PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Aplicación del Algoritmo de Búsqueda Tabú para la optimización del espacio utilizado en el llenado de contenedores

Tesis para obtener el título profesional de Ingeniero Informático

Autor:

Paul Andrés León Málaga

Asesor:

Mg. Rony Cueva Moscoso

Lima, Julio de 2021

RESUMEN

Debido a la progresiva globalización del uso de los contenedores en el transporte marítimo internacional y a la creación de una extensa variedad de estos, se tuvo la necesidad de controlar y estandarizar sus características como el tamaño, carga máxima y métodos de identificación, con este fin se crearon los estándares ISO 668 e ISO 6346.

A pesar de los beneficios del uso de los contenedores, las empresas comercializadoras han tenido problemas en optimizar el espacio utilizado dentro de estos debido principalmente a que el orden y distribución de los paquetes que son ingresados en los contenedores es inadecuado. Por lo tanto, plantear un orden de colocación óptimo de los paquetes dentro del contenedor se vuelve el principal problema para este proyecto de fin de carrera. Su solución permitirá transportar más paquetes reduciendo el espacio perdido en cada contenedor, lo que finalmente conlleva a una reducción de los costos de transporte por contenedor incurridos por la empresa.

Para este proyecto de fin de carrera se propone aplicar el algoritmo de Búsqueda Tabú para optimizar el espacio utilizado en contenedores considerando restricciones de peso y fragilidad y compararlo con el algoritmo Genético. Como objetivos específicos se tienen los siguientes puntos:

- Definir la función objetivo para ambos algoritmos.
- Adaptar un algoritmo Genético que proporcione una posible solución.
- Diseñar un algoritmo de Búsqueda Tabú que brinde una alternativa de solución.
- Implementar prototipos funcionales para ambos algoritmos.
- Realizar la experimentación numérica que permita determinar cuál es la solución más óptima.

DEDICATORIA

Este documento es la evidencia de la conclusión de varios años de esfuerzo y dedicación que no hubiera podido conseguir si no fuera por algunas personas que me apoyaron y acompañaron en toda o en gran parte de mi carrera. Por esto quiero empezar por agradecer a mi mamá quien dio todo lo que pudo y más para que yo pueda estudiar en esta universidad sin que me falte nada. También quiero agradecer a mis abuelos quienes no dudaron en ayudarme para poder finalizar los últimos ciclos.

No puedo dejar de mencionar a todos los amigos y colegas que fueron parte de este viaje, sobre todo a Pablo, Cesar y Sebastián, gracias por las risas y momentos que compartimos y, seguramente, seguiremos compartiendo.

Por último, quiero nombrar a los dos pequeños seres que me acompañaron todos los días, Yako y Camila.

Muchas gracias a todos.

Tabla de Contenido

Tabla de Contenido 1
Índice de Figuras 3
Índice de Tablas 3
Capítulo 1. Generalidades 5
1.1 Problemática 5
1.2 Objetivos 8
1.2.1 Objetivo general 8
1.2.2 Objetivos específicos 8
1.2.3 Resultados esperados 9
1.2.4 Mapeo de objetivos, resultados y verificación 9
1.3 Herramientas y Métodos 12
1.3.1 R Studio 13
1.3.2 Microsoft Visual Studio 13
1.3.3 Método del Algoritmo Genético 13
1.3.4 Método del Algoritmo de Búsqueda Tabú 14
1.3.5 Lenguaje de programación C# 14
1.3.6 SCRUM 15
1.3.7 Prueba Z 15
1.3.8 Prueba F de Fisher 15
1.3.9 Prueba Kolmogorov-Smirnov 15
1.4 Alcance, limitaciones y riesgos 16
1.4.1 Alcance 16
1.4.2 Limitaciones 16
1.4.3 Riesgos 17
1.5 Justificación y Viabilidad del proyecto 17
1.5.1 Viabilidad Técnica 17
1.5.2 Viabilidad Temporal 18
1.5.3 Viabilidad Económica 19

1.5.4 Justificación del proyecto 19

Capítulo 2. Marco Legal/Regulatorio/Conceptual/otros 20

2.1 Contenedor 20

2.2 Paleta 20

2.3 Unidad Equivalente a 20 Pies (TEU por sus siglas en inglés) 21

2.4 Puerto 21

2.5 Transporte marítimo internacional 21

2.6 Problema de Bin Packing 22

2.7 Complejidad P 22

2.8 Complejidad NP 22

2.9 Complejidad NP-Difícil 22

2.10 Optimización Combinatoria 23

2.11 Algoritmo Heurístico 23

2.12 Algoritmo Metaheurístico 23

2.13 Algoritmo Genético 24

2.14 Algoritmo Búsqueda Tabú 25

Capítulo 3. Estado del Arte 27

3.1 Enfoques para la resolución del problema 27

3.1.1 Optimización del Llenado de Contenedores para Transporte Multimodal 27

3.1.2 Hybrid greedy heuristics based on linear programming for the three-dimensional single bin-size bin packing problem 27

3.1.3 Diseño de un algoritmo de búsqueda tabú para la minimización del desperdicio de espacio en almacenes de empresas comercializadoras de tuberías 28

3.1.4 A hybrid genetic algorithm for 3D bin packing problems 28

3.2 Software comercial para la resolución del problema 29

3.2.1 3D Bin Packing 29

3.2.2 Cube-IQ Load Planning 30

3.3 Conclusiones 31

Capítulo 4. Función Objetivo 32

Capítulo 5. Estructura de datos para los algoritmos 35
Capítulo 6. Pseudocódigo del algoritmo genético 37
Capítulo 7. Pseudocódigo del algoritmo de búsqueda tabú 46
Capítulo 8. Programa e interfaz gráfica 51
Capítulo 9. Experimentación numérica 54
Capítulo 10. Conclusiones y trabajos futuros 62
10.1 Conclusiones 62
10.2 Trabajos futuros 63
Capítulo 11. Referencias 64

Índice de Figuras

Ilustración 1: Puntos Extremos de un objeto (Crainic et al., 2007) 14
Ilustración 2. Pseudocódigo Algoritmo de Búsqueda Tabú 26
Ilustración 3: Sofware 3D Bin Packing 29
Ilustración 4: Software Cube-IQ Load Planning 30
Ilustración 5: Rotaciones de una caja 33
Ilustración 6. Pantalla de ingreso de parámetros 51
Ilustración 7. Pantalla de resultados 52
Ilustración 8. Interfaz de resultados de los algoritmos 57
Ilustración 9. Resultados del Algoritmo Genético 58
Ilustración 10. Resultados del Algoritmo de Búsqueda Tabú 59

Índice de Tablas

Tabla 1 Fuente: Asociación Americana de Autoridades Portuarias, Comisión Económica para América Latina y el Caribe, Agencia Nacional de Transportes Acuáticos, Autoridad Marítima de Panamá, [16]. 6
Tabla 2. Resultados de los algoritmos 55
Tabla 3. Resultados de la prueba Z 61

ET LUX IN TENEBRIS LUCET
MCMXVII

Capítulo 1. Generalidades

1.1 Problemática

Un punto clave para tener éxito en el mercado actual de la comercialización es administrar correctamente la logística y el almacenamiento de los productos (A & A Customs Brokers, 2016). La tendencia de los últimos años sigue el camino de empresas cuya ventaja reside en la capacidad de realizar envíos internacionales de gran cantidad de productos por lo que la eficiencia en la capacidad de transporte es un requisito obligatorio para las empresas comercializadoras que quieran mantener o incrementar su presencia en el mercado (Liang et al., 2007). Los problemas más comunes que se presentan a las empresas de este rubro son la optimización de las rutas de transporte, disminución de los costos de almacenaje, métodos de transporte, envío y distribución de productos, etc. (Liang et al., 2007).

