

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

E.A.P DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

Un algoritmo de búsqueda adaptativa aleatoria y golosa para la resolución del problema de cortes

TESIS Para optar el Título Profesional de: INGENIERO DE SISTEMAS E INFORMÁTICA

AUTOR:

Dante Ganoza Salazar

Ursula Carola Solano Lazo

ASESOR: David Mauricio Sánchez

LIMA – PERÚ 2004

Agradecimientos

Al término de este trabajo que representa la culminación de nuestra tesis, queremos agradecer a las personas que han ayudado a que esta tarea llegue a su fin.

En primer lugar a Dios, por habernos permitido llegar al final de esta etapa.

A nuestro asesor, Dr. David Mauricio, quien nos guió, apoyo y ayudó para llegar a la culminación de este trabajo y nos hizo ver la importancia de la tesis.

A nuestros amigos y personas especiales, quienes nos brindaron su apoyo, ayuda y entusiasmo en todo momento.

Por último, pero no menos importante, queremos agradecer a nuestros padres, por su incesante trabajo y esfuerzo por darnos una mejor educación.

*A nuestros padres por ser nuestros ejemplo,
y esforzarse por nuestro futuro.
A nuestros maestros, quienes nos guían para
afrontar ese futuro*

UN ALGORITMO DE BÚSQUEDA, ADAPTATIVA, ALEATORIA Y GOLOSA PARA LA RESOLUCIÓN DEL PROBLEMA DE CORTES

Dante Ganoza, Ursula Solano ⁽¹⁾

Facultad de Ingeniería de Sistemas e Informática, UNMSM

Av. Germán Amézaga s/n, Ciudad Universitaria, Lima, Perú

(1) e-mail: dantegs@hotmail.com, usolano_lazo@hotmail.com

Resumen

Dado un conjunto de requerimientos lineales y un número ilimitado de barras de metal (u otro material) de tamaño estándar, con dimensión mayor a la de los requerimientos. El Problema de Cortes consiste en realizar cortes sobre las barras de tamaño estándar, de tal manera que se obtengan todos los requerimientos con el menor número de barras de tamaño estándar y el menor desperdicio posible. El problema es NP-Difícil, y presenta diversas aplicaciones en los diversos sectores de la industria, tales como la maderera, metal, plástico, etc.

La presente Tesis, muestra un Procedimiento de Búsqueda Aleatoria, Adaptativa y Golosa (GRASP), para la resolución del problema de cortes.

Experimentos numéricos realizados del algoritmo propuesto sobre 100 problemas-test, reportan una eficiencia, promedio del 95.4% para un parámetro de relajación de 0.5 y 2000 iteraciones.

El software implementado consta de 4 módulos importantes: ingreso de datos necesarios para la realización de los cortes, Algoritmos Golosos FFD (First Fit Decreasing) y BFD (Best Fit Decreasing), GRASP y Reportes.

Palabras Claves: Problema de cortes, algoritmo Goloso, GRASP

AN ALGORITHM OF GREEDY RANDOMIZED ADAPTATIVE SEARCH PROCEDURE TO SOLVE CUTTING STOCK PROBLEM

Dante Ganoza, Ursula Solano (1)

Ability of Engineering of Systems and Computer Science, UNMSM

Av. Germán Amézaga s/n, University City, Lima, Perú

(1) e-mail: dantegs@hotmail.com, usolano_lazo@hotmail.com

Abstract

Given a group of lineal requirements and a limitless number of metal bars (or another material) of standard size, with more dimension to that of the requirements. The Cutting Stock Problem consists on carrying out cuts on the bars of standard size, in such a way that all the requirements are obtained with the smallest number of bars of standard size and the minor waste possible. The problem is NP-hard, and it presents several applications in the different sectors of the industry, such as the lumberman, metal, plastic, etc.

The present Thesis shows a Procedure of Random Search, Adaptive and Greedy to solve the Cutting Stock Problem.

Carried out numeric experiments of the algorithm proposed on 100 problem-tests, they report efficiency, average of 95.4% for a parameter of relaxation of 0.5 and 2000 iterations.

The implemented software consists of 4 important modules: entrance of necessary data for the realization of the cuts, Greedy Algorithms FFD (First Fit Decreasing) and BFD (Best Fit Decreasing), GRASP and Reports.

Key words: Cutting Stock Problem, Greedy algorithm, GRASP

TABLA DE CONTENIDO

	Pág.
INTRODUCCIÓN	13
CAPÍTULO I	16
GENERALIDADES	16
1.1 Optimización	16
1.2 Técnicas exactas	17
1.3 Técnicas heurísticas	17
1.3.1 Algoritmo Goloso (Greedy)	17
1.4 Técnicas Metaheurísticas	18
1.4.1 GRASP	18
1.4.2 Algoritmo Genético	18
1.4.3 Tabú Search	19
1.4.4 Simulated Annealing	20
CAPÍTULO II	22
EL PROBLEMA DE CORTES	22
2.1 Definición del Problema	22
2.2 Variantes	24
2.2.1 Cortes en dos dimensiones	25
2.2.2 Cortes en tres dimensiones	25
2.2.3 Cortes On Line	26
2.2.4 Cortes por lotes	26
2.3 Aplicaciones	26
2.4 Métodos Existentes	28
2.4.1 Métodos Exactos	28
2.4.1.1 Generación de columnas aplicando Programación lineal	28
2.4.2 Heurísticas	30
2.4.2.1 NF	30
2.4.2.2 NFD (Next Fit Decreasing)	30
2.4.2.3 FFD (First Fit Decreasing)	30

2.4.2.4 BFD (Best Fit Decreasing)	30
2.4.2.4 FFD Eficiente	31
2.4.3 Metaheurísticas	31
2.4.3.1 Optimización Metaheurística	31
2.4.3.2 Algoritmo Genético	31
2.5 Aplicativos	32
2.5.1 Cups-F (Cutting planning software for Film)	32
2.5.2 1D Stock Cutter	32
CAPÍTULO III	34
ALGORITMO GRASP	34
3.1 Definición	34
3.2 Procedimientos GRASP	35
3.3 Fases del Algoritmo GRASP	35
3.3.1 Ingreso de Datos: Leer (instancias)	35
3.3.2 Condición de Parada	35
3.3.3 Fase de Construcción: Construcción (Solución _k)	36
3.3.4 Fase de Mejora: Mejorar (Solución _k)	38
3.3.5 Registrar la mejor Solución: Registrar (Solución _k)	39
CAPÍTULO IV	42
UN ALGORITMO DE BÚSQUEDA ADAPTATIVA ALEATORIA Y GOLOSA PARA LA RESOLUCIÓN DEL PROBLEMA DE CORTES	42
4.1 GRASP para el problema de cortes	42
4.2 Estructura de Datos	45
4.2.1 Arreglo de Cortes	45
4.2.2 Arreglo de Barras	45
4.2.3 Arreglo de residuos	45
4.3 Algoritmo GRASP Construcción	45
4.3.1 Construcción de la lista candidata RCL	46
4.3.2 La constante de relajación α	46
4.3.3 Presentación del Algoritmo GRASP Construcción	47
4.4 Registrar_Mejor_Solución	49
4.5 Complejidad del algoritmo GRASP	50
CAPÍTULO V	52
DESCRIPCIÓN DEL SISTEMA	52
5.1 Requerimientos mínimos de SW y HD	52
5.1.1 Configuración de hardware mínimo	52
5.1.2 Configuración de software mínimo	52
5.2 Descripción del Software	53

5.2.1 Estructura del Sistema	53
5.2.2 Módulos del Sistema	55
5.2.2.1 Módulo de Ingreso	55
5.2.2.2 Módulo de algoritmos	56
5.2.2.3 Reporte	58
CAPÍTULO VI	60
EXPERIMENTOS NUMÉRICOS	60
6.1 Requerimientos de Hardware y Software	60
6.2 Instancias de Pruebas	60
6.3 Resultados Numéricos	62
6.4 Análisis de Resultados	70
6.4.1 Eficiencia	70
6.4.2 Tiempo	73
CAPÍTULO VII	76
CONCLUSIONES Y RECOMENDACIONES	76
APÉNDICE A - Parámetros usados para los experimentos numéricos	79
APÉNDICE B - Instancias de prueba	80
APÉNDICE C - Manual de Usuario	99
BIBLIOGRAFIA	112

ÍNDICE DE FIGURAS

<i>Número</i>	<i>Pág.</i>
Figura 2.1: Ejemplo de Cortes	24
Figura 2.2: Ejemplo de cortes en dos dimensiones	25
Figura 2.3: Ejemplo de cortes en tres dimensiones	26
Figura 3.1: Algoritmo GRASP	35
Figura 3.2: Criterio goloso de selección del requerimiento	36
Figura 3.3: Criterio GRASP de selección del requerimiento	37
Figura 3.4: Influencia del parámetro α en el criterio de selección de la GRASP	37
Figura 3.5: Algoritmo GRASP Construcción	38
Figura 3.6: Algoritmo GRASP Mejorado	39
Figura 4.1: Criterio FFD de selección de barra a cortar	43
Figura 4.2: Algoritmo GRASP para el problema de cortes	43
Figura 4.3: Algoritmo Ordena	44
Figura 4.4: Algoritmo GRASP Construcción para el problema de cortes	47
Figura 4.5: Función que llena RCL	48
Figura 4.6: Función Eliminar dato	48
Figura 4.7: Registrar Mejor Solución	49
Figura 5.1: Estructura de CorteSoft	53
Figura 5.2: Ventana de presentación de CorteSoft	54
Figura 5.3: Ventana de la Ayuda de CorteSoft	54
Figura 5.4: Ventana del menú principal	55
Figura 5.5: Ventana de Ingreso de parámetros	56
Figura 5.6: Ventana del Algoritmo FFD	56
Figura 5.7: Ventana del Algoritmo BFD	57

Figura 5.8:	Ventana del Algoritmo GRASP	57
Figura 5.9:	Ventana de Reporte	58
Figura 6.1:	Eficiencia promedio para valores de a	71
Figura 6.2:	Promedio de las mejores eficiencias de GRASP	72
Figura 6.3:	Comparación de eficiencias	73
Figura 6.4:	Promedio de los tiempos de GRASP	74

ÍNDICE DE TABLAS

<i>Número</i>		<i>Pág.</i>
Tabla 2.1:	Ejemplo de demanda de corte	23
Tabla 2.2:	Requerimiento para corte	23
Tabla 2.3:	Requerimiento muestra	29
Tabla 6.1:	Lista de instancias de Prueba	61
Tabla 6.2:	Resultados del Grupo # 1	62
Tabla 6.3:	Resultados del Grupo # 2	63
Tabla 6.4:	Resultados del Grupo # 3	63
Tabla 6.5:	Resultados del Grupo # 4	64
Tabla 6.6:	Resultado del Grupo: STADLER	66
Tabla 6.7:	Resultados del Grupo: PAUTA	67
Tabla 6.8:	Resultados del Grupo: DEGRAEVE	67
Tabla 6.9:	Resultados del Grupo: GOULIMIS	68
Tabla 6.10:	Resultados del Grupo: HINTERDING & KHAN	68
Tabla 6.11:	Resultados del GRUPO: ALOISE & MACULAN	69
Tabla 6.12:	Tabla resumen de las eficiencias promedio	70
Tabla 6.13:	Tabla de resultados excluyendo al grupo Stadler	71
Tabla 6.14:	Comparación de Eficiencias de los Algoritmos	72
Tabla 6.15:	Tabla resumen de los tiempos promedio	73

INTRODUCCIÓN

Los modelos matemáticos y las técnicas de programación matemática nacieron para dar respuesta a la necesidad de mejorar los procesos productivos y se han venido aplicando mayoritariamente a la organización y distribución de los recursos físicos.

Se encuentra una aplicación que mejora los procesos productivos, organización y distribución de los recursos físicos en la resolución de problemas de optimización, los cuales consisten en encontrar la mejor solución (o solución óptima) de un conjunto de soluciones. Dentro de los problemas de Optimización se tiene el Problema de Cortes.

Se tienen un número ilimitado de barras (u otro material a cortar) de longitud fija L y se requiere cortarlas en barras más pequeñas de longitud $l_1, \dots, l_m$ con $l_i < L$, el Problema de Cortes consiste en realizar cortes de tal manera que el desperdicio B , es decir, la cantidad que sobra después de los cortes, sea mínima. Además debemos obtener el menor número m de barras de longitud L al realizar los cortes.

El objetivo de la presente tesis es desarrollar un sistema computacional basado en la técnica GRASP para resolver el Problema de Cortes con aplicación no sólo en barras de fierro, sino en otros materiales lineales como rollos de papel, cintas de video, tela, etc. Además, demostrar la ventaja del uso del algoritmo GRASP sobre las heurísticas Golosas FFD y BFD.

En el primer capítulo se presentan los conceptos básicos introductorias del tema de tesis; en el segundo capítulo se define el problema de cortes; en el tercer capítulo se

detalla la técnica GRASP; en el cuarto capítulo se propone la técnica GRASP para el problema de cortes; en el quinto capítulo se describe el sistema desarrollado; en el sexto capítulo se muestran los resultados de los experimentos numéricos y finalmente se indican las conclusiones y recomendaciones de la investigación.

CAPÍTULO I

GENERALIDADES

CAPÍTULO I

GENERALIDADES

En este capítulo, se presentan conceptos relevantes que ayudarán a entender puntos que se abordarán posteriormente.

1.1 Optimización

La optimización es el proceso de encontrar la mejor solución, llamada solución óptima de un conjunto de soluciones [II 97].

Se define a la Optimización, como el “proceso encaminado a obtener el mejor resultado posible bajo un conjunto de circunstancias determinadas” [CM 92].

Matemáticamente la optimización se define de la siguiente manera:

“Encontrar $x = (x_1, \dots, x_n)$, tal que maximice o minimice cierta función $f(x)$ ”

Donde:

?? x es un vector n -dimensional llamado vector decisión

?? $f(): \mathbb{R}^n \rightarrow \mathbb{R}$ es llamada función objetivo

Las técnicas de optimización hacen uso del cálculo diferencial para encontrar los valores máximos o mínimos de una función. A continuación se muestra una clasificación de dichas técnicas: *polinómicamente programables* cuando existen algoritmos de resolución que dan una respuesta tras realizar un número de operaciones, *no polinómicamente programables*, es decir, problemas *NP-Difícil*.

1.2 Técnicas exactas

Son aquellas técnicas de optimización que permiten obtener el valor óptimo absoluto de la función de coste u función objetivo. A estas técnicas pertenecen los procedimientos de optimización tradicionales.

1.3 Técnicas heurísticas

Las técnicas heurísticas tratan de métodos o algoritmos exploratorios en los cuales las soluciones se descubren por la evaluación del progreso logrado en la búsqueda de un resultado final.

Las técnicas heurísticas se definen también como técnicas de optimización basadas en procedimientos sistemáticos de prueba que ofrecen una buena solución (no necesariamente la óptima absoluta) para problemas donde el espacio de soluciones es indeterminado o lo suficientemente amplio como para que no pueda ser recorrido totalmente en un tiempo aceptable.

Se han propuesto diversos procedimientos para resolver los problemas de optimización sin tener que realizar una búsqueda exhaustiva sobre el conjunto de posibles soluciones.

A continuación se describe una técnica heurística conocida, el algoritmo Goloso.

1.3.1 Algoritmo Goloso (Greedy)

El algoritmo Greedy o goloso es un algoritmo que toma decisiones de corto alcance, basado en información inmediatamente disponible, sin importar consecuencias futuras. Se usa generalmente para resolver problemas de optimización. Eso hace que en general sean algoritmos eficientes y fáciles de implementar.

En este algoritmo se cambia aleatoriamente los valores de verdad de los átomos. Seleccionan una variable al cual cambian su valor de verdad incrementando el número de cláusulas satisfechas. El objetivo es siempre aumentar las cláusulas

satisfechas. Este procedimiento se repite hasta que ya no se consigue una mejora. Si está es la solución, el algoritmo se detiene con éxito [Hu 00].

1.4 Técnicas Metaheurísticas

Estas técnicas no garantizan la obtención de la solución óptima de un problema; sin embargo, sí generan resultados bastante aceptables en un tiempo reducido sin necesidad de un conocimiento profundo del problema a resolver. Estos tipos de procedimientos se suelen aplicar a espacios de búsqueda muy extensos, por lo que las aplicaciones paralelas son muy utilizadas para solucionar este problema de espacios tan amplios y difíciles de explorar [CM 92].

A continuación se describen las técnicas metaheurísticas más conocidas.

1.4.1 GRASP

GRASP (Greedy Randomized Adaptive Search Procedure) es una técnica de los años '80 que tiene como objetivo resolver problemas difíciles en el campo de la optimización combinatoria. Esta técnica dirige la mayor parte de su esfuerzo a construir soluciones de alta calidad que son posteriormente procesadas para obtener otras aún mejores [FR 95].

1.4.2 Algoritmo Genético

Los Algoritmos Genéticos (AG) son técnicas de búsqueda inspirados en los mecanismos de selección y genética natural, que combinan la capacidad de supervivencia de individuos (posibles soluciones o reglas de inferencia), con operadores genéticos. Mantienen una población que se renueva c. La nueva generación se construye probabilísticamente, a partir de los individuos más aptos de la generación anterior (operador de selección), combinando su “código genético” (a través de operadores de crossover y mutación) [Go 89]. A partir de este esquema, una población puede evolucionar proponiendo nuevos y mejores individuos.

Los algoritmos genéticos son procedimientos de búsqueda basados en selección y genética natural. En los algoritmos genéticos, un conjunto de soluciones del problema es mejorado para crear nuevos miembros usando pedazos y trozos de soluciones de pruebas de ajuste de soluciones existentes con la nueva parte aleatoria ocasional (mutación genética), añadidas. El problema, en muchos procedimientos de búsqueda, es que ellos se “estancan” en la óptima local, porque al parecer encontraron soluciones que parecen ser buenas, pero no son las mejores que podrían ser encontradas. Añadiendo “mutación genética”, los algoritmos genéticos son menos susceptibles de quedar estancados en las soluciones sub óptimas. Las búsquedas aleatorias, generando soluciones aleatorias y evaluándolas, evitan el problema de estancarse en soluciones sub óptimas, pero son generalmente ineficientes. Los algoritmos genéticos utilizan información histórica y combinan buenas soluciones en el conjunto de soluciones para generar nuevas soluciones, en las cuales una mejora debería ser esperada [Wa 99].

