

**Tesis para la carrera de Lic. en Ciencias de la
Computación (Plan 1982) - Universidad de Buenos Aires
(UBA) -
Facultad de Ciencias Exactas y Naturales (FCEyN)**

Deppto. COMPUTACION
INFOTECA
F.C.E.N. - U.B.A.

Autores

Horacio Enrique Haberkon - LU 0990/83
Claudio Daniel Ivaldi - LU 2217/84
Mariano Fabián Foriase - LU 1327/83

Docente a cargo de la tesis

Lic. Isabel Méndez Díaz

Tema

Optimización de la carga de camiones con cajas de
distintos tamaños

Deppto. COMPUTACION
INFOTECA
F.C.E.N. - U.B.A.

1996

INDICE

- I. Resumen/Abstract
- II. Introducción
 - II.1. Definición del Problema
 - II.2. Clasificación del Problema
 - II.3. Historia
- III. Glosario de Términos
- IV. Definición Formal del Problema
- V. Diseño del Algoritmo
 - V.1. Representación Formal del Problema
 - V.2. Espacio de Búsqueda
 - V.3. Heurísticas para Restringir el Espacio de Búsqueda
 - V.4. Estrategia de Búsqueda
- VI. Implementación del Algoritmo FHI
 - VI.1. Algoritmo FHI
 - VI.2. Diagrama de Transición de Estados
 - VI.3. Arbol de Decisión del Módulo "Verificar Objetivo"
 - VI.4. Estructuras de la Base de Datos Global BDG
- VII. Especificaciones Generales
 - VII.1. Hardware
 - VII.2. Software
 - VII.3. Criterios para Seleccionar las Herramientas
 - VII.4. Dificultades que se Encontraron Durante el Desarrollo
- VIII. Resultados Obtenidos de la Ejecución de los Casos de Prueba
 - VIII.1. Datos con los que se Ejecutó el Algoritmo FHI
 - VIII.2. Análisis Comparativo de las Soluciones según las Distintas Heurísticas entre Pedidos
 - VIII.3. Análisis Comparativo de la Performance de las Heurísticas entre Pedidos para Obtener Soluciones con el Máximo Aprovechamiento
 - VIII.4. Análisis comparativo del Índice de Mezcla de las Cajas según las Heurísticas para la Mejor Solución
 - VIII.5. Datos que permiten Monitorear el Algoritmo para la Mejor Solución de un Pedido
 - VIII.6. Comparación entre los Algoritmos FHI y el de Morabito y Arenales con el ejemplo de George y Robinson (1980) para el caso de la Mejor Solución
- IX. Tabla de Restricciones de Tipo de las Cajas
- X. Conclusiones
 - X.1. Ventajas del método implementado
 - X.2. Problemas que nuestro algoritmo no puede hoy resolver
- XI. Bibliografía

AGRADECIMIENTOS

En primer término queremos agradecer a la profesora Lic. Isabel Méndez Díaz por aceptar ser nuestra madrina en este trabajo de tesis, dando muestras siempre de su disposición, conocimiento y vocación de investigadora y docente.

También, agradecemos a la Lic. Irene Loiseau por aconsejarnos y ayudarnos a solucionar los inconvenientes durante la ausencia de la Lic. Méndez Díaz.

Por último, queremos agradecer a nuestros padres Baby y Tito Ivaldi, Sara y José Haberkon y Julia Figueroa, que nos apoyaron durante toda nuestra etapa universitaria, y a nuestras novias Cristina Noguera y María Elizabeth Oliva y esposa Mariana Griselli, las cuales nos brindaron todo su apoyo, entendimiento, paciencia y tiempo para que podamos realizar esta tesis. No nos queremos olvidar de nuestro amigo Ernesto Oliva que nos acompañó siempre durante nuestras reuniones.

I. RESUMEN / ABSTRACT

RESUMEN

Optimización de la carga de camiones con cajas de distintos tamaños

El problema que se trata es el de programar la **carga de un camión con cajas de distintos tamaños**, maximizando el aprovechamiento del espacio. La carga debe satisfacer las restricciones globales de distribución del peso en el camión, restricciones locales por el tipo y peso que soportan las cajas y restricciones de estabilidad de la carga.

Este problema está relacionado con los de **bin packing** y **cutting stock**, y tiene la característica que es un NP-Hard [7]. Por lo tanto no hay, hasta la actualidad, un algoritmo que permita solucionarlo en tiempo polinomial, a menos que se demuestre que $P=NP$.

Se desarrolló un **método de búsqueda** que se implementó en el algoritmo **FHI** que tiene la particularidad de obtener soluciones con un **p%** de aprovechamiento de la carga.

El **método de búsqueda** propuesto recorre un **árbol AND/OR**, utilizando heurísticas propias y otras obtenidas de la información recopilada, que permiten podar el espacio de soluciones del problema de forma tal que se pueden obtener resultados aceptables tanto en el aprovechamiento de la carga como en el tiempo que se insume para obtener los mismos.

En forma complementaria, se desarrolló un Sistema para la Gestión de la Carga de Camiones y un programa para graficar los resultados del algoritmo **FHI**.

ABSTRACT

Optimization of the load of trucks with boxes of different sizes

The problem consist in programming the **truck load with boxes of different sizes**, maximizing the space utilization. The load should satisfy the global restrictions of weight distribution on the truck, local restrictions for the type and weight that boxes supports and for load stability restrictions.

This problem has relationship with the **bin packing** and **cutting stock** problems, and it has the characteristic that is a NP-Hard [7]. Therefore there is not, until the present time, an algorithm that allows to solve it in polinomial time, unless it is demonstrated that $P=NP$.

We had developed a **search method** that had been implemented in the **FHI** algorithm, and has the particularity of getting solutions with a load utilization of **p%**.

The **search method** examines an **AND/OR tree**, and it using two heuristics types: the first ones were recopiled from the studied bibliography and the other ones were developed ourselves. The search method allows to prune the solutions space of problem domain, in a way that acceptable result can be obtainted for the optimization of the load in a short time.

In a same way, we developed a Loading Trucks Management System and a program that generates graphics from the **FHI** algorithm output.

- **Keywords**

Packing problems, Cutting stock problems, Three-dimensional packing, Tree search procedures, NP-Hard problems.

II. INTRODUCCION

INTRODUCCION

II.1. Definición del Problema

Una empresa tiene que transportar en un camión un pedido de cajas de diversos productos de un lugar a otro. Las mismas tienen distintas dimensiones y peso.

Las cajas se apilan sobre el piso del camión, las cuales son introducidas manualmente por operarios.

El *objetivo principal* del problema es maximizar el aprovechamiento del espacio en el camión, con restricciones en las cajas, para que no se deterioren, y en el camión, para que se cumplan las normas reglamentarias del transporte de carga y de operatoria, las cuales se detallan a continuación:

1. En las cajas:

- Cada caja tiene un límite máximo de peso que soporta.
- Sólo se pueden apilar cajas cuyo peso específico sea menor que la caja que se encuentra debajo de la misma.
- Ciertas cajas tienen sólo una base de apoyo, limitación que surge del contenido de la misma.
- Las cajas están agrupadas por tipos, los cuales limitan las cajas que pueden ir debajo. Por ejemplo, una caja que contiene un producto líquido no podría ser apilada sobre otra cuyo contenido puede ser deteriorado si se rompiera alguna de las cajas con líquido.

2. En el camión:

- Cada camión soporta un peso máximo.
- El peso debe estar distribuido de acuerdo a sectores dentro del camión. Estos sectores surgen de la posición y la cantidad de ejes que posee el camión.
- Debido a que se utiliza un camión con una única puerta de acceso en el extremo posterior del mismo, es necesario que el operario llene el camión

desde adelante hacia atrás, completando paredes de cajas. Es decir, no está permitido pisar las cajas para no romper los productos.

- No pueden quedar espacios libres entre las paredes y las cajas. En consecuencia las cajas tienen que ejercer presión hacia las 4 paredes del camión para preservar la estabilidad de las mismas. Además, la altura que alcanzan las cajas debe ser uniforme en todo el camión. Por ejemplo, no podría tener todas las cajas en la primera mitad del camión y nada en la segunda mitad.

Debido a que en la elaboración de los pedidos no se tiene en cuenta el tamaño del camión que se va a utilizar, se emplea un tipo de caja **a completar**, con la cual se completa el espacio que queda disponible en el camión (diferencia entre el volumen del camión y el volumen del pedido) y de esta forma aprovechar toda la capacidad del mismo, reduciendo los costos de transporte.

Una característica de la carga es que las cajas deben ser apiladas ortogonalmente respecto de la base del camión y una característica de las cajas es que los tamaños de las mismas no son proporcionales.

Un *objetivo secundario* es minimizar el **índice de mezcla** de las cajas dentro del camión debido a que:

- Es importante optimizar la operatoria de carga, ya que en la empresa se utiliza un sistema de cintas transportadoras para abastecer a los camiones, por lo cual se requiere que las cajas de un mismo tipo se soliciten juntas.
- Además, es necesario realizar un control de stock recibido en el momento de la descarga del camión cuando éste llega al destino del pedido.

Otro *objetivo secundario* es tratar de simplificar la operatoria de carga del camión dado que la misma se realiza en forma manual (no robotizada).

II.2. Clasificación del Problema

El problema de *bin packing de 3 dimensiones* consiste en cargar un conjunto de paralelepípedos (cajas) rectangulares $I = \{b_1, b_2, \dots, b_n\}$ dentro de un paralelepípedo (camión) rectangular B con dimensiones fijas, con el objetivo de que el volumen desperdiciado al cargar I en B sea mínimo.

Claramente, el problema descrito en el punto "definición del problema" pertenece a la categoría de *packing de 3 dimensiones*, que es una generalización natural de los clásicos problemas de packing de 1 y de 2 dimensiones.

En la literatura se pueden encontrar numerosos trabajos que resuelven distintas variantes del problema, por ejemplo, *bin packing* de 1, 2 y 3 dimensiones, cutting stock en 1, 2, y 3 dimensiones, etc. A su vez, para cada uno de estos problemas se presentan distintas variantes a partir de características que poseen los problemas reales.

En general, podemos decir que existe una relación directa entre los problemas de *bin packing* y los de *cutting stock*. Esta relación surge del hecho que cualquier problema de *bin packing*, "poner en", el cual consiste en poner cajas en una caja contenedora, puede verse como un problema de cutting stock, "cortar de", es decir, cortar piezas (cajas) a partir de una pieza de stock original (la caja contenedora), y viceversa.

En consecuencia, para resolver problemas de *bin packing* no se pueden descartar las técnicas para resolver el problema de *cutting stock*, y viceversa.

Si realizamos una abstracción de los detalles particulares de cada problema que se plantea en los trabajos recopilados, podemos encontrar que se pueden dividir en 3 grandes grupos:

1) El primer grupo resuelve distintas variantes del problema de *packing* utilizando *algoritmos* desarrollados a partir de **estrategias básicas**, como ser: First-Fit, First-Fit-Decreasing, First-Fit-Decreasing-Height, Next-Fit, etc. Dichas estrategias permiten obtener **algoritmos aproximados** que se ejecutan en tiempo **polinomial** y que brindan soluciones buenas en determinado espacio del problema, por ejemplo, suponiendo que las bases de las cajas son cuadradas. Li [10], Garey[6], Yao [17], Coffman y Lagarias [2], Smith [14] y Dowsland [5] presentan algoritmos aproximados a partir de estrategias básicas.

En particular, Yao [17], sugiere estudiar el espacio del problema para detectar áreas **malas** y emplear para dichas áreas algoritmos especializados. De esta forma, se obtienen **algoritmos compuestos** que resuelven un área mayor.

Este último resultado lo reafirma el estudio realizado por Dowsland [5], donde se comparan distintos algoritmos para resolver el problema de packing de 2 dimensiones ("carga de pallets con cajas"). Después de estudiar comparativamente distintos algoritmos, se llega a la conclusión que existe una íntima relación entre el algoritmos y el tamaño de las cajas y de los pallets. En particular se detectan algoritmos que son **complementarios**.

En el caso de Coffman y Lagarias [2], se arriba a la conclusión que la performance de algoritmos para *packing* de cuadrados (square-packing) es sensible a la distribución de probabilidades de los tamaños de los cuadrados a ser empaquetados.

Si bien la **performance del peor caso** no está acotada con este tipo de estrategias, se pueden obtener buenos resultados si se aplican al espacio del problema apropiado. Para decidirse por una de estas estrategias básicas se deberá tener en cuenta la relación que tiene con el espacio del problema, inclusive se debe analizar la posibilidad de utilizar algoritmos **compuestos** para cubrir una región mayor.

Finalmente, debemos notar que los tres primeros autores estudian el comportamiento de los algoritmos analizando la **cota asintótica de la performance** (“asymptotic performance bound”), mientras que Coffman y Lagarias [2] lo hace realizando un **análisis probabilístico** (“asymptotic probabilistic analysis”). En ambos casos se pueden sacar conclusiones teóricas generales acerca del comportamiento de dichos algoritmos. Por otra parte, Dowsland [5] comparó los resultados con el valor óptimo, obteniendo de esta forma datos prácticos que cubren, según los autores, los casos que se presentan en la realidad.

2) En el segundo grupo, la aproximación utilizada es plantear el problema como un **problema de programación lineal**. Los autores que utilizan esta aproximación son Gilmore y Gomory [8], Murgolo [12], y Morabito y Arenales [11].

El problema principal que se plantea en la resolución de estos problemas por medio de técnicas de programación lineal, es la cantidad de columnas que forman la matriz de restricciones.

Una suposición importante que realizan todos los autores es que el problema es **no restringido** (no existe limitación en la cantidad de cajas disponibles de un determinado tipo a cargar en el camión), la cual simplifica el problema considerablemente. Morabito y Arenales [11] presentan una aproximación que les permite eliminar esta limitación, la cual entra dentro del tercer grupo de soluciones.

Gilmore y Gommory [8] resuelven el problema de *cutting stock guillotinable* para los casos de 2 y 3 dimensiones donde los patrones de corte están muy limitados, ya que solamente se consideran aquéllos que se pueden realizar en **etapas**, es decir, se reemplaza el **problema multidimensional** por un **problema multietapas**. El hecho de limitar los patrones de corte a ser *guillotinables en etapas* permite resolver el problema utilizando la técnica de generación de columnas que en este caso deriva en un *problema de mochila*.