Desde la creación del contenedor en 1956, su empleo como unidad de transporte y embalaje de cargas para la comercialización de diferentes productos fue incrementado de tal manera que incluso se construyeron buques especializados específicamente para llevar contenedores (Tiba Group, 2016). Empresas alrededor del mundo encontraron en los contenedores la forma ideal de proveer mercadería a sus clientes gracias a la reducción de tiempo en el tránsito de la carga, la seguridad que proveen y, sobre todo, a la reducción de costos.

Debido a la progresiva globalización del uso de los contenedores en el transporte marítimo internacional y a la creación de una extensa variedad de estos: Refrigerados, aislantes, tanques, etc., se tuvo la necesidad de controlar y estandarizar sus características como el tamaño, carga máxima y métodos de identificación, con este fin se crearon los estándares ISO 668 (Serie 1 contenedores de carga - Clasificación, dimensiones y posiciones) e ISO 6346 (Contenedores de carga - Codificación, identificación y marca). La unidad estándar para medir un contenedor es la Unidad Equivalente a Veinte Pies (TEU por el término en inglés Twenty-foot Equivalent Unit), hace referencia a la capacidad de carga de un contenedor normalizado de veinte pies. También es usada para calcular cuántos contenedores puede llevar una embarcación o cuántos contenedores puede manejar un puerto (OECD, 2002). Gracias a estos

estándares, las empresas comercializadoras ahora pueden asegurar que sus envíos pueden ser atendidos en los principales puertos del mundo ya que la clasificación e identificación de los contenedores es obligatoria (SUNAT, 2012).

Actualmente el tráfico de contenedores en los principales puertos del mundo pone en evidencia que su uso es primordial para la comercialización de hoy en día, como puede apreciarse en la siguiente tabla:

Tabla 1 Fuente: Asociación Americana de Autoridades Portuarias, Comisión Económica para América Latina y el Caribe, Agencia Nacional de Transportes Acuáticos, Autoridad Marítima de Panamá, [16].

PUERTOS DE AMÉRICA CENTRAL Y AMÉRICA DEL SUR TRÁFICO DE CONTENEDORES 2014 - 2016 Twenty-Foot Equivalent Units - TEUs			
	2016	2015	2014
ARGENTINA			
Buenos Aires (incluye Exolgan)	1,352,311	1,427,532	1,089,507
BRAZIL			
Santos	3,564,118	3,779,999	3,684,845
CHILE			
San Antonio	1,287,658	1,077,478	1,089,301
COLOMBIA			
Cartagena (incluye SPRC, El Bosque, Contecar)	2,301,099	2,397,969	2,236,551
COSTA RICA			
Limon/Moin	1,177,374	1,106,267	1,089,507
ECUADOR			
Guayaquil	1,821,654	1,704,730	1,621,381
PANAMA			
Balboa	2,831,893	3,078,101	3,236,355
Colon	2,464,440	2,764,644	2,564,102
PERU			
Callao	2,054,970	1,900,444	1,992,473

A pesar de los beneficios del uso de los contenedores, las empresas comercializadoras han tenido problemas en optimizar el espacio utilizado dentro de estos debido principalmente a que el orden y distribución de los paquetes que son ingresados en los contenedores es inadecuado ya que no se toma en cuenta el peso, forma y volumen de estos. Este problema empeora cuando el personal encargado no posee el conocimiento ni la capacitación necesaria para realizar la tarea encomendada (Jiménez S. y Jiménez C., 2015). Por esta razón, el procedimiento de llenado de contenedores está expuesto al error humano al no contar con una especificación previa de dónde y cómo colocar cada paquete dentro del contenedor lo que, finalmente, puede ocasionar un desperdicio de espacio del contenedor y obligar a la empresa a tener que utilizar más contenedores de

los necesarios, lo que se traduce en mayores costos incurridos en alquiler o compra de estos.

Por lo tanto, plantear un orden de colocación de los paquetes dentro del contenedor con la finalidad de optimizar el espacio usado de los contenedores se vuelve el principal problema para este proyecto de fin de carrera ya que la solución de este permitirá transportar más paquetes haciendo uso de menos contenedores, lo que finalmente conlleva a una reducción de los costos de transporte por contenedor incurridos por la empresa ya que estos dependen de la cantidad de contenedores que se utilicen. Entre los costos por transporte de un contenedor completo se encuentran el flete internacional, el recargo por combustible, el ajuste por tipo de cambio, recargos que dependen del destino como el recargo por riesgo de piratería, recargo por cruzar un canal, recargo por congestión, recargo por desequilibrio de equipo y finalmente, los gastos en los puertos como los gastos de carga y descarga del contenedor en el barco, tasas portuarias y gastos relacionados a la nota de embarque (iContainers, 2002).

Este problema es conocido en la bibliografía como el *problema de llenado de contenedores en tres dimensiones* (***3D Bin Packing Problem*** en inglés). Es un problema de optimización combinatoria que se enfoca en mejorar el espacio utilizado por objetos en un espacio limitado, en este caso, un contenedor. Los objetos son colocados siguiendo ciertas restricciones para lograr el objetivo deseado. Este problema incrementa su complejidad computacional en un tiempo polinomial no determinista a medida que las variables como la cantidad de objetos o el tamaño de estos varía, por eso se le denomina NP-difícil (término para describir al conjunto de problemas que son por lo menos tan difíciles como un problema que se puede resolver en un tiempo polinómico) (Garey, M. y Johnson D., 1979). El problema de embalaje tridimensional puede simular varios problemas reales en el planeamiento de transporte y almacenamiento como el llenado de embarcaciones de contenedores, administración de almacenes, carga de paletas, etc. Se han planteado muchos enfoques para la resolución de este problema dentro de los cuales se encuentran diferentes heurísticas y metaheurísticas, siendo este último el más utilizado debido a los resultados obtenidos (Afza et al., 2014) (Kacprzak et al, 2015).