1.4.3 Tabú Search

Este procedimiento ha sido tradicionalmente usado en problemas de optimización combinatoria; el cual guía una heurística de escalamiento descendiente con el objetivo de continuar la exploración y evita un retroceso a un óptimo local desde el cual ya se ha salido. En cada iteración un movimiento admisible se aplica a la solución actual, transformándola de esta manera en una solución vecina con menor costo. Son permitidas las soluciones que incrementan la función de costo, mientras que el movimiento inverso está prohibido para algunas iteraciones, con el objetivo de evitar la ocurrencia de ciclos.

Durante el proceso de búsqueda, los movimientos son almacenados en una lista Tabú, representando la memoria de los pasos previos del algoritmo. Se cuenta con un proceso de mejoría para determinar cuando las restricciones Tabú pueden ser sobrescritas [Gl 89].

1.4.4 Simulated Annealing

Simulated Annealing es una técnica similar a la programación genética. Está basada en observaciones del proceso de enfriado de metales. El término *Annealing* está relacionado con la manera con la que los metales líquidos son enfriados lentamente para asegurar un estado de energía mínima. Se puede decir que el algoritmo de Simulated Annealing enfría la solución lentamente hasta alcanzar el objetivo más bajo posible [Il 97].

CAPÍTULO II

EL PROBLEMA DE CORTES

CAPÍTULO II

EL PROBLEMA DE CORTES

En este capítulo se definen el Problema de Cortes, las variantes, las aplicaciones en la industria u otros campos, los métodos existentes, luego se concluye con los aplicativos comerciales más conocidos.

2.1 Definición del Problema

Se tienen un número ilimitado de barras de longitud fija L y se requieren cortarlas en barras más pequeñas de longitud $l_1, \dots, l_n$ con $l_i < L$. El Problema de Cortes consiste en realizar cortes de tal manera que debemos obtener el menor número m de barras de longitud L y que el desperdicio B , es decir, la cantidad sobrante de los cortes, sea mínima.

Como ya se mencionó, el material a cortar puede no ser sólo barras de metal, sino también otros como madera, plástico, vidrio, papel, etc.

El problema a tratar se restringe a una sola dimensión, es lineal. El Problema de Cortes en una dimensión, es conocido en la literatura como *The Cutting Stock Problem (CSP)*, el cual consiste en realizar cortes sobre las barras para obtener los requerimientos con el menor número de barras [MD 02].

Dada una demanda de corte, que precisa ser suplida, para esto cortamos las barras de tamaño fijo.

Tabla 2.1: Ejemplo de demanda de corte

Tamaño	Cantidad
4300	53
2100	28
3720	45
3544.5	20
1290.75	35

Como se observa en la tabla 2.1, la cantidad de barras demandada, es un número entero (se entiende positivos mayores que cero) y el tamaño de las piezas puede ser entero o no.

Se tiene una barra estándar **L**, cuyo tamaño es 120 m., se requiere suplir el siguiente requerimiento:

Tabla 2.2: Requerimientos para cortes

Tamaño	Cantidad
30	1
15	5
45	2
90	1
75	1

Para atender los requisitos se pueden realizar los siguientes cortes:

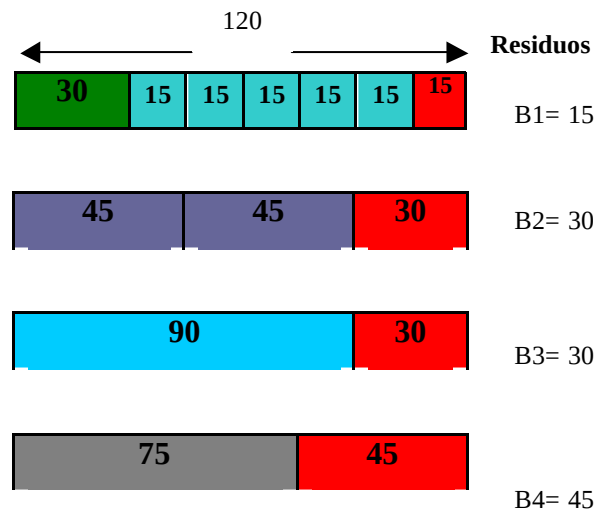


Figura 2.1: Ejemplo de Cortes

De la Figura 2.1 se observa que:

?? Se utilizan $m = 4$ barras estándar de longitud L (L_1, L_2, L_3, L_4)

?? Y el residuo es B : $B_1 + B_2 + B_3 + B_4 = 120m$

?? Índice de desperdicio = $(120 / (120 * 4)) * 100 = 25\%$

El residuo total B es aproximadamente una barra, lo que representa un índice de desperdicio de 25%.

2.2 Variantes

Los problemas de cortes se presentan en una gran cantidad de situaciones. La variedad de los problemas es tan grande como las áreas de aplicación, incluyendo disciplinas como ciencia de gerencia, la ingeniería, matemáticas, informática o la investigación operacional con diversas industrias que incorporan diversos apremios y objetivos.

En muchas industrias fabriles se requiere cortar la materia prima útil para la elaboración de sus productos o desarrollo de sus servicios. Éste es el caso en la industria de metal, la industria de madera, industria de cuero y la industria de textil, donde se requieren cortar planchas de madera, metal, tela, cuero, etc.

Los problemas de cortes no solamente son lineales, sino también de dos, tres y más dimensiones, incluyendo variantes que implican peso y otras características como el ángulo de corte, etc.

Por la dimensión de los requerimientos, el problema de cortes se divide de la siguiente manera:

2.2.1 Cortes en dos dimensiones

El Problema de Cortes en dos dimensiones, consiste en realizar cortes, como su nombre lo dice, bidimensionalmente. Por ejemplo, cortes de cartulina, planchas de madera, vidrio, metal, etc. Al igual que en el Problema de Cortes lineales, se trata de minimizar los desperdicios.

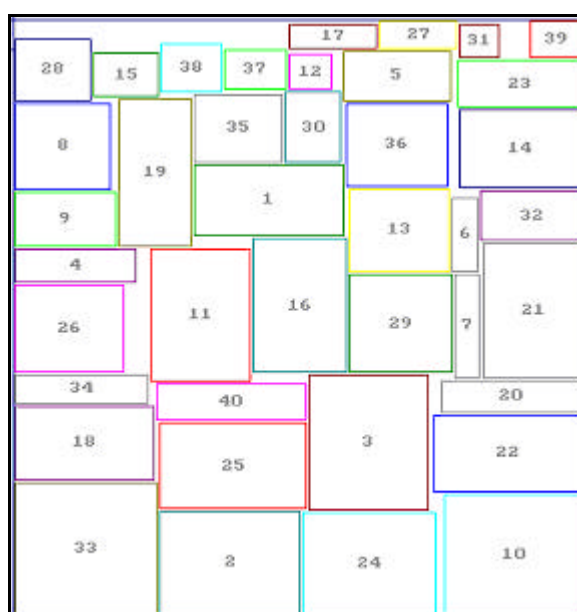


Figura 2.2: Ejemplo de cortes en dos dimensiones

2.2.2 Cortes en tres dimensiones

Esta variante consiste en realizar los cortes espacialmente, es decir, en tres dimensiones (alto, ancho y largo). Ejemplos de ellos son los embalajes, cortes de espuma para colchones, etc.

En el caso del embalaje, el límite total del peso de la carga y los límites del peso de la movilidad pueden considerarse como apremios adicionales o factores reales de la optimización.

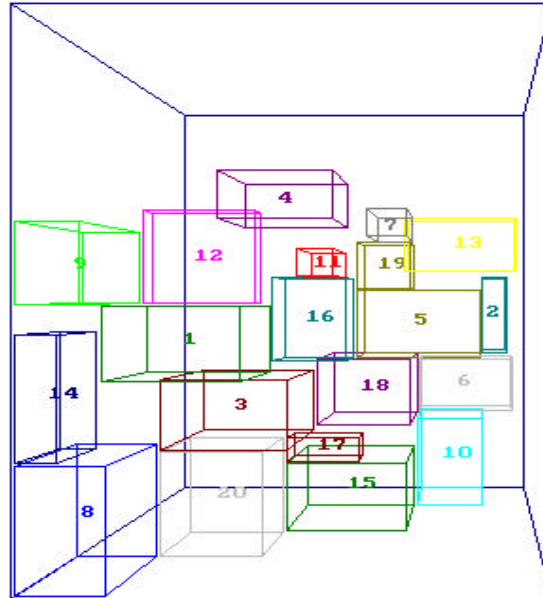


Figura 2.3: Ejemplo de cortes en tres dimensiones

Por el orden en que se realizaran los cortes, se pueden dividir de la siguiente manera:

2.2.3 Cortes On Line

Se realizan los cortes de acuerdo a la llegada del requerimiento en lugar de esperar a que los requerimientos lleguen en su totalidad. Este tipo de cortes genera alto desperdicio del material a cortar.

2.2.4 Cortes por lotes

A diferencia del anterior, los cortes se realizan cuando se tienen todos los requerimientos que se desea cumplir. El desperdicio obtenido es menor que en los cortes on line.

2.3 Aplicaciones

Las industrias tales como constructoras, madereras, papeleras, textiles,

fabricantes de cinta de video y de película de fotografía, etc. tienen problemas al realizar cortes provocando un alto porcentaje de desperdicio. Esto causa una disminución en sus ganancias o en muchos casos pérdida de los recursos.

Por ejemplo, si un cliente realiza un pedido de diversos tamaños de barras de metal a una industria de metal que fabrica barras de un tamaño estándar, esta tendrá que realizar esos cortes hasta cumplir con la demanda. Al no poseer métodos apropiados para ellos, realizan los cortes de acuerdo a su parecer, lo que le provocará, como ya se explicó, pérdida de recursos y disminución de sus ganancias lo que perjudicará su producción.

Los problemas del corte se encuentran en muchas industrias que incorporan diversos apremios por ejemplo, la industria de papel se refiere principalmente al corte de figuras regulares, mientras que en edificios, naves, textil y la industria del cuero irregular, los artículos formados arbitrariamente deben ser embalados.

Las aplicaciones se dan en las siguientes industrias:

?? Industrias fabricantes de films de plástico. Se desea minimizar el desperdicio en el corte de rollos para película (films), esto dependerá además del material, de la longitud y cantidad de la orden [Ha 98].

?? Fábricas que tienen como materia prima el aluminio. Se utiliza este material en construcción civil, por las características que posee, las cuales les ayuda a obtener ventajas sobre la competencia. [Na 97]

También existen las siguientes aplicaciones, que no tienen referencia.

?? *Industria de la Construcción*. Ellas desean minimizar el desperdicio de las barras que utilizan en la construcción de edificios, casas, puentes, etc., permitiéndoles además la reducción de costos.

?? *Industria maderera*. En esta industria la aplicación se da al momento de realizar los cortes estándar de listones de madera de un mismo ancho utilizados para la construcción de diferentes muebles.

- ?? *Industria papelera.* Al momento de suplir los requerimientos de los clientes que solicitan rollos de papel de diferente longitud, por ejemplo, de papel higiénico o papel toalla.
- ?? *Industria de cable,* se refiere aquellas que están en el negocio de venta de rollos de cable, alambre, etc., que prefieren minimizar el sobrante en el corte de cable al realizar las ventas.
- ?? *Empresas dedicadas a cortar tubos* que son usados para tuberías de edificaciones. Las tuberías son de distintos tamaños de acuerdo al tamaño del lugar donde se desea hacer las instalaciones.

2.4 Métodos Existentes

En esta sección se describen los métodos hallados en la literatura para la resolución del Problema de Cortes.

2.4.1 Métodos Exactos

La presente revisión esta basada en el trabajo de Nacif Rocha [Na 97]

2.4.1.1 Generación de columnas aplicando Programación lineal

La solución del problema puede ser descrita de manera directa, generando todos los modelos de corte viables y se utiliza un software que tenga el método Simplex implementado, por ejemplo: LINDO o CPLEX. El inconveniente de este método es que restringe la resolución a problemas pequeños.

En los casos en que los pedazos a ser cortados son pequeños en relación al tamaño de la barra o el número de piezas diferentes es grande, 15 o más tamaños diferentes medido de 1/6 del tamaño de la barra, puede generar millones de modelos viables, (ver tabla 2.3.), sólo la generación de modelos puede llevar algunas horas en un microcomputador de última generación.

Tabla 2.3: Requerimientos muestra

Tamaño	Corte
162	48
1062	32
980	56
931	152
930	16
895	16
865	8
818	428
805	40
645	16
625	40
595	32
529	428
462	436
362	152

Para resolver, se usa programación lineal, relajando la integridad de las variables. Para esto se precisa generar una cantidad de modelos o patrones de corte suficiente para generar una solución básica inicial.

Se denomina generación de columna explícita a la obtención por procesos combinatorios de un subconjunto de patrones de corte viables. Existen 2 algoritmos para implementar esa enumeración. Usando programación dinámica y usando árboles de búsqueda. Esta última es la más eficiente.

Obtenida una solución inicial, el próximo paso es la generación de columnas propiamente dicho. El usar métodos de enumeración explícita no es viable porque el número total de patrones de corte puede ser muy grande y su implementación con vectores no sería óptimo.

Usando Branch and Bound se consigue reducir el tiempo de respuesta hasta en dos veces. Pero también presenta dificultades como por ejemplo, al iniciar el

algoritmo, se precisa que las variables (*li*) estén ordenadas en forma decreciente y conservativa lo que impediría el uso de Quick Sort o ShellSort. Además, se presupone que las entradas para los algoritmos sean enteros, cosa que no es real. En toda la literatura analizada, se puede notar que para obtener la solución final entera es preciso utilizar algún tipo de heurística de redondeo o redondeo simple o alguna técnica Branch and Bound o planos de corte. Se nota que la técnica de la obtención de la solución entera se aplica después que la solución óptima relajada es obtenida.

2.4.2 Heurísticas

2.4.2.1 NF

El algoritmo heurístico NF, consiste en realizar los cortes en la barra si hubiera espacio, de lo contrario se debe utilizar una nueva barra, los objetos son considerados en cualquier orden, es decir, no se necesita realizar un ordenamiento previo.

2.4.2.2 NFD (Next Fit Decreasing)

Es un algoritmo heurístico goloso, cada pieza es suplida cortando la “barra” o recipiente en el cual cabe. Se suplen las piezas comenzando por la de mayor tamaño hasta la más pequeña. Cuando una pieza no cabe en la barra que se está cortando, esta barra ya no es usada (se cierra) y se utiliza (abre) una nueva barra.

2.4.2.3 FFD (First Fit Decreasing)

También es un algoritmo goloso. Este método realiza los cortes previo ordenamiento decreciente de demanda a ser suplida, es decir, se ordenan las piezas de mayor a menor. Cada pieza se suple en la primera barra en la que cabe. Todas las barras pueden seguir siendo usadas hasta que la última pieza sea tratada. Si no cabe en ninguna de las barras, se utilizará una barra nueva.

2.4.2.4 BFD (Best Fit Decreasing)

Es un algoritmo goloso. Este método, al igual que el anterior, hace un

ordenamiento previo de los requerimientos a cumplir. Cada pieza o requerimiento es suplido usando la barra en la que cabe y cuyo tamaño de espacio libre es el que más se aproxima al tamaño de la pieza a tratar. Todas las barras pueden seguir siendo usadas hasta que el último requerimiento sea tratado o suplido. Si no cabe en ninguna de las barras estándares se utilizara una barra nueva. Es mucho más efectiva que el FFD.

El NFD no da buenos resultados en la práctica. El BFD y el FFD poseen comportamientos similares, son eficientes con complejidades de tiempo de $O(n \log n)$ donde n es el número total de piezas o requerimiento a suplir. El FFD en particular produce resultados prácticos muy buenos, en casos en que la demanda de cada tamaño es diferente.

2.4.2.4 FFD Eficiente

Es una versión eficiente del algoritmo FFD, con complejidad $O(n^2 \log_2(N/n))$, (menor a la mencionada) para resolver el problema de cortes con demandas no unitarias, donde n es el número de tamaños diferentes de los requerimientos [MD 02].

2.4.3 Metaheurísticas

2.4.3.1 Optimización Metaheurística

Esta es una combinación de gran alcance para explotar las fuerzas de la regla de la colocación y de los mecanismos de la búsqueda del metaheurístico. La regla de la colocación tiene fuerte influencia en el tiempo del cómputo y calidad de la solución (compensación requerida).

- ?? Supera la heurística simple así como la heurística problema-específico
- ?? Fácil de poner en ejecución
- ?? Aplicable universalmente

2.4.3.2 Algoritmo Genético

Los algoritmos genéticos ofrecen una ventaja en la cual ellos pueden

encontrar un número de buenas soluciones las cuales se presentan al programador.

En este problema existe un número relativamente pequeño de barras que al ser cortadas producen un gran número de piezas. Aunque su espacio de búsqueda sea grande, éste es más pequeño que el espacio de búsqueda de los problemas de stock típico. Además, con los múltiples objetivos para minimizar el desperdicio de cortes, el programador puede preferir tener una opción de soluciones para escoger, puesto que pueden existir otros factores que son difíciles de incluir en el modelo.

2.5 Aplicativos

A continuación se presentan dos aplicativos usados comercialmente de los cuales se tiene poca información.

2.5.1 Cups-F (Cutting planning software for Film)

Software fabricado para solucionar el Problema de Cortes de plástico para films o películas sea para fotografía o video. Trabaja sobre la plataforma Windows 3.1, Windows 95 o superior. La limitación que tiene es que necesita para su funcionamiento el software IGSolver [Ha 98].

2.5.2 1D Stock Cutter

Software desarrollado para planear el diseño de cualquier corte de material lineal. Trabaja sobre la plataforma Windows (9x, NT, 2000). Los requerimientos mínimos de software y hardware son: W95, un procesador Intel 486/DX2, 800*600 SVGA. El método usado para su desarrollo es el del método de Algoritmos Genéticos. Las limitaciones que tiene el programa en su versión Standard es que el máximo total de piezas a cortar es 5000 y el número total de piezas de diferente tamaño es 500 [1].

CAPÍTULO III

ALGORITMO GRASP

CAPÍTULO III

ALGORITMO GRASP

Este capítulo se basa, fundamentalmente, en el trabajo de Mauricio Resende [FR 95]. En él se presenta todo lo referido al Algoritmo GRASP como las fases y criterios que se han tenido presente para la aplicación correcta de esta metodología en la presente tesis.

3.1 Definición

La palabra GRASP proviene de las siglas mencionadas anteriormente, que en español quiere decir *Procedimientos de Búsqueda basados en funciones o Procedimientos Golosos Aleatorios Adaptativos*. El método GRASP se desarrolló a finales de los años ochenta y aunque inicialmente se dieron a conocer en el trabajo de Feo y Resende (1989), ha tenido un desarrollo bastante fecundo para resolver problemas de la optimización combinatoria.

