UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Optimización del trazado de patrones de corte

Miguel Ángel Luque López
Tutor: José Dorronsoro Ibero

JUNIO 2021

Optimización del trazado de patrones de corte

AUTOR: Miguel Ángel Luque López
TUTOR: José Dorronsoro Ibero

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2021

Resumen (castellano)

Este Trabajo de Fin de Grado pretende considerar soluciones algorítmicas al problema de optimizar la producción de piezas bidimensionales de un determinado material recortadas dentro de una cadena de producción, de tal manera que se minimice la cantidad de material desperdiciado en el proceso de corte. El objetivo básico es minimizar el área utilizada para recortar estas piezas bajo, entre otros, los siguientes supuestos

- Se considerará un material madre donde efectuar los recortes de forma rectangular con una anchura prefijada y longitud ilimitada.
- Se considerará material desperdiciado todo aquel que se encuentre dentro de la longitud mínima del material madre que contenga todas las piezas y no se encuentre encerrado en el área de alguna pieza.
- Se considerará una colocación correcta de las piezas de manera que todas estén dentro del material madre, sin colisiones entre ellas y sin que una esté contenida en el interior de ninguna de las demás.
- Se consideran piezas delimitadas por rectas y curvas relativamente simples.

Un algoritmo debe aportar un número determinado de piezas dispuestas para su colocación en un determinado orden en el material madre, y su eficacia se medirá tanto por el porcentaje de área no desperdiciada como por el tiempo necesario para obtener su colocación. Para ello se divide el problema en tres fases: selección de pieza a introducir, proceso de colocación de dicha pieza y control de sus posibles colisiones. Todo ello ha de tener en cuenta las piezas ya colocadas y su posición, repitiéndose el proceso de colocación probando tal vez diferentes criterios de ordenación que mejoren el resultado hasta un límite de iteraciones previamente fijado.

Entre las técnicas que se vienen empleando están el método de colocación Bottom-left mediante vectores corredizos, las estrategias de prevención de colisión mediante cuadro delimitador y aproximación rectangular, así como el algoritmo NEAT, basado en técnicas evolutivas genéticas. Las mismas se considerarán en el trabajo, con un énfasis en el algoritmo NEAT.

Palabras clave (castellano)

Patrones de corte, modelo de Gilmore y Gomory, NEAT.

Abstract (English)

This Project's objective is to review the existent ways of solving the marker making problem and considering a new algorithm purposed for its solution taking in account the following conditions:

- It will be considered as "mother" material that one in which the patterns will be marked; this material will have a fixed width and an infinite length.
- It will be considered as wasted material all the area within the mother material which being into the minimal length of the material which is needed to contain all the pieces is not contained by any marker.
- A colocation will be considered correct only if all the markers are into the mother material without any collisions between them and there is no marker into any other marker.
- The considered pieces will be irregular and made of straight segments and relatively simple curved ones.

An algorithm must give a colocation proposal for the pieces to put in the mother material. Its effectiveness will be measured both by the percentage of area properly used and the necessary time for the algorithm to give a solution. To archive the solution, the problem is divided in three phases: selection of a piece to input, colocation strategy of the piece and collision control of the piece. For those phases to work it is important to consider the already allocated pieces and their position, since the algorithm will allocate them sequentially, repeating the allocation process of the pieces trying different order criteria for the allocation.

Among the used techniques the chosen one will be Bottom-left with the use of sliding vectors, collision prevention strategies with the bounding box and rectangular approximation as well as the NEAT algorithm, based on genetic algorithms which are considered in this document with special emphasis on NEAT.

Keywords (inglés)

Marker making, Gilmore and Gomory model, NEAT.

Agradecimientos

Dedico un especial reconocimiento a mi tutor José Dorronsoro por su confianza y apoyo a lo largo del proyecto, ofreciendo una ayuda sin el cual este no habría sido el mismo.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

En este capítulo se plantea el problema a resolver que nos lleva a escribir este trabajo de fin de grado y los motivos por los cuales su resolución es prioritaria. Se aporta también una breve descripción de los contenidos de cada capítulo del documento.

1.1 Motivación

El gran crecimiento de la población y, en consecuencia, de la demanda de manufacturas desde la revolución industrial, han llevado a la necesidad de encontrar nuevos métodos para abaratar la producción. De forma tradicional esto requiere un aumento de la producción, pero en las condiciones medioambientales actuales, cada vez menos capaces de mantener la economía de consumo, es importante utilizar de la forma más eficiente posible la materia prima de la que se dispone. Esto no solo es una necesidad medioambiental sino además un incentivo económico, teniendo en cuenta que el material utilizado en una manufactura representa un gran porcentaje del precio del producto; por ejemplo, en la industria textil la tela utilizada representa un 50 % del coste de la prenda.

En la industria textil se desperdicia aproximadamente un 15 % de la tela al recortar las prendas; esto representa un gran costo tanto medioambiental como económico, el cual es necesario reducir o en un caso ideal suprimir. Si bien es cierto que desde hace un tiempo existen métodos informáticos para optimizar estas pérdidas, también lo es que han surgido nuevas posibilidades, en especial dentro del campo de la inteligencia artificial, que reabren la posibilidad de enfrentarse a este problema abriendo la puerta a una posible mejora que podría abaratar la producción industrial; es importante recordar que cada pequeño porcentaje de material ahorrado implica un ahorro de potencialmente millones de euros y a su vez de su proporción cuantificable de energía necesaria para fundir planchas de aluminio, hilar rollos de tela o recortar láminas de papel. [1]

1.2 Objetivos

El objetivo de este trabajo de fin de grado es realizar un análisis del problema de optimización del trazado de patrones de corte, revisando los métodos usados para formalizar y resolver este problema, dando una especial relevancia a aquellos orientados a la resolución con piezas irregulares. Esto se hará recurriendo también a versiones más simples del problema y a medios utilizados comúnmente en virtud de reducir la complejidad del problema a nivel computacional. Antes de analizar estos métodos es conveniente explicar en qué consiste el problema que tratamos de optimizar.

El problema de optimización del trazado de patrones de corte es un problema clásico que busca distribuir piezas por un plano en dos dimensiones sin que estas colisionen ni se posicionen unas dentro de otras, ocupando el mínimo espacio posible. Este espacio a minimizar se mide normalmente en el número de piezas de material madre utilizado para colocar un conjunto de piezas; sin embargo, para este caso se busca minimizar el área que ocupen en una única pieza de material madre, rectangular, con uno de sus ejes de longitud ilimitada y el otro de una longitud prefijada.

Tras el análisis de los métodos existentes se propone una modificación del método de resolución del problema por medio de un algoritmo genético, utilizando en esta propuesta en su lugar el algoritmo NEAT para la ordenación de las piezas para finalmente comparar

experimentalmente las posibles mejoras que este cambio puede ofrecer con respecto al algoritmo original.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

Capítulo 2 Estado del Arte

- **Programación lineal como método:** En primer lugar, se analiza la formalización del problema, mostrando el problema para una única dimensión, los cambios al entrar a la segunda y el método de resolución de Gilmore y Gomory, pasando por las estructuras y algoritmos usados para la construcción de los patrones de corte.
- **Simplificación del problema:** Se analizan los métodos utilizados para simplificar los cálculos a la hora de enfrentar el problema, con ejemplos como el cuadro delimitador mínimo, la aproximación rectangular de las piezas o los vectores corredizos.
- **Detección de colisión: S**e analizan los métodos usados para la detección de colisión entre las piezas además de la trigonometría directa, abriendo las opciones a otras formas, en ocasiones mejores, de comprobar si los patrones se chocan.
- **Método de colocación:** Se analizan algunos de los métodos y heurísticas utilizados para la colocación de un conjunto de piezas en un orden dado tratando de ocupar el mínimo espacio posible.
- **Orden de colocación:** Se analizan los métodos y heurísticas utilizados para optimizar el método de colocación por medio de una elección adecuada del orden de colocación de las piezas.

Capítulo 3 Diseño

- **El algoritmo NEAT:** En primer lugar, se analizan las características y el funcionamiento del algoritmo NEAT y a continuación se analiza la configuración utilizada para el algoritmo propuesto para la optimización del trazado de patrones de corte.
- **Descripción de las piezas:** Se analiza la composición de las piezas como estructura utilizada a lo largo del algoritmo y los motivos por los que se construyen de esta manera.

Capítulo 4 Desarrollo

- **Configuración inicial:** Se analizan las maniobras a realizar antes de comenzar la ejecución del algoritmo de colocación para obtener resultados verosímiles.
- **Aplicación del algoritmo:** Se analizan las decisiones tomadas con respecto al método de colocación y se ve en detalle la forma en la que se aplican.
- **Medición de éxito:** Se discute el ratio utilizado para medir el éxito de un algoritmo de optimización del trazado de patrones de corte y se explica el método por el cual se mide este éxito.

Capítulo 5 Integración, pruebas y resultados

- **Pruebas con piezas simples:** Se ejecuta el prototipo desarrollado en sus diferentes versiones para un conjunto de piezas simples e irregulares que permite observar su proceso de aprendizaje.

- **Pruebas con piezas complejas:** Se ejecuta el prototipo desarrollado en sus diferentes versiones para un segundo conjunto de piezas relativamente complejas que además suponen un ejemplo realista de colocación en la industria.

Capítulo 6 Conclusiones y trabajo futuro

- **Conclusiones:** Se analizan los resultados obtenidos al probar el prototipo en conjunto con la experiencia adquirida de la investigación para el trabajo de fin de grado formando un análisis de los resultados.
- **Trabajo futuro:** Se plantean cambios y mejoras a añadir en un futuro dentro del prototipo desarrollado.

2 Estado del arte

En este apartado se analizan los métodos existentes para resolver y simplificar el problema de optimización del trazado de patrones de corte, viendo las ventajas e inconvenientes para cada método y explicando para qué versiones del problema son usados. Para solucionar este problema existen multitud de enfoques posibles; en este documento se recogen solo los más comunes y de entre estos los que se representan a la hora de enfrentar el problema de optimización de trazos de corte o para versiones más simples del mismo.

2.1 Programación lineal como método

Este apartado nos muestra cómo se lleva a cabo la formalización y resolución del problema mediante programación lineal. Es importante destacar que esta forma de resolución no contempla piezas irregulares como lo hace el problema que se trata de resolver en este documento, sino piezas rectangulares a distribuir por un plano. El algoritmo además contempla rollos de dimensiones limitadas en ambos ejes y el utilizar más de un rollo para recortar las piezas visto que estos rollos son limitados, esto no supone un inconveniente a la hora de reflejar nuestro problema si asumimos directamente las dimensiones de uno de los ejes como infinitas o, para simplificar los cálculos, lo bastante grande como para que quepan todas las piezas en una sola fila (o columna, dependiendo de cuál de los ejes se asuma infinito) [2].

2.1.1 Programación lineal en una dimensión

La resolución del problema por programación lineal puede observarse de forma más simple visto en una sola dimensión; es por eso por lo que este apartado muestra la resolución del problema llevado a una sola dimensión mediante programación lineal. Para este ejemplo se usan barras en lugar de rollos de material, y el problema trata de dividir las barras iniciales en barras más pequeñas solicitadas, intentando utilizar el mínimo número de barras iniciales para ello.

Para resolver este problema es necesario utilizar patrones de corte, elementos que se definen como vectores de piezas a recortar del material inicial. Para este ejemplo de una dimensión sería un vector que define las barras que se extraerán de cada barra inicial, utilizando enteros para definir cada longitud de barra a extraer. El número de patrones de corte multiplicado por el número de veces que se utiliza cada uno nos da el número de barras iniciales que necesitaremos para esa estrategia de corte y es una condición imprescindible para que esa estrategia sea válida que todas las sub-barras solicitadas para recortar se encuentren entre los patrones de corte utilizados. Se puede obtener la longitud de la barra inicial k desperdiciada utilizando el patrón de corte j restando a la longitud de esta barra inicial la de los elementos de longitud i contenidos en el patrón de corte.[3] Gilmore y Gomory formalizan el problema según las ecuaciones (2-1), (2-2) y (2-3), extraídas de la referencia [4], puede verse gráficamente representado en la Figura 2-1.

Siendo:

- N: número total de patrones de corte.
- M: número total de barras de diferente longitud.
- α_{ij}: número de barras de longitud i en el patrón de corte j.
- c_j: material sobrante al recortar el patrón j.
- d_i: número de barras de longitud i solicitadas.
- y_j: número de veces que se recorta el patrón de corte j.

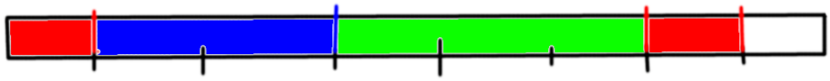
Se busca minimizar:

$$f(y) = \sum_{j=1}^{N} c_j y_j \tag{2-1}$$

Sujeto a:

$$\sum_{j=1}^{N} \alpha_{ij} y_j \geq d_i \qquad i = 1, \ldots, M \tag{2-2}$$

$$y_j \geq 0 \; y \; entero \qquad j = 1, \ldots, N \tag{2-3}$$



Las áreas coloreadas representan barras recortadas de la barra inicial

Patrón de corte 1 = {1,2,3,1}

Longitud barra inicial = 8

Material sobrante = 1

Se asume para este caso que el entero i que define el identificador de cada pieza en el patrón de corte es equivalente a la longitud de la pieza

Figura 2-1: Representación gráfica de un ejemplo de patrón de corte en el problema de optimización del trazado de patrones de corte en una dimensión.