Para este proyecto de fin de carrera se propone el diseño y construcción de un algoritmo metaheurístico que brinde una solución para el problema descrito. Los algoritmos metaheurísticos han demostrado ser capaces de brindar, en la mayoría de los casos, los mejores resultados para resolver problemas complejos en un tiempo computacional aceptable como el problema de embalaje tridimensional. En la literatura se aprecia que uno de los algoritmos más usados para obtener una solución a este problema es el Algoritmo Genético, tomando como ejemplo los trabajos de Pino, R., Fuente, D., Quesada, I. y García, N. y Wang, Hongfeng y Chen, Yanjie. Asimismo, se observa que el Algoritmo Tabú es capaz de proporcionar muy buenos resultados en problemas de optimización combinatoria (Werra, D., y Hertz, A., 1989). Con esta base, se construirá un algoritmo usado anteriormente en la literatura con la finalidad de comparar sus resultados con los obtenidos del algoritmo presentado en este proyecto, el cual estará basado en el Algoritmo Tabú.

1.2 Objetivos

1.2.1 Objetivo general

Aplicar el algoritmo de Búsqueda Tabú que permita optimizar el espacio utilizado en los contenedores para brindar una alternativa de solución al problema de llenado de contenedores considerando restricciones de peso y fragilidad.

1.2.2 Objetivos específicos

O1. Definir la función objetivo que será utilizada por el Algoritmo Genético y el algoritmo de Búsqueda Tabú.

O2. Adaptar un algoritmo genético que proporcione una posible solución al problema de llenado de contenedores.

O3. Diseñar un algoritmo de búsqueda tabú que brinde una alternativa de solución al problema de llenado de contenedores.

O4. Implementar prototipos funcionales que implementen ambos algoritmos.

O5. Realizar la experimentación numérica que permita comparar los resultados de ambos algoritmos para determinar cuál brinda la solución óptima al problema.

1.2.3 Resultados esperados

R 1. Documento que indique la función objetivo como una fórmula matemática, la cual se desea optimizar. (O1)

R 2. Documento que especifique la estructura de datos que se usará para el algoritmo genético. (O2)

R 3. Documento que contenga el pseudocódigo del algoritmo genético obtenido de la literatura. (O2)

R 4. Documento que especifique la estructura de datos que se usará para el algoritmo de búsqueda tabú. (O3)

R 5. Documento que contenga el pseudocódigo del algoritmo de búsqueda tabú. (O3)

R 6. Programa que utilice los algoritmos genético y búsqueda tabú para brindar posibles soluciones al problema. (O4)

R 7. Aplicación que permita visualizar los resultados de los algoritmos.

R 8. Documento que contenga la experimentación numérica, realizada con pruebas estadísticas, que demuestre cuál algoritmo brinda mejores resultados. (O5)

1.2.4 Mapeo de objetivos, resultados y verificación

Objetivo: Plantear la función objetivo que será utilizada por el Algoritmo Genético y el algoritmo de Búsqueda Tabú.		
Resultado	**Meta física**	**Medio de verificación**
Función objetivo	Documento	• Pruebas de funcionamiento. • Juicio experto que valide la función.

Objetivo: Diseñar un algoritmo genético que proporcione una posible solución al problema de llenado de contenedores.		
Resultado	**Meta física**	**Medio de verificación**
Pseudocódigo del algoritmo genético	Documento	Seguimiento paso a paso manualmente del pseudocódigo para comprobar un comportamiento adecuado del algoritmo genético.

Objetivo: Diseñar un algoritmo de búsqueda tabú que brinde una alternativa de solución al problema de llenado de contenedores		
Resultado	**Meta física**	**Medio de verificación**
Pseudocódigo del algoritmo de búsqueda tabú	Documento	Seguimiento paso a paso manualmente del pseudocódigo para comprobar un comportamiento adecuado del algoritmo de búsqueda tabú.

Objetivo: Implementar los algoritmos en un determinado lenguaje de programación.		
Resultado	**Meta física**	**Medio de verificación**
Algoritmo genético implementado.	Software	● Comprobación de los resultados obtenidos al ejecutar el programa paso a paso. ● Pruebas unitarias

Algoritmo de búsqueda tabú implementado.	Software	● Comprobación de los resultados obtenidos al ejecutar el programa paso a paso. ● Pruebas unitarias
Interfaz que permita ver los resultados de los algoritmos	Software	● Aplicación que permita visualizar los resultados de los algoritmos. ● Pruebas unitarias

Objetivo: Realizar la experimentación numérica que permita comparar los resultados de ambos algoritmos para determinar cuál brinda la solución más óptima al problema.		
Resultado	**Meta física**	**Medio de verificación**
Experimentación numérica que demuestre cuál algoritmo brinda mejores resultados	Documento	Análisis de los resultados de la experimentación numérica e interpretación de estos.

1.3 Herramientas y Métodos

Resultados Esperados	**Herramientas y métodos a usarse**
R1: Documento que indique la función objetivo como una fórmula matemática, la cual se desea optimizar	Pruebas de funcionamiento
R2: Documento que especifique la estructura de datos que se usará para el algoritmo genético	Microsoft Visual Studio, lenguaje C#, SCRUM
R3: Documento que contenga el pseudocódigo del algoritmo genético obtenido de la literatura.	Método del Algoritmo Genético
R4: Documento que especifique la estructura de datos que se usará para el algoritmo de búsqueda tabú.	Microsoft Visual Studio, lenguaje C#, SCRUM
R5: Documento que contenga el pseudocódigo del algoritmo de búsqueda tabú.	Método del Algoritmo Búsqueda Tabú
R6: Programa que utilice los algoritmos genético y búsqueda tabú para brindar posibles soluciones al problema.	Microsoft Visual Studio, lenguaje C#, SCRUM
R7: Aplicación que permita visualizar los resultados de los algoritmos.	Microsoft Visual Studio, lenguaje C#, SCRUM
R8: Documento que contenga la experimentación numérica, realizada con pruebas estadísticas, que demuestre cuál algoritmo brinda mejores resultados.	R Studio, Prueba F de Fisher, Kolmogorov-Smirnov, Prueba Z. Microsoft Excel

1.3.1 R Studio

Es un entorno de desarrollo integrado de código abierto para el lenguaje de programación R (RStudio, 2017). Permite utilizar todas las capacidades del lenguaje y brinda facilidades como:

- Autocompletado de código.
- Ejecución de código en R desde el editor.
- Creación y visualización de gráficos.
- Ayuda y documentación integrada.

Esta herramienta se utilizará para facilitar la realización de la experimentación numérica dado que permite calcular automáticamente los resultados de los métodos relacionados al último resultado esperado.

1.3.2 Microsoft Visual Studio

Es un entorno de desarrollo integrado que de forma predeterminada tiene compatibilidad con los lenguajes de programación C#, C, C++, JavaScript, F# y Visual Basic. Proporciona las herramientas necesarias para el diseño y construcción de software, gracias a esto permite crear aplicaciones, juegos, sitios web y servicios web entre otros (Microsoft, 2017).

