Por otra parte el cuadro 5.11 muestra el desempeño de la hiperheurística y las heurísticas de selección medido con el número de objetos extras requeridos respecto a la solución dada por el promedio de las 40 heurísticas simples.

Cuadro 5.10: Número de objetos extra en los problemas de prueba respecto al resultado obtenido por la mejor heurística simple. Experimento II.

		Heurísticas de selección									
		FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HH**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
-1	0.4										
0	86.7	17.2	57.5	8.8	58.4	16.4	55.9	24.2	58.4	24.2	38.7
1	10.0	39.2	26.3	27.2	25.6	36.9	26.8	38.7	25.7	38.7	28.2
2	2.6	22.3	9.5	24.2	9.9	20.8	10.0	22.2	9.7	22.2	15.7
3	0.4	13.6	4.0	14.9	3.6	16.1	4.1	9.0	3.3	9.0	10.2
4		5.7	2.0	14.8	1.8	6.4	2.4	4.6	2.1	4.6	5.2
5		1.7	0.6	8.7	0.6	2.7	0.7	1.0	0.6	1.0	1.8
> 5		0.4	0.1	1.5	0.1	0.6	0.1	0.3	0.1	0.3	0.2

Cuadro 5.11: Número de objetos extra en los problemas de prueba respecto al promedio obtenido por las 40 heurísticas simples. Experimento II.

		Heurísticas de selección									
		FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HH**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
< -3											0.1
-3	0.7		0.1		0.1		0.1		0.1		1.3
-2	28.0	0.1	13.7		14.6	0.1	12.7	1.1	14.6	1.1	4.8
-1	52.8	13.2	44.6	2.6	44.6	11.3	44.7	21.4	44.8	21.4	29.8
0	18.1	55.0	30.8	32.7	30.1	51.9	30.8	53.7	29.6	53.7	38.6
1	0.4	21.5	7.0	33.9	7.1	23.3	7.3	16.8	7.3	16.8	15.4
2		7.2	2.6	18.3	2.7	9.0	3.0	4.8	2.5	4.8	7.3
3		2.2	1.0	10.7	0.8	2.7	1.2	1.8	1.0	1.8	2.5
> 3		0.8	0.1	1.9	0.1	1.6	0.1	0.5	0.1	0.5	0.3

Cuadro 5.12: Conjunto de reglas desarrolladas por el AG. Experimento III.

Large Rect.	**Medium Rect.**	**Low Rect.**	**Large**	**Small**	**High**	**Short**	**Remain**	**Action**
0.41	1.03	0.81	0.67	0.04	0.04	-0.20	0.59	15
1.11	0.72	1.24	-0.27	-0.05	1.45	0.22	0.56	15
0.57	0.78	-0.25	-0.33	1.21	1.30	0.47	0.88	23
1.37	-0.54	1.09	-0.10	0.53	0.48	0.31	0.38	39
0.04	-0.07	1.04	0.03	-0.05	1.05	-0.14	0.32	15

5.4. Experimento III

Como ya se describió en el cuadro 4.3, el experimento III utiliza como problemas de entrenamiento la mitad de cada tipo de problemas generados y emplea como problemas de prueba a la otra mitad de las instancias.

5.4.1. Ejecutar el AG con los problemas de entrenamiento para obtener una hiperheurística general

El cuadro 5.12 muestra la hiperheurística generada por el experimento III, la cual estuvo conformada por cinco bloques. Nuevamente, al igual que en otros de los experimentos realizados, la heurística simple 15 aparece como acción en varias de las reglas que conforman la mejor hiperheurística. La acción 15 indica que el problema debe resolverse con la heurística de selección *Filler + FFD* y la heurística de acomodo *Máxima Adyacencia*. El cuadro 3.2 indica las heurísticas simples y de acomodo que representan cada una de las acciones.

5.4.2. Resolver problemas de prueba con la hiperheurística generada

Los cuadros 5.13 y 5.14 muestran el desempeño de las heurísticas de selección y de la hiperheurística generada por el AG al resolver las instancias de prueba. El cuadro 5.13 mide el número de objetos extras que usa cada heurística de selección y la hiperheurística comparado con el número de objetos que requiere la mejor heurística simple para resolver cada instancia, mientras que el cuadro 5.14 compara contra el número de objetos que requiere la heurística simple en promedio. Los valores de ambos cuadros indican el porcentaje de instancias de prueba (del total de 270) en que se

requirió diferente número de objetos extras.

