INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS MONTERREY
PROGRAMA DE GRADUADOS DE LA DIVISION DE ELECTRONICA, COMPUTACION, INFORMACION Y COMUNICACIONES


# Hiper-Heurística a través de Sistemas de Clasificadores para solucionar problemas de corte de material en dos dimensiones

T E S I S

MAESTRO EN CIENCIAS

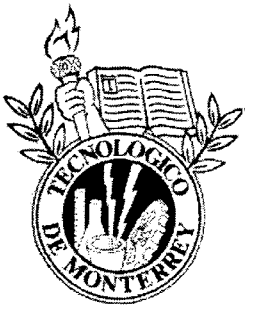
Especialidad en Sistemas Inteligentes

Por

Edgardo Javier Flores Alvarez

Diciembre 2004

# Hiper-Heurística a través de Sistemas de Clasificadores para solucionar problemas de corte de material en dos dimensiones




T E S I S

**Maestro en Ciencias**

especialidad en

**Sistemas Inteligentes**

**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Por**

**Edgardo Javier Flores Álvarez**

Diciembre 2004

# Hiper-Heurística a través de Sistemas de Clasificadores para solucionar problemas de corte de material en dos dimensiones


T E S I S

**Maestro en Ciencias**

especialidad en

**Sistemas Inteligentes**

**Instituto Tecnológico y de Estudios Superiores de Monterrey**

Por

**Edgardo Javier Flores Álvarez**

Diciembre 2004

# Hiper-Heurística a través de Sistemas de Clasificadores para solucionar problemas de corte de material en dos dimensiones

TESIS

**Maestro en Ciencias**

especialidad en

**Sistemas Inteligentes**

**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Por**

**Edgardo Javier Flores Álvarez**

Diciembre 2004

# Hiper-Heurística a través de Sistemas de Clasificadores para solucionar problemas de corte de material en dos dimensiones

por

**Edgardo Javier Flores Álvarez**

**Tesis**

Presentada al Programa de Graduados en Electrónica, Computación, Información y Comunicaciones

como requisito parcial para obtener el grado académico de

**Maestro en Ciencias**

especialidad en

**Sistemas Inteligentes**

**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Campus Monterrey**

Diciembre de 2004

A Dios, por su amor y por darme la oportunidad de seguir adelante.
A mi padre, por mi educación, valores inculcados y todo del amor que me ha brindado.
A mi madre, por su amor y confianza, que han sido factores importantes para el éxito en mis estudios.

# Reconocimientos

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

Al Dr. Hugo Terashima Marín por su apoyo incondicional en el área de sistemas inteligentes, ya que con su asesoría centró las bases de la presente investigación.

Al Dr. Manuel Valenzuela Rendón, por su contribución como sinodal en mi examen de grado.

Al Dr. Horacio Martínez, por su contribución como sinodal en mi examen de grado.

A todas las personas que de manera directa o indirecta hicieron posible la elaboración de esta investigación.

EDGARDO JAVIER FLORES ÁLVAREZ

*Instituto Tecnológico y de Estudios Superiores de Monterrey*
*Diciembre 2004*

## Capítulo 1

# Introducción

El tema de corte de material ha sido estudiado ampliamente en los últimos años debido a la gran cantidad de aplicaciones que puede tener, algunas de ellas son: empaque de materiales, carga de vehículos y contenedores limitados por la capacidad de carga, partición de problemas, calendarización de tareas en espacios limitados de tiempo, organización de localidades de memoria, organización de horarios, y corte de materiales en donde la materia prima puede tener una, dos o tres dimensiones, por ejemplo cable, madera, piel o papel.

El problema de corte de materiales también ha sido de suma importancia en cuanto a la teoría, pues ha servido de base para muchas investigaciones en el análisis del comportamiento de los algoritmos de aproximación. Esto involucra, el determinar el radio de desempeño del peor caso, identificar el límite inferior en el mejor desempeño posible y el analizar el comportamiento del caso promedio de distintos algoritmos de optimización [19].

Se ha observado un rápido desarrollo así como una gran investigación en este tema. Han surgido un gran número de documentos publicados, por diferentes disciplinas como: administración, informática, ingeniería, inteligencia artificial e investigación de operaciones [11].

En el problema clásico de corte de materiales en dos dimensiones, se tiene una secuencia de piezas, cada una de tamaño definido y se deben acomodar en patrones de corte dentro de un mínimo número de objetos de capacidad limitada. Generalmente se divide en tres grupos: Corte en una dimensión, en dos y en tres dimensiones. Aunque en realidad estos grupos están muy ligados entre sí.

Para problemas combinatorios pequeños, se han desarrollado métodos determinísticos, como programación lineal. Dado que estos métodos son exactos, encuentran la solución óptima. Sin embargo para problemas de mayor complejidad, como la mayoría de los problemas de corte, el espacio de búsqueda es mayor debido al gran número de combinaciones posibles. En este caso la búsqueda exhaustiva de posibles soluciones no es una opción práctica pues el espacio de búsqueda puede ser demasiado grande y no es posible calcular la solución óptima en un tiempo computacional razonable. El problema de corte en dos dimensiones es un problema de dificultad NP [13], es decir, no

es posible encontrar la solución óptima en un tiempo polinomial. Actualmente se han estudiado y utilizado varias heurísticas y algoritmos de aproximación que garantizan encontrar una solución cercana a la óptima para un problema dado, sin embargo no ha sido posible encontrar un algoritmo que genere una solución confiable para todas las instancias del problema y menos aún uno que encuentre una solución óptima para todos los problemas.

En esta tesis, se toma la idea básica presentada por Ross et al. [22] para solucionar problemas de empacado de contenedores en una dimensión, en esa investigación, se representa un nueva manera de utilizar algoritmos evolutivos, en lugar de utilizarlos para descubrir una solución para un problema específico, se trata de utilizarlos para fabricar un proceso de solución aplicable a muchos problemas. En este trabajo, se utilizó un sistema de clasificadores basado en la precisión de la predicción (XCS), para aprender un conjunto de reglas que asocian características de los estados del problema con diferentes heurísticas. Esto es conocido como el concepto de hiper-heurística.

El término hiper-heurística se define como una heurística de alto nivel que controla varias heurísticas de menor nivel [7]. Esta debe escoger cuándo y dónde aplicar cada una de las heurísticas de menor nivel de acuerdo a un problema de optimización dado [9]. En cada punto de decisión, la hiper-heurística debe decidir cuál heurística de menor nivel debe aplicar, basándose en cierta información del problema dado como tiempo de CPU, valores de una función objetivo, entre otros. La selección de la heurística de menor nivel es dinámica y depende de la parte del espacio de búsqueda siendo explorado en ese momento. Con el fin de hacer una selección importante de la heurística de menor nivel, la hiper-heurística debe de utilizar algún mecanismo de aprendizaje.

Este concepto se ha utilizado para resolver problemas de calendarización de eventos (Terashima et al.) [24]. Para producir soluciones que dependen de características particulares de los problemas a resolver, se utiliza un AG para evolucionar combinaciones de heurísticas y encontrar la correcta.

El presente documento muestra un método de búsqueda de una solución al problema de corte de material, guiado por la calidad de las soluciones parciales propuestas por heurísticas definidas bajo esquemas de sistemas de clasificadores. Los clasificadores son evolucionados mediante un algoritmo genético, con el fin de obtener clasificadores que propongan heurísticas, que en conjunto, representen la mejor solución posible al problema de corte.

## 1.1. Definición del Problema

Dado un conjunto $L = (a_1, a_2, ...a_n)$ de piezas a ser cortadas, cada una de tamaño $s(a_i)\epsilon(0, X]$, de un conjunto de objetos de material de tamaño $X$, el problema de corte de materiales 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 (Figura 1.1). Este problema de dificultad NP, puede complicarse dependiendo de las variables involucradas en este, como el número de lados de las figuras, las diferentes formas que puedan tomar (los ángulos de rotación permitidos), la cantidad máxima de piezas que puedan cortarse de un objeto, el número de dimensiones, y el color que deban tener. Debido a la gran diversidad de problemas y aplicaciones, Dyckhoff [11] propuso una clasificación sistemática de problemas de corte y empacado.

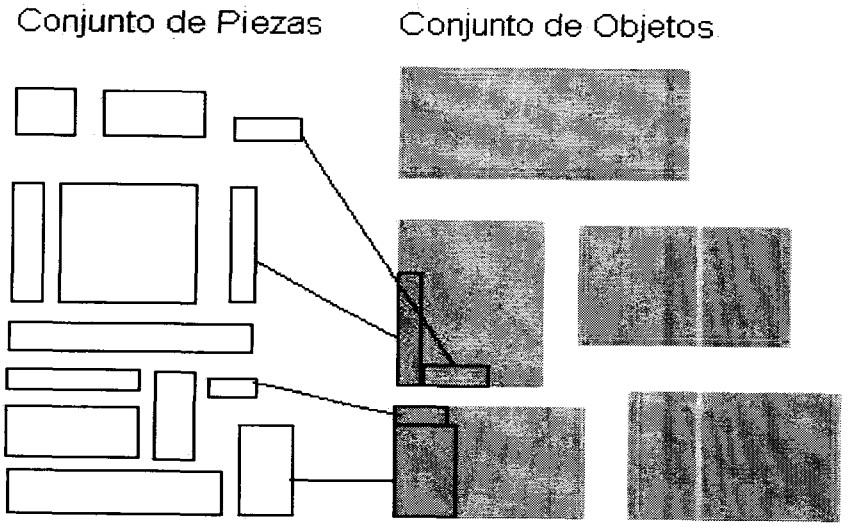



Figura 1.1: Esquema del Problema de Corte.

Los problemas de corte y empacado son problemas combinatorios de muy alta complejidad. Muchas aplicaciones importantes de problemas de corte son de tipo NP-Completo. Existen muchas aproximaciones que se utilizan cuando se quiere resolver estos problemas. Sin embargo no existe un algoritmo de orden polinomial que encuentre la solución óptima.

Una manera de resolver problemas de optimización NP-Completos es mediante el uso de algoritmos rápidos (que corren en tiempo polinomial) que no garantizan dar la mejor solución, pero dan una cercana a la óptima, a estos algoritmos normalmente se les llama algoritmos heurísticos.

En algunas aplicaciones de corte de material, soluciones aproximadas son suficientes, especialmente cuando el tiempo requerido para encontrar la solución óptima es considerado. Estas estrategias o heurísticas, normalmente son sencillas y directas, a pesar de eso para algunos problemas ofrecen sorprendentemente buenos resultados. Muchas de ellas son heurísticas concisas. Bajo este esquema se han encontrado algoritmos que funcionan bien para ciertas entradas.

Para reducir el espacio de búsqueda, en el artículo de P.Ross et al. [21] se propone utilizar algoritmos genéticos para combinar opciones de heurísticas dentro algún proceso secuencial de decisiones. Argumenta, que si las heurísticas propuestas son individualmente aceptables, la combinación de estas producirá un resultado de mejor calidad que el obtenido por cualquiera de ellas por separado.

El problema en particular que se pretende solucionar se enfoca a la distribución de figuras rectangulares de tamaño variable, con posibilidad de rotar 90 grados, dentro de una figura de mayor tamaño, también de forma rectangular.

La solución propuesta a este problema puede ayudar principalmente a:

1. Solucionar los problemas que involucran corte o almacenamiento de materiales en dos dimensiones.

2. Mostrar que los algoritmos genéticos con heurísticas son más eficientes que los algoritmos genéticos por sí solos.

3. Obtener heurísticas complejas y eficientes a partir de heurísticas simples.

4. Demostrar que los sistemas de clasificadores son una buena opción para representar y solucionar problemas de corte.

5. Comprobar que el evolucionar sistemas de clasificadores mediante algoritmos genéticos es una buena opción para encontrar las mejores heurísticas en instancias del problema, y que estas heurísticas en conjunto representan una hiper-heurística confiable y precisa.

Para mostrar el funcionamiento del algoritmo propuesto en esta tesis, se requiere de problemas de corte de materiales, que proporcionen el número de piezas a ser cortadas, así como las dimensiones de cada una de estas. También deberán contener las dimensiones de los objetos de materia prima de donde deberán ser cortadas las piezas.

## 1.2. Motivación

Los algoritmos genéticos normalmente son utilizados para resolver problemas individuales y mientras están corriendo no hay manera fácil de saber las propiedades de la salida. También puede que les tome mucho más tiempo resolver un problema que a algún otro método no evolutivo. Por lo tanto, no es sorprendente que en muchas aplicaciones prácticas, se prefiera utilizar heurísticas simples. Sin embargo, se ha probado que existen heurísticas que son mejores que otras para ciertos problemas, incluso hay heurísticas que son mejores que otras para algunas partes del mismo problema, por lo tanto es difícil seleccionar que heurística utilizar para un problema nuevo.

La motivación surge con el deseo de utilizar a los algoritmos genéticos para aprender un proceso de solución, construido de heurísticas simples, en lugar de utilizarlos para resolver problemas individuales. Este proceso de solución puede consistir en utilizar una heurística inicialmente, pero conforme va cambiando el problema puede que alguna otra heurística sea más apropiada.

Esta idea ha sido utilizada para resolver problemas de calendarización de tareas [24], y para empacado de contenedores en una dimensión [22], ofreciendo muy buenos resultados.

## 1.3. Objetivos

El objetivo general de la tesis es resolver el problema de corte de materiales rectangulares en dos dimensiones, utilizando un esquema que combina algoritmos genéticos, hiper-heurística y sistemas de clasificadores.

De lo anterior se pueden establecer los siguientes objetivos particulares:

- Definir el conjunto de heurísticas más apropiado, que ayude a solucionar de manera óptima el problema de corte de materiales.
- Representar el ambiente de corte de materiales en el sistema de clasificadores de manera estándar para que pueda ser utilizado independientemente del problema que se quiera resolver.
- Encontrar la mejor combinación de heurísticas, que en conjunto resuelvan de manera óptima el problema particular de corte en dos dimensiones.

## 1.4. Hipótesis

La hipótesis principal es que el uso de sistemas de clasificadores evolucionados por algoritmos genéticos para encontrar el proceso de solución representa una buena opción para la solución de problemas de corte en dos dimensiones. Con el fin de probar lo anterior se presentan las siguientes preguntas de investigación:

1. ¿El uso sistema de clasificadores es una buena técnica para obtener una combinación de heurísticas que representen una solución confiable?
2. ¿Cuál es la representación más adecuada del problema en el sistema de clasificadores?
3. ¿Cuál es el conjunto de heurísticas más apropiado para encontrar la solución al problema de corte en dos dimensiones?

## 1.5. Contribución

El aporte de la presente investigación consiste en mostrar una herramienta útil para resolver distintos problemas de corte de material en dos dimensiones. Esta aproximación esta basada en la representación del ambiente y su relación con heurísticas de selección y acomodo de piezas, en un sistema de clasificadores basado en la precisión de la predicción (XCS). Esta herramienta permitirá solucionar problemas combinatorios complejos de corte de materiales en dos dimensiones.

Además, en función de la hipótesis esta tesis muestra las siguientes contribuciones:

1. La programación de heurísticas de selección enfocadas al problema de corte de materiales en dos dimensiones

2. La programación de heurísticas de acomodo enfocadas al problema de corte de materiales en dos dimensiones

3. La representación de problemas de corte de materiales como un ambiente para el sistema de clasificadores XCS.

4. La codificación de los clasificadores para que representen las condiciones del problema y las acciones que resuelvan de manera eficiente el problema bajo esas condiciones.

## 1.6. Organización de la Tesis

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

El capítulo 2 muestra los antecedentes, en donde se tratan brevemente las investigaciones sobre hiper-heurística. Se explica el concepto de algoritmos genéticos, heurística y sistemas de clasificadores.

El capítulo 3 explica la manera de operar del sistema de clasificadores XCS que fue adoptado para el desarrollo de esta tesis.

El capítulo 4 describe el modelo de solución utilizado, se muestra la representación del problema de corte de materiales y la relación de este con las heurísticas simples.

El capítulo 5 presenta el desempeño del sistema de clasificadores en la construcción de hiper-heurísticas, tratando ejemplos de la literatura.

El capítulo 6 presenta, por último, las conclusiones y algunas sugerencias para trabajos futuros.

## Capítulo 3

# Sistema de Clasificadores XCS

En este capítulo se describe de manera detallada el sistema de clasificadores XCS que es utilizado en esta investigación. Se explica la estructura de los clasificadores, así como sus parámetros y la manera en que son evaluados de acuerdo a S.W. Wilson [28]. Se mencionan algunas ventajas de utilizar este sistema de clasificadores sobre los sistemas de clasificadores comunes.

## 3.1. Antecedentes

Debido al uso no restringido de AG's, los sistemas de clasificadores LCS (Learning Classifier Systems) por si solos, presentan el problema de encontrar reglas que concuerdan con partes del problema totalmente diferentes y por lo tanto, la combinación de tales clasificadores normalmente lleva a clasificadores sin sentido. Se han realizado algunas investigaciones al respecto, Booker [6] propone aplicar procesos de matanza en el AG en el conjunto de concordancia ($M$). También Smith y Valenzuela-Rendón [23] investigaron un método de comparición que previniera los problemas mencionados [26].

El sistema de clasificadores XCS resuelve este problema utilizando AG's en el conjunto de acciones, una función para medir el desempeño de éstas y aprendizaje por refuerzo basado en Q-learning.

La ventaja del sistema de clasificadores XCS es la aproximación que realiza está basada en el desempeño. El resultado es un sistema de clasificadores que no solo evoluciona a los clasificadores para mejores acciones, si no que hace un mapeo completo del problema. Es decir, XCS evoluciona un modelo de comportamiento que determina la calidad de cada acción posible en todas las situaciones o instancias del problema del ambiente encontrado. Además, XCS adapta el mecanismo de Q-learning para aprendizaje por refuerzo.

Tradicionalmente en los sistemas de clasificadores el parámetro de *strength*, sirve tanto como para la predicción de la futura rentabilidad o paga como para la aptitud del clasificador para el algoritmo genético. Sin embargo, la paga predecida puede representar a la aptitud de manera inadecuada, por ejemplo, un algoritmo con mala predicción,

puede que en cierto momento sea el mejor para el ambiente dado. En el sistema de clasificadores XCS, cada clasificador mantiene una predicción de la paga esperada, pero la aptitud del clasificador no está dada por la predicción, mas bien, la aptitud es un valor separado, basado en la medida de la precisión de la predicción, en lugar de ser la predicción en sí.

Con la aptitud basada en la precisión, en combinación con el algoritmo genético, resulta una población que tiende a formar un mapeo completo y exacto X x A => $P$ desde entradas y acciones hasta predicciones de la paga.

## 3.2. Medición de la aptitud

En muchos clasificadores, el parámetro de fuerza de un clasificador estima la paga que el clasificador recibirá cuando se satisface su condición, y su acción es escogida por el sistema. La fuerza por lo tanto es importante, pues es utilizada para escoger a la acción mas rentable y para medir la aptitud para los componentes de descubrimiento del algoritmo genético, es decir, los clasificadores con mayor fuerza tienen mayor probabilidad de ser seleccionados para reproducción y mutación.

Basar la aptitud en la fuerza es razonable, sin embargo hay varios problemas:

1. Diferentes nichos del ambiente, normalmente tienen diferentes niveles de pago [5], nicho significa un conjunto de estados del ambiente. Para prevenir que los clasificadores en nichos de paga muy alta invadan a toda la población, es necesario implementar una técnica de compartición, en la cual la paga es dividida entre clasificadores activos en lugar de darle a cada uno el valor completo (un análisis completo es presentado por Horn et. al. [18].)

2. La técnica de compartición elimina los efectos de invasión, pero entonces, la fuerza de un clasificador ya no predice la paga, en lugar de eso, el total de la paga compartida (entre los clasificadores que concuerdan con la situación y tiene la misma acción) predice la paga. La división de la predicción se vuelve problemática, pues un clasificador dado, con un valor de fuerza, frecuentemente es involucrado en muchos conjuntos, y así el significado del valor de la fuerza no es claro.

3. El algoritmo genético no puede distinguir a un clasificador preciso con paga moderada sobre un conjunto de clasificadores que tienen la misma paga en promedio y por lo tanto tiene que adivinar.

4. Los sistemas de clasificadores utilizan símbolos "don't care" (#) en la sintaxis de sus condiciones y por lo tanto permite la formación de generalizaciones. Sin embargo, bajo el concepto de aptitud conforme al a paga, parece no haber una tendencia clara, o razón teórica para que generalizaciones precisas evolucionen.

Debido a los problemas anteriores, parece razonable pensar que existe una base más apropiada para la aptitud del clasificador en lugar de ser solo el pago. Una primera aproximación se puede pensar a partir de los últimos dos problemas: si el pago estimado no distingue entre clasificadores precisos o generales, ¿porqué no que el pago sea la precisión?. El XCS propone el cambio de aptitud basada en el pago a aptitud basada en la precisión de la predicción. En el sistema de clasificadores XCS el parámetro fuerza se divide en tres:

1. Predicción: El promedio de el pago recibido
2. Error de la predicción: Promedio de la medida del error en el parámetro de la predicción
3. Aptitud: Una función inversa del error de la predicción

La predicción se utiliza al seleccionar las acciones y la aptitud es usada en el algoritmo genético.

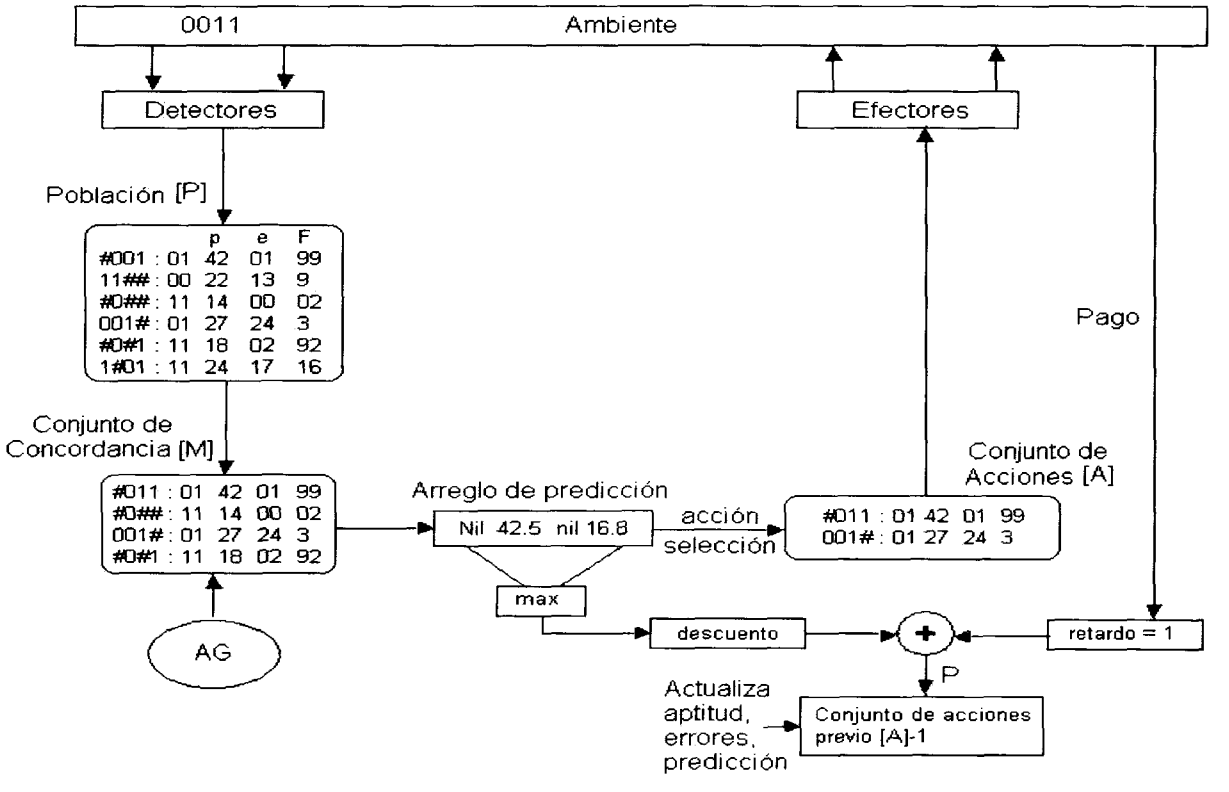



Figura 3.1: Esquema del XCS con el AG en conjunto de concordancia [M].

La figura 3.1 es una representación general del sistema, el cual se muestra en una interacción con el ambiente a través de detectores para sensar las entradas y efectores para monitorear las acciones. Además, el ambiente provee una paga en ciertos tiempos.

La caja etiquetada $[P]$ contiene la población de clasificadores y muestra algunos clasificadores ejemplo, la parte izquierda de cada clasificador consiste de una sola condición, la parte derecha codifica una acción. Los parámetros asociados con cada clasificador son predicción, error de la predicción y aptitud simbolizados por $p$, $\epsilon$ y $F$ respectivamente. La población tiene un tamaño máximo $N$ y puede ser inicializada de diferentes maneras: $N$ clasificadores generados de manera aleatoria, con semillas de clasificadores potencialmente útiles o sin clasificadores. Los valores iniciales de $p$, $\epsilon$ y $F$ pueden ser inicializados de manera arbitraria

## 3.3. Estructura de los clasificadores en el sistema XCS

Como todos los LCS, XCS interactúa con el ambiente, percibiendo situaciones $\sigma$, codificadas en código binario de longitud $L$, ejecutando acciones $\alpha$, y encontrando retroalimentaciones escalares $\rho$. De manera similar, el conocimiento en XCS es representado por una población [P] de clasificadores. La población en XCS es de longitud fija $N$. Cada clasificador es combinado con varios parámetros adicionales. Los clasificadores en XCS están estructurados de la siguiente manera:

- La parte de la condición $C$ especifica el subconjunto de situaciones en las que el clasificador es aplicable.
- La parte de la acción/clasificación $A$ especifica la acción/clasificación del clasificador.
- La retroalimentación de predicción $p$ estima la paga promedio encontrada después de la ejecución de la acción especificada.
- El error de predicción $\epsilon$ estima el error actual de $p$.
- El desempeño $F$ es una medida de la exactitud de $p$ con respecto a todos los clasificadores compitiendo.
- La experiencia $exp$ almacena qué tan seguido han sido actualizados los parámetros del clasificador.
- La estampa de tiempo $ts$ es el tiempo que el clasificador estuvo en el conjunto en donde el AG fue aplicado.
- El tamaño del conjunto de acciones $as$ estima el tamaño del conjunto de acciones al cual pertenece el clasificador.

- La numerosidad *num* refleja cuantos micro-clasificadores (clasificadores usuales) representa este macroclasificador.

La parte de la condición es codificada como una cadena dentro del alfabeto $\{0, 1, \#\}$ de longitud $L$. El símbolo $\#$, también llamado como "don't-care", puede representar a ambos cero y uno. La notación de un macroclasificador es importante únicamente para propósitos de eficiencia. El representar clasificadores idénticos, no como varios micro-clasificadores, sino como un macroclasificador aumenta la velocidad del proceso de comparación [M].

## 3.4. Componentes de desempeño

Dada una entrada, existe un conjunto de concordancia $[M]$. El sistema forma una predicción del sistema $P(a_i)$ para cada acción $a_i$ representada en $[M]$. Los valores de $P(a_i)$ son colocados en el arreglo de predicción y una acción es seleccionada.

Existen muchos métodos de selección de acciones, el sistema puede simplemente seleccionar la acción con la mayor predicción (selección determinística), así mismo, la acción puede ser seleccionada de manera probabilística, con la probabilidad de selección proporcional $P(a_i)$ (selección de rueda de ruleta). En algunos casos, la acción puede ser seleccionada completamente de manera aleatoria. Una vez que la acción es seleccionada, el sistema forma un conjunto de acciones [A] que consiste en aquellos clasificadores en [M] que contienen a la acción seleccionada, esa acción es mandada al actuador y una recompensa inmediata $r_{imm}$ (o no inmediata) es regresada por el ambiente.

## 3.5. Componentes de refuerzo

Los componentes de refuerzo del XCS consisten en actualizar los parámetros $p$, $\epsilon$ y $F$ de los clasificadores en el conjunto de acciones $[A]_{-1}$ como se muestra en la figura 3.1. Los valores de $p$ son ajustados por la técnica de $Q$ *learning*, la cual es implementada como se muestra en la figura 3.1 por la combinación de tomar el máximo $P(a_i)$ del arreglo de predicción, reduciéndolo al multiplicarlo por un factor $\psi(0 < \psi \leq 1)$ y agregándole cualquier premio obtenido en el paso anterior. La cantidad resultante, llamada simplemente $P$, es utilizada para ajustar las predicciones $p_j$ de los clasificadores en $[A]_{-1}$ utilizando la regla delta estándar de Widrow-Hoff con el parámetro de aprendizaje $\beta(0 < \beta \leq 1)$. Esto es $p_j \leftarrow p_j + \beta(P - p_j)$.

Sin embargo, para cada clasificador en $[A]_{-1}$ la actualización comienza primero por recalcular la aptitud $F_j$, utilizando el valor actual de $\epsilon_j$, después $\epsilon_j$ es ajustado utilizando $P$ y el valor actual de $p_j$. Para esto, la técnica de Widrow-Hoff es utilizada

para ajustar $\epsilon_j$ hacia la diferencia absoluta $|P - p_j|$, es decir $\epsilon_j \leftarrow \epsilon_j + \beta(|P - p_j| - \epsilon_j)$. Finalmente $p_j$ es ajustado. (El ajuste de $F$ y $\epsilon$ hacen el "componente por refuerzo").

La regla de Widrow-Hoff es utilizada para $p$, $\epsilon$, como parte del ajuste de F sólo después de que el clasificador ha sido ajustado por lo menos $\frac{1}{\beta}$ veces. Antes de eso, los nuevos valores de $p_j$ en el cuarto ajuste serán únicamente un cuarto de la suma de los primeros cuatro valores de P, si $\frac{1}{\beta} > 4$. Esta técnica de dos fases permite que los valores de los parámetros se muevan mas rápidamente a su promedio "verdadero", y hace que el sistema sea menos sensitivo a valores iniciales posiblemente arbitrarios.

## 3.6. Componente de descubrimiento

La idea de ejecutar el algoritmo genético en un subconjunto de la población (por ejemplo en el conjunto de comparación) en lugar de ejecutarse en toda la población fue idea de Booker [5]. El XCS inicialmente siguió la idea de Booker en ejecutar el AG en el conjunto de concordancia $[M]$ como se muestra en la figura 3.1, sin embargo el AG ahora se ejecuta en el conjunto de acciones $[A]_{-1}$ o $[A]$ (ver figura 3.2). La razón está en que si un conjunto de comparación dado contiene clasificadores precisos y muy generales para cada una de las posibles acciones, tendían frecuentemente condiciones diferentes, pero si ese era el caso el cruce no sería benigno, fácilmente generaría clasificadores imprecisos al cruzar padres con diferentes acciones, por esta razón se utiliza el GA en el conjunto de acción $[A]$ en lugar de usarlo en el conjunto de concordancia $[M]$ [29].