Se utilizará esta herramienta ya que permite agilizar el desarrollo de los algoritmos gracias a sus características de autocompletado de código y fácil depuración de tal manera que se pueda invertir mayor tiempo en la lógica de los algoritmos.

1.3.3 Método del Algoritmo Genético

El método que se usará para construir el algoritmo genético es el descrito por Holland (Holland, 1975) y parte del trabajo de Teodor Crainic, Guido Perboli y Roberto Tadei sobre el concepto de los puntos extremos, el cual hace referencia a los puntos resultantes cada vez que se ingresa un nuevo elemento al contenedor como se muestra en la imagen (Crainic et al., 2007):

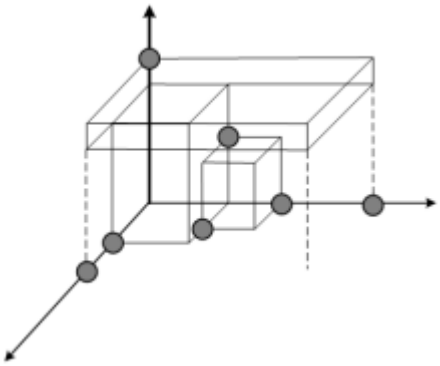


Ilustración 1: Puntos Extremos de un objeto (Crainic et al., 2007)

Se utilizará este método ya que ha demostrado brindar buenos resultados al momento de calcular los puntos disponibles para colocar nuevos objetos dentro del contenedor.

1.3.4 Método del Algoritmo de Búsqueda Tabú

El método que se usará para diseñar el algoritmo de búsqueda tabú será básicamente el mismo que utilizó Fred Glover para describir el algoritmo. Este método se basa en sacar provecho de estrategias de aprendizaje utilizando estructuras de datos especializadas para este trabajo, las cuales se adaptan a los cambios que ocurren a lo largo de la ejecución del algoritmo (Glover, 1989).

Dado que la propuesta de solución planteada en este proyecto de fin de carrera implica el diseño de un algoritmo de búsqueda tabú, es imperativo usar este método para su desarrollo.

1.3.5 Lenguaje de programación C#

C# es un lenguaje de programación orientada a objetos construido para desarrollar aplicaciones y programas web, en la nube, de escritorio y más. Posee una sintaxis similar a la del lenguaje Java y será utilizado junto con el entorno de desarrollo Visual Studio (Microsoft, 2017).

Se utilizará este lenguaje de programación para desarrollar los algoritmos ya que es un lenguaje amigable y relativamente fácil de usar. Otra razón es que este lenguaje se puede utilizar en Visual Studio.

1.3.6 SCRUM

Es un proceso en el que se utilizan buenas prácticas para obtener el resultado más cercano a lo óptimo de un proyecto. El origen de estas buenas prácticas se da a partir de una investigación sobre la forma de trabajar de equipos que han demostrado ser productivos. Este proceso se basa en entregas parciales, donde cada una de estas consta de una iteración la cual proporciona un resultado completo, es decir, un parte del proyecto funcional (SCRUM, 2016).

Esta metodología se utilizará a lo largo del desarrollo de los algoritmos ya que es ideal para este proyecto de fin de carrera donde se requerirán entregas parciales cada determinado tiempo.

1.3.7 Prueba Z

Es una prueba estadística que se utiliza para comparar las medias de dos muestras bajo ciertas condiciones previas. Estas condiciones son que ambas muestras sigan una distribución normal y que posean varianzas homogéneas (Sprinthall, 2011).

Esta prueba se utilizará en la experimentación numérica para discernir cuál de los dos algoritmos brinda mejores resultados.

1.3.8 Prueba F de Fisher

La prueba F de Fisher, creada por Ronald Fisher, es una prueba estadística con la que se puede comparar las varianzas de dos poblaciones de manera tal que se determine cuál de las dos varía más sobra la otra. También se puede utilizar para contrastar la hipótesis de la igualdad de las medias de las medias de dos poblaciones (Fisher, 1992).

Esa prueba se utilizará como parte de la experimentación numérica ya que permitirá determinar cuál de los dos algoritmos es mejor que el otro.

1.3.9 Prueba Kolmogorov-Smirnov

La prueba Kolmogorov-Smirnov permite determinar si una muestra sigue una determinada distribución. En este caso se utilizará con la finalidad de verificar que los resultados de los algoritmos siguen una distribución normal. Las pruebas de este tipo se suelen llamar pruebas de bondad de ajuste (Lopes, 2011).

Al igual que el método anterior, este método se utilizará como parte de la experimentación numérica dado que sus resultados son un requisito para la prueba Z.

1.4 Alcance, limitaciones y riesgos

1.4.1 Alcance

Este proyecto busca brindar una alternativa de solución al problema del llenado de contenedores de exportación de medidas estandarizadas. Para esto se utilizarán dos algoritmos, uno genético y otro de búsqueda tabú, con el propósito de discernir cuál de estos es mejor para solucionar este problema. Con la finalidad de comprobar la eficiencia del algoritmo de búsqueda tabú, se procederá a diseñarlo e implementarlo, mientras que el algoritmo genético será adaptado a partir de un algoritmo obtenido de la revisión bibliográfica. Para este proyecto de fin de carrera se deben considerar los siguientes puntos:

- El algoritmo de búsqueda tabú será aplicable a toda clase de contenedor ya que se considerarán que estos siguen medidas definidas por estándares internacionales.
- Los resultados de los algoritmos no serán necesariamente los mejores debido al factor de aleatoriedad implícito en estos. Sin embargo, los resultados serán cercanos a los óptimos ya que se aplicarán heurísticas de decisión basadas en la función objetivo.
- El algoritmo genético utilizará parte de investigaciones y trabajos de otros autores y su resultado será utilizado como parte inicial para el algoritmo de búsqueda tabú.
- Se tendrá un módulo donde se podrán verificar los resultados de ambos algoritmos de tal manera que sea fácil la comparación de sus resultados.

1.4.2 Limitaciones

Este proyecto de fin de carrera afronta las siguientes limitaciones:

- El tiempo de ejecución de los algoritmos se verá afectado dependiendo del equipo en donde sean ejecutados y de la complejidad de los mismos.

1.4.3 Riesgos

Riesgo	Impacto en el proyecto	Medidas correctivas para mitigar
Planeamiento inadecuado del proyecto.	Presentación de los entregables fuera de la fecha programada.	Reestructuración de la planeación con ayuda del asesor.
Pérdida del documento del proyecto.	Retrabajo y tiempo extendido para la presentación de entregables	Mantener por lo menos un respaldo físico y uno virtual.
Poca disponibilidad del asesor.	Presentación de entregables con errores	Presentar avances de los entregables con anticipación

1.5 Justificación y Viabilidad del proyecto

1.5.1 Viabilidad Técnica

Respecto a las nociones técnicas de este proyecto de fin de carrera no se cuenta con grandes limitaciones, esto gracias a que las metodologías, métodos y herramientas que se usarán han sido aplicados con anterioridad a lo largo de la carrera universitaria.