2.1.2 Cambios al entrar en la segunda dimensión

Una vez visto el problema en una dimensión podemos añadir una segunda; para ello será necesario extender la forma en la que usamos los patrones de corte para que puedan ser utilizados en dos dimensiones, de manera que ahora se entiende un patrón de corte como un conjunto de piezas seleccionadas para su distribución en el material a lo ancho de una fila. La altura de esta fila será igual a la de la pieza más alta que se encuentre en el patrón de corte, de forma que todas las piezas dentro del patrón queden encerradas en el área de la fila.

Como en la ecuación (2-4) se expone, nuestro objetivo es minimizar el área sobrante, dividida en dos componentes:

1. La anchura sobrante al final de cada fila.
2. La largura sobrante dentro de cada fila por la diferencia de altura entre sus elementos.

Para definir la largura perdida por cada tipo de pieza s_i se usa la ecuación (2-5), estando definida como la suma de la largura perdida por las piezas de tipo i; a su vez para cada una de estas la altura perdida es la diferencia de altura con la fila que define el patrón de corte en el que se encuentren. Esta altura se representará como área perdida una vez multiplicada por la anchura de la pieza que ha perdido esa altura; mientras tanto, el área perdida por anchura

sobrante en un patrón de corte se calculará multiplicando esta anchura no utilizada c_{jk} por la altura de la fila x_{jk} que forma el patrón de corte j. Tal como describe la ecuación (2-6) es además destacable que la suma de la anchura utilizada en un patrón de corte no puede ser mayor que la anchura del rollo de forma que no pueden ser negativas ni la anchura ni la largura sobrante ni el número de piezas de un tipo determinado ni la altura de un patrón de corte, es decir, que ningún patrón de corte puede salirse del material al ser colocado.

- A: área sobrante.
- a_{ijk}: es el número de piezas de ancho w_i que se cortan con el patrón j en la pieza de material madre k.
- x_{jk}: es la altura de la fila del rollo k que es cortada según el patrón j.
- c_{jk}: es la anchura tela restante del rollo k siendo cortada según el patrón j.
- n: número de piezas de diferente anchura o largura.
- m_z: número de diferentes patrones de corte utilizados.
- h: número de rollos de material madre.
- s_i: largura perdida a causa de los huecos generados por las piezas de tipo i.
- w_i: anchura de la pieza i.
- l_i: longitud de la suma de las piezas de tipo i.

Las ecuaciones (2-4), (2-5) y (2-6) se extraen de la referencia [5] y se puede observar un ejemplo gráfico en la Figura 2-2

Podemos formalizar el problema como aquel que busca minimizar:

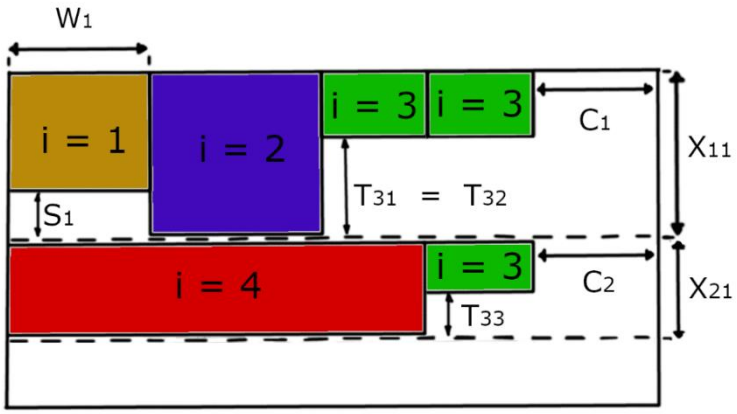
$$minimizar\ A = \sum_{k=1}^{h} \sum_{j=1}^{m_k} c_{jk} x_{jk} + \sum_{i=1}^{n} w_i s_i \tag{2-4}$$

Sujeto a:

$$\sum_{k=1}^{h} \sum_{j=1}^{m_k} a_{ijk} x_{jk} - s_i = l_i\ para\ todo\ i \tag{2-5}$$

$$x_{jk}, S_i, c_{jk}, a_{ijk} \geq 0\ para\ todo\ i, j\ y\ k \tag{2-6}$$

A continuación, se presentan dos métodos a la hora de generar estos patrones de corte: La creación de un árbol de búsqueda que genere las posibles combinaciones de elementos en cada patrón de corte y el algoritmo de generación de patrones de corte. Los patrones de corte generados no son suficiente para resolver el problema, pues debe encontrarse un equilibrio entre ellos que recorte todas las piezas necesarias dejando la menor área desperdiciada posible. Es en este punto del problema donde hará falta recurrir al modelo de Gilmore y Gomory para seleccionar la combinación de patrones de corte que resultan en una solución viable.



Rollo k = 1

$S = T_{31} + T_{32} + T_{33}$

Patrón de corte j = 1: {1,2,3,3}

Patrón de corte j = 2: {4,3}

S_i se forma con la suma de la largura desperdiciada de todas las figuras de tipo i en todos los rollos k en todos los patrones de corte j. Para este ejemplo se asumen un rollo y dos patrones de corte.

Figura 2-2: Representación gráfica del problema de optimización del trazado de patrones de corte en dos dimensiones.

Árbol de búsqueda de patrones de corte

Este algoritmo colocará las piezas en un orden secuencial y para decidir de qué forma hacerlo se define un árbol de búsqueda que ordena los posibles patrones de corte. Este árbol llena los niveles en función del número de elementos que han sido ya propuestos para ser añadidos, con orden de propuesta de más a menos ancho, para un determinado patrón de corte j; es decir, en el nivel i del árbol k se encontrarán las distribuciones de patrones de corte que corten un numero variable de 0 a a_{ijk} unidades de cada una de las i piezas revisadas. La raíz del árbol está representada con la anchura de la pieza de material madre y en cada uno de los nodos se le sustraen los correspondientes elementos recortados. De esta manera en las hojas del árbol se observa el fragmento de material desechado en la anchura y estas representan las posibles combinaciones de piezas incluidas en el patrón de corte j, siendo para este modelo m_z equivalente al número de hojas.[5][6][7]

Algoritmo de generación de patrones de corte

Este algoritmo ordena las piezas existentes por anchuras en un orden decreciente y define una matriz de corte de dimensiones m_z x n. A continuación, comienza a rellenar su primera fila, definida por el patrón de corte 1 y se le asigna el número de piezas a cortar de cada tipo (representadas en cada columna).

El criterio de colocación inicial pretende colocar las piezas más anchas hasta llegar al punto de que no quepan (o queden) más de ese tipo, como refleja la ecuación (2-8). Una vez no es posible colocar más piezas del tipo más ancho se pasa a colocar piezas de la siguiente anchura más grande siguiendo el mismo criterio, pero teniendo en cuenta las piezas ya colocadas para el patrón de corte actual; podemos ver esto reflejado en la ecuación (2-7), y este proceso se repite hasta que no quepan más piezas de ningún tipo en el primer patrón de corte. Una vez llenada la primera fila podemos determinar la cantidad de material que ha quedado sin usar (desperdiciado) para este patrón de corte usando la ecuación (2-9). Esta

fórmula a su vez también es útil para calcular el material desperdiciado del resto de patrones de corte que utilizaremos. Las ecuaciones (2-7), (2-8) y (2-9) se extraen de la referencia [5]

A la nomenclatura descrita en a comienzos del apartado 2.1.2 añadimos w'_k, que representa la anchura del rollo de material madre k

$$a_{ijk} = \text{Suelo}((w'_k - \sum_{z=1}^{i-1} a_{z1k}\, w_z)/w_i) \tag{2-7}$$

$$a_{1jk} = \text{Suelo}(\frac{w'_k}{w_i}) \tag{2-8}$$

$$c_{jk} = w'_k - \sum_{z=1}^{n} a_{zjk}\, w_z \tag{2-9}$$

Una vez llenada la primera columna se entra en el bucle de llenado de la matriz. Este bucle imita el anterior patrón de corte generado, quitando de este una de las piezas colocadas con la intención de sustituirla por piezas menos anchas aplicando de nuevo la ecuación (**2-7**) y generando en el proceso un nuevo patrón de corte diferente a todos los anteriores. Para ello empieza probando a quitar de los segundos elementos menos anchos uno a uno y cuando no le quedan que quitar pasa al siguiente elemento por orden de anchura hasta haber probado todas las combinaciones. El bucle se lleva a cabo en este orden:

- Asignar al índice i la posición del penúltimo elemento de la lista (n-1), el segundo menos ancho.
- Para la fila apuntada por j y el elemento apuntado por i: si el valor $a_{ijk} = 0$ se salta al paso 4, de lo contrario se crea una nueva fila y se apunta a ella (j = j+1) llenándola bajo los siguientes criterios:
  - A los nodos en columnas anteriores a i se les asigna el valor de su fila anterior
    - $a_{zjk} = a_{z(j-1)k}$ (z:1…i-1)
  - Al nodo de la columna i se le asigna el valor de la fila anterior menos uno
    - $a_{ijk} = a_{i(j-1)k} - 1$
  - A los nodos en columnas posteriores a i se les asigna un valor siguiendo la ecuación (2-7)
- Calcular el ancho desperdiciado por medio de la ecuación (2-9) y volver al paso 1
- Decrementar el índice i (i = i-1) y si es 0 terminar el algoritmo; de lo contrario volver al paso 2.[5]

Si se lee el algoritmo con cuidado se observa que el resultado es una matriz equivalente al árbol de búsqueda, es decir, que hemos representado todas las permutaciones posibles de anchuras a lo largo de la tela, obteniendo los patrones de corte viables.

Modelo de Gilmore y Gomory

Hasta este punto del problema se han calculado los posibles patrones de corte sobre el eje horizontal, tratando el problema como una serie de filas a extraer del rollo. Para continuar será necesario recalcular estos patrones de corte también en el eje vertical, de forma que nos queden dos tablas resultantes: una para los patrones de corte posibles aplicados como filas y otras para los patrones de corte posibles aplicados como columnas.

El modelo de Gilmore y Gomory se propone una vez calculados los posibles patrones de corte con la intención de seleccionar la combinación óptima de estos y resolver finalmente el problema. Para ello la creación de estos modelos consta de dos fases:

- Se selecciona el patrón de corte para uno de los ejes, ya sea a lo ancho o a lo largo del material, aunque para este documento se asume que se usa el ancho. Para ello se define j como un patrón de corte factible y λ_j^0 como el patrón de corte que usamos en esta etapa.
- Se selecciona el patrón de corte que se aplica en el eje restante para dejar la pieza recortada y se define como λ_j^s al patrón seleccionado, siendo $s \in \{1, 2, ..., i\}$ la pieza cuya anchura sea la limitante para el resto de las piezas del patrón de corte seleccionado para esta fase, es decir, el número que identifique a la pieza de mayor anchura que se recorte en el patrón vertical js, a su vez, j representa el número de patrón de corte de anchura s que se utiliza. Para llevar un orden claro se asume que el número que define a cada pieza i es menor cuanto menos ancha es la pieza.

Para estas fases el objetivo es minimizar la ecuación (2-10):

$$z = \sum_{j=1}^{m_k} \lambda_j^0 \tag{2-10}$$

Sujeto a:

1. La limitación de que la suma de todas las piezas de cada tipo insertadas a través de los patrones de corte de la fase 1 debe ser igual a la suma de todas las piezas de su tipo correspondiente insertadas a través de los patrones de corte de la fase 2.
2. El número de piezas recortadas de cada tipo debe ser menor o igual al número de piezas demandadas del tipo correspondiente.
3. El número de veces que se aplica cada patrón de corte debe ser entero y positivo. [6][7]

En la Figura 2-3 se observa cómo se dispone el patrón horizontal usado en la primera fase λ_j^0 para el único rollo del ejemplo y sobre este se trazan los patrones de corte verticales que terminan de dividir el rollo para recortar las piezas requeridas en su interior De utilizarse más de un rollo se tendría más patrones horizontales utilizados (uno por rollo) sobre los cuales se proyectarían a su vez un numero de patrones verticales equivalente al número de elementos dentro del patrón de corte horizontal, formando una solución al problema siempre que la suma de las piezas colocadas agote el número de piezas a recortar.

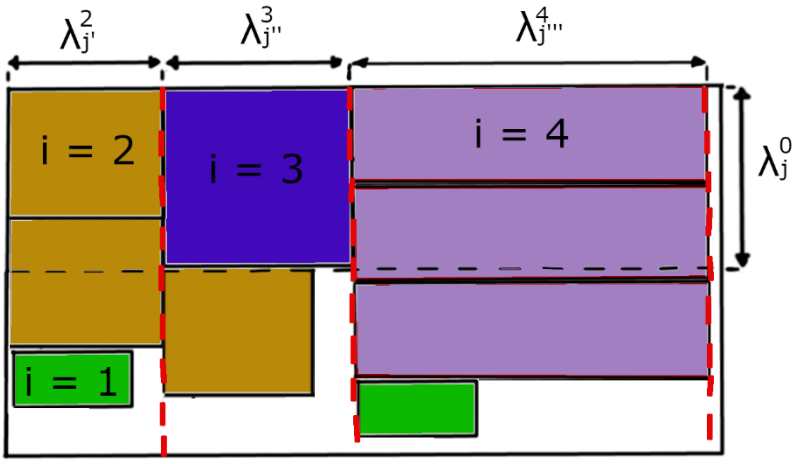


Figura 2-3: Ejemplo gráfico de colocación según el modelo de Gilmore y Gomory

2.2 Simplificación del problema

En esta sección se analizan los algoritmos utilizados para simplificar el problema, la mayoría de estos se centran en la reducción del número de comprobación de colisiones a realizar; ya sea por medio de un procesado previo de las colisiones con un modelo más simple, el uso de la geometría de la pieza en cuestión para evitar comprobaciones en determinadas posiciones o la construcción de piezas más simples que nos quiten la necesidad de comparar colisiones con la pieza real a determinadas distancias.