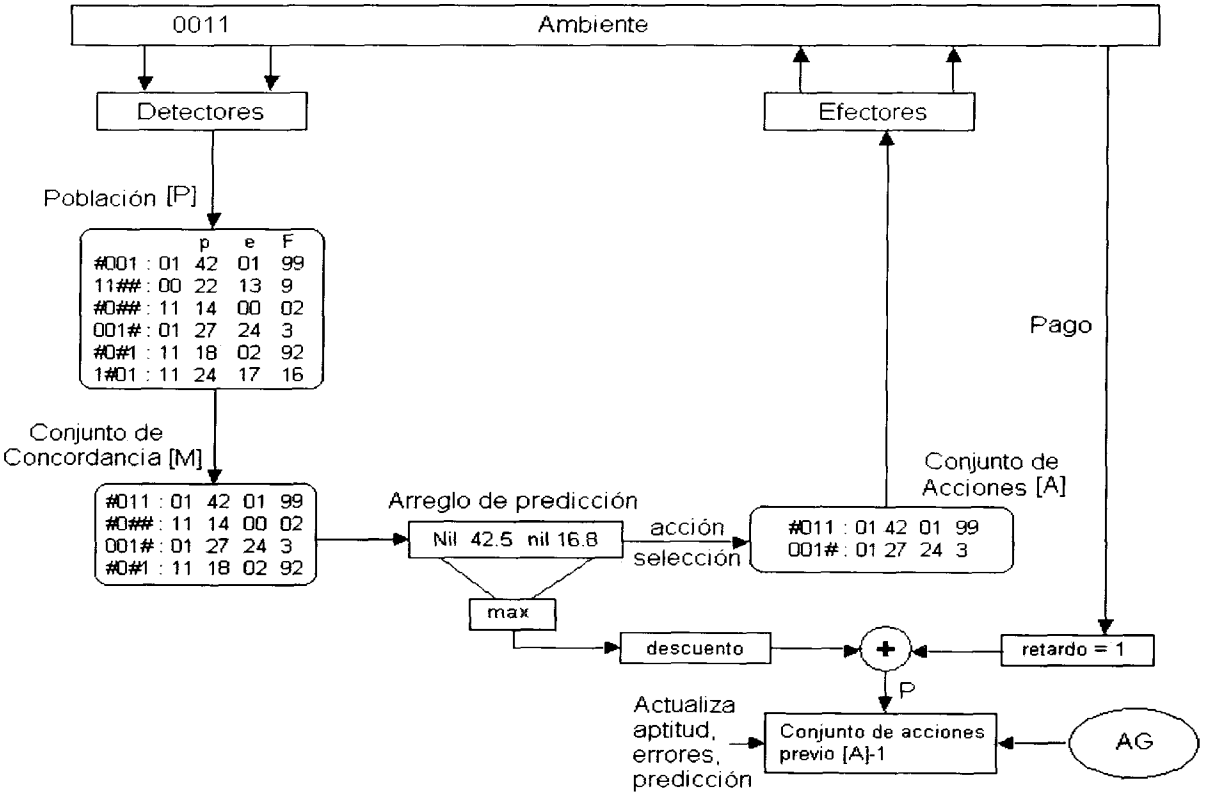



Figura 3.2: Esquema del XCS.

Como se puede ver en la figura 3.2 el algoritmo genético actúa en el conjunto de acciones $[A]$, selecciona dos clasificadores de $[A]$ con probabilidades proporcionales a su aptitud, copia a los clasificadores, efectúa cruce en las copias con probabilidad $X$ y con probabilidad $\mu$ efectúa una mutación en ellos. Si $[P]$ contiene menos de $N$ miembros, las copias son insertadas dentro de la población y no ocurre ningún borrado por compensación. De lo contrario dos clasificadores son borrados estocásticamente de $[P]$ para hacer espacio.

Además del algoritmo genético, el componente de descubrimiento contiene un mecanismo de reemplazo para utilizar en dos circunstancias especiales. Primero, si sucede que ningún clasificador en el conjunto de concordancia $[M]$ es igual a la entrada dada, el XCS crea un nuevo clasificador con una condición que sea igual a la entrada y una acción seleccionada de manera aleatoria. El nuevo clasificador es insertado en $[P]$, y el XCS forma un nuevo conjunto $[M]$ y prosigue como siempre. El reemplazo también es utilizado si el sistema está encerrado en un ciclo, por ejemplo si el mecanismo de selección de acción causa que el sistema persistentemente retroceda y avance entre dos posiciones del ambiente, la creación de un nuevo clasificador con una condición igual a la entrada y una acción aleatoria normalmente es suficiente para romper ese ciclo.

## 3.7. Cálculo de la aptitud

La definición de aptitud en el sistema de clasificadores XCS es diferente a la definición en los sistemas de clasificadores tradicionales. En el XCS la aptitud está basada en la precisión de la predicción de la paga del clasificador, en lugar de ser la predicción por sí misma.

La aptitud de un clasificador es actualizada cada vez que este pertenece a $[A]_{-1}$. La aptitud es actualizada por una cantidad que depende en la precisión del clasificador relativa con las precisiones de otros clasificadores en el conjunto. Hay tres pasos en el cálculo.

1. La precisión de cada clasificador $k_j$ es calculada. La precisión se define como una función del valor actual de $\epsilon_j$, esto es $k_j = exp[(ln\alpha)(\epsilon_j - \epsilon_0)/\epsilon_0]$ para $\epsilon_j > \epsilon_0$, de lo contrario 1. Esta función cae de manera exponencial para $\epsilon_j > \epsilon_0$.

2. Una precisión relativa es calculada $k'_j$ para cada clasificador, dividiendo su precisión por el total de las precisiones en el conjunto.

3. Finalmente, la precisión relativa es utilizada para ajustar la aptitud del clasificador $F_j$. si la aptitud ha sido ajustada por lo menos $1/\beta$ veces, $F_j \leftarrow F_j + \beta(k'_j - F_j)$, de lo contrario $F_j$ es el promedio de los valores actuales y previos de $k'_j$.

Dado que las precisiones relativas suman 1, el total de los ajustes a los miembros del conjunto $[A]_{-1}$ es constante. El efecto es que los distintos conjuntos de acciones dadas en el conjunto de concordancia $[M]$ tienen aproximadamente la misma aptitud total. Dado que la reproducción depende de la aptitud, aproximadamente el mismo número de clasificadores será asociado con cada acción que es representada en el conjunto $[M]$.

## 3.8. Macroclasificadores

Cada vez que el XCS genera un nuevo clasificador, la población es escaneada para ver si el nuevo clasificador tiene la misma condición y acción que la de algún clasificador existente. Si es así el nuevo clasificador no es añadido a la población, pero el campo de numerosidad en el clasificador existente es incrementada en uno. Si no existe un clasificador con condición y acción idéntica, el nuevo clasificador es añadido a la población con su campo de numerosidad inicializado en uno. A estos clasificadores se les conoce como macroclasificadores. Esta técnica acelera el proceso de búsqueda de clasificadores en la población $[P]$ concordantes a una entrada.

Para asegurar que el sistema se comporte como si consistiera de $N$ clasificadores normales, todas las funciones del sistema están escritas de tal manera que son sensibles a la numerosidad. Por ejemplo, para calcular la precisión relativa, un macroclasificador con numerosidad $n$ será tratado como $n$ clasificadores por separado.

## 3.9. Lista de parámetros del sistema

En la descripción anterior del XCS se ha mencionado a la mayoría de los parámetros del sistema. Estos se resumen a continuación.

$N$ - Tamaño de la población

$\beta$ - Proporción de aprendizaje para predicción, error de predicción y aptitud.

$\theta$ - Ejecuta el AG en este conjunto $[M]$ si el número de pasos promedio desde la última vez que se corrió el AG es mayor que $\theta$.

$\epsilon_0$, $\alpha$ - Parámetros para la función de precisión.

$X$ - Probabilidad de cruce por invocación del AG.

$\mu$ - Probabilidad de mutación

$\phi$ - Si la predicción total en $[M]$ es menor que $\phi$ veces la media de la población [P], ocurre cubrimiento.

$P_{\#}$ - Probabilidad de un # en una posición en la condición del clasificador, creada por reemplazo y en las condiciones de clasificadores en una población inicial generada aleatoriamente.

$p_b$, $\epsilon_p$ y $F_I$ - Predicción, error de predicción, y aptitud asignada a cada clasificador en la población inicial.

## 3.10. Mecanismo de generalización

En el XCS, los clasificadores que predicen con mayor precisión obtienen mayor aptitud. Si la condición de un clasificador es altamente específica uno esperaría que fuera más precisa que una con condición menos específica dado que tiene que predecir la paga sobre un número más pequeño de situaciones. Si la aptitud está basada en precisión, los clasificadores más específicos deben de ganar sobre los menos específicos. Entonces ¿por qué el sistema de clasificadores XCS busca encontrar clasificadores más generales que sean precisos? La respuesta se basa en la observación que bajo un AG, el éxito de reproducción no sólo depende de aptitud, sino también de las oportunidades de reproducción. Considere dos (micro)clasificadores $C1$ y $C2$ teniendo la misma acción, en donde la condición de $C2$ es una generalización de $C1$. Es decir, la condición de $C2$ puede ser generada a partir de la condición de $C1$ cambiando uno o más de los ceros o unos por "don't-cares" (#), suponga que $C1$ y $C2$ tienen el mismo $\epsilon$, y son igualmente precisos, ¿Cuál ganará? Cada vez, $C1$ y $C2$ ocurren en el mismo conjunto de acciones,

sus valores de aptitud son actualizados en la misma cantidad, sin embargo, como $C2$ es una generalización de $C1$, tratará de ocurrir en más conjuntos de comparación $[M]$ que $C1$ y por lo tanto probablemente en más conjuntos de acciones. Dado que el $AG$ corre en el conjunto de acciones, $C2$ tendrá más oportunidades de reproducción. Cuando $C1$ y $C2$ se vuelvan a encontrar en un conjunto de acciones, la numerosidad de $C2$ será mayor que $C1$, y eventualmente $C2$ desplazará a $C1$ de la población.

## 3.11. Ambiente de un paso

En un ambiente de un paso el pago externo es recibido en cada intervalo de tiempo y la entrada del ambiente para cada paso es completamente independiente del paso anterior. Los problemas de categorización de información, típicamente son de un paso, dado que una decisión es tomada, y un pago de acuerdo a la calidad de la decisión es recibido.

## 3.12. Ambiente de múltiples pasos

En problemas secuenciales o de multiples pasos, el pago puede ocurrir (pero no necesariamente) en cualquier intervalo de tiempo, y la entrada en un intervalo de tiempo es dependiente por lo menos de la última entrada y de la última acción del sistema. En su forma más simple, el sistema busca obtener tanta recompensa como sea posible, y debe de aprender asociaciones entre entradas del ambiente y sus propias acciones que lleven a la recompensa.

En el ambiente básico de multiples pasos, la siguiente entrada y el pago (si hay alguno) encontrado por el sistema, depende únicamente de la entrada actual $x$ y la acción actual $a$; no hay mas dependencia histórica. La predictibilidad de $y$ dado $x$ y $a$ es posible gracias a la técnica ampliamente utilizada "Q-learning" [27].

## 3.13. Resumen

En este capítulo se presentó el esquema completo del sistema de clasificadores XCS. Se explicaron los parámetros que forman a cada clasificador dentro del sistema de clasificadores y la manera en que estos clasificadores son evaluados por el sistema para ser evolucionados y formar a mejores individuos que representan mejores soluciones al problema.

## Capítulo 4

# Modelo de Solución

En este capítulo se muestra la representación del problema de corte de materiales rectangulares en dos dimensiones como un ambiente para el sistema de clasificadores XCS. Se explica la función de evaluación utilizada y la relación de las acciones de los clasificadores con las heurísticas simples.

## 4.1. Antecedentes

En esta investigación se tomaron conjuntos de piezas rectangulares, las cuales requerían ser cortadas de un conjunto de materiales en almacén, también llamados objetos, con altura $H$ y longitud $W$. Cada pieza $j \epsilon J = 1, ..., n$ es caracterizada por su altura $h_j \leq H$ y anchura $w_j \leq W$. El problema de corte y empacado de materiales en dos dimensiones busca patrones de corte que contengan todas las $n$ piezas de tal manera que el número total de objetos en almacén requeridos sea el mínimo.

Para la creación de la hiper-heurística, a partir de las heurísticas simples, se utilizó el sistema de clasificadores XCS. El sistema de clasificadores XCS evoluciona un modelo de comportamiento que determina la calidad de cada acción posible en todas las situaciones o instancias del problema, así las acciones posibles son el conjunto de heurísticas que se tienen para resolver el problema.

El XCS se ejecuta hasta obtener una serie de reglas. Estas reglas deben ser las indicadas para que dada una cierta condición de cambio en el ambiente, se ejecute una determinada heurística. El proceso de aprendizaje del XCS debe de encontrar las reglas que mejor evaluación tienen para las diferentes instancias del problema. La hiper-heurística formada por dichas reglas puede ser seguida como mapa para saber el orden en que las cuchillas de la máquina deberán ser acomodadas para realizar los cortes.

El esquema completo del algoritmo que se utilizó para resolver el problema de corte, este se ilustra en la figura 4.1.

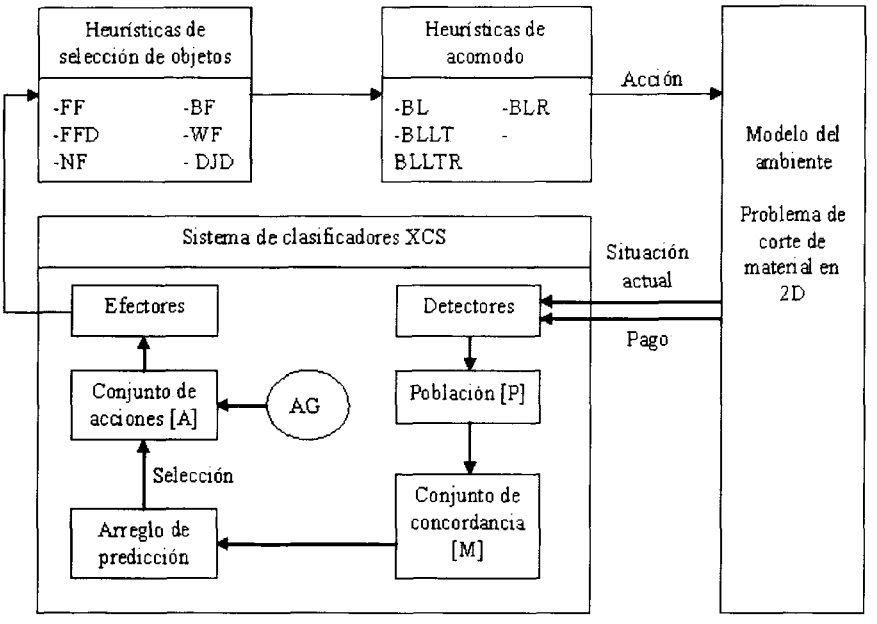



Figura 4.1: Modelo de Hiper-Heurística con Sistema de Clasificadores XCS.

Los componentes principales del modelo mostrado en la figura 4.1 se enlistan a continuación.

**Detectores.-** Los detectores en el sistema de clasificadores XCS, reciben la siguiente información del ambiente:

- La situación actual de problema, la cual contiene información como el porcentaje y el tamaño de piezas restantes por acomodar.
- Pago. El valor del pago recibido por el sistema de clasificadores es evaluado por el ambiente dependiendo de la calidad de la solución propuesta para resolver el problema.

**Población.-** El conocimiento en el XCS es representado en una población que contiene a un grupo de individuos (clasificadores) cuyo número es controlado. Cada individuo representa una solución a la situación del problema. Entre mayor sea la población, mayor el número de soluciones representadas.

**Conjunto de concordancia $M$.-** Dada una entrada, o una situación dada del problema, existe un conjunto de individuos que concuerdan con esta situación.

**Arreglo de predicción.-** El sistema forma una predicción $P(a_i)$ para cada acción $a_i$ representada en $[M]$.

**Conjunto de acciones $A$.-** Las acciones son seleccionadas en base a la predicción y colocadas en el conjunto de acciones $[A]$ que consiste en aquellos clasificadores en $[M]$ que contienen a la acción seleccionada.

**Algoritmo genético** $AG$.- El algoritmo genético actúa en el conjunto de acciones $[A]$, selecciona clasificadores de $[A]$ con probabilidades proporcionales a su aptitud, copia a los clasificadores, efectúa cruce en las copias con probabilidad $X$ y con probabilidad $\mu$ efectúa una mutación en ellos.

**Efectores o actuadores.-** Los actuadores del sistema de clasificadores XCS indican las acciones a tomar, es decir contienen la información necesaria para saber qué heurística de selección y qué heurística de acomodo utilizar para resolver la situación actual del problema.

**Heurísticas de selección.-** Las heurísticas de selección proponen los objetos y el orden en el que las piezas deben de ser acomodadas dentro de éstos.

**Heurísticas de acomodo.-** Las heurísticas de acomodo indican las coordenadas en las que las piezas se deben de colocar dentro de los objetos.

**Medio ambiente.-** El medio ambiente es de vital importancia para todo sistema inteligente. Si el sistema puede interactuar con el medio ambiente es posible que evolucione satisfactoriamente. Por lo tanto el ambiente debe de informar al sistema, de la manera mas clara posible, la situación actual del problema y hacerle saber que tan buena o que tan mala ha sido la solución propuesta. Esto último lo realiza mediante el pago.

## 4.2. Representación del ambiente

Para lograr el objetivo de encontrar un conjunto adecuado de reglas, que asociaran a una heurística con una descripción del estado actual del problema, se codificó el problema como un ambiente para el sistema de clasificadores XCS. Este ambiente le informa al sistema de clasificadores de que tamaño son los objetos de material, la cantidad de piezas que aún faltan por acomodar y el tamaño de las piezas disponibles para ser acomodadas. Cada clasificador, en el sistema de clasificadores, asocia la situación actual del ambiente con una heurística. La parte de la condición de los clasificadores es una adaptación de la representación presentada por Ross [22], ésta contiene a las variables que representan el estado actual del problema, como el espacio disponible de materia prima que aún puede ser utilizada para cortar más piezas o el porcentaje de piezas que faltan por acomodar. La parte de la acción contiene a la combinación de heurísticas (una de selección y una de acomodo), que servirá para acomodar las piezas adecuadas según las condiciones.

Durante la corrida, el ambiente va a informando al sistema de clasificadores, cuál es la situación actual del problema, con esta información el sistema de clasificadores busca por nuevas reglas que concuerden con esta situación y propone la actuación de

nuevas heurísticas, las cuales acomodan las piezas de acuerdo a su criterio, con ésto, el ambiente actualiza las dimensiones de la materia prima disponible. El criterio de terminación para que el sistema de clasificadores detenga la prueba se da cuando la demanda de piezas ha sido satisfecha.

### 4.2.1. Representación del problema de corte como ambiente de un paso

En un ambiente de un paso el pago externo es recibido en cada intervalo de tiempo y la entrada del ambiente para cada paso es completamente independiente del paso anterior. Representando al ambiente de corte de esta manera, una decisión es tomada (una combinación de heurísticaa es propuesta), y un pago de acuerdo a la calidad de la decisión es recibido.

### 4.2.2. Representación del problema de corte como ambiente de múltiples pasos

En problemas secuenciales o de múltiples pasos, el pago ocurre ya que se ha resuelto totalmente el problema, y la entrada en un intervalo de tiempo es dependiente por lo menos de la última entrada y de la última acción del sistema. Representando al ambiente de corte de esta manera, una secuencia de decisiones son tomadas (varias combinaciones de heurísticas son propuestas), y un pago de acuerdo a la calidad de la solución entregada por esta combinación de acciones es recibido.

## 4.3. Representación de reglas

Una regla determina la relación que existe entre la condición y la acción. El sistema de reglas del sistema de clasificadores, de acuerdo a la codificación de las condiciones y las acciones, se muestra en la tabla 4.1.

Tabla 4.1: Estructura de las reglas en los clasificadores del XCS.

| *Condición* | | | | | | | | | | | | *Acción* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SH | MH | LH | SW | MW | LW | SA | MA | LA | HA | R | → | HSC | HAC |

- La parte de la condición contiene la siguiente información:
  - Representación de la altura

    **SH** - Porcentaje de piezas cuya altura es menor a un tercio de la altura total del objeto

**MH** - Porcentaje de piezas cuya altura es menor a un medio y mayor a un tercio de la altura total del objeto

**LH** - Porcentaje de piezas cuya altura es mayor a un medio de la altura total del objeto

- Representación de la anchura

**SW** - Porcentaje de piezas cuya anchura es menor a un tercio de la anchura total del objeto

**MW** - Porcentaje de piezas cuya anchura es menor a un medio y mayor a un tercio de la anchura total del objeto

**LW** - Porcentaje de piezas cuya anchura es mayor a un medio de la anchura total del objeto

- Representación del área

**SA** - Porcentaje de piezas cuya área es menor a un cuarto del área total del objeto

**MA** - Porcentaje de piezas cuya área es mayor a un cuarto y menor a un tercio del área total del objeto

**LA** - Porcentaje de piezas cuya área es mayor a un tercio y menor a un medio del área total del objeto

**HA** - Porcentaje de piezas cuya área es mayor a un medio del área total del objeto

- La parte de la acción contiene la siguiente información:

- Representación del área

**HSC** - Heurística de selección

**HAC** - Heurística de acomodo

## 4.4. Codificación de la condición de los clasificadores

La representación del estado del problema, es una adaptación de representación presentada en Ross [22]. Se definieron cuatro categorías para clasificar a las piezas de acuerdo al valor de su área. Los intervalos utilizados para cada una de las categorías se muestran en la tabla 4.2.

Tabla 4.2: Intervalo del área de las piezas.

| *Categoría* | *Intervalo de área* |
|---|---|
| Huge | mayor a 1/2 del área total del objeto |
| Large | mayor a 1/3 y menor o igual a 1/2 del área del objeto |
| Medium | mayor a 1/4 y menor o igual a 1/3 del área del objeto |
| Small | menor a 1/4 del área del objeto |

Así mismo, se definieron tres categorías para clasificar a las piezas de acuerdo al valor de su altura y tres para clasificarlas de acuerdo a su anchura. Los intervalos utilizados para cada una de las categorías se muestran en las tablas 4.3 y 4.4.

Tabla 4.3: Intervalo del altura de las piezas.

| *Categoría* | *Intervalo de altura* |
|---|---|
| Large | mayor a 1/2 de la altura total del objeto |
| Medium | mayor a 1/3 y menor o igual a 1/2 de la altura del objeto |
| Small | menor a 1/3 de la altura del objeto |

Tabla 4.4: Intervalo del anchura de las piezas.

| *Categoría* | *Intervalo de anchura* |
|---|---|
| Large | mayor a 1/2 del ancho total del objeto |
| Medium | mayor a 1/3 y menor o igual a 1/2 del ancho del objeto |
| Small | menor a 1/3 del ancho del objeto |

Las piezas que aún no han sido acomodadas son examinadas, y se calcula el porcentaje de piezas totales para cada una de las categorías, representando esto en la condición del clasificador de acuerdo a la tabla 4.5.

Tabla 4.5: Porcentaje de piezas en un intervalo de tamaño.

| *Bits* | *Porcentaje de piezas en categoría* |
|---|---|
| 0 0 | 0-10 % |
| 0 1 | 10-20 % |
| 1 0 | 20-50 % |
| 1 1 | 50-100 % |

Del mismo modo, se calcularon y representaron las piezas totales que aún faltaban por acomodar. Los porcentajes obtenidos para cada instancia de tiempo en el problema se representa en la percepción del ambiente de acuerdo a la tabla 4.6.

Tabla 4.6: Porcentaje de piezas restantes por acomodar.

| *Bits* | *Porcentaje de piezas restantes* |
|---|---|
| 0 0 0 | 0-14 % |
| 0 0 1 | 14-28 % |
| 0 1 1 | 28-42 % |
| 0 1 0 | 42-56 % |
| 1 1 0 | 56-70 % |
| 1 0 0 | 70-84 % |
| 1 1 1 | 84-100 % |

## 4.5. Codificación de las acciones de los clasificadores

Para seleccionar entre el conjunto de heurísticas, las acciones se representaron mediante números enteros, cada acción indica la decisión de cuál estrategia tomar en la condición actual del ambiente. En la tabla 4.7 se muestra un listado de las acciones y su relación con las heurísticas de selección y de acomodo correspondientes.

Tabla 4.7: Representación de acciones.

| *Acción* | *Heurística de selección* | *Heurísticas de acomodo* |
|---|---|---|
| 1 | First Fit (FF) | Bottom-Left (BL) |
| 2 | | Bottom-Left Rotate (BLR) |
| 3 | | Improved Bottom-Left(BLLT) |
| 4 | | Improved Bottom-Left Rotate(BLLTR) |
| 5 | First Fit Decreasing (FFD) | Bottom-Left (BL) |
| 6 | | Bottom-Left Rotate (BLR) |
| 7 | | Improved Bottom-Left(BLLT) |
| 8 | | Improved Bottom-Left Rotate(BLLTR) |
| 9 | First Fit Increasing (FFI) | Bottom-Left (BL) |
| 10 | | Bottom-Left Rotate (BLR) |
| 11 | | Improved Bottom-Left(BLLT) |
| 12 | | Improved Bottom-Left Rotate(BLLTR) |
| 13 | Filler+FFD | Bottom-Left (BL) |
| 14 | | Bottom-Left Rotate (BLR) |
| 15 | | Improved Bottom-Left(BLLT) |
| 16 | | Improved Bottom-Left Rotate(BLLTR) |
| 17 | Next Fit (NF) | Bottom-Left (BL) |
| 18 | | Bottom-Left Rotate (BLR) |
| 19 | | Improved Bottom-Left(BLLT) |
| 20 | | Improved Bottom-Left Rotate(BLLTR) |
| 21 | Next Fit Decreasing (NFD) | Bottom-Left (BL) |
| 22 | | Bottom-Left Rotate (BLR) |
| 23 | | Improved Bottom-Left(BLLT) |
| 24 | | Improved Bottom-Left Rotate(BLLTR) |
| 25 | Best Fit (BF) | Bottom-Left (BL) |
| 26 | | Bottom-Left Rotate (BLR) |
| 27 | | Improved Bottom-Left(BLLT) |
| 28 | | Improved Bottom-Left Rotate(BLLTR) |
| 29 | Best Fit Decreasing (BFD) | Bottom-Left (BL) |
| 30 | | Bottom-Left Rotate (BLR) |
| 31 | | Improved Bottom-Left(BLLT) |
| 32 | | Improved Bottom-Left Rotate(BLLTR) |
| 33 | Worst Fit (WF) | Bottom-Left (BL) |
| 34 | | Bottom-Left Rotate (BLR) |
| 35 | | Improved Bottom-Left(BLLT) |
| 36 | | Improved Bottom-Left Rotate(BLLTR) |
| 37 | Djang and Finch (DJD) | Bottom-Left (BL) |
| 38 | | Bottom-Left Rotate (BLR) |
| 39 | | Improved Bottom-Left(BLLT) |
| 40 | | Improved Bottom-Left Rotate(BLLTR) |

Cada una de las heurísticas propuestas inserta por lo menos una pieza, esto garantiza un cambio en el estado del problema.

## 4.6. Función de Evaluación

Para poder evaluar el desempeño de cada una de las heurísticas, se realizaron pruebas representando al ambiente para el sistema de clasificadores de la siguiente manera:

- Como un ambiente de un solo paso, en el cual la paga por la acción propuesta es otorgada en cada acción.
- Como un ambiente de múltiples pasos o secuencial, en donde la paga es otorgada por la solución completa.

Inicialmente se utilizó una evaluación basada en el porcentaje promedio de material utilizado de todos los objetos, por ejemplo, si las piezas acomodadas ocupaban un 98 % del espacio total de los objetos, la paga era de 98. Sin embargo, con el fin de promover que cada uno de los objetos tuviera el menor desperdicio posible, en lugar de evaluar el rendimiento general de todos, se decidió utilizar una función de evaluación que premiara a las heurísticas que propusieran objetos que tuvieran menor desperdicio y de igual manera castigara a las heurísticas que propusieran soluciones en donde hubiera mayor desperdicio. Para lograr lo anterior se utilizó la siguiente función de evaluación:

$$f = \frac{\sum_{i=1}^{N} g(x_i)}{N} \tag{4.1}$$

En donde $N$ es el número de objetos, y $g(x_i)$ está dada por la siguiente ecuación.

$$g(x_i) = {x_i}^2 \tag{4.2}$$

La variable $x_i$ se obtiene con la siguiente fórmula.

$$x_i = \frac{AUO_i}{ATO_i} \tag{4.3}$$

en donde $AUO_i$ es el área utilizada por la piezas en el objeto $i$, $ATO_i$ es el área total del objeto $i$

En la gráfica de la figura 4.2 se puede observar que esta función asigna mucho mejor evaluación a las heurísticas que proponen una mejor utilización del área total del objeto.

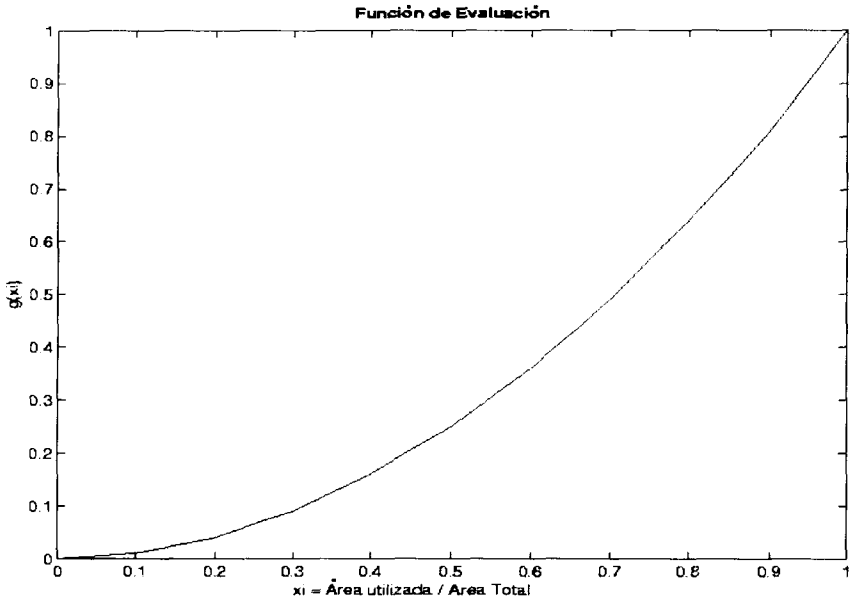



Figura 4.2: Función de evaluación para el problema de corte.