Asimismo, las metodologías mencionadas anteriormente, como SCRUM, siguen las mejores prácticas internacionales, las cuales servirán como base y apoyo para poder lograr los objetivos de este proyecto.

En relación con esto último, es necesario mencionar que, tanto las herramientas como el soporte necesario que se utilizarán están disponibles en la universidad por lo que no habrá problemas para acceder a estas.

Finalmente, cabe resaltar que para el desarrollo de los algoritmos se cuenta con los conocimientos del asesor quien posee vasta experiencia sobre estos y a su vez se cuenta con amplia información disponible en la literatura en caso sea necesaria para resolver las dudas que puedan surgir a lo largo del desarrollo de este proyecto de fin de carrera.

1.5.2 Viabilidad Temporal

La viabilidad temporal es uno de los principales factores a considerar ya que, a pesar de la complejidad del proyecto, normalmente solo se cuenta con el tiempo que dure el curso de Proyecto de Tesis 2 para poder desarrollarlo, sin embargo, en este caso se contará con el tiempo adicional de los meses de verano. Por lo tanto, con una correcta definición del cronograma del proyecto, este será temporalmente viable. A continuación, se presenta un posible cronograma:

Tarea	**Fecha de inicio**	**Fecha de fin**
Definición de la función objetivo	08/01/2018	20/01/2018
Desarrollo de la estructura de datos del algoritmo genético	22/01/2018	01/02/2018
Desarrollo del pseudocódigo del algoritmo genético	03/02/2018	18/02/2018
Desarrollo de la estructura de datos del algoritmo de búsqueda tabú	20/02/2018	28/02/2018
Desarrollo del pseudocódigo del algoritmo de búsqueda tabú	02/03/2018	12/03/2018
Implementación del algoritmo genético	14/03/2018	31/03/2018
Implementación del	02/04/2018	22/04/2018

algoritmo de búsqueda tabú		
Implementación de interfaz gráfica	24/04/2018	10/05/2018
Realizar la experimentación numérica	12/05/2018	19/05/2018

1.5.3 Viabilidad Económica

La viabilidad económica no representa un impedimento para este proyecto ya que todas las herramientas a utilizar están disponibles dentro de la universidad o ya se poseen. Por lo tanto, no hay ningún costo a considerar.

1.5.4 Justificación del proyecto

Hoy en día, las empresas comercializadoras no solo deben preocuparse de cuánto venden, sino también de cómo realizan las ventas. En un mundo globalizado como el de hoy la mejor opción para vender al exterior es transportar la mercadería mediante el uso de los contenedores, sin embargo, el empleo de estos y de su espacio debe ser el óptimo de tal manera que se utilice la menor cantidad de contenedores en cada envío de mercadería ya que por cada contenedor que se envíe los costos por compra o alquiler y los fletes aumentan. En la actualidad hay algunas herramientas que proporcionan una alternativa de solución al problema del desperdicio de espacio en los contenedores, pero estas herramientas sólo consideran el espacio utilizado, más no otros factores importantes como la distribución del peso y la fragilidad de los objetos.

Este proyecto de fin de carrera permitirá tener en consideración los aspectos antes mencionados para brindar una alternativa de solución, la cual servirá a las empresas comercializadoras a tener un plan de llenado de los contenedores más cercano a lo óptimo y con las consideraciones pertinentes respecto a las características de los objetos en un ambiente real.

Capítulo 2. Marco Legal/Regulatorio/Conceptual/otros

En este apartado se presentarán los conceptos clave para un mejor entendimiento del documento en general. Estos conceptos abarcan el problema definido en el capítulo 1 y el desarrollo del algoritmo para brindar la solución al problema.

2.1 Contenedor

Según el Convenio Aduanero sobre Contenedores de 1972 un contenedor es "un elemento de equipo de transporte (cajón portátil, tanque movible u otro elemento análogo)" resistente para que pueda ser utilizado repetidas veces por diferentes medios de transporte sin necesidad de manipular el contenido. El contenedor posee las características necesarias para que pueda ser movido, cargado o descargado fácilmente (UNECE, 1972).

En otras palabras, un contenedor es una caja de tamaño estandarizado que permite facilitar el proceso de transporte de mercadería. Puede ser de varios tipos: isotermo, frigorífico, calorífico, de temperatura controlado, etc.

Los contenedores más usados son los contenedores estándar para carga no perecedera:

- Contenedor estándar para carga no perecedera de 20 pies (largo: 6 metros, ancho: 2.4 metros, alto: 2.5 metros), tiene un peso máximo de 24000 kg.
- Contenedor estándar para carga no perecedera de 40 pies (largo: 12 metros, ancho: 2.4 metros, alto: 2.5 metros), tiene un peso máximo de 30480 kg.
- Contenedor High Cube, tiene las mismas características que el contenedor de 40 pies, pero una altura de 2.8 metros.

2.2 Paleta

Una paleta o también llamada palé es una plataforma horizontal mayormente de madera que puede ser movida o elevada por una pequeña grúa hidráulica. Se utiliza para apilar objetos sobre esta, de tal manera que todos los objetos puedan ser movidos como un solo bloque sólido. Según el estándar ISO 6780 (Pallets para el manejo intercontinental de materiales - principales dimensiones y tolerancias) las medidas en centímetros de las paletas son las siguientes (LeBlanc, 2016):

- Para Norte América: 1219x1016

- Para Europa y Asia: 1000x1200
- Para Australia: 1165x1165
- Para Norte América, Europa y Asia: 1067x1067
- Para Asia: 1100x1100
- Para Europa: 800x1200

2.3 Unidad Equivalente a 20 Pies (TEU por sus siglas en inglés)

Se refiere a la capacidad de carga de un contenedor normalizado ISO de veinte pies para medir el tamaño de los contenedores. También se usa para calcular cuántos contenedores puede llevar una embarcación o para medir la capacidad de un puerto marítimo (OECD, 2002).

2.4 Puerto

Es el lugar geográfico donde se encuentran los terminales especializados para desarrollar las actividades portuarias. Los puertos se clasifican de la siguiente manera (MINCETUR, 2015):

- Por la titularidad de sus instalaciones pueden ser públicos o privados.
- Por la ocupación y uso de sus instalaciones pueden ser de uso general (están a disposición de cualquiera) o de uso exclusivo (el propietario decide quién lo usa).
- Por la actividad que se desarrolla en ellos pueden ser especializados (comerciales, turísticos, industriales, minero-industriales, pesqueros y marinos) o multipropósito (atienden diversas actividades portuarias).
- Por su ubicación pueden ser marítimos, fluviales o lacustres.
- Por su alcance y ámbito pueden ser nacionales o regionales.

