ICT-UNPA-41-2012
ISSN: 1852- 4516
Aprobado por Resolución Nro. 0348/12-R-UNPA

MANEJO DE RESTRICCIONES CON ALGORITMOS EVOLUTIVOS APLICADOS AL PROBLEMA DE OPTIMIZACION DE CORTE Y EMPAQUETADO

Beca de Investigación para Estudiantes Avanzados

Alumno Eduardo Burgos, Carrera Ing. En Sistemas UNPA - UACO, eaburgos@msn.com
Ing. Marta Lasso, Docente Investigador UNPA, mlasso@uaco.unpa.edu.ar

UNIVERSIDAD NACIONAL DE LA PATAGONIA AUSTRAL.
UNIDAD ACADÉMICA CALETA OLIVIA.
MARZO 2011

Resumen:
Se considera el problema de encontrar un patrón de empaquetamiento de rectángulos de distintos tamaños, dentro de un rectángulo de mayor tamaño cuyo objetivo consiste en que el área desperdiciada sea mínima.
El problema de corte de piezas rectangulares pertenece a la familia de problemas de corte y empaque y sus aplicaciones se pueden observar en industrias de perfiles metálicos, corte de maderas, papel, plástico o vidrio en donde los componentes rectangulares tienen que ser cortados desde grandes hojas de material realizando cortes perpendiculares a los ejes horizontales y verticales.
Este problema de empaquetamiento y corte es de interés académico y científico ya que sirve como base para problemas de mayor complejidad.

ICT-UNPA-41-2012
ISSN: 1852- 4516
Aprobado por Resolución Nro. 0348/12-R-UNPA

1. INTRODUCCIÓN.

En este trabajo se va a abordar un problema industrial real, consistente en la colocación de unas figuras rectangulares dentro de una superficie de base rectangular con el fin de aprovechar al máximo el material de dicha superficie o que el área desperdiciada sea mínima.
Este problema de empaquetamiento en dos dimensiones es de interés científico y académico ya que sirve como plataforma para problemas de mayor complejidad, como el de tres dimensiones. Es un problema complejo, dado que los patrones de empaquetamiento incrementan exponencialmente con el número de rectángulos que deben ser empacados. La magnitud del espacio de búsqueda es mayor que el del problema del *agente viajero*.
Por ejemplo, si el número de placas a ser ubicadas es de 25, el tamaño del espacio de soluciones está dado por 2^25*25! mientras que el espacio de soluciones del viajero es del orden de 10^31 para un caso con igual número de ciudades considerando que todas las ciudades estén conectadas entre si. 2^25*25!>10^31. Esta es una de las razones para usar las técnicas metaheurísticas como herramienta de solución. [2,4]
Con el objetivo de encontrar un patrón de empaquetado de altura mínima y tomando como punto de partida un algoritmo genético básico, en este trabajo investigamos las ventajas de utilizar operadores de recombinación y mutación que incorporan conocimiento específico del problema, tal como información respecto de la asignación de las piezas. En particular, analizamos el comportamiento de esta metaheurística al variar distintos operadores de recombinación y mutación y estudiamos sus ventajas relativas para resolver el problema, con el objetivo de encontrar la mejor configuración. A fin de reducir el desperdicio en cada nivel, se aplica un operador adicional, llamado operador de relocalización, a cada hijo generado. También las ventajas de agregar semillas en la población inicial, las cuales se generan utilizando un conjunto de reglas de construcción; la característica esencial es incluir información del problema en cuestión, tal como ancho de piezas, área de las piezas,
etc. La intención de este estudio es que la población inicial resulte más especializada para el problema bajo análisis. El principal objetivo es hallar un algoritmo genético buscando la mejor relación entre exploración y explotación del proceso de búsqueda. [3]

2. DOCUMENTO.

2.1 MARCOS DE REFERENCIA.

2.1.1 MARCO HISTÓRICO.

Las actividades, del becario se inician con una etapa de investigación intensiva para actualizar el conocimiento del estado del arte y de ahí en más enfocarse en la revisión

bibliográfica en profundidad de los temas particulares de la presente propuesta a fin de determinar e implementar los nuevos conceptos desarrollados.
En todos los casos, se busca transferir la experiencia adquirida hasta el momento sobre campos de aplicación de interés general (ej., corte y empaquetad) y aprovechar el potencial de las distintas variantes de la metaheurísticas para resolver problemas de mayor complejidad vinculados al área industrial con características dinámicas, incorporar conceptos avanzados de manera tal que el proceso de búsqueda pueda ser realizado de manera más eficiente.
Es indudable el impacto que el desarrollo de las metaheurísticas ha tenido en muchos campos de aplicación, tanto en el ámbito académico como en problemas del mundo real. Por otro lado, en los últimos años se ha producido un interesante avance en este ámbito dado por los desarrollos teóricos y de aplicación alcanzado en las distintas metaheurísticas por separado y también por el alto grado de integración entre las mismas, lo que ha permitido crear híbridos de muy buen rendimiento como así también la posibilidad de incorporar conceptos de diseño que son comunes a ellas. Esta situación abre la posibilidad, entre otras cosas, al estudio de los diferentes mecanismos que se podrían incorporar a las metaheurísticas a fin de mejorar sus capacidades exploratorias, como por ejemplo incorporando mecanismos avanzados o alterativos para explotar la experiencia de búsqueda y mejorar las decisiones inmediatas respecto a la dirección de la exploración del espacio de búsqueda. Adicionalmente, se prevé explotar dichas capacidades sobre problemas concretos del mundo real: planificación o scheduling dinámico en problemas de corte y empaquetado.