## 4.7. Parámetros del sistema

Para los experimentos reportados en el presente trabajo, se asignaron los siguientes valores para los parámetros del sistema.

$N$ - Tamaño de la población: 800

$\beta$ - Proporción de aprendizaje para predicción, error de predicción y aptitud.: 0.2

$\theta$ - Ejecuta el AG si el número de pasos promedio desde la última vez que se corrió el AG es mayor que $\theta$: 25

$\epsilon_0$ - Límite de error bajo el cuál la precisión del clasificador es puesta en uno: 10

$\alpha$ - Parámetro para la función de precisión: 0.1

$X$ - Probabilidad de cruce por invocación del AG: 0.8

$\mu$ - Probabilidad de mutación: 0.04

$\phi$ - Si la predicción total en $[M]$ es menor que $\phi$ veces la media de la población [P], ocurre cubrimiento.

$P_{\#}$ - Probabilidad de un # en una posición en la condición del clasificador, creada por reemplazo y en las condiciones de clasificadores en una población inicial generada aleatoriamente: 0.5

$p_b$ - Predicción asignada en la población inicial: 10.0

$\epsilon_p$ - Error de predicción asignado a la población inicial: 0.0

$F_I$ - Aptitud asignada a cada clasificador en la población inicial: 0.01

## 4.8. Resumen

En este capítulo se mostró la representación del problema de corte de materiales rectangulares en dos dimensiones como un ambiente en el sistema de clasificadores XCS. Se explicó la función de evaluación utilizada, la representación del estado actual del problema en los clasificadores, la estructura de las reglas, y la relación de las acciones de los clasificadores con las heurísticas simples.

## Capítulo 5

# Experimentos y análisis de resultados

En este capítulo se describen los experimentos realizados a lo largo del desarrollo de esta investigación. Se analiza de manera independiente el comportamiento de cada una de las heurísticas y se presentan los resultados obtenidos para las diferentes instancias de problemas. En los experimentos, se utilizaron dos conjuntos de problemas:

1. Se seleccionaron problemas de la literatura para corte de materiales en dos dimensiones
2. Se generaron problemas aleatorios para corte de materiales en dos dimensiones

Para cada uno de los experimentos se presenta un análisis y una serie de observaciones. Se muestran las tablas comparativas de los mejores resultados obtenidos por las diferentes técnicas mencionadas. Al final de este capítulo, se presenta un análisis general de los resultados obtenidos.

## 5.1. Cálculo del límite inferior continuo

El límite inferior continuo determina el número mínimo de objetos necesarios para satisfacer la demanda de piezas. Como se mencionó en la sección 2.1.2, el límite inferior continuo para el problema de corte de material en dos dimensiones se calcula de acuerdo a la siguiente fórmula:

$$L_o = \left\lceil \frac{\sum_{j=1}^{n} h_j w_j}{HW} \right\rceil \tag{5.1}$$

En donde: $H$ es la altura de los objetos, $W$ es la anchura de los objetos y cada una de las piezas $j \epsilon J = 1, ..., n$ tiene una altura $h_j \leq H$ y una anchura $w_j \leq W$.

## 5.2. Experimentos y análisis de resultados con problemas de la literatura

Para esta investigación, se trabajó con instancias de problemas obtenidos de la literatura para el problema de corte de materiales en dos dimensiones. Los problemas cgcut1-cgcut3 están descritos en Christofides y Whitlock [8], los problemas gcut1-gcut13 y ngcut1-ngcut12 están descritos en Beasley OR-Library [4].

Se compararon los resultados obtenidos, con la combinación óptima de piezas propuesta en Martello y Vigo [20], para resolver estos problemas de corte.

Para cada unos de los problemas, se muestran tablas comparativas que presentan el nombre del problema, el número de piezas ($n$), el valor del límite inferior continuo $L_o$, el valor de la solución óptima ($z$) propuesta en Martello y Vigo [20],el valor de la mejor heurística encontrada ($BH$), el valor de la mejor solución obtenida con la hiper-heurística mediante el sistema de clasificadores XCS bajo el esquema de pago en un paso ($HH-XCS-SS$) y el valor de la mejor solución obtenida con la hiper-heurística mediante el sistema de clasificadores XCS bajo el esquema de pago en múltiples pasos ($HH-XCS-MS$)

### 5.2.1. Propiedades de los problemas

En la literatura existen diversos problemas de corte de materiales en dos dimensiones, algunos de estos son:

**Problemas de corte de material en dos dimensiones guillotinable -** Existen 16 problemas de este tipo en el repositorio de J.E.Beasley [2], tres disponibles en los archivos cgcut1-cgcut3, y los otros trece en los archivos gcut1-gcut13. En la tabla 5.1 se muestran las características de los problemas cgcut1-cgcut3. Cada problema tiene características diferentes, por ejemplo, en el problema cgcut1, las piezas son muy pequeñas en comparación con el tamaño de los objetos. En el problema cgcut2 existe una relación inversa entre el ancho y la altura de las piezas, es decir, las piezas que son muy anchas son poco altas, y las piezas que son poco anchas, son muy altas. En el problema cgcut3, en general las piezas son muy anchas y existe una relación directa entre la anchura y la altura, entre más anchas son las piezas, generalmente son más altas.

Tabla 5.1: Características de los problemas cgcut1-cgcut3.

| *cgcut1* | | *cgcut2* | | *cgcut3* | |
|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto |
| 15 | 10 | 40 | 70 | 40 | 70 |
| Número de piezas | | Número de piezas | | Número de piezas | |
| 16 | | 23 | | 62 | |

En las tablas 5.2, 5.3, 5.4 y 5.5 se muestran las características de los problemas gcut1-cgut13. Para estos problemas, el número de piezas y las dimensiones de los objetos varía en cada problema. Para los problemas gcut1-gcut4, el ancho y alto de los objetos es 250, para los problemas gcut5-gcut8 es 500, para los problemas gcut9-gcut12 es 1000 y para el problema 13 es 3000.

Tabla 5.2: Características de los problemas gcut1-gcut4.

| *gcut1* | | *gcut2* | | *gcut3* | | *gcut4* | |
|---|---|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto | ancho | alto |
| 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 |
| Número de piezas | | Número de piezas | | Número de piezas | | Número de piezas | |
| 10 | | 20 | | 30 | | 50 | |

Tabla 5.3: Características de los problemas gcut5-gcut8.

| *gcut5* | | *gcut6* | | *gcut7* | | *gcut8* | |
|---|---|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto | ancho | alto |
| 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 |
| Número de piezas | | Número de piezas | | Número de piezas | | Número de piezas | |
| 10 | | 20 | | 30 | | 50 | |

Tabla 5.4: Características de los problemas gcut9-gcut12.

| *gcut*9 | | *gcut*10 | | *gcut*11 | | *gcut*12 | |
|---|---|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto | ancho | alto |
| 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| Número de piezas | | Número de piezas | | Número de piezas | | Número de piezas | |
| 10 | | 20 | | 30 | | 50 | |

Tabla 5.5: Características del problema gcut13.

| *gcut*13 | |
|---|---|
| Objetos | |
| ancho | alto |
| 3000 | 3000 |
| Número de piezas | |
| 32 | |

**Problemas de corte de material en dos dimensiones no guillotinable -** Existen doce problemas de este tipo en el repositorio de J.E.Beasley [2], estos se pueden encontrar en los archivos ngcut1-ngcut12. En las tablas 5.6, 5.7, 5.8 y 5.9 se muestran las características de estos problemas.

Tabla 5.6: Características de los problemas ngcut1-ngcut3.

| *ngcut*1 | | *ngcut*2 | | *ngcut*3 | |
|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto |
| 10 | 10 | 10 | 10 | 10 | 10 |
| Número de piezas | | Número de piezas | | Número de piezas | |
| 10 | | 17 | | 21 | |

Tabla 5.7: Características de los problemas ngcut4-ngcut6.

| *ngcut*4 | | *ngcut*5 | | *ngcut*6 | |
|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto |
| 15 | 10 | 15 | 10 | 15 | 10 |
| Número de piezas | | Número de piezas | | Número de piezas | |
| 7 | | 14 | | 15 | |

Tabla 5.8: Características de los problemas ngcut7-ngcut9.

| *ngcut*7 | | *ngcut*8 | | *ngcut*9 | |
|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto |
| 20 | 20 | 20 | 20 | 20 | 20 |
| Número de piezas | | Número de piezas | | Número de piezas | |
| 8 | | 13 | | 18 | |

Tabla 5.9: Características de los problemas ngcut10-ngcut12.

| *ngcut*10 | | *ngcut*11 | | *ngcut*12 | |
|---|---|---|---|---|---|
| Objetos | | Objetos | | Objetos | |
| ancho | alto | ancho | alto | ancho | alto |
| 30 | 30 | 30 | 30 | 30 | 30 |
| Número de piezas | | Número de piezas | | Número de piezas | |
| 13 | | 15 | | 22 | |

La información proporcionada en los archivos de los problemas es la siguiente:

- Número de piezas a ser cortadas (m)
- Longitud y ancho de los objetos rectangulares
- Para cada una de las piezas i ($i = 1, ....., m$), ancho y alto

### 5.2.2. Problemas de corte de material guillotinable con restricciones

Dado un conjunto de requerimientos rectangulares y un número ilimitado de láminas rectangulares de dimensiones mayores a los requerimientos. El problema de corte de material guillotinable consiste en realizar cortes rectos de extremo a extremo sobre las láminas hasta obtener todos los requerimientos con el menor número de láminas. El problema es NP-Difícil y presenta diversas aplicaciones en los diversos sectores de la industria como la metal mecánica, maderera, vidrios, entre otros.

#### Desempeño de heurísticas individuales

En la tabla 5.10 se muestra un cuadro comparativo del desempeño de las distintas heurísticas simples para los problemas cgcut1-cgcut3. Se puede observar que para los problemas cgcut1 y cgcut2, las heurísticas que proponen un orden de acomodo descendente en la selección de piezas ($FFD$, $Filler + FFD$ y $DJD$) ofrecen excelentes resultados al acomodar las piezas en el número óptimo de objetos, independientemente si las piezas son rotadas o no. En el problema cgcut1, el 45 % de las heurísticas acomodan por si solas a todas las piezas en el número óptimo de objetos. En el problema cgcut2, no es conveniente utilizar heurísticas de acomodo que roten a las piezas, esto se debe a que los objetos son altos y angostos, y a que en las piezas de mayor área, el valor de la altura es mayor que el de la anchura.

Para el problema cgcut3, las heurísticas que proponen los mejores resultados son aquellas que hacen un ordenamiento descendente y rotan las piezas antes de acomodarlas en los objetos.

Tabla 5.10: Desempeño de heurísticas individuales para los problemas cgcut1-cgcut3.

| *Heurística* | | | *Problema* | | |
|---|---|---|---|---|---|
| No. | H. de selección | H. de acomodo | cgcut1 | cgcut2 | cgcut3 |
| 1 | FF | BL | **2** | 3 | 33 |
| 2 | | BLR | 3 | 3 | 25 |
| 3 | | BLLT | **2** | 3 | 33 |
| 4 | | BLLTR | 3 | 3 | 25 |
| 5 | FFD | BL | **2** | **2** | 24 |
| 6 | | BLR | **2** | **2** | **20** |
| 7 | | BLLT | **2** | **2** | 24 |
| 8 | | BLLTR | **2** | **2** | **20** |
| 9 | FFI | BL | 3 | **2** | 33 |
| 10 | | BLR | 3 | 3 | 24 |
| 11 | | BLLT | 3 | **2** | 33 |
| 12 | | BLLTR | 3 | 3 | 24 |
| 13 | Filler + FFD | BL | **2** | **2** | 24 |
| 14 | | BLR | **2** | **2** | **20** |
| 15 | | BLLT | **2** | **2** | 24 |
| 16 | | BLLTR | **2** | **2** | **20** |
| 17 | NF | BL | **2** | 3 | 33 |
| 18 | | BLR | **2** | 3 | 24 |
| 19 | | BLLT | **2** | 3 | 33 |
| 20 | | BLLTR | **2** | 3 | 24 |
| 21 | NFD | BL | **2** | **2** | 24 |
| 22 | | BLR | **2** | 3 | **20** |
| 23 | | BLLT | **2** | **2** | 24 |
| 24 | | BLLTR | **2** | 3 | **20** |
| 25 | BF | BL | 4 | 3 | 34 |
| 26 | | BLR | 3 | 4 | 24 |
| 27 | | BLLT | 4 | 3 | 34 |
| 28 | | BLLTR | 3 | 4 | 24 |
| 29 | BFD | BL | 3 | 3 | 25 |
| 30 | | BLR | 3 | 4 | **20** |
| 31 | | BLLT | 3 | 3 | 25 |
| 32 | | BLLTR | 3 | 4 | **20** |
| 33 | WF | BL | 3 | 3 | 34 |
| 34 | | BLR | 3 | 4 | 23 |
| 35 | | BLLT | 3 | 3 | 34 |
| 36 | | BLLTR | 3 | 4 | 23 |
| 37 | DJD | BL | 3 | 3 | 24 |
| 38 | | BLR | 3 | 3 | **20** |
| 39 | | BLLT | 3 | 3 | 24 |
| 40 | | BLLTR | 3 | 3 | **20** |
| **Límite inferior** ($L_o$) | | | **2** | **2** | **16** |

### Desempeño de hiper-heurística

En la tabla 5.11 se muestra un cuadro comparativo entre el número de contenedores utilizados en las mejores soluciones reportadas por Martello y Vigo ($z$) [20], la mejor heurística ($BH$), la hiper-heurística encontrada por el sistema de clasificadores XCS bajo el esquema de pago en un paso ($HH-XCS-SS$) y la hiper-heurística encontrada por el sistema de clasificadores XCS bajo el esquema de pago en múltiples pasos ($HH-XCS-MS$) para los problemas cgcut1-cgcut3. Se puede observar que la solución encontrada por $HH-XCS$ utiliza el mismo número de objetos para satisfacer la demanda de piezas que el encontrado por la mejor de las heurísticas para este tipo de problemas.

Tabla 5.11: Resultados en instancias de problemas de corte en dos dimensiones guillotinables con restricciones.

| Problema | cgcut1 | cgcut2 | cgcut3 |
|---|---|---|---|
| n | 16 | 23 | 62 |
| Límite inferior ($L_o$) | 2 | 2 | 16 |
| $z$ | 2 | 2 | 23 |
| $BH$ | 2 | 2 | 20 |
| $HH-XCS-SS$ | **2** | **2** | 20 |
| $HH-XCS-MS$ | **2** | **2** | 20 |

Tabla 5.12: Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte de material guillotinable con restricciones.

| Problema | cgcut1 | cgcut2 | cgcut3 |
|---|---|---|---|
| $HH-XCS-SS$ | 2125.0 | 2000.0 | 9422.0 |
| $HH-XCS-MS$ | 281.0 | 4125.0 | 36047.0 |

Las heurísticas de mejor desempeño ($BH$) en todos los casos para este tipo de problemas son $FFD$ y $Filler+FFD$ combinadas con heurísticas de acomodo que no utilizan la propiedad de rotar las piezas.

En la tabla 5.12 se muestra el tiempo promedio (en milisegundos) requerido para hallar la mejor solución encontrada por la hiper-heurística mediante el sistema de clasificadores XCS ($HH-XCS$). Se puede observar que para los problemas cgcut1 y cgcut2, se obtuvo en un tiempo promedio menor a 5 segundos, para todas las corridas, es decir, siempre le son fáciles de resolver. Sin embargo, para el problema cgcut3, el algoritmo $HH-XCS$ encuentra la mejor solución reportada en la tabla 5.11, después

de varios segundos, esto se debe a que el número de piezas a acomodar es mucho mayor, lo cual implica que el número de objetos a utilizar será mayor y por lo tanto se utilice un mayor número de heurísticas.

### Análisis de resultados para el ambiente de un paso

Se corrieron experimentos realizando el pago por el desempeño de cada una de las heurísticas conforme iban siendo seleccionadas y al final de cada experimento se obtuvo una hiper-heurística que representaba una solución al problema. Se seleccionaron a aquellas hiper-heurísticas que propusieron el menor número de objetos en la solución. Para calcular qué tan frecuente es utilizada cada heurística en este conjunto de hiper-heurísticas, se sumó el número total de veces que el sistema de clasificadores utiliza a cada una de las heurísticas en las hiper-heurísticas y se dividió este resultado entre el número total de heurísticas utilizadas. En las gráficas 5.1 y 5.2, se muestra el porcentaje de utilización de cada una de las heurísticas en el conjunto de hiper-heurísticas para los problemas cgcut1 y cgcut3 respectivamente, cuando se realiza el pago de esta manera.

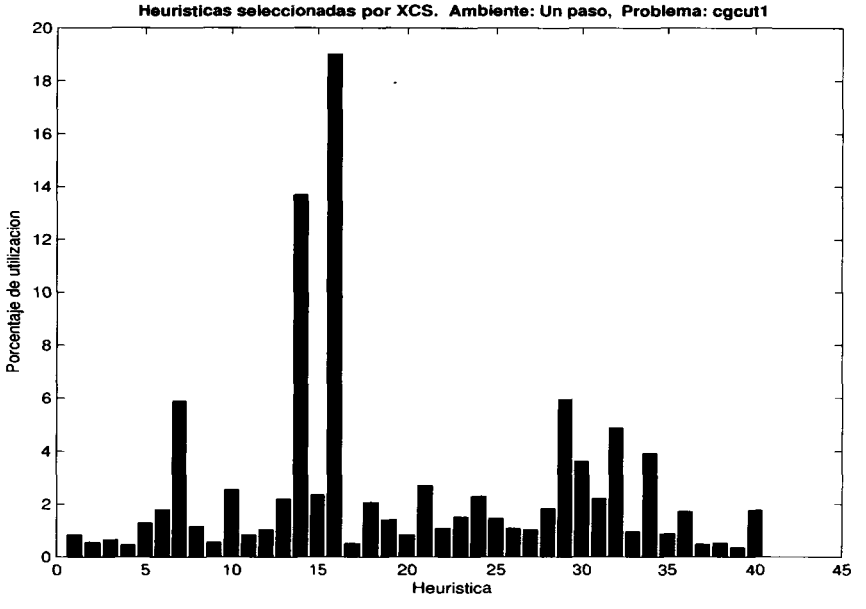



Figura 5.1: Heurísticas seleccionadas por el XCS en un ambiente de un paso para el problema cgcut1.

En la gráfica de la figura 5.1, se puede apreciar que las heurísticas utilizadas con mayor frecuencia para encontrar la mejor solución para el problema cgcut1 son la número 14 ($Filler+FFD$ combinada con $BLR$) y la número 16 ($Filler+FFD$ en combinación con$+BLLTR$. Analizando el comportamiento de las heurísticas individuales (tabla 5.10), se puede observar que estas heurísticas proponen una buena solución para

todos los problemas (cgcut1-cgcut3). Estos resultados reflejan que el sistema de clasificadores XCS identifica de manera eficiente las propiedades de las mejores heurísticas para las diferentes instancias del problema.

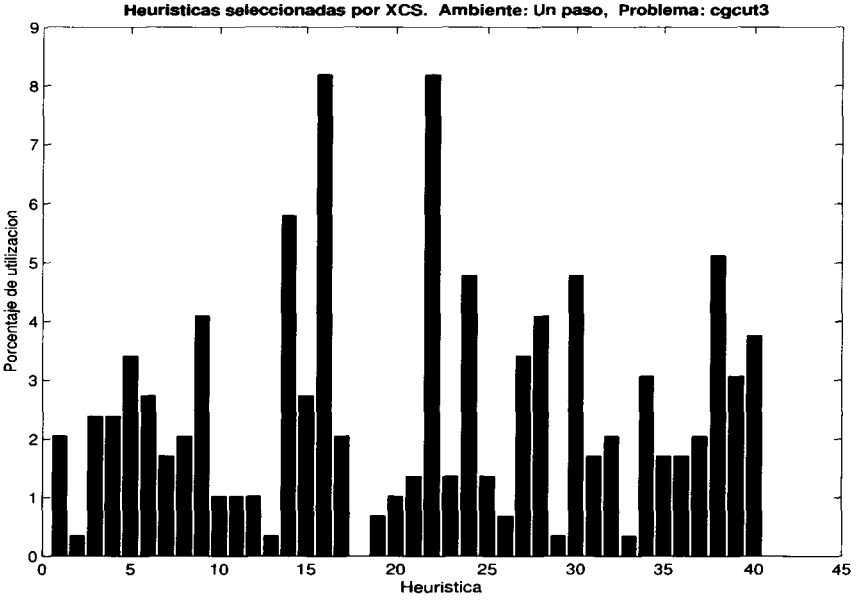



Figura 5.2: Heurísticas seleccionadas por el XCS en un ambiente de un paso para el problema cgcut3.

### Análisis de resultados para el ambiente de múltiples pasos

Como se mencionó en la sección 4.2.2, en problemas de múltiples pasos, el pago ocurre ya que se ha resuelto totalmente el problema. Para cada uno de los problemas de corte de material guillotinable (cgcut1-cgcut3) se seleccionaron a aquellas hiper-heurísticas que propusieron el menor número de objetos en la solución al problema. Como en el ambiente de un paso, para calcular que tan frecuente es utilizada cada una de las heurísticas simples en este conjunto de hiper-heurísticas, se sumó el número total de veces que aparece cada heurística en este conjunto de hiper-heurísticas y se dividió el resultado entre el número total de heurísticas utilizadas. En las gráficas de las figuras 5.3, 5.4 y 5.5 se muestran los resultados obtenidos en la solución a los problemas de corte de material cgcut1, cgcut1 y cgcut3 respectivamente, asignando el pago de esta manera. Se puede observar, que a diferencia del ambiente de un paso, al sistema de clasificadores le es difícil identificar cuál heurística es la que tiene mejor rendimiento, por lo que se enfoca a buscar la combinación de estas que mejor solucionen el problema.

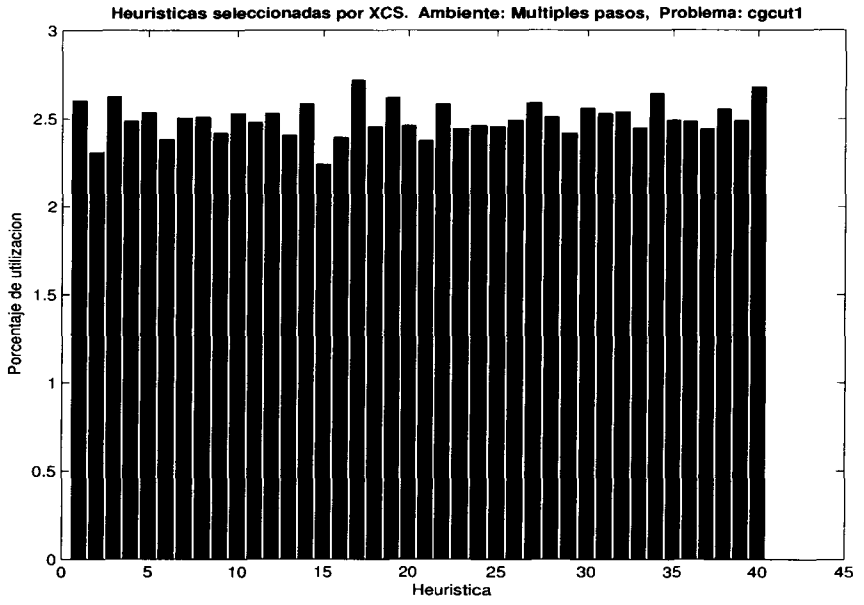



Figura 5.3: Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema cgcut1.

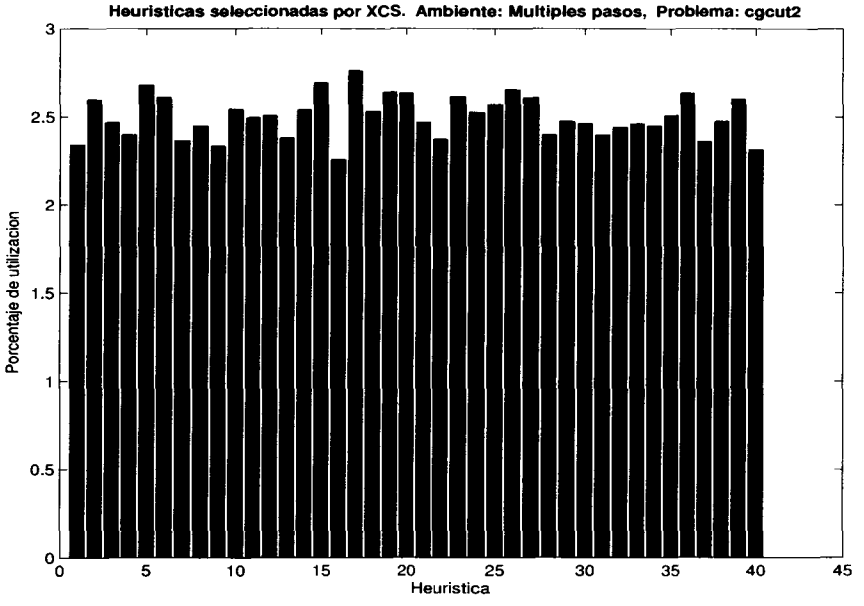



Figura 5.4: Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema cgcut2.

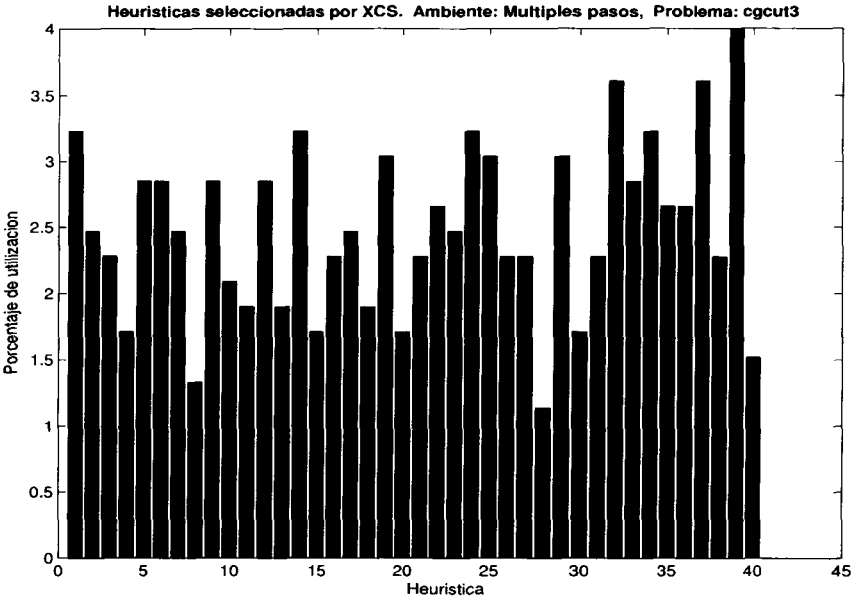



Figura 5.5: Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema cgcut3.

Analizando los resultados se puede apreciar que el sistema de clasificadores interactuando con un ambiente de múltiples pasos, identifica de manera eficiente la mejor combinación de heurísticas que solucionan este problema de corte de material.

### 5.2.3. Problemas de corte de material guillotinable sin restricciones

#### Desempeño de heurísticas individuales

En la tabla 5.13 se muestra una tabla comparativa del desempeño de las distintas heurísticas simples para los problemas gcut1-gcut7. Se puede observar que las heurísticas de selección $FFD$, $Filler + FFD$ y $NFD$ combinadas con heurísticas de acomodo que consideran la posibilidad de rotación proponen una buena solución para los problemas gcut1 y gcut2, por lo tanto se podría pensar que para estos problemas es conveniente rotar las piezas antes de ser acomodadas cuando se soluciona el problema con heurísticas simples.

En los problemas gcut3-gcut11, las heurísticas de selección $FFD$ y $Filler + FFD$ ofrecen buenos resultados en combinación con las heurísticas de acomodo de abajo-izquierda sin importar si las piezas son rotadas o no. En el problema gcut5, estas heurísticas encuentran por si solas el limite inferior continuo. Para el problema gcut11 la mejor solución es obtenida por algoritmo de selección $DJD$ en combinación con las heurísticas de acomodo que rotan las piezas antes de colocarlas dentro del objeto.

En la tabla 5.13 se puede observar que los mejores resultados para el problema gcut13, se obtienen cuando se utiliza la heurística de acomodo BLLT, esto probablemente se debe a que el tamaño de las piezas es demasiado pequeño respecto a los objetos, lo que permite que las piezas entrantes vayan llenando los espacios libres de manera eficiente al ser desplazadas sobre el borde de las piezas ya acomodadas.