Cuadro 5.13: Número de objetos extra en los problemas de prueba respecto al resultado obtenido por la mejor heurística simple. Experimento III.

		Heurísticas de selección									
		FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HH**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
0	91.1	10.8	55.4	4.5	56.4	9.7	53.7	18.1	55.9	18.1	35.2
1	7.8	37.7	24.8	23.8	24.5	34.9	25.1	35.1	24.3	35.1	27.9
2	1.1	24.2	10.4	23.2	10.3	22.8	11.0	24.9	10.8	24.9	17.6
3		15.5	5.5	17.4	5.0	17.7	6.0	11.3	4.7	11.3	11.3
4		7.5	2.4	13.8	2.4	8.7	2.0	5.9	2.8	5.9	4.8
5		2.1	1.3	8.9	1.2	3.6	1.9	2.5	1.3	2.5	2.4
> 5		2.2	0.3	8.3	0.2	2.6	0.3	2.1	0.2	2.1	0.8

Cuadro 5.14: Número de objetos extra en los problemas de prueba respecto al promedio obtenido por las 40 heurísticas simples. Experimento III.

		Heurísticas de selección									
		FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HH**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
< -3	2.6		0.6		0.6		0.4		0.6		0.1
-3	6.7		2.3		2.5		2.3		2.1		0.6
-2	30.4	0.4	18.3		19.1	0.3	17.3	1.2	18.7	1.2	8.0
-1	50.0	13.8	43.2	1.5	43.2	10.0	43.7	22.9	44.0	22.9	36.4
0	10.4	53.2	22.5	30.6	21.7	50.9	22.5	48.6	21.4	48.6	30.6
1		20.6	8.8	31.4	9.1	23.7	8.8	17.2	9.2	17.2	14.7
2		7.5	3.3	19.4	3.2	8.8	3.9	5.9	3.1	5.9	6.9
3		3.1	0.6	12.2	0.5	4.3	0.9	3.1	0.8	3.1	2.5
> 3		1.5	0.2	4.9	0.1	2.0	0.2	1.1	0.1	1.1	0.2

5.5. Experimento IV

Este experimento intercambia los problemas de entrenamiento y prueba respecto al experimento III. Esto relaciona a los experimientos III y IV del mismo modo que los experimentos I y II (Ver cuadro 4.3).

Cuadro 5.15: Conjunto de reglas desarrolladas por el AG. Experimento IV.

Large Rect.	**Medium Rect.**	**Low Rect.**	**Large**	**Small**	**High**	**Short**	**Remain**	**Action**
0.84	-0.14	-0.07	1.11	1.20	0.75	0.70	0.59	6
0.10	1.94	-0.02	0.82	0.20	0.37	1.07	-0.07	15
0.10	1.94	0.13	0.70	0.45	0.77	0.93	0.29	25
0.01	0.48	-0.18	0.99	1.28	0.24	0.35	0.42	15
0.84	-0.14	-0.07	1.11	1.03	0.75	0.70	0.59	23
-0.47	0.02	1.32	1.70	1.12	0.78	-0.07	0.44	25
1.00	0.92	0.10	0.84	0.60	1.57	1.22	0.53	17
0.92	0.78	0.81	0.82	0.37	0.47	0.10	0.87	24

5.5.1. Ejecutar el AG con los problemas de entrenamiento para obtener una hiperheurística general

El cuadro 5.15 muestra la hiperheurística generada por el experimento IV, la cual estuvo conformada por ocho bloques diferentes. Es la hiperheurística con más reglas diferentes en los 4 experimentos. Aquí la única acción repetida es, nuevamente, la heurística 15. Tiene sentido que esta acción aparezca habitualmente en las hiperheurísticas generadas, pues fue la que solucionó mejor el 20.9 % de todas las instancias de entrenamiento y prueba.

5.5.2. Resolver problemas de prueba con la hiperheurística generada

Al igual que en los primeros tres experimentos, se generaron los cuadros 5.16 y 5.17, los cuales muestran el desempeño de las heurísticas de selección y de la hiperheurística generada por el AG al resolver las instancias de prueba.