2.2.1 Cuadro delimitador mínimo

Esta técnica consiste en calcular el rectángulo mínimo que contiene a la pieza a analizar cuyos lados sean paralelos a los ejes x e y. El objetivo de este cuadro es hacer lo más fácil posible calcularlo con el objetivo de usarlo para un cálculo de colisiones previo al definitivo que determine qué elementos están lo suficiente cerca como para necesitar comprobar su posible colisión.

2.2.2 Aproximación poligonal

Dada la naturaleza irregular de algunas piezas que es posible encontrar en este problema se han buscado diversas formas de simplificar los cálculos. Uno de los métodos más usados es la aproximación poligonal, que calcula un número de segmentos rectos que marcan los límites de la pieza en sustitución de la traza irregular que la compone. Este método en efecto reduce el peso de los cálculos que se han de realizar y acelera el algoritmo; sin embargo, según el método utilizado pueden producirse pérdidas de material o problemas en la detección de colisiones entre los dientes que se forman entre las rectas que ahora componen los límites de la traza. [8]

Caben destacar tres métodos que se utilizan a la hora de crear estos límites.

-El primero y más simple de estos métodos es la aproximación rectangular, que convierte las curvas a rectas aproximando su forma a la de un rectángulo que la contenga. El funcionamiento de esta técnica mejora si se usan varios rectángulos para aproximar la forma de la curva dado que se desperdicia menos material, pero esto implica una mayor lentitud a la hora de aplicar el algoritmo y un mayor número de rectas con las que comparar colisiones.[9] Ejemplo visible en la Figura 2-4

-El segundo método consiste en el cálculo de una envolvente convexa, es decir, una cápsula que recoge en su interior todos los puntos que se definen para componer la figura. El número de puntos seleccionados para definir la figura antes de calcular la envolvente convexa puede ser tan alto o limitado como se considere. Es por eso que para este método se debe tener en cuenta en cuenta que si no se es cuidadoso a la hora de seleccionar los puntos para delimitar la aproximación habrá partes de la figura que no estén contenidas en ella; esto es especialmente conflictivo a la hora de comprobar la colisión entre dos figuras, dado que esta aproximación intenta simplificar estos cálculos, pero si no es capaz de detectar una colisión mientras que sucede, el esfuerzo habrá sido inútil. También cabe destacar que una selección de muy pocos puntos de referencia para el cálculo de esta aproximación generaría más desperdicio de material y que una selección muy detallada aumentaría la complejidad del algoritmo, pudiendo hacerlo para algunos casos incluso más lento que en caso de no usar esta aproximación. [10] Ejemplo visible en la Figura 2-5

-El tercer y último método consiste en la utilización de una cuadrícula para transformar la pieza, definiendo a los polígonos aproximados por el conjunto de cuadrados que definen la pieza, esta técnica se trata de nuevo más adelante al ser un estándar para la simplificación del cálculo de colisiones [8] Ejemplo visible en la Figura 2-6

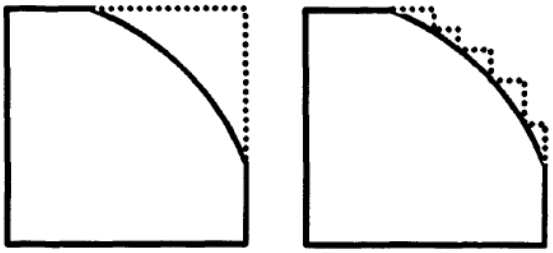


Figura extraída de la referencia [9]

Figura 2-4: Ejemplo visual de una aproximación rectangular

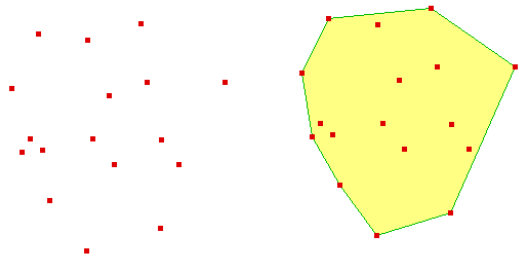


Figura extraída de la referencia [10]

Figura 2-5: Ejemplo visual de una envolvente convexa

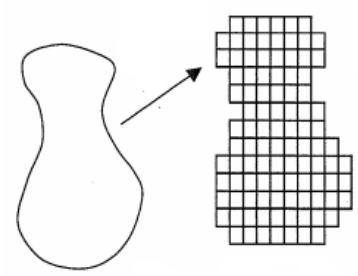


Figura extraída de la referencia [8]

Figura 2-6: Ejemplo visual de una transformación cuadricular

Incluso tras la aplicación de estas simplificaciones la detección de colisiones aun representa una gran carga computacional. Por ese motivo algunos autores se inclinan por utilizar pruebas de colisión antes de la ejecución del algoritmo principal alargando la fase de previa de procesado de datos en virtud de reducir el tiempo total de ejecución. [9]

2.2.3 Vector corredizo

Esta técnica es una de las bases del funcionamiento del software de marcado de trazos de corte semiautomatizado. Su aplicación asume una colocación secuencial de las piezas a lo largo del material por lo que es clave el orden en el que estas son introducidas. La segunda variable clave es la decisión de un vector por el cual se desplaza en línea recta la pieza correspondiente a cada iteración hasta colisionar con alguna otra pieza previamente colocada o el límite del material. Una vez alcanzada la colisión se repetirá el proceso, calculando, con ayuda de heurísticas, una nueva dirección para tantos vectores como iteraciones máximas se permitan hasta que la posición alcanzada cumpla unos requisitos mínimos para considerar la colocación satisfactoria.

Esta técnica simplificara el problema notablemente, pues sin su uso es necesario comprobar si existe una colisión para cada movimiento de la pieza, mientras que, con la ayuda de este vector, se calcula la distancia a la que se genera el choque avanzando en la dirección indicada y una vez calculada la pieza puede moverse por el vector libremente con coste computacional cero y sabiendo que no va a colisionar con ninguna otra pieza hasta alcanzar la distancia calculada, con la cual también conoceremos el punto de impacto exacto.[11]. Cabe destacar que, aunque esta técnica es muy útil para polígonos los cálculos en estos vectores se vuelven más farragosos cuando se aplican a piezas irregulares.

2.2.4 Rotación selectiva

A la hora de rotar las piezas no es posible probar todos los ángulos posibles por lo que nos vemos obligados a tener que reducir el número de ángulos a probar. Sin embargo, existen situaciones en las que se pueden reducir el número de ángulos en los que rotar una pieza a colocar sin aumentar las probabilidades de generar un mayor desperdicio de material; la clave para ello es la simetría de las piezas.

Aquellas piezas que, por ejemplo, sean simétricas en dos ejes perpendiculares no necesitan rotar más de 180 grados. Otras, como los cuadrados, que son simétricas en 4 ejes distribuidos cada eje a 45 grados del anterior, no necesitan rotar más de 90. Otro ejemplo son las piezas simétricas en todos sus ejes (un círculo) que no necesiten ser rotadas en absoluto.[12]

2.2.5 Grafos de restricción

Los grafos de restricción forman una estructura descriptiva n dimensional que nos permite almacenar la colocación de las piezas en el plano de forma eficiente siempre que sean rectangulares (hablando de dos dimensiones) u ortoedros (hablando de 3 dimensiones). Para definir los grafos que describirán la colocación se añade uno por cada dimensión del problema. En el caso de dos dimensiones tendremos un grafo oeste (G_w), con un nodo inicial W y un grafo sur (Gs) con un nodo inicial S; ambos nodos iniciales estarán conectados a todos los nodos que representan cada pieza y cada uno de estos nodos pieza está conectado direccionalmente a aquellos a los que preste soporte en el eje que el grafo represente. De esta manera se puede hallar la posición de una pieza respecto de cada eje recorriendo la ruta más larga en cada grafo que llegue hasta él y sumando la altura o anchura (según el grafo que se consulte) de los elementos recorridos en el camino. De la misma manera uno podría determinar la posición de todos los rectángulos con la ayuda de los grafos en tiempo $O(n^2)$.[13] En la Figura 2-7 se muestra un ejemplo de grafos de restricción.

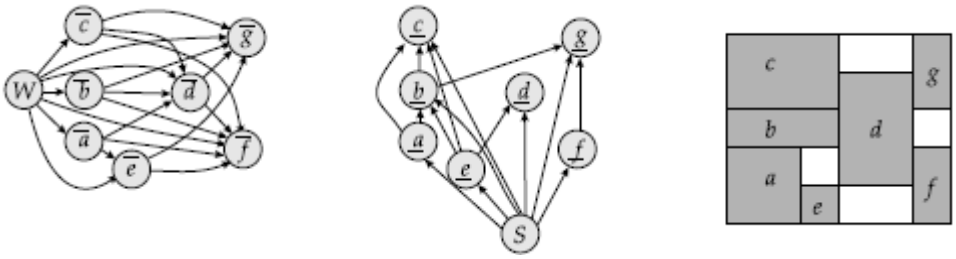


Figura extraída de la referencia [13]

Figura 2-7: Ejemplo gráfico de la aplicación de grafos de restricción

2.3 Detección de colisión

Aunque en algunos casos concretos la utilización de trigonometría directa para comprobar la colisión entre dos objetos pueda ser útil, existen otros tantos métodos que para casos concretos alivian la carga computacional respecto de este a veces rudimentario método.

2.3.1 NFP (no-fit Polygon)

Un NFP es una estructura poligonal utilizada para el control de colisiones generada en el preprocesamiento de las piezas. Esta estructura se construye para dos piezas y es útil para detectar la colisión entre esas dos piezas. Para definir un NFP respecto de los polígonos a y b se define un punto de referencia p en el polígono a, se arrastra la pieza a por el borde de la pieza b hasta haber recorrido toda su superficie y se crea un polígono con los segmentos dibujados por el punto p en el proceso. Este polígono resultante es el NFP de a y b respecto del punto p y nos permite saber cuándo las piezas están en colisión comprobando si el punto de referencia se encuentra dentro del NFP; es decir, que el área definida por el NFP es aquella en la que de encontrarse el punto de referencia existe una colisión entre las piezas.[14] Se observa un ejemplo de construcción de NFP en la Figura 2-8

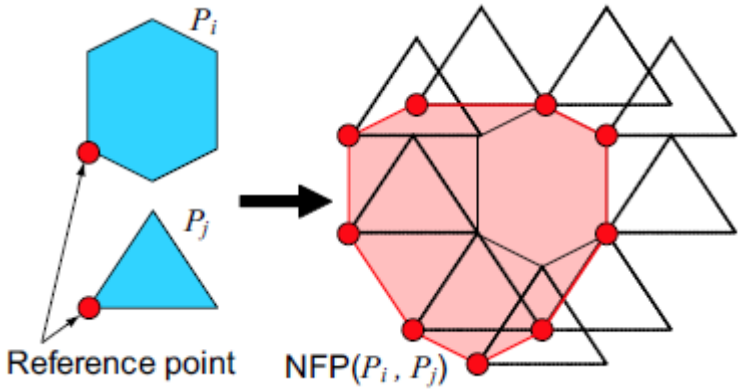


Imagen extraída de la referencia [14]

Figura 2-8: Ejemplo gráfico de la construcción de un NFP

Definiendo esta área podemos facilitar los cálculos de colisión entre dos objetos, pues la carga computacional dura se ejecutará solo en la primera iteración; para el resto de ellas solo hará falta comprobar si el punto de referencia se encuentra o no en el área descrita.

Este método también es útil para piezas irregulares, aunque cabe destacar que dificulta su cálculo. Sin embargo, es un método problemático en lo que a rotaciones se refiere, pues si se rota alguna de las piezas que componen el NFP tendrá que redefinirse el área de colisión.

También es posible generar un NFP entre el rectángulo que supone el material madre y un determinado polígono; este se denomina IFP("Inner-Fit-Polygon") y determina los límites en los que la pieza se sale del material.[15]

2.3.2 Cuadrícula/Mapa de bits

Esta técnica consiste en definir el plano como una matriz a la que, para hacer más sencilla su visualización se compara a un mapa de bits. En este mapa se le da un valor nulo a las casillas que no contengan ningún trozo de pieza y un valor diferente a aquellas que contengan algún trozo de pieza, inscribiendo las piezas en el mapa de una en una. Si se da el caso de que al intentar marcar una casilla como ocupada plasmando una pieza en el mapa resulta que la casilla ya estaba marcada se puede afirmar que existe una colisión en esas coordenadas.