2.1.2 MARCO CONCEPTUAL.

En un problema de empaquetado de rectángulos se dispone de un conjunto de piezas rectangulares o placas con longitudes conocidas y se desean distribuir estas en un objeto rectangular de mayor tamaño, denominado plancha, de forma que se minimice el área desperdiciada en la plancha o se obtenga una altura mínima.
Este problema se conoce comúnmente como el problema restringido bidimensional de corte guillotina. Un corte es de tipo guillotina si cuando se aplica sobre un rectángulo produce dos nuevos rectángulos, es decir, si el corte va de un extremo a otro del rectángulo original.
La consideración de diferentes tipos de restricciones puede originar problemas de empaquetamiento totalmente distintos. [5]
Pueden considerarse las siguientes tres situaciones referidas al conocimiento que se tiene del objeto en que hay que distribuir las piezas:

1) No son conocidos ni el largo ni el ancho de la plancha. El propósito es distribuir las piezas de tal forma que el rectángulo que esta distribución determina sea el de menor área.
2) El largo y el ancho de la plancha son conocidos. Se pretende, por tanto, distribuir en el objeto aquellas piezas que optimicen la función objetiva determinada.
3) Sólo es conocido el ancho de la plancha mientras que el largo se considera flexible. En este tipo de problema se desea encontrar la distribución de piezas que minimice el alto de la plancha.

La minimización de la función objetivo está sujeta al cumplimiento de las siguientes restricciones, a fin de obtener una solución posible:

1) El ancho y la altura de cada placa no debe exceder las dimensiones de la plancha.
2) Todos los cortes realizados sobre la plancha se consideran sin grosor.
3) Pueden haber varias placas del mismo tipo.
4) No se permiten rotaciones de 90º. Esto significa que una placa de ancho w y altura l es distinta a una pieza de ancho l y altura w.
5) Las placas deben ser ubicadas dentro de la plancha de forma ortogonal. Esto quiere decir que los lados de las placas son paralelos a los lados de la plancha.

El objetivo del problema consiste en ubicar las placas sobre la plancha de forma que la diferencia entre el área de la plancha y el área total ocupada por las placas sea mínima. [6]
Una alternativa factible de solución será aquella en la que todas las placas quedan contenidas dentro de la plancha, ubicadas de forma ortogonal y sin solaparse unas con otras. Por lo tanto, se pretende encontrar una solución factible que minimice la altura o el área desperdiciada del material.

2.1.3 MARCO TEÓRICO.

Las principales características del Algoritmo Genético consisten en mantener constante el tamaño de la población de alternativas de solución, de manera que en cada iteración se reemplaza una alternativa de la población usando un eficiente mecanismo de modificación de la misma pero teniendo en cuenta que no se admiten configuraciones repetidas dentro de la población. En cada iteración la población es reemplazada sistemáticamente por un único descendiente generado. [6,7]
Esta estrategia tiene la ventaja de permitir encontrar múltiples soluciones y además conservar la diversidad del conjunto de alternativas. El algoritmo que describe el método es el siguiente:

1) Se genera, con aleatoriedad, una población inicial de soluciones.
2) Se obtienen las funciones objetivo, penalizando las soluciones no factibles.
3) Se seleccionan dos padres usando el método de selección por torneo. En este método, dos configuraciones son elegidas de forma aleatoria y se elige uno de los padres considerando el mejor valor en función objetivo, el otro es descartado. El proceso se ejecuta dos veces para obtener dos padres.
4) El proceso de recombinación y mutación se realiza de forma integrada ya que se deben tener en cuenta consideraciones especiales del problema. Estos operadores actúan sobre los padres escogidos en el paso anterior.
En esta fase se realizan mejoras en la factibilidad y en la optimización, teniendo en cuenta la filosofía de bloques constructivos.
5) Se reemplaza un individuo de la población por la solución encontrada tomando como base lo siguiente:
    a) Si la alternativa actual no es factible y a su vez es menos infactible que la peor infactible de la población, entonces reemplazar la peor infactible por la

alternativa actual. Es decir, entre dos soluciones no factibles se prefiere la solución menos infactible.

b) Si la configuración es factible y existe por lo menos una infactible en la población actual, entonces reemplazar la peor infactible por la alternativa actual. Es decir, entre una solución factible y una no factible se prefiere la solución factible.

c) Si la configuración es factible y todas las alternativas de la población actual son factibles, entonces reemplazar la alternativa con peor función objetivo por la alternativa actual. Lo anterior se realiza sólo si la alternativa actual es de mejor calidad que la peor de la población. Es decir, entre dos soluciones factibles se prefiere la solución con mejor función objetivo.

El proceso de recombinación, en este problema en particular, debe ser realizado cuidadosamente ya que es posible obtener soluciones no factibles al implementar este operador de la forma tradicional. Si se aplicara este tipo de recombinación se corre el riesgo de obtener una configuración en la cual una placa podría ocupar dos posiciones diferentes sobre la plancha.

Se propone incorporar la filosofía de la recombinación a través de la conservación de bloques constructivos. El proceso consiste en usar dos puntos de corte sobre los padres seleccionados por torneo los cuales son representados por el vector π Posteriormente se aplica un operador de cruzamiento especial. Este operador permite recombinar dos cromosomas sin perder los bloques del cromosoma que ya se había construido. El proceso de mutación en el algoritmo genético tradicional es bastante simple. Consiste en intercambiar con aleatoriedad dos genes de los cromosomas obtenidos anteriormente, En el problema el proceso de mutación es un poco distinto. Debido a que es posible afectar cualquiera de los dos cromosomas que conforman una alternativa de solución.

La mutación consiste en escoger por ruleta una de las secciones del hijo seleccionado. Se da mayor probabilidad a las secciones de peor calidad y se modifica la sección escogida siguiendo las siguientes estrategias:

1) Modificar el vector de piezas original haciendo permutaciones suaves usando el operador de mutación tradicional. Es decir, escogiendo de forma aleatoria dos elementos del vector, de la sección ganadora en la ruleta, e intercambiándolos entre si.

2) Modificar el vector de piezas haciendo permutaciones, escogiendo de forma aleatoria dos niveles del vector, de la sección ganadora en la ruleta, e intercambiándolos entre si.