Los resultados de Gilmore y Gommory [8] son extendidos por Morabito y Arenales [11] para resolver el problema de 3 dimensiones.

Murgolo [12] resuelve el problema de *packing*. Si bien el problema es de programación lineal entera, el mismo se descompone en varios pasos. En el primero se relaja el problema, luego se resuelve el problema de programación lineal, el resultado se convierte en una solución entera y luego se consideran las cajas pequeñas que fueron descartadas en el primer paso. Es importante destacar que el autor considera que esta aproximación solamente tiene interés teórico, excepto para el caso en que el óptimo requiere una cantidad de BINs (contenedores) muy grande.

Resolver el problema como un problema de programación lineal es una aproximación útil para algunos casos, como los que plantean los autores antes referidos. Sin embargo, para los problemas en que no se pueden limitar los cortes a ser guillotinales en etapas y no restringido, las técnicas empleadas no resultan efectivas.

3) En el tercer grupo, Morabito y Arenales [11] y Beasley [1], utilizan **métodos de búsqueda** en un árbol.

Como se mencionó en el punto 2), Morabito y Arenales [11] presentan soluciones de programación lineal para el caso de *cutting-stock guillotnable en etapas* para resolver el problema de *packing de 3 dimensiones*. Además, presentan una aproximación que se aplica a dicho problema, la cual se puede aplicar también a problemas que no requieren que el cutting sea en etapas y/o no restringido. Dicha aproximación se basa en un método de búsqueda en un árbol AND/OR.

Beasley [1] resuelve el problema de *cutting stock de 2 dimensiones no-guillotnable no-restringido*. El problema lo plantea como un problema de programación lineal entera donde las variables pueden tomar valores 0 y 1. Se debe tener presente que la búsqueda sólo se realiza en la 3ra. etapa del algoritmo que presentan. Finalmente, los resultados que ofrece el autor son para tamaños de piezas de stock muy chicas (menos de 30 centímetros) y pocos tipos de piezas a cortar.

Desarrollar *algoritmos basados en estrategias básicas* permite dar solución a problemas de *packing*, pero con muchas limitaciones o características particulares. Se debe tener en cuenta que existe una estrecha relación entre el algoritmo y los datos de entrada. Para extender el área del problema solucionable se debe pensar en algoritmos combinados. Estas técnicas permiten resolver muchos problemas reales pero quedan otros más complejos sin resolver.

Utilizar las técnicas de **programación lineal** permite resolver problemas con serias limitaciones en los patrones de corte (*cutting guillotnable en etapas*).

Si bien la implementación de **métodos de búsqueda** no escapa al problema de la explosión combinatoria, permite obtener soluciones satisfactorias, que no se pueden obtener con las técnicas anteriores. Es decir, permiten encarar problemas más complejos, pudiendo explotar información del dominio del problema [9].

Notar que en el grupo 3) se emplean **métodos de búsqueda** en un árbol, mientras que en los grupos 1) y 2) utilizan **algoritmos determinísticos** [9].

El problema de *bin packing de 1 dimensión* es un problema NP-hard, por lo tanto el problema de *bin packing de 3 dimensiones* es también NP-hard por obvias razones. En

consecuencia, no hay, hasta la actualidad, un algoritmo que permita solucionarlo en tiempo polinomial, a menos que se demuestre que $P=NP$.

II.3. Historia

El trabajo lo iniciamos pensando en desarrollar un algoritmo tomando ideas de la bibliografía recopilada, que utilizan estrategias básicas para resolver algunas variantes del problema de bin packing y cutting stock. Después de varios intentos llegamos a la conclusión que dicha aproximación no nos permitiría obtener buenos resultados.

El próximo paso fue analizar otras aproximaciones, en particular, las que se basaban en técnicas para resolver problemas de programación lineal. Si bien nuestro problema es de programación entera, la condición que restringe a las variables a ser enteras se podría relajar, pero como existen restricciones que no son lineales, llegamos a la conclusión de que no se pueden aplicar dichas técnicas.

Finalmente, decidimos implementar un algoritmo basado en métodos de búsqueda en un árbol. Esta aproximación nos pareció que era la mejor pero teníamos algunas dudas debido a que permanecía latente el problema de explosión combinatoria.

Comenzamos la implementación tratando de buscar toda la información posible acerca del problema que nos permitiera obtener heurísticas capaces de podar suficientemente el árbol sin sacrificar el objetivo del problema.

Una vez implementado el algoritmo comenzamos con el ajuste que nos permitió obtener los resultados que presentamos en este trabajo de Tesis.

Uno de los problemas que se nos presentó fue que resultaba muy complejo chequear los resultados y comprobar que el algoritmo hacía lo que se suponía debía hacer. La primera vez dibujamos el resultado manualmente y fue muy dificultoso. Entonces empezamos a buscar una herramienta que nos permitiera resolver este nuevo problema. La búsqueda fue exitosa ya que encontramos que con el producto Mathematica se podían realizar fácilmente los dibujos de las soluciones de nuestro algoritmo.

Paralelamente desarrollamos un Sistema para la Gestión de la Carga de Camiones que nos permitiera manejar la información necesaria del problema.

Una vez finalizado el algoritmo realizamos las pruebas con 13 casos reales y sacamos las conclusiones.

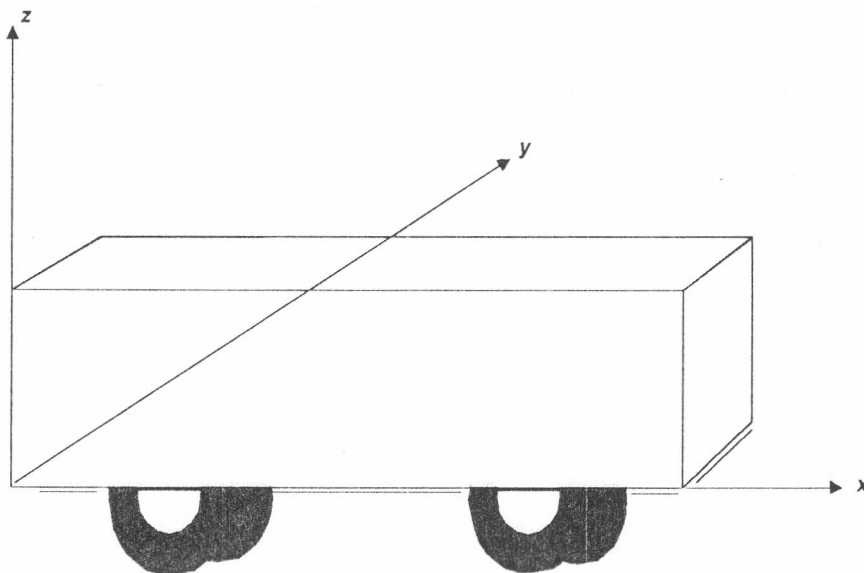
III. GLOSARIO DE TERMINOS

GLOSARIO DE TERMINOS

Camión: Es el **bin** o **container** que se desea llenar con **cajas**. Los atributos son los que se describen a continuación:

- largo
- ancho
- alto
- peso máximo que soporta
- sectores

El **camión (CM)** define un sistema de coordenadas cartesianas donde el **largo** corresponde al eje **x**, el **ancho** corresponde al eje **y** y el **alto** corresponde al eje **z**.

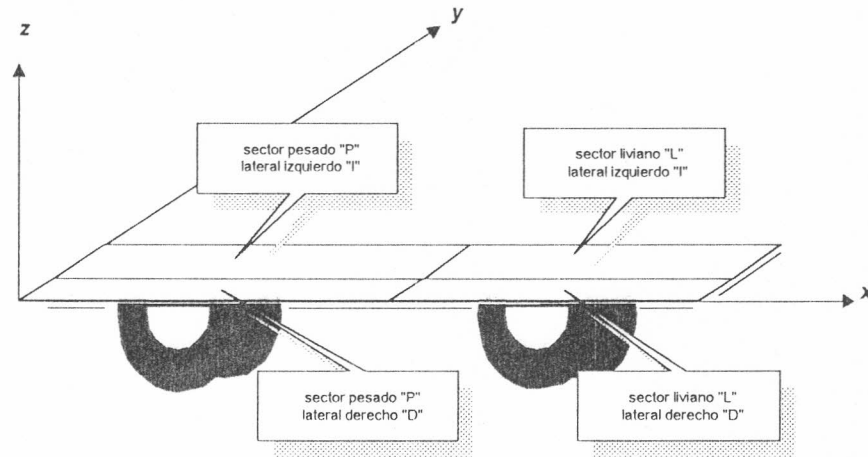


Sectores: Los sectores son áreas dentro del camión. Cada uno de estos sectores impone restricciones de peso en el momento de realizar la carga. Estas restricciones dependen de cada camión, dado que los mismos tienen ejes que difieren en cantidad y resistencia de peso. Para simplificar, se considera que el camión se divide en dos sectores: **liviano (L)** y **pesado (P)**, sin necesidad de ser ambos exactamente del mismo largo.

A su vez, cada sector, se divide en laterales **izquierdo (I)** y **derecho (D)**. Esto se debe a que la carga debe estar distribuida de tal forma que el peso se encuentre equilibrado entre el sector izquierdo y el derecho, es decir, que el peso en el lateral **I** no difiera más de un porcentaje dado respecto del lateral **D**.

Los atributos de cada sector son los que se describen a continuación:

- tipo de sector [L | P]
- inicio: coordenada del eje x donde comienza el sector.
- fin: coordenada del eje x donde termina el sector.
- peso que soporta: es el peso máximo que se puede asignar al sector.
- diferencia entre laterales: es la diferencia máxima de peso que puede haber entre el lateral **I** y el lateral **D** del sector correspondiente.



Cajas: Las cajas (CA) son los objetos a cargar en el camión. Los atributos son los que se describen a continuación:

- largo
- ancho
- alto
- peso
- peso que soporta
- peso específico
- peso específico que soporta
- tipo de caja
- restricciones de tipo de la caja

Pedido: El pedido (PE) es un conjunto de **cajas** de las cuales se requiere cargar una cierta **cantidad solicitada**:

- caja
- cantidad solicitada

Orientaciones: Las cajas tienen varias **orientaciones ortogonales** posibles, una por cada cara de apoyo (6 caras) y para cada cara cuatro rotaciones, es decir, 24 orientaciones.

Formas: Las cajas tienen **orientaciones** posibles, una por cada base de apoyo.

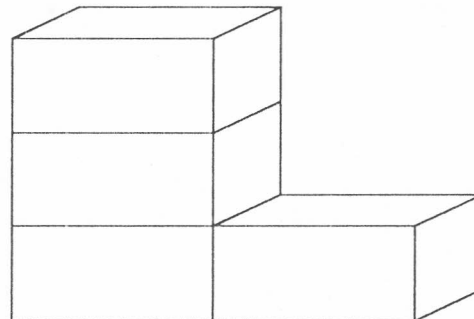
Dadas n instancias de una caja, una **forma (FO)** queda definida por:

- **orientación**
- cantidad de cajas en el eje x
- cantidad de cajas en el eje y
- cantidad de cajas en el eje z

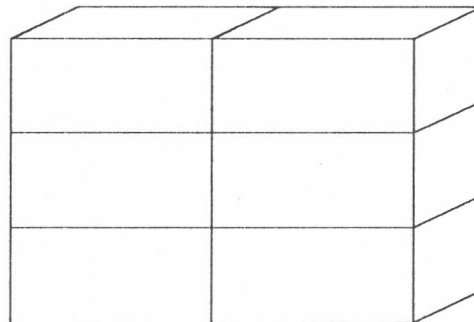
donde $n = x \times y \times z$

Esta definición hace referencia a **formas regulares**. El resto de las formas, las **formas irregulares**, se pueden generar a partir de las anteriores.

Forma irregular



Forma regular

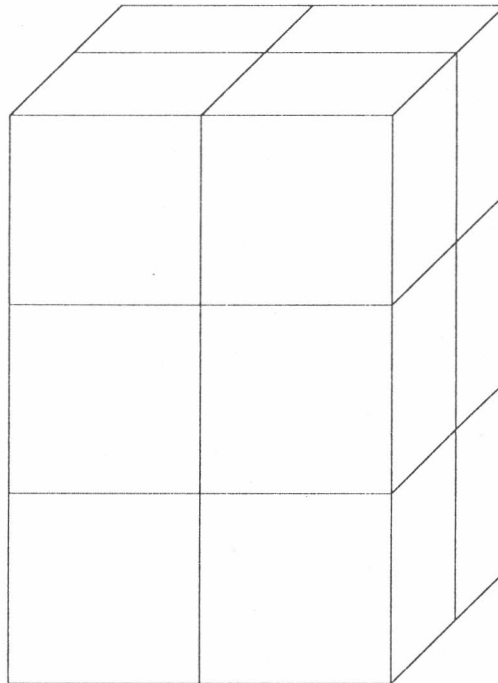


Plataforma vacía: Una plataforma vacía (**PL**) es una superficie de apoyo en la cual se pueden poner cajas. Las mismas pueden estar directamente en la base del camión o pueden estar sobre bloques de cajas colocadas previamente.

Una plataforma vacía se describe de la siguiente forma:

- coordenadas origen (x, y, z).
- superficie de apoyo ($\Delta x, \Delta y$): es el ancho y el largo de la plataforma vacía.
- altura (Δz): es la altura vacía que hay entre la plataforma vacía y el techo del camión.
- restricciones de tipo: es una lista de todos los tipos de cajas que **no** pueden ir sobre la plataforma vacía.
- restricciones de peso.

Bloque: Un bloque (**BL**) es un conjunto de instancias de una **caja CA** dada, apiladas con determinada **forma FO**, las cuales son colocadas en una **plataforma vacía PV** dada. El bloque es la unidad de carga que se considera en el algoritmo (ver descripción del algoritmo).