GRASP es un procedimiento iterativo en donde cada paso consiste en una fase de *construcción* y una de *mejora*. En la fase de construcción se aplica un procedimiento heurístico constructivo para obtener una buena solución inicial. Esta solución se mejora en la segunda fase mediante un algoritmo de búsqueda local. La mejor de todas las soluciones examinadas se guarda como resultado final [Mi 94].

?? *Criterio Goloso* de seleccionar la mejor alternativa es relajado de tal manera que se pueda obtener un conjunto de posibles alternativas de solución.

?? El *criterio GRASP* consiste en la selección aleatoria de una decisión desde el conjunto de posibles alternativas.

3.2 Procedimientos GRASP

La Metodología GRASP para construir una solución en lugar de considerar apenas un candidato disponible para el seleccionado (criterio usado por el algoritmo goloso), crea un conjunto de candidatos del que seleccionará uno de ellos aleatoriamente. Por tal motivo, la metodología permite construir varias soluciones diferentes.

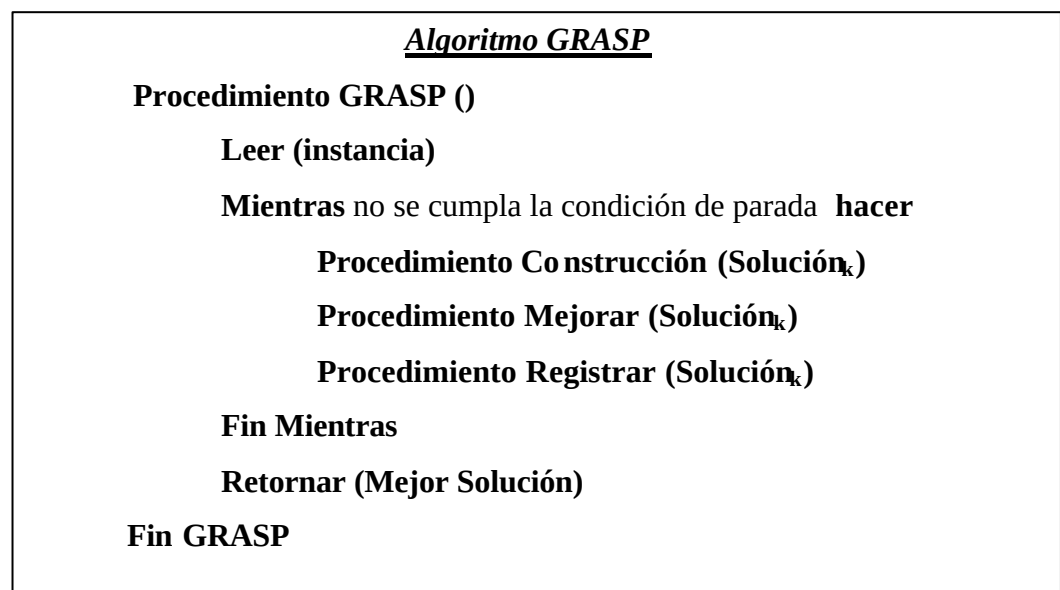


Figura 3.1: Algoritmo GRASP

3.3 Fases del Algoritmo GRASP

3.3.1 Ingreso de Datos: Leer (instancias)

En este paso se ingresa lo siguiente:

- ?? Los datos o instancias que serán procesados por el sistema
- ?? El parámetro de relajación (α)
- ?? Los datos de la condición de parada, que pueden ser: el número máximo de iteraciones a realizar o el tiempo máximo de procesamiento entre otros.

3.3.2 Condición de Parada

En cada iteración la GRASP genera una solución y para que esto no ocurra

indefinidamente, se pueden considerar las siguientes condiciones de parada:

- ?? *Condición de Optimalidad*: es una condición matemática que si es verificada para una solución, entonces se afirma que ésta es óptima.
- ?? *Condición de Tiempo*: se refiere al tiempo máximo de procesamiento del algoritmo; terminado este tiempo, el algoritmo deberá terminar presentando la mejor solución.
- ?? *Condición de número de iteraciones*: se refiere al número máximo de iteraciones que el algoritmo deberá realizar. Al término de éstas, el algoritmo terminará y presentará el mejor resultado hallado.
- ?? *Híbrido*: consiste en la combinación de dos o más condiciones de parada.

3.3.3 Fase de Construcción: Construcción (Solución_k)

Procedimiento que consiste en construir una solución golosa y aleatoria, empleando un criterio Goloso. Este es el siguiente:

Criterio Goloso: Mejor $\{f(x): x \in N\}$

Figura 3.2: Criterio goloso de selección del requerimiento para el problema de cortes

Donde:

f : es una función Golosa

N : conjunto de variables (objetos) no tratadas, cumpliéndose en el inicio que N es exactamente el conjunto E .

El criterio dice: el candidato a ser parte del conjunto solución es el elemento (variable u objeto) no tratado que presenta **mejor** (mayor o menor) valor de la función mérito u objetivo. El criterio GRASP modifica este criterio goloso ampliándolo de forma tal que se pueden construir varios conjuntos de soluciones diferentes. El criterio es el siguiente:

$f = \text{Mejor } \{f(x): x \in N\}$ $d = \text{Peor } \{f(x): x \in N\}$ $RCL = \{x \in N: f - a(B - d) = f(x) = f\}$ $\text{Elegido} = \text{Random}(RCL)$

Figura 3.3: Criterio GRASP de selección del requerimiento para el problema de cortes

El conjunto RCL es el conjunto de candidatos, llamado conjunto de candidatos restringidos. El parámetro a , es llamado el parámetro de relajación y es responsable por transformar la solución construida en aleatoria.

El criterio indica que el candidato a ser parte del conjunto solución es un elemento seleccionado en forma aleatoria desde el conjunto RCL. Éste es un elemento no tratado aún (no incluido en la solución), cuyo valor de la función de mérito no necesariamente es el mejor para los elementos no tratados, pero esta próxima de éste. Este criterio, permite construir soluciones diferentes en cada iteración GRASP.

$a = 0$		Criterio totalmente goloso
$a = 1$		Criterio totalmente aleatorio
$0 < a < 1$		Criterio Goloso + aleatorio

Figura 3.4: Influencia del parámetro a en el criterio de selección de la GRASP

El componente **adaptativo** de GRASP es dado porque, en algunos casos, la función golosa puede variar en cada iteración de la fase de construcción, esto significa que el valor asignado a cada una de las candidatas es actualizado en cada iteración.

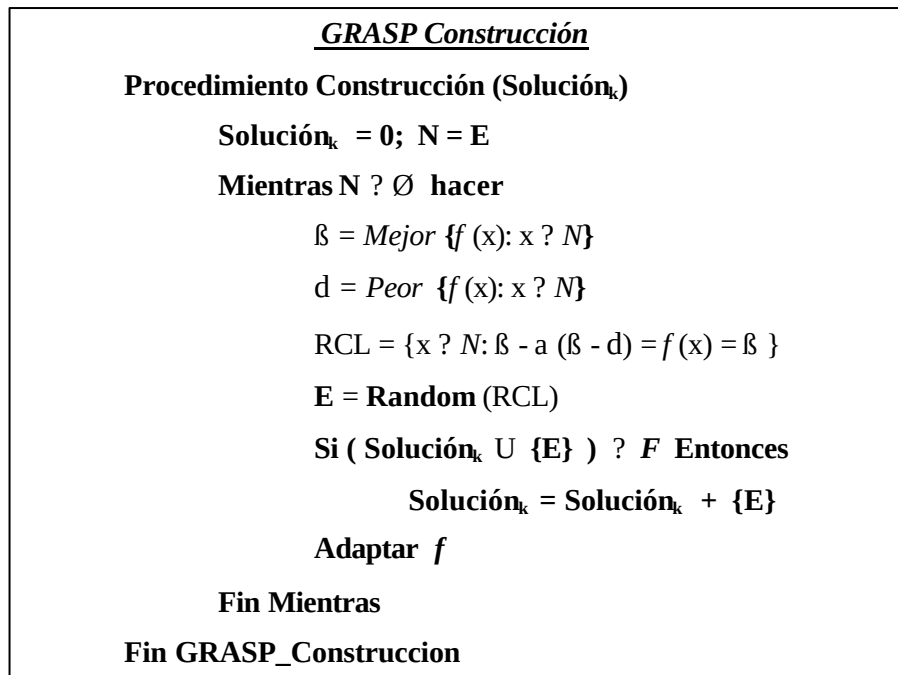


Figura 3.5: Algoritmo GRASP Construcción

Al término de la fase construcción se tiene una instancia **Solución** del problema, aunque, su naturaleza de optimalidad está presta a ser mejorada.

3.3.4 Fase de Mejora: Mejorar (Solución_k)

Como en el caso de las heurísticas, las soluciones construidas por GRASP no son necesariamente óptimas, por lo que es necesaria la aplicación de un procedimiento de mejora de la solución encontrada en la fase de construcción como tentativa de mejorar cada solución hallada.

Lo primero en esta fase es definir una vecindad de soluciones alrededor de la solución hallada anteriormente. Esta vecindad puede ser determinada intercambiando uno a uno los elementos de la solución con aquellos que no pertenezcan a la solución, siempre que se verifique la propiedad **F**.

Construida la vecindad de soluciones, se hace una búsqueda de la mejor solución entre las soluciones de la vecindad y la solución a mejorar. La búsqueda puede limitarse a una simple comparación de soluciones. Si se determina una solución

que mejora la solución a mejorar, entonces se reemplaza la solución y se repite el proceso. El procedimiento termina cuando no es posible mejorar la solución.

Seguidamente se comienza a realizar los reemplazos entre esta nueva y mejorada solución y las probables mejores soluciones que se dan en su nueva vecindad. Para esto se emplea procedimientos de búsqueda, por lo que se incluye esta fase intermedia dentro de la mejora. Este proceso finaliza cuando ya no se pueda encontrar ninguna mejor solución dentro de la vecindad.

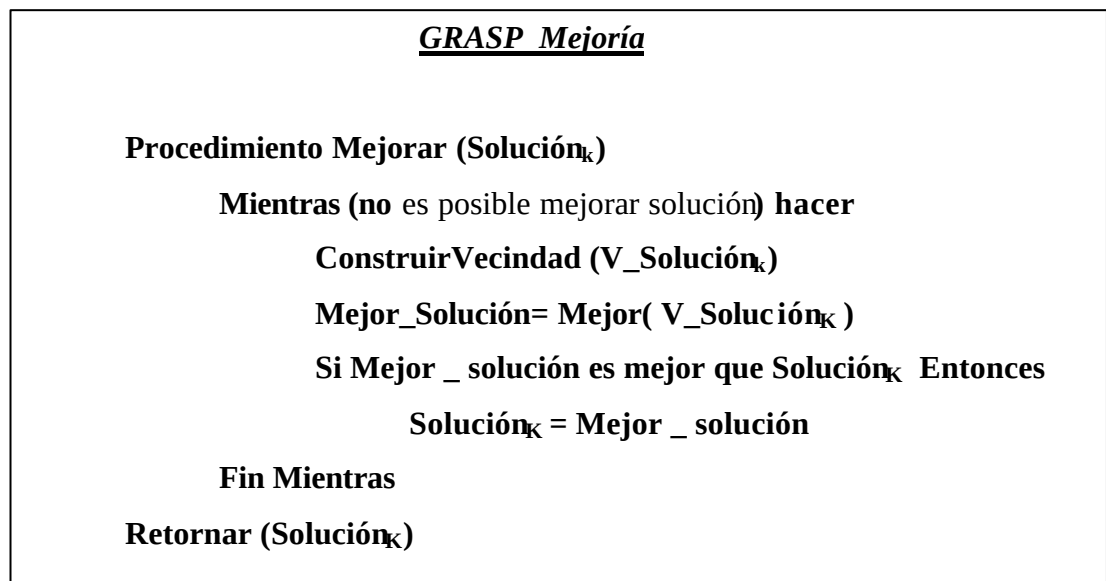


Figura 3.6. Algoritmo GRASP Mejorado

3.3.5 Registrar la mejor Solución: Registrar (Solución_k)

En este procedimiento se compara la solución que se tiene como la mejor, (iteración $i = 1$), con la solución encontrada en la iteración $i+1$; si la solución encontrada en la iteración $i+1$ es mejor que el de la iteración i , entonces la nueva mejor solución será el de la iteración $i+1$, de lo contrario se mantiene la mejor solución. Se continúa así hasta terminar las iteraciones del programa. Al finalizar se registra la mejor solución encontrada.

En resumen el método GRASP presenta dos principales fases. La primera Fase se refiere a la construcción goloso _ aleatorio de una solución y la segunda fase a la

mejora de la solución construida. En cada iteración GRASP se registra la mejor solución encontrada. El algoritmo termina cuando alguna condición de parada es verificada.

La calidad de la solución depende del parámetro de relajación α y de la condición de parada. Estas afirmaciones serán observadas en los experimentos numéricos para el Problema de Cortes.

CAPÍTULO IV

UN ALGORITMO DE BÚSQUEDA ADAPTATIVA ALEATORIA Y GOLOSA PARA LA RESOLUCIÓN DEL PROBLEMA DE CORTES

CAPÍTULO IV

UN ALGORITMO DE BÚSQUEDA ADAPTATIVA ALEATORIA Y GOLOSA PARA LA RESOLUCIÓN DEL PROBLEMA DE CORTES

En este capítulo se propone un algoritmo GRASP para resolver el Problema de Cortes con barras, u otro material lineal, cuya longitud es de tamaño estándar. El Algoritmo propuesto está basado en el trabajo de Mauricio y Delgadillo [MD 02].

4.1 GRASP para el problema de cortes

Se considera la siguiente notación

- ?? **L**: la longitud estándar de barras a utilizar
- ?? **n**: el número total de cortes requeridos
- ?? **E(i)**: el tamaño del requerimiento “i”
- ?? **i**: el requerimiento, donde $0 < i = n$

Se asume que la demanda de cada uno de los requerimientos, es unitario. En caso que la demanda no sea unitaria, bastará replicar el requerimiento cuantas veces sea su demanda.

El criterio que usamos para seleccionar la barra que atenderá a un requerimiento de tamaño E (i) es el Goloso FFD, es decir:

$$k = \text{Min } \{i: E(i) = R_i\}$$

$$i = i + 1$$

Figura 4.1: Criterio FFD de selección de barra a cortar

Donde la barra seleccionada es “ k ”, si el valor de k es $m+1$, entonces se trata de una nueva barra.

El algoritmo desarrollado consta de dos partes principales que son:

- ?? GRASP Construcción
- ?? Registrar Mejor Solución

A continuación se ilustra el algoritmo GRASP en forma genérica.

```

Función Grasp (n, E (), alfa, nIter, L)
{1} Inicio
{2} conta = 0
{3} ordenar (E (), n)
    {4} Mientras conta < nIter hacer
        {4.1} GRASP_Construccion (alfa, n, E (), L)
        {4.2} Registrar_Mejor_Solucion (conta, mmin)
        {4.3} conta = conta + 1
    Fin_mientras
{5} m = mmin
{6} Fin función

```

Figura 4.2. Algoritmo GRASP para el problema de cortes

Como se muestra en el algoritmo los parámetros de entrada son:

- ?? La constante **n** almacena el número de requerimientos, es decir, el número de piezas.
- ?? El arreglo **E ()**, que almacena los tamaños requeridos para cubrir la demanda, es decir el tamaño de cada pieza.

- ?? La constante **L**, almacena la longitud estándar de las barras a utilizar.
- ?? El parámetro de relajación **a** es utilizado en GRASP_contrucción.
- ?? La constante **nIter**, almacena el número máximo de iteraciones permitidas, el cual es usado como criterio de parada.

Comentarios

- ?? La rutina **ordenar ()** esta basado en el algoritmo burbuja doble. La línea {3} ordena los tamaños de mayor a menor de los valores del arreglo E () con la finalidad de optimizar la velocidad del algoritmo. Seguidamente, se muestra la función Ordenar.

```

Función ordenar (vector ( ) , n)
    j = n
    Mientras j > 1 hacer
        i =1
        Mientras j > 1 hacer
            Si Vector ( i ) < Vector( j ) hacer
                Temp = Vector (i)
                Vector (i) = Vector (j)
                Vector (j) = Temp
            Fin si
            i = i+1
        Fin Mientras
        j = j-1
    Fin Mientras
Fin ordenar

```

Figura 4.3. Algoritmo Ordenar

- ?? La Figura de ordenación tiene una complejidad igual a $O(n^2)$ donde **n** es el número de elementos a ordenar.
- ?? Entre las líneas {4.} y {5} se realizan distintas fases del algoritmo GRASP

4.2 Estructura de Datos

Para el desarrollo e implementación del algoritmo GRASP se requiere de determinadas estructura de datos en donde se almacena la información necesaria para la ejecución del programa. A continuación se detallan estas estructuras.

4.2.1 Arreglo de Cortes

Esta estructura vectorial tiene la siguiente disposición:

E :arreglo [1..n]

Donde se almacena los tamaños requeridos de los cortes para cubrir la demanda

4.2.2 Arreglo de Barras

Esta estructura vectorial tiene la siguiente disposición:

B : Matriz [1..m] [1..c]

Donde en cada fila de la matriz **B** contiene los diferentes cortes realizados en dicha barra siendo **m** el número de barras utilizadas para cubrir los requerimientos de la demanda y **c** es una constante que indica el número máximo de cortes realizado por barras.

4.2.3 Arreglo de residuos

Esta estructura vectorial tiene la siguiente disposición:

r : arreglo [1..m]

Donde cada elemento del arreglo **r** contiene los residuos de los cortes realizados de las barras del arreglo **B** siendo **m** en número de barras utilizadas para cubrir los requerimientos de las demandas.

4.3 Algoritmo GRASP Construcción

Es la parte central del programa porque en esta función se halla una solución, teniendo como criterio el algoritmo goloso FFD, pero a diferencia de éste, en lugar de elegir la pieza de mayor tamaño, se genera una lista candidata donde se elige

aleatoriamente un requerimiento o pieza de esta lista que no necesariamente sea el de mayor tamaño.

4.3.1 Construcción de la lista candidata RCL

Para la construcción de la lista candidata se toma los siguientes criterios:

- ?? Encontrar un valor d igual al menor tamaño del requerimiento no atendido.
- ?? Encontrar β igual al mayor tamaño de los requerimientos.
- ?? Encontrar a , la constante de relajación cuyo valor fluctúa entre 0 y 1.