2.5 Transporte marítimo internacional

Es el traslado de carga o pasajeros por el mar de un país a otro. Para el caso del traslado de cargas, se utilizan buques mercantes (barcos con una o varias cubiertas con características adecuadas para actividades marítimas) que son capaces de transportar contenedores o carga suelta. Los buques portacontenedores tienen bodegas verticales en donde el contenedor se mueve mediante rieles con sistemas automáticos y poseen diferentes equipos que le permiten manejar el 60% de su capacidad bajo la cubierta del

buque y el resto, sobre esta. Estos buques han permitido cubrir la demanda de la economía mundial que sigue creciendo gracias a que son capaces de realizar viajes cada vez más largos con mayor carga sobre ellos (MINCETUR, 2015).

2.6 Problema de Bin Packing

En general, el problema de llenado de contenedores consiste en tener un conjunto de n cajas rectangulares, cada una con sus dimensiones definidas por largo x_i, ancho y_i y alto z_i donde i pertenece a I = {1, 2, 3, …, n}, y un número ilimitado de contenedores idénticos con largo X, ancho Y y alto Z. Las n cajas deben ser introducidas de tal manera que se utilice la menor cantidad posible de contenedores (Martello et al., 2000). Este problema tiene variaciones en una, dos y tres dimensiones, siendo el último caso el más complejo y más usado en casos de la realidad como el almacenamiento interno en paletas, almacenamiento de productos terminados en anaqueles, llenado de contenedores, camiones, buques o aviones, etc.

2.7 Complejidad P

Es la clase de complejidad que incluye al conjunto de problemas de decisión cuya solución se puede calcular en un tiempo computacional polinómico mediante un algoritmo determinístico, es decir, un algoritmo que, bajo los mismos datos de entrada, siempre obtiene los mismos resultados (Garey & Johnson, 1979).

2.8 Complejidad NP

Es la clase de complejidad que incluye al conjunto de problemas de decisión donde es posible calcular una solución mediante un algoritmo no determinístico, es decir, un algoritmo que, bajo los mismos datos de entrada, no siempre se obtiene los mismos resultados, y verificar su correctitud en tiempo computacional polinómico (Garey & Johnson, 1979).

2.9 Complejidad NP-Difícil

También llamado NP-Completo, es la clase de complejidad que, dentro del conjunto de problemas NP, se clasifican como más difíciles. Según Stephen Cook, un problema X es

NP-Completo si pertenece al conjunto de problemas NP y si todo problema que pertenezca al conjunto NP se puede convertir, mediante transformación polinómica, en *X* (Garey & Johnson, 1979).

2.10 Optimización Combinatoria

Es el estudio de problemas de optimización que consiste en calcular la solución óptima dentro del conjunto finito de soluciones posibles a un determinado problema (Papadimitriou & Steiglitz, 1998). Este tipo de problemas se pueden resolver mediante búsqueda exhaustiva, sin embargo, la cantidad de soluciones posibles puede llegar a ser tan grande que no es factible hallar la mejor solución haciendo uso de este método. En un caso simple, el número de posibles soluciones aumenta en el orden de *n!*, por lo que el tiempo computacional necesario para resolverlo mediante búsqueda exhaustiva podría estar en el rango de días, meses o incluso años dependiendo de la complejidad del problema.

2.11 Algoritmo Heurístico

Un algoritmo heurístico es aquel algoritmo que puede calcular una solución lo suficientemente buena para un problema, pero si tener la seguridad de que se puede probar que esta solución es óptima o correcta (Michalewicz & Fogel, 2013).

Los algoritmos heurísticos no se enfocan en tener la mejor precisión o calidad en las soluciones que brindan, sin embargo, compensan estas debilidades con poca carga computacional (Brownlee, 2011).

2.12 Algoritmo Metaheurístico

Los algoritmos metaheurísticos tienen el propósito de extender las funcionalidades de los algoritmos heurísticos al combinar dos o más métodos heurísticos o procedimientos. En un algoritmo metaheurístico, estos procedimientos son considerados cajas negras de tal manera que estos se pueden reemplazar por otros más simples o más complejos sin afectar la estructura general del algoritmo (Brownlee, 2011).

Los algoritmos metaheurísticos se caracterizan por la capacidad que tienen de guiar el proceso de búsqueda mediante ciertas condiciones y restricciones. Por último, cabe mencionar que este tipo de algoritmos no se enfocan en un tipo de problema específico, por ejemplo, el algoritmo tabú puede ser utilizado para resolver problemas como el problema del viajero o el problema de bin packing (Blum & Roli, 2003).

2.13 Algoritmo Genético

Es un algoritmo metaheurístico que intenta simular la evolución genética y la selección natural que ocurre en la realidad. Este algoritmo imita los procesos de la naturaleza necesarios para la evolución, siendo uno de los más conocidos, la supervivencia del más apto (Holland, 1975). El algoritmo genético requiere de una población o generación inicial generada aleatoriamente, donde cada individuo de la población, llamado también cromosoma, representará a una posible solución del problema y está asociado a un determinado valor *fitness* (valor que representa cuán adecuada es una solución), el cual indica cuán buena es la solución dependiendo de la función objetivo que se plantee. Son estos individuos los que pasarán por el proceso de evolución, el cual consiste básicamente en tres operaciones:

- Selección: operación que consiste escoger a los mejores individuos de la generación, los que tengan el mejor valor *fitness*, para que pasen sus genes a la siguiente generación.
- Cruce: operación que simula la reproducción entre dos individuos. Consiste en seleccionar dos individuos de la población, mediante la operación de Selección, para que puedan combinar sus genes y crear un nuevo individuo para la siguiente generación, el cual idealmente, tendrá un mejor valor *fitness*. La representación binaria de una operación de cruce es la siguiente:
    - ➢ Individuo 1: “00011101”
    - ➢ Individuo 2: “01101011”
    - ➢ Nuevo Individuo: “00011011”
- Mutación: operación que consiste en cambiar algunos de los genes de ciertos individuos de la población con la finalidad de establecer cierta diversidad entre

generaciones y evitar tener muchas soluciones cercanas. Esta operación tiene una probabilidad muy baja de ocurrir ya que, de ocurrir repetidas veces, será difícil converger en una solución. La representación binaria de una operación de mutación es la siguiente:

- ➢ Individuo 1: “11101100”
- ➢ Nuevo individuo: “11110000”

Finalmente, el algoritmo concluirá y devolverá la mejor solución encontrada luego de alcanzar la condición de parada, esta puede ser un número de iteraciones predeterminado, número de generaciones creadas, etc.