IV. DEFINICION FORMAL DEL PROBLEMA

DEFINICION FORMAL DEL PROBLEMA

Sea n la cantidad de tipos de cajas que conforman el pedido en la lista LCAP (Lista de Cajas Pedidas), a_j representa la cantidad de cajas de tipo j , x_j , y_j , z_j son el largo, ancho y alto de la caja de tipo j respectivamente, v_j representa el volumen de la caja de tipo j , $bmax_j$, $bmin_j$ son las cantidades máxima y mínima pedidas de la caja tipo j respectivamente, p_j es el peso de la caja de tipo j . X largo del camión, Y ancho del camión, Z alto del camión, V representa el volumen total del camión, P es el peso máximo que puede transportar el camión, P_k es el peso máximo que soporta el sector k , DML_k diferencia de peso máxima entre laterales, $p^x(c_j)$, $p^y(c_j)$, $p^z(c_j)$ son las proyecciones de la caja c_j sobre los ejes x , y y z respectivamente, $f(p_j \cdot a_j, l, k)$ es una función que asigna la proporción del peso de las cajas de tipo j que geométricamente estén contenidas en forma total o parcial en el lateral l del sector k .

$$\text{Maximizar } \sum_{j=1}^n v_j \cdot a_j$$

sujeto a

$$bmin_j \leq a_j \leq bmax_j \quad j=1 \dots n$$

$$\sum_{j=1}^n v_j \cdot a_j \leq V$$

$$\sum_{j=1}^n p_j \cdot a_j \leq P$$

$$\sum_{j=1}^n f(p_j \cdot a_j, I, k) + \sum_{j=1}^n f(p_j \cdot a_j, D, k) \leq P_k \quad k=1 \dots 2$$

$$\left| \sum_{j=1}^n f(p_j \cdot a_j, I, k) - \sum_{j=1}^n f(p_j \cdot a_j, D, k) \right| \leq DML_k \quad k=1 \dots 2$$

$$p^x(c_j) + x_j \leq X \quad j=1 \dots n$$

$$p^y(c_j) + y_j \leq Y \quad j=1 \dots n$$

$$p^z(c_j) + z_j \leq Z \quad j=1 \dots n$$

• **Otras Restricciones:**

- ◇ Contenido de las cajas
- ◇ Estabilidad de la carga
- ◇ Peso que pueden soportar las cajas
- ◇ Controlar el índice de mezcla de un tipo de caja dentro de la carga

Estas restricciones no se expresaron matemáticamente debido al alto grado de dificultad para representarlas. Si se expresaran dicha restricciones matemáticamente, aún tendríamos que encontrar una técnica que permita resolver el modelo expresado.

V. DISEÑO DEL ALGORITMO

DISEÑO DEL ALGORITMO

V.1. Representación Formal del Problema

Dado el problema P y dada la Base de Datos Global (BDG), se comienza por un Estado Inicial (I o E_0) y por medio de Operaciones (O) que se aplican a la BDG, se pasa del Estado E_i al E_{i+1} . Estas Reglas se aplican hasta que se cumpla la Condición de Terminación u Objetivo (C).

$$P = \langle I, O, C \rangle$$

La Base de Datos Global es la siguiente:

BDG [LPV, LCAP, LBL, PA]

donde

LPV = Lista de Plataformas Vacías (PV)

LCAP = Lista de Cajas Pedidas (CAP)

LBL = Lista de Bloques (BL)

PA = Parámetros

y

BL[CM, PV, CA, FO, LCA, LFO, GL]

CM = Camión compuesto por una Lista de Sectores del Camión (SE)

PV = Plataforma Vacía donde se apoya BL, que es un Elemento de LPV

CA = Caja utilizada para BL, que es un Elemento de LCAP

FO = Forma de BL

LCA = Subconjunto de la LCAP donde CA no está incluida

LFO = Lista de Formas de BL donde FO no está incluida

GL = Datos Globales

El *estado Inicial* E_0 está representado por todas las Listas vacías menos la Lista de Plataformas Vacías (LPV) que está compuesta por un solo elemento que es la superficie total del camión, la Lista de Cajas Pedidas (LCAP) y las estructuras de Datos Globales y de Parámetros.

Cambiar del Estado E_i al E_{i+1} significa que haya un cambio en la BDG, y este cambio se realiza a partir de la aplicación de las Operaciones que intervienen en el Problema.

En nuestro caso particular existe una Operador que es acomodar un BL en CM.

Las *condiciones* para aplicar el Operador son cumplir con las siguientes Restricciones:

Globales sobre el Camión:

- de Peso que soportan los Sectores del Camión
- de Peso que soporta el Camión
- de diferencia de Peso entre los Laterales de los Sectores del Camión
- de Desperdicio de la Carga

Locales sobre las Plataformas:

- de Contenido
- Geométricas
- de Peso que soportan las Cajas

Las *acciones* que se toman si se cumplen las condiciones son:

- Incrementar el Peso sobre el Sector y en el Camión
- Generar nuevas PV en LPV
- Decrementar la cantidad de cajas disponibles de LCAP
- Incrementar la cantidad de cajas asignadas de LCAP
- Eliminar CA de LCAP cuando se hayan acomodado la cantidad mínima
- Sacar PV de LPV

Al cambiar de E_i a E_{i+1} se verifica el *Objetivo* del Problema para decidir cual es el camino a seguir. El *Estado Objetivo* es aquel en que LPV o LCAP están vacías y el aprovechamiento haya superado el mínimo solicitado.

V.2. Espacio de Búsqueda

El **espacio de búsqueda** lo podemos representar por medio de un ARBOL AND/OR compuesto por 3 tipos de nodos:

- Plataformas vacías (PV)
- Cajas (CA)
- Formas (FO)

El *Nodo Inicial* es la plataforma vacía correspondiente al camión vacío (PV_0).

En esta plataforma vacía se acomoda una de las cajas $CA_1, CA_2, \dots, CA_n$, de acuerdo a las reglas impuestas. Se debe tener en cuenta que para la caja CA_i , se necesita cargar cierta cantidad de instancias cs_i .

Por último, dadas PV_0 y CA_i , se puede armar un Bloque (BL_1) de cajas con instancias de CA_i para poner en PV_0 , utilizando alguna de las formas $FO_1, FO_2, \dots, FO_m$.

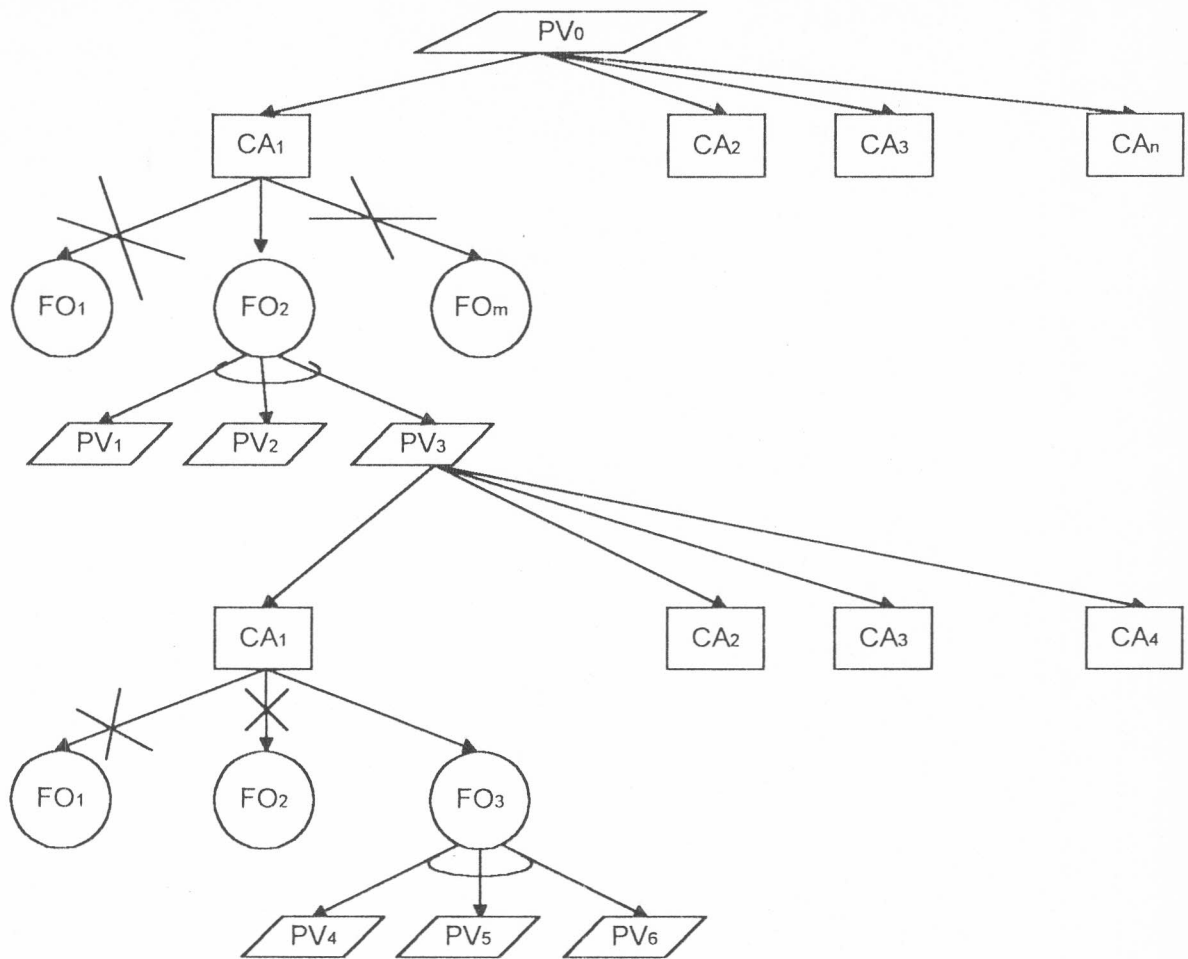
Este proceso de colocar un bloque BL_i de cajas CA_i en la plataforma PV_j genera **nuevas plataformas vacías** $PV_{j1}, PV_{j2}, \dots, PV_{jk}$. Es decir, se produce un cambio de Estado en la BDG, donde se plantean k nuevos problemas ("reducción de problemas"), los cuales **no son independientes entre sí** ("no conmutativo"[9]). Esto se debe a que el **problema es restringido** [11], con lo cual, las cajas disponibles en el momento de intentar completar una PV dependen del orden en que se recorre la lista LPV. Por ejemplo, dado un nodo A que posee dos sucesores AND, sean B y C, no es equivalente resolver primero B y después C, que resolver C y después B.

Nota: El hecho de ser un **problema restringido** no significa que tiene restricciones, sino que la cantidad de cajas a ser cargadas, de un determinado tipo, está limitado por un máximo ($bmax_j$) y un mínimo ($bmin_j$). Es condición suficiente para que sea un problema restringido que: $bmax_j < [X/x_j][Y/y_j][Z/z_j]$, donde x_j, y_j , y z_j son el largo, ancho y alto de la caja de tipo j respectivamente, X largo del camión, Y ancho del camión y Z alto del camión.

En resumen, el árbol de búsqueda comienza con el nodo inicial PV_0 , se abre en varias ramas (OR), una por cada caja CA , y a su vez, cada nodo CA se abre en varias ramas (OR), una por cada forma FO . Esta terna de nodos (PV, CA, FO) genera nuevas plataformas vacías, es decir, ramas AND, las cuales deben ser resueltas para arribar a la solución.

Una solución está dada por un **subárbol AND/OR**, cuyos nodos finales cumplen con el Objetivo del Problema que es que LPV o LCAP estén vacías y el aprovechamiento haya superado el mínimo solicitado. Este Objetivo tiene que ser verificado cada vez que se arma un Bloque para saber si se está en un camino correcto, si se llegó a la solución o decidir que acciones tomar en el caso de que no se cumplan las restricciones. El árbol de decisión para verificar los objetivos es descrito en la sección de Implementación del Algoritmo.

El árbol que se genera se detalla a continuación:



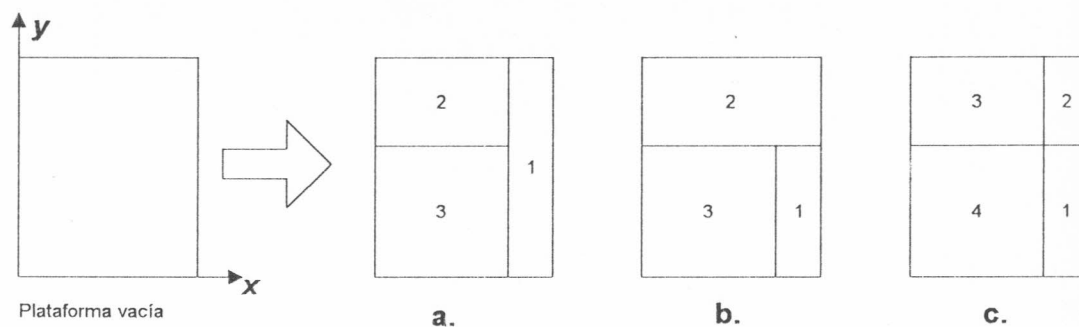
V.3. Heurísticas para Restringir el Espacio de Búsqueda

Dado que el árbol de búsqueda es demasiado grande para ser examinado exhaustivamente, empleamos varias heurísticas que permiten eliminar ramas no prometedoras.

1. Heurística de extracción de plataformas vacías:

Generar plataformas vacías: son un conjunto de **plataformas vacías nuevas** $\{PV_1, \dots, PV_n\}$ que se surgen después de poner un **bloque BL**.

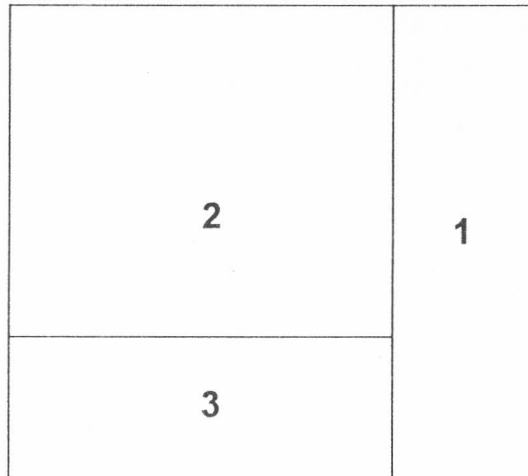
Existen básicamente 3 criterios para generar plataformas vacías a partir de una dada. En la siguiente figura se muestra una plataforma vacía (vista desde arriba) y, después de poner un bloque en el área 3, se generan nuevas plataformas vacías.