Esta técnica de detección de colisiones viene con el problema de que requiere una gran cantidad de memoria, además de que es muy costosa a nivel procesador por el hecho de que requiere plasmar el área de cada pieza en la matriz para cada posición de cada pieza. Además, a mayor precisión de la matriz requiere un coste de procesamiento acorde. Es especialmente útil para formas irregulares dado que evita el uso de operaciones complejas para la intersección de las piezas.[16]

2.4 Método de colocación

Los métodos de colocación expuestos a continuación parten de la base de que la colocación de las piezas viene dada de forma secuencial, no empezando a colocar una pieza hasta que la anterior no tenga una posición fija decidida

2.4.1 Heurística Bottom-left

Esta heurística posiciona la pieza a colocar en la esquina superior derecha y la desplaza hacia abajo hasta que colisione con otra pieza o el final del material. Después hace lo mismo, pero hacia la izquierda. Al colisionar esta segunda vez repite la secuencia desde el principio hasta que no sea posible desplazarse más ni hacia la izquierda ni hacia abajo o se llegue a un máximo de iteraciones predefinido. Existen variantes de esta heurística que, en lugar de mantener el bucle, tras impactar con algún objeto, se desplazan por su superficie mientras que el vector los lleve a la izquierda y abajo en alguna medida; sin embargo, son usadas para piezas regulares. La ventaja de esta heurística es su velocidad y simplicidad.[17]

2.4.2 Método constructivo

Se coloca la pieza inicial en la esquina inferior izquierda y a partir de ella se definen 3 puntos (x_{max}, y_{max}), (x_{max}, 0) y (0, y_{max}). Partiendo de estos puntos se colocará la esquina inferior izquierda o derecha de la siguiente pieza, siendo desplazada posteriormente hacia abajo, hasta colisionar y hacia la izquierda hasta colisionar para cada uno de los posibles puntos de partida. La versión de colocación de la pieza que haya avanzado más hacia abajo y hacia la izquierda será aquella que se acepte y para la siguiente pieza a colocar se añaden a la lista de posibles puntos de partida cinco puntos nuevos respecto de la última pieza colocada: (x_{max}, y_{max}), (x_{max}, 0), (0, y_{max}), (x_{max}, y_{min}) y (x_{min}, y_{max}), no manteniendo los puntos de partida repetidos ni utilizando aquellos inviables con el posicionamiento inicial de la pieza sin colisión.

Como variante de este método se contemplan otros criterios de aceptación para la inserción de la pieza, entre ellos aceptar la inserción con la que se pueda crear el mínimo rectángulo

en área que contenga a todas las piezas. También se contemplan como variantes cambios en los puntos de partida, como por ejemplo usar en lugar de los propuestos las cuatro esquinas que definen el rectángulo más pequeño que contiene la pieza.[17]

2.4.3 Heurística de horizonte(skyline)

Antes de explicar esta heurística es necesario especificar que solo se utiliza para el "strip packing problem" y el "square packing problem" (el problema de embalaje sobre una tira o sobre un cuadrado) en dos dimensiones. Ambos problemas son similares al que se analiza, pero no contemplan la existencia de objetos no rectangulares y por tanto tampoco la posibilidad de necesitar rotar estos.

Para aplicar esta heurística es necesario crear un nuevo set de variables en el problema, que compondrán lo que se define como horizonte(skyline). Asumiendo que se colocan las piezas de una en una y apiladas, empezando por la parte inferior del plano bidimensional se define un contorno que envuelve la parte superior del conjunto de piezas. Este contorno es el horizonte y se define por los segmentos horizontales que lo componen. Estos segmentos deben cumplir para su definición correcta una serie de reglas:

1. Cada segmento S_j debe tener una coordenada y diferente a S_{j+1}.
2. La coordenada x del extremo derecho del segmento S_j debe ser la misma que la del extremo izquierdo de S_{j+1}.

Una vez está definido el concepto de horizonte se puede usar su heurística para realizar la colocación de piezas. Esta heurística nos aporta unas reglas básicas sobre cuándo un segmento del horizonte es utilizable para colocar una pieza sobre él, teniendo en cuenta que se asume que una pieza puede colocarse poniendo su esquina inferior izquierda sobre el extremo izquierdo del segmento S_j o poniendo su esquina inferior derecha sobre el extremo derecho del segmento.

El extremo izquierdo de S_j es candidato si y solo si S_{j-1} tiene una coordenada y mayor o si S_j tiene su extremo izquierdo en la mínima coordenada x. De la misma forma el extremo derecho de S_j es candidato si y solo si S_{j+1} tiene una coordenada y mayor o si S_j tiene su extremo derecho en la máxima coordenada x.

Una vez colocada la pieza en cada iteración se recalcula la línea de horizonte, partiendo los segmentos, si es necesario, en las secciones en las que, por incluir la nueva pieza, se observe una antes inexistente diferencia en la coordenada y del horizonte y se continúa con el proceso. [18]

2.5 Orden de colocación

Los algoritmos genéticos son una herramienta muy útil a la hora de optimizar la distribución de patrones pues, aunque se han analizado diferentes métodos heurísticos para resolver el problema, no existe uno que sea mejor que los demás para todos los conjuntos de piezas a analizar; de la misma manera, para definir el orden de colocación de los elementos tampoco existe un patrón siempre mejor. Por este motivo se contempla al algoritmo genético como la solución, siendo muy eficaz a la hora de explorar el espacio muestral, deshaciéndose no a mucho tardar de los individuos que exploren zonas sin utilidad y aumentando el número de aquellos que exploren las estrategias prometedoras.

Por ello en la referencia [19] se propone un individuo con 3 campos:

1. Una permutación que contiene el orden en el que los elementos se introducen.
2. La rotación que aplicar a cada uno de los elementos a añadir.
3. La heurística que utilizar para colocar estos elementos.

En ausencia de un algoritmo genético para ordenar los elementos existe la posibilidad de utilizar heurísticas que generalmente dan buenos resultados, como pueden ser la de colocar primero las piezas más grandes o la de agrupar las piezas de un mismo tipo en la colocación. Esta última puede además combinarse con el algoritmo genético, como se hace en la referencia [19] reduciendo la dimensión del problema en la medida en la que haya muchas piezas repetidas, aunque a su vez cerrando las puertas a resultados óptimos que se encuentren fuera de esta heurística.

3 Diseño

En este apartado se analizan las estructuras y tecnologías que se utilizan para la construcción del prototipo para la resolución del problema de optimización del trazado de patrones de corte y se explica la manera en la que estas se utilizan para la construcción de este, ofreciendo una reflexión del motivo por el cual se cree que darán un buen resultado para la resolución del problema.

3.1 El algoritmo NEAT

El método NEAT (Neuro Evolution of Augmenting Topologies) ofrece una variante de algoritmo genético que a su vez mantiene una estructura de nodos y conexiones entre estos que recuerda a la de una red neuronal; sin embargo, sus peculiaridades la diferencian entre estas por la flexibilidad que se le da a esta estructura.

El genoma de este algoritmo está compuesto por dos tipos de genes:

-Genes nodo: Estos genes almacenan los nodos de entrada, salida e intermedios que se encuentran en un genoma.

-Genes de conexión: Estos genes almacenan un nodo origen, un nodo destino, el peso de la conexión entre ambos, un indicador de activación y un identificador de innovación.

Se distribuye de esta manera para permitir que cada individuo de una población NEAT pueda tener una estructura diferente, siendo un algoritmo que promueve la exploración de las posibles topologías de redes neuronales que puedan resultar óptimas para el problema, permitiendo de esta manera un crecimiento progresivo de esta estructura, además de la eliminación de nodos innecesarios en la red y la supervivencia de diferentes especies dentro de la población para explorar, dentro de lo posible, el espacio muestral sin eliminar especies antes de permitirles llegar a su máximo desarrollo. [20][21]

3.1.1 Fases NEAT

A continuación, se enumeran las fases de la ejecución del entrenamiento del algoritmo NEAT, que se estructuran de la misma manera que las de un algoritmo genético, contemplando selección, recombinación y mutación para la construcción de una población de soluciones al problema planteado.

Selección

Para seleccionar individuos en la población el algoritmo primero divide los genomas por especies, definiendo al primer elemento de la primera iteración como especie inicial y asignando a esta especie todos los individuos con un cierto rango de similitud genética. Los individuos fuera del rango de todas las especies existentes en la población definen una nueva especie, en la que a su vez se introducirán los individuos en el rango de similitud. Cabe explicar que estas especies son excluyentes entre sí, por lo que un individuo solo puede formar parte de una especie. Este sistema existe por una de las condiciones antes explicada: el algoritmo pretende eliminar de la población solo un determinado porcentaje de cada especie, evitando suprimir a las especies menos desarrolladas que aún no han tenido tiempo de desarrollar su potencial pese a que su fitness no sea óptimo.

Recombinación

A la hora de combinar dos individuos sale a relucir la peculiar distribución del genoma; esto es debido a la existencia de genes de conexión en algunos genomas, pero no para otros,

existencia que podemos comprobar gracias al id de innovación, único para cada gen de conexión y gracias al cual en esta etapa de la iteración podemos saber qué genes existen para ambos padres y cuáles son nuevos. Con la ayuda de este identificador se marcan los genes que existen únicamente en uno de los padres como:

- disjunto (disjoint gene) si tienen un identificador igual o menor a aquel de mayor valor del padre con menor máximo identificador de innovación, es decir, aquellos de menor o igual innovación que el gen más innovador del padre sin las innovaciones más novedosas.
- en exceso (excess gene) si dado el identificador del gen de innovación máximo del padre con un máximo menor estos genes tienen un identificador de innovación de mayor valor, es decir, todo aquel que no sea disjunto.

Esta distinción se realiza porque al recombinar los genes y encontrar genes únicamente en uno de los padres los mantenemos directamente de ser disjuntos, pero si son en exceso lo haremos únicamente si el padre que los aporte tiene un mayor valor fitness que el otro. El resto de los genes de conexión se recombinan como si de un algoritmo genético común se tratase.

Mutación

La mutación en el algoritmo NEAT se contempla sobre 4 diferentes partes del genoma:

- Se crea un nuevo gen de enlace con un peso y unos nodos origen y destino aleatorios y le asigna un nuevo identificador de innovación sumando uno al máximo existente antes de él.
- Se crea un nuevo gen nodo partiendo un gen de conexión en dos, el primero manteniendo el origen inicial y apuntando al nuevo nodo y el segundo manteniendo su destino inicial y partiendo del nuevo nodo.
- Se modifica el estado activación/desactivación de un gen de conexión.
- Se modifica el peso de un gen de enlace.

3.1.2 NEAT para ordenación

El diseño original de la ordenación por medio de un algoritmo genético busca hallar patrones de colocación en base a prueba y error. Es funcional para encontrar mínimos locales con una relativa eficacia para este ejercicio; sin embargo, teniendo en cuenta que cada ejecución de este problema supone un número muy alto de operaciones básicas, generadas por el control de colisiones al colocar las piezas, es de crítica importancia maximizar la velocidad de aprendizaje, captando lo más rápido posible los patrones de ordenación y sus ángulos aplicados en un número mínimo de intentos de colocación. El problema ofrece sus resultados óptimos para un número ilimitado de piezas y aunque no podemos ni acercarnos a simular esto, sí que es óptimo maximizar el número de piezas a colocar para este problema, este es otro de los motivos por los que se entiende que el número de iteraciones para el aprendizaje será mínimo.

Una vez vistas estas condiciones, este documento propone un algoritmo que memorice patrones de una forma más dinámica, aunque también se base para ello en la prueba y el error, para ello se revisan los planteamientos de resolución del problema mediante programación lineal, que siguen el orden de:

1. Crear los posibles patrones de corte aplicables a una fila y a una columna.
2. Crear las posibles combinaciones de un patrón de corte de una fila con los patrones de columnas.

3. Dados los patrones posibles, buscar la combinación óptima de aquellos más útiles

Al ver un planteamiento que propone que el problema se simplifica al asumir la repetición de los patrones de corte óptimos aplicados de forma dinámica, este documento plantea la duda de si una estructura basada en redes neuronales podría memorizar y aplicar más dinámicamente que un algoritmo genético los patrones de ordenación óptimos para este problema, bajo la teoría de que estos tienden a repetirse, así como lo harían las rotaciones optimas.

Una vez planteada esta duda se propone el uso de esta estructura, sustituyendo la fase de ordenación, que usaría el algoritmo genético para generar directamente el vector que define el orden de las colocaciones, por un bucle de selecciones de piezas, que lleva a cabo la selección de la pieza a colocar de entre las restantes solo una vez la colocación de la anterior ha concluido. De esta manera se aprovechan los datos respectivos a la posición de las piezas ya colocadas para seleccionar, en una velocidad mínima, la pieza a colocar usando una información que el algoritmo genético simplemente ignora.

Para este problema no conocemos la solución óptima y de hallarla tampoco podríamos probar que lo es. Esto podría suponer un problema para una red neuronal normal, pues no podría realizar un entrenamiento sin este dato; es por eso por lo que se necesita una estructura que permita entrenar la red como si de un algoritmo genético se tratase, es decir, con una función fitness. Además de esto sabemos que la organización de las neuronas de la red y su profundidad óptimas pueden variar en función de las piezas a introducir; con esto se llega a la conclusión de que NEAT es el algoritmo adecuado para poner a prueba este método de resolución. NEAT creará una población que para cada individuo procesará la pieza adecuada a colocar para cada movimiento y que entrenará con el fitness resultante de la colocación completa generando una topología de red para la memorización de patrones y por tanto la selección de piezas adecuada para cada dimensión del eje fijo y grupo de piezas definidas.

El algoritmo base

Para utilizar el algoritmo previamente descrito en sustitución del algoritmo genético para la fase de decisión de orden de colocación de piezas del problema, se definen como neuronas de entrada de la red un número n de segmentos que describan la posición de las últimas piezas colocadas, anotando de ellos:

- Sus coordenadas normalizadas en cada uno de los ejes
- El tipo de segmento (recto o circular)
- El sentido en el que se encuentra la curvatura (cóncava o convexa respecto de la normal)

Estos segmentos serán los considerados más superficiales y darán a la red neuronal los datos que necesita para conocer “a grosso modo” el contorno de la figura que forman las piezas ya colocadas. Estos son elegidos descriptores del algoritmo base pues permiten a la red hacerse una idea de dónde va a ser colocada la pieza que seleccione después y en qué tipo de segmentos va a colisionar. Con estos datos puede seleccionar una pieza cuya forma se adapte bien a las posibles colisiones y que encaje bien en su posición final, como podría ser por ejemplo el caso de una pieza con un segmento cóncavo seleccionada en el ángulo adecuado para acabar encajada en el segmento convexo de otra pieza.