5.6. Discusión general

Cada hiperheurística construida se analizó contra las heurísticas de selección generando dos cuadros comparativos. Al analizar cada uno de los 10 cuadros que representan el desempeño de las cinco hiperheurísticas construidas (dos en el experimento I y una en cada uno de los experimentos II, III y IV), es evidente la superioridad de la hiperheurística sobre cada una de las heurísticas de selección.

Cuadro 5.16: Número de objetos extra en los problemas de prueba respecto al resultado obtenido por la mejor heurística simple. Experimento IV.

		Heurísticas de selección									
		FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HH**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
0	87.8	11.3	56.5	4.7	57.7	10.4	54.9	16.5	57.5	16.5	34.5
1	10.7	33.9	23.7	24.2	22.9	31.5	25.3	38.0	23.4	38.0	32.7
2	1.1	26.2	10.5	21.1	10.6	25.5	10.2	22.7	10.0	22.7	15.4
3		13.7	4.3	16.4	4.2	15.1	4.2	10.8	4.4	10.8	9.0
4		8.7	3.3	15.6	3.2	9.6	3.8	7.0	3.2	7.0	4.6
5		3.5	1.4	9.5	1.3	4.5	1.4	2.3	1.3	2.3	2.9
> 5		2.6	0.2	8.5	0.2	3.3	0.3	2.7	0.2	2.7	0.9

Cuadro 5.17: Número de objetos extra en los problemas de prueba respecto al promedio obtenido por las 40 heurísticas simples. Experimento IV.

		Heurísticas de selección									
		FF	**FFD**	**FFI**	**Filler**	**NF**	**NFD**	**BF**	**BFD**	**WF**	**DJD**
Obj.	**HH**	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40
< -3	2.2		0.6		0.6		0.4		0.6		0.1
-3	6.6		2.0		2.3		2.0		2.1		1.5
-2	29.5	0.5	18.9		19.6	0.4	18.4	1.5	19.8	1.5	7.6
-1	49.8	12.6	44.0	2.0	43.1	9.8	44.2	20.9	43.4	20.9	38.0
0	11.4	49.5	20.9	26.8	21.1	47.7	21.1	49.4	20.8	49.4	29.7
1	0.4	22.6	9.3	31.7	9.2	24.4	9.1	16.1	9.0	16.1	13.6
2		8.8	3.2	21.2	3.2	10.0	3.6	7.1	3.4	7.1	6.4
3		3.8	0.9	11.8	0.7	4.2	1.0	3.6	0.7	3.6	2.4
> 3		2.2	0.1	6.4	0.1	3.6	0.2	1.3	0.1	1.3	0.8

El cuadro 5.18 resume, con un solo valor, el desempeño de la hiperheurística de cada experimento. En promedio, utilizar la hiperheurística reduce en más de un objeto la solución propuesta por las 40 heurísticas simples.

Cuadro 5.18: Número de objetos extra utilizados por la hiperheurística en los problemas de prueba respecto al promedio obtenido por las 40 heurísticas simples.

Objetos extras	
Experimento I, 1^a*réplica*	-1.63
Experimento I, 2^a*réplica*	-1.66
Experimento II	-1.11
Experimento III	-1.41
Experimento IV	-1.37

El experimento I fue el experimento cuya hiperheurística mostró mejor el desempeño relativo al desempeño de las heurísticas simples. A grandes rasgos, los problemas de entrenamiento de este experimento fueron un tanto menos irregulares que las instancias de prueba. Según el cuadro 4.2, los problemas de entrenamiento tuvieron un lado mínimo promedio mayor, una menor razón (*lado mayor*)/(*lado menor*) y un mayor número de rectángulos iniciales. Estas características con que fueron generados hacen que, en general, las piezas de las instancias de entrenamiento tiendan más a la rectangularidad. De la misma manera los problemas de prueba de este experimento son, en general, más irregulares que los de entrenamiento.

El experimento II es la antítesis del experimento I. Tiene el desempeño más modesto de los cuatro experimentos y dado que intercambia los problemas de entrenamiento y prueba con el experimento I, la hiperheurística generada por el experimento II se prueba en instancias algo menos irregulares que con las que fue entrenada.