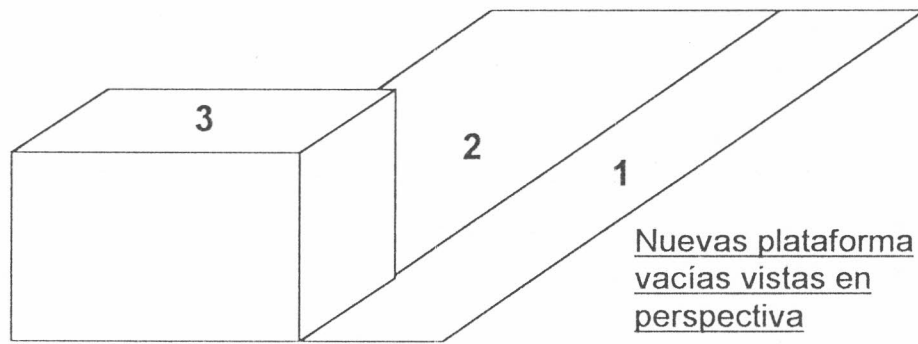
Decidimos utilizar el caso **a.** para generar plataformas vacías por los siguientes motivos:

- i. Coincide con un criterio más general para cargar el camión, que es completar **lechos** en el sentido transversal.
- ii. El caso **b.** tiende a generar plataformas en el sentido longitudinal, por ejemplo, al colocar el primer bloque en el camión, la nueva plataforma vacía correspondiente al sector 2, ocuparía todo el largo del camión.
- iii. La desventaja de utilizar el caso **c.** radica en que se produce una mayor **fragmentación** y no aporta ventajas respecto de los casos **a.** y **b.** ya que es un caso particular de éstos.

Nota: hay otras formas de generar.



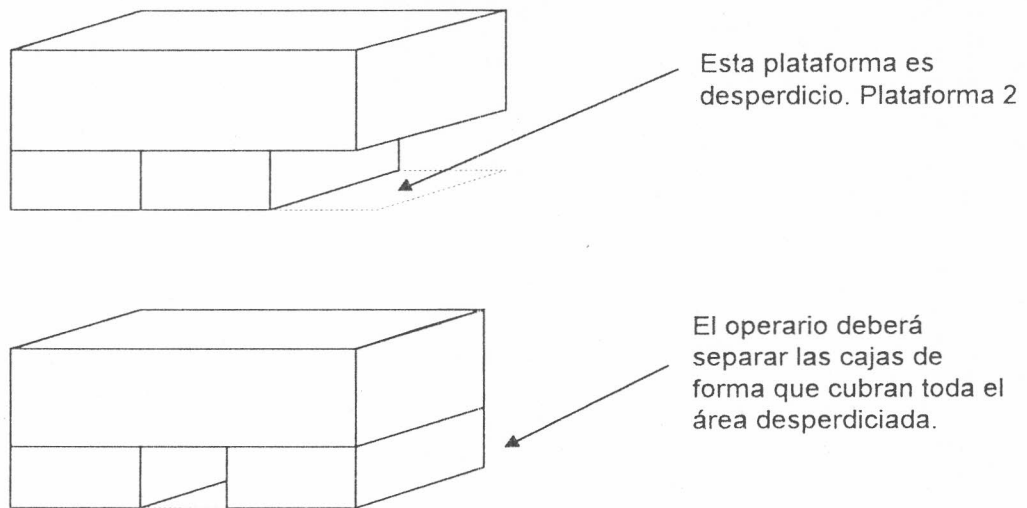
Nuevas plataformas
vacías vistas desde
arriba.



Nuevas plataforma
vacías vistas en
perspectiva

Según el criterio descripto anteriormente, sólo se generan algunos tipos de plataformas, con lo cual se reduce el espacio de búsqueda.

Un concepto que hay que destacar es la *Estabilidad Flexible de la Carga*. Cuando se coloca un Bloque de un Tipo de Cajas sobre una Plataforma Vacía, puede ocurrir que dicho Bloque no ocupe exactamente la base de la Plataforma y por lo tanto se genera un espacio libre que podrá ser una nueva Plataforma Vacía. Si en este espacio libre no entra la menor caja de todas las disponibles, entonces este espacio es llamado “desperdicio” y el Bloque que se está colocando generará una superficie de apoyo menor que la que ocupó y por lo tanto se crearán Plataformas cada vez más pequeñas. Para evitar esto, las plataformas que se crean tomarán este desperdicio y lo agregaran a su superficie real creando Plataformas más grandes para que así se puedan acomodar sobre ellas bloques del mayor tamaño posible. Esto se puede lograr permitiendo distribuir las cajas que conforman al bloque de forma de cubrir la superficie total de la plataforma que lo sustenta como vemos en la siguiente figura:



2. Heurística de selección de plataformas vacías (HSPV): “X ASCENDENTE”.

Debido a que este es un **problema restringido**, las plataformas vacías (LPV) **no son independientes entre sí**, por lo tanto, al establecer un orden para recorrer dicha lista se reduce el espacio de búsqueda.

Decidimos aplicar la heurística “X ASCENDENTE” para recorrer LPV, esto es, ordenadas por la coordenada X de la plataforma vacía en forma ascendente.

Esta heurística, además de reducir el espacio de búsqueda, tiende a llenar **lechos** del camión de forma tal que se puede conocer el desperdicio “por tramos”. En consecuencia se puede aplicar la cota descrita en *Bondad de la Solución*, para podar el árbol.

Bondad de la Solución: dado que la carga se realiza a lo largo del eje X, es posible controlar, en el momento que se selecciona una Plataforma Vacía para ser utilizada cuya base se encuentre en $y=0$ y $z=0$ para un x dado, el aprovechamiento relativo al punto donde se encuentra comparándolo con la proporción del desperdicio máximo permitido ya que si en ese punto el desperdicio acumulado es significativo es muy probable que no se encuentre en un camino prometedor y por lo tanto hay que deshacer el camino que hasta ese momento se había obtenido.

Este desperdicio máximo permitido es fijo para una solución (**p% de aprovechamiento**) y se decrementa a medida que se obtienen soluciones (se incrementa el aprovechamiento).

Otras alternativas a esta solución podrían haber sido aplicar la heurística “Z ascendente” o “superficie de PV ascendente”, entre otras.

La heurística “Z ascendente” tiende a completar el camión por niveles, por lo tanto no podemos conocer la bondad de la solución correspondiente a la rama que se está examinando hasta que se completa todo el camión.

3. Heurística de extracción de cajas (HEC): Extraer las cajas que satisfacen las siguientes reglas:

- Restricción de Peso Específico (depende de la plataforma vacía)
- Restricciones de Tipo (depende de la plataforma vacía).
- Limitación geométrica: extraer una caja si entra por lo menos 1 instancia de la caja en alguna de las orientaciones posibles.

4. Heurística de refinamiento de cajas (HRC): No usamos.

5. Heurística de selección de cajas (HSC): Utilizamos alguna de las siguientes heurísticas:

- a) Seleccionar la caja de mayor Peso Específico.
- b) Peso Específico y Volumen Pedido:
 - Si la PV está en el piso del camión
seleccionar la caja de mayor peso específico.
 - Sino
seleccionar la caja de mayor volumen pedido.
 - Fin Si
- c) Seleccionar la caja de mayor Volumen Pedido

Dado que todas las heurísticas de selección de cajas mantienen ordenada la lista LCA siempre de la misma forma, independientemente de la cantidad asignada y se selecciona el primer elemento de la misma, esto asegura que el *Indice de Mezcla* de la carga sea bajo, ya que si una caja fue colocada en un bloque se tratará de tomar en el siguiente bloque.

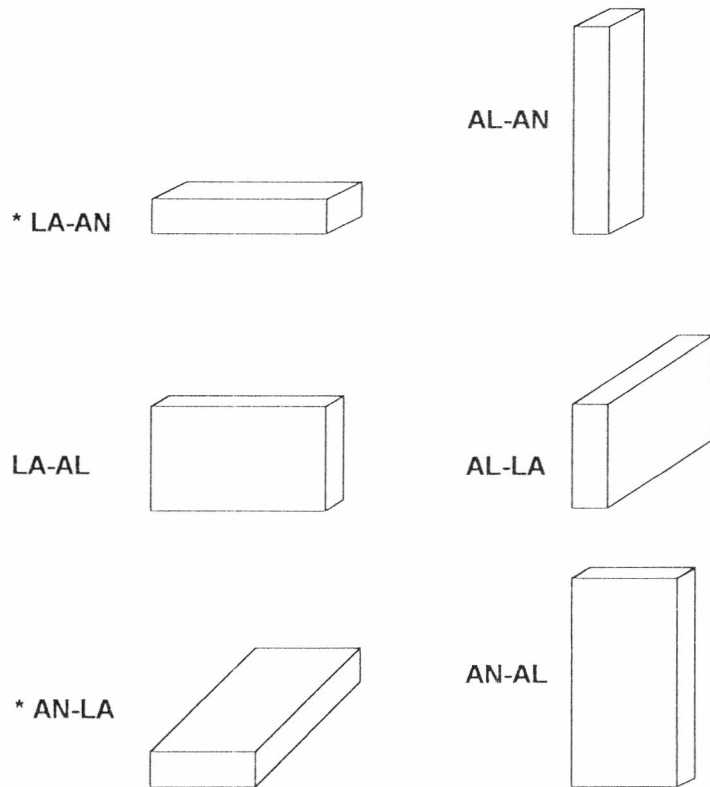
6. Heurísticas de extracción de Formas (HEF): Utilizamos las siguientes heurísticas.

- Solamente se extraen **formas** de acuerdo a las limitaciones que imponen la PV_i y CA_i
- Heurística para extraer formas regulares.
- Heurística de “Largo de Plataforma Vacía Acotada”
 - Si PV está en el piso
 - los bloques no deben superar un largo dado (ej. 2 mts.) y de esta manera las formas tienden a ser altas.
 - Fin Si

- Regla: Respetar las limitaciones de peso.

Para considerar las dimensiones de las cajas, necesarias para el cálculo de las formas, se limita la cantidad de **orientaciones**. Como se mencionó en el Glosario, cada caja tiene 24 orientaciones ortogonales. Eliminando las orientaciones geoméricamente equivalentes obtenemos un conjunto de 6. Sin embargo en la práctica interesa utilizar solo 2 (Son las marcadas con * en la Figura de las Orientaciones). Esto se debe a los siguientes motivos:

- Hay cajas que por problemas de estibaje (flechas de estibaje) solamente tienen 2 orientaciones, por ejemplo aquellas cajas que contienen productos que se pueden derramar.
- El resto de las cajas se pueden colocar en cualquiera de las 6 orientaciones posibles, sin embargo, algunas son mejores que otras en cuanto a la cantidad de niveles apilables. Esto se debe a que las condiciones de estibaje varían según la orientación.
- Además, se tomó en cuenta un problema operativo, esto es, resulta poco práctico para los operarios tener que controlar qué orientación debe utilizarse para estibar la caja. Por otra parte, implicaría un mayor tiempo de carga.



Orientaciones (LA=Largo, AN=Ancho y AL=Alto)
La primera coordenada es paralela al Largo del Camión

Es importante notar que desde el punto de vista del algoritmo, esta simplificación, conduce a una mayor performance, debido a que el espacio de búsqueda es reducido considerablemente.

7. Heurística de Refinamiento de Formas (HRF):

Se ordenan las formas de acuerdo a una de las heurísticas que se detallan a continuación y se refinan todas menos la primera (criterio de **único sucesor**):

- a) Volumen de la Caja Seleccionada por la cantidad de cajas de las Formas Extraídas (Volumen de la Formas Extraídas) y si hay distintas Formas que tienen el mismo Volumen se ordenan primero por las mas anchas y luego por las largas de las Formas.
- b) Superficie de la Base y si hay bases iguales se ordenan en forma descendente de altura.
- c) Ancho, largo y alto de las Formas Extraídas en orden descendente.

8. Heurística de Selección de Formas (HSF):

Se selecciona la primera forma de la lista de Formas Extraídas.

V.4. Estrategia de Búsqueda

La Estrategia de Búsqueda en el árbol descrito es *Depth First* con *Backtracking Dirigido*.

Depth First: Una vez expandida la raíz, siendo una alternativa tan buena como cualquier otra, se elige la primera (por convención la más a la izquierda), y se expande ésta ignorando momentáneamente el resto. Se sigue de esta manera hacia lo profundo hasta que se encuentra la solución o no hay más sucesores. En este último caso se retrocede (backtracking) hasta el nodo más reciente que contenga una alternativa no explorada.

Backtracking Dirigido: Cuando no se encuentran sucesores que cumplan las condiciones para ser expandidos, se realiza un Backtracking analizando la causa de la falla y retrocediendo hasta el nodo que la originó, sin utilizar información adicional. Saltea el o los Estados que tienen elementos que originaron la falla.

VI. IMPLEMENTACION DEL ALGORITMO FHI

IMPLEMENTACION DEL ALGORITMO FHI

Como vimos en la sección de Diseño del Algoritmo, el Espacio Búsqueda se representó con un ARBOL AND/OR, y para implementarlo se utilizó una estructura de Pila en donde se almacenan los distintos estados de la Base de Datos Global. Dicha estructura de Pila es una Lista de Bloques (*LBL*) en donde se encuentra toda la información acerca de los Bloques (*BL*) que se armaron para realizar la carga de las cajas dentro del camión y además, todas las listas adicionales para que en caso de que el Algoritmo no se encuentre en un camino prometedor para obtener una solución y se deba volver atrás o si se halló una solución y se desea continuar con la búsqueda de una mejor, se pueda restaurar el Estado de la BDG para continuar con el Algoritmo.