Tabla 5.13: Desempeño de heurísticas individuales para los problemas gcut1-gcut7.

| *Heurística* | | | *Problema* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| No. | Selección | Acomodo | gcut1 | gcut2 | gcut3 | gcut4 | gcut5 | gcut6 | gcut7 |
| 1 | FF | BL | 5 | 7 | 11 | 16 | 4 | 8 | 13 |
| 2 | | BLR | 5 | 7 | 10 | 17 | 4 | 8 | 13 |
| 3 | | BLLT | 5 | 7 | 11 | 16 | 4 | 8 | 13 |
| 4 | | BLLTR | 5 | 7 | 10 | 17 | 4 | 8 | 13 |
| 5 | FFD | BL | 5 | 7 | **8** | 14 | **3** | **7** | **11** |
| 6 | | BLR | **4** | 6 | **8** | 14 | **3** | **7** | **11** |
| 7 | | BLLT | 5 | 7 | **8** | 14 | **3** | **7** | **11** |
| 8 | | BLLTR | **4** | 6 | **8** | 14 | **3** | **7** | **11** |
| 9 | FFI | BL | 6 | 9 | 12 | 22 | 4 | 10 | 16 |
| 10 | | BLR | 6 | 8 | 13 | 21 | 4 | 9 | 16 |
| 11 | | BLLT | 6 | 9 | 12 | 22 | 4 | 10 | 16 |
| 12 | | BLLTR | 6 | 8 | 13 | 21 | 4 | 9 | 16 |
| 13 | Filler + FFD | BL | 5 | 7 | **8** | 14 | **3** | **7** | **11** |
| 14 | | BLR | **4** | 6 | **8** | 14 | **3** | **7** | **11** |
| 15 | | BLLT | 5 | 7 | **8** | 14 | **3** | **7** | **11** |
| 16 | | BLLTR | **4** | 6 | **8** | 14 | **3** | **7** | **11** |
| 17 | NF | BL | 5 | 7 | 11 | 16 | 4 | 8 | 13 |
| 18 | | BLR | 5 | 7 | 11 | 17 | 5 | 8 | 13 |
| 19 | | BLLT | 5 | 7 | 11 | 16 | 4 | 8 | 13 |
| 20 | | BLLTR | 5 | 7 | 11 | 17 | 5 | 8 | 13 |
| 21 | NFD | BL | 5 | 7 | **8** | 14 | 4 | **7** | 12 |
| 22 | | BLR | **4** | 6 | **8** | 14 | 4 | 8 | 12 |
| 23 | | BLLT | 5 | 7 | **8** | 14 | 4 | **7** | 12 |
| 24 | | BLLTR | **4** | 6 | **8** | 14 | 4 | 8 | 12 |
| 25 | BF | BL | 7 | 8 | 11 | 17 | 5 | 10 | 15 |
| 26 | | BLR | 7 | 9 | 11 | 17 | 4 | 10 | 14 |
| 27 | | BLLT | 7 | 8 | 11 | 17 | 5 | 10 | 15 |
| 28 | | BLLTR | 7 | 9 | 11 | 17 | 4 | 10 | 14 |
| 29 | BFD | BL | 6 | 8 | 9 | 15 | 5 | 8 | 13 |
| 30 | | BLR | 5 | 7 | 10 | 16 | 4 | 8 | 13 |
| 31 | | BLLT | 6 | 8 | 9 | 15 | 5 | 8 | 13 |
| 32 | | BLLTR | 5 | 7 | 10 | 16 | 4 | 8 | 13 |
| 33 | WF | BL | 7 | 7 | 11 | 16 | 4 | 10 | 13 |
| 34 | | BLR | 7 | 9 | 10 | 16 | 4 | 10 | 13 |
| 35 | | BLLT | 7 | 8 | 11 | 16 | 4 | 10 | 13 |
| 36 | | BLLTR | 7 | 9 | 10 | 16 | 4 | 10 | 13 |
| 37 | DJD | BL | 5 | 7 | 9 | 16 | **3** | 8 | 12 |
| 38 | | BLR | **4** | 6 | 10 | 15 | 4 | **7** | **11** |
| 39 | | BLLT | 5 | 7 | 9 | 16 | **3** | 8 | 12 |
| 40 | | BLLTR | **4** | 6 | 10 | 15 | 4 | **7** | **11** |
| **Límite inferior** ($L_o$) | | | **3** | **5** | **7** | **12** | **3** | **6** | **7** |

Tabla 5.14: Desempeño de heurísticas individuales para los problemas gcut1-gcut4.

| *Heurística* | | | *Problema* | | | | | |
|---|---|---|---|---|---|---|---|---|
| No. | Selección | Acomodo | gcut8 | gcut9 | gcut10 | gcut11 | gcut12 | gcut13 |
| 1 | FF | BL | 16 | 4 | **8** | 10 | 18 | 3 |
| 2 | | BLR | 16 | 4 | **8** | 10 | 19 | 3 |
| 3 | | BLLT | 16 | 4 | **8** | 10 | 18 | 3 |
| 4 | | BLLTR | 16 | 4 | **8** | 10 | 19 | 3 |
| 5 | FFD | BL | **14** | **3** | **8** | 9 | 17 | 3 |
| 6 | | BLR | **14** | **3** | **8** | 9 | **16** | 3 |
| 7 | | BLLT | **14** | **3** | **8** | 9 | 17 | **2** |
| 8 | | BLLTR | **14** | **3** | **8** | 9 | **16** | 3 |
| 9 | FFI | BL | 20 | 5 | 10 | 13 | 24 | 3 |
| 10 | | BLR | 20 | 4 | 10 | 12 | 24 | 3 |
| 11 | | BLLT | 20 | 5 | 10 | 13 | 24 | 3 |
| 12 | | BLLTR | 20 | 4 | 10 | 12 | 24 | 3 |
| 13 | Filler + FFD | BL | **14** | **3** | **8** | 9 | 17 | 3 |
| 14 | | BLR | **14** | **3** | **8** | 9 | **16** | 3 |
| 15 | | BLLT | **14** | **3** | **8** | 9 | 17 | **2** |
| 16 | | BLLTR | **14** | **3** | **8** | 9 | **16** | 3 |
| 17 | NF | BL | 16 | 4 | **8** | 10 | 18 | 3 |
| 18 | | BLR | 16 | 4 | **8** | 9 | 19 | 3 |
| 19 | | BLLT | 16 | 4 | **8** | 10 | 18 | 3 |
| 20 | | BLLTR | 16 | 4 | **8** | 9 | 19 | 3 |
| 21 | NFD | BL | 15 | **3** | 8 | 9 | 17 | 3 |
| 22 | | BLR | **14** | 4 | 9 | 9 | **16** | 3 |
| 23 | | BLLT | 15 | **3** | 8 | 9 | 17 | **2** |
| 24 | | BLLTR | **14** | 4 | 9 | 9 | **16** | 3 |
| 25 | BF | BL | 17 | 4 | 9 | 12 | 19 | 4 |
| 26 | | BLR | 17 | 4 | 9 | 10 | 19 | 4 |
| 27 | | BLLT | 17 | 4 | 9 | 12 | 19 | 4 |
| 28 | | BLLTR | 17 | 4 | 9 | 10 | 19 | 4 |
| 29 | BFD | BL | 16 | 4 | 9 | 10 | 18 | 4 |
| 30 | | BLR | 15 | 5 | 9 | 9 | 17 | 4 |
| 31 | | BLLT | 16 | 4 | 9 | 10 | 18 | 3 |
| 32 | | BLLTR | 15 | 5 | 9 | 9 | 17 | 4 |
| 33 | WF | BL | 15 | 4 | 9 | 12 | 19 | 3 |
| 34 | | BLR | 15 | 4 | **8** | 10 | 19 | 3 |
| 35 | | BLLT | 15 | 4 | 9 | 12 | 19 | 3 |
| 36 | | BLLTR | 16 | 4 | **8** | 10 | 19 | 3 |
| 37 | DJD | BL | 15 | **3** | **8** | 10 | 19 | 3 |
| 38 | | BLR | 16 | **3** | 9 | **8** | 18 | 3 |
| 39 | | BLLT | 15 | **3** | **8** | 10 | 19 | 3 |
| 40 | | BLLTR | 16 | **3** | 9 | **8** | 18 | 3 |
| **Límite inferior** ($L_o$) | | | **12** | **3** | **6** | **7** | **13** | **2** |

En las tablas 5.13 y 5.14 se puede apreciar que de manera general, las heurísti-

cas $FFD$ y $Filler + FFD$ presentan una mejor solución que las demás heurísticas de selección para los problemas de tipo guillotinables sin restricciones.

### Desempeño de hiper-heurística

Los resultados presentados en las tablas 5.15 y 5.16 muestran que las soluciones propuestas por la hiper-heurística encontrada por el sistema de clasificadores XCS para los problemas gcut4, gcut6, gcut8 y gcut10 utilizan un menor número de objetos para satisfacer la demanda de piezas que si se utilizara a la mejor de las heurísticas para este tipo de problemas, para los problemas gcut5, gcut9 y gcut13 se obtiene el límite inferior continuo.

Tabla 5.15: Resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones.

| Problema | gcut1 | gcut2 | gcut3 | gcut4 | gcut5 | gcut6 | gcut7 |
|---|---|---|---|---|---|---|---|
| n | 10 | 20 | 30 | 50 | 10 | 20 | 30 |
| Límite inferior ($L_o$) | 3 | 5 | 7 | 12 | 3 | 5 | 9 |
| $z$ | 5 | 6 | 8 | 14 | 3 | 7 | 11 |
| $BH$ | 4 | 6 | 8 | 14 | 3 | 7 | 11 |
| $HH-XCS-SS$ | 4 | 6 | 8 | **13** | 3 | **6** | 11 |
| $HH-XCS-MS$ | 4 | 6 | 8 | **13** | 3 | **6** | 11 |

Tabla 5.16: Resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones.

| Problema | gcut8 | gcut9 | gcut10 | gcut11 | gcut12 | gcut13 |
|---|---|---|---|---|---|---|
| n | 50 | 10 | 20 | 30 | 50 | 32 |
| Límite inferior ($L_o$) | 12 | 3 | 6 | 7 | 13 | 2 |
| $z$ | – | 3 | 7 | 9 | 16 | 2 |
| $BH$ | 14 | 3 | 8 | 8 | 16 | 2 |
| $HH-XCS-SS$ | **13** | **3** | **7** | **8** | **16** | **2** |
| $HH-XCS-MS$ | **13** | **3** | **7** | **8** | **16** | **2** |

Tabla 5.17: Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones.

| Problema | gcut1 | gcut2 | gcut3 | gcut4 | gcut5 | gcut6 | gcut7 |
|---|---|---|---|---|---|---|---|
| $HH-XCS-SS$ | 812.0 | 703.0 | 1172.0 | 14328.0 | 2031.0 | 502781.0 | 142594.0 |
| $HH-XCS-MS$ | 219.0 | 1344.0 | 1500.0 | 51688.0 | 469.0 | 146344.0 | 4614422.0 |

Tabla 5.18: Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones.

| Problema | gcut8 | gcut9 | gcut10 | gcut11 | gcut12 | gcut13 |
|---|---|---|---|---|---|---|
| $HH-XCS-SS$ | 25328.0 | 1875.0 | 1612797.0 | 23844.0 | 3613000 | 254896.0 |
| $HH-XCS-MS$ | 4.478E7 | 10219.0 | 5518625.0 | 184750.0 | 3.361e8 | 5278512.0 |

### Análisis de resultados para el ambiente de un paso

En la figura 5.6 se muestra el comportamiento del XCS en cuanto a la frecuencia de selección de heurísticas para formar las hiper-heurísticas de mejor desempeño cuando interactúa con un ambiente en donde el pago por la calidad de la solución es obtenido cada vez que se propone una heurística. En esta misma figura se puede observar que las heurísticas seleccionadas con menor frecuencia son la 25,27 y 34, observando el comportamiento de estas heurísticas individuales (tabla 5.13) se encuentra que precisamente son aquellas con el menor desempeño. Del mismo modo, las heurísticas 14 y 16, que son seleccionadas con mayor frecuencia por el sistema de clasificadores, presentan individualmente el mejor desempeño.

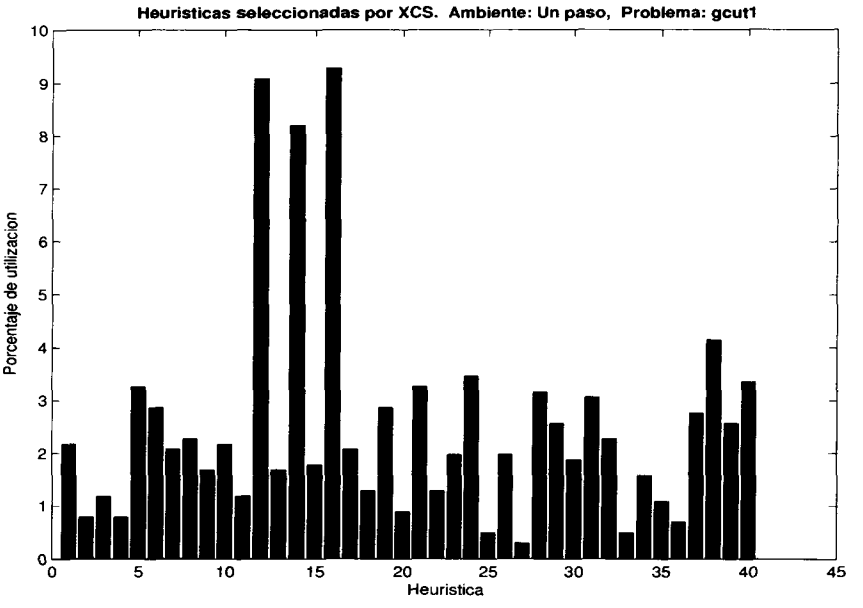



Figura 5.6: Heurísticas seleccionadas por el XCS en un ambiente de un paso para el problema gcut1.

### Análisis de resultados para el ambiente de múltiples pasos

En la figura 5.7 se muestra la frecuencia en la que son seleccionadas las heurísticas para formar una hiper-heurística cuando el pago por la calidad de la solución es obtenido una vez que se ha terminado de solucionar el problema completo.

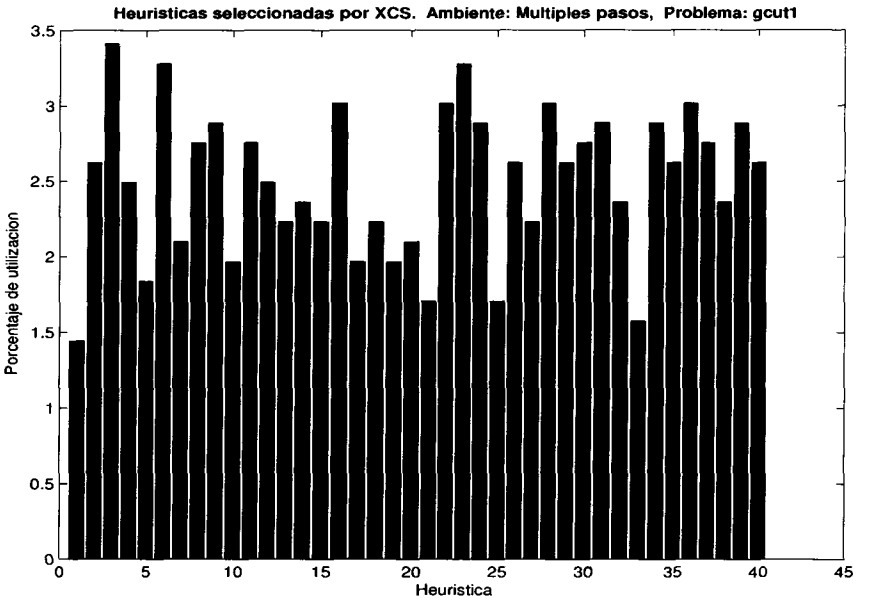



Figura 5.7: Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema gcut1.

### 5.2.4. Problemas de corte de material no guillotinable

En los problemas de corte de material no guillotinable no se pueden realizar cortes rectos de extremo a extremo sobre las láminas. En este tipo de problemas es más complicado realizar los cortes, pues el desplazamiento de las cuchillas se debe de acoplar a la forma de cada una de las piezas.

#### Desempeño de heurísticas individuales

En la tabla 5.19 se muestra una cuadro comparativo del desempeño de las distintas heurísticas simples para los problemas ngcut. Se puede observar que las heurísticas de selección $FF$, $NF$ y $DJD$ proponen una buena solución para el problema ngcut1 y ngcut2. Para los problemas ngcut5 y ngcut11 los mejores resultados son obtenidos de la combinación de las heurísticas de selección $FFD$, $Filler + FFD$, $NFD$ y $DJD$ con las heurísticas de acomodo $BL$ y $BLT$. Analizando las características de los dos últimos problemas (Apéndice A) podemos observar que son muy similares en cuanto a la forma de sus piezas y al número de éstas. De acuerdo a estos resultados, se puede observar que existen heurísticas que son muy eficientes para ayudar a resolver algunos problemas, pero para algunos otros puede que no sean tan eficientes.

Tabla 5.19: Desempeño de heurísticas individuales para los problemas ngcut1-ngcut6.

| *Heurística* | | | *Problema* | | | | |
|---|---|---|---|---|---|---|---|
| No. | H. de selección | H. de acomodo | ngcut1 | ngcut2 | ngcut3 | ngcut5 | ngcut6 |
| 1 | FF | BL | **3** | **4** | 4 | 4 | **3** |
| 2 | | BLR | **3** | **4** | 4 | 5 | 4 |
| 3 | | BLLT | **3** | **4** | 4 | 4 | **3** |
| 4 | | BLLTR | **3** | **4** | 4 | 5 | 4 |
| 5 | FFD | BL | 4 | **4** | 4 | **3** | **3** |
| 6 | | BLR | **3** | **4** | 4 | 4 | **3** |
| 7 | | BLLT | 4 | **4** | 4 | **3** | **3** |
| 8 | | BLLTR | **3** | **4** | 4 | 4 | **3** |
| 9 | FFI | BL | **3** | **4** | 4 | 4 | **3** |
| 10 | | BLR | 4 | 6 | 4 | 4 | 4 |
| 11 | | BLLT | **3** | **4** | 4 | 4 | **3** |
| 12 | | BLLTR | 4 | 6 | 4 | 4 | 4 |
| 13 | Filler + FFD | BL | 4 | **4** | 4 | **3** | **3** |
| 14 | | BLR | **3** | **4** | 4 | 4 | **3** |
| 15 | | BLLT | 4 | **4** | 4 | **3** | **3** |
| 16 | | BLLTR | **3** | **4** | 4 | 4 | **3** |
| 17 | NF | BL | **3** | **4** | 4 | 4 | **3** |
| 18 | | BLR | **3** | **4** | 4 | 7 | **3** |
| 19 | | BLLT | **3** | **4** | 4 | 4 | **3** |
| 20 | | BLLTR | **3** | **4** | 4 | 7 | **3** |
| 21 | NFD | BL | 4 | **4** | 4 | **3** | **3** |
| 22 | | BLR | **3** | **4** | 4 | 5 | **3** |
| 23 | | BLLT | 4 | **4** | 4 | **3** | **3** |
| 24 | | BLLTR | **3** | **4** | 4 | 5 | **3** |
| 25 | BF | BL | 4 | 5 | 5 | 5 | 4 |
| 26 | | BLR | 4 | 5 | 5 | 6 | 4 |
| 27 | | BLLT | 4 | **4** | 5 | 5 | **3** |
| 28 | | BLLTR | 4 | 5 | 5 | 6 | 4 |
| 29 | BFD | BL | 5 | 5 | 5 | 4 | 4 |
| 30 | | BLR | 4 | 5 | 5 | **3** | 4 |
| 31 | | BLLT | 5 | 5 | 5 | 4 | 4 |
| 32 | | BLLTR | 4 | 5 | 5 | **3** | 4 |
| 33 | WF | BL | **3** | **4** | 4 | 4 | **3** |
| 34 | | BLR | 4 | 5 | 4 | 4 | 4 |
| 35 | | BLLT | **3** | **4** | 4 | 4 | **3** |
| 36 | | BLLTR | 4 | 5 | 4 | 4 | 4 |
| 37 | DJD | BL | **3** | **4** | 5 | 3 | 4 |
| 38 | | BLR | **3** | **4** | 4 | 4 | **3** |
| 39 | | BLLT | **3** | **4** | 5 | 3 | 4 |
| 40 | | BLLTR | **3** | 4 | 4 | 4 | **3** |
| **Límite inferior** ($L_o$) | | | **2** | **3** | **3** | **3** | **2** |

Tabla 5.20: Desempeño de heurísticas individuales para los problemas ngcut8-ngcut12.

| *Heurística* | | | *Problema* | | | | |
|---|---|---|---|---|---|---|---|
| No. | H. de selección | H. de acomodo | ngcut8 | ngcut9 | ngcut10 | ngcut11 | ngcut12 |
| 1 | FF | BL | **2** | 4 | **3** | 4 | 4 |
| 2 | | BLR | **2** | 4 | **3** | 5 | 4 |
| 3 | | BLLT | **2** | 4 | **3** | 4 | 4 |
| 4 | | BLLTR | **2** | 4 | **3** | 5 | 4 |
| 5 | FFD | BL | **2** | 4 | 4 | **3** | 4 |
| 6 | | BLR | **2** | 4 | **3** | 4 | 4 |
| 7 | | BLLT | **2** | 4 | 4 | **3** | 4 |
| 8 | | BLLTR | **2** | 4 | **3** | 4 | 4 |
| 9 | FFI | BL | 3 | 4 | **3** | 4 | 4 |
| 10 | | BLR | 3 | 6 | **3** | 4 | 5 |
| 11 | | BLLT | 3 | 4 | **3** | 4 | 4 |
| 12 | | BLLTR | 3 | 6 | **3** | 4 | 5 |
| 13 | Filler + FFD | BL | **2** | 4 | 4 | **3** | 4 |
| 14 | | BLR | **2** | **3** | **3** | 4 | 4 |
| 15 | | BLLT | **2** | 4 | 4 | **3** | 4 |
| 16 | | BLLTR | **2** | **3** | **3** | 4 | 4 |
| 17 | NF | BL | **2** | 4 | **3** | 4 | 4 |
| 18 | | BLR | **2** | **3** | **3** | 7 | 4 |
| 19 | | BLLT | **2** | 4 | **3** | 4 | 4 |
| 20 | | BLLTR | **2** | **3** | **3** | 7 | 4 |
| 21 | NFD | BL | **2** | 4 | 4 | **3** | 4 |
| 22 | | BLR | **2** | 4 | 4 | 5 | 4 |
| 23 | | BLLT | **2** | 4 | 4 | **3** | 4 |
| 24 | | BLLTR | **2** | 4 | 4 | 5 | 4 |
| 25 | BF | BL | 3 | 5 | 4 | 5 | 5 |
| 26 | | BLR | 3 | 5 | 4 | 6 | 5 |
| 27 | | BLLT | 3 | 5 | 4 | 5 | 5 |
| 28 | | BLLTR | 3 | 5 | 4 | 6 | 5 |
| 29 | BFD | BL | 3 | 5 | 5 | 4 | 5 |
| 30 | | BLR | 3 | 5 | 4 | **3** | 6 |
| 31 | | BLLT | 3 | 5 | 5 | 4 | 5 |
| 32 | | BLLTR | 3 | 4 | 4 | **3** | 6 |
| 33 | WF | BL | **2** | 4 | **3** | 4 | 4 |
| 34 | | BLR | 3 | 4 | 4 | 4 | 4 |
| 35 | | BLLT | **2** | 4 | **3** | 4 | 4 |
| 36 | | BLLTR | 3 | 4 | 4 | 4 | 4 |
| 37 | DJD | BL | 3 | 4 | 4 | **3** | 5 |
| 38 | | BLR | 3 | 4 | **3** | **3** | 4 |
| 39 | | BLLT | 3 | 4 | 4 | **3** | 4 |
| 40 | | BLLTR | 3 | 4 | **3** | **3** | 4 |
| **Límite inferior** ($L_o$) | | | **2** | **3** | **2** | **2** | **3** |

### Desempeño de hiper-heurística

Los resultados presentados en las tablas 5.21 y 5.22 muestran que las soluciones propuestas por la hiper-heurística encontrada por el sistema de clasificadores XCS para los problemas ngcut. Se puede observar que para este tipo de problemas, el sistema de clasificadores encuentra el límite inferior ($L_0$) en casi todas las instancias.

Tabla 5.21: Resultados en instancias de problemas de corte en dos dimensiones no guillotinables.

| Problema | ngcut1 | ngcut2 | ngcut3 | ngcut4 | ngcut5 | ngcut6 |
|---|---|---|---|---|---|---|
| n | 50 | 17 | 21 | 7 | 14 | 15 |
| Límite inferior ($L_o$) | 2 | 3 | 3 | 2 | 3 | 2 |
| $z$ | 3 | 4 | 3 | 2 | 3 | 3 |
| $BH$ | 3 | 4 | 4 | - | 3 | 3 |
| $HH-XCS-SS$ | 3 | **3** | **3** | - | **3** | 3 |
| $HH-XCS-MS$ | 3 | **3** | **3** | - | **3** | 3 |

Tabla 5.22: Resultados en instancias de problemas de corte en dos dimensiones no guillotinables.

| Problema | ngcut7 | ngcut8 | ngcut9 | ngcut10 | ngcut11 | ngcut12 |
|---|---|---|---|---|---|---|
| n | 8 | 13 | 18 | 13 | 15 | 22 |
| Límite inferior ($L_o$) | 1 | 2 | 3 | 2 | 2 | 3 |
| $z$ | 1 | 2 | 3 | 3 | 2 | 3 |
| $BH$ | - | 2 | 3 | 3 | 3 | 4 |
| $HH-XCS-SS$ | - | **2** | **3** | **2** | **2** | **3** |
| $HH-XCS-MS$ | - | **2** | **3** | **2** | **2** | **3** |

Tabla 5.23: Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones no guillotinables.

| Problema | ngcut1 | ngcut2 | ngcut3 | ngcut4 | ngcut5 | ngcut6 |
|---|---|---|---|---|---|---|
| $HH-XCS-SS$ | 2000.0 | 3016.0 | 2860.0 | - | 2579.0 | 2203.0 |
| $HH-XCS-MS$ | 1125.0 | 8719.0 | 4234.0 | - | 2578.0 | 1937.0 |

Tabla 5.24: Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones no guillotinables.

| Problema | ngcut7 | ngcut8 | ngcut9 | ngcut10 | ngcut11 | ngcut12 |
|---|---|---|---|---|---|---|
| $HH - XCS - SS$ | - | 2484.0 | 2796.0 | 8828.0 | 1938.0 | 2578.0 |
| $HH - XCS - MS$ | - | 2266.0 | 2281.0 | 4157.0 | 2312.0 | 2453.0 |

## 5.3. Experimentos y análisis de resultados con problemas propuestos

Se crearon nuevos problemas de corte de material en dos dimensiones con el fin de analizar el comportamiento del algoritmo propuesto en el presente trabajo. Estos problemas de prueba se construyeron con la ayuda de algoritmos generadores de problemas.

Se distinguió e implemento un conjunto de algoritmos para la generación automática de problemas de corte en dos dimensiones. Estos permiten la creación de problemas guillotinables y no guillotinables, en donde la solución óptima es conocida. El problema generado es almacenado en archivos de texto que contienen el ancho y alto de los objetos y de cada una de las piezas.

Los generadores de problemas son flexibles en términos del tipo de problema y al tamaño de las piezas y objetos.

Un problema de empacado perfecto es aquel para el cuál se conoce por lo menos una solución óptima. La solución óptima en este trabajo se define como el arreglo de rectángulos en una serie de objetos, en donde el área total de los rectángulos es exactamente igual al área total de los objetos.

### 5.3.1. Problemas de corte de material guillotinable

Con el objetivo de crear problemas con una solución óptima, un objeto es seleccionado y dividido en rectángulos mas pequeños de acuerdo al siguiente algoritmo [17].

```
Calcula el número de rectángulos a producir
Mientras el número de rectángulos haya sido alcanzado:
   Obten el siguiente rectángulo ( el más grande )
   Selecciona un lado aleatorio del rectángulo
   Coloca un punto aleatorio en el lado seleccionado
   Construye una linea perpendicular a este lado desde el punto
   Crea 2 nuevos rectángulos mas pequeños
   Verifica que las dimensiones de los nuevos rectángulos
```

```
sean validos
    Si los rectángulos son válidos
    {
        Borra el rectángulo grande
        Agrega los dos rectángulos pequeños
    }
```

El número total de rectángulos se calcula de acuerdo a la siguiente ecuación.

$$n = 1 + i \tag{5.2}$$

en donde $n$ es el número de rectángulos en el problema y $i$ es el número de veces que se ejecuta el ciclo

Un ejemplo de problema de corte generado con este algoritmo se ilustra en la figura 5.9, se puede observar que siempre es posible cortar el objeto de extremo a extremo hasta obtener todas las piezas requeridas.

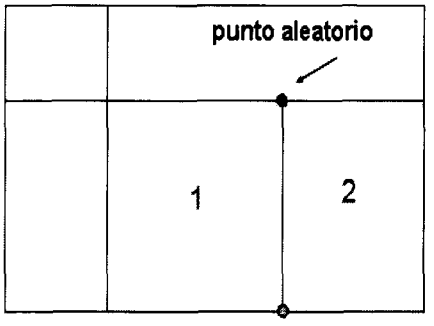



Figura 5.8: División de un nuevo rectángulo en dos nuevos rectángulos.

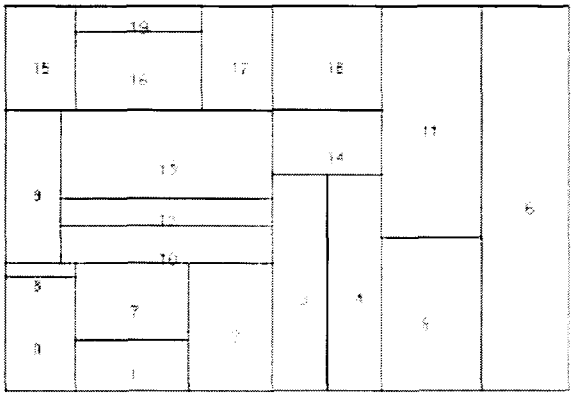


Figura 5.9: Ejemplo de problema de corte guillotinable generado.

### 5.3.2. Problemas de corte de material no guillotinable

Al igual que en la generación de problemas guillotinables, un objeto es seleccionado y este es dividido en rectángulos mas pequeños. La regla de diseño para los problemas no guillotinables se ilustra en la figura 5.10. La siguiente ecuación describe el número total de rectángulos creados por esta regla.

$$n = 1 + 4 * i \tag{5.3}$$

en donde $i$ es el número de veces que se ejecuta el ciclo y $n$ es el número de rectángulos en el problema.

El algoritmo utilizado para la generación de estos problemas se muestra a continuación.

```
Calcula el número de rectángulos a producir
Mientras el número de rectángulos haya sido alcanzado:
   Obten el siguiente rectángulo ( el más grande )
   Coloca dos puntos aleatorios dentro del rectángulo
   crea un nuevo rectángulo utilizando esos puntos
   Verifica que las dimensiones del rectángulo nuevo
    sean validos
        Si las dimensiones son válidas
        {
            Expande los puntos BL y TR: vertical y horizontal
                 Determina los puntos de intersección con
                 los lados del rectángulo grande
            Expande los puntos TL y BR
                 Determina los puntos de intersección con
                 los lados del rectángulo grande
            Crea cuatro nuevos rectángulos
            Borra el rectángulo grande
            Agrega los cinco rectángulos pequeños
        }
en donde
TR = superior derecha
TL = superior izquierda
BL = inferior izquierda
BR = inferior derecha
```

Un ejemplo de problema de corte generado con este algoritmo se ilustra en la figura 5.11. Se puede observar que no es posible cortar el objeto de extremos a extremo para obtener las piezas requeridas.