La orientación del algoritmo a la selección de piezas optimas en tiempo de colocación puede llevar a la conclusión de que la pieza optima a colocar es de un tipo de pieza del que no

quedan existencias, es decir, de la que ya se han colocado todas las necesarias. Es por estos casos que la salida de la red estará compuesta por:

- El índice de elegibilidad para cada uno de los tipos de pieza, siendo aquella con mayor índice, de la que queden existencias, la elegida para ser colocada
- La rotación óptima seleccionada para cada uno de los tipos de pieza posible, de manera que si no quedasen existencias de la pieza óptima para la posición no se aplicase la rotación óptima para esta a la pieza finalmente seleccionada.

Datos de entrada adicionales

Además de los argumentos de entrada utilizados para el algoritmo base este documento propone otros dos tipos de argumentos que en combinación con el primero podrían mejorar los resultados y acelerar el aprendizaje.

La primera propuesta consiste en añadir neuronas de entrada que contengan, bajo la misma estructura de almacenamiento que en el algoritmo base, los segmentos que contiene cada tipo de pieza, es decir, esta propone dar información sobre la forma de cada una de las piezas a colocar. De esta manera la red neuronal podría hacerse una idea de las dimensiones relativas de la pieza, visto que las coordenadas introducidas para los segmentos están normalizadas. Esto le permite entrenar generar estrategias de ordenación adaptadas a las dimensiones de las piezas y requiere de la red menos experimentación para conocer cómo cada una de las piezas a colocar choca con cada una de las ya colocadas. Estos nuevos datos deberían ser especialmente útiles en aquellos casos en los que se utilicen muchos tipos de piezas diferentes.

La segunda propuesta consiste en añadir neuronas de entrada que contengan las piezas restantes de cada uno de los tipos. Estas serían almacenadas normalizadas con respecto a la cantidad de piezas de la pieza más necesitada y darían a la red la capacidad de mantener un equilibrio en su orden de colocación, de forma que no se consuman en un principio todas las piezas que encajan inmediatamente mejor dejando para el final piezas que no encajan nada bien. Estos nuevos datos deberían ser especialmente útiles para acelerar el aprendizaje en las primeras iteraciones, ofreciendo en un principio colocaciones más homogéneas sobre las que mutar.

Medición de fitness

Dado que no se busca que el algoritmo haga una inserción lo más eficiente posible para una única inserción sino para el total de inserciones, la medición del fitness y aplicación de la recombinación y selección se realizará al haber colocado la totalidad de las piezas. El índice usado para determinar la eficacia (fitness) de cada individuo será la altitud máxima alcanzada por la totalidad de las piezas, siendo el fitness optimo aquel que eleve al mínimo posible la coordenada máxima en el eje. Aunque se intente minimizar el porcentaje de material desperdiciado, para resolver el problema no se hace necesario calcular esta cada iteración dado que utilizamos siempre el mismo número de piezas y el mismo valor para el eje fijo al ejecutar el algoritmo. Es por esto por lo que para alcanzar un mínimo óptimo para una combinación de piezas a colocar será necesario ejecutar el algoritmo diseñado varias veces, alterando para cada ejecución las dimensiones del eje fijo en virtud de hallar la anchura de tela adecuada para recortar las piezas.

3.2 Descripción de las piezas

A la hora de almacenar y calcular la posible colisión de las piezas es de suma importancia elegir un modelo lo más eficiente posible. Dado que en este problema nos enfrentamos a la existencia de piezas irregulares se debe contemplar además la existencia de piezas con segmentos curvos.

La solución propuesta comienza por limitar la capacidad descriptiva pasando a entender las curvas únicamente como trozos de circunferencia; esto permite describir cualquier curva, aunque puede ser problemático a la hora de definir curvas muy irregulares, que no suelen ser frecuentes en las industrias que puedan requerir el uso de las tecnologías descritas en este documento. Definimos por tanto a cada pieza por el conjunto de segmentos que la componen, teniendo a los segmentos curvos definidos por tres puntos y un sentido. Estos son: los dos extremos del segmento, el centro de la circunferencia que define la curva y el sentido en el que esta está dibujada, es decir, para las dos posibles partes de la circunferencia que pueden unir los dos puntos se especifica cuál es la deseada respecto de el orden en que se describen los puntos extremos del segmento, calculándose en la fase de preprocesamiento el radio para facilitar los cálculos.

Entendiéndose que el interior de la pieza se encuentra a la derecha del segmento viendo el primer punto definido como abajo y el segundo como arriba un sentido negativo indica que la curva se adentra en el interior de la pieza (derecha) mientras que un sentido positivo indica que la curva se avanza hacia el exterior de la pieza (izquierda). Esta forma de ver el problema facilita los cálculos de colisiones permitiendo encontrar las colisiones en curva como colisiones con una circunferencia. Después de hallar los puntos de colisión solo resta comprobar si estos puntos se encuentran en la sección de circunferencia descrita realmente en la figura.

Este sistema de descripción de piezas además cuenta con la ventaja de cargar unos datos mínimos, pero sumamente descriptivos para llevar al algoritmo NEAT como argumentos de entrada.

4 Desarrollo

Este apartado muestra la forma en la que las tecnologías y los métodos descritos se unen para la construcción final del prototipo, dando detalles de la configuración necesaria en estos, de las peculiaridades y metodologías de cada parte de la ejecución y concluyendo con los principios detrás del sistema de comprobación de éxito y la composición de este.

4.1 Configuración inicial

El problema que se trata de resolver en este documento tiene en cuenta uno de los ejes de un tamaño ilimitado; sin embargo, a la hora de introducir los datos de las posiciones al algoritmo NEAT crea un cierto conflicto, dado que el dato de la coordenada y no tiene un máximo, y estos datos deben estar normalizados en proporción a el marco que los contiene (en valores de cero a uno), por lo que para poder acotarlo se determina un límite superior que simula el infinito para el eje y, se determina el límite del eje x y se listan todos los tipos de piezas posibles junto con la cantidad de cada uno de ellos a colocar.

Una vez definido el marco se construyen las piezas a colocar creando un conjunto de segmentos unidos entre sí que formen una pieza cerrada. El orden de descripción de estos elementos es relevante, pues como se explica en el capítulo de diseño, los segmentos definen el interior de la pieza. Esto será relevante para definir el sentido en el que se trazan las curvas, por lo que para diseñar una pieza adecuadamente se debe construir definiendo sus segmentos en sentido horario, de manera que si se define una pieza siempre podamos saber en qué dirección está su interior y en cual su exterior calculando la normal a cada segmento.

Para cada pieza construida se determina un número de unidades necesario en cada paquete de piezas; por ejemplo, en el paquete de piezas necesario para construir una camiseta harían falta una pieza frontal, una pieza para la espalda, un cuello y dos mangas. Una vez definido el número de unidades para cada pieza en un paquete, se determina el número de paquetes a introducir en el algoritmo, para este caso, el número de camisetas a recortar.

4.2 Aplicación del algoritmo

Una vez conocida la composición básica del algoritmo NEAT, del método de colocación Bottom-left y de la comprobación de colisiones con uso del cuadro delimitador, se revisa de qué manera el algoritmo aplica estas tecnologías en conjunto en cada una de sus fases:

- Selección: sustituye a la antes llamada ordenación y usando NEAT ofrece una pieza y una rotación.
- Colocación: posiciona la pieza en su rotación seleccionada según corresponda.
- Control de colisiones: se hace cargo de que el proceso de colocación se lleve a cabo sin generar incongruencias.

4.2.1 Selección

El proceso que antes se analizaba como ordenación ya no puede considerarse como tal, pues para el prototipo desarrollado no se genera un orden de colocación, sino que se selecciona para cada iteración dentro de una misma colocación el tipo de pieza que se considera óptimo para colocar. Esto implicará la necesidad de introducir un sistema para controlar la cantidad de piezas restantes de cada tipo, lo que hará algo más lento el proceso que en el algoritmo genético, que únicamente necesita un orden para su vector de piezas. El sistema introducirá a la red del algoritmo NEAT las entradas a la red de la versión del algoritmo en uso y con estos datos se generarán los índices de elegibilidad y las rotaciones para cada tipo de pieza. Con estos datos el sistema de conteo de piezas selecciona la pieza con mayor índice de

elegibilidad de entre las que aún tengan piezas disponibles y la pondrá, junto con su rotación correspondiente a disposición del algoritmo de colocación

4.2.2 Colocación

La estrategia de colocación decidida será Bottom-left, no por ser la óptima para la innovadora estrategia de ordenación, sino por ser una válida tanto para la nueva estrategia como para el algoritmo genético al que sustituye y con la que se pueden generar comparaciones prácticas entre ambas estrategias que ayuden a ver las diferencias entre ambas para diferentes muestras de datos y configuraciones.

Para gestionar el proceso de colocación de las piezas el procedimiento requiere para cada colocación:

- Seleccionar la pieza marcada como la más favorable según NEAT de entre las que aún queden existencias sin colocar.
- Aplicar a esta pieza la rotación seleccionada.
- Marcar el cuadro delimitador de la pieza; este permanecerá almacenado y será óptimo para poder determinar las piezas a comparar para el cálculo del vector corredizo, sobre todo, teniendo en cuenta que para la estrategia Bottom-left los lados de estos cuadros delimitadores son paralelos a los vectores de desplazamiento, simplificando los cálculos y marcando solo y únicamente las piezas con las que impactaría de avanzar en alguna de las direcciones.
- Colocar la pieza sobre el límite del eje y que simula el infinito y en el límite del eje x, pero para este estando la pieza aun completamente dentro de él.
- Utilizar el vector corredizo en el eje y para comprobar los posibles impactos y descender en el eje hasta el impacto.
- Utilizar el vector corredizo en el eje x para comprobar los posibles impactos y descender en el eje hasta el impacto.
- Ejecutar las siguientes instrucciones de colocación un numero n de veces a configurar (en las pruebas ejecutadas n = 2):
  - Utilizar el vector corredizo en el eje y para comprobar los posibles impactos y descender una distancia r por encima del límite de impacto, entrando en el cuadro delimitador de la pieza; si no impacta con ninguna pieza solo desciende hasta el eje.
  - Se comprueba si existe un impacto real entre las piezas; de no haberlo se avanza hasta que lo haya o se llegue al eje.
  - Se divide r a la mitad.
  - Se retrocede a ritmo r hasta que deje de haber impactos. Se comparará en busca de impactos la pieza a colocar con aquellas cuyos cuadros delimitadores colisionen con el suyo.
  - Se reestablece el valor de r y se repite la ejecución para el eje x.

4.2.3 Control de colisiones

El control de colisiones del prototipo tiene por objetivo minimizar al máximo posible el número de comparaciones necesarias para comprobar o descartar la existencia de colisiones entre todas las piezas, por eso al colocar las piezas separamos las comparaciones de colisión en dos etapas:

- La comparación de colisión de cuadros delimitadores es la primera y más superficial de las etapas de comparación y su ejecución comprueba si dos cuadros delimitadores chocan. Para ello se intenta hacer el mínimo número de comparaciones de media y no excluir ninguna posible colisión de la lista por lo que la estrategia usada consiste en

comprobar una a una si alguna de las esquinas del cuadro delimitado a se encuentran dentro de los límites del cuadro delimitador b, definidos por su esquina de valor máximo en ambos ejes y su esquina de valor mínimo en ambos ejes; después comprueba si ambos rectángulos forman un rectángulo en su área en intersección, es decir, si se tocan sin que la esquina de uno esté en los límites del otro, caso menos probable y por ello analizado después. La comparación se hace de esta manera considerando que es más rápida en general que comprobar la intersección de las rectas de los rectángulos, que además no contempla que un rectángulo esté dentro del otro y viendo que para este método se reutilizan la mayoría de los cálculos usando relativamente pocas operaciones básicas, con un máximo de 24.

- La comparación de colisión de piezas implica que el área de dos cuadros delimitadores se está tocando y que para verificar si existe o no una colisión real entre las piezas el único método es comprobar las colisiones entre los segmentos que componen cada una de las piezas. Para ello haremos uso de tres tipos de comparaciones en función del tipo de segmentos que compongan la comparación:
  - Comparación entre recta y recta.
  - Comparación entre recta y curva.
  - Comparación entre curva y curva.

  Para ello se debe tener en cuenta lo explicado en el capítulo anterior en relación con la codificación usada para las curvas, que simplificará los cálculos teniendo en cuenta que estas son secciones de circunferencias. Por lo tanto, para comprobar si existe una colisión con una de estas curvas la trataremos como una circunferencia y posteriormente si hay posibles puntos de colisión se comprueba si estos se encuentran realmente en la curva

Comprobación de punto en curva

Para comprobar si el punto de colisión con la circunferencia hallado se encuentra realmente en la sección de circunferencia que debemos tener en cuenta; se calculan los vectores que unen al centro de la circunferencia con los extremos del segmento y con el punto a comprobar. Estos se llamarán:

- a: para el primer punto del segmento descrito.
- b: para el segundo punto.
- p: para el punto a comprobar.