VI.1. Algoritmo FHI

1. Inicializar la Estructura de Datos Global
2. Mientras no se consuma el tiempo de Proceso y no se haya recorrido todo el Arbol de Búsqueda
 - 2.1 Seleccionar Plataforma Vacía
 - Si LPV es vacía
 - Ir al Paso 2.9 (*Verificar Objetivo*)
 - Sino
 - Si el Desperdicio Acumulado relativo a la posición de la Plataforma es superior al Desperdicio Máximo Permitido proporcional (Bondad de la Solución en la Sección Diseño del Algoritmo) entonces
 - Ejecutar el Proceso *Backtracking Dirigido*
 - Sino
 - Seleccionar la primera Plataforma Vacía (PV) de la Lista de Plataformas Vacías (LPV) - Estado E_i (ver módulo *Seleccionar Plataforma Vacía*)
 - Eliminar PV de LPV
 - Fin Si
 - Fin Si

2.2. Extraer Cajas

- Extraer la Lista de Cajas (LCA) de la Lista de Cajas Pedidas (LCAP) de acuerdo a aquellas cajas que cumplen con las restricciones de la Plataforma Vacía seleccionada (ver módulo *Extraer Cajas*)
- Si la Lista LCA es vacía entonces
 - Ejecutar el Proceso *Backtracking Dirigido*
- Fin Si

2.3. Refinar Cajas

- Refinar la Lista de Cajas extraídas (LCA) borrándolas de la lista de acuerdo a las Heurísticas de Refinamiento de Cajas (En particular no se realiza refinamiento alguno, pero este paso no se elimina porque en caso de ser necesario, puede ser implementado en este punto) (ver módulo *Refinar Cajas*)
- Si la Lista LCA es vacía entonces
 - Ejecutar el Proceso *Backtracking Dirigido*
- Fin Si

2.4. Seleccionar Caja

- Si la Lista LCA es vacía entonces
 - Ejecutar el Proceso *Backtracking Dirigido*
- Sino
 - Ordenar LCA de acuerdo a la Heurística de Selección de Caja (HSC)
 - Seleccionar una Caja (CA) de la Lista de Cajas extraídas refinadas (LCA) de acuerdo a las Heurísticas de Selección de Cajas (ver módulo *Seleccionar Cajas*)
 - Eliminar CA de LCA
- Fin Si

2.5. Extraer Formas

- Extraer la Lista de Formas (LFO) en que se puede acomodar determinada cantidad de cajas CA dado el volumen de PV en función de las restricciones de peso que soportan las cajas apiladas debajo de PV y las propias de CA (ver módulo *Extraer Formas*)
- Si la Lista LFO es vacía entonces
 - Ir al Paso 2.4. (Seleccionar Caja)
- Fin Si

2.6. Refinar Formas

- Ordenar LFO de acuerdo a la Heurística de Selección de Formas (HSF)
- Refinar la Lista de Formas extraídas (LFO) borrando de la lista de acuerdo a las Heurísticas de Refinamiento de Formas (Este refinamiento es en forma irrevocable) (ver módulo *Refinar Formas*)
- Si la Lista LFO es vacía entonces

- Ir al Paso 2.4. (Seleccionar Caja)
- Fin Si

2.7. Seleccionar Forma

- Si la Lista LFO es vacía entonces
 - Ir al Paso 2.4. (Seleccionar Caja)
- Fin Si
- Seleccionar una Forma (FO) de la Lista de Formas extraídas refinadas (LFO) de acuerdo a las Heurísticas de Selección de Formas (ver módulo *Seleccionar Formas*)
- Eliminar FO de LFO

2.8. Poner Bloque

- Armar un Bloque (BL) en la Plataforma Vacía PV con la Caja CA de la Forma FO - Estado E_{i+1} .
- Generar nuevas Plataformas Vacías.
- Si se asignaron todas las cajas solicitadas de CA, entonces eliminarla de LCAP.
- Actualizar las estructuras de Datos Globales (DG) y de Parámetros (PA)
- Agregar BL a la Pila de Bloques (LBL) con las listas LCA y LFO y las estructuras de CA, LSE y GL, para que si no se encuentra en un camino alejado de la solución se pueda volver hacia atrás (ver módulo *Poner Bloque*)

2.9. Verificar el Objetivo del Problema

- Si se agotó el Tiempo Máximo de Proceso entonces
 - Ir al Paso 3 (Finalizar el Algoritmo FHI)
- Sino
 - Si el Camino Actual hacia la Solución no es prometedor
 - Ejecutar el Proceso *Backtracking Dirigido*
 - Sino
 - Si se obtuvo una Solución
 - Decrementar el Desperdicio Máximo Permitido
 - Restaurar la BDG hasta un Estado que cumpla con el Desperdicio Máximo Permitido
 - Ir al Paso 2.4. (Seleccionar Caja)
 - Fin Si
 - Fin Si
- Fin Mientras

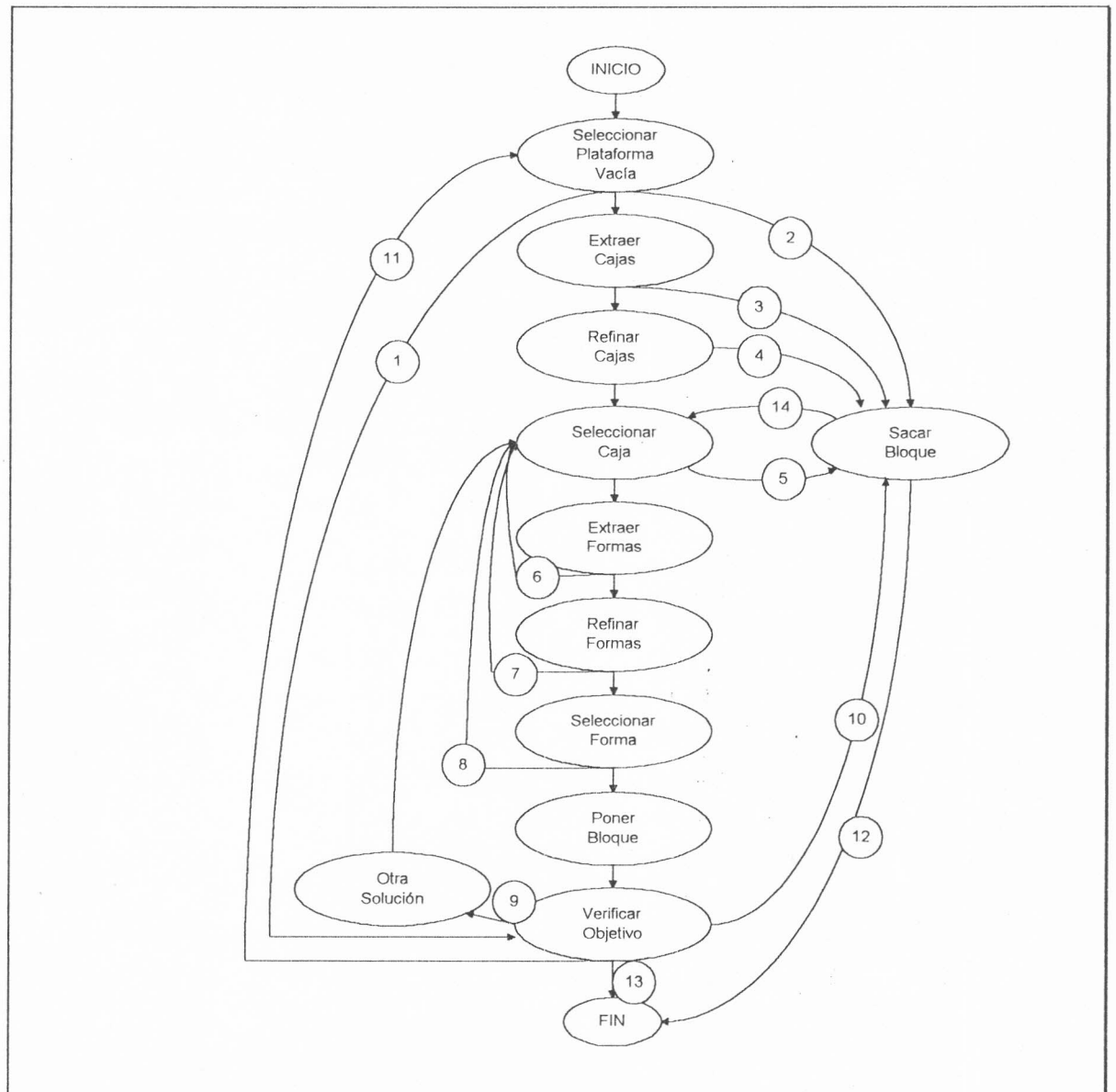
3. Finalizar el Algoritmo FHI

Proceso *Backtracking Dirigido*

- Si se recorrió todo el Arbol de Búsqueda entonces
 - Ir al Paso 3 (Finalizar el Algoritmo FHI)
- Sino
 - Desapilar de LBL restaurando la BDG hasta llegar al Estado que contiene el Bloque que dio origen a la Plataforma que tiene el problema (ver módulo *Sacar Bloque*)
 - Ir al Paso 2.4. (Seleccionar Caja)
- Fin Si

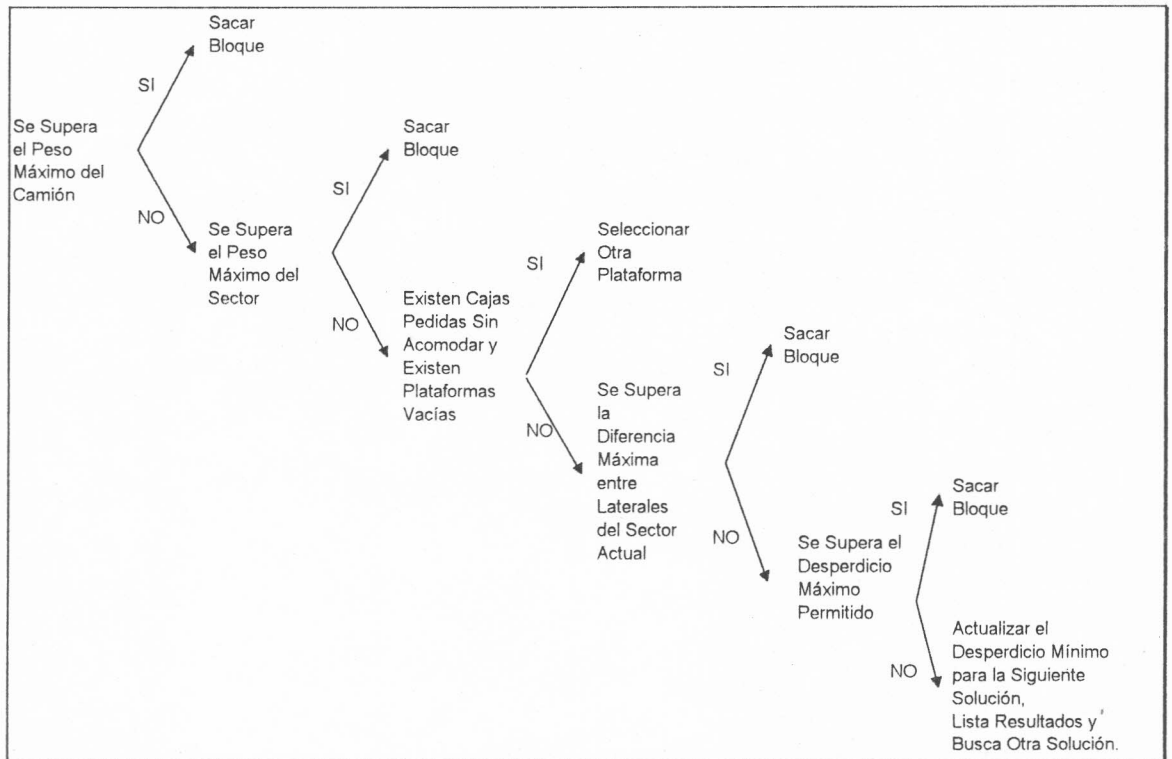
Fin Backtracking Dirigido

VI.2. Diagrama de Transición de Estados

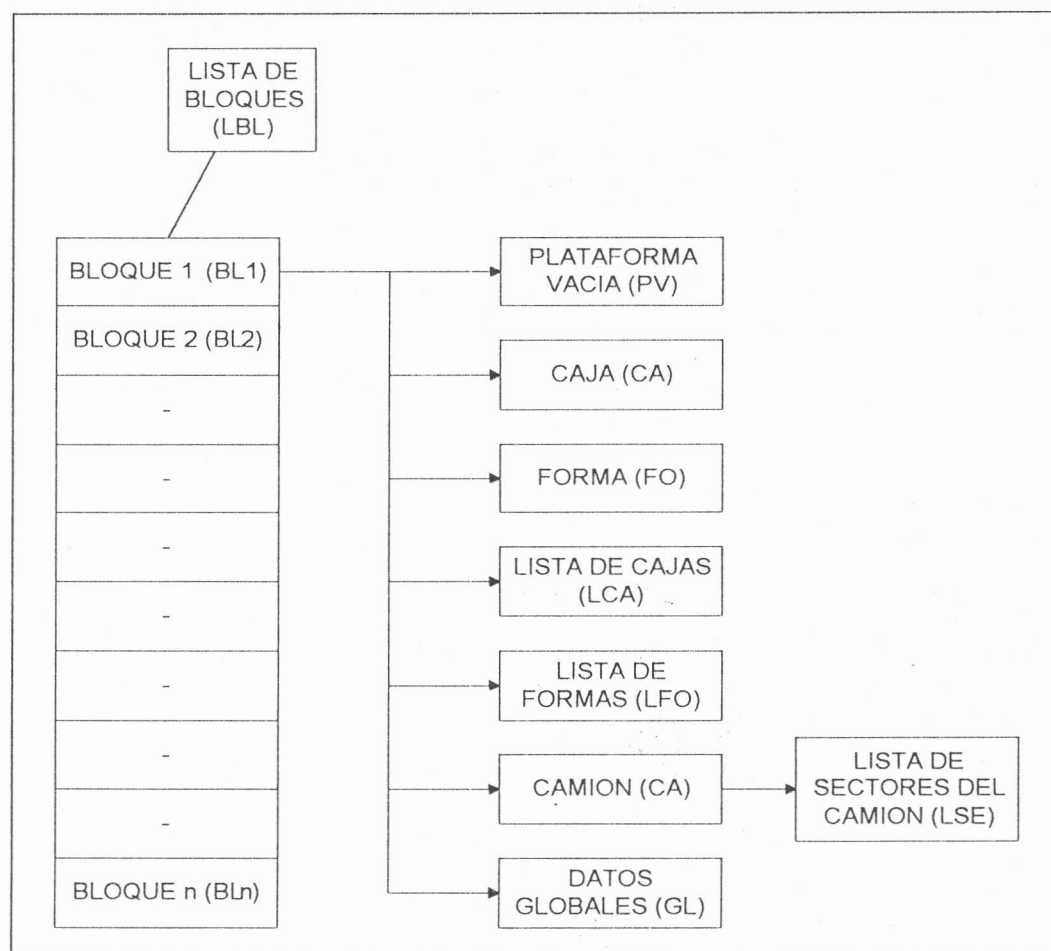


1. No existen Plataformas Vacías.
2. El Desperdicio Acumulado supera el Desperdicio Máximo Permitido.
3. La lista de Cajas está vacía.
4. La lista de Cajas está vacía.
5. La lista de Cajas está vacía.
6. La lista de Formas está vacía.
7. La lista de Formas está vacía.
8. La lista de Formas está vacía.
9. Se obtuvo una Solución y se busca otra.
10. El camino actual no es prometedor.
11. Continuar con otra Plataforma Vacía.
12. Se recorrió todo el árbol de búsqueda.
13. Se agotó el Tiempo Máximo de Proceso.
14. Si no se recorrió todo el árbol de búsqueda entonces luego de Sacar Bloque se Selecciona Caja.