En general, determinar la RCL es igual a todo corte que cumpla con los siguientes requisitos:

$$RCL = \{j \in E(i) : \beta - a(\beta - d) \leq E(j) \leq \beta\}$$

De esta forma ya no sólo se tiene que la solución golosa es la única que conforma la solución final al problema, sino que hay varias soluciones posibles que han surgido del margen que la constante de relajación ha proporcionado. Es desde este punto donde se puede empezar a realizar mejoras o refinamiento al criterio goloso, de una forma metaheurística, a fin de que GRASP arroje una mejor respuesta (en eficiencia) que el algoritmo goloso simple.

4.3.2 La constante de relajación a

La constante de relajación es un valor, el cual es elegido por el usuario, que se encuentra entre 0 y 1. Además nos proporciona un intervalo permisible donde se puede encontrar soluciones óptimas para una programación.

Obsérvese que:

- ?? Si a es igual a 0: el algoritmo GRASP se convierte en el algoritmo goloso. Por eso se denomina relajación totalmente golosa.
- ?? Si a es igual a 1: el algoritmo tendrá un rango muy grande para formar la lista RCL por lo que se denomina relajación totalmente aleatoria.

Para la tesis, mediante pruebas estadísticas, se ha determinado el valor de la constante de relajación $\alpha=0.4$. Para las pruebas test desarrollados en la tesis se han tomado en cuenta tres valores de α : 0.3, 0.4 y 0.5.

4.3.3 Presentación del Algoritmo GRASP Construcción

El algoritmo considera el ordenamiento previo del arreglo de los cortes requeridos, y consiste en la selección de los cortes candidatos comprobando que cumpla con las condiciones para la lista candidata. Veamos la ilustración con el algoritmo GRASP Construcción.

```

GRASP_Construccion (alfa, n, E (), L)
{1} Inicio
{2} m = 0, r (1) = L
{4} Mientras n > 0    hacer
    {4.1} num_rcl = 0
    {4.2}  $\beta$  = E (0)
    {4.3} d = E(n - 1)
    {4.4} llenar_RCL (E (), n, alfa,  $\beta$ , d, rcl (), num_rcl)
    {4.5} num_rcl = Redondeo (num_rcl * Rnd)
    {4.6} resultado= rcl (num_rcl)
    {4.7} eliminar_dato( E(), n, num_rcl )
    {4.8} flag=false , k=1
    {4.9} Mientras k ? m and Flag =false Hacer
        {4.9.1} Si r(k) >= resultado Hacer
            {4.9.1.1} Flag=true
        {4.9.2} Fin Si
        {4.9.3} k=k+1
    {4.10} Fin Mientras
    {4.11} Si Flag = flase Hacer
        {4.11.1} m = m + 1
        {4.11.2} b(k)(1) = resultado
        {4.11.3} r(k) = L - resultado
        {4.11.4} Sino          j=1
        {4.11.5}    Mientras b(k, j) > 0
                    j = j + 1
                    fin mientras
                    b(k, j) = resultado
        {4.11.6} r(k) = r(k) - resultado
    {4.12} Fin Si
{5} Fin Mientras

```

Figura 4.4: Algoritmo GRASP Construcción para el problema de cortes

Comentarios

?? La rutina **llenar_RCL** de la línea {4.4} genera la lista candidata y lo guarda en el arreglo **rcl()** con un tamaño de **num_rcl** datos que conforman la lista.

Función llenar_RCL (rcl (), e(), n, a, ß, d, num_rcl)

Entero ee

Para ee = 0 hasta n - 1 **hacer**

Si e(ee) >= ß - a * (ß - d) **entonces**

 rcl(num_rcl) = e(ee)

 num_rcl = num_rcl + 1

Sino

 ee = n - 1

Fin Si

Fin Para

Fin llenar_RCL

Figura 4.5: Función que llena RCL

?? La complejidad de la función **llenar_RCL** es n, es decir, $O(n)$.

?? En las líneas {4.5} y {4.6} escogemos aleatoriamente una candidata y lo guardamos en la variable **resultado**.

?? En la línea {4.7} eliminamos el dato **resultado** de la lista candidata.

Función Eliminar_dato (E (), n, num_rcl)

Para i = num_rcl **hasta** n-1

 E (i)=E (i+1)

Fin Para

 n = n-1

Fin Eliminar_dato

Figura 4.6: Función Eliminar dato

?? Entre las líneas {4.8} y {4.10} realizamos un recorrido de todos los residuos

$r()$ de las barras utilizadas buscando el primero donde encaje.

?? Finalmente entre las líneas {4.11} y {4.12}, actualizamos las barras y residuos de ellas según si utilizamos una barra nueva o es una barra usada.

?? La complejidad de la función Eliminar_dato es $O(n)$.

?? El algoritmo *GRASP Construcción*, como se muestra en la figura 4.4, tiene una complejidad de $O(n^2)$.

4.4 Registrar_Mejor_Solución

Es la parte donde almacenamos y actualizamos los datos de la mejor solución encontrada hasta ese momento, es decir, de la solución que tenga menor número de barras utilizadas. A continuación la ilustración del algoritmo:

```
Registrar_Mejor_Solución (conta, mmin)  
Inicio  
Si  $m < mmin$  or  $conta = 0$  Hacer  
    Para  $i = 0$  hasta  $m$   
        Mientras  $b(i)(j) > 0$  hacer  
             $b\_min(i)(j) = b(i)(j)$   
             $r\_min(i) = r(i)$   
             $j = j + 1$   
        Fin mientras  
    Fin para  
     $mmin = m$   
Fin si  
Fin GRASP_Mejor_Solucion
```

Figura 4.7: Registrar Mejor Solución

Comentarios

?? El parámetro de entrada **conta** indica si hay alguna solución existente con quien comparar.

?? El parámetro de entrada **mmin** almacena el número de barras utilizadas en la mejor solución

?? Registrar_Mejor_Solución (conta, mmin), como se muestra en la figura 4.7, esta función tiene una complejidad de $O(n^2)$.

4.5 Complejidad del algoritmo GRASP

Para realizar el cálculo de esta complejidad se ha dividido el algoritmo en sus respectivas funciones.

?? Complejidad GRASP:

?? Complejidad Ordenar: $O(n^2)$

?? Complejidad {4} {5} = nIter (n^2)

?? Complejidad GRASP_Construcción: $O(n^2)$

?? Complejidad Registrar_Mejor_Solución $O(n^2)$

Por lo tanto la complejidad de GRASP es: **$O(n^2)$**

CAPÍTULO V

DESCRIPCIÓN DEL SISTEMA

CAPÍTULO V

DESCRIPCIÓN DEL SISTEMA

En este capítulo se describe el sistema implementado, al cual se le ha llamado CorteSoft. En este software se han implementado los algoritmos FFD, BFD y el GRASP; se describen los requerimientos mínimos de software y hardware, la estructura del sistema y una breve descripción de cada uno de los módulos que forman parte de CorteSoft.

5.1 Requerimientos mínimos de SW y HD

5.1.1 Configuración de hardware mínimo

Para un funcionamiento adecuado de CorteSoft, se requiere el siguiente hardware:

- ?? Procesador Pentium MMX o mayor
- ?? Velocidad 400 MHZ o superior
- ?? Espacio libre en disco de 5 MB
- ?? Lectora de CD para su instalación.
- ?? Mouse y teclado

5.1.2 Configuración de software mínimo

- ?? Sistema Operativo Windows 98, XP, NT, Millennium, Professional, 2000 Server.
- ?? Office 2000, XP instalado.

5.2 Descripción del Software

El sistema desarrollado para resolver el Problema de Cortes se encuentra básicamente constituido de los algoritmos GRASP Construcción, que se detalló en el capítulo anterior, y de los algoritmos FFD y BFD.

El software ha sido desarrollado básicamente utilizando:

- ?? Visual Basic del paquete Visual Studio 6.0,
- ?? Office XP o 2000 (MS Word para los reportes y MS Excel para guardar los datos)

La metodología que se ha implementado para el desarrollo del algoritmo, es la metaheurística GRASP.

5.2.1 Estructura del Sistema

El sistema esta conformado por estos módulos excepto por el de Experimentos numéricos.

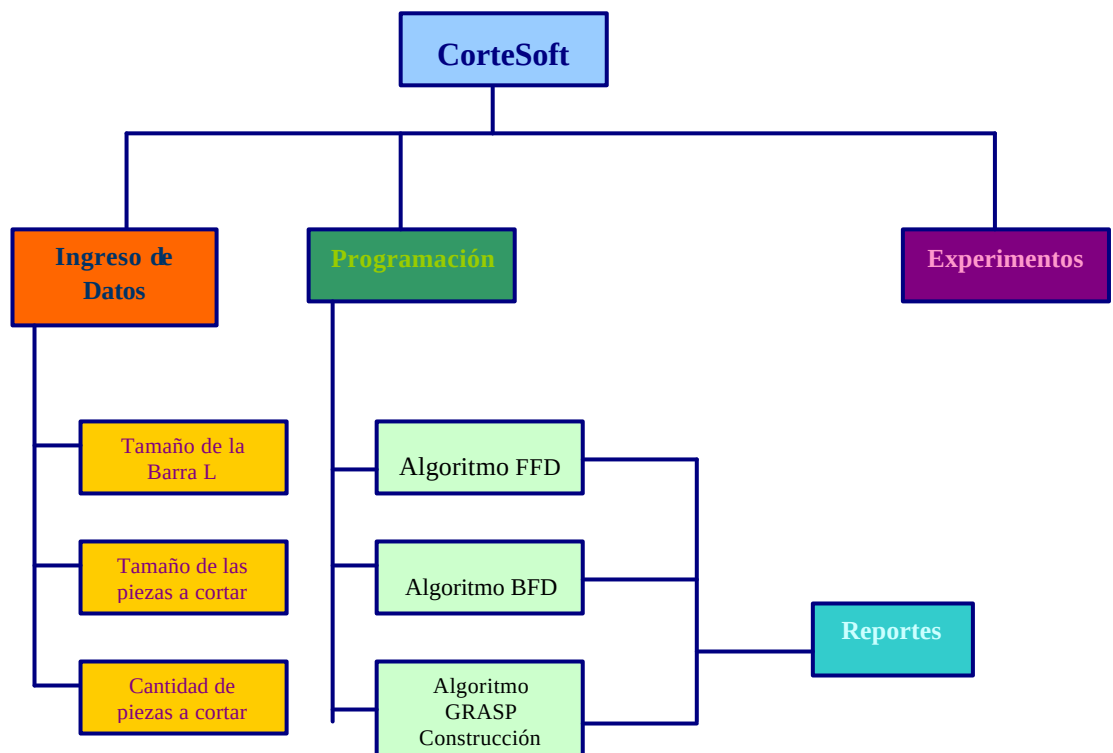


Figura 5.1: Estructura de CorteSoft

Seguidamente se presenta las ventanas del sistema, para que se tenga una mejor visión de CorteSoft.



Figura 5.2: Ventana de presentación de CorteSoft

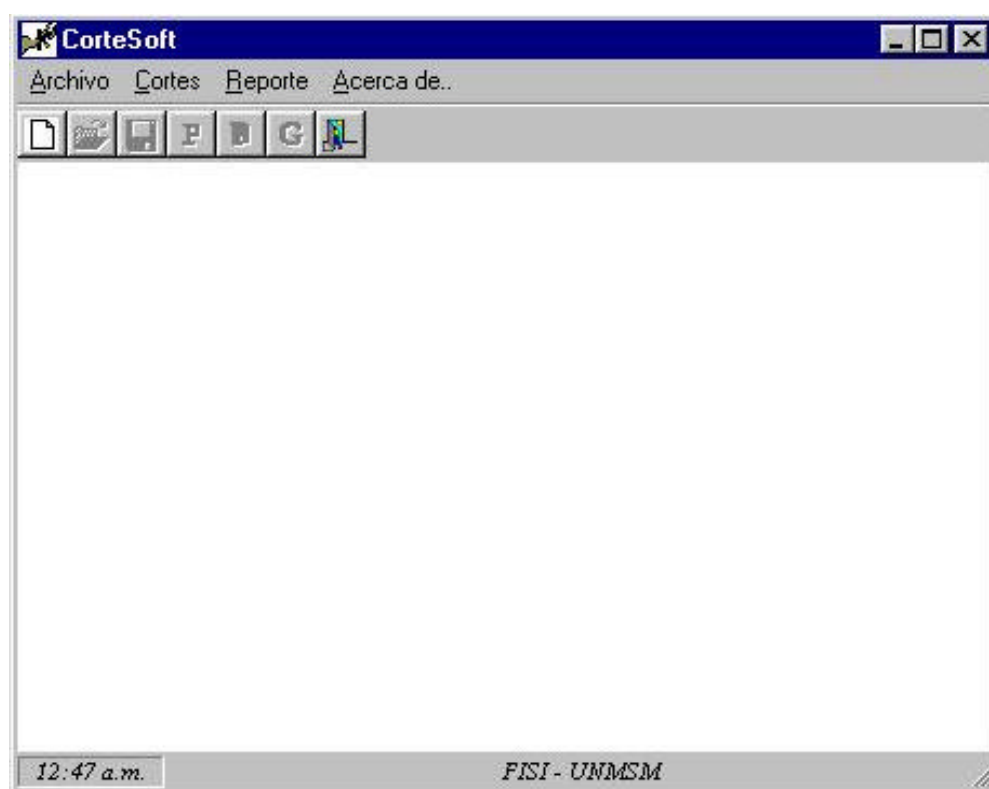


Figura 5.3: Ventana del menú principal de CorteSoft

CorteSoft consta además de un módulo de ayuda, a la cual se podrá acceder cuando el usuario más lo requiera.



Figura 5.4: Ventana de Ayuda de CorteSoft

5.2.2 Módulos del Sistema

5.2.2.1 Módulo de Ingreso

Permite la creación de la estructura de datos que será empleada en la implementación del sistema.

Entre ellas se tiene:

- ?? Ingreso de la Longitud de la barra **L**
- ?? Tamaño de los cortes
- ?? Cantidad de cada corte

Si no se desea ingresar los datos de los cortes, el menú principal le da la opción de ingresarlos de archivo guardados en algún dispositivo.

Ingreso de Parámetros
Ingrese los siguientes datos:

Barra
Longitud: Costo (S/.):

Piezas
Tamaño: Cantidad:

Corte (l.)	Tamaño	Cantidad
------------	--------	----------

Figura 5.5: Ventana de Ingreso de parámetros

5.2.2.2 Módulo de algoritmos

Reúne a los algoritmos que serán ejecutados por el sistema:

- ?? Algoritmo FFD, este algoritmo se ejecuta una vez. En seguida se muestran los resultados.

FFD (First Fit Decreasing)
Resultados:

Cortes							Residuos	
	Corte 1	Corte 2	Corte 3	Corte 4	Corte 5	Corte 6	Residuo	Valor
Barra 1	1*3853	1*2100					Residuo1	47
Barra 2	1*3853	1*2100					Residuo2	47
Barra 3	1*3853	1*2100					Residuo3	47
Barra 4	1*3853	1*2100					Residuo4	47
Barra 5	1*3733	1*2200					Residuo5	67
Barra 6	1*3733	1*2200					Residuo6	67
Barra 7	2*2853						Residuo7	294
Barra 8	2*2853						Residuo8	294
Barra 9	2*2773						Residuo9	454
Barra 10	2*2773						Residuo10	454
Barra 11	2*2773						Residuo11	454
Barra 12	2*2773						Residuo12	454
Barra 13	2*2773						Residuo13	454
Barra 14	2*2753						Residuo14	494
Barra 15	2*2200	1*1353					Residuo15	247
							Residuo16	247

Nº de Barras usadas:
Indice de Desperdicio:
Tiempo:

Figura 5.6: Ventana del Algoritmo FF

?? Algoritmo BFD, este algoritmo también se ejecuta una vez. En seguida se muestran los resultados.

BFD (Best Fit Decreasing)
Resultados:

Cortes						Residuos	
	Corte 1	Corte 2	Corte 3	Corte 4	Corte 5	Corte	Valor
Barra 1	1*3853	1*2100				Residuo1	47
Barra 2	1*3853	1*2100				Residuo2	47
Barra 3	1*3853	1*2100				Residuo3	47
Barra 4	1*3853	1*2100				Residuo4	47
Barra 5	1*3733	1*2200				Residuo5	67
Barra 6	1*3733	1*2200				Residuo6	67
Barra 7	2*2853					Residuo7	294
Barra 8	2*2853					Residuo8	294
Barra 9	2*2773					Residuo9	454
Barra 10	2*2773					Residuo10	454
Barra 11	2*2773					Residuo11	454
Barra 12	2*2773					Residuo12	454
Barra 13	2*2773					Residuo13	454
Barra 14	2*2753					Residuo14	494
Barra 15	2*2200	1*1353				Residuo15	247
						Residuo16	247

Nº de Barras usadas: 216
Indice de Desperdicio: 4.34 %
Tiempo: 0:0:1

Salir

Figura 5.7: Ventana del Algoritmo BFD

?? Algoritmo GRASP, para ejecutar este algoritmo, se necesita ingresar el parámetro de relajación α y el número de iteraciones del algoritmo.

GRASP (Procedimiento de Búsqueda Randomica, Adaptativa y Golosa)
Resultados:

☒ Ingresar parámetros

Parámetros
Alfa: 0.3 Nº de Iteraciones: 10 Ejecutar GRASP

Cortes						Residuos:	
	Corte 1	Corte 2	Corte 3	Corte 4	Corte 5	Corte	Valor
Barra 1	1*3853	1*2053				Residuo1	94
Barra 2	1*3853	1*2053				Residuo2	94
Barra 3	1*3733	1*2200				Residuo3	67
Barra 4	1*3853	1*2053				Residuo4	94
Barra 5	1*3733	1*2200				Residuo5	67
Barra 6	1*3853	1*2100				Residuo6	47
Barra 7	1*2773	1*2753				Residuo7	474
Barra 8	1*2853	1*2773				Residuo8	374
Barra 9	2*2773					Residuo9	454
Barra 10	2*2773					Residuo10	454
Barra 11	1*2853	1*2200	1*653			Residuo11	294
Barra 12	1*2753	1*2200	1*653			Residuo12	394
Barra 13	2*2200	1*1253				Residuo13	347
Barra 14	1*2853	1*2773				Residuo14	374
						Residuo15	374

Nº de Barras usadas: 215 Tiempo: 0:0:13
Indice de Desperdicio: 3.9 %

Salir

Figura 5.8: Ventana del Algoritmo GRASP

5.2.2.3 Reporte

Esta parte del software permite al usuario además de visualizar los resultados de los cortes, la facilidad de imprimir o guardar los reportes. Estos reportes se dan en Office 2000 (Word). Al ser creado este reporte inmediatamente se genera un código con el que será guardado el archivo.