Ahora se calculan los ángulos que forma el vector a con b y p, ángulos que se llamará ab y ap, considerando estos ángulos en el sentido antihorario. Se determina que el punto se encuentra en la curva solo para los siguientes casos:

- La curva es cóncava y el ángulo ab es mayor o igual al ángulo ap
- La curva es convexa y el ángulo ab es menor o igual al ángulo ap

Nota: para este documento se considerará curva cóncava a aquella cuya curvatura se extienda hacia el interior de la pieza y convexa a aquella en la que la curvatura se extienda hacia el exterior de la pieza

4.3 Medición de éxito

En 1986 Harrison presentó cuatro criterios para juzgar el éxito resolviendo el problema de optimización de trazos de corte; adaptando estos principios a la terminología usada para esta formulación del problema se entenderían de la siguiente manera [9]:

1. Los patrones de corte usados deben generar un nivel aceptable de desperdicio (en lugar de mínimo).
2. Un patrón de corte debe comenzar y terminar en un mismo rollo de material madre.
3. La longitud del recorrido del conjunto de los patrones de corte usados debe ser tan alta como sea posible, lo que se interpreta como que la ocupación del rollo de material madre por parte de los patrones de corte debe maximizarse.
4. Colocaciones en rollos de material madre que resulten en un numero pequeño restante de piezas por colocar deben ser evitados.

La tercera regla no es del todo aplicable a esta versión del problema, visto que la longitud de uno de los ejes es variable en función de la pieza que más adentrada se encuentre en el eje. Es por ello por lo que se adaptan los principios de esta regla a, en lugar de maximizar la longitud de piezas en el material madre, maximizar el porcentaje de área ocupado por piezas.

En virtud de cumplir esta regla se apostará por maximizar el número de piezas a introducir en el material madre, dado que un número mayor de piezas, vista la flexibilidad del eje variable, no debería, por lo general, reducir el número de colocaciones óptimas, mientras que un aumento en el número de piezas ofrece más permutaciones entre sus posiciones, aumentando el espacio muestral y ofreciendo en la mayoría de los casos resultados mejores.

La segunda y cuarta regla deben ser ignoradas, pues para nuestro problema asumimos que el resultado es un único rollo de material madre que debe recoger la totalidad de las piezas y esto hace ambas reglas inaplicables.

Queda solo enfrentar la primera regla, que se interpreta como determinar cómo mejora en la colocación únicamente los cambios que se representen en una cantidad de tiempo aceptable, pues el objetivo de este tipo de software no es solo producir un buen resultado, sino hacerlo en un tiempo aceptable; tal como dice, no se busca el resultado mínimo, sino uno aceptable. Es por esto por lo que para comprobar el éxito del cambio aportado en el diseño se medirán dos variables: el porcentaje de área ocupada y el tiempo de ejecución.

4.3.1 Cálculo de área

Para poder obtener unos resultados congruentes es necesario conocer el área de las piezas que se introducen, para poder extraer de ellos el porcentaje de utilización de material. Este proceso se llevará a cabo en la fase de preprocesamiento, por lo que, aunque suponga una carga computacional intensa no debe suponer un problema teniendo en cuenta que solo se ejecuta una vez por cada tipo de pieza.

Transformación de las piezas en polígonos

Para simplificar el cálculo de áreas el primer paso es convertir las secciones de circunferencias en rectas e ir añadiendo a los cálculos el área de más o de menos a ajustar con el cambio; sin embargo, el producir este cambio puede causar problemas para algunas piezas, en el caso de que la conversión de curva a recta haya producido un cruce.

Es por esto por lo que para estos casos no bastará con convertir el segmento directamente en una recta y sumar (de ser convexa) o restar (de ser cóncava) el área de la sección de círculo. Para estos casos podemos encontrarnos con tres tipos de cruces:

- Con el extremo de un segmento o la totalidad de este, lo cual no supone un conflicto, pues únicamente es una tangencia y no un cruce.
- Con un número par de rectas, lo cual nos genera una incongruencia en la figura resultante y obliga a la determinación de otra sustitución para la curva, cuya solución aportada será la división de la curva en dos rectas en lugar de una, utilizando como punto de

intersección de ambas el punto intermedio del arco. Esta operación se repetirá para cada arco con intersecciones con rectas hasta que deje de haber incongruencias.

- Con una curva (ambas convexas): aplicar la misma solución que para la interacción con rectas
- Con una curva (alguna o ambas cóncavas): retrasar la conversión en recta de la curva hasta que la curva con la que generaba el conflicto esté convertida en recta, dado que para estas curvas solo se produce un conflicto mutuo para casos muy excepcionales. Aun así, si se da el inusual caso de una pieza con una cadena de conflictos que regresa a la primera curva, se debe imitar la solución aplicada para la intersección con rectas. Esta delegación tiene el objetivo de reducir el número de rectas que componen cada curva para esta conversión, dado que, si tenemos, por ejemplo, una curva cóncava entrando en una convexa y tratamos de convertir en recta la curva cóncava directamente, el número de rectas que la componen puede ser muy alto, llegando a infinito si están encajadas.

Fórmula del área de Gauss

Una vez convertida la pieza en un polígono podemos utilizar algoritmos para el cálculo de su área de forma directa. El decidido en este caso es el determinante de Gauss, que nos permitirá una resolución rápida y sencilla para calcular el área del polígono. La fórmula aplicada se encuentra en la ecuación (4-1)

- X_i: Coordenada en el eje x del punto i que define el polígono.
- Y_i: Coordenada en el eje x del punto i que define el polígono.

$$Area = \frac{1}{2}\left(\begin{vmatrix} X_1 & X_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} X_2 & X_3 \\ y_2 & y_3 \end{vmatrix} + \cdots + \begin{vmatrix} X_{n-1} & X_n \\ y_{n-1} & y_n \end{vmatrix} + \begin{vmatrix} X_n & X_1 \\ y_n & Y_1 \end{vmatrix}\right) \quad (4\text{-}1)$$

Una vez calculada esta área la sumamos con el ajuste de la conversión de curvas a rectas del anterior apartado y obtenemos la superficie que ocupa la pieza original, pudiendo ahora comprobar el porcentaje de área que ocupan las piezas en el material madre para los experimentos realizados en el apartado siguiente.

5 Integración, pruebas y resultados

En este capítulo se crean bases de datos con diferentes piezas a colocar y se comprueba en base a la colocación de estas como de buenas son cada una de las versiones del prototipo desarrollado y las peculiaridades que cada una muestra para los resultados simulados.

5.1 Pruebas con piezas simples

En primer lugar, se crea una base de datos con figuras simples, pero que contengan los posibles tipos de segmentos para que pueda verse el funcionamiento del algoritmo para con todo tipo de piezas. Es importante para este experimento probar esta base de datos, pues permite ver cómo se comporta el prototipo que utiliza NEAT frente al algoritmo genético para piezas sencillas en función del tiempo y el número de generaciones.

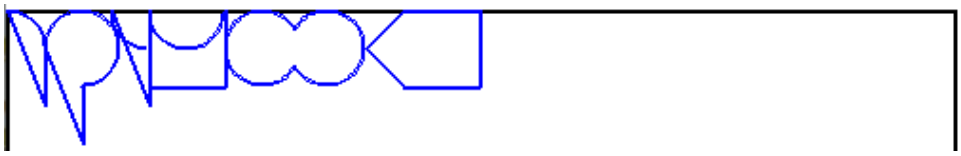


Figura 5-1: Tipos de pieza utilizados en la base de datos 1 en un marco de 500 pixeles de ancho.

Una vez creada la base de datos inicial se entrenan para una cantidad de piezas determinada de cada tipo tanto el algoritmo genético como el prototipo base, de manera que se visualicen las diferencias entre el aprendizaje de cada una en función del número de generaciones y del tiempo de ejecución para un tamaño de población equivalente entre ambas.

Como se observa en los resultados representados en la Figura 5-2 el prototipo que usa NEAT adquiere ya en la primera iteración una ventaja considerable frente al algoritmo genético, encontrando para alguno de sus individuos un patrón de colocación inicial lo bastante bueno como para partir de él y a partir de este ir progresando. A partir de este punto adquiere un rápido progreso en las primeras iteraciones en el que coge distancia respecto del algoritmo genético, una distancia de algo más del 5% de ocupación de área adicional que el algoritmo genético no es capaz de alcanzar antes de llegar al periodo de estabilidad de la gráfica.

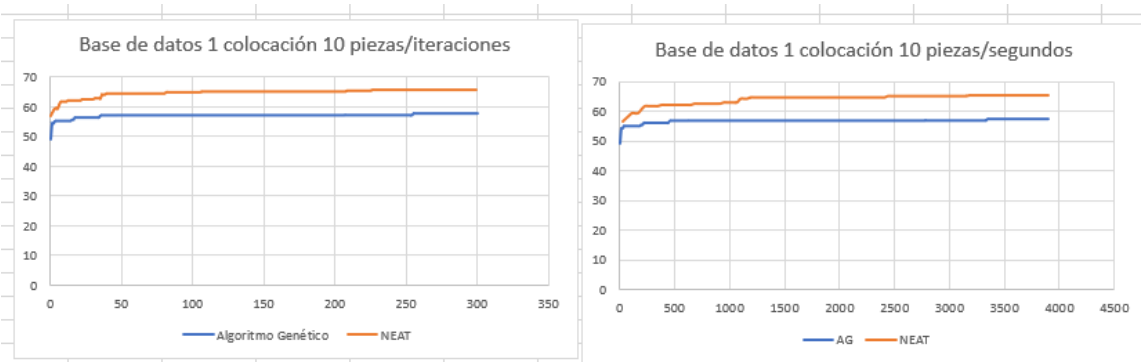


Figura 5-2: Porcentaje de área utilizada para colocar la base de datos 1 con 10 piezas de cada tipo para un eje fijo de 300 pixeles en la versión original del algoritmo y el prototipo base que usa NEAT en función del número de iteraciones y de los segundos usados.

Cabe destacar ante estos resultados que el algoritmo genético utilizado tiene libertad de selección de rotaciones y orden al igual que el prototipo; esto le permite encontrar los

resultados óptimos, pero limita su velocidad de mejora y su eficacia inicial con respecto a otras versiones que usen heurísticas como las descritas en el estado del arte. El principio de estas heurísticas es proponer colocaciones iniciales más estables que una distribución aleatoria, limitar sus ordenaciones o limitar las rotaciones posibles. Sin embargo, al usar estas heurísticas, aunque se aumente el empuje inicial del algoritmo se limita sus resultados mucho más que con el prototipo con NEAT, que da libertad total de elección de rotaciones y colocaciones a la red.

Una vez comparada la diferencia entre el algoritmo genético y el prototipo inicial, se incrementa el número de piezas en diferentes cantidades, se repite el proceso de aprendizaje del prototipo, documentando la diferencia en el área ocupada para cada cantidad de piezas de cada tipo en el set. Esta prueba es necesaria dado que en principio una de las ventajas de automatizar el trazado de patrones de corte es que permite una cierta escalabilidad, y que para un mayor número de piezas a distribuir debería de ser capaz de obtener una colocación que utilice un porcentaje mayor del área. Además, esto nos permite hacernos una idea de cómo de escalable es el algoritmo para piezas sencillas, viendo en qué medida aumenta el tiempo de ejecución de cada ciclo por cada pieza añadida.

Realizando este experimento se utilizan tres conjuntos de piezas en función del número de piezas de cada tipo, siendo en total 60, 120 y 180 piezas y se generan las gráficas en la Figura 5-3. Comparando la primera muestra con la segunda, que tiene el doble de piezas se observa, que por la estrategia de utilización utilizada y tal vez también en cierta medida por el aumento de la topología de la red el tiempo de ejecución hace más que duplicarse, acercándose a triplicarse en la última iteración. Este aumento en el tiempo de ejecución supone una diferencia tan grande en función del número de piezas que para la tercera muestra se decide reducir la población a la mitad y ejecutar solo una décima parte de los ciclos que ejecutan las otras dos muestras, viendo que por ello no se alcanza la fase de madurez en su aprendizaje, pero que a su vez, para estas pocas ejecuciones el tiempo de ejecución es casi igual al de la segunda muestra, siendo cada ciclo, en promedio, 26.5 veces más lento que en la muestra inicial, aunque de tener una población del mismo tamaño sería alrededor de 50 veces más lento para una muestra solo 3 veces más grande.

También cabe destacar, vistas las dos primeras muestras, cómo en la fase de crecimiento de estas funciones logarítmicas se crea un aumento muy notable de del porcentaje de área en uso, siendo solo necesarias para ver esta diferenciación 30 iteraciones. Esto nos hace ver la importancia de maximizar el número de piezas por colocación, que siempre que se dé margen al algoritmo para aprender los patrones óptimos para el número de piezas repercutirá en un resultado mejor.

Se observa por último que los resultados en las ordenaciones iniciales son peores para un número mayor de piezas. Esto puede deberse a la menor exploración del espacio muestral en la primera iteración dado que se mantiene el mismo tamaño de población para un número de colocaciones posibles mayor. Esto no representa un problema siempre y cuando se alcance el punto de madurez del aprendizaje, pero en caso de no poseer suficiente tiempo o capacidad de computación, como es el caso para la tercera muestra analizada en la Figura 5-3, es preferible utilizar un número de piezas menor para alcanzar unos resultados óptimos.