De los descendientes anteriores, se escoge uno usando selección proporcional (ruleta) dando mayor probabilidad a las alternativas con mejor valor de función objetivo, incluyendo la alternativa original. [2]

OPERADORES GENÉTICOS

Se describen los operadores de recombinación y mutación aplicados en estos algoritmos. Por un lado, la recombinación de dos soluciones tentativas deberá permitir intensificar la búsqueda en las regiones del espacio de búsqueda definida por los dos padres (operador

binario). Por el otro lado, la mutación deberá permitir la diversificación de una solución (operador unario) para poder escapar de un óptimo local y también introducir diversidad genética en la población. Esta última se puede perder gradualmente a medida que el algoritmo converge durante la búsqueda.
Se presentan varios operadores de recombinación clásicos para permutaciones, incluyendo la descripción completa de un operador de recombinación que incluye información del problema en su procedimiento.

OPERADORES DE RECOMBINACIÓN

Se describe uno de los cinco operadores de recombinación estudiados comúnmente en este tipo de problemas. Cuatro de estos operadores se centran en combinar información de orden o adyacencia presente en los padres. Operadores como *Partial Mapped Crossover* (PMX), *Order Crossover* (OX) y *Cycle Crossover* (CX) consideran la posición y orden de las piezas como opuesto ejes, es decir, enlaces entre las piezas. Mientras que la idea general del *Edge Recombination* (ER) es preservar la conexión de una pieza con otras. El principal problema de estos operadores es que no utilizan información sobre el problema que se está intentando resolver. Por consiguiente el quinto operador, denominado *Best Inherited Level Recombination* (BILX), incorpora conocimiento específico del problema en su mecanismo a fin de reducir el área desperdiciada.
Considerando que empaquetados parciales residentes en el interior de una solución parecen ser los bloques de construcción naturales en este problema, la intención es que la recombinación transmita estos bloques de construcción de padres a hijos.
El operador llamado BILX, propuesto para adaptarse a este problema, incorporando en su procedimiento información sobre la distribución de las piezas. Este operador transmite los mejores niveles (grupos de piezas que definen un nivel en el patrón de empaquetado) de uno de los padres a la descendencia; es decir, trasmite aquellos niveles con el menor desperdicio. De esta manera, los niveles heredados podrían mantenerse o incluso capturar algunas piezas de sus niveles vecinos, dependiendo de cuán compactos son los niveles.
El operador BILX funciona de la siguiente manera. Sea nl la cantidad de niveles en uno de los padres Padre1. En una primera etapa, se calcula el desperdicio resultante para cada uno de los niveles nl de Padre1. Luego, se asigna a cada nivel un valor de probabilidad de selección, inversamente proporcional a sus valores de desperdicio. Con estas probabilidades se seleccionan nl/2 niveles de Padre1 empleando selección proporcional al desperdicio. Las piezas πi pertenecientes a los niveles seleccionados se colocan en las primeras posiciones del hijo. Mientras que las posiciones restantes se llenan con las piezas que no pertenecen a esos niveles, en el orden en que aparecen en el otro padre Padre2. [3] Suele ser necesario un procedimiento de reparación a fin de garantizar la factibilidad del hijo generado.

OPERADORES DE MUTACIÓN

Se describe los cuatro operadores de mutación estudiados y que son específicos para resolver los problemas de corte y empaquetado. [9] La mutación permite preservar la diversidad

genética en sistemas biológicos, pero desde el punto de vista de optimización, no permite que la población tenga una convergencia prematura hacia un óptimo local.
El más simple de los operadores es intercambio de piezas *Piece Exchange* (PE), el cual selecciona dos piezas en forma aleatoria en un cromosoma e intercambia sus posiciones.
Otro operador similar a PE es intercambio de niveles *Level Exchange* (SE), pero en este caso considerando niveles en su procedimiento. Para ello elige dos niveles del patrón de empaquetado en forma aleatoria y los intercambia en el cromosoma.
Otra de las mutaciones propuestas es intercambio del mejor y peor nivel *Best and Worst Level Exchange* (BWSE), las piezas del mejor nivel, aquel con el menor desperdicio, son reubicadas en las primeras posiciones del cromosoma, mientras que los piezas del peor nivel se mueven al final.
Finalmente, la mutación reubicar las piezas del último nivel *Last Level Rearrange* (LLR) toma las piezas pertenecientes al último nivel del patrón de empaquetado y las trata de reacomodar en otros niveles. Para ello, toma la primer pieza πi del último nivel y analiza todos los niveles del patrón de empaquetado (siguiendo una heurística NFDH modificada) tratando de hallar un lugar para esa pieza. Si la búsqueda no es exitosa, la pieza πi no se mueve. Este proceso se repite por cada una de las piezas pertenecientes al último nivel. Durante este proceso, las posiciones de las piezas dentro del cromosoma son consistentemente reacomodadas, produciendo una reducción de la altura total de la plancha.
Los movimientos pueden ayudar a que los niveles involucrados, o sus vecinos, puedan tomar piezas de los niveles próximos, reduciendo así su desperdicio. [3]

OPERADOR DE RELOCALIZACION

Además de operadores de recombinación y mutación utilizados para resolver el problema de corte y empaquetado, aparece un nuevo operador para el problema atendiendo a sus restricciones, el operador de relocalización, su función es la de mejorar la solución obtenida tras la recombinación y la mutación.
Este operador considera las piezas en el orden dado por la permutación π. La pieza πi es asignada en el primer nivel donde exista suficiente espacio en una pila existente o en una nueva, siempre respetando las restricciones del problema. Si no se halla espacio, se crea una nueva pila para contener a πi en un nuevo nivel en la parte restante del strip. El proceso anterior se repite hasta asignar todas las piezas de π.
Por otra parte se procede de una manera similar, pero con la diferencia de que una pieza πi se asigna en el nivel (entre los que puedan acomodarla siguiendo las restricciones del problema) con el menor desperdicio, ya sea en una pila existente o en una nueva. Estos pasos se repiten hasta ubicar todas las piezas de la permutación π.
Como paso final se debe producir una reorganización adecuada del cromosoma para que refleje el nuevo patrón de empaquetado, de forma tal que se pueda llegar el mismo patrón tras aplicar la heurística NFDH (utilizada en la función de evaluación) al cromosoma. [3]