Los problemas de entrenamiento y prueba tenían un grado de dificultad muy similar entre sí en los experimentos III y IV. En estos dos experimentos los tipos de instancias generadas fueron divididos por igual en problemas de entrenamiento y prueba. Esto permitió al AG entrenar con problemas muy similares con los que luego sería probado. Esto explica el desempeño promedio muy similar de las heurísticas generadas por estos dos experimentos.

Se observa el hecho de que las hiperheurísticas tengan mejor desempeño (relativo al desempeño de las heurísticas simples) en cuanto mayor sea la irregularidad de los problemas en que se prueba.

El tiempo de cómputo empleado en resolver las 541 instancias con las 40 heurísticas simples es de aproximadamente de 12 horas. Esto es lo que corresponde a la primera

fase del modelo de solución (Ver figura 3.1). El código se programó de tal modo, que genera un archivo de salida con la aptitud de las heurísticas simples en cada instancia de entrenamiento. Este archivo es necesario en la segunda fase del modelo para calcular la función de evaluación de los individuos del algoritmo genético (Ver ecuaciones 2.3 y 2.4). De esta manera, cuando el conjunto de instancias disponibles se clasifica de manera diferente en los dos subconjuntos de entrenamiento y prueba, es necesario repetir el cómputo. En la serie de experimentos realizados en esta investigación, esta fase se ejecutó una vez para los experimentos I y II y también se ejecutó una segunda vez para los experimentos III y IV.

Respecto a la fase 2 del modelo de solución, el tiempo empleado por el equipo de cómputo para ejecutar el algoritmo genético y generar cada hiperheurística fue similar en cada uno de los cuatro experimentos, quedando alrededor de 11 horas por hiperheurística. El cuadro 5.19 muestra el tiempo empleado en cada experimento y réplica y se observa, como era de esperarse, alta correspondencia entre el tiempo empleado con el tamaño de los problemas de entrenamiento (medido en número de piezas). De hecho, con base en los experimentos realizados, se presenta un índice de correlación de 0.95 entre tamaño promedio de instancias de entrenamiento y el tiempo empleado para generar la hiperheurística. Si el tamaño de las instancias fuera medida en función del número de objetos en su solución óptima, también se observaría una relación similar con el tiempo requerido para generar una hiperheurística.

Cuadro 5.19: Tiempo requerido para la generación de cada hiperheurística en relación al tamaño promedio de los problemas de entrenamiento.

Experimento	Tiempo requerido (horas)	Tamaño promedio de instancia de entrenamiento (número de piezas)
Experimento I, 1^a*réplica*	9.1	42.0
Experimento I, 2^a*réplica*	9.5	42.0
Experimento II	13.1	49.1
Experimento III	10.8	45.6
Experimento IV	12.2	45.6

La fase 3 del modelo de solución, que consiste en resolver cada instancia de prueba con la hiperheurística, es sensiblemente la más rápida. Su duración es apenas de unos cuantos minutos en todas las instancias de prueba.

Resolver una instancia con la hiperheurística generada es mucho más rápido que resolverla con las 40 heurísticas simples para después elegir la mejor y usar ésta como mejor solución. El tiempo en que una hiperheurística resuelve una instancia depende de

la rapidez de las acciones que le corresponda ejecutar, ya que el tiempo de ejecución de cada acción varía, pero en promedio, puede decirse que la hiperheurística resuelve una instancia en aproximadamente 1/40 del tiempo que llevaría encontrar la mejor solución de las heurísticas simples. Esta diferencia de tiempo hace muy conveniente el uso de la hiperheurística para resolver una instancia ya que la hiperheurística por lo general igualará el desempeño de la mejor heurística simple (Ver cuadros 5.6, 5.10, 5.13 y 5.16). En conclusión, si se cuenta con un conjunto de instancias y suficiente tiempo para entrenar una algoritmo genético conforme al modelo propuesto, el generar una hiperheurística representa una buena inversión de tiempo ya que con ella se podrá encontrar una muy buena solución a cada nueva instancia que se presente.

5.7. Resumen

En este capítulo se dieron a conocer los resultados obtenidos en cada uno de los experimentos realizados haciendo un análisis de los mismos. Los resultados logrados fueron comparados con los producidos por las heurísticas simples. Finalmente, se elaboró un análisis general del desempeño mostrado por el enfoque propuesto donde se discutió acerca de la eficacia de la hipótesis planteada inicialmente.

