****

**FACULTAD DE INGENIERÍA DE SOFTWARE**

**CICLO 2019-02**

**COMPLEJIDAD ALGORÍTMICA**

**SECCIÓN**

**WV72**

**TEMA DEL PROYECTO:**

**ALGORITMO DE CORTE/EMPAQUETADO**

**PROFESOR DEL CURSO:**

**LUIS MARTIN CANAVAL SANCHEZ**

**TRABAJO PRESENTADO POR LOS ALUMNOS:**

**ANDRE GABRIEL GONZALES SONCCO**

**DIEGO ALONZO HILARIO CALLUPE**

**LUIS ALBERTO ORTIZ CENTENO**

**UPC VILLA, AGOSTO DEL 2019**

**INTRODUCCIÓN:**

Actualmente el problema de corte y empaquetado es muy común en los sectores industriales como los que trabajan con cuero y calzado. El problema se enlaza al corte de las plantillas de calzado, dichos cortes son realizados teniendo en cuenta varios factores: el contorno irregular del cuero, la textura del cuero o defectos inherentes y el contorno no rectangular del corte a ser efectuado. La tecnología actual hace que el diseño de los cortes sea generado por una computadora, pero dichos cortes son realizados por un maestro cortador, este mismo observa el menor desperdicio del cuero. Este proceso no asegura que la elección del corte sea el mejor. También este problema se encuentra en otras industrias como lo son de: papel, vidrio, plásticos, confección de ropas, derivados del cartón, etc. Principalmente al momento de utilizar la materia prima no siempre se aprovecha el 100%, lo cual siempre habrá un desperdicio luego de los cortes y al momento que se realiza el empaquetamiento no se optimiza el espacio que deben ocupar los recortes. Para estas empresas el material desperdiciado equivale a dinero desperdiciado que aporta al encarecimiento del producto final y puede terminar en un producto no competitivo como consecuencia.

La magnitud del problema de corte y empaquetamiento 2D se encuentra en su utilización en varios sectores productivos, como tecnología clave para la optimización de residuos de la materia prima incurrida en el proceso de producción que como resultado aumenta el nivel de competitividad de las industrias.

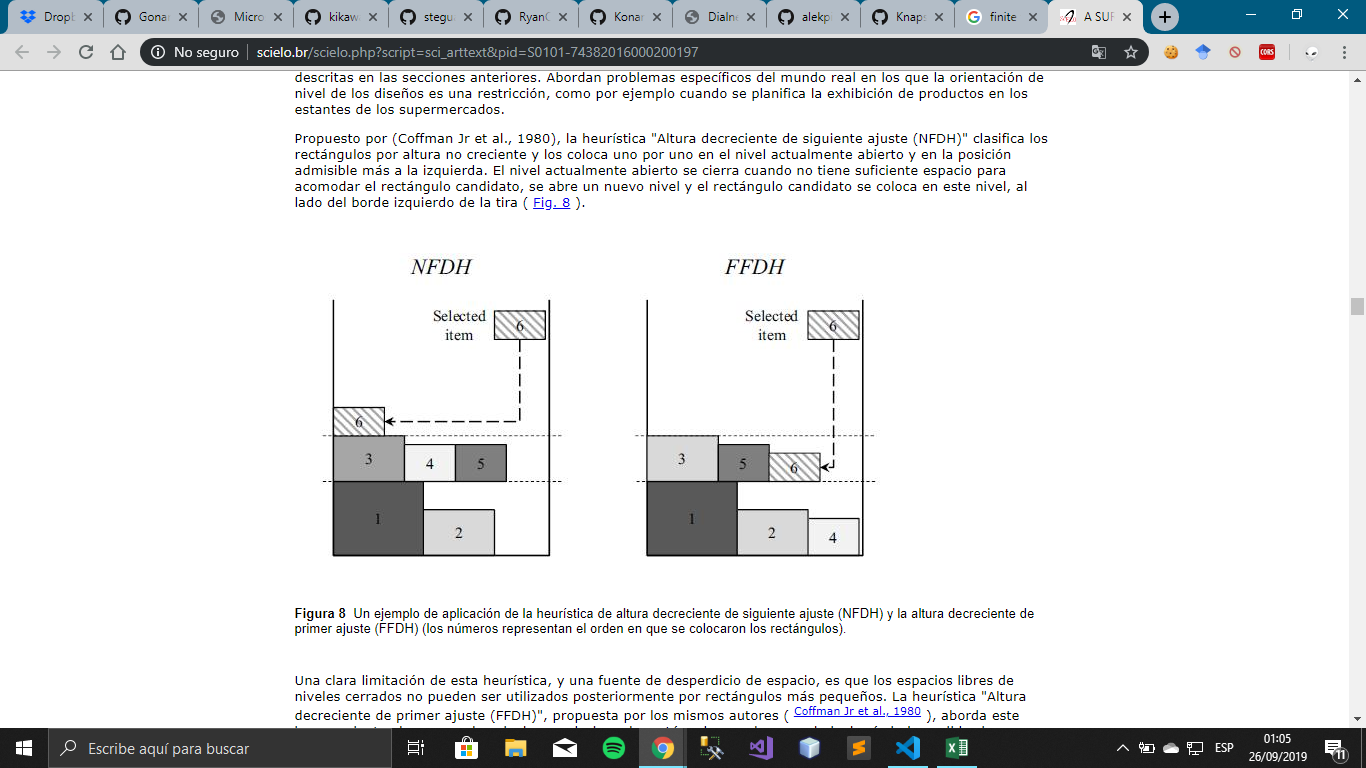
En este proyecto se implementará 1 algoritmos de corte y 2 de empaquetamiento para poder darle solución a un problema real, mostrando su funcionamiento y complejidad, así como también poner en práctica los conocimientos adquiridos en el curso en términos de complejidad algorítmica.