ICT-UNPA-41-2012
ISSN: 1852- 4516
Aprobado por Resolución Nro. 0348/12-R-UNPA

INICIALIZACIÓN DE LA POBLACIÓN

La inicialización de la población determina el proceso de creación de los individuos para el primer ciclo del algoritmo. Normalmente, la población inicial se forma de individuos creados con aleatoriedad. El rendimiento de un algoritmo genético es a menudo relacionado con la calidad de esa población inicial. La calidad está dada en dos medidas importantes: la aptitud promedio de los individuos y la diversidad en la población. Al tener una población inicial con mejores valores de fitness, se pueden encontrar con mayor rapidez mejores individuos finales. [10,11,12] Hay muchas formas de manejar esta diversidad inicial. La propuesta que se sigue es comenzar con una población sembrada con cromosomas buenos, denominados semillas, creados siguiendo algunas reglas simples de construcción, propuestas para el problema concreto que se está abordando. Es de esperar que esta incorporación de semillas a la población inicial nos permita obtener buenos patrones de empaquetado en las primeras etapas de la búsqueda, es decir, se logre una evolución más rápida. Estas reglas incluyen conocimiento del problema, como dimensiones de las piezas, y también incorporan ideas de las heurísticas *Best Fit Decreasing Height* (BFDH) y *First Fit Decreasing Height* (FFDH).
Con esta propuesta, los individuos de la población inicial se generan en dos pasos. En el primero, los cromosomas se producen por un muestreo aleatorio del espacio de búsqueda siguiendo una distribución uniforme. Luego, cada cromosoma se somete a un paso de modificación al aplicar alguna de las reglas de construcción que describiremos a continuación, con el objetivo de mejorar la ubicación de las piezas en el patrón de empaquetado correspondiente.

Reglas de construcción para generar la población inicial.

1) Ordena en forma decreciente las piezas considerando el ancho.
2) Ordena en forma creciente las piezas considerando el ancho.
3) Ordena en forma decreciente las piezas considerando la altura.
4) Ordena en forma creciente las piezas considerando la altura.
5) Ordena en forma decreciente las piezas considerando el área.
6) Ordena en forma creciente las piezas considerando el área.
7) Ordena las piezas en forma decreciente considerando en forma alternada anchos y alturas.
8) Ordena las piezas alternando entre anchos decrecientes y alturas crecientes.
9) Ordena las piezas en forma creciente considerando en forma alternada anchos y alturas.
10) Ordena las piezas alternando entre anchos crecientes y alturas decrecientes.
11) Reorganiza las piezas siguiendo una heurística BFDH.
12) Reorganiza las piezas siguiendo una heurística FFDH.

El objetivo de estas reglas es producir individuos con buenos valores de fitness y también introducir diversidad en la población inicial. De esta manera, ordenar las piezas considerando su ancho (Reglas 1 y 2) deberá incrementar la probabilidad de apilar piezas y, en consecuencia, producir niveles más densos. Por otra parte, ordenar las piezas por altura (Reglas 3 y 4) generará niveles con menor desperdicio de espacio en cada una de las pilas de esos niveles, especialmente cuando las alturas de las piezas son muy similares. Las reglas 11 y 12 reubicarán las piezas con el objetivo de reducir el espacio desperdiciado en cada uno de los niveles. Finalmente, las reglas comprendidas entre la 7 y la 10 han sido introducidas para

incrementar a la diversidad inicial. De esta manera, las reglas propuestas no sólo son útiles para generar la población inicial, sino que varias de ellas se pueden usar como simples algoritmos voraces para realizar una búsqueda local durante el proceso de optimización. El operador relocalización es un ejemplo basado en la regla 11 y 12. [3]

2.2 RESULTADOS ANALISIS Y DISCUSIÓN.

2.2.1 MATERIALES Y MÉTODO.

Se utilizaron para la investigación y el desarrollo de la herramienta (Figura 1), IDE NetBeans 6.9.1 Platform are basado en software de netbeans.org, con licencia bajo the Common Development and Distribution License (CDDL) and the GNU General Public License version2. Con la utilización de JAVA como lenguaje de programación por su portabilidad y la amplia utilización en el área académica.
La herramienta lograda en la investigación esta en una etapa de desarrollo, porque no cuenta con todas sus funcionalidades habilitadas, ya que su utilización es carácter educativo y de aprendizaje, para la demostración de la un aplicación real de los *algoritmos genéticos*, por esto la limitación de siempre trabajar con las mismas piezas sin poder agregar, eliminar ni modificar las mismas.