Capítulo 6

Conclusiones

En este capítulo se resumen las principales contribuciones y conclusiones generadas por la presente investigación. Así también, se incluyen algunas ideas con las cuales podría extenderse el trabajo realizado.

6.1. Conclusiones y contribuciones

La investigación aquí presentada tuvo el objetivo de diseñar e implementar un algoritmo para la generación de hiperheurísticas que resuelvan instancias del problema de empacado irregular de material en dos dimensiones. Esto se logró al adaptar e implementar 40 heurísticas simples y posteriormente utilizarlas una y otra vez en un Algoritmo Genético para resolver un conjunto de problemas de entrenamiento. La población del Algoritmo Genético se evolucionó hasta que encontró una serie de reglas que indican las condiciones de una instancia que hacen más apropiado el uso de determinada heurística simple. A este conjunto de reglas se le llamó hiperheurística.

Se diseñaron y corrieron cuatro diferentes experimentos para probar la bondad del método propuesto. Los resultados obtenidos por las hiperheurísticas generadas al resolver una serie de problemas de prueba fueron bastante satisfactorios en todos los experimentos demostrando que, en promedio, es mejor el uso de una hiperheurística que el de una heurística simple, por buena que esta sea. Con estos resultados se comprueba la validez de la hipótesis de investigación planteada en un inicio.

Las contribuciones del presente trabajo son:

1. El diseño e implementación de nuevas heurísticas de acomodo, variantes de las que actualmente se describen en la literatura.

2. La definición de una manera de caracterizar el estado de una instancia de empacado irregular de material en dos dimensiones.

3. La comprobación de la efectividad del empleo de un algoritmo genético con cromosomas de longitud variable para el desarrollo de hiperheurísticas genéricas para el problema de empacado irregular de material en dos dimensiones.

6.2. Trabajo futuro

En investigaciones posteriores, el algoritmo propuesto podría evolucionar para:

1. Mejorar las heurísticas simples de selección y acomodo, descartando o modificando aquellas que no parezcan ser buenas en prácticamente ningún grupo de instancias.

2. Probar otras representaciones del estado del problema. Por ejemplo, los bloques de los cromosomas del AG podrían incluir valores relacionados con el número de lados de las piezas.

3. Permitir la rotación de piezas tipo 'espejo' (o reflexión).

4. Permitir otro tipo de figuras irregulares de dos dimensiones; por ejemplo: polígonos cóncavos, figuras que incluyan arcos o curvas, figuras que incluyan orificios.

5. Probar la generación de hiperheurísticas en otros problemas según la clasificación de Dyckhoff [13]. Por ejemplo:

   a) Problemas donde los objetos no son todos idénticos y/o regulares.

   b) Problemas donde el número de objetos es finito y se maximiza el área de las piezas a cortar.

   c) Problemas donde se tiene un sólo objeto y se minimiza la longitud empleada en recortar todas las piezas.

   d) Problemas con figuras en tres dimensiones.

6. Generalizar el modelo de solución propuesto para resolver otros tipos de problemas de optimización.

Bibliografía

[1] S. Anand, C. McCord, R. Sharma, and T. Balachander. An integrated machine vision based system for solving the nonconvex cutting stock problem using genetic algorithms. *Journal of Manufacturing Systems*, 18(6):396, 1999.

[2] S. Baase and A. V. Gelder. *Algoritmos Computacionales, Introducción al análisis y diseño*. Addison Wesley, tercera edición, 2000.

[3] J. A. Bennell and K. A. Dowsland. Hybridising tabu search with optimisation techniques for irregular stock cutting. *Management Science*, 47(8):1160, agosto 2002.

[4] J. A. Bennell, K. A. Dowsland, and W. B. Dowsland. The irregular cutting-stock problem - a new procedure for deriving the no-fit polygon. *Computers & OR*, 28(3):271–287, 2001.

[5] E. K. Burke, E. Hart, G. Kendall, J.Ñewall, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern research technolology. In *Handbook of Metaheuristics*, pages 457–474. Kluwer Academic Publishers, 2003.

[6] E. K. Burke, R. Hellier, G. Kendall, and G. Whitwell. A new bottom-left-fill heuristic algorithm for the 2D irregular packing problem. Aceptada para publicación en Operations Research, 2006.