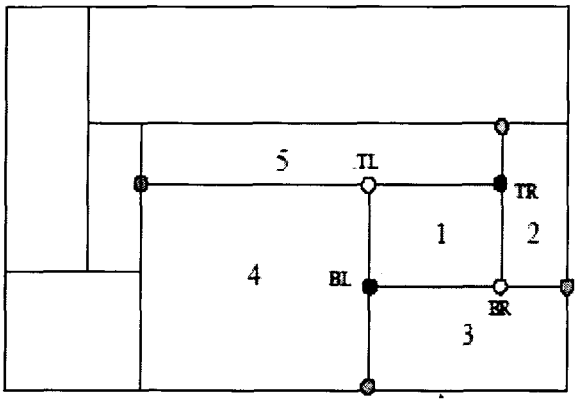



Figura 5.10: División de un nuevo rectángulo en cinco nuevos rectángulos para un problema no guillotinable.

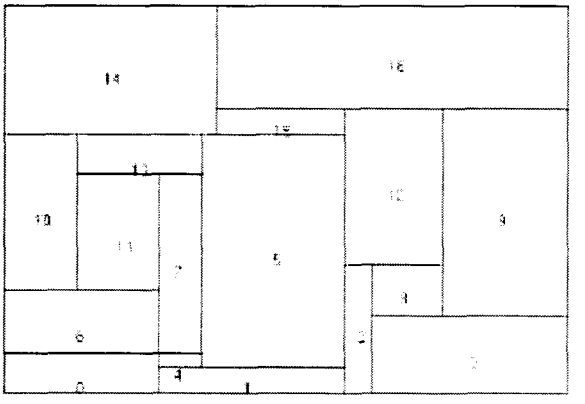


Figura 5.11: Ejemplo de problema de corte no guillotinable generado.

### 5.3.3. Desempeño de heurísticas individuales

En la tabla 5.25 se muestra un cuadro comparativo de el desempeño de las distintas heurísticas simples para los problemas propuestos. Los problemas $G1$, $G2$ y $G3$ son de tipo guillotinables y los problemas $NG1$, $NG2$ y $NG3$ son problemas no guillotinables.

Para el problema $G1$, los resultados muestran que generalmente es conveniente rotar las piezas antes de ser acomodadas. En las soluciones a los problemas $G1$ y $NG3$ se puede observar que el utilizar el algoritmo de acomodo BLLTR es favorable para resolver estas instancias de problemas, sin embargo, para el problema $G3$ no es conveniente utilizar esta heurística de acomodo pues provoca un incremento en el número de objetos necesarios para satisfacer la demanda de piezas.

Tabla 5.25: Desempeño de heurísticas individuales para los problemas propuestos.

| *Heurística* | | | *Problema* | | | | | |
|---|---|---|---|---|---|---|---|---|
| No. | H. de selección | H. de acomodo | G1 | G2 | G3 | NG1 | NG2 | NG3 |
| 1 | FF | BL | 7 | **13** | **19** | 7 | 12 | 18 |
| 2 | | BLR | 7 | 14 | **19** | 7 | 13 | 17 |
| 3 | | BLLT | 7 | **13** | **19** | 7 | 12 | 18 |
| 4 | | BLLTR | 7 | **13** | 20 | 7 | 12 | 18 |
| 5 | FFD | BL | 8 | **13** | **19** | **6** | **11** | 17 |
| 6 | | BLR | 7 | **13** | **19** | **6** | 12 | 17 |
| 7 | | BLLT | 8 | **13** | **19** | 7 | **11** | 17 |
| 8 | | BLLTR | 7 | **13** | **19** | **6** | 12 | **16** |
| 9 | FFI | BL | 9 | 15 | 24 | 8 | 16 | 21 |
| 10 | | BLR | 9 | 15 | 23 | 7 | 16 | 20 |
| 11 | | BLLT | 8 | 14 | 22 | 8 | 16 | 20 |
| 12 | | BLLTR | 9 | 15 | 23 | 7 | 15 | 21 |
| 13 | Filler + FFD | BL | 8 | **13** | **19** | **6** | **11** | 17 |
| 14 | | BLR | 7 | 14 | **19** | **6** | **11** | 17 |
| 15 | | BLLT | 8 | **13** | **19** | 7 | **11** | 17 |
| 16 | | BLLTR | 7 | 14 | 20 | **6** | **11** | 17 |
| 17 | NF | BL | 7 | **13** | **19** | 7 | 12 | 18 |
| 18 | | BLR | 7 | 14 | **19** | 7 | 13 | 17 |
| 19 | | BLLT | 7 | **13** | **19** | 7 | 12 | 18 |
| 20 | | BLLTR | **6** | 14 | 20 | 7 | 12 | 18 |
| 21 | NFD | BL | 8 | **13** | 20 | **6** | **11** | 17 |
| 22 | | BLR | 7 | 14 | 20 | **6** | 12 | 16 |
| 23 | | BLLT | 8 | **13** | 20 | 7 | **11** | 17 |
| 24 | | BLLTR | 7 | **13** | **19** | **6** | 12 | **16** |
| 25 | BF | BL | 9 | 15 | 21 | 8 | 13 | 18 |
| 26 | | BLR | 8 | 14 | 21 | 7 | 13 | 19 |
| 27 | | BLLT | 9 | 15 | 20 | 8 | 13 | 18 |
| 28 | | BLLTR | 8 | 14 | 21 | 7 | 13 | 19 |
| 29 | BFD | BL | 9 | 14 | 21 | 7 | 12 | 18 |
| 30 | | BLR | 8 | 14 | 22 | 7 | 12 | 18 |
| 31 | | BLLT | 9 | 14 | 21 | 8 | 12 | 18 |
| 32 | | BLLTR | 8 | 14 | 20 | 7 | 12 | 18 |
| 33 | WF | BL | 7 | **13** | 19 | 7 | 12 | 18 |
| 34 | | BLR | 8 | **13** | 21 | 7 | 13 | 19 |
| 35 | | BLLT | 9 | **13** | 20 | 7 | 12 | 17 |
| 36 | | BLLTR | 8 | **13** | 20 | 7 | 13 | 19 |
| 37 | DJD | BL | 7 | 14 | 20 | **6** | 12 | 19 |
| 38 | | BLR | 7 | **13** | **19** | **6** | **11** | 18 |
| 39 | | BLLT | 7 | 14 | **19** | **6** | 12 | 19 |
| 40 | | BLLTR | 7 | **13** | 20 | **6** | **11** | 18 |
| **Límite inferior** ($L_o$) | | | **5** | **10** | **15** | **5** | **10** | **15** |

### 5.3.4. Desempeño de hiper-heurística

En la tabla 5.26 se muestra una tabla comparativa en donde se muestra el límite inferior ($L_o$), el mejor resultado obtenido por las heurísticas individuales ($BH$), el resultado obtenido por la hiper-heurística encontrada por el sistema de clasificadores XCS bajo el esquema de pago en un paso ($HH - XCS - SS$) y el resultado obtenido por la hiper-heurística encontrada por el sistema de clasificadores XCS bajo el esquema de pago en múltiples pasos ($HH - XCS - MS$).

En los problemasÑG1 yÑG2, la hiper-heurística obtiene el valor óptimo.

Tabla 5.26: Resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones.

| Problema | G1 | G2 | G3 | NG1 | NG2 | NG3 |
|---|---|---|---|---|---|---|
| n | 39 | 79 | 120 | 33 | 54 | 103 |
| Límite inferior ($L_o$) | 5 | 10 | 15 | 5 | 10 | 15 |
| $BH$ | 6 | 13 | 19 | 6 | 11 | 16 |
| $HH - XCS - SS$ | **6** | **12** | **17** | **5** | **10** | **16** |
| $HH - XCS - MS$ | **6** | **12** | **17** | **5** | **10** | **16** |

Tabla 5.27: Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones.

| Problema | G1 | G2 | G3 | NG1 | NG2 | NG3 |
|---|---|---|---|---|---|---|
| $HH - XCS - SS$ | 6859.0 | 242406.0 | 178766.0 | 13766.0 | 125094.0 | 24062.0 |
| $HH - XCS - MS$ | 8062.0 | 306812.0 | 513516.0 | 51937.0 | 637719.0 | 42047.0 |

## 5.4. Resumen

En este capítulo se mostraron los resultados obtenidos de una serie de experimentos con problemas de corte de materiales. Se obtuvieron y analizaron combinaciones de heurísticas para cada uno de los problemas de corte de material. Del mismo modo, se comparó el desempeño de las heurísticas simples. Finalmente se mostraron tablas comparativas de los mejores resultados encontrados por las diferentes técnicas mencionadas.

## Capítulo 2

# Antecedentes

En este capítulo se presentan las áreas que influyen en el trabajo de esta investigación, se describe brevemente la clasificación de los problemas de corte según sus características y el funcionamiento de heurísticas simples que se han desarrollado para resolver este problema. También se presentan los conceptos básicos de hiper-heurística, una nueva técnica en búsqueda y optimización.

## 2.1. Corte de Materiales en Dos Dimensiones

Analizando a los diferentes problemas y aplicaciones de corte de materiales, se puede observar que muchos de ellos tienen la misma estructura lógica. Para poder intercambiar información entre las diferentes aplicaciones y disciplinas, Dyckhoff [11] ha identificado características y propiedades comunes y sistemáticamente presenta una clasificación de los problemas de corte y empacado.

### 2.1.1. Clasificación de Dyckhoff

Dyckhoff [11] propone una clasificación para integrar y distinguir entre los diferentes tipos de problemas de corte. Hace un análisis para identificar propiedades comunes entre estos problemas y desarrolla una estructura lógica básica.

#### Estructura Lógica Básica

La estructura lógica básica puede ser determinada de la siguiente manera:

1. Existen dos grupos de información básica, cuyos elementos definen cuerpos geométricos de formas definidas (piezas) en una o más dimensiones.

   *a*) El material de donde van a ser cortadas las piezas (objetos).

   *b*) La lista de las piezas pequeñas.

2. El proceso de corte y empacado realiza combinaciones geométricas de las piezas de menor tamaño dentro del área de los objetos de mayor tamaño.

Distingue a los problemas de corte de acuerdo a sus dimensiones espaciales. También muestra la estrecha relación que existe entre los problemas de corte y empacado de materiales que resulta de la dualidad de material y espacio. En este caso, los problemas de corte pueden ser vistos como si se quisiera empacar a las piezas de menor tamaño, dentro de los objetos de mayor tamaño. Del mismo modo los problemas de empacado pueden ser vistos como corte de los objetos de mayor tamaño, en piezas de menor tamaño.

### Características elementales

**Dimensión.-** . La característica más importante es la dimensionalidad, esta se refiere al menor número de dimensiones de números reales necesarias para describir la geometría de las figuras. Los problemas de más de tres dimensiones pueden obtenerse cuando se considera al tiempo o al peso como dimensiones.

**Calidad de medición.-** Se puede distinguir a los problemas según la manera en que las piezas son medidas.

- Medición discreta. Por ejemplo números naturales.
- Medición continua. Por ejemplo números reales.

**Forma de las figuras.-** Otra característica de los problemas de corte y empacado es la forma de las piezas y de los objetos. La figura de una pieza o de un objeto esta definida como su representación geométrica en el espacio de dimensiones. La forma de una figura es determinada según su tamaño y su orientación. Es importante también distinguir entre figuras regulares y figuras irregulares. Las figuras regulares son aquellas que pueden ser descritas en unos cuantos parámetros. Las formas irregulares son aquellas que son asimétricas o no convexas. Ejemplos de formas regulares son círculos, triángulos y rectángulos.

**Surtido.-** El surtido de las piezas también es importante para caracterizar un problema de corte y empacado. El surtido esta dado por las formas y el número de figuras permitidas.

**Disponibilidad.-** El surtido de las piezas no dice nada en cuanto a la cantidad de los objetos y piezas consideradas. La disponibilidad de los objetos o piezas permitidas se refiere a:

- Límites superior e inferior en su cantidad.
- La secuencia u orden.

- La fecha en que el objeto debe de ser empacado o cortado.

**Restricción de patrones.-** Se distinguen cuatro grupos de restricción de patrones que pueden ser considerados para la construcción de combinaciones geométricas de pequeñas piezas en un objeto:

- Distancias mínimas o máximas entre pequeñas piezas o entre cortes que dividen a los objetos.
- La orientación de las pequeñas piezas en relación de unas con otras o en relación del objeto de mayor tamaño.
- La frecuencia de las pequeñas piezas en algún patron.
- El tipo y el número de cortes permitidos. De acuerdo a esto, se distinguen entre patrones de corte ortogonales y no ortogonales. Los cortes ortogonales pueden distinguirse en guillotinables o no guillotinables, la complejidad de los patrones de corte guillotinables depende en el número de direcciones de la cuchilla.

**Restricciones de asignación.-** La asignación de pequeñas piezas a objetos de mayor tamaño se puede realizar en dos pasos, el primero es arreglar a las pequeñas piezas en patrones y el segundo es asignar esos patrones a los objetos más indicados. Puede haber restricciones de acuerdo al tipo de asignación. Una propiedad importante en cuanto al tipo de asignación, tanto para las piezas como para los objetos, es saber si todas o solo algunas de ellas serán asignadas a un patron correspondiente.

**Objetivos.-** No siempre es fácil decidir si una características es geométrica o combinatoria. Los objetivos de los problemas de corte normalmente tienen aspectos tanto geométricos como combinatorios, además pueden ser atribuidos a las piezas, a los objetos o a los patrones. Un objetivo se refiere al criterio a ser maximizado o minimizado. Los criterios a ser satisfechos son tratados como restricciones del proceso de corte y empacado.

**Estado de la información y variabilidad.-** Se tiene que establecer si la información de un problema es determinística, estocástica o incierta y si es fija o variable en ciertos intervalos.

### Clasificaciones combinadas

La clasificación de Dyckhoff describe cuatro características importantes de los problemas de corte y empacado. Estas se muestran en la tabla 2.1.

Tabla 2.1: Características de los problemas de corte y empacado.

| *Característica* | *Símbolo* | *Descripción* |
|---|---|---|
| Dimensionalidad | 1 | Unidimensional |
| | 2 | Bidimensional |
| | 3 | Tridimensional |
| | N | N-dimensional con $N \geq 3$ |
| Tipo de asignación | B | Todos los objetos y una selección de piezas |
| | V | Una selección de objetos y todas las piezas |
| Ordenamiento de objetos grandes | O | Un objeto |
| | I | Figuras idénticas |
| | D | Diferentes figuras |
| Ordenamiento de objetos pequeños | F | Pocas piezas (de diferentes figuras) |
| | M | Muchas piezas de muchas figuras diferentes |
| | R | Muchas piezas de pocas figuras |
| | C | Figuras congruentes |

La característica mas importante es la dimensionalidad, pues con esta se define la geometría de los patrones. A los problemas de más de tres dimensiones son obtenidos cuando se expanden a dimensiones no espaciales como el tiempo o el peso.

El tipo de asignación describe si todos los objetos y piezas o únicamente una parte de ellos debe estar asignada para obtener la solución al problema. En $B$, una selección de pequeñas piezas tiene que ser combinada en patrones de tal manera que un patrón correspondiente es asignado a cada objeto de mayor tamaño. En $V$ todas las piezas tienen que ser combinadas en patrones que están asignados a una selección de objetos de mayor tamaño.

Tres clasificaciones se distinguen con respecto a la disponibilidad de los objetos.

- Únicamente existe un objeto (problema de *knapsack*).
- Todos los objetos tienen la misma figura.
- Todos los objetos tienen diferentes figuras.

El ordenamiento, no sólo considera la forma de los objetos y piezas, sino también su número. Cuatro características se distinguen de acuerdo al ordenamiento de las piezas pequeñas.

- Existen pocas piezas de diferentes figuras.
- Hay muchas piezas, la mayoría de ellas teniendo diferentes figuras.
- Hay muchas piezas pequeñas, pero de pocas figuras.

- Todas las piezas pequeñas son congruentes.

Combinando las características de dimensionalidad, asignación y ordenamiento, se pueden obtener $4X2X3X4 = 96$ diferentes características de los problemas de corte y empacado. A los problemas de corte y empacado se se les denotará con los cuatro símbolos respectivos $\alpha$ / $\beta$ / $\rho$ / $\phi$. Por ejemplo 3/B/O/F se refiere a un problema tridimensional en donde un objeto va a ser empacado con una sección de piezas.

### 2.1.2. El límite inferior continuo

El límite inferior conocido para el problema de corte en dos dimensiones es obtenido cuando se permite que cada pieza sea cortada en cualquier número de piezas más pequeñas. A este se le llama límite inferior continuo $L_o$ y puede ser calculado en tiempo $O(n)$.

$$L_o = \left\lceil \frac{\sum_{j=1}^{n} h_j w_j}{HW} \right\rceil \tag{2.1}$$

En donde: $H$ es la altura de los objetos, $W$ es la anchura de los objetos y cada una de las piezas $j \epsilon J = 1, ..., n$ tiene una altura $h_j \leq H$ y una anchura $w_j \leq W$.

## 2.2. Problemas NP-completos

Uno de los aspectos más importantes en la teoría de complejidad computacional es la definición de clases de problemas. En un problema de optimización se plantea la pregunta si existe o no una solución razonable. De acuerdo a la complejidad del algoritmo se pueden distinguir las siguientes clases:

### 2.2.1. Clase P

Esta clase describe a los problemas que pueden ser resueltos por algoritmos que requieren a lo más una función polinomial del tamaño de la entrada. La membresía de esta clase significa que aún instancias grandes del problema pueden ser resueltas con rutinas exactas. Incrementos en el tamaño del problema normalmente tienen un impacto pequeño en el tiempo computacional.

### 2.2.2. Clase NP

La definición de esta clase esta basada en la solución de un problema en lugar de estar basada en el algoritmo. Si la solución correcta (por ejemplo una respuesta afirmativa) puede ser revisada en tiempo polinomial, el problema es conocido como polinomialmente no determinístico (NP). A pesar de que los problemas en esta clase son

fáciles de verificar, no son fáciles de resolver. Bajo esta definición, la clase NP contiene a la clase P. Con el fin de comprender los problemas de clase NP-hard y NP completos, primero se introduce el concepto de reducción. Un problema X se reduce a X', si un algoritmo de tiempo polinomial para X' implica la existencia de un algoritmo de tiempo polinomial para X.

### 2.2.3. Clase NP-Hard

Los problemas para los cuales los problemas NP se reducen polinomialmente se conocen como NP-hard. Por lo tanto, un algoritmo de tiempo polinomial para un problema NP-hard, produciría un algoritmo para cada miembro de NP al mismo tiempo.

### 2.2.4. Clase NP-Completo

A los problemas que se encuentran en la clase NP-hard y también en la clase NP se les llama NP-completos. Se ha demostrado que el problema de empacado de cajas rectangulares es de tipo NP-completo.

El corte de materiales es uno de los primeros problemas para los cuales se encontraron algoritmos polinomiales que garantizaban estar a un factor constante de la solución óptima. Dado que ese factor constante es relativamente pequeño, estos algoritmos de aproximación y sus variantes son muy útiles en la práctica [1].

## 2.3. Heurísticas para problemas de corte

Existen muchas aproximaciones que se utilizan cuando se quiere resolver problemas de tipo NP-Completo. Aunque no exista un algoritmo de orden polinomial, pueden haber diferencias significativas en las complejidades de los algoritmos conocidos, uno podría, de manera tradicional, querer desarrollar el más eficiente posible. Nos podríamos concentrar en el comportamiento en el caso promedio en lugar de el comportamiento en el peor de los casos y buscar algoritmos que son mejores que otros de acuerdo a ese criterio o de manera más realista, podríamos buscar algoritmos que funcionen bien para entradas que ocurren regularmente.

Una manera de resolver problemas de optimización NP-Completos es mediante el uso de algoritmos rápidos (que corren en tiempo polinomial) que no garantizan dar la mejor solución, pero dan una cercana a la óptima, a estos algoritmos se le llama algoritmos de heurística. Una heurística es un argumento derivado de la experiencia que se utiliza para obtener conocimiento o algunos resultados, por medio de proposiciones inteligentes en lugar de seguir alguna fórmula preestablecida [1].

En algunas aplicaciones de corte de material, soluciones aproximadas son suficientes, especialmente cuando el tiempo requerido para encontrar la solución óptima es

considerado. Estas estrategias o heurísticas, normalmente son sencillas y directas, pero para algunos problemas ofrecen sorprendentemente buenos resultados. Muchas de ellas son heurísticas concisas. Es por eso que es muy común optar por utilizar algún tipo de heurística, sacrificando la garantía de encontrar la solución óptima, por la velocidad y garantizando obtener una solución de una calidad aceptable. Se ha desarrollado un gran número de heurísticas, algunas basadas en resultados experimentales o por un argumento basado en un problema específico para el cual la heurística en cuestión es aceptable.

Para el problema de corte de material en dos dimensiones, existen dos grupos de heurísticas: Las heurísticas de selección que se encargan de proponer cuál pieza es la más apropiada y en que objeto debe ser acomodada en un momento dado, y las heurísticas de acomodo que se encargan de buscar un espacio en el objeto, en el cual quepa la pieza, desperdiciando el menor material posible de acuerdo a diferentes criterios de evaluación.

### 2.3.1. Heurísticas de selección [17]

Existen diferentes algoritmos heurísticos para de selección de piezas y de objetos, la mayoría de estos asume que se conoce el total de piezas a ser cortadas y las dimensiones de cada una de ellas. A continuación se mencionan algunos de estos algoritmos.

#### Heurística Next Fit (NF)

En este algoritmo, el último objeto que fue abierto, es el que es utilizado para acomodar las piezas, por ejemplo: el objeto no vacío $O_j$ es el último objeto abierto con el índice mas grande $j$ (Asumimos que los objetos están indexados $O_1, O_2, ...$). Para empacar una pieza $a_i$, el algoritmo analiza el tamaño de la pieza $a_i$ ($s(a_i) \leq 1 - nivel(O_j)$), si es menor, propone que $a_i$ sea acomodado en $O_j$, dejando a ese objeto abierto, de lo contrario, cierra el objeto y propone que $a_i$ sea acomodado en un nuevo objeto $O_{j+1}$, el cual ahora es el último objeto abierto. Este algoritmo puede ser implementado para que corra en tiempo lineal.

#### Heurística First Fit (FF)

En este algoritmo, la restricción de acomodar las piezas solo en el último objeto abierto, no es considerada. Esta heurística considera a todos los objetos parcialmente llenos como posibles destinos para la pieza a ser acomodada. Primero se intenta acomodar la pieza en el primer objeto (el que tenga el menor índice), por ejemplo, si existe un objeto parcialmente abierto para el cual $nivel(O_j) + s(a_i) \leq 1$ se propone que se acomode la pieza $a_i$, el objeto con el menor índice que cumpla esta condición. De lo

contrario, se abre un nuevo objeto y se propone a $a_i$ como su primer pieza. La heurística First Fit, parece tomar ventaja de el rango más amplio de objetos que selecciona.

### Heurística Best Fit (BF)

La pieza $a_i$ es acomodada en el objeto parcialmente lleno $O_j$ con el mayor valor $nivel(O_j) \leq 1 - s(a_i)$, en caso de empate, opta por el objeto de menor índice.

### Heurística Worst Fit (WF)

La pieza $a_i$ es acomodada en el objeto parcialmente lleno $O_j$ con el menor valor $nivel(O_j) \leq 1 - s(a_i)$, en caso de empate, opta por el objeto de menor índice.

### Heurística Almost Worst Fit (AWF)

La pieza $a_i$ es acomodada en el objeto parcialmente lleno $O_j$ con el segundo menor valor $nivel(O_j) \leq 1 - s(a_i)$. Si $a_i$ no cabe en ningún objeto, abre un nuevo objeto.

### Heurística First Fit Decreasing (FFD) [22]

Este algoritmo de aproximación utiliza una heurística muy simple, esta tiene una complejidad en tiempo de $\Theta(n^2)$, y produce buenas soluciones. La estrategia del $FirstFit$ simple coloca la pieza en el primer objeto que encuentra. La estrategia algoritmo first fit decreasing (FFD) es una modificación que ordena las piezas de manera decreciente antes de acomodarlas.

La lista de piezas puede ser ordenada en tiempo $\Theta(n \log n)$. Para la búsqueda de un nuevo objeto para una pieza se lleva a lo más $n(n-1)/2$ veces, totalizadas sobre todas las piezas. Todas las demás instrucciones son ejecutadas a lo más $n$ veces, de tal manera que la complejidad en tiempo en el peor de los casos es $\Theta(n^2)$. La heurística FFD, no siempre encuentra el número óptimo de objetos.

### Heurística Next Fit Decreasing (NFD)

En este algoritmo, se ordenan las piezas de manera decreciente antes de ser acomodadas, después, una pieza es posicionada en el último objeto abierto si es posible, de lo contrario, un nuevo objeto es abierto y entonces la pieza se coloca en este nuevo objeto.

### Heurística Djang and Finch's (DJD) [7]

Esta heurística se enfoca a llenar contenedor por contenedor de la siguiente manera: Las piezas son ordenadas de mayor a menor y son colocadas en el contenedor hasta que este ha sido llenado por lo menos a 1/3 de su capacidad. Posteriormente

inicializa un desperdicio $w$ permitido, inicialmente cero ($w = 0$). La heurística busca por un objeto que llene al contenedor con un desperdicio $w$, si no lo encuentra, busca por dos objetos que llenen al contenedor teniendo un desperdicio $w$. Si falla, busca por la combinación de tres objetos que llenen al contenedor teniendo un desperdicio $w$. Si esto también falla, incrementa la variable $w = w + 1$ y repite.

#### Heurística Djang and Finch's more Tuples (DJT)

Una modificación de la heurística DJD que considera combinaciones de hasta cinco piezas en lugar de buscar la combinación con tres piezas.

### 2.3.2. Heurísticas para acomodo

Las siguientes heurísticas pertenecen a la clase de abajo-izquierda. En estas técnicas el algoritmo de estabilidad no permitirá que una pieza posicionada en la parte inferior-izquierda se mueva [17].

#### Algoritmo BL

Comenzando por la parte superior derecha, cada pieza se desliza hacia abajo tan lejos como sea posible y luego hacia la izquierda tan lejos como sea posible del objeto. Las operaciones sucesivas de los movimientos horizontales y verticales se repiten hasta que la pieza llega a una posición estable. El pseudo-código es el siguiente.

```
Algoritmo_BL(S, O)
        {
        Mientras haya piezas que acomodar:
           Obten la siguiente pieza
           Acomodarlo en la esquina superior derecha del objeto
                Mientras las distancias vertical Y horizontal = 0:
                {
                Calcula la distancia vertical al acomodo parcial
                Mueve la pieza la distancia vertical
                Calcula la distancia horizontal al acomodo parcial
                Mueve el rectángulo la distancia horizontal
                }
        }
```

Una posición válida es encontrada cuando la pieza choca con el acomodamiento parcial en sus partes inferior e izquierda. La mayor desventaja de esta rutina consiste

en la creación de espacios vacíos, por otro lado su complejidad de tiempo es $O(n^2)$, en donde $n$ es el número piezas a ser cortadas.

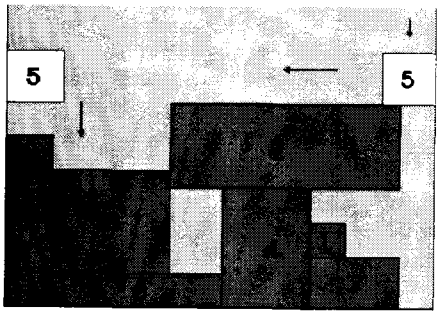



Figura 2.1: Bottom-Left Placement Rule (BL).

### Algoritmo BLLT

Así como el algoritmo BL, este algoritmo comienza colocando a la pieza en la parte superior derecha, después es deslizada hacia abajo tan lejos como sea posible. En lugar de mover la pieza hacia la izquierda tan lejos como sea posible, mueve la pieza hacia la izquierda siempre y cuando se pueda deslizar a lo largo de la parte superior de las piezas ubicados en la parte inferior. Tan pronto como encuentre una esquina, es movida otra vez verticalmente hacia abajo. En el caso de haber una colisión con otras piezas en el acomodo, las operaciones de movimiento son detenidas y la pieza queda acomodada en esa posición. Una posición estable es alcanzada cuando la pieza no puede ser movida hacia la izquierda o hacia abajo sin que haya una colisión. A continuación se presenta el pseudo-código:

```
Algoritmo_BLLT(S, O)
        {
        Mientras haya piezas que acomodar:
           Obten la siguiente pieza
           Acomodarla en la esquina superior derecha del objeto
                Mientras las distancias vertical Y horizontal = 0:
                {
                Calcula la distancia vertical al acomodo parcial
                        Mueve la pieza la distancia vertical
                Calcula la distancia horizontal al acomodo parcial
                Calcula la longitud horizontal del camino a lo largo
                de las piezas en la parte inferior
                        Mueve la pieza la distancia mínima de las
                        dos distancias horizontales calculadas
                }
        }
```

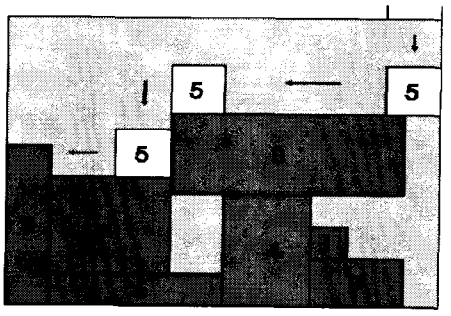



Figura 2.2: Improved Bottom-Left Placement Rule (BLLT).

### Algoritmo BLF

Puesto que los dos algoritmos anteriores tienden a generar espacios de áreas grandes, se utiliza este algoritmo. La estrategia consiste en colocar una pieza en la posición más baja disponible, acomodándolo en la parte izquierda de la posición disponible. Sin embargo la mayor desventaja recae en su complejidad de tiempo la cual es $O(n^3)$.

```
Algoritmo_BLF(S, O)
 {
 Mientras haya piezas que acomodar:
    Obten la siguiente pieza
        Mientras no se hayan intentado todas las posiciones y se haya hecho
        una inserción
           Obten la siguiente posición de la lista
           Revisa si el área asignada a la posición es lo suficientemente grande
           Si el área asignada es lo suficientemente grande
                Realiza una prueba de inserción: con todas las piezas
                        que pueden intersectar en esa posición.
                        Para, cuando la primera inserción es detectada
                Si se realizo inserción
                        Actualiza el tamaño del área
                Si no se realizó la inserción
                        Inserta la pieza en esta posición
                        Itera la pieza hasta una posición estable
                            abajo-izquierda
                        Obten los puntos superior-izquierdo e inferior-derecho
                            de esta pieza.
                        Inicializa el área asignada con anchura igual a la
                            distancia hacia la frontera derecha y altura igual a
                            la distancia a la frontera superior.
                        Actualiza la lista de posiciones, insertando los puntos
```

```
                superior-izquierdo e inferior-derecho de esta pieza
            Borra a los puntos innecesarios de la lista.
            Ordena la lista de posiciones en tamaño de
                abajo-izquierda
    }
}
```

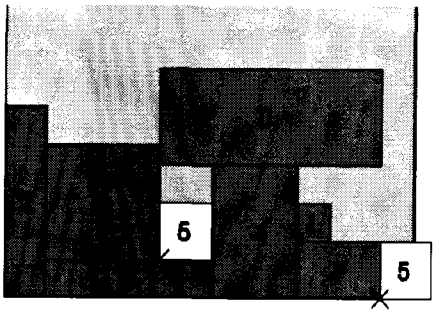



Figura 2.3: Bottom-Left Fill Placement Rule (BLF).