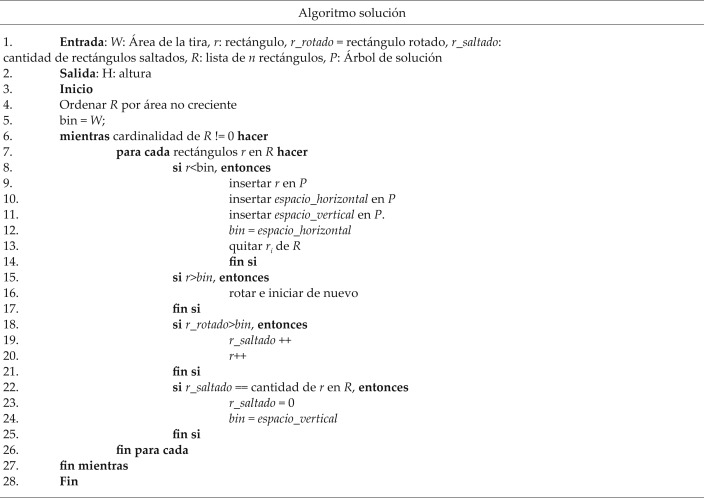
**OBJETIVOS:**

El objetivo del proyecto es poder plantear y analizar los algoritmos de corte y empaquetamiento obteniendo los tiempos en que resuelven el problema cada uno y la complejidad algorítmica (BigO). Además, poder visualizar el desperdicio que origina cada algoritmo. Estos puntos servirán para poder mostrar las ventajas y desventajas de los algoritmos de corte y empaquetamiento, y poder seleccionar cual es el más eficiente.

**MARCO TEÓRICO DE LAS ESTRATEGIAS UTILIZADAS:**

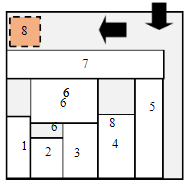
Next-Fit Decreasing Height (NFDH):

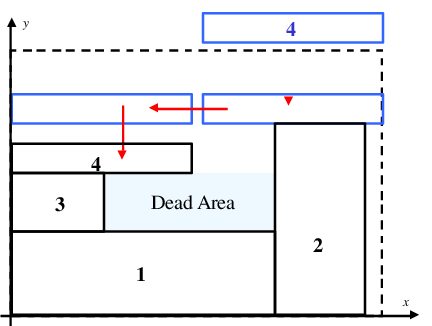
Esta forma parte de la heurística basada en niveles de Strip Packing Problem, que tiene como idea central la colocación de rectángulos en niveles , es decir, cortes paralelos de guillotina en todo el ancho de la tira. Esta heurística fue propuesta por Coffman Jr en 1980. El algoritmo clasifica los ítems (rectángulos) por altura de manera decreciente, donde el ítem con mayor altura define la altura de cada nivel, y los coloca uno por uno en el nivel actual y en la posición más a la izquierda. Se cambiará de nivel cuando en el nivel actual no se tiene suficiente espacio para acomodar al siguiente rectángulo, donde este pasará al nivel superior colocado en la posición más a la izquierda.