[7] E. K. Burke and G. Kendall. Implementation and performance improvement of the evaluation of a two dimensional bin packing problem using the no fit polygon, 1999.

[8] A. J. Crispin, P. Clay, G. E. Taylor, T. Bayes, and D. Reedman. Genetic algorithms applied to leather lay plan material utilization. In *Proceedings of the Institution of Mechanical Engineers*, volume 217, 12, page 1753. ProQuest Science Journals, 2003.

[9] K. Deb and D. E. Goldberg. mGA in C: A messy genetic algorithm in C, 1991.

[10] K. A. Dowsland and W. B. Dowsland. Solution approaches to irregular nesting problems. *European Journal of Operational Research*, 84:506–521, 1995.

[11] K. A. Dowsland, W. B. Dowsland, and J. A. Bennell. Jostling for position: local improvement for irregular cutting patterns. *Journal of the Operational Research Society*, 49(6):647–658, 1998.

[12] K. A. Dowsland, S. Vaid, and W. B. Dowsland. An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research*, 141(2):371–381, September 2002.

[13] H. Dychoff. A topology of cuting and packing problems. *European Journal of Operation Research*, 44:145–159, 1990.

[14] C. J. Farías. Hiperheurísticas mediante un algoritmo genético con longitud variable para resolver problemas de corte de material en dos dimensiones. Master's thesis, Tecnológico de Monterrey, mayo 2006. Asesor: Dr. Hugo Terashima.

[15] K. Fujita, S. Akagji, and N. Kirokawa. Hybrid approach for optimal nesting using a genetic algorithm and a local minimisation algorithm. In *Proceedings of the 19th Annual ASME Design Automation Conference, Part 1 (of 2)*, volume 65, part 1, pages 477–484, Albuquerque, NM, USA, 1993.

[16] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[17] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[18] D. E. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: motivation, analysis and first results. *Complex Systems*, 3:493–530, 1989.

[19] A. M. Gomes and J. F. Oliveira. A 2-exchange heuristic for nesting problems. *European Journal of Operations Research*, 141:359–370, 2002.

[20] R. B. Grinde and T. M. Cavalier. A new algorithm for the minimal-area convex enclosure problem. *European Journal of Operation Research*, 84(3):522–538, 1995.

[21] M. Hifi and R. MHallah. A best-local position procedure-based heuristic for two-dimensional layout problems. *Studia Informatica Universalis, International Journal on Informatics*, 2(1):33–56, 2002.

[22] M. Hifi and R. MHallah. A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes. *International Transactions in Operational Research*, 10:195–216, 2003.

[23] J. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[24] L. Hu-yao and H. Yuan-jun. Algorithm for 2D irregular-shaped nesting problem based on the nfp algorithm and lowest-gravity-center principle. *Journal of Zhejiang University SCIENCE A*, 7(4):570–576, 2006.

[25] L. Hu-yao and H. Yuan-jun. Nfp-based nesting algorithm for irregular shapes. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 963–967, New York, NY, USA, 2006. ACM Press.

[26] J. F. Oliveira, A. M. Gomes, and J. S. Ferreira. Topos - a new constructive algorithm for nesting problems. *OR Spektrum*, 22:263–284, 2000.

[27] P. Ross, J. G. Marín-Blázquez, S. Schulenburg, and E. Hart. Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In *Proceedings of GECCO 2003. Lecture Notes in Computer Science*, volume 2724, pages 1295–1306. Springer-Verlag, 2003.

[28] H. Terashima-Marín, C. J. F. Zárate, P. Ross, and M. Valenzuela-Rendón. A ga-based method to produce generalized hyper-heuristics for the 2D-regular cutting stock problem. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 591–598, New York, NY, USA, 2006. ACM Press.

[29] A. Uday, E. D. Goodman, and A. A. Debnath. Nesting of irregular shapes using feature matching and parallel genetic algorithms. In E. D. Goodman, editor, *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 429–434, San Francisco, California, USA, 2001.

[30] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. Faculty of Economics and Management, Otto von Guericke University, Magdeburg. Working Paper No. 24, Last Revision: 2006-01-16, 2006.

[31] T.-H. Wu, J.-F. Chen, C. Low, and P.-T. Tang. Nesting of two-dimensional parts in multiple plates using hybrid algorithm. *International Journal of Production Research*, 41(16):3883–3900, 2003.