## 2.4. Algoritmo genético

Carlos Darwin en su libro "El origen de las especies" [10] menciona que la supervivencia de acuerdo a la aptitud lleva a mejores miembros de una especie. El proceso evolutivo que propone puede pensarse como una solución a un problema de optimización, en donde el problema a ser optimizado es la aptitud de los individuos de una población en evolución. En la naturaleza, el problema a ser optimizado puede verse como el desarrollo de cualidades que incrementan las probabilidades de sobrevivir y reproducirse (pasando material genético a futuras generaciones). Estas cualidades son difíciles de predecir y por lo tanto varían según las especies e incluso pueden depender del estado actual de la evolución de otra especie.

Se podrían definir poblaciones virtuales de individuos representando soluciones parciales a un problema, en donde su aptitud corresponde a qué tan bien es resuelto el problema y evolucionar esa población para obtener individuos más capaces para resolver el problema. Al programa computacional que implementa este proceso se le llama algoritmo genético (AG). El algoritmo genético, propuesto inicialmente por John Holland [16], es un método de optimización ciega, pues no utiliza más información que aquella proporcionada por la evaluación de la función objetivo. Es un método estocástico de búsqueda (diseño, optimización) basado en la mecánica de la selección natural y la idea darwiniana de la supervivencia de acuerdo a la aptitud. Como en los algoritmos genéticos naturales, se trata de encontrar nuevas y mejores soluciones al problema, mejorando a las soluciones actuales, esto se logra extrayendo las aptitudes

más deseables de una generación de soluciones y combinandolas para formar una nueva generación.

En un algoritmo genético, las posibles soluciones a un problema son codificadas en forma de tiras de caracteres de un alfabeto reducido (generalmente binario) que se asemejan a los cromosomas de los seres vivos.

El algoritmo genético evoluciona una población de estos individuos aplicando operadores genéticos como selección, cruce y mutación. La calidad de la solución esta medida por una función de evaluación que mide la aptitud. De a acuerdo a la aptitud de los individuos de la población, los individuos son seleccionados para reproducción. Dado que los individuos contribuyen a la creación de la siguiente generación de acuerdo con su aptitud, se espera que las siguientes generaciones evolucionen hasta converger a una solución óptima o cercana a la óptima.

La terminología de los algoritmos genéticos viene de la biología, dado que es un modelo biológico el que se esta siendo emulado. Los pasos que sigue el algoritmo genético son los siguientes:

1. Generar población inicial.
2. Selección de los mejores individuos de acuerdo a su aptitud.
3. Cruce de parejas permitiendo la recombinación de material genético.
4. Mutación de algunos individuos.
5. Se generan nuevas poblaciones y regresa al paso dos, hasta que se cumple un criterio de terminación.

### 2.4.1. Población

Una población generalmente es representada por un grupo de individuos cuyas propiedades están especificadas por sus genomas, los cuales están hechos de uno o más cromosomas. El número de genes en un cromosoma puede depender de acuerdo al problema que se este tratando de solucionar por el AG. Un simple cromosoma puede estar hecho de un conjunto de bits, cada uno especificando verdadero o falso para algún atributo, sin embargo, generalmente atributos de dos estados no son suficientes, por otro lado, el cromosoma puede estar formado por genes, y cada gen puede tomar un valor entre 0 y $n-1$. El número de individuos en una población normalmente es controlado. En general, entre mayor sea la población, mayor el número de soluciones representadas y habrá mayor oportunidad de encontrar la solución óptima. Sin embargo hay una contradicción entre el tamaño de la población y el tiempo computacional. La población inicial normalmente es generada aleatoriamente, pero se puede generar con individuos con ciertas características.

### 2.4.2. Aptitud

En la naturaleza, la aptitud de un individuo esta representada por su habilidad de pasar su material genético, esto involucra su capacidad de sobrevivir en donde existe competencia por la supervivencia, pues tiene que sobrevivir si desea reproducirse.

### 2.4.3. Selección

Existen varios métodos de selección, pero la mayoría se basa en asignar probabilidades de selección a cada uno de los miembros de la población de acuerdo a su aptitud. Algunos métodos de selección son: selección por rueda de ruleta, selección por ranqueo y selección por torneo.

#### Selección por rueda de ruleta

En este método, los padres son seleccionados de la población probabilísticamente de manera proporcional a su aptitud. Una manera de ver esto es como una gráfica de pastel, en donde cada miembro de la población posee una pieza de pastel cuyo tamaño esta directamente relacionado con qué tan apto es ese individuo. Cuando un padre es seleccionado de la población, la rueda (el pastel), es girado, y en donde se detenga, se selecciona ese individuo.

#### Selección por ranqueo

La selección por ranqueo, selecciona a los individuos, de mejor a peor de acuerdo a su aptitud. Durante la selección, un individuo con ranqueo $i$ es seleccionado con probabilidad $p^i$ en donde $p$ es un número entre 0 y 1. El algoritmo itera por toda la población (ordenada por su aptitud). El primer individuo en el ranqueo es seleccionado con probabilidad $p$. Si no es seleccionado, la iteración continúa hacia el siguiente individuo que también es seleccionado con probabilidad $p$. Este proceso continúa hasta que un individuo es seleccionado. Si la iteración termina sin que ningún individuo haya sido seleccionado, comienza de nuevo desde el principio.

#### Selección por torneo

En la selección por torneo, un grupo de individuos es seleccionado de la población de manera aleatoria, posteriormente se selecciona al individuo más apto de ese grupo. El tamaño del grupo seleccionado puede variar.

### 2.4.4. Cruce

Cruce es la recombinación de material genético entre los individuos seleccionados. Es una simulación del proceso reproductivo sexual que provee intercambio genético. Se ha desarrollado un gran número de operadores de cruce para varios propósitos. El cruce puede ser aplicado después o durante la selección, con cierta probabilidad definida inicialmente.

#### Cruce de un punto

Uno de los primeros y más simples es el cruce de un punto, en donde el punto de cruce es seleccionado de manera aleatoria, y los dos cromosomas padres intercambian información a partir de ese punto, como se muestra en la tabla 2.2

Tabla 2.2: Demostración de un cruce de un punto en cromosomas arbitrarios de longitud 6 con punto de cruce 2.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Padre 1 | : | A | B | C | D | E | F |
| Padre 2 | : | P | Q | R | S | T | E |
| Hijo 1 | : | A | B | R | S | T | E |
| Hijo 2 | : | P | Q | C | D | E | F |

#### Cruce de dos puntos

Dos puntos de cruce son seleccionados de manera aleatoria en el cromosoma e intercambia la información genética de los cromosomas de los padres entre esos puntos. Ejemplo de este operador de cruce se muestra en la tabla 2.3.

Tabla 2.3: Demostración de un cruce de dos puntos en cromosomas arbitrarios de longitud 6.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Padre 1 | : | A | B | C | D | E | | F |
| Padre 2 | : | P | Q | R | S | T | | O |
| Hijo 1 | : | A | B | R | S | T | | F |
| Hijo 2 | : | P | Q | C | D | E | | O |

#### Cruce uniforme

Un operador de cruce decide (con alguna probabilidad), que padre contribuirá con cada uno de los valores de los genes en los cromosomas hijos. Esto permite a los cromosomas padres ser mezclados a nivel de genes en lugar de ser mezclados por segmentos.

En la tabla 2.4 se muestra un ejemplo de este operador de cruce, en donde la probabilidad del operador de cruce es 0.5, se puede apreciar que aproximadamente la mitad de los genes en los hijos provienen del padre 1 y la otra mitad provienen del padre 2.

Tabla 2.4: Demostración de un cruce uniforme en cromosomas arbitrarios de longitud 8.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Padre 1 | : | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Padre 2 | : | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| Hijo 1 | : | $1_1$ | $0_2$ | $1_2$ | $0_1$ | $0_2$ | $0_1$ | $1_1$ | $1_2$ |
| Hijo 2 | : | $0_2$ | $1_1$ | $0_1$ | $0_2$ | $1_1$ | $1_2$ | $1_2$ | $0_1$ |

**Cruce aritmético**

El cruce aritmético es un operador de cruce que linealmente combina dos vectores de los cromosomas padres para producir dos nuevos hijos de acuerdo a las siguientes ecuaciones.
Hijo 1 = $a*$ Padre 1 $+(1-a)*$ Padre 2
Hijo 2 = $(1-a)*$ Padre 1 $+a*$ Padre 2
en donde $a$ es una variable aleatoria (seleccionada antes de cada operación de cruce). Para ilustrar este operador de cruce, considere el ejemplo mostrado en la tabla 2.5, en donde $a$= 0.7.

Tabla 2.5: Demostración de un cruce aritmético en cromosomas arbitrarios.

| | | | | | |
|---|---|---|---|---|---|
| Padre 1 | : | (0.3) | (1.4) | (0.2) | (7.4) |
| Padre 2 | : | (0.5) | (4.5) | (0.1) | (5.6) |
| Hijo 1 | : | (0.36) | (2.33) | (0.17) | (6.86) |
| Hijo 2 | : | (0.402) | (2.981) | (0.149) | (6.842) |

### 2.4.5. Mutación

Con la mutación se intenta simular una mutación aleatoria en la naturaleza. Esta se utiliza en los algoritmos genéticos con el fin de promover diversidad de individuos en el espacio de búsqueda y prevenir que haya convergencia prematura. En un cromosoma seleccionado para mutación, se selecciona uno de sus genes de manera aleatoria. En la tabla 2.6 se muestra un ejemplo de mutación.

Tabla 2.6: Demostración de mutación en la localidad 1 en un cromosoma arbitrario de longitud 8.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Cromosoma | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| Mutación | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

### 2.4.6. Generación de una nueva población

Existe una distinción entre algoritmos de estado estable y algoritmos generacionales. Estos dos métodos representan los extremos de remplazar una parte de la población y remplazar a todos los miembros en cada generación de la evolución. A esto se le conoce como elitismo, en donde se mantiene a un cierto número de los mejores individuos de cada generación y los copia a la siguiente generación. Bajo estas definiciones, un rango de elitismo alto simularía un AG de estado estable y un elitismo bajo simularía en AG generacional.

### 2.4.7. Representación

La representación de la solución del problema en un algoritmo genético es de fundamental importancia. Primero que nada, los parámetros de la función objetivo que afectan al problema de optimización deben ser identificados y codificados. Para esta representación normalmente se utiliza código binario, en donde los parámetros son representados por cadenas de ceros y unos.

Un algoritmo genético procesa implícitamente patrones de bits llamados esquemas [25]. A continuación se muestran algunos ejemplos de esquemas.

$$\#010\# = \{0\mathbf{0100}, 0\mathbf{0101}, 1\mathbf{0100}, 1\mathbf{0101}\}$$
$$00\#\#\#=\{\mathbf{00}000, \mathbf{00}001, ...., \mathbf{00}111\ \}$$

La variable # o "don't care", puede tomar el valor de cero o uno. El funcionamiento de un algoritmo genético se puede entender a través del análisis de esquemas. Se analiza en comportamiento del número esperado de representantes de los esquemas en la población al pasar las generaciones [14].

Un AG asigna un número exponencialmente creciente de representantes a los esquemas que tienen alta evaluación, es decir, aquellos que tienen características que les dan ventaja sobre los demás.

Una forma de representar el desempeño de un algoritmo, es mediante una gráfica de mejor encontrado vs. número de evaluaciones de la función objetivo, a esta gráfica se le llama curva de mejor encontrado (ver figura 2.4). Si dos algoritmos tienen una curva igual en el promedio, entonces el algoritmo con menor desviación estándar es más deseable porque su resultado es más repetible [25].

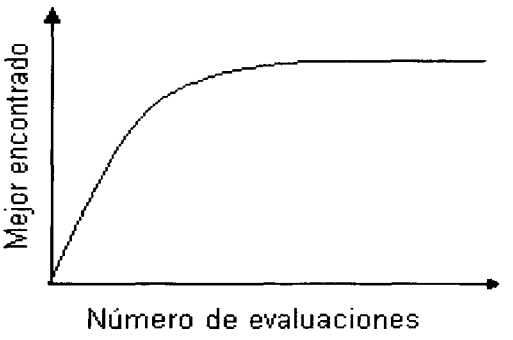



Figura 2.4: Curva de mejor encontrado típica para un algoritmo de optimización ciega que realiza maximización.

## 2.5. Sistemas de Clasificadores

Un sistema de clasificadores es un sistema de aprendizaje de máquinas que aprende sintácticamente conjuntos de reglas simples (llamadas clasificadores) para guiar el desempeño en un ambiente arbitrario [14]. Un sistema de clasificadores consiste en tres componentes principales:

1. Sistema de regla y mensaje
2. Distribución de sistemas de crédito
3. Algoritmo genético

El sistema de regla y mensaje de un sistema de clasificadores es un sistema especial de producción. Un sistema de producción es un esquema computacional que utiliza reglas como su único dispositivo algorítmico. Aunque hay una gran variación en sintaxis entre sistemas de producción, las reglas son generalmente de la siguiente forma:

**si** condición **entonces** acción.

El significado de la regla de producción es que la acción puede ser tomada (la regla es "disparada") cuando la condición es satisfecha.

En primera instancia, la restricción a un simple dispositivo para la representación de conocimiento puede parecer demasiado restringida. Se ha demostrado que sistemas de producción son computacionalmente completos. Su poder en representación del conocimiento involucra más que esto. También son computacionalmente convenientes. Una sola regla o un conjunto pequeño de reglas pueden representar un conjunto complejo de pensamientos de manera compacta.

A pesar del creciente uso de sistemas expertos en aplicaciones, los sistemas tradicionales basados en reglas han sido menos sugeridos en situaciones con necesidad de

aprendizaje. Uno de los principales obstáculos de aprendizaje ha sido la sintaxis compleja de reglas. Los sistemas de clasificadores parten de la idea de restringir una regla a una representación de longitud fija. Esta restricción tiene dos beneficios. Primero, todas las cadenas bajo el alfabeto permitido son sintácticamente significantes. Segundo, una representación de cadena fija permite operadores de cadenas de tipo genético. Esto deja la puerta abierta, para una búsqueda con algoritmos genéticos del espacio del reglas permisibles.

Los sistemas de clasificadores utilizan activación de reglas paralelas, mientras que los sistemas expertos tradicionales utilizan activación de reglas seriales. Durante cada ciclo comparativo, un sistema experto tradicional activa una sola regla. Este procedimiento de regla por regla es un cuello de botella para la producción creciente, y la diferencia entre arquitecturas de sistemas expertos competidores concierne con la selección de las mejores estrategias de activación de reglas para cualquier tipo de problema. Los sistemas de clasificadores solucionan este cuello de botella permitiendo activación paralela de reglas durante un ciclo comparativo dado. Al hacer ésto, los sistemas de clasificadores permiten múltiples actividades para ser coordinados simultáneamente. Cuando se deban hacer opciones entre acciones de ambientes específicos o cuando el tamaño del conjunto de reglas comparativas deba ser cortado para acomodar la lista del mensaje de longitud fija, estas opciones son pospuestas al último momento posible, y el arbitraje es realizado competitivamente.

Se debe remarcar el paralelismo impulsado por la arquitectura del sistema de clasificadores y reconocer que este paralelismo puede permitir implementaciones de sistemas de clasificadores en hardware extremadamente rápidas.

En sistemas expertos tradicionales, el valor o grado de una regla relativo a otras reglas es fijado por el programador en conjunto con el experto o el grupo de expertos siendo emulados. En un sistema basado en reglas, no nos podemos dar este lujo, el valor relativo de diferentes reglas es una de las piezas clave de la información que debe de ser aprendida. Para facilitar este tipo de aprendizaje, los sistemas de clasificadores forzan a los clasificadores a coexistir en una economía basada en servicios de información. Una competencia es mantenida entre clasificadores en donde el derecho de responder mensajes relevantes lo tiene el mayor licitador, con el pago subsiguiente de las ofertas sirviendo como fuente del ingreso a remitentes de mensajes previamente exitosos.

Los clasificadores pueden ser reproducidos, cruzados y mutados. De modo que no sólo puede aprender el sistema, sino que también puede descubrir reglas nuevas y posiblemente mejores a partir de combinaciones de las reglas viejas.

## 2.6. Hiper-heurística

Hiper-heurística es una técnica motivada por el objetivo de incrementar el nivel de generalidad en el cual puede operar un sistema de optimización. El término ha sido definido para describir el proceso de usar ciertas heurísticas para seleccionar a otras heurísticas para resolver el problema actual. La mayoría de las investigaciones en el área de heurísticas se enfocan al uso de estas para operar directamente en un problema. Por ejemplo, la mayoría de las investigaciones en Computación Evolutiva en calendarización de tareas, consideran poblaciones de horarios y la idea básica de que la población evolucionará. Sin embrago, una aproximación evolutiva al problema de calendarización de tareas [24], se referirá a una población de heurísticas para el problema y estas evolucionarán con el tiempo.

Una de las principales motivaciones de utilizar hiper-heurísticas, es que son más sencillas de utilizar que los métodos específicos para propósitos especiales y el objetivo es producir soluciones de buena calidad en un entorno más general. Además se trata de desarrollar métodos de aprendizaje que inteligentemente seleccionen heurísticas de acuerdo con la situación actual del problema.

### 2.6.1. El concepto y sus orígenes

Para muchos problemas reales, la búsqueda exhaustiva de soluciones no es una propuesta práctica. El espacio de búsqueda puede ser demasiado grande, incluso, en ocasiones puede que no exista una forma conveniente de definir el espacio de búsqueda. Por ejemplo, puede haber restricciones complejas que den al espacio de soluciones factibles una forma muy complicada. Por lo tanto es común que se utilizen aproximaciónes con algoritmos heurísticos, sacrificando la garantía de encontrar la solución óptima por un aumento en la velocidad y probablemente también con la garantía de obtener un buen nivel en la calidad de la solución.

El término heurística en algunas ocaciones es utilizado para referenciar a todo un algorítmo de búsqueda y en otras ocasiones es utilizado para referirse a un proceso de decisión particular.

### 2.6.2. Concepto de hiper-heurística

Dado que diferentes heurísticas simples tienen diferentes fortalezas y debilidades, tiene sentido combinarlas de alguna manera para cada una cubra las debilidades de las otras [7]. Una forma simple de hacer esto es de la siguiente manera:

```
1  if (SituaciónProblema(P) == s_1)
2      Aplica(heurística_1, P);
```

3 else if (SituaciónProblema(P) == $s_2$)
4 Aplica(heurística$_2$, P);
5 else...

en donde $s_i$ con $i = (1, ..., n)$ es una situación definida del problema y $P$ es el problema actual. La función Aplica(heurística$_i$, $P$) utilizará la heurística$_i$ para solucionar la situación $s_i$ del problema $P$.

Una lógica extrema de esta aproximación sería un algoritmo conteniendo un enunciado infinito de switch enumerando todos los posibles problemas y aplicando la mejor heurística conocida para cada uno. La idea clave de las hiper-heurísticas es utilizar miembros de un conjunto de heurísticas y modificar el estado del problema. La clave de esta observación es simple: La fortaleza de una heurística se basa en su habilidad de tomar decisiones en la ruta hacia la fabricación de una solución excelente. Por lo tanto, ¿porque no tratar de asociar a cada heurística con las condiciones del problema y por lo tanto aplicar diferentes heurísticas a diferentes partes o fases del proceso de solución?.

### 2.6.3. Trabajos Relacionados

Un ejemplo de utilización de hiper-heurísticas es el problema de calendarización *open − shop*. En este problema, se tienen $j$ trabajos, cada uno consistiendo de un determinado número de tareas. Una tarea consiste en visitar una cierta máquina en una longitud de tiempo específica de esa tarea. La tarea asociada con un trabajo puede realizarse en cualquier orden. Fang et al [15] utilizó un algorítmo genético que construye la solución de la siguiente manera. Un cromosoma representa una serie de pares de enteros $[t_0, h_0, h_1, ...]$ interpretado de izquierda a derecha y significando, para cada $i$, considerar el trabajo incompleto $t_i$ y utilizar la heurística $h$ para seleccionar una tarea a insertar en el horario en la posición más temprana en donde quepa. Algunas de las heurísticas utilizadas:

- Selecciona la tarea con el mayor tiempo de procesamiento
- Selecciona la tarea con el menor tiempo de procesamiento
- De las tareas que pueden comenzar lo más temprano posible, selecciona la que tenga el mayor tiempo de procesamiento.
- De todas las operaciones que pueden ser insertadas en el horario, selecciona aquella que mejor quepa (es decir, deja el menor tiempo de oseo posible).

Un ejemplo real del uso de hiper-heurística esta descrito en [12], en donde el problema era calendarizar la colección y entrega de pollos vivos de granjas ubicadas en

Escocia y el norte de Inglaterra, hacia una o dos fábricas procesadoras con el fin de satisfacer las demandas de los supermercados. Las órdenes de los clientes cambian semana tras semana y en ocaciones día por día. La tarea era calendarizar el trabajo realizado por camiones. El objetivo principal era mantener a las fábricas atendidas con trabajo sin que fuera necesario tener a pollos vivos esperando en el patio de la fábrica por razones veterinarias y legales. Esto se complicaba por muchas restricciones. Por ejemplo, había muchos tipos de camiones, diferenciados por los días que trabajaban, la hora en que comenzaban a trabajar, el mínimo de carga que podían llevar y su capacidad máxima de carga. Había restricciones en el que las granjas tenían que ser visitadas, para minimizar el riesgo potencial de difundir enfermedades de los pollos. El objetivo principal no era producir horarios óptimos en términos de costo, porque los requerimientos del trabajo podían cambiar en poco tiempo. El objetivo principal era crear buenos horarios satisfaciendo todas las restricciones. La solución eventual utilizó dos algoritmos genéticos. Uno utilizó una selección de heurística para descomponer el conjunto total de órdenes de los clientes en tareas individuales y asignar esas tareas a camiones. El otro comenzaba con estas asignaciones y evolucionaba el horario para las llegadas de los camiones a las fábricas, de estos horarios, era posible determinar cual camión llegaría a cada granja y a que hora y por lo tanto construir un horario completo para todos los participantes. En el primer algoritmo genético, el cromosoma especificaba una permutación de las órdenes de los clientes y después dos secuencias de heurísticas. La primer secuencia de opciones trabajaba por toda la permutación y se desplazaba de una a otra en las cargas utilizando heurísticas simples acerca de como particionar la orden completa del cliente en cargas convenientes, la segunda secuencia de opciones sugería como asignar estas cargas a los camiones. El resultado final cumplió con los requerimientos del proyecto.

En los ejemplos anteriores, el algoritmo genético fue utilizado para evolucionar una secuencia finita de opciones heurísticas, con una opción por cada paso en el proceso de la construcción de la solución completa. A pesar de que los resultados generalmente fueron buenos, aún es poco satisfactorio porque el método únicamente evoluciona soluciones a instancias específicas de los problemas y no puede solucionar problemas demasiado grandes. Terashima et. al. [24] describe una aproximación diferente a hiper-heurísticas que soluciona este problema mencionado. Ese documento se enfoca a la solución de problemas de asignación de exámenes universitarios a gran escala. Se tiene un número de intervalos de tiempo en los cuales puede haber un examen, y se tienen salones de distintas capacidades. Dos exámenes no pueden aplicarse al mismo tiempo, si existe un estudiante que lleve ambos cursos. Se puede aplicar más de un exámen en un salón en el mismo intervalo de tiempo, si no existe un estudiante que lleve ambos cursos y el salón tiene la capacidad suficiente. Algunos exámenes pueden estar restringidos para evitar ciertos intervalos de tiempo, por ejemplo a alguna hora en la que no esté disponible el instructor. También existen algunas restricciones *suaves* que sería bueno respetar,

pero pueden ser violadas si es necesario. Por ejemplo, si se desea que ningún estudiante tenga exámenes consecutivos en el mismo día y también es deseable que los exámenes de mayor duración sean primero, para tener más tiempo para evaluarlos antes de que termine el periodo de exámenes. Estos problemas pueden involucrar la asignación de miles de exámenes y cientos de miles de estudiantes.

El enfoque utilizado por Terashima et. al. [24] es suponer que existe un algoritmo de construcción de asignación de horarios del orden general mostrado a continuación.

```
/*Realiza la primera fase de la construcción*/

mientras(condición(x) == FALSA) {
  evento = Aplica_heurística_a_evento(H1);
  intervalo_de_tiempo = Aplica_heurística_a_intervalo(H2, evento);
  Agrega_a_horario(evento, intervalo_de_tiempo);
}

/*Realiza segunda fase de construcción*/
  mientras(no(completo())) {
  evento = Aplica_heurística_a_evento(H3);
  intervalo_de_tiempo = Aplica_heurística_a_intervalo(H4, evento);
  Agrega_a_horario(evento, intervalo_de_tiempo);
}
```

La idea es utilizar un algoritmo genético para evolucionar las opciones $H1$, $H2$, $H3$, $H4$ y la condición $X$ que determina cuando cambiar de la primera fase a la segunda. Algunas de las heurísticas y posibles condiciones involucraban parámetros adicionales como si se permitiría y que tanto *backtracking* en la toma de decisiones, o cambiarse a la fase dos después de colocar $N$ eventos. La lógica de utilizar este algoritmo es que muchos problemas de asignación de horarios, necesitan resolver un conjunto de problemas inicialmente (por ejemplo, el problema de empacado de materiales para poner a los exámenes más largos en cuartos con suficiente capacidad), pero un conjunto de problemas distinto en una etapa diferente de la construcción de la solución. Las restricciones *suaves* son resueltas mediante heurísticas, por ejemplo, con el fin de reducir las probabilidades de que un estudiante tenga dos examenes en intervalos de tiempo adyacentes, una heurística puede considerar intervalos de tiempo en un orden que de a los intervalos adyacentes una muy baja prioridad.

Un cromosoma era evaluado al construir la asignación de horarios, midiendo la calidad del resultado. De manera interesante, este método resolvió problemas muy grandes de manera satisfactoria. Los autores también construyeron una búsqueda de fuerza bruta en el espacio de los cromosomas con el fin de evaluar si el algoritmo genético estaba entregando resultados de muy buena calidad.

### 2.6.4. Marco de Trabajo de la Hiper-Heurística

Una heurística típicamente trabaja en un problema directamente, normalmente con el dominio del conocimiento incorporado en el algoritmo. Sin embargo, el marco de trabajo de la hiper-heurística opera en un nivel mayor de abstracción y por lo regular no tiene conocimiento del dominio. Únicamente tiene acceso a un conjunto de heurísticas de menor nivel que puede llamar, pero sin tener conocimiento del propósito o la función de la heurística de menor nivel dada. La motivación para esta representación es que una vez que el algoritmo de hiper-heurística ha sido desarrollado, se pueden resolver nuevos dominios del problema únicamente remplazando el conjunto de heurísticas de menor nivel y la función de evaluación, la cuál indica la calidad de la solución dada. El esquema general de una hiper-heurística se muestra en la figura 2.5.

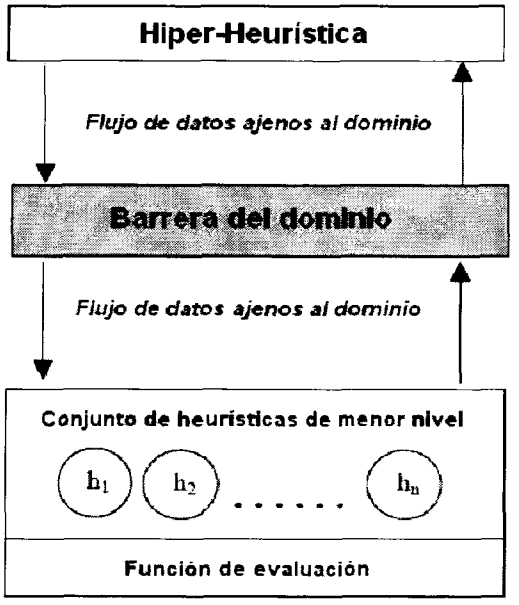



Figura 2.5: Esquema de Hiper-Heurística.

Esta figura muestra que existe una barrera entre las heurísticas de menor nivel y la hiper-heurística, el dominio del conocimiento no tiene permitido pasar esta barrera, por lo tanto la hiper-heurística no tiene el conocimiento del dominio bajo el cual esta operando. Únicamente sabe que tiene $n$ heurísticas de menor nivel a las cuales puede llamar y sabe que se le pasarán los resultados de una solución dada una vez que haya sido evaluada por la función de evaluación.

Existe una interfaz bien definida entre la hiper-heurística y las heurísticas de menor nivel. Las razones de esto son dos.

1. Permite a la hiper-heurística comunicarse con las heurísticas de menor nivel uti-

lizando una interfaz estándar, de lo contrario la hiperheurística necesitaría una interfaz separada para cada heurística de menor nivel lo cuál obviamente no tiene sentido. Además, facilita el intercambio de información entre las heurísticas de menor nivel y la hiper-heurística. Además se ha incluido un componente que le indique a la heurística de menor nivel cuanto tiempo tiene que ejecutarse. La motivación para esta idea es que se llame a cada heurística en turno dándole un tiempo determinado y la heurística que mejor se desempeñe, en el tiempo permitido, es la que se utiliza a la solución del problema. En conjunto con este componente, la interfaz también define si la heurística de menor nivel debe aplicar sus cambios a la solución actual o si solo debe reportar qué efecto tendría si aplicara dichos cambios. La idea es que la hiper-heurística pueda preguntar a cada heurística de menor nivel que también puede comportarse contra una solución dada. Entonces la hiper-heurística puede decidir que heurísticas deben ser utilizadas para actualizar la solución.

2. Permite un rápido desarrollo para otros dominios. Cuando se implementa en un nuevo problema , el usuario debe proporcionar el conjunto de heurísticas de menor nivel y una función de evaluación. Si las heurísticas de menor nivel siguen una interfaz estándar, la hiper-heurística no tiene que ser alterada de ninguna manera y será capaz de resolver el nuevo problema.