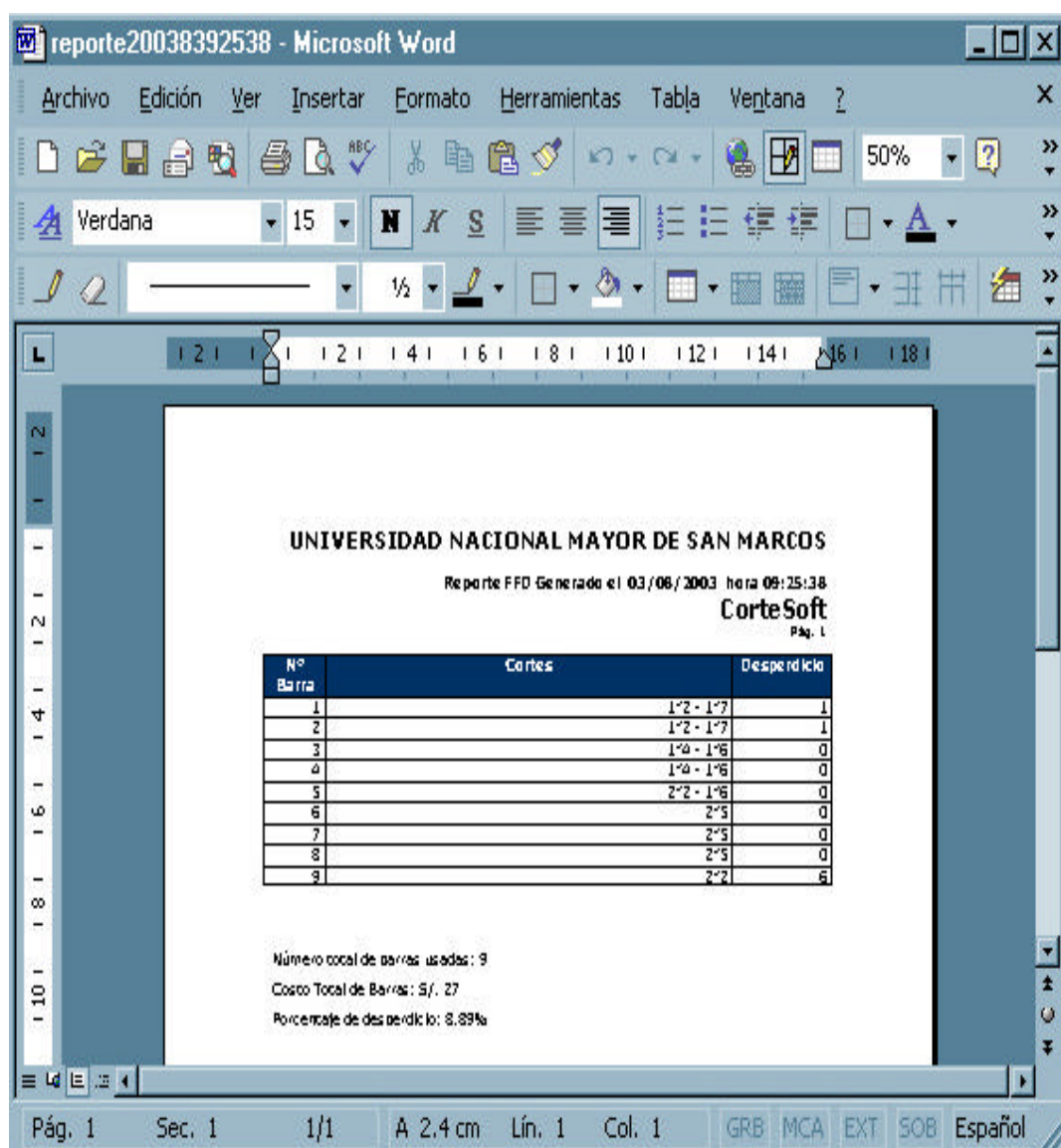


Figura 5.9: Ventana de Reporte

CAPÍTULO VI

EXPERIMENTOS NUMÉRICOS

CAPÍTULO VI

EXPERIMENTOS NUMÉRICOS

6.1 Requerimientos de Hardware y Software

A continuación se da una descripción del software y hardware utilizados en el desarrollo del aplicativo de esta tesis, CorteSoft.

6.1.1 Hardware

Microprocesador:	Intel Pentium IV
Velocidad:	2.0 GHZ
Memoria RAM:	256 MB
Sistema de disco:	Maxtor 40Gb
Sistema de video:	NVIDIA GeForce MX / MX 400
Sistema de archivo:	32 b

6.1.2 Software

Sistema Operativo:	Windows 98
Compilador:	Visual Basic 6.0
Reporte:	Office 2000

6.2 Instancias de Pruebas

Se presenta a continuación los grupos de instancias usados para la demostración y análisis de resultados. Para mayor referencia, revisar el Anexo. B.

Tabla 6.1: Lista de instancias de Prueba

Grupos	Autor	N.º de Instancias
Grupo1	Nacif Rocha	10
Grupo2	Nacif Rocha	6
Grupo3	Nacif Rocha	2
Grupo4	Nacif Rocha	35
Grupo5	Stadtler	22
Grupo6	Pauta	2
Grupo7	Degraeve	2
Grupo8	Goulimis	2
Grupo9	Hinterding & Khan	4
Grupo10	Aloise & Maculan	15
Total:		100

6.3 Resultados Numéricos

Tabla 6.2

Resultados del Grupo # 1

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Óptimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
20-347	215	96.1	215	96.1	215	96.1	214	96.6	214	96.6	214	96.6	214	96.6	214	96.6	214	96.6	216	95.7	216	95.7	96.6	207
20-348	56	98.2	56	98.2	56	98.2	56	98.2	56	98.2	56	98.2	56	98.2	56	98.2	56	98.2	57	96.4	57	96.4	98.2	55
20-349	186	96.7	186	96.7	186	96.7	186	96.7	186	96.7	186	96.7	186	96.7	186	96.7	186	96.7	186	96.7	186	96.7	96.7	180
CM-063	147	96.5	147	96.5	147	96.5	146	97.2	146	97.2	146	97.2	146	97.2	146	97.2	146	97.2	147	96.5	147	96.5	97.2	142
CM-096	46	82.1	46	82.1	46	82.1	46	82.1	46	82.1	46	82.1	46	82.1	46	82.1	46	82.1	46	82.1	46	82.1	82.1	39
ME-001	81	96.2	81	96.2	81	96.2	81	96.2	81	96.2	81	96.2	81	96.2	81	96.2	81	96.2	81	96.2	81	96.2	96.2	78
ME-011	32	89.7	32	89.7	32	89.7	32	89.7	32	89.7	32	89.7	32	89.7	32	89.7	32	89.7	32	89.7	32	89.7	89.7	29
ME-028	191	97.3	191	97.3	191	97.3	191	97.3	190	97.8	190	97.8	191	97.3	191	97.3	190	97.8	192	96.7	192	96.7	97.8	186
ME-029	214	96.6	214	96.6	214	96.6	213	97.1	213	97.1	213	97.1	211	98.1	211	98.1	211	98.1	214	96.6	214	96.6	98.1	207
ME-030	298	97.9	298	97.9	298	97.9	298	97.9	298	97.9	298	97.9	299	97.6	299	97.6	299	97.6	301	96.9	301	96.9	97.9	292

Tabla 6.3

Resultados del Grupo # 2

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Óptimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
ME-004	344	95.8	344	95.8	344	95.8	353	93.0	353	93.0	353	93.0	352	93.3	352	93.3	353	93.0	346	95.2	346	95.2	95.8	330
ME-003	357	97.4	357	97.4	357	97.4	357	97.4	357	97.4	357	97.4	357	97.4	357	97.4	357	97.4	365	95.1	365	95.1	97.4	348
ME-011	397	90.3	395	90.9	395	90.9	395	90.9	395	90.9	395	90.9	396	90.6	396	90.6	395	90.9	371	97.5	371	97.5	90.9	362
ME-010	708	96.3	708	96.3	708	96.3	708	96.3	708	96.3	708	96.3	709	96.2	709	96.2	709	96.2	715	95.3	715	95.3	96.3	683
ME-001	680	98.7	680	98.7	680	98.7	680	98.7	680	98.7	680	98.7	680	98.7	680	98.7	680	98.7	680	98.7	680	98.7	98.7	671
ME-019	397	90.3	395	90.9	395	90.9	395	90.9	395	90.9	395	90.9	396	90.6	396	90.6	395	90.9	371	97.5	371	97.5	90.9	362

Tabla 6.4

Resultados del Grupo # 3

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Optimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
ME-010	971	96.9	971	96.9	971	96.9	971	96.9	970	97.0	970	97.0	972	96.8	972	96.8	972	96.8	980	96.0	980	96.0	97	942
ME-050	128	96.8	128	96.8	128	96.8	128	96.8	128	96.8	128	96.8	129	96.0	129	96.0	129	96.0	129	96.0	129	96.0	96.8	124

Tabla 6.5
Resultados del Grupo # 4

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Óptimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
P-001	77	95.9	77	95.9	77	95.9	76	97.3	76	97.3	76	97.3	76	97.3	76	97.3	76	97.3	77	95.9	77	95.9	97.3	74
P-002	94	95.6	94	95.6	94	95.6	94	95.6	94	95.6	94	95.6	93	96.7	93	96.7	93	96.7	94	95.6	94	95.6	96.7	90
P-003	43	97.6	43	97.6	43	97.6	43	97.6	43	97.6	43	97.6	43	97.6	43	97.6	43	97.6	43	97.6	43	97.6	97.6	42
P-004	22	100.0	22	100.0	22	100.0	22	100.0	22	100.0	22	100.0	22	100.0	22	100.0	22	100.0	22	100.0	22	100.0	100	22
P-005	343	99.1	343	99.1	343	99.1	343	99.1	343	99.1	343	99.1	344	98.8	344	98.8	344	98.8	343	99.1	343	99.1	99.1	340
P-006	24	95.7	24	95.7	24	95.7	24	95.7	24	95.7	24	95.7	23	100.0	23	100.0	23	100.0	24	95.7	24	95.7	100	23
P-007	23	95.5	23	95.5	23	95.5	23	95.5	23	95.5	23	95.5	23	95.5	23	95.5	23	95.5	23	95.5	23	95.5	95.5	22
P-008	55	92.2	55	92.2	55	92.2	55	92.2	55	92.2	55	92.2	55	92.2	55	92.2	55	92.2	56	90.2	56	90.2	92.2	51
P-009	68	100.0	68	100.0	68	100.0	68	100.0	68	100.0	68	100.0	68	100.0	68	100.0	68	100.0	68	100.0	68	100.0	100	68
P-010	21	100.0	21	100.0	21	100.0	21	100.0	21	100.0	21	100.0	21	100.0	21	100.0	21	100.0	21	100.0	21	100.0	100	21
P-011	505	99.6	505	99.6	505	99.6	505	99.6	505	99.6	505	99.6	506	99.4	506	99.4	506	99.4	506	99.4	506	99.4	99.6	503
P-012	20	100.0	20	100.0	20	100.0	20	100.0	20	100.0	20	100.0	20	100.0	20	100.0	20	100.0	20	100.0	20	100.0	100	20
P-013	35	93.9	35	93.9	35	93.9	35	93.9	35	93.9	35	93.9	35	93.9	35	93.9	35	93.9	34	97.0	34	97.0	93.9	33
P-014	21	95.0	21	95.0	21	95.0	21	95.0	21	95.0	21	95.0	21	95.0	21	95.0	21	95.0	21	95.0	21	95.0	95	20
P-015	86	100.0	86	100.0	86	100.0	87	98.8	87	98.8	87	98.8	87	98.8	87	98.8	87	98.8	86	100.0	86	100.0	100	86
P-016	62	96.7	62	96.7	62	96.7	62	96.7	62	96.7	62	96.7	62	96.7	62	96.7	62	96.7	62	96.7	62	96.7	96.7	60
P-017	108	95.1	108	95.1	108	95.1	107	96.1	107	96.1	107	96.1	107	96.1	107	96.1	107	96.1	107	96.1	107	96.1	96.3	103
P-018	598	94.5	598	94.5	598	94.5	598	94.5	598	94.5	598	94.5	598	94.5	598	94.5	598	94.5	598	94.5	598	94.5	94.5	567
P-019	40	94.7	40	94.7	40	94.7	40	94.7	40	94.7	40	94.7	40	94.7	40	94.7	40	94.7	40	94.7	40	94.7	94.7	38

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Óptimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
P-020	71	98.6	71	98.6	71	98.6	71	98.6	71	92.6	71	93.6	71	92.6	71	92.6	71	92.6	70	100.0	70	100.0	92.6	70
P-021	50	95.8	50	95.8	50	95.8	50	95.8	50	95.8	50	95.8	49	97.9	49	97.9	49	97.9	49	97.9	49	97.9	97.9	48
P-022	204	99.5	204	99.5	204	99.5	204	99.5	204	99.5	204	99.5	204	99.5	204	99.5	204	99.5	204	99.5	204	99.5	99.5	203
P-023	108	98.1	108	98.1	108	98.1	108	98.1	108	98.1	108	98.1	109	97.2	109	97.2	109	97.2	108	98.1	108	98.1	98.1	106
P-024	77	97.3	77	97.3	77	97.3	77	97.3	77	97.3	77	97.3	77	97.3	77	97.3	77	97.3	76	98.7	76	98.7	97.3	75
P-025	190	93.3	190	93.3	190	93.3	189	93.8	189	93.8	189	93.8	186	95.5	186	95.5	186	95.5	190	93.3	190	93.3	95.5	178
P-026	125	94.1	125	94.1	125	94.1	125	94.1	125	94.1	125	94.1	125	94.1	125	94.1	125	94.1	125	94.1	125	94.1	94.1	118
P-027	88	92.7	88	92.7	88	92.7	88	92.7	88	92.7	88	92.7	88	92.7	88	92.7	88	92.7	88	92.7	88	92.7	92.7	82
P-028	199	91.8	199	91.8	199	91.8	199	91.8	199	91.8	199	91.8	199	91.8	200	91.3	200	91.3	200	91.3	200	91.3	91.8	184
P-029	272	99.3	272	99.3	272	99.3	272	99.3	272	99.3	272	99.3	272	99.3	272	99.3	272	99.3	272	99.3	272	99.3	99.3	270
P-030	402	97.4	402	97.4	402	97.4	398	98.5	398	98.5	398	98.5	402	97.4	402	97.4	402	97.4	403	97.2	403	97.2	98.5	392
P-031	115	96.4	115	96.4	115	96.4	115	96.4	115	96.4	115	96.4	115	96.4	115	96.4	115	96.4	117	94.6	117	94.6	96.4	111
P-032	53	94.0	53	94.0	53	94.0	53	94.0	53	94.0	53	94.0	53	94.0	53	94.0	53	94.0	53	94.0	53	94.0	94	50
P-033	129	96.8	129	96.8	129	96.8	129	96.8	129	96.8	129	96.8	129	96.8	129	96.8	129	96.8	129	96.8	129	96.8	96.8	125
P-034	75	100.0	75	100.0	75	100.0	75	100.0	75	100.0	75	100.0	75	100.0	75	100.0	75	100.0	75	100.0	75	100.0	100	75
P-035	70	92.3	70	92.3	70	92.3	69	93.8	69	93.8	69	93.8	69	93.8	69	93.8	69	93.8	70	92.3	70	92.3	93.8	65

Tabla 6.6

Resultado del Grupo: STADLER

Código	a = 0.3						a = 0.4						a = 0.5						FFD		RFD		Mejor eficiencia GRASP	Óptimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
113816610	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	78	291
113859642	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	100	1
115084640	6	100	6	100	6	100	6	100	6	100	6	100	6	100	6	100	6	100	6	100	6	100	100	6
118384640	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	100	1
119082640	4	100	4	100	4	100	4	100	4	100	4	100	100	4	100	4	100	4	100	4	100	4	100	4
119246640	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	100	2
130808640	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	100	2
130810640	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	67	3
130813610	637	78.2	637	78.2	637	78.2	637	78.2	637	78.2	637	78.2	637	78.2	637	78.2	637	78.2	637	78.2	637	78.2	78	523
130813640	146	98.6	146	98.6	146	98.6	146	98.6	146	98.6	146	98.6	146	98.6	146	98.6	146	98.6	146	98.6	146	98.6	99	144
130814640	13	100	13	100	13	100	13	100	13	100	13	100	13	100	13	100	13	100	13	100	13	100	100	13
130816810	9	100	9	100	9	100	9	100	9	100	9	100	9	100	9	100	9	100	9	100	9	100	100	9
130817640	20	100	20	100	20	100	20	100	20	100	20	100	20	100	20	100	20	100	21	95	21	95	100	20
130819640	3	100	3	100	3	100	3	100	3	100	3	100	3	100	3	100	3	100	3	100	3	100	100	3
130820640	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	100	2
130821640	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	100	1
132068610	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	356	77.7	78	291
132070640	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	100	2
712644640	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	67	3
717832610	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	100	2
717833640	10	88.9	10	88.9	10	88.9	10	88.9	10	88.9	10	88.9	10	88.9	10	88.9	10	88.9	10	88.9	10	88.9	89	9
717837810	467	78.4	467	78.4	467	78.4	467	78.4	467	78.4	467	78.4	467	78.4	467	78.4	467	78.4	467	78.4	467	78.4	78	384

Tabla 6.7

Resultados del Grupo: PAUTA

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Optimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
ac01	130	95.2	130	95.2	130	95.2	130	95.2	130	95.2	130	95.2	131	94.4	131	94.4	131	94.4	130	95.2	130	95.2	95.2	124
ac02	15	92.9	15	92.9	15	92.9	15	92.9	15	92.9	15	92.9	15	92.9	15	92.9	15	92.9	15	92.9	15	92.9	92.9	14

Tabla 6.8

Resultados del Grupo: DEGRAEVE

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Optimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	Nº barras	Eficiencia (%)	Nº barras	Eficiencia (%)		
Ferro1	39	97.4	39	97.4	39	97.4	39	97.4	39	97.4	39	97.4	39	97.4	39	97.4	39	97.4	39	97.4	39	97.4	97	38
Ferro2	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100	100	10

Tabla 6.9

Resultados del Grupo: GOULIMIS

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Óptimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
goulim1	56	90.2	56	90.2	56	90.2	56	90.2	56	90.2	55	92.2	55	92.2	55	92.2	55	92.2	56	90.2	56	90.2	92	51
goulim2	61	96.6	61	96.6	61	96.6	62	94.9	62	94.9	62	94.9	62	94.9	62	94.9	62	94.9	62	94.9	62	94.9	97	59

Tabla 6.10

Resultados del Grupo: HINTERDING & KHAN

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia GRASP	Óptimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
p-001a	9	87.5	9	87.5	9	87.5	9	87.5	9	87.5	9	87.5	9	87.5	9	87.5	9	87.5	9	87.5	9	87.5	88	8
p-002a	23	100	23	100	23	100	23	100	23	100	23	100	23	100	23	100	23	100	23	100	23	100	100	23
p-003a	15	100	15	100	15	100	15	100	15	100	15	100	16	93.3	15	100	15	100	16	93.3	16	93.3	100	15
p-004a	20	94.7	20	94.7	20	94.7	20	94.7	20	94.7	20	94.7	19	100	19	100	19	100	20	94.7	20	94.7	100	19

Tabla 6.11

Resultados del GRUPO: ALOISE & MACULAN

Código	a = 0.3						a = 0.4						a=0.5						FFD		BFD		Mejor eficiencia. GRASP	Optimo
	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	1000 iter.	Eficiencia (%)	2000 iter.	Eficiencia (%)	4000 iter.	Eficiencia (%)	N.º barras	Eficiencia (%)	N.º barras	Eficiencia (%)		
P-001	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-002	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-003	3	50	3	50	3	50	3	50	3	50	3	50	2	100	2	100	2	100	3	50	3	50	100	2
P-004	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-005	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-006	3	50	3	50	3	50	3	50	3	50	3	50	2	100	2	100	2	100	3	50	2	100	100	2
P-007	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-008	3	50	3	50	3	50	3	50	3	50	3	50	3	50	3	50	3	50	3	50	2	100	50	2
P-009	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-010	3	50	3	50	3	50	2	100	2	100	2	100	2	100	2	100	2	100	3	50	2	100	100	2
P-011	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-012	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-013	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	2	100	3	50	3	50	100	2
P-014	3	50	3	50	3	50	2	100	2	100	2	100	2	100	2	100	2	100	3	50	2	100	100	2
P-015	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	4	66.7	3	100	3	100	3	100	4	66.7	4	66.7	100	3

6.4 Análisis de Resultados

6.4.1 Eficiencia

Esta tabla muestra las eficiencias promedio por grupo, por parámetro a y con diferente número de iteraciones.