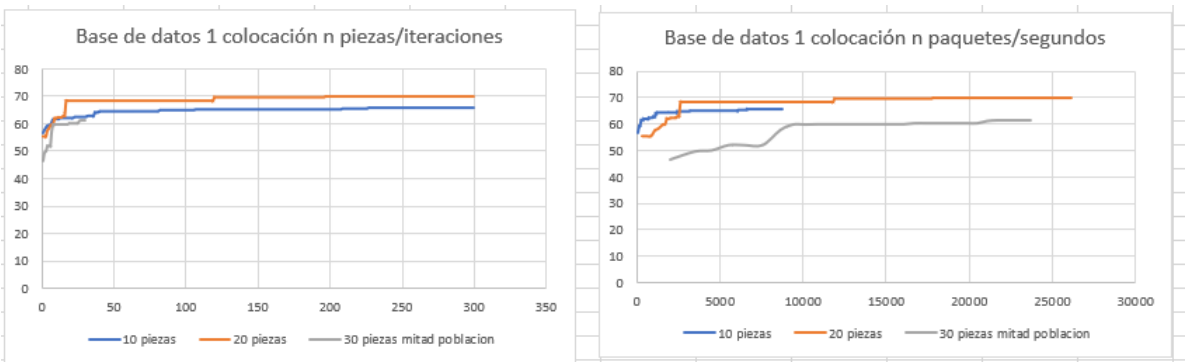


Figura 5-3: Porcentaje de área utilizada en el prototipo para diferentes cantidades de piezas de cada tipo de la base de datos 1 en función del tiempo de ejecución y el número de iteraciones con 300 pixeles en el eje fijo.

Una vez concluidas las pruebas para con el algoritmo NEAT base se prueba a añadir más datos de entrada, comprobando si mejora su capacidad de colocación de piezas o su velocidad de aprendizaje añadiendo:

- Neuronas indicando la posición relativa de los segmentos que contiene cada tipo de pieza.
- Neuronas indicando la cantidad restante a colocar de cada tipo de pieza.
- Un número diferente de neuronas describiendo los segmentos ya colocados.

Llamaremos al algoritmo base utilizado V1 y a las dos primeras modificaciones V2 y V3 respectivamente. V3 incluye los cambios realizados para V2, dado que con estos tiene más sentido conocer el numero restante de piezas a colocar. La última modificación, visto que el número de segmentos utilizado en un principio para describir las piezas era más alto de lo necesario (utilizando para la versión base 50 segmentos), pasará a utilizar únicamente el número de segmentos de la pieza con más segmentos de la base de datos, de manera que, en lugar de describir la capa más superficial colocada, describe la posición de la última pieza colocada, que podría ser suficiente para elegir la colocación. Dadas estas características se llamará a esta versión la versión reducida y para la base de datos 1 esta se aplicará con 4 segmentos como entrada. Esta última modificación será especialmente útil para la base de datos más compleja, que con total seguridad necesitará más segmentos para describir cada uno de los objetos ya colocados.

Como se observa en la Figura 5-4, cada una de las versiones añade datos de utilidad y acelera el aprendizaje gradualmente, tanto por iteraciones como por tiempo, cabe destacar como la versión 3 no solo consigue el resultado optimo antes que la versión 2 sino que además utiliza menos tiempo incluso teniendo más neuronas de entrada.

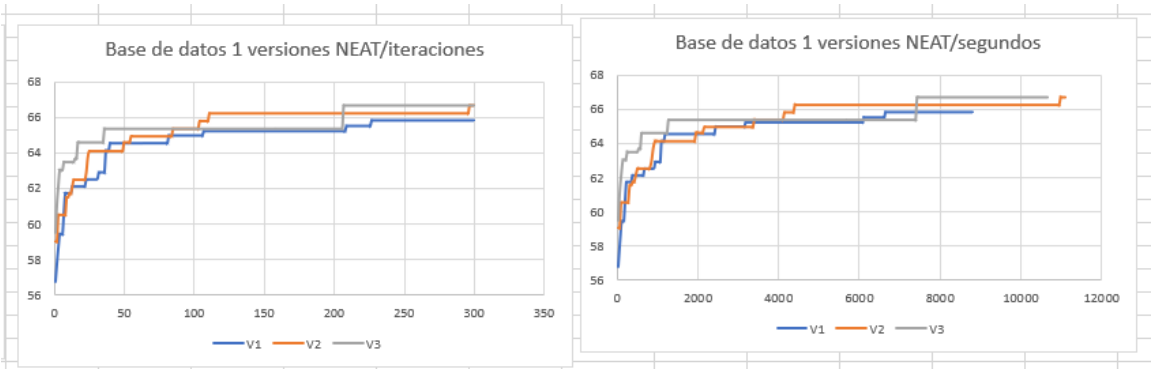


Figura 5-4: Porcentaje de área utilizada para las versiones 1, 2 y 3 del prototipo usando la base de datos 1 en función del tiempo de ejecución y el número de iteraciones con 300 pixeles en el eje fijo.

Al comparar la versión reducida con el algoritmo base en la Figura 5-5 los resultados son menos claros; se observa que en el muy corto plazo la versión reducida alcanza unos resultados iniciales mejores, pero que la versión base lo alcanza rápidamente y en función del tiempo los resultados finales quedan equilibrados por el hecho de que las ejecuciones del algoritmo reducido son más rápidas en tiempo de ejecución, pero descubren nuevas posibilidades más despacio en función de las iteraciones.

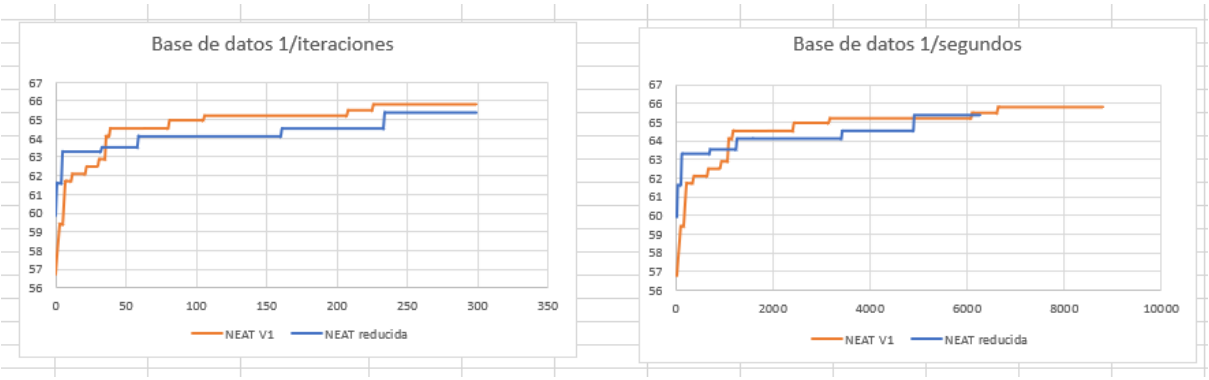


Figura 5-5: Porcentaje de área utilizada para las versiones base y reducida del prototipo usando la base de datos 1 en función del tiempo de ejecución y el número de iteraciones con 300 pixeles en el eje fijo.

5.2 Pruebas con piezas complejas

Se crea una base de datos con piezas más complejas que las anteriores y representante de un ejemplo realista del problema. Para este documento se ha escogido como caso ejemplo las piezas necesarias para una camiseta, construidas en base a la descripción dada en la referencia [22]

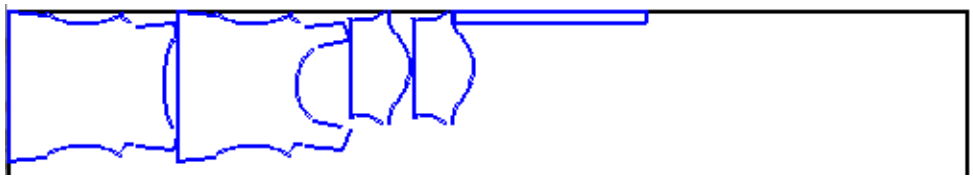


Figura 5-6: Piezas utilizadas para componer una camiseta en la base de datos 2 en un marco de 500 pixeles de ancho.

Visto que en este caso se usan más piezas de un tipo que de los demás, en concreto en el caso de las piezas de tipo manga se necesitan 2 por camiseta, se denomina a partir de aquí paquete al conjunto de piezas necesarias para componer una camiseta y en lugar de alterar el número de piezas de cada tipo usando el mismo número de cada uno como se hizo para la base de datos 1, las variaciones en esta base de datos se realizarán con respecto al número de paquetes a introducir.

Se ejecutan las pruebas realizadas para la base de datos de piezas simples, comprobando en qué medida el hecho de que las piezas sean más complejas afecta a la velocidad de aprendizaje del algoritmo que usa NEAT y si el hecho de que las piezas sean más complejas altera la diferencia entre los resultados de cada algoritmo obtenida en la anterior base de datos. Visto que la duración de las pruebas es mucho mayor en esta base de datos por cada paquete añadido, las pruebas de aprendizaje por versiones se realizan en un principio con 5 paquetes de piezas.

Tras realizar esta prueba se observa en la Figura 5-7 que el resultado en función de la versión NEAT utilizada se invierte, alcanzando los peores resultados con la versión antes óptima y los mejores con la versión base del prototipo, siendo aun notablemente peor la solución final aportada por el algoritmo genético. También se observa que la diferencia entre el tiempo de ejecución del algoritmo genético y la versión base del prototipo se acercan mucho, visto que en la base de datos 1 la versión base tardaba algo más del doble en ejecutarse y en este experimento tarda solo un 20% más. Esto se atribuye al aumento en la complejidad de las piezas, que supone un mayor tiempo invertido en la comprobación de colisiones con respecto al tiempo dedicado a construir las estructuras genéticas y de NEAT. Esto va en aumento según aumentan el número de piezas y su complejidad, motivo por el cual es razonable plantear el uso del prototipo incluso si su ejecución es algo más lenta que la versión con el algoritmo genético, visto que para los casos que requieran mayor capacidad de computación el tiempo para gestionar estas estructuras llegará a ser despreciable.

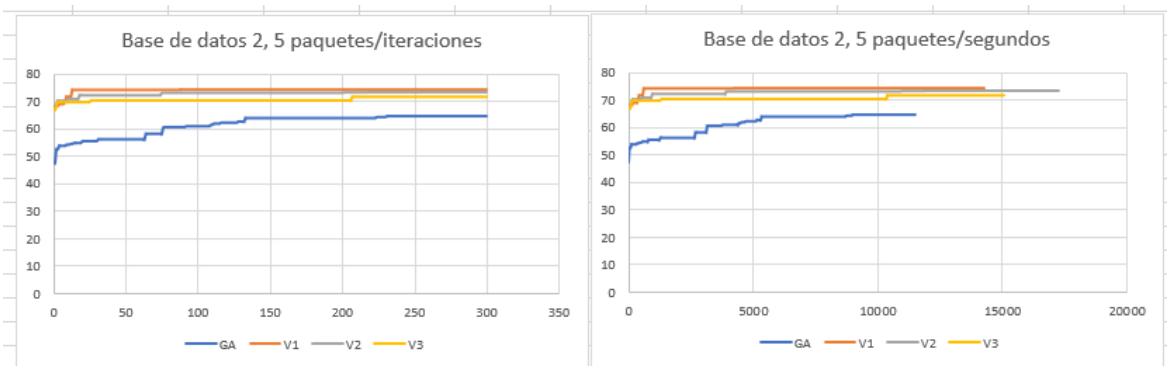


Figura 5-7: Porcentaje de área utilizada para las versiones con algoritmo genético y con NEAT 1, 2 y 3 usando la base de datos 2 con 5 paquetes en función del tiempo de ejecución y el número de iteraciones con 300 pixeles en el eje fijo.

Viendo los malos resultados que aportan para la colocación de 5 paquetes las versiones con mayor complejidad se decide probar una colocación de 10 paquetes para comprobar si se mantienen estos resultados para piezas como las de la base de datos 2 y, como se observa en la Figura 5-8, no es así. Viendo los positivos resultados de las versiones de mayor complejidad en el experimento con 10 paquetes, que se asemejan más a los resultados de estas versiones en la primera base de datos, se llega a la conclusión de que estas versiones del prototipo funcionan mejor para un número de piezas mayor y que para uno lo suficientemente pequeño, solo suponen una carga que ralentiza el algoritmo.

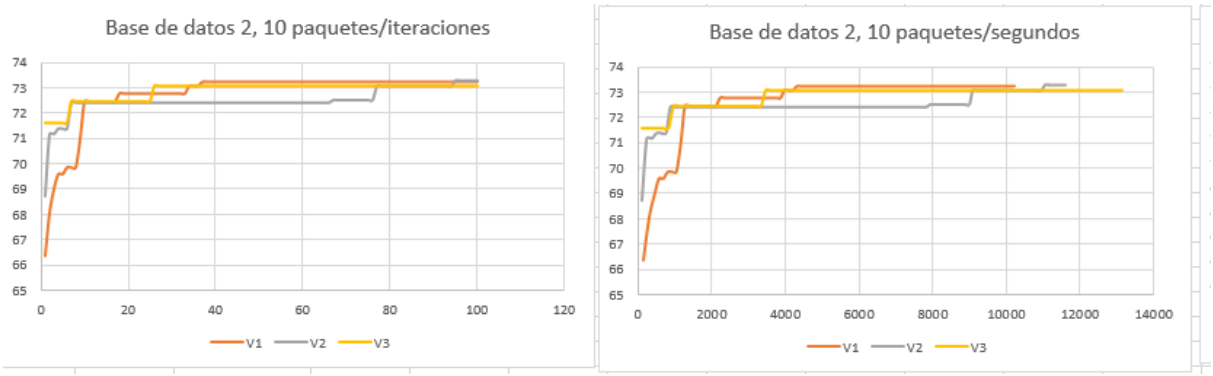


Figura 5-8: Porcentaje de área utilizada para las versiones 1, 2 y 3 del prototipo usando la base de datos 2 con 5 paquetes en función del tiempo de ejecución y el número de iteraciones con 300 pixeles en el eje fijo.