2.14 Algoritmo Búsqueda Tabú

Es un algoritmo metaheurístico que busca resolver problemas de optimización. Su principal característica es que cuenta con una heurística específica para evitar que se visiten áreas del espacio de búsqueda recientemente visitadas. A esta característica se le llama memoria de corto plazo y se representa por una lista tabú de tamaño específico, esta estructura almacena los movimientos más recientes y los marca como movimientos prohibidos para las siguientes *n* iteraciones. Gracias a esta heurística, la búsqueda tabú evita quedar encerrada en un estado cíclico, es decir, escapa de los óptimos locales mientras se acerca más al óptimo global (Glover, 1989).

El algoritmo empieza con una solución inicial y una lista tabú vacía. Luego comienza un bucle hasta llegar a una condición de parada, la cual es definida en la fase de diseño del algoritmo. Dentro del bucle se genera un vecindario (conjunto de soluciones) a partir de la solución actual, la cual será la solución inicial para la primera iteración. Para cada vecino de este vecindario se comprueba si no se encuentra en la lista tabú, si no está marcado como prohibido, se calcula si es una mejor solución haciendo uso de la función fitness, de ser el caso, el vecino se asigna como solución actual y se agrega a la lista tabú. A continuación, se presenta un pseudocódigo de la Búsqueda Tabú (Brownlee, 2011):

```
Entrada: TamañoListaTabu
Salida: MejorSolucion

MejorSolucion <- CrearSolucionInicial()

ListaTabu <- {}

Mientras(No CondicionSalida)

 ListaCandidatos <- {}

 Para(Candidato en Vecindario(MejorSolucion))

  Si(No EnListaTabu(Candidato,ListaTabu))

   ListaCandidatos <- Candidato

  Fin Si

 Fin Para

 Candidato <- EncontrarMejorCandidato(ListaCandidatos)

 Si(Fitness(Candidato) > Fitness(MejorCandidato))

  MejorCandidato <- Candidato

  ListaTabu <- MejorCandidato

  Mientras(Tamaño(ListaTabu) > TamañoListaTabu)

   BorrarUltimo(ListaTabu)

  Fin Mientras

 Fin Si

Fin Mientras

Devolver(MejorCandidato)
```

Ilustración 2. Pseudocódigo Algoritmo de Búsqueda Tabú

Capítulo 3. Estado del Arte

Disminuir el desperdicio del espacio usado en los contenedores es un problema común, pero de características complejas ya que es posible considerar varios factores como el peso total, distribución del peso, fragilidad de los objetos, etc. Se han planteado diversas formas para resolver este problema, pero considerando solamente algunos factores por separado. En este apartado se presentan algunos intentos de brindar una solución al problema planteado con la finalidad de evidenciar el trabajo alrededor del objetivo de disminuir el desperdicio del espacio utilizado en contenedores.

3.1 Enfoques para la resolución del problema

3.1.1 Optimización del Llenado de Contenedores para Transporte Multimodal

En este proyecto se utiliza un algoritmo genético para calcular una solución al problema de llenado de contenedores para una empresa en particular. La solución propuesta se basa en formar una imagen de cómo estarían las cajas dentro del contenedor. Para esto primero se genera la secuencia de llenado de las cajas más grandes con la ayuda de un algoritmo genético y luego se colocan las cajas más pequeñas en los espacios libres. Si el contenedor no está lo suficientemente lleno se trata de completar el espacio con cajas de un pedido para otro cliente, formando un contenedor multicliente, con la restricción que el pedido tiene que tener el mismo destino.

El método fue probado con dos casos complejos donde se tienen varios tipos de cajas de diferentes tamaños, obteniendo un 80% y 60% de volumen ocupado del contenedor respectivamente. Cabe resaltar que, para el último caso, el porcentaje se puede aumentar si se trabaja con contenedores multicliente (Pino et al., 2017).

3.1.2 Hybrid greedy heuristics based on linear programming for the three-dimensional single bin-size bin packing problem

En este proyecto se propone una solución para el problema de llenado de un solo contenedor utilizando una heurística codiciosa simple de programación lineal. La propuesta es dividir el problema en series de problemas de la mochila en tres dimensiones. Primero se selecciona un subconjunto de objetos donde cada uno de estos

puede ser ingresado en el contenedor. Luego se coloca el subconjunto seleccionado en el contenedor. El proceso se repite hasta que se empaquen todos los objetos. Finalmente se considera una segunda heurística que es una versión mejorada de la primera, donde en la primera fase se optimiza el subconjunto de objetos seleccionado.

Ambas heurísticas presentaron resultados por lo menos tan buenos como los presentes en la literatura (Hifi et al., 2014).

3.1.3 Diseño de un algoritmo de búsqueda tabú para la minimización del desperdicio de espacio en almacenes de empresas comercializadoras de tuberías

En esta tesis se plantea una solución al problema de llenado de almacenes utilizando un algoritmo tabú. En particular se aborda el problema de colocar tuberías en anaqueles de un almacén considerando la particularidad de su forma cilíndrica. Para generar la solución inicial del algoritmo tabú se utilizó un algoritmo GRASP (algoritmo que forma soluciones aleatorias y las va mejorando con cada iteración). El algoritmo GRASP también se utilizó con la finalidad de comparar sus resultados con los del algoritmo tabú.

Finalmente se comprueba, mediante experimentación numérica, que el algoritmo tabú es mejor que el algoritmo GRASP y por lo tanto proporciona una buena solución al problema planteado (Rodríguez, 2014).

3.1.4 A hybrid genetic algorithm for 3D bin packing problems

En este proyecto se propone un algoritmo híbrido genético para brindar una solución al problema del llenado de contenedores considerando algunas rotaciones de los objetos. Para el algoritmo se usa la representación de un diploide (juego de dos cromosomas) donde el primer cromosoma representa la secuencia en la que los objetos serán introducidos y el segundo cromosoma indica la rotación del respectivo objeto. El algoritmo también usa una heurística que es una versión mejorada del método fondo

izquierdo más profundo con relleno la cual busca combinar varios objetos en uno de tal forma que el espacio utilizado sea el mayor posible.

Se utilizó un conjunto de problemas de prueba con tres posibles rotaciones propuesto por el método de Bischoff y Ratcliff para la experimentación, esta demostró que el algoritmo es un buen optimizador para el problema del llenado de contenedores en tres dimensiones (Wang et al., 2010).

3.2 Software comercial para la resolución del problema

3.2.1 3D Bin Packing

Este software usa diferentes algoritmos para colocar objetos dentro de contenedores (cajas) dependiendo de la necesidad del cliente, entre estos se encuentran:

- Llenado de un solo contenedor: Permite saber qué objetos de la orden son posibles empaquetar en el contenedor o vehículo.
- Llenado de varios contenedores: Permite encontrar una solución para empaquetar todos los objetos en la menor cantidad de contenedores posible.
- Ajustamiento de contenedor: Permite definir el tamaño del contenedor que requiere para empaquetar una cantidad definida de objetos.
- Llenado máximo de un contenedor: Calcula la cantidad máxima de objetos del mismo tamaño que se pueden colocar dentro de un contenedor.