Tabla 6.12

Tabla resumen de las eficiencias promedio

Grupo	Nº de instancias	a = 0.3			a = 0.4			a = 0.5		
		1000	2000	4000	1000	2000	4000	1000	2000	4000
Grupo 1	10	94.7	94.7	94.7	94.9	95	95	95	95	95
Grupo2	6	94.8	95	95	94.5	94.5	94.5	94.5	94.5	94.5
Grupo3	2	96.8	96.8	96.8	96.8	96.9	96.9	96.4	96.4	96.4
Grupo4	35	96.5	96.5	96.5	96.7	96.7	96.7	96.8	96.8	96.8
Grupo Stadler	22	92.4	92.4	92.4	92.4	92.4	92.4	95.33	92.4	92.4
Grupo Pauta	2	94.01	94.01	94.01	94.01	94.01	94.01	93.61	93.61	93.61
Grupo Degraeve	2	98.68	98.68	98.68	98.68	98.68	98.68	98.68	98.68	98.68
Grupo Goulimis	2	93.4	93.4	93.4	92.56	95.56	93.54	93.54	93.54	93.54
Grupo Hinterding & Khan	4	95.5	95.5	95.5	95.5	95.5	95.5	95.2	96.88	96.88
Grupo Aloise & Maculan	15	81.11	81.11	81.11	87.78	87.78	87.78	96.67	96.67	96.67
Eficiencia Promedio	100	92.91	92.92	92.92	93.96	94.03	93.99	95.33	95.4	95.4

Resumen

Los resultados de la tabla muestran que en el peor de los casos, la eficiencia promedio es de **92.91** que se cumple para $a = 0.3$ con 1000 iteraciones de GRASP y que en el mejor de los casos, la eficiencia promedio es de **95.4** para un $a = 0.5$ con 2000 y 4000 iteraciones del algoritmo GRASP.

Si se extraen los datos del Grupo Stadler, los resultados serian los siguientes:

Tabla 6.13: Tabla de resultados excluyendo al grupo Stadler

	1000	2000	4000
$a = 0.3$	93	93.1	93.1
$a = 0.4$	94.4	94.5	94.4
$a = 0.5$	96.2	96.2	96.2

De esto se tiene que en el peor de los casos la eficiencia es **93%** con $a = 0.3$ y 1000 iteraciones, y en el mejor de los casos la eficiencia es de **96.2%** con $a = 0.5$ para 1000, 2000 y 4000 iteraciones.

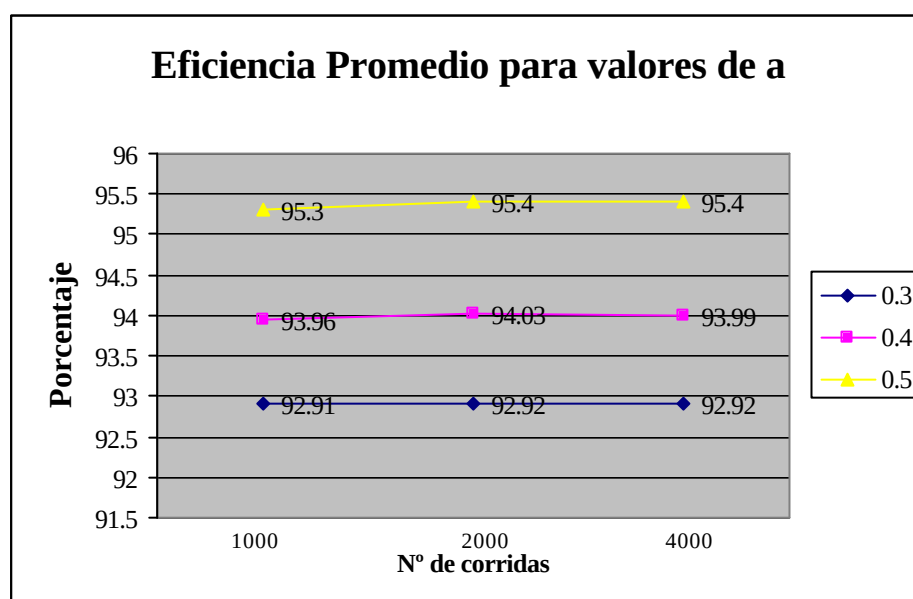


Figura 6.1: Eficiencia promedio para valores de a

La siguiente tabla presenta una comparación de los promedios de las mejores eficiencias de GRASP con las eficiencias de FFD y BFD. Este cuadro se cálculo eligiendo la mejor de las eficiencias para cada grupo, promediándolo entre el número de grupos.

Tabla 6.14: Comparación de Eficiencias de los Algoritmos

Grupos	Eficiencia GRASP (%)	Eficiencia FFD (%)	Eficiencia BFD(%)
Grupo 1	95.05	94.4	94.4
Grupo2	95	96.54	96.54
Grupo3	96.9	95.97	95.97
Grupo4	96.96	96.65	96.65
Grupo Stadler	92.4	92.17	92.17
Grupo Pauta	94.1	94	94
Grupo Degraeve	98.7	98.7	98.7
Grupo Goulimis	94.4	92.6	92.6
Grupo Hinterding & Khan	96.9	93.88	93.88
Grupo Aloise & Maculan	96.7	51.11	64.44
Eficiencia Promedio	95.711	90.602	91.935

Resumen

De los resultados obtenidos se observa que el algoritmo GRASP es más eficiente que las heurísticas desarrolladas (FFD, BFD), con un valor de 95.711% sobre un 90.602% de FFD y un 91.935% de BFD.

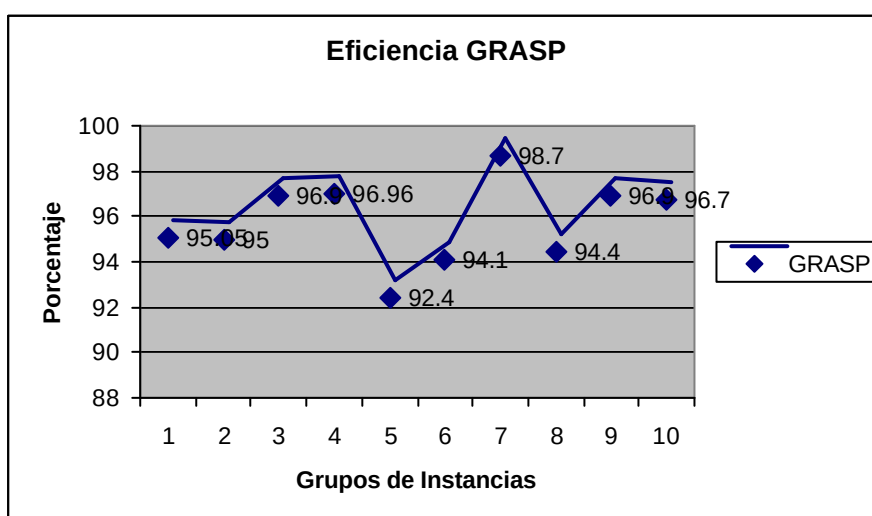


Figura 6.2: Gráfica del promedio de la mejor eficiencia de GRASP

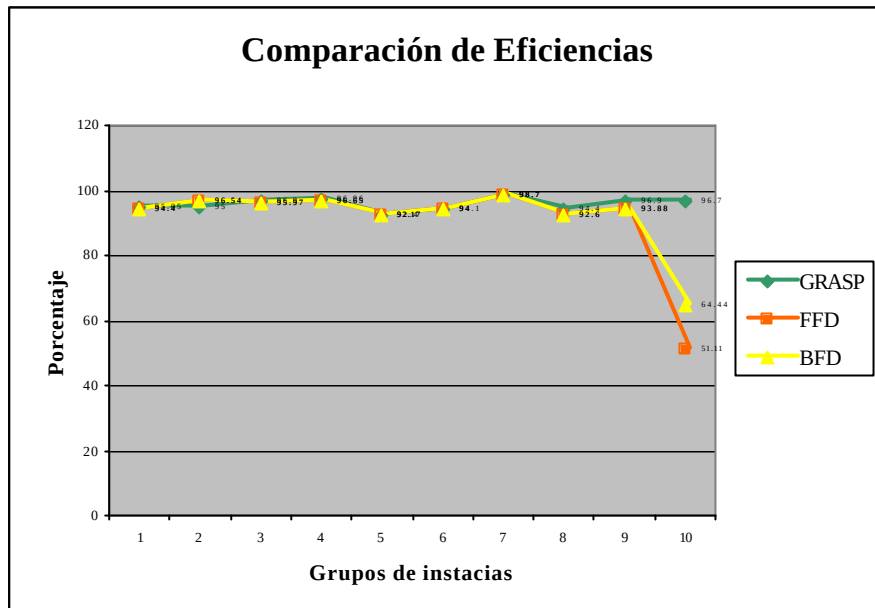


Figura 6.3: Gráfica del promedio de la mejor eficiencia de GRASP

6.4.2 Tiempo

La tabla presenta los resultados de los tiempos promedios por grupo y el tiempo promedio total por parámetro a y número de iteraciones.

Tabla 6.15

Tabla resumen de los tiempos promedio

Grupo	Nº de instancias	a = 0.3			a = 0.4			a = 0.5		
		1000	2000	4000	1000	2000	4000	1000	2000	4000
Grupo 1	10	43s36ms	1m31s	3m1s54ms	46s54ms	1m35s	3m10s42ms	53s18ms	1m49s12ms	3m44s
Grupo2	6	11m53s50ms	1m54s20ms	47m56s	12m36s20ms	25m39s10ms	51m31s30ms	12m58s	26m3s40ms	51m1s40ms
Grupo3	2	6m15s	12m41s	26m15s	8m18s	8m58s	18m4s30ms	8m48s	17m43s	35m32s
Grupo4	35	1m10s53ms	2m22s	4m46s22ms	1m13s34ms	2m27s33ms	4m55s26ms	1m23s31ms	2m47s9ms	5m29s27ms
Grupo Stadler	22	36s30ms	1m13s14ms	2m28s27ms	38s44ms	1m18s3ms	2m38s30ms	37s52ms	1m16s49ms	2m44s5ms
Grupo Pauta	2	2s30ms	4s30ms	8s	2s30ms	4s30ms	10s	3s	6s	11s
Grupo Degraeve	2	10s30ms	21s	42s30ms	10s	19s30ms	39s30ms	10s	19s	40s
Grupo Goulimis	2	1s	2s30ms	4s30ms	1s30ms	2s30ms	5s30ms	2s	5s	6s
Grupo Hinterding & Khan	4	1s	1s	1s45s	1s	1s	2s15ms	1s	1s	2s
Grupo Aloise & Maculan	15	1s	1s	1s24ms	1s	1s	1s20ms	1s	1s	1s16ms
Tiempo Promedio	100	1m28s	2m57s	5m56ms31ms	1m35s17ms	3m2s8ms	5m56s55ms	1m40s37ms	3m22s11ms	6m41s34ms

Estos resultados fueron obtenidos al realizar una suma de productos entre el número de instancias por sus respectivos tiempos promedios, los cuales luego se promediaron sobre el total de instancias. De los resultados, se observa que en el mejor de los casos el tiempo promedio es 1m28s para $a=0.3$ con 1000 iteraciones del algoritmo GRASP y en el peor de los casos el tiempo es de 6m41s34ms para $a=0.5$ con 5000 iteraciones de GRASP. Se observa también que para $a=0.3$, los tiempos promedios son menores con respecto a los otros dos valores de a . Esto se visualiza mejor en la siguiente gráfica.

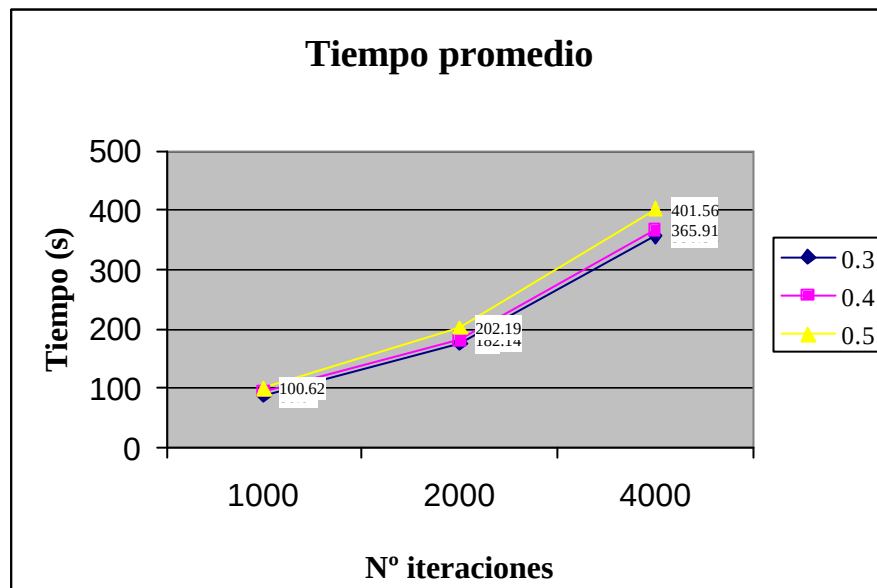


Figura 6.4: Gráfica del promedio de los tiempos de GRASP

CAPÍTULO VII

CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO VII

CONCLUSIONES Y RECOMENDACIONES

El Problema de Cortes lineales es un tema de interés, particularmente, en las industrias, que buscan reducir sus costos de producción al minimizar el desperdicio. Gracias a trabajos como este se puede contribuir a solucionar este tipo de problemas.

El software desarrollado permite no sólo probar la metaheurística sino también las Heurísticas FFD y BFD soportando grandes cantidades de volumen a comparación de otros similares en el mercado. Además permite guardar la información de la demanda.

Después de realizar las pruebas experimentales y comparativas, se ha concluido que, en general GRASP mejora, en la mayoría de los casos, las soluciones de las heurísticas desarrolladas, es decir, sus resultados se acercan más al resultado óptimo minimizando el desperdicio en los cortes.

La ventaja del GRASP frente al FFD y BFD radica en que éste mejora los resultados de las soluciones, pero su desventaja es su tiempo de procesamiento, el cual es mucho mayor que el de las heurísticas FFD y BFD.

El parámetro de relajación α que utiliza GRASP para encontrar una mejor solución varía según el problema. Las pruebas se realizaron con valores de α igual a 0.3, 0.4 y 0.5.

Los experimentos numéricos sugieren en general usar $\alpha = 0.5$, y con un número de iteraciones de por lo menos 2000.

Si bien se ha mejorado la solución llegando a un 95.4, para mejorar esta solución, se recomienda implementar el módulo mejoría del algoritmo GRASP.

La velocidad de procesamiento puede ser mejorada implementando una versión eficiente del algoritmo para el uso de demandas no unitaria de las piezas a cortar.

ANEXOS

APÉNDICE A - Parámetros usados para los experimentos numéricos

A1 Parámetros

Valores de $\alpha = 0.3, 0.4, 0.5$

Número de Iteraciones = 1000, 2000, 4000

APENDICE B – Instancias de Prueba

Grupo #1: Nacif Rocha [Na 97]

Piezas de aluminio extraídos de una obra de porte pequeño para medio, con peso líquido de cerca de 3 toneladas.