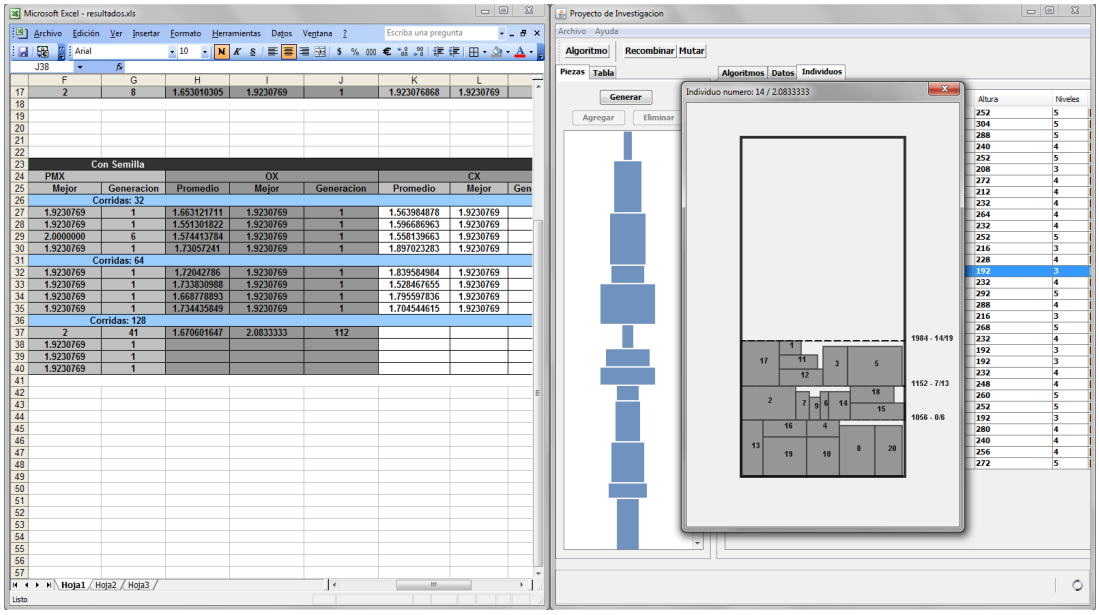

Figura 1. Desarrollo de la aplicaron con de Resultados.

DESCRIPCIÓN DEL ALGORITMO

El algoritmo genético que utilizamos en este trabajo para resolver el corte de guillotina. La estructura genérica del algoritmo se presenta en el *Algoritmo*. Este algoritmo crea una población inicial P (0) de μ soluciones de una manera aleatoria (uniforme). Luego evalúa esas soluciones; para lo cual se usa un algoritmo de asignación que acomoda las piezas en la plancha de material para construir un patrón de empaquetado factible. A continuación, la población pasa por un ciclo, denominado evolución. Este ciclo consiste en la selección de dos padres por torneo binario y la aplicación de algunos operadores genéticos para crear la nueva solución (hijo).
El nuevo individuo generado reemplaza al peor de la población sólo si es mejor. El criterio de parada para el ciclo es alcanzar un número máximo de evaluaciones. La mejor solución es identificada como el mejor individuo hallado, el cual presenta la mínima altura de material necesaria para empaquetar todas las piezas. [3]

Algoritmo pseudocódigo del algoritmo genético utilizado

```
t = 0; {generacion actual}
inicializar(P(t));
evaluar(P(t));
while (t < max evaluaciones) do
        padres = seleccionarPadres(P(t));
        hijo = aplicarRecombinacion(padres);
        hijo = aplicarMutacion(hijo);
        evaluar(hijo);
        P(t + 1) = seleccionar(P(t), {hijo});
        t = t + 1;
end while
retornar la mejor solución de P(t)
```

REPRESENTACIÓN

En la mayoría de los casos, aplicar un algoritmo genético significa desarrollar una representación especializada para codificar una solución candidata para un problema dado. Una búsqueda efectiva deberá dificultarse si se busca por las coordenadas x e y de cada elemento, ya que la cantidad de soluciones es inmensurable y no es sencillo eliminar la superposición de piezas. Para sobrellevar esta dificultad, se han propuesto varios esquemas de codificación y muchos algoritmos para problemas de empaquetado regular están basados en esos esquemas de codificación. Uno de los esquemas más populares es representar la solución por una permutación de M elementos, donde una solución codificada (es decir, una permutación de M elementos) especifica el orden en el que se asignarán las piezas. La cantidad de posibles soluciones es O (M!), la cual es más pequeña que para otros esquemas de codificación. Cada permutación corresponde a una distribución sin solapamiento de elementos. Un algoritmo de descodificación calcula la distribución de una solución codificada al especificar las ubicaciones de los elementos uno a uno, el cual define la

traslación entre la solución codificada a patrones de empaquetado (distribución de los elementos).
Un cromosoma es una permutación π = (π1, π2, ..., πM) de M números naturales, donde cada número representa los identificadores de las piezas (Figura 2). Como ejemplo consideremos el siguiente problema con 20 piezas (M = 20) y una plancha de material de ancho W y altura ilimitado.
Un posible cromosoma es π = (6,20,8,4,5,9,12,15,1,3,10,11,13,17,18,2,19,14,7,16), donde 8 indica la pieza 8, 4 la pieza 4, 5 la pieza 5, y así sucesivamente. El cromosoma da un orden para considerar las piezas, es cual es luego usado por el algoritmo de descodificación (también llamado de asignación), representado por la función de evaluación. Este algoritmo dispondrá las piezas en la tira de material (también denominada plancha) para construir el patrón de empaquetado. Algoritmo genético genera la mejor permutación posible para que el algoritmo de asignación halle un patrón de empaquetado óptimo, el cual minimiza la altura de la plancha de material requerida.

Pieza	X	Y	ID
0	12	40	6
1	40	72	20
2	52	72	8
3	48	24	4
4	80	56	5
5	16	32	9
6	64	24	12
7	80	24	15
8	32	16	1
9	40	56	3
10	48	56	10
11	56	16	11
12	32	80	13
13	56	64	17
14	64	24	18
15	80	48	2
16	64	56	19
17	32	40	14
18	24	40	7
19	64	24	16

Figura 2. Tabla de piezas.