*Algoritmo solución de Strip Packing Problem*

Bottom Left (BL):

Este algoritmo de empaquetamiento también sigue la heurística de Strip Packing. Consiste en ordenar todas las piezas cortadas de manera no creciente, luego se empaqueta de forma que la siguiente pieza vaya lo más cerca posible de la parte inferior, y luego lo más cerca posible de la izquierda sin superponerse con ninguna otra pieza embalada. Hay que tener en cuenta que este algoritmo no rota las piezas a empaquetar simplemente las acomoda.





**ANÁLISIS DE LA COMPLEJIDAD DE LAS ESTRATEGIAS:**

* Análisis de la complejidad algorítmica de Next-Fit Decreasing Height (NFDH):

Primero, se requiere de un tiempo O(nlogn) ya que se ordena los ítems con el algoritmo de ordenamiento Quicksort de forma descendente. La función del algoritmo NFDH sin el ordenamiento es de O(n). En conclusión, el algoritmo en su totalidad tiene un tiempo de O(nlogn + n) que es igual a O(nlogn). Adicionalmente, con la función %timeit se puede apreciar que el tiempo de ejecución dependiendo al número de ítems(rectángulos) que se le agrega.

**Algoritmo de ordenamiento con tiempo asintótico O(nlogn):**

def partition(arr, ini, fin):

i = ini - 1

pivote = arr[fin][1]

for j in range(ini,fin):

if arr[j][1] > pivote:

i = i + 1

arr[i],arr[j] = arr[j],arr[i]

arr[i+1],arr[fin] = arr[fin],arr[i+1]

return i+1

def quicksort(arr, ini, fin):

n = len(arr)

if ini < fin:

pi = partition(arr, ini, fin)

quicksort(arr, ini, pi-1)

quicksort(arr, pi+1, fin)

**Algoritmo NFDH con tiempo asintótico O(n):**

def nfdhSPP(arrItem, anchBin, altoBin):

n = len(arrItem)

quicksort(arrItem, 0, n-1)

bins = 0

y = altoBin

########

arrItem[0] = (arrItem[0][0], arrItem[0][1], 0, y - arrItem[0][1], bins)

newEst = arrItem[0][3]

for i in range(1, n):

varx = arrItem[i-1][2] + arrItem[i-1][0] + arrItem[i][0]

if varx <= anchBin: ## Si el ancho del item es menor o igual al anchoBin, agregar

arrItem[i] = (arrItem[i][0], arrItem[i][1], arrItem[i-1][2] + arrItem[i-1][0], y - arrItem[i][1], bins)

else:

y = newEst ## Actualiza posicion y para un nuevo estante

vary = y - arrItem[i][1]

if vary >= 0: ## Si el alto del item es mayor o igual a 0, agregar

newEst = newEst - arrItem[i][1]

arrItem[i] = (arrItem[i][0], arrItem[i][1], 0, vary, bins)

else:

newEst = arrItem[i][3]

y = altoBin

bins = bins + 1 ## Agregamos una plancha nueva

arrItem[i] = (arrItem[i][0], arrItem[i][1], 0, y - arrItem[i][1], bins)

return arrItem, bins

Se puede apreciar que en este algoritmo de ordenamiento está incluido en la función donde se encuentra el algoritmo de empaquetado, por lo que el tiempo asintótico total es O(nlogn)

* Análisis de complejidad algorítmica de Bottom Left:

La complejidad de este algoritmo está dado por O(n^2)

**CONCLUSIONES (VENTAJAS Y DESVENTAJAS):**

Sobre Next-Fit Decreasing Height (NFDH):

* Ventajas:
* Se vuelve más eficiente a medida de que la dimensión de los ítems se incrementan.
* Desventajas:
* Una clara desventaja de este algoritmo es el desperdicio de espacio, ya que los espacios libres de los niveles cerrados ya no pueden ser ocupados.
* En comparación con las otras dos heurísticas de nivel, NFDH es la que genera mayor desperdicio.
* Este algoritmo no presenta características de rotación de piezas para optimizar el espacio de la plancha.
* El tiempo de respuesta de ejecución es mayor en comparación con los otros algoritmos.

En conclusión, el algoritmo NFDH genera mucho desperdicio de espacio, por lo que se recomienda usar otros algoritmos de empaquetamiento y de preferencia que tengan un buen tiempo asintótico como O().