Tamaño de Barras: 6000mm

Código	Tamaño	Cantidad
ME-028	1962	176
	1462	16
	980	56
	818	428
	805	24
	595	32
	529	428
	381	192
20-347	3853	4
	3733	2
	2853	4
	2773	10
	2753	2
	2200	10
	2100	4
	2053	76
	1853	5
	1353	12
	1253	436
	1053	218
	653	218
	553	76
ME-029	1162	48
	1062	32
	980	56
	931	152
	930	16
	895	16
	865	8
	818	428

Código	Tamaño	Cantidad
	805	40
	931	16
	625	40
	595	32
	529	428
	462	436
	362	152
20-348	2153	36
	1853	15
	1200	10
	1053	76
	953	5
	828	50
	753	28
	653	28
20-349	553	76
	3003	5
	2953	4
	2903	3
	2853	2
	2803	1
	2753	13
	2553	20
	2450	10
	2400	8
	2350	6
	2300	4
	2250	2
	2200	32
	1853	13

Código	Tamaño	Cantidad
	1653	8
	1253	516
	1053	10
	953	116
CM-063	2528	20
	1828	10
	1628	8
	1228	302
	1227	222
	1027	10
	927	116
	628	8
CM-096	2978	5
	2928	4
	2878	3
	2828	2
	2778	1
	2728	13
	2477	10
	2427	8
	2377	6
	2327	4
	2277	2
	2227	32
	1828	3
ME-001	826	20
	800	6
	726	16
	653	20
	640	16
	628	12
	615	8
	603	4
	590	52
	565	80
	526	176
	524	428
ME-011	2910	5
	2860	4
	2810	3
	2760	2
	2710	1

Código	Tamaño	Cantidad
	2660	13
	2640	10
	1760	8
	1560	4
	1160	44
CM-030	1105	80
	1080	64
	1055	48
	1030	32
	1005	16
	980	232
	862	20
	831	20
	818	428
	805	24
	762	272
	731	16
	658	80
	645	64
	633	48
	620	32
	608	16
	595	208
	570	80
	531	176
	529	428

Grupo #2: Nacif Rocha [Na 97]

Piezas de aluminio extraídos de una obra de porte pequeño para medio, con peso líquido de cerca de 18 toneladas.

Tamaño de Barras: 6000mm

Código	Tamaño	Cantidad
ME-001	801	80
	776	920
	751	720
	726	120
	706	240
	701	2048
	676	1104
	626	40
	576	40
	526	40
	476	40
	426	40
	416	40
	411	40
	406	40
	401	40
	376	40
	351	40
	346	40
	341	40
	336	40
	331	40
	326	40
ME-003	1653	20
	1603	230
	1553	180
	1503	30
	1463	60
	1453	512
	1403	276
	1303	10
	1203	10
	1103	10
	1003	10
	903	10
	883	10
	873	10
	863	10
	853	10
	803	10
	753	10

Código	Tamaño	Cantidad
ME-004	743	10
	733	10
	723	10
	713	10
	703	10
	1552	20
	1512	230
	1502	180
	1402	20
	1392	60
	1362	512
	1352	10
	1302	256
	1252	20
	1152	20
	1102	10
	1052	20
	932	10
	922	10
	912	10
	902	10
	852	10
	802	10
	792	10
	782	10
	772	10
	762	10
	752	10
ME-010	1600	40
	1560	460
	1550	360
	1450	40
	1440	120
	1410	1024
	1400	20
	1350	512
	1300	40
	1200	40
	1150	20
	1100	40

Código	Tamaño	Cantidad
	980	20
	970	20
	960	20
	950	20
	900	20
	850	20
	840	20
	830	20
	820	20
	810	20
	800	20
ME-011	1710	20
	1660	230
	1610	180
	1560	30
	1520	60
	1510	512
	1460	276
	1360	10
	1260	10
	1160	10
	1060	10
	960	10
	940	10
	930	10
	920	10
	910	10
	860	10
	810	10
	800	10
	790	10
	780	10
	770	10
	760	10
ME-019	1710	20
	1660	230
	1610	180
	1560	30
	1520	60
	1510	512
	1460	276
	1360	10
	1260	10
	1160	10

Código	Tamaño	Cantidad
	1060	10
	960	10
	940	10
	930	10
	920	10
	910	10
	860	10
	810	10
	800	10
	790	10
	780	10
	770	10
	760	10

Grupo # 3: Nacif Rocha [Na 97]

Piezas de aluminio extraídos de una obra de porte pequeño para medio, con peso líquido de cerca de 26 toneladas.

Tamaño de Barras: 6000mm

Código	Tamaño	Cantidad
ME-050	998	20
	993	20
	988	20
	983	20
	978	20
	976	20
	973	20
	968	20
	963	20
	962	75
	960	70
	958	65
	957	20
	956	60
	954	55
	952	50
	882	45
	872	40
	862	35
	852	30
	767	25
	762	20
	757	15
	752	10
ME-010	1600	40
	1560	460
	1550	360
	1450	40
	1440	120
	1410	1024
	1400	20
	1650	512
	1300	40
	1200	40
	1150	20
	1100	40
	1046	40
	1041	40
	1036	40
	1031	40
	1026	40
	1024	40

Código	Tamaño	Cantidad
	1021	40
	1016	40
	1011	40
	1010	150
	1008	140
	1006	130
	1005	40
	1004	120
	1002	110
	1000	100
	980	20
	970	20
	960	20
	950	20
	930	90
	920	80
	910	70
	900	80
	850	20
	840	20
	830	20
	820	20
	815	50
	810	60
	805	30
	800	40

Grupo # 4: : Nacif Rocha [Na 97]

Piezas de aluminio extraídos de un conjunto de 10 diferentes obras, de varios portes.

Tamaño de Barras: 6000mm a 6100mm

Código	Tamaño	Cantidad
P-001	1950	8
	1850	4
	1450	230
	1250	32
	1150	32
	550	17
P-002	1500	64
	1439	32
	1433	12
	1431	64
	950	32
	948	54
	939	96
	931	96
	689	2
	683	4
	681	2
	533	30
	433	2
	333	2
P-003	2974	54
	1874	16
	1474	2
	1174	45
	974	1
	774	1
P-004	2000	8
	1900	4
	1500	4
	1300	32
	1200	32
	600	32
P-005	6050	32
	3050	94
	2520	34
	2120	2
	2050	172
	2020	2

Código	Tamaño	Cantidad
	1950	36
	1550	144
	1350	32
	1250	284
	1050	120
	950	64
	850	2
	650	189
	550	172
	450	98
P-006	2000	2
	1200	64
	900	32
	600	26
	400	32
P-007	1950	4
	1850	2
	1450	50
	1250	16
	1150	16
	550	16
P-008	2456	34
	2056	2
	1950	4
	1850	2
	1450	115
	1250	16
	1150	16
	550	1
P-009	950	4
	941	64
	844	32
	800	64
	747	32
	600	32
	550	96
	544	26
	541	134

Código	Tamaño	Cantidad
P-010	453	14
	441	136
	344	32
	914	4
	808	32
	600	32
	514	96
P-011	508	26
	308	32
	1848	16
	1748	8
	1500	4
	1454	128
	1439	32
	1433	24
	1432	128
	1431	160
	1348	460
	1274	102
	1048	64
	1000	55
	948	216
	939	96
	931	300
	914	4
	904	64
	882	64
	881	64
	874	6
	808	32
	800	64
	750	64
	689	2
	687	32
	683	8
	681	4
	600	32
	533	60
	514	96
	511	30
	508	26
	500	47
	481	134

Código	Tamaño	Cantidad
	472	45
	448	4
	435	8
	433	4
	419	192
	393	14
	382	64
	381	136
	333	4
	308	32
	300	64
	235	192
	232	52
	132	64
P-012	860	32
	600	64
	560	14
	436	14
	400	64
	337	32
P-013	1200	18
	1120	45
	1050	73
	1000	36
	872	2
	560	10
	525	6
	426	4
P-014	2100	18
	2000	10
	1200	18
	1120	15
	1050	14
	1000	4
P-015	2000	4
	1200	192
	1000	46
	900	64
	600	186
	500	68
	400	64

Código	Tamaño	Cantidad
P-016	2974	64
	1874	32
	1500	4
	1000	38
	500	38
	483	96
P-017	1439	32
	1433	12
	1432	64
	1431	64
	1000	17
	948	54
	939	96
	931	96
	882	32
	689	2
	683	4
	681	2
	542	60
	535	60
	533	30
	500	9
	433	2
	333	2
P-018	1439	704
	1427	176
	1000	119
	939	2112
	689	44
P-019	3500	16
	1974	40
	1000	64
	761	30
	738	3
P-020	674	3
	654	80
	641	256
	544	6
	541	160

Código	Tamaño	Cantidad
	450	128
	224	220
P-021	901	160
	662	60
	618	80
	414	128
P-022	1048	268
	933	24
	932	148
	919	12
	901	80
	875	12
	693	120
	662	30
	639	3
	618	80
	581	256
	484	6
	481	160
	445	160
	414	128
	326	120
	297	6
	287	160
	185	256
	164	220
P-023	3500	34
	2974	30
	1974	80
	1000	128
	700	128
	600	34
P-024	283	110
	2975	4
	1675	64
	1475	181
	1175	5
	973	56

Código	Tamaño	Cantidad
P-025	2250	128
	2185	4
	2150	120
	1800	112
	1200	256
P-026	2210	64
	2145	2
	2110	60
	1165	112
	1161	12
	1158	248
P-027	2210	64
	2145	2
	2110	60
	1165	56
	1161	8
	1158	124
P-028	1069	8
	876	8
	784	672
	765	672
	687	40
	527	16
	253	40
P-029	1065	32
	1062	372
	1044	8
	806	8
	716	32
	706	484
	668	64
	628	128
	556	4
	554	120
	544	112
	530	12
	518	253
	516	248
	501	8
	489	120
	478	250

Código	Tamaño	Cantidad
	476	12
	312	128
	243	12
	237	250
P-030	1148	256
	1132	2
	1083	8
	1069	224
	1048	240
	856	504
	809	496
	709	480
	559	32
	554	258
	532	32
P-031	457	224
	452	224
	809	124
	806	8
	716	32
	709	120
	706	484
	559	8
	556	4
	457	224
P-032	688	128
	561	8
	549	120
	538	262
P-033	2300	8
	1200	114
	1154	6
	750	128
	704	128
	600	288
	554	404
P-034	1675	64
	1475	181
	1175	5
	973	56

Código	Tamaño	Cantidad
P-035	1975	1
	1775	4
	1675	65
	1475	180
	1175	4

Grupo STADTLER

Piezas de aluminio – problemas test propuestos por Hartmut Stadler [St 90]

Tamaño de Barras: 6005mm

Código	Tamaño	Cantidad	Código	Tamaño	Cantidad
112542610	776	34	130808640	965	4
	728	30		894	4
	667	1194	130810640	2086	4
	657	400		1903	4
	656	984	130813610	6000	156
	615	1200		4602	32
	564	34		4352	412
	490	4		2452	48
	488	60		2102	26
	429	122		130	674
	381	560	130813640	2394	16
	276	600		2194	16
113816610	6000	82		1996	2
	4750	21		1994	16
	4500	223		1985	2
	3285	3		1956	4
	2600	32		1947	2
	2250	25		1938	6
113859642	2130	1		1738	6
				1538	6
115084640	2135	6		1447	2
	1385	2		1338	6
	1380	2		1138	6
	1260	10		938	8
	885	2		928	2
	750	2		912	2
118384640	2064	2		900	2
	910	1		885	2
119082640				813	2
	2500	2		738	6
	2000	4		554	360
	1125	1		553	340
	1000	2		538	6
	875	2		492	10
119246640	760	1		421	10
				373	704
	1135	5		312	16
	1010	1	130814640	2435	2
	510	1			

Código	Tamaño	Cantidad
	2029	4
	2020	2
	2013	4
	1991	4
	1274	2
	963	2
	962	2
	956	4
	928	4
	910	2
	572	24
	407	24
130816810	1352	20
	1196	20
130817640	2029	8
	2013	6
	1279	4
	1206	4
	1154	4
	1057	10
	1013	12
	963	2
	962	2
	956	12
	938	4
	932	4
	779	2
	698	2
	654	2
	644	4
	450	12
	448	2
	325	24
	274	35
130819640	1854	2
	1814	4
	1305	2
	786	2
	671	2
130820640	1991	4
	928	4

Código	Tamaño	Cantidad
130821640	1063	2
	894	2
132068610	6000	82
	4750	21
	4500	223
	3285	3
	2600	32
	2250	25
132070640	1135	5
	1010	1
	510	1
712644640	2130	4
	1903	4
717832610	1925	2
	1539	2
	1000	2
717833640	4011	2
	2135	6
	1135	22
	1010	5
717837810	2400	400
	1750	800

Grupo PAUTA

Estos datos han sido extraídos del trabajo de R.R. Mota [Mo 89]

Tamaño de Barras: 6005mm

Código	Tamaño	Cantidad
AC-01	6770	11
	6050	32
	5970	32
	5500	32
	5470	32
	4920	12
	4700	16
	4400	16
	4000	50
	3550	32
	3350	32
	2600	17
AC-02	7300	5
	6900	2
	6400	2
	5800	4
	5400	2
	4900	2
	3400	4
	2250	4
	2050	3
	1870	4
	1700	5
	1500	2
	1370	3

Grupo DEGRAEVE

Problemas test extraídos del trabajo de Degraeve y Bemelmans [DB 97]

Tamaño de Barras: 6000mm para Ferro1 y 3000mm para Ferro2

Código	Tamaño	Cantidad
Ferro1	1910	15
	1850	15
	1790	30
	1110	15
	1101	45
	550	15
	500	75
	340	15
Ferro2	560	15
	491	15
	390	30

Grupo GOULIMIS

Cortes de madera/papel – extraídos del trabajo de Goulimis [Gu 90]

Tamaño de Barras: 4300mm

Código	Tamaño	Cantidad
goulim01	2350	2
	2250	4
	2200	4
	2100	15
	2050	6
	2000	11
	1950	6
	1900	15
	1850	13
	1700	5
	1650	2
	1350	9
	1300	3
	1250	6
	1200	10
	1150	4
	1100	8
	1050	3
goulim02	2200	24
	2150	4

Código	Tamaño	Cantidad
	2100	12
	2050	18
	2000	2
	1950	7
	1900	4
	1850	2
	1750	11
	1650	4
	1550	11
	1450	2
	1300	7
	1250	3
	1200	3
	1150	1
	1100	6
	1050	6
	1000	6
	950	7
	900	7
	850	6

Grupo HINTERDING & KHAN

Extraídos del trabajo de Hinterding [Hi 94]

Tamaño de Barras: 140mm para (P-001a) ,150 para (P-002a) y 250 para (P-003a, P-004a)

Código	Tamaño	Cantidad
P-001 ^a	100	3
	90	1
	80	2
	70	4
	60	2
	50	1
	40	2
	30	5
P-002a	100	8
	90	5
	80	5
	70	8
	60	7
	50	5
	40	8
	30	4
P-003a	100	6
	90	4
	80	6
	70	15
	60	5
	50	6
	40	12
	30	6
P-004a	120	1
	110	8
	100	6
	90	4
	80	7
	70	15
	60	12
	50	7

Grupo ALOISE & MACULAN

Extraídos del trabajo de Aloise y Maculan [AM 91]

Tamaño de Barras: 100mm

Código	Tamaño	Cantidad	Código	Tamaño	Cantidad
P-001	57	1	P-007	61	1
	48	1		57	1
	26	2		15	2
	23	1		13	1
	20	1		12	2
P-002				7	1
	57	1		4	1
	45	1		3	1
	29	1		1	1
	26	2	P-008	84	1
	17	1		41	1
P-003	73	1		27	1
	47	1		20	1
	31	1		12	1
	22	1		9	1
	15	1		7	1
	12	1	P-009	60	1
P-004	67	1		53	1
	61	1		15	2
	22	1		14	1
	20	1		9	2
	10	1		6	3
	9	1		5	1
	4	2		2	1
	3	1	P-010	74	1
P-005	71	1		49	1
	64	1		46	1
	24	1		9	1
	12	2		6	1
	9	1		5	2
	8	1		2	3
P-006	70	1	P-011	66	1
	41	1		65	1
	33	1		26	1
	26	1		9	2
	15	1		7	1
	7	1		6	1
	6	1		5	2
	2	1		2	1

Código	Tamaño	Cantidad
P-012	39	1
	36	1
	34	1
	30	1
	11	1
	10	2
	9	1
	7	1
	6	1
	5	1
	3	1
P-013	53	1
	48	1
	17	1
	9	3
	8	3
	7	2
	6	2
	4	1
	1	1
P-014	46	1
	45	1
	28	1
	27	1
	14	1
	13	1
	10	1
	8	1
	6	1
	3	1
P-015	69	1
	63	1
	43	1
	37	1
	29	1
	28	1
	20	1
	11	1

APÉNDICE C - Manual de Usuario

CORTESOFT

MANUAL DE USUARIO

CorteSoft 2003
Edición Empresarial
Versión 1.0

CorteSoft - Manual de Usuario

El software descrito en el presente manual está sujeto a un contrato de licencia y sólo puede ser utilizado según los términos del mismo.

Advertencia:

Este programa está protegido por las leyes del Derecho de Autor y otros tratados internacionales. La reproducción o distribución no autorizada de este programa o parte de ella, pueden dar lugar a responsabilidades punidas de acuerdo a ley.

Tabla de Contenidos

SECCION 1 PARA EMPEZAR

Capítulo 1 Instalación de CorteSoft

Instalación CorteSoft

Requisitos de Software

Requisitos de Hardware

Procedimientos de Instalación

Desinstalación CorteSoft

Capítulo 2 Introducción a CorteSoft

SECCION 2 FUNCIONAMIENTO DE CORTESOFT

Capítulo 3 Funcionamiento de CorteSoft

Inicio de navegación en CorteSoft

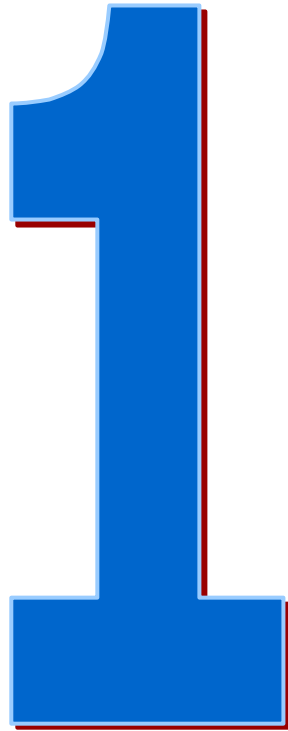
Ingreso de Parámetros

Ejecución de FFD

Ejecución de BFD

Ejecución de GRASP

Generación de Reporte



Para Empezar

CAPÍTULO 1

Instalación de CorteSoft

Bienvenidos a CorteSoft, una herramienta eficaz y económica, que soluciona su problema al realizar cortes lineales (en una dimensión). Visualiza los cortes en cada barra, calcula los desperdicios producidos al realizar los cortes y realiza cálculos que le permiten a su empresa mejorar su producción ahorrando recursos al disminuir sus pérdidas.

Instalación CorteSoft

Antes de realizar la instalación de CorteSoft, dedique un momento a revisar los requisitos de software y hardware enumerados en esta sección.