En la Figura 5-9 se observan los resultados del experimento con la versión reducida del algoritmo, resultados que siguen el patrón del caso de la base de datos 1 y que nos llevan a la conclusión de que el uso de la versión reducida del algoritmo no ofrece ni un mejor resultado, ni un mejor tiempo de ejecución para un resultado lo suficientemente bueno, por norma general.

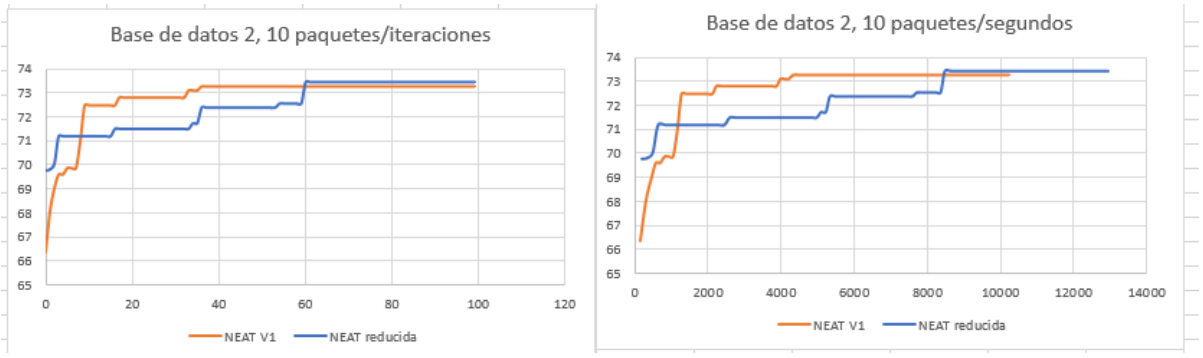


Figura 5-9: Porcentaje de área utilizada para las versiones base y reducida del prototipo usando la base de datos 2 con 10 paquetes en función del tiempo de ejecución y el número de iteraciones con 300 pixeles en el eje fijo.

Finalmente se adopta la configuración óptima según el resultado de las pruebas, que será la versión 3 del prototipo, y se utiliza para intentar lograr el mejor resultado posible alterando para ello las dimensiones del eje fijo del material madre. El resultado de esta prueba es vital, pues representa un ejemplo de uso real de este software y a partir de los resultados de este experimento veremos cómo se desenvuelve en óptimas condiciones.

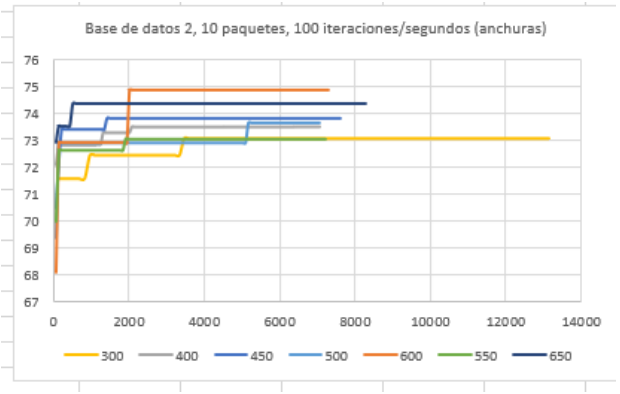


Figura 5-10: Porcentaje de área utilizada para la versión 3 del prototipo usando la base de datos 2 con 10 paquetes en función del tiempo de ejecución en 100 iteraciones para distintas dimensiones de eje fijo.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El objetivo de este trabajo de fin de grado ha sido proponer un nuevo algoritmo para resolver el problema de optimización del trazado de patrones de corte. Este nuevo algoritmo se apoya en aquellos métodos ya conocidos para resolver este problema mutando la resolución de este problema mediante un algoritmo genético y las tres fases de comprobación de colisiones, ordenación y colocación. La inclusión de NEAT como parte del proceso de ordenación es el cambio principal con respecto a este algoritmo original y es a raíz de este cambio que surgen los demás en el planteamiento de la resolución de este problema, buscando optimizar esta nueva estructura a las condiciones del problema.

Una vez analizadas las técnicas usadas originalmente para la resolución del problema y planteados los cambios necesarios para construir el nuevo prototipo, este documento también pretende analizar si estos cambios suponen una mejora real con respecto al algoritmo original. Tras llevar a cabo esta comprobación de forma experimental y bajo los criterios de Harrison para analizar algoritmos para la resolución de este problema, se llega a la conclusión de que este algoritmo sí que supone una mejora con respecto al uso del algoritmo genético para la fase de ordenación.

Las mejoras por parte del prototipo comienzan en la existencia de una mayor libertad de elección de ordenaciones, que vienen a causa de la ausencia de necesidad ni recomendación de heurísticas que limiten la cantidad de colocaciones a considerar, dada su tendencia a memorizar patrones de ordenación y rotación y a aplicarlas en las situaciones en las que encajen de forma adecuada dada una colocación de las piezas anteriores determinada. Esta diferencia se ve muy claramente en las pruebas realizadas, al ver el lento aprendizaje de un algoritmo genético si se le permite ordenar y rotar libremente las piezas.

El prototipo ya en la primera generación genera patrones de ordenación al usar una red neuronal como estructura, en lugar de usar un array predispuesto con las colocaciones. La tendencia a la repetición de un cierto patrón con respecto a la ordenación y la rotación de las piezas parece ofrecer para los casos analizados un uso más eficaz del espacio, incluso si esos patrones de colocación y rotación no son los óptimos. Esto es así por el hecho de que esta estructura facilita un cierto patrón en su generación de resultados de forma natural, al tender a repetir una misma colocación y rotación al enfrentarse a una misma situación y a asignar una rotación similar en los elementos del mismo tipo. Podría considerarse que esta tendencia equivaldría a otras heurísticas utilizadas para un algoritmo genético; sin embargo, es una tendencia que lleva, sin limitar la exploración del espacio muestral, a un mucho más rápido resultado prometedor del que mutar para encontrar la ordenación óptima.

Por estos motivos y tantos otros revisados a lo largo de este documento se alcanzan unos resultados satisfactorios, que aun a falta de probar el algoritmo para más tipos de piezas y en mayores tiempos de procesamiento, reflejan lo prometedora que es esta tecnología y lo útil que puede alcanzar a ser en la resolución de este problema y dan a ver las diferencias que encuentra con respecto a la resolución del problema mediante un algoritmo genético, dejando ver las nuevas ventajas que esta estructura trae para la resolución del problema.

6.2 Trabajo futuro

El prototipo desarrollado en este trabajo de fin de grado representa las primeras pruebas utilizando una tecnología bastante nueva como es NEAT para este problema. Es por eso por lo que la mayor parte de heurísticas utilizadas se basan en la lógica y la intuición y que aún quedan muchas estrategias por probar que con total seguridad exprimirán todavía más el potencial que esta tecnología tiene para resolver este problema. Además, existen muchas combinaciones posibles de piezas que tan vez se hubieran comportado de diferente manera o hubiesen requerido otras combinaciones de neuronas para alcanzar sus mejores resultados en estos experimentos. Es por estos motivos que apunto a que para desarrollar este prototipo en el futuro sería necesario una experimentación más exhaustiva con las entradas a la red neuronal del algoritmo NEAT, concretamente la introducción de valores más procesados que los utilizados en este experimento, como podrían ser:

- Valores que indiquen la proporción de llenado de un número de filas imaginarias que conformen el plano. Estos podrían ser delimitados además por dos valores: la proporción de área de la fila ocupada por figuras o la posición del elemento más avanzado en la fila, si se utilizan estrategias de colocación como Bottom-left.
- El identificador de los últimos tipos de piezas colocados; con este dato el algoritmo podría funcionar de una forma similar a la resolución del problema por programación lineal antes vista, pero en lugar de conocer varios patrones de corte aplicables a una fila o columna, tendría una red entrenada para la predisposición a colocar con la estrategia de colocación usada ciertas combinaciones de elementos.
- Los mismos datos de entrada utilizados en el experimento, pero filtrados de forma que en lugar de los últimos segmentos colocados, se introduzcan solo los colocados más arriba y más a la derecha, esto aplicando también la colocación Bottom-left.

Además de probar estas entradas también resta probar otras estrategias y heurísticas de colocación además de Bottom-left, sin olvidarnos de aquellas solo usables para piezas más regulares, con las cuales también podría ser interesante probar el algoritmo.

Ni el algoritmo de optimización del trazado de patrones de corte del mercado ni el prototipo desarrollado son capaces de simular un rollo de material lo bastante largo como para contener todas las prendas que se demandan de una fábrica en un día; es por eso que existen estudios dedicados a calcular qué piezas de ropa y de que talla, si estamos hablando de ropa, deben colocarse con que otros antes de aplicar el algoritmo para satisfacer la demanda sin necesidad de utilizar únicamente un único patrón de corte que contenga la proporción adecuada de cada pieza demandada para toda la fábrica. La utilización de un algoritmo como el descrito que apoye al prototipo podría ser útil para generar, en lugar de uno solo, una secuencia de patrones de corte óptimos para varias combinaciones de piezas, tal como se hace en la resolución por Gilmore y Gomory y, una vez hallados estos patrones de corte, hallar las combinaciones que cumplan con la demanda.

Por último, sería también razonable invertir en el desarrollo de un sistema para la introducción de las piezas con la estructura demandada por el algoritmo, de manera que la construcción de estas se haga de forma dinámica y teniendo en cuenta la eficacia del algoritmo, pero también de una forma intuitiva que permita el uso de esta herramienta en un entorno industrial realista en el que, en la medida de lo posible, se facilite al usuario una construcción cómoda y sencilla de estas piezas. Esta última propuesta podría ser especialmente útil para el apropiado uso de esta herramienta, más aún teniendo en cuenta

que este problema será más sencillo o complicado para el algoritmo en función de la complejidad de las piezas construidas y que la estructura dedicada a estas solo supone una mejora en la eficacia de los cálculos si se construyen con sentido.

Referencias

[1] Timo Rissanen, 2013, Zero-waste fashion design: a study at the intersection of cloth, fashion design and pattern cutting, University of Technology, Sídney.

[2] Diana Andrea Peña Calderón, Juan Pablo Orejuela Cabrera, Cristiam Andrés Gil González, 2017, El Problema de patrones de corte, clasificación y enfoques http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1692-82612017000100112

[3] A. Ogunranti, Ayodeji E. Oluleye, 2016, Minimizing Waste (Off-cuts) Using Cutting Stock Model: The Case of One-Dimensional Cutting Stock Problem in Wood Working Industry, Omniascience, Estados Unidos.

[4] H. Hideki Yanasse, M.J. Pinto Lamosa, 2004, An integrated cutting stock and sequencing problem, Elsevie.

[5] Saad M.A. Suliman, 2001, Pattern generating procedure for the cutting stock problem, Bahrain, Elsevie.

[6] Sisca Octarina, 2019, Gilmore and Gomory model on two-dimensional multiple stock size cutting stock problem, IOP publishing, Indonesia.

[7] Sisca Octarina, 2018, Implementation of pattern generation algorithm in forming Gilmore and Gomory model for two-dimensional cutting stock problem, IOP publishing, Indonesia.

[8] Hamish T Dean, 2002, Minimizing Waste in the 2-Dimensional Cutting Stock Problem, Nueva Zelanda, University of Canterbury.

[9] Charlotte Jacobs-Blecha, 1990, Apparel advanced manufacturing technology demonstration, U.S. Defense logistics agency.

[10] Alberto Márquez, Fundamentos Geometría Computacional Tema 1: Cierre Convexo, http://asignatura.us.es/fgcitig/contenidos/gctem1ma.htm

[11] Carlos Amaral, Joao Bernardo, and Joaquim Jorge, 1990, Marker-Making using automatic placement of irregular shapes for the garment industry, Lisboa, Portugal, Pergamon Press.

[12] A. Ramesh Babu, N. Ramesh Babu, 2000, A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms, India, Elsevier.

[13] Jens Egeblad, 2008, Heuristics for Multidimensional Packing Problems, University of Copenhagen.

[14] Shunji Umetani, Mutsunori Yagiura, Takashi Imamichi, Shinji Imahori, Koji Nonobe y Toshihide Ibaraki, A guided local search algorithm based on a fast neighborhood search for the irregular strip packing problem, Japón.

[15] Pedro Rocha, 2019, Robust NFP generation for Nesting problems, Porto, Portugal

[16] F. Ellis, 2001, Introduction to GIS, Australia, The University of Melbourne, https://geogra.uah.es/patxi/gisweb/GISModule/GIST_Raster.htm#fig9

[17] H. Terashima-Marín, P. Ross, C.J. Farías-Zárate, E. López-Camacho, M. Valenzuela-Rendón, 2008, Generalized hyper-heuristics for solving 2D Regular and Irregular Packing Problems .

[18] Lijun Wei, Wee-Chong Oon, Wenbin Zhu, Andrew Lim, 2011, A skyline heuristic for the 2D rectangular packing and strip packing problems, Hong Kong, Elsevier.

[19] Daniel Domović, Tomislav Rolich, Marin Golub, 2018, Hiper-Huristic approach for improving marker efficiency, Zagreb, Croatia, University of Zagreb, AUTEX Research Journal.

[20] Kenneth O. Stanley, Risto Miikkulainen, 2002, Evolving Neural Networks through Augmenting Topologies, E.E.U.U., The MIT Press Journals.

[21] M. Sempere, F. Gallego, F. Llorens, M. Pujol, R. Rizo, 2014, Un entorno de aprendizaje neuro evolutivo: Screaming Racers, Alicante, España.

[22] G. Cabanillas, 2014, Patrón de camiseta básica con manga: DIY, Patrones Mujer, Madrid.