VI.3. Arbol de Decisión del Módulo “Verificar Objetivo”



VI.4. Estructuras de la Base de Datos Global BDG



♦ PLATAFORMA VACIA (PV)

- ° N° Plataforma
- ° N° Plataforma que la originó
- ° Posición de Comienzo en el eje X
- ° Posición de Comienzo en el eje Y
- ° Posición de Comienzo en el eje Z
- ° Tamaño de la Plataforma en el eje X
- ° Tamaño de la Plataforma en el eje Y
- ° Restricción de Peso Específico que soporta la Plataforma
- ° Lista de Tipos de Caja que no pueden colocarse sobre la Plataforma
- ° Lista de los Tipos de Caja que se colocaron debajo de la Plataforma y del Peso que se puede apilar sobre ésta para los Tipos de Caja
- ° Código del Sector del Camión al que pertenece la Plataforma

♦ LISTA DE PLATAFORMAS VACIAS (*LPV*)

Lista de Plataformas Vacías (*PV*)

♦ CAJA PEDIDA (*CAP*)

- ° Código de Caja
- ° Largo de la Caja
- ° Ancho de la Caja
- ° Peso de la Caja
- ° Peso que Soporta la Caja
- ° Peso Específico de la Caja
- ° Peso Específico que Soporta la Caja
- ° Código de Presentación de la Caja
- ° Lista de Cajas que No se Pueden Colocar Arriba de la Caja
- ° Cantidad de Cajas Pedidas
- ° Cantidad Mínima de Cajas Pedidas
- ° Cantidad Asignada de Cajas
- ° Volumen de la Caja
- ° Volumen Total de las Cajas Pedidas

♦ LISTA DE CAJAS PEDIDAS (*LCAP*)

Lista de Cajas Pedidas (*CAP*)

♦ LISTA DE CAJAS (*LCA*)

Subconjunto de la Lista de Cajas Pedidas (*LCAP*)

♦ CAJA (*CA*)

Un elemento de la Lista de Cajas (*LCAP*)

♦ FORMA (*FO*)

- ° Orientación de la Caja
- ° Cantidad de Cajas en el eje X
- ° Cantidad de Cajas en el eje Y
- ° Cantidad de Cajas en el eje Z

♦ LISTA DE FORMAS (*LFO*)

Lista de Formas (*FO*)

♦ CAMION (*CA*)

- ° Lista de Sectores que componen el Camión (*LSE*)
- ° Largo del Camión
- ° Ancho del Camión
- ° Alto del Camión
- ° Peso Neto que Soporta el Camión
- ° Peso Acumulado en la Carga
- ° Volumen Acumulado en la Carga

♦ SECTOR DEL CAMION (*SE*)

- ° Código del Sector del Camión
- ° Comienzo del Sector en el eje X
- ° Fin del Sector en el eje X
- ° Peso que Soporta el Sector
- ° Peso Acumulado en el Lateral Izquierdo del Sector
- ° Peso Acumulado en el Lateral Derecho del Sector
- ° Diferencia de Peso Acumulado Máxima Permitida entre los Laterales del Sector

♦ LISTA DE SECTORES DEL CAMION (*LSE*)

Lista de Sectores del Camión (*SE*)

♦ DATOS GLOBALES (*GL*)

- ° Código de la Caja de Menor Largo Durante la Carga
- ° Código de la Caja de Menor Ancho Durante la Carga
- ° Código de la Caja de Menor Alto Durante la Carga
- ° Volumen del Desperdicio Máximo Permitido

♦ PARAMETROS (*PA*)

- ° N° del Pedido
- ° Código del Camión
- ° Volumen del Desperdicio Mínimo Permitido
- ° Mejora al Desperdicio Permitido para la próxima solución
- ° Porcentaje Límite del Peso Específico que Soporta una Caja
- ° Porcentaje Límite de los Niveles Apilables que Soporta una Caja
- ° Heurística de Selección de Caja utilizada
- ° Heurística de Selección de Forma utilizada
- ° Cantidad de Plataformas Generadas
- ° Cantidad de Cajas Generadas
- ° Cantidad de Formas Generadas
- ° Cantidad de Bloques Generados
- ° Hora de Inicio de la ejecución del Algoritmo
- ° Cantidad de Tipos de Cajas que se solicitaron
- ° Código de la Caja utilizada como Relleno del Volumen del Pedido cuando no complete el Volumen del Camión
- ° Largo Máximo de la Plataforma
- ° Volumen del Pedido
- ° Peso del Pedido
- ° Tiempo Máximo de Ejecución del Algoritmo

♦ BLOQUE (*BL*)

- ° N° Bloque
- ° Plataforma Vacía (*PV*)
- ° Lista de Cajas (*LCA*)
- ° Caja (*CA*)
- ° Lista de Formas (*LFO*)
- ° Forma (*FO*)
- ° Camión (*CA*)
- ° Datos Globales (*GL*)

♦ LISTA DE BLOQUES (*LBL*)

Elementos de la estructura Bloque (*BL*)

VII. ESPECIFICACIONES GENERALES

ESPECIFICACIONES GENERALES

VII.1. Hardware

Los equipos que se utilizaron fueron PC y estos se detallan a continuación:

1. **PC 486 DX2:** Procesador: 66 Mhz (con coprocesador matemático)
Memoria RAM: 8 MB
Disco Rígido: 120 MB
2. **PC 486 SL:** Procesador: 25 Mhz (con coprocesador matemático)
(Notebook) Memoria RAM: 8 MB
Disco Rígido: 250 MB

Para el desarrollo del Algoritmo de Carga de Cajas dentro de un Camión y para la ejecución de los utilitarios que se detallan en el punto "Software" se utilizaron ambas PCs, y las soluciones del Algoritmo se obtuvieron con PC identificada con el número 1.

VII.2. Software

El Sistema Operativo utilizado en las PCs fue MS-DOS Versión 6.2 TM.

El Software con el que se desarrolló el Algoritmo y el Sistema de Gestión de Carga de Cajas dentro de un Camión fue CA-Clipper Versión 5.2d TM de Computer Associates International, Inc. (1993) y los módulos fueron linkeditados con Blinker Versión 3.0 TM de Assembler Software Manufacturers, Inc. (1993).

Para graficar la Carga de las Cajas dentro del Camión se utilizó el Software Mathematica Versión 2.0 for MS-DOS 386/7 TM de Wolfram Research, Inc. (1991). Se capturaron las pantallas de la Carga realizada con el Mathematica, para luego insertarlas en el Procesador de Texto, con el PCS - Professional Capture Systems DOSCapture Versión 1.02 TM de JASC, Inc. (1993).

Para realizar toda la documentación, se utilizó MS Windows 3.1 TM, MS Word 6.0 for Windows TM y el VISIO 2.0 for Windows de Shapeware Corporation (1993) en el cual se realizaron todos los gráficos adicionales.

VII.3. Criterios para Seleccionar las Herramientas

Se realizó el desarrollo del Algoritmo y del Sistema de Gestión de Carga de Cajas dentro de un Camión en Clipper 5.2d debido a que los autores estaban familiarizados con este lenguaje y por lo tanto el tiempo de implementación fue más corto que si se hubiese elegido otro lenguaje. Por otro lado, Clipper tiene un manejo de vectores muy poderoso y se utilizó esta propiedad para implementar las estructuras de datos en vectores, todas ellas en memoria, con lo cual se incrementa la velocidad de procesamiento debido a que se utiliza un menor acceso a unidades de disco.

VII.4. Dificultades que se Encontraron Durante el Desarrollo

El Algoritmo fue implementado en primer término en el lenguaje Clipper 5.01 y con el linkeditor Blinker 2.0. Cuando se ejecutaba el algoritmo se producía un error no documentado, totalmente aleatorio, producido por el Manejador de Memoria Virtual ("Error 5333"). Esto ocurría en algunos casos de prueba y cuando se trataba de arreglarlo, como por ejemplo asignando más memoria EMS (Expanded Memory System) o XMS (Extended Memory System.), se arreglaban los casos que fallaban pero luego dejaban de funcionar otros.

Se buscaron otras alternativas, como ser utilizar la versión de Clipper 5.2 y el Blinker 3.0 (este último permite trabajar en modo protegido), pero esto no solucionó el problema. El error que se generaba también estaba relacionado con el Manejador de Memoria Virtual y el mensaje del error producido era "VM Integrity Failure".

Otro intento de solucionar dicho error fue utilizar la librería para Clipper 5.2 llamada SIX, la cual provee un Driver para manejo de bases de datos reemplazando el Driver nativo de Clipper. De esta forma podíamos trabajar con arrays en campos memos de un archivo, con lo cual pensábamos evitar una posible falta de memoria RAM. Este intento no dió el resultado esperado.

También se utilizó el linkeditor EXOSPACE. Este linkeditor reemplaza el Manejador de Memoria Virtual del Clipper 5.2 por uno propio. Sin embargo, esta alternativa tampoco solucionó el problema.

El error fue solucionado completamente con un patch de Clipper que actualiza la versión 5.2 en 5.2d.

VIII. RESULTADOS

RESULTADOS

VIII.1. Datos con los que se Ejecutó el Algoritmo FHI

- **Tiempo Máximo de Ejecución:** 10 minutos
- **Largo Máximo de la Plataforma:** 2 metros
- **Desperdicio Permitido Inicial:** 0.15%
- **Límite Peso Específico que Soporta una Caja:** 0.20%
- **Límite Máximo de Niveles Apilables de las Cajas:** 0.20%
- **Mejora del Aprovechamiento entre las Soluciones:** 0.10%

- **CAMION N° 1114**

- **Largo:** 1200 cm
- **Ancho:** 235 cm
- **Alto:** 250 cm
- **Peso Neto que Soporta:** 21000 kg
- **Volumen del Camión:** 70500000 cm³
- **Sector Pesado:**
 - **Inicio:** 0 cm
 - **Fin:** 600 cm
 - **Peso Soporta :** 16000 kg
 - **Diferencia Máxima de Peso entre Laterales:** 1600 kg
- **Sector Liviano:**
 - **Inicio:** 600 cm
 - **Fin:** 1200 cm
 - **Peso Soporta:** 8000 kg
 - **Diferencia Máxima de Peso entre Laterales:** 800 kg

- **CAMION N° MORA (PEDIDO N° 111)**

- **Largo:** 579.3 cm
- **Ancho:** 223.6 cm
- **Alto:** 226.1 cm
- **Peso Neto que Soporta:** 90000 kg
- **Volumen del Camión:** 29287067 cm³

- **Sector Pesado:**
 - **Inicio:** 0 cm
 - **Fin:** 289 cm
 - **Peso Soporta :** 90000 kg
 - **Diferencia Máxima de Peso entre Laterales:** 9000 kg

- **Sector Liviano:**
 - **Inicio:** 289 cm
 - **Fin:** 579.3 cm
 - **Peso Soporta:** 90000 kg
 - **Diferencia Máxima de Peso entre Laterales:** 9000 kg