Un ejemplo que sigue este marco de trabajo es descrito por Ross [22]. El enfoque aquí esta en los procesos de aprendizaje de soluciones aplicables a muchas instancias de problemas en lugar de aprender soluciones individuales. Este proceso deberá ser capaz de seleccionar una de varias heurísticas simples en los diferentes estados del problema, gradualmente transformando el problema de un estado inicial a un estado resuelto. La primer aplicación de dicho modelo fue para resolver el problema de empacado en una dimensión. En este trabajo, se utilizó un sistema de clasificadores basado en la precisión de la predicción (XCS), para aprender un conjunto de reglas que asocian características de los estados del problema con ocho diferentes heurísticas. El conjunto de reglas es utilizado de la siguiente manera: dadas las características iniciales de un problema $P$, una heurística $H$ es seleccionada para empacar un contenedor, gradualmente alterando las características del problema que aún queda por resolver. En cada paso, una regla apropiada al estado actual del problema $P'$ es seleccionada y el proceso se repite hasta que todos los objetos han sido empacados. Este enfoque es probado utilizando 890 problemas de la literatura, de los cuales 667 fueron utilizados para entrenar al $XCS$ y 223 para pruebas. El método ($HH$) logró los resultados óptimos en 78.1. % de los problemas de entrenamiento y 74.6 % en el resto de las pruebas [7].

## 2.7. Resumen

En el presente capítulo se estudiaron las principales características del problema de corte de materiales, así como algunos métodos utilizados para resolverlo. Se explicó el funcionamiento de diversos algoritmos heurísticos utilizados para resolver este problema, tanto para la selección de piezas como para el acomodo de estas dentro de los objetos. Se presentó la teoría básica de los algoritmos genéticos y de los sistemas de clasificadores. También se mensionaron algunos trabajos relacionados con hiper-heurística y se explicó la relación que hay entre las heurísticas simples con una hiper-heurística.

## Capítulo 6

# Conclusiones

En este capítulo se mencionan las aportaciones y se presentan conclusiones de los experimentos realizados a lo largo de esta investigación. Adicionalmente se sugieren algunas ampliaciones que pudieran hacerse a partir de este trabajo.

## 6.1. Contribuciones y conclusiones

En el presente documento se describieron y reportaron resultados experimentales con un sistema de clasificadores, el XCS, en el cual la aptitud esta basada en la precisión del la predicción del clasificador, no en la predicción por si misma, y el algoritmo genético es ejecutado en el conjunto de acciones, en lugar de ser sobre toda la población. Los resultados muestran que el XCS es capaz de formar reglas que representan y solucionan de manera eficiente a todas las instancias del problema completo. Los experimentos y análisis llevados acabo durante el desarrollo de esta tesis nos muestran una serie de contribuciones que pueden ser resumidas en los siguientes puntos:

- Se utilizó exitosamente un algoritmo útil para resolver distintos problemas de corte de material en dos dimensiones.
- Se definió la representación del ambiente y su relación con heurísticas de selección y acomodo de piezas, en un sistema de clasificadores basado en la precisión de la predicción (XCS).
- Se programó un conjunto de heurísticas de acomodo y selección enfocadas al problema de corte de material en dos dimensiones.

Analizando los resultados de las mejores soluciones encontradas para los problemas, se puede apreciar que el sistema de clasificadores, identifica de manera eficiente la mejor combinación de heurísticas que solucionan a los distintos problemas de corte de material.

En la aproximación en donde el pago por la calidad de la solución es recibido cada vez que cambia el estado del problema (y que se propone una nueva heurística), el

algoritmo genético en el sistema de clasificadores opta por quedarse con los individuos de la población que contienen a las mejores heurísticas. Por otro lado, en la aproximación en donde el pago por la calidad de la solucion es recibido una vez que se ha resuelto todo el problema, el algoritmo genético encuentra la secuencia correcta de heurísticas que obtiene el mejor pago y propone el mejor resultado.

Con los resultados presentados se comprueba que si las heurísticas propuestas son individualmente aceptables, la combinación de estas producirá un resultado de mejor calidad que el obtenido por cualquiera de ellas por separado.

## 6.2. Trabajo futuro

Finalmente se sugieren algunas ideas que pudieran extender la investigación presentada con el fin de mejorar el desempeño del algoritmo propuesto, en problemas específicos. Algunos puntos en los que se podría trabajar son los siguientes:

- En el presente trabajo se intenta resolver un problema muy específico de corte de materiales, actualmente, las coordenadas de las piezas se representan con dos puntos en el espacio de dos dimensiones. Se podría adaptar de tal manera que acepte patrones de corte triangulares, para ésto, se necesitarían tres puntos en el espacio, con lo cual se tendrían que adaptar los algoritmos de acomodo de piezas. También se podría extender a corte de materiales circulares o posiblemente corte de figuras irregulares, que pudieran adaptarse a los requerimientos industriales.

- Se podría trasladar el problema a tres o más dimensiones, para cubrir aplicaciones más complejas que involucran un mayor número de restricciones, por ejemplo corte de materiales en tres dimensiones mediante robots manipuladores.

- La presente investigación se podría tomar como base para solucionar problemas de empacado, que pudieran optimizar el acomodo de material en bodegas o contenedores, aprobechando al máximo el espacio disponible.

## Apéndice A

# Problemas de la literatura para corte de materiales en dos dimensiones

Existen en la literatura diversos problemas de corte de materiales en dos dimensiones, algunos de estos son

1. Problemas de corte en dos dimensiones guillotinables. Actualmente existen 3 problemas en el repositorio de J.E.Beasley [2] Los tres problemas estan disponibles en los archivos cgcut1, ..., cgcut3.

2. Problemas de corte en dos dimensiones sin restricciones. Actualmente existen 13 problemas de este tipo en el repositorio de J.E.Beasley [2] Estos problemas estan disponibles en archivos bajo los nombres gcut1, ..., gcut13.

3. Problemas de corte en dos dimensiones con restricciones no guillotinables. Este repositorio de datos contiene los 12 problemas de J.E.Beasley [3] que se pueden encontrar en archivos bajo los nombres ngcut1, ..., ngcut12.

   La información proporcionada en estos archivos es la siguiente:

- Número de piezas a ser cortadas (m)
- Longitud y ancho de los objetos rectangulares
- Para cada una de las piezas i ($i = 1, ....., m$), ancho y alto

Una lista completa de diversas áreas para problemas de optimización se pueden encontrar en OR-Library [4]

Tabla A.1: Instancias de problemas de corte en dos dimensiones guillotinables con restricciones.

| | *cgcut1* | | *cgcut2* | | *cgcut3* | |
|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | |
| | ancho 15 | alto 10 | ancho 40 | alto 70 | ancho 40 | alto 70 |
| | Número de piezas 16 | | Número de piezas 23 | | Número de piezas 62 | |
| no. | ancho | alto | ancho | alto | ancho | alto |
| 1 | 8 | 4 | 21 | 22 | 31 | 43 |
| 2 | 8 | 4 | 31 | 13 | 31 | 43 |
| 3 | 3 | 7 | 9 | 35 | 31 | 43 |
| 4 | 8 | 2 | 9 | 35 | 31 | 43 |
| 5 | 8 | 2 | 9 | 35 | 30 | 41 |
| 6 | 8 | 2 | 9 | 24 | 30 | 41 |
| 7 | 3 | 4 | 9 | 24 | 29 | 39 |
| 8 | 3 | 4 | 9 | 24 | 29 | 39 |
| 9 | 3 | 4 | 30 | 7 | 29 | 39 |
| 10 | 3 | 4 | 30 | 7 | 29 | 39 |
| 11 | 3 | 4 | 11 | 13 | 28 | 38 |
| 12 | 3 | 3 | 11 | 13 | 28 | 38 |
| 13 | 3 | 3 | 11 | 13 | 28 | 38 |
| 14 | 3 | 2 | 10 | 14 | 28 | 38 |
| 15 | 3 | 2 | 14 | 8 | 27 | 37 |
| 16 | 2 | 1 | 14 | 8 | 27 | 37 |
| 17 | | | 14 | 8 | 27 | 37 |
| 18 | | | 12 | 8 | 26 | 36 |
| 19 | | | 12 | 8 | 26 | 36 |
| 20 | | | 12 | 8 | 26 | 36 |
| 21 | | | 13 | 7 | 26 | 36 |
| 22 | | | 13 | 7 | 25 | 35 |
| 23 | | | 13 | 7 | 25 | 35 |
| 24 | | | | | 25 | 35 |
| 25 | | | | | 24 | 34 |
| 26 | | | | | 24 | 34 |
| 27 | | | | | 24 | 34 |
| 28 | | | | | 24 | 34 |
| 29 | | | | | 33 | 23 |
| 30 | | | | | 33 | 23 |
| 31 | | | | | 33 | 23 |
| 32 | | | | | 33 | 23 |
| 33 | | | | | 22 | 32 |
| 34 | | | | | 22 | 32 |
| 35 | | | | | 22 | 32 |
| 36 | | | | | 31 | 21 |
| 37 | | | | | 31 | 21 |
| 38 | | | | | 31 | 21 |
| 39 | | | | | 29 | 18 |
| 40 | | | | | 29 | 18 |
| 41 | | | | | 29 | 18 |
| 42 | | | | | 17 | 27 |
| 43 | | | | | 17 | 27 |
| 44 | | | | | 15 | 24 |
| 45 | | | | | 15 | 24 |
| 46 | | | | | 16 | 25 |
| 47 | | | | | 16 | 25 |
| 48 | | | | | 16 | 25 |
| 49 | | | | | 16 | 25 |
| 50 | | | | | 15 | 24 |
| 51 | | | | | 23 | 14 |
| 52 | | | | | 23 | 14 |
| 53 | | | | | 23 | 14 |
| 54 | | | | | 23 | 14 |
| 55 | | | | | 21 | 12 |
| 56 | | | | | 21 | 12 |
| 57 | | | | | 21 | 12 |
| 58 | | | | | 19 | 11 |
| 59 | | | | | 19 | 11 |
| 60 | | | | | 19 | 11 |
| 61 | | | | | 19 | 11 |
| 62 | | | | | 9 | 17 |

Tabla A.2: Instancias de problemas de corte en dos dimensiones guillotinables sin restricciones

| | *gcut*1 | | *gcut*2 | | *gcut*3 | | *gcut*4 | |
|---|---|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | | Objetos | |
| | ancho 250 | alto 250 | ancho 250 | alto 250 | ancho 250 | alto 250 | ancho 250 | alto 250 |
| | Número de piezas 10 | | Número de piezas 20 | | Número de piezas 30 | | Número de piezas 50 | |
| no. | ancho | alto | ancho | alto | ancho | alto | ancho | alto |
| 1 | 167 | 184 | 120 | 133 | 66 | 80 | 139 | 103 |
| 2 | 114 | 118 | 135 | 186 | 164 | 107 | 166 | 99 |
| 3 | 167 | 152 | 86 | 75 | 64 | 184 | 88 | 74 |
| 4 | 83 | 140 | 103 | 73 | 121 | 86 | 174 | 139 |
| 5 | 70 | 86 | 66 | 85 | 163 | 135 | 128 | 109 |
| 6 | 143 | 166 | 135 | 97 | 85 | 98 | 175 | 179 |
| 7 | 120 | 160 | 91 | 175 | 81 | 102 | 62 | 82 |
| 8 | 66 | 148 | 131 | 176 | 103 | 186 | 178 | 87 |
| 9 | 87 | 141 | 71 | 176 | 152 | 106 | 121 | 77 |
| 10 | 69 | 165 | 153 | 72 | 176 | 139 | 125 | 97 |
| 11 | | | 87 | 148 | 111 | 118 | 73 | 84 |
| 12 | | | 168 | 107 | 69 | 169 | 69 | 97 |
| 13 | | | 118 | 90 | 146 | 133 | 122 | 159 |
| 14 | | | 140 | 109 | 112 | 112 | 66 | 69 |
| 15 | | | 132 | 159 | 133 | 160 | 111 | 118 |
| 16 | | | 152 | 93 | 63 | 129 | 165 | 86 |
| 17 | | | 135 | 68 | 163 | 152 | 71 | 83 |
| 18 | | | 121 | 158 | 110 | 155 | 163 | 162 |
| 19 | | | 68 | 94 | 96 | 136 | 67 | 132 |
| 20 | | | 155 | 76 | 92 | 142 | 121 | 152 |
| 21 | | | | | 84 | 143 | 68 | 150 |
| 22 | | | | | 119 | 133 | 138 | 124 |
| 23 | | | | | 71 | 71 | 141 | 105 |
| 24 | | | | | 146 | 84 | 183 | 122 |
| 25 | | | | | 93 | 86 | 171 | 63 |
| 26 | | | | | 89 | 86 | 101 | 64 |
| 27 | | | | | 101 | 146 | 103 | 95 |
| 28 | | | | | 172 | 73 | 110 | 107 |
| 29 | | | | | 73 | 169 | 184 | 158 |
| 30 | | | | | 99 | 99 | 85 | 97 |
| 31 | | | | | | | 86 | 92 |
| 32 | | | | | | | 127 | 155 |
| 33 | | | | | | | 92 | 147 |
| 34 | | | | | | | 106 | 97 |
| 35 | | | | | | | 177 | 108 |
| 36 | | | | | | | 164 | 150 |
| 37 | | | | | | | 68 | 84 |
| 38 | | | | | | | 62 | 151 |
| 39 | | | | | | | 123 | 134 |
| 40 | | | | | | | 158 | 178 |
| 41 | | | | | | | 70 | 140 |
| 42 | | | | | | | 173 | 117 |
| 43 | | | | | | | 147 | 136 |
| 44 | | | | | | | 141 | 145 |
| 45 | | | | | | | 84 | 164 |
| 46 | | | | | | | 144 | 141 |
| 47 | | | | | | | 92 | 103 |
| 48 | | | | | | | 156 | 98 |
| 49 | | | | | | | 160 | 111 |
| 50 | | | | | | | 127 | 131 |

Tabla A.3: nstancias de problemas de corte en dos dimensiones guillotinables sin restricciones

| | *gcut5* | | *gcut6* | | *gcut7* | | *gcut8* | |
|---|---|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | | Objetos | |
| | ancho 500 | alto 500 | ancho 500 | alto 500 | ancho 500 | alto 500 | ancho 500 | alto 500 |
| | Número de piezas 10 | | Número de piezas 20 | | Número de piezas 30 | | Número de piezas 50 | |
| no. | ancho | alto | ancho | alto | ancho | alto | ancho | alto |
| 1 | 198 | 205 | 313 | 305 | 348 | 220 | 277 | 242 |
| 2 | 179 | 155 | 292 | 222 | 255 | 184 | 233 | 177 |
| 3 | 364 | 236 | 330 | 253 | 191 | 249 | 182 | 216 |
| 4 | 272 | 147 | 212 | 256 | 212 | 194 | 147 | 134 |
| 5 | 352 | 145 | 132 | 189 | 348 | 322 | 213 | 127 |
| 6 | 343 | 245 | 149 | 296 | 171 | 369 | 315 | 212 |
| 7 | 132 | 174 | 294 | 137 | 206 | 327 | 261 | 175 |
| 8 | 164 | 250 | 225 | 345 | 221 | 277 | 210 | 281 |
| 9 | 282 | 356 | 345 | 220 | 250 | 202 | 243 | 256 |
| 10 | 342 | 151 | 337 | 177 | 205 | 226 | 159 | 314 |
| 11 | | | 189 | 300 | 193 | 280 | 217 | 147 |
| 12 | | | 234 | 321 | 364 | 311 | 260 | 349 |
| 13 | | | 335 | 272 | 264 | 224 | 222 | 199 |
| 14 | | | 354 | 244 | 244 | 347 | 296 | 127 |
| 15 | | | 149 | 169 | 270 | 338 | 263 | 302 |
| 16 | | | 355 | 165 | 224 | 236 | 226 | 250 |
| 17 | | | 260 | 220 | 168 | 177 | 264 | 344 |
| 18 | | | 210 | 241 | 285 | 347 | 237 | 179 |
| 19 | | | 194 | 372 | 315 | 294 | 245 | 137 |
| 20 | | | 287 | 134 | 247 | 219 | 137 | 328 |
| 21 | | | | | 315 | 302 | 321 | 154 |
| 22 | | | | | 129 | 147 | 200 | 174 |
| 23 | | | | | 268 | 273 | 270 | 128 |
| 24 | | | | | 350 | 352 | 299 | 179 |
| 25 | | | | | 186 | 258 | 165 | 370 |
| 26 | | | | | 365 | 374 | 198 | 182 |
| 27 | | | | | 145 | 259 | 155 | 295 |
| 28 | | | | | 284 | 184 | 278 | 219 |
| 29 | | | | | 129 | 198 | 184 | 155 |
| 30 | | | | | 146 | 351 | 187 | 270 |
| 31 | | | | | | | 208 | 293 |
| 32 | | | | | | | 261 | 204 |
| 33 | | | | | | | 344 | 168 |
| 34 | | | | | | | 213 | 135 |
| 35 | | | | | | | 362 | 162 |
| 36 | | | | | | | 362 | 374 |
| 37 | | | | | | | 237 | 322 |
| 38 | | | | | | | 281 | 308 |
| 39 | | | | | | | 155 | 344 |
| 40 | | | | | | | 210 | 163 |
| 41 | | | | | | | 325 | 225 |
| 42 | | | | | | | 360 | 225 |
| 43 | | | | | | | 346 | 347 |
| 44 | | | | | | | 131 | 144 |
| 45 | | | | | | | 284 | 284 |
| 46 | | | | | | | 262 | 228 |
| 47 | | | | | | | 146 | 178 |
| 48 | | | | | | | 254 | 164 |
| 49 | | | | | | | 332 | 238 |
| 50 | | | | | | | 313 | 304 |

Tabla A.4: Problemas de corte en dos dimensiones guillotinables sin restricciones.

| | *gcut*9 | | *gcut*10 | | *gcut*11 | | *gcut*12 | |
|---|---|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | | Objetos | |
| | ancho 1000 | alto 1000 | ancho 1000 | alto 1000 | ancho 1000 | alto 1000 | ancho 1000 | alto 1000 |
| | Número de piezas 10 | | Número de piezas 20 | | Número de piezas 30 | | Número de piezas 50 | |
| no. | ancho | alto | ancho | alto | ancho | alto | ancho | alto |
| 1 | 310 | 426 | 730 | 300 | 541 | 745 | 572 | 665 |
| 2 | 673 | 468 | 269 | 717 | 344 | 301 | 482 | 640 |
| 3 | 426 | 463 | 463 | 369 | 413 | 294 | 264 | 594 |
| 4 | 325 | 498 | 642 | 464 | 266 | 341 | 349 | 566 |
| 5 | 555 | 540 | 453 | 329 | 543 | 388 | 276 | 660 |
| 6 | 292 | 455 | 455 | 667 | 367 | 701 | 745 | 422 |
| 7 | 343 | 341 | 506 | 482 | 526 | 707 | 434 | 622 |
| 8 | 362 | 491 | 560 | 362 | 286 | 706 | 446 | 433 |
| 9 | 305 | 688 | 483 | 260 | 614 | 289 | 668 | 506 |
| 10 | 459 | 607 | 693 | 345 | 348 | 592 | 405 | 635 |
| 11 | | | 381 | 510 | 673 | 431 | 486 | 320 |
| 12 | | | 456 | 586 | 475 | 362 | 554 | 549 |
| 13 | | | 457 | 453 | 562 | 439 | 392 | 491 |
| 14 | | | 707 | 658 | 530 | 638 | 459 | 528 |
| 15 | | | 639 | 650 | 609 | 375 | 426 | 579 |
| 16 | | | 691 | 359 | 540 | 274 | 728 | 359 |
| 17 | | | 434 | 700 | 486 | 634 | 644 | 465 |
| 18 | | | 576 | 291 | 272 | 377 | 382 | 323 |
| 19 | | | 728 | 739 | 623 | 306 | 268 | 352 |
| 20 | | | 545 | 742 | 266 | 714 | 324 | 512 |
| 21 | | | | | 341 | 336 | 277 | 426 |
| 22 | | | | | 340 | 296 | 254 | 685 |
| 23 | | | | | 531 | 479 | 483 | 532 |
| 24 | | | | | 571 | 589 | 678 | 414 |
| 25 | | | | | 674 | 496 | 546 | 590 |
| 26 | | | | | 560 | 497 | 716 | 640 |
| 27 | | | | | 509 | 389 | 542 | 400 |
| 28 | | | | | 650 | 332 | 634 | 596 |
| 29 | | | | | 474 | 285 | 497 | 696 |
| 30 | | | | | 419 | 503 | 417 | 300 |
| 31 | | | | | | | 290 | 605 |
| 32 | | | | | | | 321 | 520 |
| 33 | | | | | | | 543 | 290 |
| 34 | | | | | | | 599 | 269 |
| 35 | | | | | | | 467 | 679 |
| 36 | | | | | | | 583 | 421 |
| 37 | | | | | | | 729 | 723 |
| 38 | | | | | | | 465 | 590 |
| 39 | | | | | | | 614 | 585 |
| 40 | | | | | | | 446 | 327 |
| 41 | | | | | | | 734 | 616 |
| 42 | | | | | | | 479 | 606 |
| 43 | | | | | | | 656 | 278 |
| 44 | | | | | | | 537 | 355 |
| 45 | | | | | | | 604 | 519 |
| 46 | | | | | | | 746 | 596 |
| 47 | | | | | | | 439 | 281 |
| 48 | | | | | | | 339 | 405 |
| 49 | | | | | | | 273 | 526 |
| 50 | | | | | | | 625 | 568 |

Tabla A.5: Instancias de problemas de corte en dos dimensiones guillotinables sin restricciones.

| | *gcut*13 Objetos | |
|---|---|---|
| | ancho 3000 | alto 3000 |
| | Número de piezas 32 | |
| no. | ancho | alto |
| 1 | 365 | 185 |
| 2 | 378 | 200 |
| 3 | 410 | 165 |
| 4 | 425 | 148 |
| 5 | 425 | 296 |
| 6 | 439 | 116 |
| 7 | 464 | 1006 |
| 8 | 520 | 205 |
| 9 | 520 | 350 |
| 10 | 540 | 530 |
| 11 | 549 | 1413 |
| 12 | 549 | 1882 |
| 13 | 553 | 496 |
| 14 | 555 | 755 |
| 15 | 555 | 496 |
| 16 | 555 | 659 |
| 17 | 567 | 473 |
| 18 | 572 | 592 |
| 19 | 572 | 975 |
| 20 | 572 | 1175 |
| 21 | 572 | 1575 |
| 22 | 572 | 1390 |
| 23 | 572 | 1490 |
| 24 | 572 | 1590 |
| 25 | 572 | 1690 |
| 26 | 572 | 1890 |
| 27 | 610 | 625 |
| 28 | 660 | 490 |
| 29 | 690 | 447 |
| 30 | 949 | 445 |
| 31 | 949 | 478 |
| 32 | 970 | 463 |

Tabla A.6: Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones.

| | *ngcut1* | | *ngcut2* | | *ngcut3* | |
|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | |
| | ancho | alto | ancho | alto | ancho | alto |
| | 10 | 10 | 10 | 10 | 10 | 10 |
| | Número de piezas | | Número de piezas | | Número de piezas | |
| | 10 | | 17 | | 21 | |
| no. | ancho | alto | ancho | alto | ancho | alto |
| 1 | 3 | 7 | 1 | 10 | 3 | 2 |
| 2 | 3 | 7 | 1 | 10 | 3 | 2 |
| 3 | 8 | 2 | 1 | 10 | 7 | 2 |
| 4 | 8 | 2 | 5 | 3 | 7 | 2 |
| 5 | 10 | 2 | 5 | 3 | 7 | 2 |
| 6 | 5 | 4 | 9 | 3 | 4 | 5 |
| 7 | 5 | 4 | 9 | 3 | 4 | 5 |
| 8 | 5 | 4 | 9 | 3 | 4 | 1 |
| 9 | 2 | 9 | 6 | 1 | 4 | 1 |
| 10 | 2 | 9 | 6 | 1 | 1 | 10 |
| 11 | | | 6 | 1 | 8 | 4 |
| 12 | | | 3 | 8 | 4 | 2 |
| 13 | | | 3 | 8 | 4 | 2 |
| 14 | | | 3 | 8 | 3 | 7 |
| 15 | | | 4 | 1 | 3 | 7 |
| 16 | | | 7 | 3 | 3 | 7 |
| 17 | | | 7 | 3 | 6 | 2 |
| 18 | | | | | 6 | 2 |
| 19 | | | | | 6 | 2 |
| 20 | | | | | 9 | 1 |
| 21 | | | | | 9 | 1 |

Tabla A.7: Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones.

| | *ngcut4* | | *ngcut5* | | *ngcut6* | |
|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | |
| | ancho | alto | ancho | alto | ancho | alto |
| | 15 | 10 | 15 | 10 | 15 | 10 |
| | Número de piezas | | Número de piezas | | Número de piezas | |
| | 7 | | 14 | | 15 | |
| no. | ancho | alto | ancho | alto | ancho | alto |
| 1 | 15 | 2 | 10 | 3 | 2 | 9 |
| 2 | 15 | 2 | 10 | 3 | 10 | 2 |
| 3 | 7 | 3 | 9 | 3 | 10 | 2 |
| 4 | 9 | 1 | 12 | 2 | 10 | 2 |
| 5 | 8 | 3 | 12 | 2 | 2 | 4 |
| 6 | 12 | 2 | 12 | 2 | 2 | 8 |
| 7 | 12 | 2 | 11 | 3 | 2 | 8 |
| 8 | | | 12 | 3 | 3 | 8 |
| 9 | | | 12 | 3 | 4 | 1 |
| 10 | | | 12 | 3 | 6 | 4 |
| 11 | | | 11 | 1 | 6 | 4 |
| 12 | | | 11 | 1 | 6 | 4 |
| 13 | | | 11 | 1 | 10 | 3 |
| 14 | | | 2 | 10 | 11 | 2 |
| 15 | | | | | 4 | 5 |

Tabla A.8: Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones.

| | *ngcut7* | | *ngcut8* | | *ngcut9* | |
|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | |
| | ancho 20 | alto 20 | ancho 20 | alto 20 | ancho 20 | alto 20 |
| | Número de piezas 8 | | Número de piezas 13 | | Número de piezas 18 | |
| no. | ancho | alto | ancho | alto | ancho | alto |
| 1 | 1 | 9 | 15 | 3 | 14 | 2 |
| 2 | 1 | 9 | 6 | 6 | 1 | 5 |
| 3 | 1 | 9 | 6 | 6 | 20 | 4 |
| 4 | 16 | 3 | 6 | 6 | 20 | 4 |
| 5 | 18 | 3 | 13 | 1 | 12 | 3 |
| 6 | 20 | 2 | 13 | 1 | 12 | 3 |
| 7 | 3 | 1 | 8 | 7 | 12 | 3 |
| 8 | 3 | 1 | 8 | 7 | 11 | 8 |
| 9 | | | 18 | 5 | 11 | 8 |
| 10 | | | 18 | 5 | 11 | 6 |
| 11 | | | 18 | 5 | 11 | 6 |
| 12 | | | 12 | 4 | 7 | 9 |
| 13 | | | 8 | 3 | 17 | 5 |
| 14 | | | | | 7 | 14 |
| 15 | | | | | 7 | 14 |
| 16 | | | | | 1 | 7 |
| 17 | | | | | 1 | 7 |
| 18 | | | | | 1 | 7 |

Tabla A.9: Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones.

| | *ngcut10* | | *ngcut11* | | *ngcut12* | |
|---|---|---|---|---|---|---|
| | Objetos | | Objetos | | Objetos | |
| | ancho 30 | alto 30 | ancho 30 | alto 30 | ancho 30 | alto 30 |
| | Número de piezas 13 | | Número de piezas 15 | | Número de piezas 22 | |
| no. | ancho | alto | ancho | alto | ancho | alto |
| 1 | 1 | 30 | 3 | 23 | 16 | 6 |
| 2 | 1 | 30 | 3 | 23 | 24 | 1 |
| 3 | 1 | 30 | 3 | 23 | 24 | 1 |
| 4 | 25 | 7 | 29 | 5 | 24 | 1 |
| 5 | 27 | 9 | 21 | 2 | 6 | 28 |
| 6 | 27 | 9 | 21 | 2 | 6 | 28 |
| 7 | 27 | 9 | 21 | 2 | 6 | 28 |
| 8 | 30 | 2 | 17 | 11 | 8 | 23 |
| 9 | 30 | 2 | 17 | 11 | 8 | 23 |
| 10 | 30 | 2 | 17 | 11 | 8 | 23 |
| 11 | 26 | 7 | 14 | 7 | 5 | 1 |
| 12 | 26 | 7 | 14 | 7 | 5 | 1 |
| 13 | 26 | 7 | 8 | 5 | 5 | 1 |
| 14 | | | 8 | 5 | 6 | 26 |
| 15 | | | 21 | 8 | 6 | 26 |
| 16 | | | | | 2 | 30 |
| 17 | | | | | 2 | 30 |
| 18 | | | | | 9 | 11 |
| 19 | | | | | 9 | 11 |
| 20 | | | | | 9 | 11 |
| 21 | | | | | 4 | 30 |
| 22 | | | | | 16 | 13 |

## Apéndice B

# Problemas propuestos en el presente trabajo para corte de materiales en dos dimensiones

Se crearon nuevos problemas de prueba para corte de material en dos dimensiones. Estos problemas de prueba se construyeron con la ayuda de algoritmos generadores de problemas.

Se distinguen dos tipos de problemas guillotinables ($G1$, $G2$ y $G3$) y no guillotinables ($NG1$, $NG2$ y $NG3$), en donde la solución óptima es conocida. El problema generado es almacenado en archivos de texto que contienen el ancho y alto de los objetos y de cada una de las piezas.