El patrón respeta las restricciones guillotina y de empaquetado en tres etapas. En la primera etapa se realizan cortes horizontales para obtener los distintos niveles. En la siguiente etapa los cortes son verticales para obtenerse las pilas y finalmente en la tercera etapa se efectúan cortes horizontales para obtener las piezas y el área o material de desperdicio. [3]

FUNCIÓN DE EVALUACIÓN

Los algoritmos genéticos son guiados por los valores calculados por una función objetivo (fitness) para cada solución tentativa, hasta que se halla un óptimo o una solución aceptable. En este problema, el objetivo es minimizar la altura de la plancha, necesaria para construir el patrón de empaquetado correspondiente a una solución π dada.
Para determinar la calidad de una solución, primero es necesario obtener la distribución de las piezas. A fin de generar un patrón por niveles en tres etapas adoptamos una heurística *Next Fit Decreasing Height (*NFDH) modificada, la cual ha probado ser un poco más eficiente. [8,9]
Esta heurística, dada una secuencia de piezas ordenadas como entrada (una permutación de

piezas), construye el patrón de empaquetado al colocar piezas en pilas (piezas vecinas con el mismo ancho se apilan una sobre la otra) y luego pilas en niveles paralelos a la base de la plancha. Las pilas se justifican hacia el borde izquierdo. Una vez que se crea una nueva pila o nivel, los anteriores no se reconsideran. El *Algoritmo* incluye una descripción de alto nivel de este algoritmo.
El algoritmo comienza con la primer pieza π1 del vector π = (π1, π2, ..., πM). Las variables que utiliza son: strip (representa la plancha), l (referencia al nivel actual) y s (hace referencia a la pila actual)

Algoritmo proceso de asignación de la heurística NFDH Modificada
NFDH(π: vector, M:integer, W:integer)

```
iniciar contadores:
i = 1; {pieza actual}
iniciar strip:
strip.alto = 0;
while (i ≤M) do
        iniciar nivel(π i, l);
        strip.alto = strip.alto + l.alto;
        while (i ≤ M and factible π i en l) do
                iniciar pila(π i, s, l);
                while (i ≤M and factible π i en s) do
                        push pila(πi, s, l);
                        s.alto = s.alto + altura(π i);
                        l.desperdicio = l.desperdicio - área(π i);
                        i = i + 1;
                        end while
        end while
end while
setear l para ser el último nivel;
return strip.alto, l;

iniciar nivel(π i:pieza, l: nivel)
l.alto = altura(πi);
l.anchoRest = W;
l.desperdicio = strip.area;

iniciar pila(π i:pieza s: pila, l:level)
s.ancho = ancho(π i);
s.alto = 0;
l.anchoRest = l.anchoRest − ancho(π i);

push pila(π i :pieza, s: pila, l: nivel)
if l.alto < s.alto + altura(π i) then
        desperdicio = l.desperdicio + (s.alto + alto(π i) − l.alto) ×W;
        if (desperdicio−área(π i) ≤ l.desperdicio) then
        l.desperdicio = desperdicio;
        l.alto = s.alto + altura(π i);
        strip.alto = strip.alto + (s.alto + altura(π i) − l.alto);
        end if
end if
```

Tanto el strip como el primer nivel se inicializan con la altura de π1 (l.alto = altura (π1)), y la primera pila de ese nivel con el ancho de π1 (s.ancho = ancho (π1)). Una vez creado un nivel, la próxima pieza, πi, de π comienza la segunda pila del nivel actual si se cumplen las siguientes condiciones: (a) la altura (πi) no excede la altura del nivel, l.alto, y (b) el ancho (πi) no excede el ancho restante del nivel, l.anchoRest. Si la siguiente pieza j tiene ancho igual a ancho (πi), entonces son apiladas una sobre la otra si la altura de la pila resultante, s.alto, no excede l.alto. En el caso de que la próxima pieza πj difiere de ancho (πi) o s.alto excede l.alto, entonces se comienza una nueva pila con πj si ancho (πj) ≤ l.anchoRest; en caso contrario, se comienza un nuevo nivel con πj . Este proceso se repite hasta empaquetar todas las piezas del vector π (Figura 3).

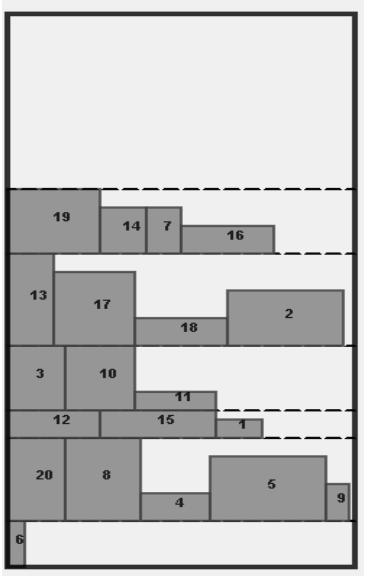

Figura 3. Primer proceso de asignación.

Esta versión de NFDH modificada permite extender la altura del nivel si se produce un decremento en el desperdicio total del nivel en las siguientes situaciones: en el caso de apilar piezas y que la altura de la pila s.alto resultante exceda la altura del nivel o cuando la altura de una pieza en una nueva pila sea más alta que el nivel l.alto.
La nueva pieza puede ser ubicada en el nivel si el área de la nueva pieza es mayor o igual al área del desperdicio producido al alargar el.
Como ya hemos mencionado, el fitness de una solución π se define como la altura necesaria de la plancha para construir el correspondiente patrón de empaquetado, definido como la suma de las alturas de los niveles generados. Es importante considerar que dos patrones de empaquetado podrían tener el mismo alto (por lo tanto sus fitness resultarían iguales) aunque, desde el punto de vista de reuso del desperdicio, uno de ellos puede ser mejor porque el desperdicio en el último nivel (el cual se conecta con el resto del strip) es mayor que el presente en el último nivel del otro patrón de empaquetado. Los patrones de empaquetado que dejan la parte más grande sin uso en el último nivel son más prometedores.