- **HEURISTICAS**

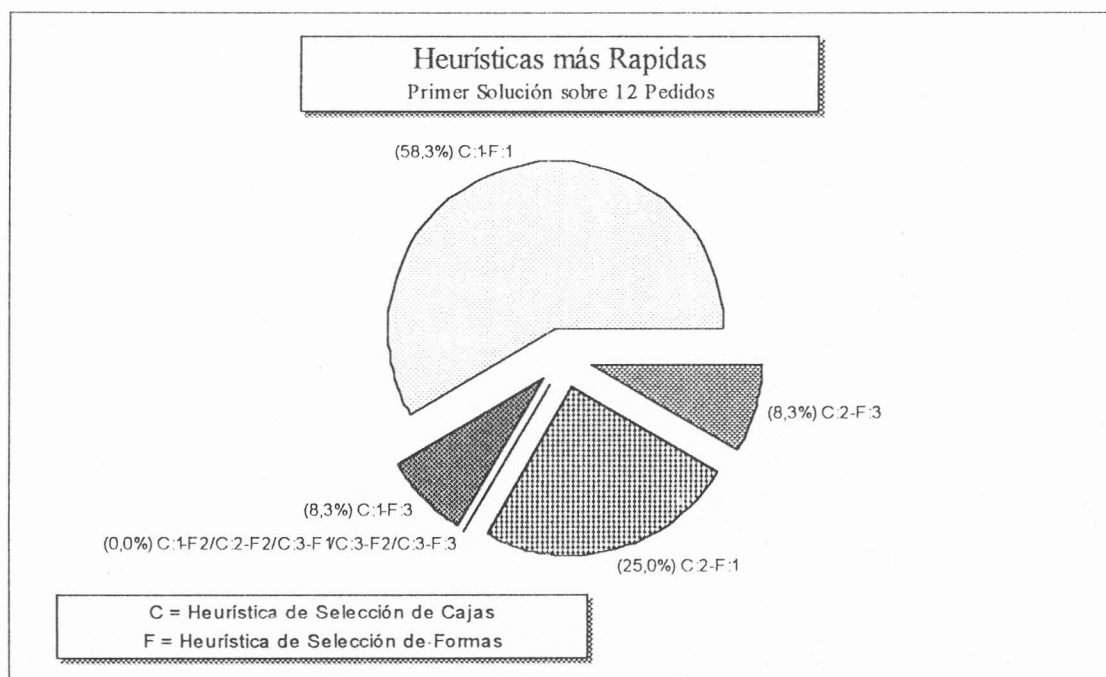
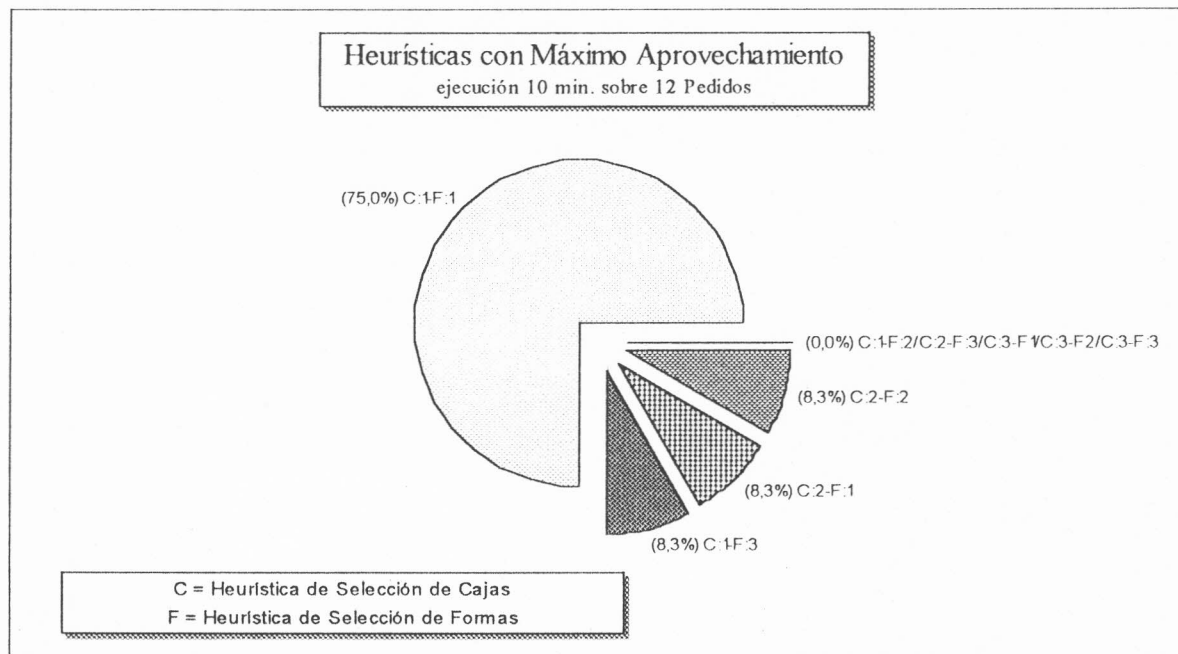
C = de Selección de Cajas

1. Seleccionar la caja de mayor Peso Específico.
2. Peso Específico y Volumen Pedido:
Si la PV está en el piso del camión
seleccionar la caja de mayor peso específico.
Sino
seleccionar la caja de mayor volumen pedido.
Fin Si
3. Seleccionar la caja de mayor Volumen Pedido

F = de Selección de Formas

1. Volumen de la Caja Seleccionada por la cantidad de cajas de las Formas Extraídas (Volumen de la Formas Extraídas) y si hay distintas Formas que tienen el mismo Volumen se ordenan primero por las mas anchas y luego por las largas de las Formas.
2. Superficie de la Base y si hay bases iguales se ordenan en forma descendente de altura.
3. Ancho, largo y alto de las Formas Extraídas en orden descendente.

VIII.2. Análisis Comparativo de las Soluciones según las Distintas Heurísticas entre Pedidos



De estos gráficos se puede observar que sobre *12 Pedidos* en donde se ejecutó el algoritmo de carga de cajas en un camión, la heurística **C:1-F:1** (Heurística de Selección de Cajas: lista de Cajas Pedidas ordenadas por Peso Especifico de la Caja en orden descendente - Heurística de Selección de Formas: de la lista de Formas Extraídas la que ocupe más volumen del espacio libre de la Plataforma Seleccionada), en un 75% de las veces obtuvo la mejor solución (máximo aprovechamiento) en un lapso de 10 minutos de ejecución y en un 58% obtuvo la primera solución más rápido que las otras heurísticas.

VIII.3. Análisis Comparativo de la Performance de las Heurísticas entre Pedidos para Obtener Soluciones con el Máximo Aprovechamiento

HEURISTICAS											
	# T.Cajas	C:1-F:1	C:1-F:2	C:1-F:3	C:2-F:1	C:2-F:2	C:2-F:3	C:3-F:1	C:3-F:2	C:3-F:3	
		%	%	%	%	%	%	%	%	%	
PEDIDOS	476	16	91,95	89,47	87,72	90,50	89,54	87,11	85,84	89,21	87,71
	563	19	90,62	---	88,15	85,14	87,06	86,75	87,29	---	87,07
	564	44	87,20	---	---	85,11	---	---	---	---	---
	810	47	88,91	87,65	87,05	---	---	---	---	---	---
	811	45	85,45	---	---	---	---	---	---	---	---
	812	43	89,60	---	---	87,44	86,39	85,22	88,86	---	85,11
	813	57	86,76	85,04	85,44	---	---	---	---	---	---
	814	48	87,22	86,81	85,15	---	---	---	---	---	---
	815	48	85,33	86,65	---	85,05	---	---	---	---	---
	816	7	90,57	92,45	91,16	91,18	93,66	87,99	89,56	92,33	90,77
	817	41	88,47	---	---	85,22	---	---	87,02	---	---
	818	32	86,56	85,36	87,33	86,70	---	85,10	85,15	85,24	---

C = Heurística de Selección de Cajas
F = Heurística de Selección de Formas
T.Cajas = Cantidad de Tipos de Cajas en el pedido

De los 12 casos en los que se ejecutó el algoritmo de carga dentro de camiones con un tiempo máximo de 10 minutos de ejecución, la única heurística que llegó siempre a obtener por lo menos una solución fue la **C:1-F:1** que describimos con anterioridad y además obtiene, en la mayoría de los casos, la mejor solución entre las distintas heurísticas para un mismo pedido.

Cabe destacar que la heurística C:2-F:2 (Heurística de Selección de Caja: ordenar por Peso Específico cuando se encuentra en el piso y por Volumen de las Cajas del Pedido en otro caso- Heurística de Selección de Formas: Superficie de la Forma con Base mayor) fue la que tuvo el mejor aprovechamiento entre todos los pedidos (pedido 816: 93,66%), pero en dicha oportunidad la cantidad de tipos de cajas a acomodar era muy pequeña, en comparación con los demás.

VIII.4. Análisis comparativo del Índice de Mezcla de las Cajas según las Heurísticas para la Mejor Solución con 10 minutos de ejecución

Pedido: 563
Tiempo máx. ejecución: 10 minutos

HEURÍSTICAS															
	C-1-F-1		C-1-F-3		C-2-F-1		C-2-F-2		C-2-F-3		C-3-F-2		C-3-F-3		
	#	cm	#	cm	#	cm	#	cm	#	cm	#	cm	#	cm	
CAJAS	018	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	023	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	2	43.16	1	0.00
	025	2	136.72	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	026	1	0.00	1	0.00	1	0.00	0	0.00	0	0.00	1	0.00	1	0.00
	029	2	62.50	2	86.72	1	0.00	1	0.00	1	0.00	2	164.92	2	733.74
	062	0	0.00	0	0.00	0	0.00	1	0.00	1	0.00	0	0.00	0	0.00
	075	2	249.50	4	392.49	2	760.96	1	0.00	3	633.56	3	516.16	3	632.44
	104	1	0.00	2	23.53	2	78.60	2	388.11	3	339.50	3	198.47	3	238.08
	106	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	110	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	160	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00
	161	1	0.00	1	0.00	2	77.94	1	0.00	1	0.00	0	0.00	1	0.00
	166	3	518.15	4	461.32	4	798.54	3	520.29	4	166.70	1	0.00	1	0.00
	197	0	0.00	1	0.00	1	0.00	1	0.00	2	492.12	2	181.02	3	199.47
	704	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	2	853.79	2	374.22
	706	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	708	2	377.05	1	0.00	1	0.00	1	0.00	0	0.00	2	102.42	1	0.00
	712	1	0.00	1	0.00	1	0.00	1	0.00	2	31.40	2	369.17	2	560.77
	718	1	0.00	1	0.00	1	0.00	1	0.00	2	268.25	1	0.00	2	194.22

C = Heurística de Selección de Cajas
F = Heurística de Selección de Formas
= Cantidad de Bloques en donde se encuentra un mismo tipo de caja
cm = Distancia total entre los Bloques en donde se encuentran un mismo tipo de caja

Pedido: 812
 Tiempo máx. ejecución: 10 minutos

HEURISTICAS													
	C-1-F-1		C-2-F-1		C-2-F-2		C-2-F-3		C-3-F-1		C-3-F-3		
	#	cm	#	cm	#	cm	#	cm	#	cm	#	cm	
CAJAS	007	1	0.00	1	0.00	0	0.00	1	0.00	0	0.00	1	0.00
	015	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	0	0.00
	017	2	186.20	1	0.00	2	47.70	1	0.00	1	0.00	1	0.00
	020	1	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
	030	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	031	1	0.00	1	0.00	0	0.00	0	0.00	0	0.00	1	0.00
	032	2	33.38	1	0.00	0	0.00	0	0.00	0	0.00	0	0.00
	036	2	703.55	2	198.72	1	0.00	2	271.12	1	0.00	1	0.00
	037	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00
	062	1	0.00	0	0.00	1	0.00	1	0.00	1	0.00	2	127.05
	068	1	0.00	1	0.00	1	0.00	3	291.30	3	37.90	0	0.00
	076	1	0.00	3	57.58	1	0.00	3	36.05	1	0.00	2	22.45
	083	2	110.68	1	0.00	1	0.00	1	0.00	0	0.00	0	0.00
	090	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	0	0.00
	093	1	0.00	1	0.00	2	269.70	1	0.00	4	159.05	2	32.43
	094	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	2	42.16
	095	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	0	0.00
	104	2	42.29	4	131.22	3	208.77	4	318.45	3	389.93	2	391.38
	105	2	296.28	3	627.01	1	0.00	3	134.40	2	119.25	2	26.58
	106	2	469.06	0	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	108	1	0.00	1	0.00	2	138.00	2	83.49	1	0.00	0	0.00
	113	1	0.00	6	627.24	4	277.51	1	0.00	3	32.81	4	456.79
	116	3	56.57	2	89.94	2	128.12	1	0.00	3	389.45	1	0.00
	126	1	0.00	3	127.97	1	0.00	3	161.12	2	319.51	1	0.00
	132	0	0.00	5	564.06	4	584.86	1	0.00	3	320.64	2	255.64
	169	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	160	1	0.00	1	0.00	1	0.00	2	54.06	1	0.00	0	0.00
	162	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00
	166	2	21.83	1	0.00	2	344.18	1	0.00	2	140.75	2	186.06
	169	1	0.00	4	65.82	1	0.00	3	70.24	1	0.00	1	0.00
	179	2	23.40	0	0.00	0	0.00	0	0.00	1	0.00	1	0.00
	180	2	21.83	2	122.13	5	695.35	4	488.18	2	15.63	2	517.04
	185	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00
	192	1	0.00	1	0.00	0	0.00	0	0.00	0	0.00	0	0.00
	193	1	0.00	1	0.00	1	0.00	2	55.61	1	0.00	2	182.88
	194	2	35.98	0	0.00	2	256.58	0	0.00	1	0.00	2	148.25
	195	1	0.00	4	504.09	4	124.56	5	406.63	1	0.00	1	0.00
	702	1	0.00	1	0.00	1	0.00	3	685.44	1	0.00	2	30.48
	705	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
	724	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	0	0.00
755	1	0.00	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	
758	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	0	0.00	
758	2	77.49	1	0.00	1	0.00	1	0.00	1	0.00	0	0.00	

Como se observa, el *Indice de Mezcla* de las Cajas Asignadas es **bajo** para todas las heurísticas, ya que se acomodaron las cajas de un mismo tipo en bloques bastantes cercanos entre si o en uno solo. Como todas las heurísticas tienen distinto aprovechamiento, no es posible sacar una conclusión de cual es la mejor de todas.

Lo que se trata de evitar con un bajo Indice de Mezcla es que un tipo de caja no se disperse en el camión, para que en el momento de la descarga sea sencillo controlar la cantidad de cajas recibidas. En particular el camión en donde se acomodaron las cajas de los dos pedidos anteriormente detallados tiene una longitud de 12 metros.

VIII.5. Datos que permiten Monitorear el Algoritmo para la Mejor Solución de un Pedido

Pedido:	563
Tiempo máx. ejecución:	10 minutos
# Tipos de Cajas:	19
# Cajas Pedidas:	3894
Volumen Camión:	70500000 cm ³
Volumen Pedido:	70485508 cm ³
Peso Pedido:	20389 kgr.