Requisitos de Software

Para poder utilizar CorteSoft, su PC debe cumplir los siguientes requisitos mínimos de Software:

- ?? Sistema Operativo Windows® 98 Windows 2000, Windows NT, Windows 2000 Server, Windows 2003 Server, Windows Millennium, Windows XP. (Este producto no se ejecuta bajo DOS).
- ?? Office 2000, XP. (Excel, Word)

Requisitos de Hardware

Para poder utilizar CorteSoft, su PC debe cumplir los siguientes requisitos mínimos de Hardware:

- ?? Procesador Pentium MMX o mayor
- ?? Velocidad 400 MHZ o superior
- ?? 64 MB de memoria RAM, (memoria adicional recomendada).
- ?? Vídeo VGA de 256 colores o superior.
- ?? Espacio libre en disco de 5 MB
- ?? Lectora de CD para su instalación.
- ?? Tarjeta de sonido opcional.
- ?? Mouse y teclado

Procedimientos de Instalación

Para instalar:

1. Inicie Windows (si aún no está ejecutándose).
2. Inserte el CD de CorteSoft en la unidad de CD-ROM.
3. Haga clic en el archivo **Setup** , lo cual iniciará la instalación copiando archivos necesarios para ello.
4. Terminada la copia de archivos se mostrará una pantalla de bienvenida, haga click en aceptar para iniciar la instalación propiamente dicha.
5. Siga las instrucciones que aparezcan en la pantalla.

Consejo: En la mayoría de los casos, las opciones que aparecen seleccionadas en el programa de instalación son las más adecuadas. A menos que se necesite instalar algo poco corriente, se deben aceptar las opciones predeterminadas.

Desinstalación CorteSoft

Puede eliminar CorteSoft del sistema totalmente.

Para suprimir CorteSoft del sistema:

1. Vaya al Panel de Control y haga click sobre el icono de Agregar o quitar programas.
2. Elija CorteSoft y haga click en el botón eliminar.
3. Siga las instrucciones que se muestran en pantalla para desinstalar CorteSoft de su sistema.

CAPÍTULO 2

Introducción a CorteSoft

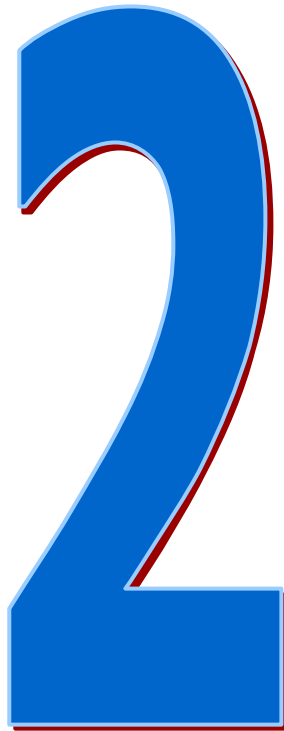
CorteSoft es un software que le permite simular el cortado de barras de manera eficaz.

Realiza cortes mediante tres metodologías:

- ?? FFD (First Fit Decreasing)
- ?? BFD (Best Fit Decreasing)
- ?? GRASP (Greedy Randomized Adaptive Search Procedure)

Estos tres metodologías solucionan el problema de cortes de manera distinta dando como resultado el número de barras que se han utilizado para cumplir con la demanda y el porcentaje de desperdicio al realizar los cortes.

CorteSoft soporta gran volumen de datos a diferencia de otros productos existentes en el mercado, utiliza metodologías llamadas heurísticas (FFD, BFD) y metaheurísticas (GRASP) que brindan soluciones eficientes permitiéndole un ahorro de recursos y una pérdida inútil de dinero.



Funcionamiento de CorteSoft

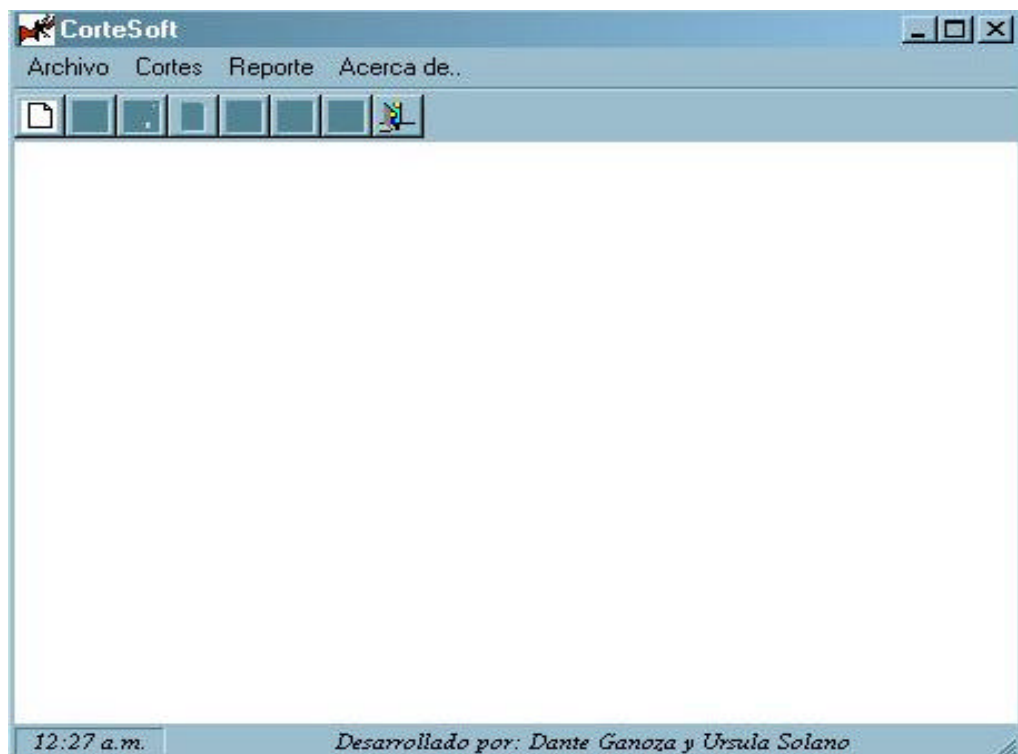
CAPÍTULO 3

Funcionamiento de CorteSoft

En este capítulo, se explican los procedimientos básicos para la utilización de CorteSoft. Éstas son las operaciones que va a realizar con todos los programas de CorteSoft.

Inicio de navegación en CorteSoft

Para iniciar CorteSoft, haga clic en el botón Inicio y seleccione Programas > CorteSoft



La pantalla principal de CorteSoft es el punto de partida para cualquier actividad. Si hace clic en el menú Archivo > Nuevo o en el icono , verá la pantalla de ingreso de parámetros y se habilitarán las opciones del menú. Las opciones del menú principal situados en la parte superior pueden realizar funciones en varias zonas del programa.

Ingreso de Parámetros

Son ingresados en la pantalla de Ingreso de Parámetros. Los principales parámetros son: Tamaño de la barra estándar, Tamaño de las piezas que se desean suplir y las cantidades correspondientes. Presionar el botón **Aceptar** de la sección **Piezas** para ingresarlos a la grilla. Si los datos son mal ingresados se mostraran mensajes de avisos.

Terminados de ser ingresados los datos, se presiona el botón **Ejecutar** para tener todos los datos cargados y por lo tanto proceder a ejecutar las metodologías. Se activaran las opciones **G**, **F** y **B** y el

menú correspondiente. Puede eliminar el último dato ingresado presionando el botón **Eliminar**

Ingreso de Parámetros

Ingrese los siguientes datos:

Barra

Longitud: 6000 Aceptar

Piezas

Tamaño: Cantidad: Aceptar

Corte(li)	Tamaño	Cantidad
L1	2978	5
L2	2928	4
L3	2878	3
L4	2828	2
L5	2778	1
L6	2728	13
L7	2477	10
L8	2427	8
L9	2377	6
L10	2327	4
L11	2277	2
L12	2227	2

Ejecutar

Eliminar

Ejecución de FFD

Esta metodología se ejecuta al hacer clic en  o en el menú Cortes>FFD. La pantalla muestra la manera en que se realizaron los cortes por cada barra, con sus respectivos desperdicios. Muestra también datos resumen como el total de barras usadas en el proceso de corte, el tiempo de proceso y el porcentaje total de desperdicio. Para salir haga clic en el botón **Salir**.

FFD (First Fit Decreasing)							
Resultados :							
Cortes						Residuos	
	Corte 1	Corte 2	Corte 3	Corte 4	Corte 5	Corte	Valor
Barra 1	2978	2978					Residuo1 44
Barra 2	2978	2978					Residuo2 44
Barra 3	2978	2928					Residuo3 94
Barra 4	2928	2928					Residuo4 144
Barra 5	2928	2878					Residuo5 194
Barra 6	2878	2878					Residuo6 244
Barra 7	2828	2828					Residuo7 344
Barra 8	2778	2728					Residuo8 494
Barra 9	2728	2728					Residuo9 544
Barra 10	2728	2728					Residuo10 544
Barra 11	2728	2728					Residuo11 544
Barra 12	2728	2728					Residuo12 544
Barra 13	2728	2728					Residuo13 544
Barra 14	2728	2728					Residuo14 544
Barra 15	2477	2477					Residuo15 1046
							Residuo16 1046

N° de Barras usadas: 46
 Indice de Desperdicio: 17.32 %
 Tiempo : 0:0:2

Salir

Ejecución de BFD

Esta metodología se ejecuta al hacer clic en **B** o en el menú Cortes>BFD. La pantalla muestra la manera en que se realizaron los cortes por cada barra, con sus respectivos desperdicios. Muestra también datos resumen como el total de barras usadas en el proceso de corte , el tiempo o de proceso y el porcentaje total de desperdicio. Para salir haga clic en el botón **Salir**.

BFD (Best Fit Decreasing)							
Resultados:							
Cortes						Residuos	
	Corte 1	Corte 2	Corte 3	Corte 4	Corte 5	Corte	Valor
Barra 1	2978	2978					Residuo1 44
Barra 2	2978	2978					Residuo2 44
Barra 3	2978	2928					Residuo3 94
Barra 4	2928	2928					Residuo4 144
Barra 5	2928	2878					Residuo5 194
Barra 6	2878	2878					Residuo6 244
Barra 7	2828	2828					Residuo7 344
Barra 8	2778	2728					Residuo8 494
Barra 9	2728	2728					Residuo9 544
Barra 10	2728	2728					Residuo10 544
Barra 11	2728	2728					Residuo11 544
Barra 12	2728	2728					Residuo12 544
Barra 13	2728	2728					Residuo13 544
Barra 14	2728	2728					Residuo14 544
Barra 15	2477	2477					Residuo15 1046
							Residuo16 1046

N° de Barras usadas: 46
 Indice de Desperdicio: 17.32 %
 Tiempo : 0:0:0

Salir

Ejecución de GRASP

Esta metodología se ejecuta al hacer clic en  o en el menú Cortes>GRASP. Para ejecutarlo primero se deben ingresar (si así lo desea) dos parámetros. El programa le permite elegir si ingresa o no estos parámetros. El valor del parámetro alpha debe estar entre 0 y 1. Para mostrar los resultados, hacer clic en el botón **Aceptar**. Muestra los datos resumen al igual que en los otros dos casos. Para salir haga clic en el botón **Salir**.



GRASP (Procedimiento de Búsqueda Randomica, Adaptativa y Golosa)
Resultados:

Parámetros

Alfa : 0.3 N° de Iteraciones : 2

Cortes	Corte 1	Corte 2	Corte 3	Corte 4	Corte 5	Residuos:	Valor
Barra 1	1*2778	1*2728				Residuo1	494
Barra 2	1*2928	1*2728				Residuo2	344
Barra 3	1*2928	1*2728				Residuo3	344
Barra 4	1*2978	1*2728				Residuo4	294
Barra 5	2*2728					Residuo5	544
Barra 6	1*2978	1*2878				Residuo6	144
Barra 7	2*2728					Residuo7	544
Barra 8	1*2878	1*2728				Residuo8	394
Barra 9	1*2978	1*2728				Residuo9	294
Barra 10	1*2928	1*2728				Residuo10	344
Barra 11	1*2828	1*2728				Residuo11	444
Barra 12	1*2978	1*2878				Residuo12	144
Barra 13	1*2978	1*2928				Residuo13	94
Barra 14	1*2828	1*2728				Residuo14	444
						Residuo15	1046
						Residuo16	1246

N° de Barras usadas: 46 Tiempo : 0:0:0

Indice de Desperdicio: 17.32 %

Generación de Reporte

Para esto haga clic en el botón  del menú principal o en Reporte>Ver. Enseguida se cargará el reporte en MS Word.

reporte200372314343 - Microsoft Word

Archivo Edición Ver Insertar Formato Herramientas Tabla Ventana ?

Verdana 10

50%

Reporte Generado el 23/7/2003 hora 01:43:45

CorteSoft
Pag. 1

Nº Barra	Cortes	Desperdicio
1	1*2728 - 1*2778	494
2	1*2728 - 1*2828	344
3	1*2728 - 1*2928	344
4	1*2728 - 1*2978	294
5	2*2728	544
6	1*2878 - 1*2978	144
7	2*2728	544
8	1*2728 - 1*2878	394
9	1*2728 - 1*2978	294
10	1*2728 - 1*2928	344
11	1*2728 - 1*2828	444
12	1*2878 - 1*2978	144
13	1*2928 - 1*2978	94
14	1*2728 - 1*2828	444
15	2*2677	1046
16	2*2377	1246
17	1*2327 - 1*2427	1246
18	1*2327 - 1*2477	1196
19	1*2327 - 1*2527	1246
20	1*2427 - 1*2477	1096
21	1*2327 - 1*2477	1196
22	1*2427 - 1*2477	1096
23	1*2427 - 1*2477	1096
24	1*2377 - 1*2477	1146
25	1*2427 - 1*2477	1096
26	2*2427	1146
27	2*2377	1246
28	1*2377 - 1*2477	1146
29	2*2227	1546
30	2*2227	1546
31	2*2227	1546
32	2*2227	1546
33	2*2227	1546

BIBLIOGRAFIA

- [1] www.astrokettle.com/pr1dsc.html
- [AM 91] Aloise D, Maculan N, “*Uma classe de algoritmos aproximativos decrescente para o problema ‘bin packing’*”, Revista Brasileira de Computacao, V.6 N.3 P3-12, Jan. / Mar. 1991.
- [Be 00] BECKE VON DER, C.,. “*Glosario de Carlos von der Becke*” www.telepolis.com, Mayo 19. 2000
- [CM 92] CAMPELLO R, MACULAN N, “*Algoritmos Heurísticos*”. Brasil. 1992
- [DB 92] DEGRAEVE Z., BEMELMANS R., “*Cutting stock problem at Metal Aarschot*”.Departament of Applied Economic Sciences – Katholieke Universiteit Leuven (obtenido de internet).
- [De 99] DELGADO S. Cristina, “*Diseño de Metaheurísticos Híbridos para problemas de rutas con flota heterogénea (2 parte): y concentración heurística*”, Burgos-España.1999
- [Dy 90] DYCKHOFF H., “*A typology of cutting and packing problems*” European Journal of Operation Research 44 (1990) 197-208. 1990.
- [FR 95] FEO T.A, RESENDE M.G.C., “*Greedy Randomized Adaptive Search Procedures*”, Journal of Global Optimization, 6:109-133, 1995
- [GG 61] GILMORE P.C., GOMORY R.E., “*A linear programming approach to*

the cutting stock problem". Operations Research 9 (6) , 849-859, 1961

- [Go 89] GOLDBERG, D., "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison-Wesley, Reading, MA, 1989.
- [GK 13] GRADISAR M, KLIJAJIĆ M, "*A sequential heuristic procedure for one dimension cutting*", European Journal of Operation Research 114 (1999) 557-568.
- [Gl 89] GLOVER F, "*Tabú Search*". Part I. ORSA Journal on Computing, 1(3):190-206, 1989.
- [GR 97] GRADIŠAR M, RESINOVIC G, "*Evaluation of algorithms for one dimensional cutting*". Faculty of Organisational Sciences, University of Maribor, 4000 Kranj, Prešernova ulica 11, Slovenia , Faculty of Economics, University of Ljubljana, 1000 Ljubljana, Kardeljeva ploščad 17, Slovenia, 1997
- [Gu 90] GOULIMIS C. "*Optional solutions for the cutting stock problem*". European Journal of Operations Research 44 (1990)197-208.
- [Ha 98] HATORI S. "*One Dimensional Cutting Stock for Plastic Film*". Tonen System Plaza Inc. Shibuya - ku, Tokyo 150, Japan .1998.
- [Hi 94] HINTERDING R. "*Genetic Algorithms for cutting: an exploration of mapping schemes*". Department of Computer and Mathematical Sciences -Victoria University of Technology- Technical Report 40 COMP 12 (August, 1994).
- [Hu 00] HUTCHINSON, B. "*Complejidad Computacional*", 8 Feb 2000, www.mor.itesm.mx.
- [Il 97] Ilog, Inc., "*Optimization Technology White Paper: A comparative study of optimization techniques*", www.ilog.com, 1997

- [MD 02] MAURICIO, D, DELGADILLO R., *“Algoritmos GRASP para el Problema de Cortes”*, Relatorio Técnico N.º 01/2002, FISI-UNMSM, Universidad Nacional Mayor de San Marcos, 2002.
- [Mi 94] MARTÍ R. *“Greedy Randomized Adaptive Search Procedures”*, 1994 www.uv.es/~rmarti/grasp.html
- [Mo 89] MOTA R. *“Optimizacao das Perdas em Cortes de Barras para Estruturas de Concreto Armado: Um Sistema Computacional”*, Resumos XXII SOBRAPO (1989).
- [Na 97] NACIF R, *“Optimización de corte de barras”*. Tesis aprobada el 05 de febrero de 1997, Belo Horizonte - Brasil.
- [Pe 96] PÉREZ, A., *“Una Introducción a la Computación Evolutiva”*, Marzo 1996.
- [St 87] STADLER, H, *“A one-dimensional cutting stock problem in the aluminum industry and its solution”*. European Journal of Operation Research 44 (1990) 209-223.
- [Tu 00] TUPIA M., *“Una GRASP para resolver el problema de la Programación de Tareas”*. Pontificia Universidad Católica del Perú.
- [Va 98] VANCE, P. *“Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem”*. Department of Industrial Engineering, Auburn University Auburn, Alabama 36849-5346 July 30, 1998.
- [Wa 99] WAGNER B., *“A genetic algorithm solution for one-dimensional bundled stock cutting”* European Journal Of Operation Research 117(1999) 368-381.