Función de fitness

$F(\pi) = strip.alto - (l.desperdicio/l.alto \times W)$

Donde strip.alto es la altura del patrón de empaquetado correspondiente a la permutación π, l.desperdicio es el área del desperdicio reusable en el último nivel l, l.alto es la altura del último nivel y W es el ancho de la plancha. El segundo operando de la ecuación indica la proporción de área desperdiciada en el último nivel respecto del área del último nivel (strip.alto ×W). [3]

2.2.2 RESULTADOS.

Se ha trabajado bajo el desarrollo realizado, analizando el comportamiento de los algoritmos genéticos para un problema en particular. Con las mismas piezas en todos los casos (Figura 1), Con los operadores de combinación (BLIX, PMX, OX, CX) y de mutación (PE, SE, BWSE, LLR) con y sin semilla para una población de 32 individuos con 32, 64, 128 corridas o generaciones. Incluido en operador de relocalización en evolución de cada individuo. Obteniendo así rápidamente los valores cercanos al óptimo ideal (Figura 4), Que en todos los casos fue aceptable para cada corrida en ejecución. Logrando así el menor desperdicio del material de plancha para el corte las piezas.

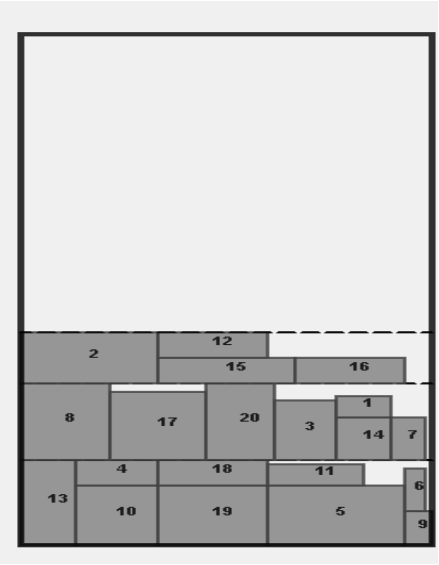

Figura 4. Individuo optimo ideal.

En los datos estadísticos (Figura 5) logrados sin semilla se pudo analizar que en promedio las recombinaciones mas adecuadas son BLIX y PMX por la utilización de conservación de bloques de construcción que se propaga a toda la población a lo largo de la evolución del algoritmo. Mientras que OX y CX si bien han obtenido valores importantes como individuo pero con la realización de varias corridas, sin impactar en promedio general, esto también se

puede deber al proceso de selección, ya que hay una parte de esta que se realiza por azar. No así si este proceso se realizaría por selección por torneo, que utilizaría un proceso más elitista para la selección de individuos en el momento de seleccionar dos padres en la evolución de la población.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		BLIX			PMX			OX			CX		
2		Promedio	Mejor	Generacion	Promedio	Mejor	Generacion	Promedio	Mejor	Generacion	Promedio	Mejor	Generacion
3	Corridas: 32												
4	PE	1.7101870067	1.8518518	1	1.645270154	1.9230769	2	1.582972839	1.8518518	12	1.637863766	1.8518518	4
5	SE	1.7831595354	1.9230769	3	1.621186357	1.9230769	14	1.575144097	1.9230769	1	1.521758545	1.9230769	1
6	BWSE	1.7303547449	1.9230769	1	1.62108282	1.8518518	9	1.599278409	1.9230769	5	1.439661954	1.9230769	5
7	LLR	1.720910132	1.7857143	1	1.680202156	1.8518518	1	1.692955293	1.9230769	6	1.439084664	1.9230769	19
8	Corridas: 64												
9	PE	1.916860025	1.9230769	10	1.629091524	1.9230769	5	1.647692349	1.9230769	7	1.8098565	1.9230769	47
10	SE	1.851851821	1.8518518	4	1.642360132	1.9230769	4	1.636927776	1.9230769	8	1.59620915	1.9230769	1
11	BWSE	1.534562536	1.8518518	1	1.900636274	1.9230769	3	1.655395355	1.9230769	7	1.51928068	1.9230769	1
12	LLR	1.923076868	1.9230769	3	1.851851821	1.8518518	11	1.580890924	1.9230769	1	1.78571427	1.7857143	1
13	Corridas: 128												
14	PE	1.774913579	1.7857143	1	1.880939573	1.9230769	5	1.56576933	1.9230769	35	1.802908886	1.9230769	3
15	SE	1.775234882	2	120	1.778520554	1.9230769	4	1.587505233	2	50	1.560367517	2	79
16	BWSE	1.618303582	1.7857143	1	2	2	10	1.630970187	1.9230769	16	1.923076868	1.9230769	18
17	LLR	1.736571588	1.9230769	1	2	2	8	1.653010305	1.9230769	1	1.923076868	1.9230769	1
18													

Figura 5. Resultados de fitness obtenidos, sin semilla de inicio

En base a la cantidad de corridas o generaciones de población se pude observar que en promedio de la población tiende a mejorar su fitness en una alta cantidad de corridas, mejorando así el promedio total de la muestra.
En las estadísticas (Figura 6) logradas con la iniciación con semilla se pudo alcanzar un optimo aceptable rápidamente ya que esta población inicial se prepara de dista forma ante la asignación de las piezas, esto también perjudica en la diversidad de la población, ya que solo se acota en pocas solucione. Pero también en promedio las recombinaciones mas adecuadas son BLIX y PMX por la utilización de conservación de bloques de construcción que se propaga a toda la población a lo largo de la evolución del algoritmo. Mientras que OX y CX en promedio han obtenido mejores valores que la estadística anterior.