Heurísticas	Nodos Generados	Tiempo (seg.)	%Apro-vechado	Volumen Acumulado (cm ³)	Peso Acumulado	Cajas Acomodadas	Tipos Cajas Sin Acomodar
C-1-F-1	13691	154.18	90.62	63889961	18853	3640	6
C-1-F-3	13966	337.96	88.15	62143839	18348	3538	5
C-2-F-1	5329	37.57	85.14	60025412	18272	3527	5
C-2-F-2	3445	246.90	87.06	61379306	17518	3359	7
C-2-F-3	11005	326.47	86.75	61155669	17696	3418	7
C-3-F-1	24008	282.70	87.29	61541067	17645	3459	7
C-3-F-3	17894	476.48	87.07	61385049	17487	3450	12

C = Heurística de Selección de Cajas
F = Heurística de Selección de Formas

Pedido:	812
Tiempo máx. ejecución:	10 minutos
# Tipos de Cajas:	43
# Cajas Pedidas:	3208
Volumen Camión:	70500000 cm ³
Volumen Pedido:	70479689 cm ³
Peso Pedido:	18113 kgr.

Heurísticas	Nodos Generados	Tiempo (seg.)	%Apro-vechado	Volumen Acumulado (cm ³)	Peso Acumulado	Cajas Acomodadas	Tipos Cajas Sin Acomodar
C-1-F-1	41728	521.40	89.60	63169855	16593	2951	4
C-2-F-1	8930	81.90	87.44	61648527	16205	2878	11
C-2-F-2	29060	586.88	86.39	60868066	15805	2811	14
C-2-F-3	17753	256.55	85.22	60081500	15644	2763	14
C-3-F-1	10057	53.82	88.86	62643498	15346	2645	22
C-3-F-3	24760	309.17	85.11	60000713	14239	2539	22

C = Heurística de Selección de Cajas
F = Heurística de Selección de Formas

VIII.6. Comparación entre los Algoritmos FHI y el de Morabito y Arenales con el ejemplo de George y Robinson (1980) para el caso de la Mejor Solución

Pedido:	111
Tiempo máx. ejecución:	10 minutos
# Tipos de Cajas:	8
# Cajas Pedidas:	784
Volumen Pedido:	26324653 cm ³
Volumen Camión:	29287067 cm ³

Algoritmos	Nodos Gene- rados	Tiempo (seg.)	%Apro- vechado	Volumen Acumula- do (cm ³)	Cajas Acomo- -dadas	Tipos Cajas Sin Acomodar
Morabito- Arenales	3679397	3529	89.89*	26324653	784	0
FHI (C:1-F:1)	1537	13.34	88.40	25888468	776	1

(*) Solución Óptima

La ejecución del Algoritmo de Morabito y Arenales llega a una solución óptima debido a que considera las 6 orientaciones posibles de acomodar una caja y por lo tanto el árbol de búsqueda es mayor que el del Algoritmo FHI que solo considera 2 orientaciones. Por otro lado, el Algoritmo de Morabito y Arenales no contempla restricciones de peso, de tipo de caja, el índice de mezcla ni la distribución de peso en los sectores del camión, que si verifica el Algoritmo FHI.

IX. TABLA DE RESTRICCIONES DE TIPOS DE CAJAS

TABLA DE RESTRICCIONES DE TIPO DE LAS CAJAS

		A R R I B A															
		GP	GO	KR	GL	OB	PA	GA	GF	CO	CE	JP	FS	FC	FL	BT	BL
A	GP	S	S	S	S	S	S	S	S	N	N	S	N	N	N	N	N
	GO	N	S	N	N	S	N	S	S	N	N	S	N	N	N	N	N
	KR	N	S	S	S	S	N	S	S	N	N	S	N	N	N	N	N
	GL	N	S	S	S	S	N	S	S	N	N	S	N	N	N	N	N
	OB	N	S	N	N	S	N	N	S	N	N	S	N	N	N	N	N
B	PA	N	S	S	S	S	S	S	S	N	N	S	N	N	N	N	N
A	GA	N	S	S	S	S	N	S	S	S	N	S	N	N	N	N	N
	GF	N	N	N	N	N	N	N	S	N	N	N	N	N	N	N	N
I	CO	N	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
	CE	N	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S
O	JP	N	S	N	N	S	N	N	S	N	N	S	N	N	N	N	N
	FS	N	S	S	S	S	S	S	S	N	N	S	S	N	N	N	N
	FC	N	S	S	S	S	S	S	S	N	S	S	S	S	N	N	N
	FL	N	S	S	S	S	S	S	S	N	S	S	S	S	S	N	N
	BT	N	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N
	BL	N	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

La información que contiene la tabla se interpreta de la siguiente forma:

1. Si entro a la tabla por la fila GO y recorro las columnas de izquierda a derecha obtendré la siguiente información:

- La caja que pertenece al tipo GO no puede ser colocada debajo de una caja que pertenece al tipo GP.
- La caja que pertenece al tipo GO puede ser colocada debajo de una caja que pertenece al mismo tipo.
- La caja que pertenece al tipo GO no puede ser colocada debajo de una caja que pertenece al tipo KR.

2. Si entro a la tabla por la columna GO y recorro las filas de arriba hacia abajo obtendré la siguiente información:

- La caja que pertenece al tipo GO puede ser colocada sobre una caja que pertenece al tipo GP.

- La caja que pertenece al tipo GO puede ser colocada sobre una caja que pertenece al mismo tipo.
- La caja que pertenece al tipo GO puede ser colocada sobre una caja que pertenece al tipo KR.
- La caja que pertenece al tipo GO no puede ser colocada sobre una caja que pertenece al tipo GF.

X. CONCLUSIONES

CONCLUSIONES

Se desarrolló un **método de búsqueda** que se implementó en el algoritmo **FHI** para el problema de la “programación de la carga de un camión con cajas de distintos tamaños”. Este problema pertenece a la categoría general de bin packing y cutting stock, y tiene la característica que es NP-Hard. Por lo tanto no hay, hasta la actualidad, un algoritmo que permita solucionarlo en tiempo polinomial, a menos que se demuestre que $P=NP$.

El método de búsqueda propuesto recorre un árbol AND/OR utilizando heurísticas que permiten podar el espacio del problema de forma tal que se pueden obtener soluciones aceptables desde el punto de vista de los objetivos planteados (el objetivo primario de maximizar el aprovechamiento bajo ciertas restricciones y los objetivos secundarios, como ser el “índice de mezcla” y la “facilidad de operatoria”) y del tiempo de respuesta. Los tests realizados sobre 13 casos reales, uno de los cuales fue tomado del trabajo de Morabito y Arenales [11], son los que nos permiten llegar a dicha conclusión.

Algunas de las heurísticas utilizadas para lograr los objetivos son:

La utilización de un porcentaje **p%**, denominado “bondad de la solución”, permite balancear el compromiso de **tiempo** con el de **optimalidad de la solución**.

Al implementar varias heurísticas para seleccionar las cajas y varias para seleccionar las formas, se pudo **estudiar comparativamente** la calidad de las mismas al ser aplicadas a los casos de prueba antes mencionados. De las nueve combinaciones que se estudiaron, hay una que dio los mejores resultados. Sin embargo esto no significa que sea la mejor combinación de heurísticas para aplicar siempre. Solamente nos permite afirmar que dentro del marco de la solución propuesta y para los casos de prueba seleccionados, dicha combinación de heurísticas dio muy buenos resultados.

Los **patrones regulares** considerados en la generación de formas, permitieron reducir considerablemente el espacio de soluciones del problema, lo cual posibilitó junto con otras heurísticas, arribar a soluciones en tiempos aceptables.

La heurística **largo de plataforma acotado** contribuyó a generar “buenas formas”, evitando armar bloques que se extendieran a lo largo del eje x (largo del camión).

De una importante empresa del rubro alimenticio, donde la programación de la carga se realiza en forma manual, por una persona con experiencia, se obtuvieron los siguientes datos sobre 25 pedidos:

• Aprovechamiento mayor al 90%	47%
• Aprovechamiento entre el 85 y el 90%	48%
• Aprovechamiento menor al 85%	5%

Por otra parte, los resultados obtenidos en este trabajo, basados en 12 pedidos (de la misma empresa, pero diferentes), con la heurística C:1-F:1, con un tiempo máximo de ejecución de 10 minutos, son:

• Aprovechamiento mayor al 90%	25%
• Aprovechamiento entre el 85 y el 90%	75%
• Aprovechamiento menor al 85%	0%

En primer lugar debemos notar que esta comparación solamente se refiere al objetivo primario del problema, esto es, “maximizar el aprovechamiento del camión”.

Con respecto a los objetivos secundarios, el “índice de mezcla” y “facilidad de operatoria”, dicha empresa no guarda información de la forma en que cargó los camiones con los pedidos.

En segundo lugar hay que tener presente que la persona que programa la carga no puede saber si se cumplen o no las restricciones, la carga se realiza en forma “aproximada”.

La programación de la carga por parte de la empresa solo consta de un ordenamiento de los tipos de caja del pedido, quedando en manos del operario la decisión de cómo acomodar las cajas dentro del camión. El tiempo promedio que lleva el ordenamiento del pedido es de 15 minutos.

Esta comparación nos permite pensar que los resultados obtenidos con el algoritmo FHI son aceptables, dado que el mismo contempla todas las restricciones del problema y la empresa no.

X.1. Ventajas del método implementado

- ✓ Disminución de costos, al posibilitar aprovechar la capacidad de carga disponible.
- ✓ Conservar la calidad de los productos:
 - ✓ Se logra al satisfacer las restricciones de peso y tipo de los productos.
 - ✓ Esto implica al mismo tiempo disminución de costos.

- ✓ Evitar problemas legales, al satisfacer las restricciones impuestas por las normas vigentes del transporte de carga.
- ✓ Disminución de tiempos en el ciclo de venta en los siguientes puntos:
 - ✓ En la programación de la carga.
 - ✓ En la carga del camión.
 - ✓ En la descarga.
 - ✓ Facilidad de realizar el inventario en el destino.
- ✓ No se requieren expertos para programar la carga.
- ✓ El conocimiento del experto puede ser aprovechado para enriquecer el método implementado.
- ✓ Es fácil de usar, solamente se ingresan los datos del pedido y del camión; luego se ejecuta el algoritmo con los parámetros deseados y, finalmente, se imprime el listado de carga programada y se grafica la solución obtenida.
- ✓ Es modular.
- ✓ Es fácil de incorporar nuevas heurísticas.
- ✓ Brinda opciones gráficas que permitieron realizar un chequeo adicional de la correctitud de las soluciones obtenidas y cómo se comportaban las heurísticas para reducir el índice de mezcla y asegurar la estabilidad de la carga.
- ✓ Representa muy bien los resultados:
 - ✓ Listados.
 - ✓ Gráficos.
- ✓ Al considerar únicamente dos orientaciones de cajas, y al entregar un listado detallado de la carga programada, se evita que el operario cometa errores.
- ✓ Al tener parametrizados el tiempo máximo de ejecución y el desperdicio máximo permitido, se obtiene un balance entre el tiempo de ejecución y el aprovechamiento.

X.2. Problemas que nuestro algoritmo no puede hoy resolver

- ✓ Programar la carga de camiones con acoplado o en varios camiones.
- ✓ Eventos que cambian las características del problema original:
 - ✓ Si por ejemplo el camión está chocado, el volumen a completar presenta irregularidades que nuestro algoritmo no contempla.
 - ✓ Si las cajas están húmedas, se produce una modificación no prevista de las dimensiones y que nuestro algoritmo tampoco puede resolver.

XI. BIBLIOGRAFIA

BIBLIOGRAFIA

• Trabajos Consultados

1. Beasley, J.E. - An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure - Operational Research 33, pp. 49-64, 1985.
2. Coffman, E.G. Jr. and Lagarias, J.C. - Algorithms for Packing Squares: A Probabilistic Analysis - SIAM J. COMPUT, Vol. 18, No. 1, pp. 166-185, February 1989.
3. De Cani, Phillip - A Note on the Two-Dimensional Rectangular Cutting-Stock Problem - Journal of the Operational Research Society, Vol. 29, No. 7, pp. 703-706, 1978.
4. Dirección Nacional de Vialidad - Nuevas Normas para el Transporte Nacional e Internacional de Carga entre los Países del Mercosur - 1992.
5. Dowsland, Kathryn A. and Dowsland, William - A Comparative Analysis of Heuristics for the Two-Dimensional Packing Problem - Paper for Euro VI Conference, July 1983.
6. Garey, Michael R.; Graham, R.L.; Johnson, David S. and Yao, Andrew Chi-Chih - Resource Constrained Scheduling as Generalized Bin Packing - Journal of Combinatorial Theory (A), 21, pp. 257-298, 1976.
7. Garey, Michael R. and Johnson, David S. - Computers and Intractability - A Guide to the Theory of NP-Completeness - W.H. Freeman, San Francisco, 1979.
8. Gilmore, P.C. and Gomory, R.E. - Multistage Cutting Stock Problems of Two and More Dimensions - OPNS. RES. No 13, pp. 94-120, 1964.
9. Kvitka, Adolfo M. - Resolución de Problemas con Inteligencia Artificial. - Kapelusz. Serie EBAI, 1988.
10. Li, Keqin and Cheng, Kam-Hoi - On Three-Dimensional Packing - SIAM J. COMPUT. Vol 19, No 5, pp. 847-867, October 1990.
11. Morabito, Reinaldo and Arenales, Marcos - An AND/OR-graph Approach to the Container Loading Problem - Int. Trans. Opl. Res. Vol. 1, No. 1, pp. 59-73, 1994.

12. Murgolo, Frank D. - An Efficient Approximation Scheme for Variable-Sized Bin Packing - SIAM J. COMPUT, Vol. 16, No. 1, pp. 149-161, February 1987.
13. Nilsson, N. - Principles of Artificial Intelligence - Tioga Press, 1980.
14. Smith, Adrian and De Cani, Phillip - An Algorithm to Optimize the Layout of Boxes in Pallets - J. OPL. RES. SOC. Vol 31, No 7, pp. 573-578, 1980.
15. Wolfram, Stephen - Mathematica, A System for Doing Mathematics by Computer, Second Edition - Addison Wesley Publishing Company, 1993.
16. Wood, Derick - Theory of Computation - John Wiley & Sons, Inc., 1987.
17. Yao, Andrew Chi-Chih - New algorithms for Bin Packing - Journal of the Association for Computing Machinery, Vol. 27, No 2, pp. 207-227, April 1980.