Tabla B.1: Instancias de problemas de corte en dos dimensiones guillotinables.

| $G1$ | | |
|---|---|---|
| Objetos | | |
| ancho 250 | | alto 250 |
| Número de piezas 39 | | |
| no. | ancho | alto |
| 1 | 19 | 250 |
| 2 | 18 | 250 |
| 3 | 21 | 250 |
| 4 | 127 | 7 |
| 5 | 127 | 31 |
| 6 | 158 | 78 |
| 7 | 57 | 172 |
| 8 | 127 | 86 |
| 9 | 92 | 44 |
| 10 | 7 | 206 |
| 11 | 16 | 206 |
| 12 | 69 | 206 |
| 13 | 250 | 61 |
| 14 | 37 | 189 |
| 15 | 213 | 9 |
| 16 | 50 | 180 |
| 17 | 163 | 19 |
| 18 | 163 | 29 |
| 19 | 2 | 132 |
| 20 | 250 | 19 |
| 21 | 250 | 58 |
| 22 | 51 | 173 |
| 23 | 40 | 173 |
| 24 | 78 | 173 |
| 25 | 81 | 173 |
| 26 | 161 | 51 |
| 27 | 161 | 81 |
| 28 | 88 | 172 |
| 29 | 13 | 172 |
| 30 | 158 | 87 |
| 31 | 158 | 37 |
| 32 | 60 | 250 |
| 33 | 32 | 250 |
| 34 | 104 | 126 |
| 35 | 54 | 126 |
| 36 | 11 | 250 |
| 37 | 54 | 250 |
| 38 | 127 | 106 |
| 39 | 127 | 20 |

Tabla B.2: Instancias de problemas de corte en dos dimensiones guillotinables.

| *G2* | | | | | |
|---|---|---|---|---|---|
| Objetos | | | | | |
| ancho 250 | | | alto 250 | | |
| Número de piezas 79 | | | | | |
| no. | ancho | alto | no. | ancho | alto |
| 1 | 37 | 250 | 41 | 31 | 250 |
| 2 | 16 | 250 | 42 | 135 | 77 |
| 3 | 45 | 250 | 43 | 135 | 93 |
| 4 | 96 | 86 | 44 | 135 | 80 |
| 5 | 25 | 164 | 45 | 84 | 80 |
| 6 | 71 | 164 | 46 | 84 | 17 |
| 7 | 56 | 133 | 47 | 84 | 153 |
| 8 | 56 | 117 | 48 | 60 | 250 |
| 9 | 250 | 52 | 49 | 26 | 250 |
| 10 | 2 | 198 | 50 | 164 | 54 |
| 11 | 48 | 198 | 51 | 41 | 196 |
| 12 | 134 | 36 | 52 | 13 | 196 |
| 13 | 134 | 90 | 53 | 110 | 12 |
| 14 | 134 | 72 | 54 | 110 | 12 |
| 15 | 66 | 14 | 55 | 5 | 146 |
| 16 | 66 | 184 | 56 | 160 | 104 |
| 17 | 31 | 250 | 57 | 90 | 104 |
| 18 | 11 | 250 | 58 | 91 | 146 |
| 19 | 40 | 250 | 59 | 40 | 146 |
| 20 | 111 | 72 | 60 | 49 | 172 |
| 21 | 32 | 178 | 61 | 61 | 172 |
| 22 | 30 | 250 | 62 | 110 | 146 |
| 23 | 27 | 250 | 63 | 4 | 146 |
| 24 | 250 | 38 | 64 | 54 | 250 |
| 25 | 175 | 61 | 65 | 102 | 23 |
| 26 | 175 | 60 | 66 | 94 | 55 |
| 27 | 175 | 35 | 67 | 11 | 227 |
| 28 | 175 | 56 | 68 | 32 | 227 |
| 29 | 7 | 212 | 69 | 59 | 227 |
| 30 | 68 | 92 | 70 | 94 | 132 |
| 31 | 68 | 120 | 71 | 94 | 63 |
| 32 | 250 | 23 | 72 | 12 | 250 |
| 33 | 24 | 153 | 73 | 58 | 250 |
| 34 | 226 | 59 | 74 | 111 | 62 |
| 35 | 110 | 94 | 75 | 46 | 188 |
| 36 | 116 | 94 | 76 | 65 | 188 |
| 37 | 86 | 74 | 77 | 69 | 7 |
| 38 | 164 | 74 | 78 | 69 | 100 |
| 39 | 66 | 178 | 79 | 69 | 143 |
| 40 | 13 | 178 | | | |

Tabla B.3: Instancias de problemas de corte en dos dimensiones guillotinables.

| *G3* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Objetos | | | | | | | | |
| ancho 250 | | | | | alto 250 | | | |
| Número de piezas 120 | | | | | | | | |
| no. | ancho | alto | no. | ancho | alto | no. | ancho | alto |
| 1 | 145 | 106 | 41 | 120 | 73 | 81 | 130 | 17 |
| 2 | 9 | 250 | 42 | 130 | 20 | 82 | 53 | 214 |
| 3 | 20 | 250 | 43 | 60 | 177 | 83 | 77 | 214 |
| 4 | 145 | 18 | 44 | 60 | 177 | 84 | 120 | 30 |
| 5 | 7 | 250 | 45 | 84 | 145 | 85 | 13 | 201 |
| 6 | 34 | 126 | 46 | 46 | 145 | 86 | 7 | 201 |
| 7 | 111 | 126 | 47 | 88 | 68 | 87 | 26 | 250 |
| 8 | 250 | 18 | 48 | 162 | 68 | 88 | 44 | 250 |
| 9 | 57 | 232 | 49 | 9 | 250 | 89 | 180 | 1 |
| 10 | 18 | 232 | 50 | 172 | 41 | 90 | 180 | 33 |
| 11 | 175 | 30 | 51 | 5 | 110 | 91 | 70 | 216 |
| 12 | 61 | 202 | 52 | 167 | 5 | 92 | 110 | 65 |
| 13 | 114 | 91 | 53 | 146 | 105 | 93 | 110 | 151 |
| 14 | 114 | 111 | 54 | 21 | 105 | 94 | 21 | 201 |
| 15 | 27 | 250 | 55 | 159 | 24 | 95 | 79 | 201 |
| 16 | 42 | 250 | 56 | 11 | 226 | 96 | 181 | 9 |
| 17 | 250 | 42 | 57 | 148 | 105 | 97 | 40 | 241 |
| 18 | 250 | 9 | 58 | 17 | 250 | 98 | 2 | 241 |
| 19 | 250 | 21 | 59 | 44 | 250 | 99 | 64 | 241 |
| 20 | 36 | 97 | 60 | 30 | 250 | 100 | 22 | 241 |
| 21 | 95 | 97 | 61 | 10 | 121 | 101 | 53 | 241 |
| 22 | 119 | 97 | 62 | 12 | 250 | 102 | 14 | 250 |
| 23 | 14 | 81 | 63 | 86 | 137 | 103 | 55 | 250 |
| 24 | 250 | 32 | 64 | 238 | 51 | 104 | 250 | 62 |
| 25 | 23 | 126 | 65 | 238 | 62 | 105 | 69 | 106 |
| 26 | 79 | 126 | 66 | 152 | 73 | 106 | 20 | 82 |
| 27 | 250 | 24 | 67 | 152 | 64 | 107 | 64 | 106 |
| 28 | 39 | 81 | 68 | 53 | 250 | 108 | 117 | 106 |
| 29 | 197 | 81 | 69 | 16 | 250 | 109 | 188 | 82 |
| 30 | 148 | 10 | 70 | 172 | 8 | 110 | 42 | 82 |
| 31 | 250 | 12 | 71 | 172 | 91 | 111 | 138 | 7 |
| 32 | 250 | 52 | 72 | 178 | 32 | 112 | 138 | 114 |
| 33 | 57 | 186 | 73 | 98 | 109 | 113 | 44 | 250 |
| 34 | 193 | 55 | 74 | 80 | 109 | 114 | 206 | 28 |
| 35 | 56 | 131 | 75 | 9 | 109 | 115 | 21 | 222 |
| 36 | 64 | 131 | 76 | 169 | 87 | 116 | 33 | 222 |
| 37 | 73 | 131 | 77 | 169 | 22 | 117 | 50 | 222 |
| 38 | 148 | 106 | 78 | 46 | 250 | 118 | 102 | 28 |
| 39 | 148 | 10 | 79 | 26 | 250 | 119 | 102 | 17 |
| 40 | 130 | 85 | 80 | 250 | 19 | 120 | 102 | 177 |

Tabla B.4: Instancias de problemas de corte en dos dimensiones no guillotinables.

| $NG1$ | | |
|---|---|---|
| Objetos | | |
| ancho<br>10 | | alto<br>10 |
| Número de piezas<br>33 | | |
| no. | ancho | alto |
| 1 | 1 | 5 |
| 2 | 5 | 6 |
| 3 | 6 | 4 |
| 4 | 5 | 1 |
| 5 | 1 | 5 |
| 6 | 1 | 6 |
| 7 | 2 | 3 |
| 8 | 2 | 8 |
| 9 | 3 | 1 |
| 10 | 2 | 2 |
| 11 | 1 | 5 |
| 12 | 3 | 5 |
| 13 | 9 | 3 |
| 14 | 3 | 6 |
| 15 | 5 | 9 |
| 16 | 8 | 1 |
| 17 | 2 | 7 |
| 18 | 5 | 3 |
| 19 | 1 | 7 |
| 20 | 4 | 9 |
| 21 | 5 | 1 |
| 22 | 5 | 8 |
| 23 | 6 | 2 |
| 24 | 4 | 3 |
| 25 | 4 | 5 |
| 26 | 8 | 5 |
| 27 | 2 | 8 |
| 28 | 6 | 2 |
| 29 | 3 | 2 |
| 30 | 2 | 6 |
| 31 | 5 | 1 |
| 32 | 2 | 3 |
| 33 | 5 | 4 |

Tabla B.5: Instancias de problemas de corte en dos dimensiones no guillotinables.

| $NG2$ | | | | | |
|---|---|---|---|---|---|
| Objetos | | | | | |
| ancho 10 | | | alto 10 | | |
| Número de piezas 54 | | | | | |
| no. | ancho | alto | no. | ancho | alto |
| 1 | 7 | 3 | 41 | 8 | 7 |
| 2 | 1 | 5 | 42 | 9 | 3 |
| 3 | 8 | 5 | 43 | 1 | 8 |
| 4 | 2 | 8 | 44 | 2 | 2 |
| 5 | 9 | 2 | 45 | 1 | 3 |
| 6 | 2 | 4 | 46 | 5 | 4 |
| 7 | 9 | 2 | 47 | 6 | 6 |
| 8 | 1 | 6 | 48 | 4 | 9 |
| 9 | 3 | 4 | 49 | 5 | 1 |
| 10 | 1 | 4 | 50 | 1 | 5 |
| 11 | 5 | 5 | 51 | 8 | 9 |
| 12 | 6 | 3 | 52 | 9 | 1 |
| 13 | 1 | 7 | 53 | 1 | 6 |
| 14 | 2 | 1 | 54 | 2 | 4 |
| 15 | 4 | 1 | | | |
| 16 | 5 | 9 | | | |
| 17 | 9 | 1 | | | |
| 18 | 1 | 2 | | | |
| 19 | 5 | 8 | | | |
| 20 | 3 | 3 | | | |
| 21 | 6 | 6 | | | |
| 22 | 9 | 4 | | | |
| 23 | 1 | 7 | | | |
| 24 | 4 | 3 | | | |
| 25 | 1 | 4 | | | |
| 26 | 8 | 8 | | | |
| 27 | 9 | 2 | | | |
| 28 | 1 | 6 | | | |
| 29 | 2 | 4 | | | |
| 30 | 1 | 5 | | | |
| 31 | 8 | 9 | | | |
| 32 | 9 | 1 | | | |
| 33 | 1 | 6 | | | |
| 34 | 2 | 4 | | | |
| 35 | 6 | 8 | | | |
| 36 | 1 | 9 | | | |
| 37 | 7 | 1 | | | |
| 38 | 3 | 9 | | | |
| 39 | 9 | 1 | | | |
| 40 | 1 | 5 | | | |

Tabla B.6: Instancias de problemas de corte en dos dimensiones no guillotinables.

| $NG3$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Objetos | | | | | | | | |
| ancho 10 | | | | | | alto 10 | | |
| Número de piezas 103 | | | | | | | | |
| no. | ancho | alto | no. | ancho | alto | no. | ancho | alto |
| 1 | 5 | 4 | 41 | 2 | 2 | 81 | 3 | 4 |
| 2 | 4 | 5 | 42 | 3 | 4 | 82 | 4 | 4 |
| 3 | 1 | 9 | 43 | 3 | 6 | 83 | 2 | 6 |
| 4 | 6 | 1 | 44 | 6 | 4 | 84 | 3 | 2 |
| 5 | 2 | 3 | 45 | 4 | 8 | 85 | 1 | 3 |
| 6 | 2 | 4 | 46 | 7 | 2 | 86 | 3 | 9 |
| 7 | 4 | 6 | 47 | 1 | 3 | 87 | 4 | 1 |
| 8 | 8 | 1 | 48 | 4 | 6 | 88 | 6 | 4 |
| 9 | 5 | 6 | 49 | 5 | 2 | 89 | 5 | 2 |
| 10 | 4 | 9 | 50 | 1 | 5 | 90 | 1 | 3 |
| 11 | 9 | 1 | 51 | 2 | 3 | 91 | 6 | 3 |
| 12 | 1 | 7 | 52 | 2 | 7 | 92 | 1 | 5 |
| 13 | 6 | 3 | 53 | 7 | 2 | 93 | 6 | 1 |
| 14 | 5 | 5 | 54 | 3 | 9 | 94 | 1 | 1 |
| 15 | 1 | 9 | 55 | 5 | 1 | 95 | 3 | 4 |
| 16 | 6 | 1 | 56 | 1 | 2 | 96 | 4 | 4 |
| 17 | 4 | 6 | 57 | 3 | 4 | 97 | 1 | 5 |
| 18 | 9 | 4 | 58 | 4 | 1 | 98 | 2 | 3 |
| 19 | 4 | 8 | 59 | 5 | 3 | 99 | 5 | 3 |
| 20 | 3 | 9 | 60 | 6 | 2 | 100 | 4 | 4 |
| 21 | 7 | 1 | 61 | 6 | 6 | 101 | 9 | 6 |
| 22 | 3 | 9 | 62 | 1 | 7 | 102 | 1 | 9 |
| 23 | 7 | 1 | 63 | 7 | 3 | 103 | 6 | 1 |
| 24 | 2 | 3 | 64 | 3 | 9 | | | |
| 25 | 3 | 6 | 65 | 9 | 1 | | | |
| 26 | 5 | 3 | 66 | 7 | 5 | | | |
| 27 | 1 | 6 | 67 | 1 | 7 | | | |
| 28 | 3 | 3 | 68 | 8 | 3 | | | |
| 29 | 1 | 2 | 69 | 2 | 8 | | | |
| 30 | 8 | 3 | 70 | 9 | 2 | | | |
| 31 | 2 | 5 | 71 | 2 | 4 | | | |
| 32 | 3 | 5 | 72 | 3 | 6 | | | |
| 33 | 2 | 4 | 73 | 5 | 4 | | | |
| 34 | 3 | 6 | 74 | 5 | 8 | | | |
| 35 | 5 | 1 | 75 | 7 | 2 | | | |
| 36 | 2 | 5 | 76 | 2 | 6 | | | |
| 37 | 4 | 2 | 77 | 8 | 2 | | | |
| 38 | 4 | 1 | 78 | 2 | 8 | | | |
| 39 | 4 | 9 | 79 | 4 | 2 | | | |
| 40 | 8 | 1 | 80 | 1 | 2 | | | |

# Bibliografía


[1] Sara Baase and Allen Van Gelder. *Computer Algorithms, Introduction to Design and Analysis.* Addison Wesley, April 2000.

[2] J E Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research*, pages 297–306, 1985.

[3] J E Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research 33*, pages 49–64, 1985.

[4] J E Beasley. Beasley operations research library. *Collection of problems for 2D packing and cuting*, 2003.

[5] L. B. Booker. Intelligent behavior as an adaptation to the task envoronment. *Computer and Communication Sciences*, 1982.

[6] L.B. Booker. Improving the performance of genetic algorithms in classifier systems. In *Proceedings of the First International Conference on Genetic Algorothms.* Schawer, 1985.

[7] Edmun Burke, Emma Hart, Graham Kendall, Jim Newall, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern research technology. In *Handbook of Metaheuristics.* Kluwer Academic Publishers, 2003.

[8] N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, pages 30–44, 1977.

[9] Peter Cowling, Graham Kendall, and Erick Soubeiga. Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. *Automated Scheduling, optimisation and Planning, University of Nottingham*, 1999.

[10] Charles Darwin. *The Origin of Species by Means of Natural Selection.* W.Clowes and sons, London, 1 edition, October 1859.

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

[12] E.Hart, P.M.Ross, and J.Nelson. Solving a real world problem using an evolving heuristically driven schedule builder. In *Evolutionary Computation*, pages 61–80. 1998.

[13] Garey and Johnson. *NP-Problems.* Addison Wesley, 1 edition, 1998.

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

[15] H.L.Fang, P.M.Ross, and D.Corne. A promising hybrid ga/heuristic aproach for open-shop scheduling problems. In European Conference on Artifitial Inteligence, editor, *Procedings of ECAI*, pages 590–594. A. Cohn, 1994.

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

[17] Eva Hopper. *Packing utilising Evolutionary Algorithms and other Meta-Heuristic Methods.* PhD thesis, University of Wales, School of Engineering, 2000.

[18] Horn J., Goldberg D. E., and Deb K. Implicit nitching in a learning classifier system; nature way. *Evolutionary Computation*, pages 37–66, 1994.

[19] E.G Coffman Jr., M.R. Garey, and D.S. Johnson. *Approximation Algorithms for NP-Hard Problems*, chapter 2-Approximation Algorithms for Bin Packing: A Survey, pages 46–93. PWS Publishing Boston, 1996.

[20] Silvano Martello and Daniele Vigo. Exact solution of the two-dimensional finite bin packing problem. *Dipartimento di Elettronica, Informatica e Sistematica*, 1998.

[21] Peter Ross and Hemma Hart. Using evolutionary algorithms to solve problems by combining choices of heuristics. *Evolutionary optimization*, pages 229–252, 2002.

[22] Sonia Schulenburg, Javier Marín-Blázquez, Peter Ross, and Emma Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. *Learning Classifier Systems*, pages 942–948, 2000.

[23] R.E. Smith and M. Valenzuela-Rendon. A study of the rule development in a learning classifier system. In *Proceedings of the Third International Conference on Genetic Algorothms.* J. D. Schaffer, 1989.

[24] Hugo Terashima-Marín, Peter Ross, and Manuel Valenzuela-Rendón. Evolution of constrait satisfaction strategies in examination timetabling. *Proceding of the Genetic and Evolutionary Computation Conference*, pages 635–642, 1999.

[25] Manuel Valenzuela. Algoritmos genéticos. Clases de Maestría, 2002.

[26] Martin. V.Butz. *Anticipatory Learning Classifier Systems.* kluwer Academic Publishers, 2002.

[27] C. Watkins. Learning from delayed rewards. *Cambridge University*, 1989.

[28] Stewart. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, pages 149–175, 1995.

[29] Stewart. W. Wilson. Generalization in the XCS classifier system. *Evolutionary Computation*, 1998.

# Contenido

**Reconocimientos** vi

**Resumen** vii

**Índice de tablas** xii

**Índice de figuras** xv

**Capítulo 1. Introducción** 1
1.1. Definición del Problema . . . . . 2
1.2. Motivación . . . . . 4
1.3. Objetivos . . . . . 5
1.4. Hipótesis . . . . . 5
1.5. Contribución . . . . . 6
1.6. Organización de la Tesis . . . . . 6

**Capítulo 2. Antecedentes** 7
2.1. Corte de Materiales en Dos Dimensiones . . . . . 7
2.1.1. Clasificación de Dyckhoff . . . . . 7
2.1.2. El límite inferior continuo . . . . . 11
2.2. Problemas NP-completos . . . . . 11
2.2.1. Clase P . . . . . 11
2.2.2. Clase NP . . . . . 11
2.2.3. Clase NP-Hard . . . . . 12
2.2.4. Clase NP-Completo . . . . . 12
2.3. Heurísticas para problemas de corte . . . . . 12
2.3.1. Heurísticas de selección [17] . . . . . 13
2.3.2. Heurísticas para acomodo . . . . . 15
2.4. Algoritmo genético . . . . . 18
2.4.1. Población . . . . . 19
2.4.2. Aptitud . . . . . 20
2.4.3. Selección . . . . . 20

2.4.4. Cruce . . . 21
2.4.5. Mutación . . . 22
2.4.6. Generación de una nueva población . . . 23
2.4.7. Representación . . . 23
2.5. Sistemas de Clasificadores . . . 24
2.6. Hiper-heurística . . . 26
2.6.1. El concepto y sus orígenes . . . 26
2.6.2. Concepto de hiper-heurística . . . 26
2.6.3. Trabajos Relacionados . . . 27
2.6.4. Marco de Trabajo de la Hiper-Heurística . . . 30
2.7. Resumen . . . 32

**Capítulo 3. Sistema de Clasificadores XCS** **33**
3.1. Antecedentes . . . 33
3.2. Medición de la aptitud . . . 34
3.3. Estructura de los clasificadores en el sistema XCS . . . 36
3.4. Componentes de desempeño . . . 37
3.5. Componentes de refuerzo . . . 37
3.6. Componente de descubrimiento . . . 38
3.7. Cálculo de la aptitud . . . 40
3.8. Macroclasificadores . . . 40
3.9. Lista de parámetros del sistema . . . 41
3.10. Mecanismo de generalización . . . 41
3.11. Ambiente de un paso . . . 42
3.12. Ambiente de múltiples pasos . . . 42
3.13. Resumen . . . 42

**Capítulo 4. Modelo de Solución** **43**
4.1. Antecedentes . . . 43
4.2. Representación del ambiente . . . 45
4.2.1. Representación del problema de corte como ambiente de un paso 46
4.2.2. Representación del problema de corte como ambiente de múltiples pasos . . . 46
4.3. Representación de reglas . . . 46
4.4. Codificación de la condición de los clasificadores . . . 47
4.5. Codificación de las acciones de los clasificadores . . . 49
4.6. Función de Evaluación . . . 51
4.7. Parámetros del sistema . . . 52
4.8. Resumen . . . 53

**Capítulo 5. Experimentos y análisis de resultados** **54**
5.1. Cálculo del límite inferior continuo . . . . . 54
5.2. Experimentos y análisis de resultados con problemas de la literatura . . 55
5.2.1. Propiedades de los problemas . . . . . 55
5.2.2. Problemas de corte de material guillotinable con restricciones . 59
5.2.3. Problemas de corte de material guillotinable sin restricciones . . 65
5.2.4. Problemas de corte de material no guillotinable . . . . . 72
5.3. Experimentos y análisis de resultados con problemas propuestos . . . . 76
5.3.1. Problemas de corte de material guillotinable . . . . . 76
5.3.2. Problemas de corte de material no guillotinable . . . . . 78
5.3.3. Desempeño de heurísticas individuales . . . . . 79
5.3.4. Desempeño de hiper-heurística . . . . . 81
5.4. Resumen . . . . . 81

**Capítulo 6. Conclusiones** **82**
6.1. Contribuciones y conclusiones . . . . . 82
6.2. Trabajo futuro . . . . . 83

**Apéndice A. Problemas de la literatura para corte de materiales en dos dimensiones** **84**

**Apéndice B. Problemas propuestos en el presente trabajo para corte de materiales en dos dimensiones** **92**

**Bibliografía** **99**

**Vita** **102**

# Hiper-Heurística a través de Sistemas de Clasificadores para solucionar problemas de corte de material en dos dimensiones

Edgardo Javier Flores Álvarez, M.C.
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2004

Asesor de la tesis: Dr. Hugo Terashima Marín

El problema de corte y empacado de materiales involucra una gran cantidad de aplicaciones, por ejemplo, empacado de productos, carga de vehículos y contenedores, corte de materiales, calendarización de tareas en intervalos definidos de tiempo, organización de localidades de memoria, entre otros. Es por ésto que ha sido ampliamente estudiado en los últimos años.

En esta tesis, se muestra un método que utiliza sistemas de clasificadores para orientar la búsqueda de una hiper-heurística, que represente la mejor solución al problema. Un sistema de clasificadores es un sistema de aprendizaje de máquinas que aprende sintácticamente conjuntos de reglas simples (llamadas clasificadores). Una hiper-heurística se refiere a la selección inteligente de la heurística o el algoritmo correcto para una situación dada.

Para lograr lo anterior, se toman en consideración las soluciones parciales propuestas en diferentes instancias de tiempo por diferentes heurísticas definidas. Cada clasificador, en el sistema de clasificadores, propone dos heurísticas, una para seleccionar que pieza acomodar en los patrones de corte y otra para ubicar su posición y orientación dentro del material. Estas heurísticas representan una solución parcial al problema. Los clasificadores son evolucionados mediante un algoritmo genético para encontrar las mejores heurísticas en las diferentes instancias de tiempo, para formar así una hiper-heurística general que represente la mejor combinación de las heurísticas simples y por lo tanto la mejor solución al problema.

La solución planteada en este documento de tesis se enfoca al problema de corte de figuras rectangulares en dos dimensiones. Los resultados obtenidos por el método

propuesto presentan soluciones satisfactorias a diversos problemas obtenidos de la literatura y a problemas generados aleatoriamente. Se espera que el algoritmo propuesto, ayude a solucionar diversas aplicaciones industriales para corte de materiales, tanto en la industria textil como la industria del vidrio y de lámina.

# Índice de tablas


2.1. Características de los problemas de corte y empacado. . . . . 10
2.2. Demostración de un cruce de un punto en cromosomas arbitrarios de longitud 6 con punto de cruce 2. . . . . 21
2.3. Demostración de un cruce de dos puntos en cromosomas arbitrarios de longitud 6. . . . . 21
2.4. Demostración de un cruce uniforme en cromosomas arbitrarios de longitud 8. . . . . 22
2.5. Demostración de un cruce aritmético en cromosomas arbitrarios. . . . . 22
2.6. Demostración de mutación en la localidad 1 en un cromosoma arbitrario de longitud 8. . . . . 23

4.1. Estructura de las reglas en los clasificadores del XCS. . . . . 46
4.2. Intervalo del área de las piezas. . . . . 48
4.3. Intervalo del altura de las piezas. . . . . 48
4.4. Intervalo del anchura de las piezas. . . . . 48
4.5. Porcentaje de piezas en un intervalo de tamaño. . . . . 48
4.6. Porcentaje de piezas restantes por acomodar. . . . . 49
4.7. Representación de acciones. . . . . 50

5.1. Características de los problemas cgcut1-cgcut3. . . . . 56
5.2. Características de los problemas gcut1-gcut4. . . . . 56
5.3. Características de los problemas gcut5-gcut8. . . . . 56
5.4. Características de los problemas gcut9-gcut12. . . . . 57
5.5. Características del problema gcut13. . . . . 57
5.6. Características de los problemas ngcut1-ngcut3. . . . . 57
5.7. Características de los problemas ngcut4-ngcut6. . . . . 58
5.8. Características de los problemas ngcut7-ngcut9. . . . . 58
5.9. Características de los problemas ngcut10-ngcut12. . . . . 58
5.10. Desempeño de heurísticas individuales para los problemas cgcut1-cgcut3. 60
5.11. Resultados en instancias de problemas de corte en dos dimensiones guillotinables con restricciones. . . . . 61

5.12. Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte de material guillotinable con restricciones. . . . . 61
5.13. Desempeño de heurísticas individuales para los problemas gcut1-gcut7. 67
5.14. Desempeño de heurísticas individuales para los problemas gcut1-gcut4. 68
5.15. Resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones. . . . . 69
5.16. Resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones. . . . . 69
5.17. Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones. . 70
5.18. Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones. . 70
5.19. Desempeño de heurísticas individuales para los problemas ngcut1-ngcut6. 73
5.20. Desempeño de heurísticas individuales para los problemas ngcut8-ngcut12. 74
5.21. Resultados en instancias de problemas de corte en dos dimensiones no guillotinables. . . . . 75
5.22. Resultados en instancias de problemas de corte en dos dimensiones no guillotinables. . . . . 75
5.23. Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones no guillotinables. . . . . 75
5.24. Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones no guillotinables. . . . . 76
5.25. Desempeño de heurísticas individuales para los problemas propuestos. . 80
5.26. Resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones. . . . . 81
5.27. Tiempo promedio requerido para hallar los resultados en instancias de problemas de corte en dos dimensiones guillotinables sin restricciones. . 81

A.1. Instancias de problemas de corte en dos dimensiones guillotinables con restricciones. . . . . 85
A.2. Instancias de problemas de corte en dos dimensiones guillotinables sin restricciones . . . . . 86
A.3. nstancias de problemas de corte en dos dimensiones guillotinables sin restricciones . . . . . 87
A.4. Problemas de corte en dos dimensiones guillotinables sin restricciones. . 88
A.5. Instancias de problemas de corte en dos dimensiones guillotinables sin restricciones. . . . . 89
A.6. Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones. . . . . 90

A.7. Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones. . . . . 90
A.8. Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones. . . . . 91
A.9. Instancias de problemas de corte en dos dimensiones no guillotinables con restricciones. . . . . 91

B.1. Instancias de problemas de corte en dos dimensiones guillotinables. . . 93
B.2. Instancias de problemas de corte en dos dimensiones guillotinables. . . 94
B.3. Instancias de problemas de corte en dos dimensiones guillotinables. . . 95
B.4. Instancias de problemas de corte en dos dimensiones no guillotinables. . 96
B.5. Instancias de problemas de corte en dos dimensiones no guillotinables. . 97
B.6. Instancias de problemas de corte en dos dimensiones no guillotinables. . 98

# Índice de figuras


1.1. Esquema del Problema de Corte. . . . . 3

2.1. Bottom-Left Placement Rule (BL). . . . . 16
2.2. Improved Bottom-Left Placement Rule (BLLT). . . . . 17
2.3. Bottom-Left Fill Placement Rule (BLF). . . . . 18
2.4. Curva de mejor encontrado típica para un algoritmo de optimización ciega que realiza maximización. . . . . 24
2.5. Esquema de Hiper-Heurística. . . . . 30

3.1. Esquema del XCS con el AG en conjunto de concordancia [M]. . . . . 35
3.2. Esquema del XCS. . . . . 39

4.1. Modelo de Hiper-Heurística con Sistema de Clasificadores XCS. . . . . 44
4.2. Función de evaluación para el problema de corte. . . . . 52

5.1. Heurísticas seleccionadas por el XCS en un ambiente de un paso para el problema cgcut1. . . . . 62
5.2. Heurísticas seleccionadas por el XCS en un ambiente de un paso para el problema cgcut3. . . . . 63
5.3. Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema cgcut1. . . . . 64
5.4. Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema cgcut2. . . . . 64
5.5. Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema cgcut3. . . . . 65
5.6. Heurísticas seleccionadas por el XCS en un ambiente de un paso para el problema gcut1. . . . . 71
5.7. Heurísticas seleccionadas por el XCS en un ambiente de múltiples pasos para el problema gcut1. . . . . 72
5.8. División de un nuevo rectángulo en dos nuevos rectángulos. . . . . 77
5.9. Ejemplo de problema de corte guillotinable generado. . . . . 77

5.10. División de un nuevo rectángulo en cinco nuevos rectángulos para un problema no guillotinable. . . . . 79
5.11. Ejemplo de problema de corte no guillotinable generado. . . . . 79