	Con Semilla											
	BLIX			PMX			OX			CX		
	Promedio	Mejor	Generacion	Promedio	Mejor	Generacion	Promedio	Mejor	Generacion	Promedio	Mejor	Generacion
Corridas: 32												
PE	1.9026303627	1.9230769	1	1.781327542	1.9230769	1	1.663121711	1.9230769	1	1.563384878	1.9230769	1
SE	1.9230768681	1.9230769	1	1.654113151	1.9230769	1	1.551301822	1.9230769	1	1.596686963	1.9230769	1
BWSE	1.9230768681	1.9230769	1	1.786738101	2.0000000	6	1.574413784	1.9230769	1	1.558139663	1.9230769	1
LLR	1.9230768681	1.9230769	1	1.900506198	1.9230769	1	1.73057241	1.9230769	1	1.897023283	1.9230769	1
Corridas: 64												
PE	1.750275385	1.9230769	1	1.812664133	1.9230769	1	1.72042786	1.9230769	1	1.839584984	1.9230769	1
SE	1.923076868	1.9230769	1	1.853128828	1.9230769	1	1.733830988	1.9230769	1	1.528467655	1.9230769	1
BWSE	1.923076868	1.9230769	1	1.708382241	1.9230769	1	1.668778893	1.9230769	1	1.795597836	1.9230769	1
LLR	1.923076868	1.9230769	1	1.923076868	1.9230769	1	1.734435849	1.9230769	1	1.704544615	1.9230769	1
Corridas: 128												
PE	1.906676363	1.9230769	1	1.868804246	2	41	1.670601647	2.0833333	112	1.651359521	1.9230769	1
SE	1.923076868	1.9230769	1	1.923076868	1.9230769	1	1.669839337	1.9230769	1	1.575326003	1.9230769	1
BWSE	1.923076868	1.9230769	1	1.479932856	1.9230769	1	1.732745171	2.0833333	100	1.469490338	1.9230769	1
LLR	1.736571588	1.9230769	1	1.923076868	1.9230769	1	1.736310445	1.9230769	1	1.923076868	1.9230769	1

Figura 6. Resultados obtenidos, con semilla de inicio

En base a los tipos de mutación podemos observar que pueden ser mejor aprovechadas a lo largo de las generaciones logrando así diversificar las soluciones y no llegar al estancamiento de las mismas.
En esta diversificación se pudo obtener el óptimo ideal ya que en la mayoría de los casos se cierra en la solución del el óptimo aceptable.

3. CONCLUSIÓN.

Se ha investigado el comportamiento del algoritmo genético para resolver un problema de corte y empaquetado con restricciones. Se ha buscado operadores específicos del problema comparado con operadores tradicionales. Se analizo distintos métodos de generación de la población inicial a fin de generar individuos potencialmente prometedores y la capacidad de mantener una población diversificada al usar una población inicial mejorada, con respecto a la eficiencia (esfuerzo) y la calidad de las soluciones encontradas. Proporciona una buena diversidad genética de las soluciones iniciales y también una convergencia más rápida sin caer en un óptimo local. Esto demuestra que la ejecución de un algoritmo genético es sensible a la calidad de sus poblaciones iniciales.
Por otra parte, la incorporación del operador de relocalización en el proceso evolutivo de los algoritmos propuestos, proporcionando un muestreo más rápido del espacio de búsqueda en todas las instancias analizadas y exhibiendo una relación entre exploración y explotación satisfactoria, lo cual marca su influencia para obtener buenos patrones de empaquetado, promoviendo una intensificación en ciertas regiones prometedoras del espacio de búsqueda, que será aprovechada en generaciones posteriores por el algoritmo.

4. AGRADECIMIENTOS.

Se reconoce a la UNPA-UACO por su apoyo a la iniciación de la investigación, la cooperación, y criticas constructivas proporcionadas por las mimas.

5. REFERENCIAS.

[1] Gómez, Alberto, De la fuente, David y Priore, Paolo. 2000. Resolución del Problema de Strip-Packing Mediante la Metaheurística Algoritmos Genéticos. Universidad de Oviedo. Paper. p1-6.
[2] Eliana Mirledy Toro y Mauricio Granada Echeverri. 2007. Problema de Empaquetamiento Rectangular Bidimensional Tipo Guillotina Resuelto por Algoritmos Genéticos. Universidad Tecnológica de Pereira. Scientia et Technica, ISSN 0122-1701. p321-326.
[3] Carolina Salto. 2009. Metaheurísticas Hibridas Paralelas para Problemas Industriales de Corte, Empaquetado y otros Relacionados. Universidad Nacional de San Luís. Tesis Doctoral.
[4] S. Jakobs. 1996. Theory And Methodology On Genetic Algorithms For The Packing Of Polygons. European Journal Of Operational Research. Vol.88, p87-100.
[5] F. Parreño. 2004. Algoritmos Heurísticos Y Exactos Para Problemas De Corte No Guillotina En Dos Dimensiones. Universidad De Valencia. Servei De Publicacions. Tesis Doctoral.
[6] T.W. Leung , Chi Kin Chan, M. Troutt. 2003. Application Of A Mixed Simulated Annealing-Genetic Algorithm Heuristic For The Two-Dimensional Orthogonal Parking Problem. European Journal Of Operational Research. 145. p530–542.
[7] Vendramini E. 2007. Optimización del problema de cargamento de contenedores usando una metaheurísticas eficiente. UNESP. Tesis Doctoral.
[8] J. Puchinger and G. Raidl. 2004. An evolutionary algorithm for column generation in integer programming: An effective approach for 2D bin packing. PPSN. p642–651
[9] C. Salto, J.M. Molina, and E. Alba. 2006. Evolutionary algorithms for the level strip parking problem. Proceedings of NICSO. p137–148.
[10] R. Ahuja, J. Orlin, and A. Tiwari. 2000. A greedy genetic algorithm for the quadratic assignment problem. Computers & Operations Research. p917–934.
[11] C.R. Reeves. 1995. A genetic algorithm for flowshop sequencing. Computers & Operations Research. p5–13.
[12] Ahuja and J. Orlin. 1997. Developing fitter genetic algorithms. INFORMS Journal on Computing. p251–253.

6. ANEXOS.

- CD con el ejecutable (tipo .jar) de la herramienta desarrollada en lenguaje JAVA.
- Documento JAVADOC con la información correspondiente al modelo del código fuente de programa.
- Publicación interna sobre manejo de restricciones con algoritmos evolutivos aplicados al problema de optimización de corte y empaquetado.