A su vez brinda un API del cual se puede consumir un servicio para la opción de incluir sus algoritmos en proyectos separados de este software (3DBinPacking, 2012).

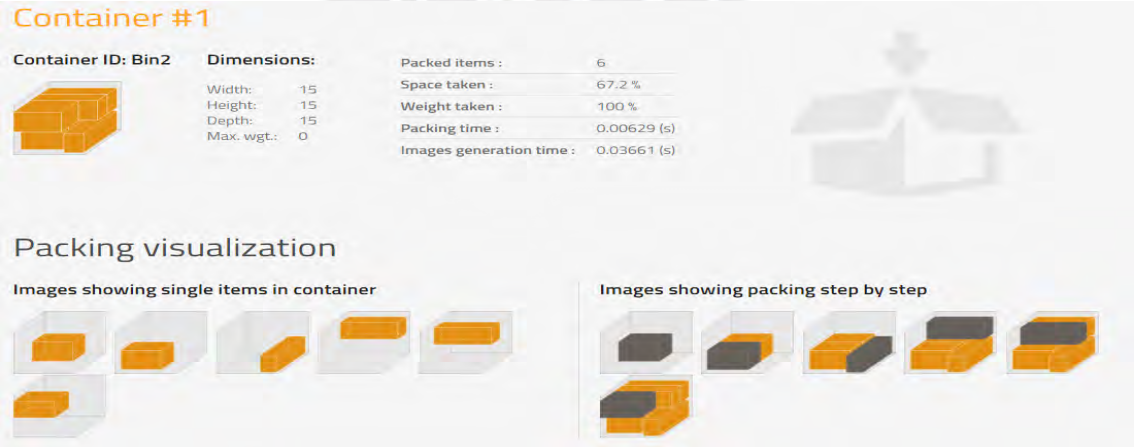


Ilustración 3: Sofware 3D Bin Packing

3.2.2 Cube-IQ Load Planning

Este software posee su propio motor de carga y se basa en optimizar la relación volumen/peso en el llenado de camiones y contenedores. Posee una interfaz gráfica que permite construir una solución manual o modificar una proporcionada por el programa. Entre sus funcionalidades principales se encuentran:

- Crear un plan de llenado a partir de reglas definidas por el usuario como secuencia de llenado, cargas parciales y distribución del peso.
- Brindar instrucciones de llenado en diagramas.
- Optimizar las cargas bajo reglas de orientación y apilamiento.
- Brindar la secuencia inversa al llenado para la descarga de los objetos.

Posee una versión web y una versión de escritorio. La versión web fue pensada para empresas que desean compartir información con sus clientes, de esta forma, ellos podrán realizar un seguimiento a sus órdenes antes de su llegada (MagicLogic Optimization Inc., 1996).

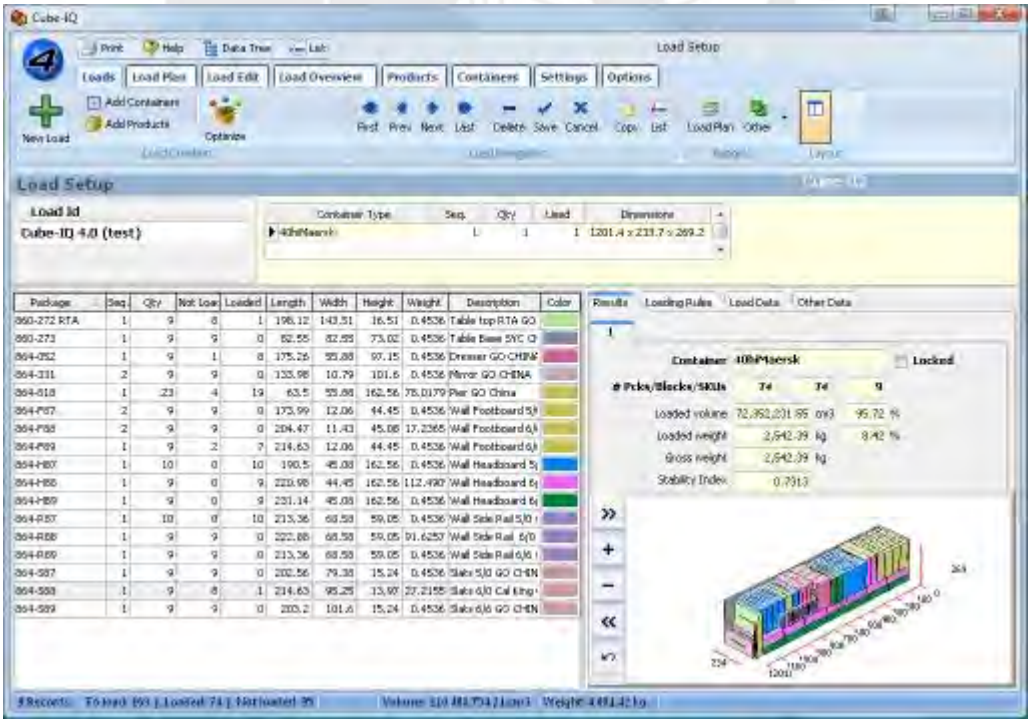


Ilustración 4: Software Cube-IQ Load Planning

3.3 Conclusiones

Con la base de este capítulo se puede evidenciar que las alternativas de solución para el problema de llenado de contenedores poseen varios enfoques y solo priorizan ciertos aspectos de este como el peso de los paquetes o pedidos relacionados a la carga de los contenedores. Otras propuestas consideran menos restricciones, por ejemplo, consideran una altura infinita para el contenedor, de tal modo que proponen una alternativa de solución rápida, pero poco realista.

Tomando en consideración lo mencionado, este proyecto de fin de carrera pretende brindar una alternativa de solución para el problema de llenado de contenedores la cual será de ayuda para las personas encargadas de realizar esta tarea ya que se basará en restricciones reales como el peso, dimensiones, volumen y distribución del peso de los paquetes dentro del contenedor de manera tal que se optimice el espacio utilizado y se distribuya la carga para mejorar la estabilidad de la misma.

Capítulo 4. Función Objetivo

4.1 Introducción

En esta sección se presentará el resultado esperado 1, el cual está directamente relacionado al objetivo 1 (O 1) de la lista de objetivos específicos. Este primer resultado, la función objetivo, busca maximizar la cantidad de cajas colocadas en el menor número de contenedores.

4.2 Planteamiento del problema

Con la finalidad de poder colocar la mayor cantidad de cajas en cada contenedor, se busca encontrar los puntos exactos dentro de estos en donde posicionar cada caja. Una propiedad importante a considerar son los volúmenes de las cajas a colocar dado que muchas de las restricciones de este problema están relacionadas a esta característica propia de cada caja. Esta importancia se puede plasmar en la siguiente formalización de la función objetivo:

$$\text{Max } \frac{\sum_{i=1}^{N} \frac{1}{Ci} \sum_{j=1}^{M} Vj \; x \; kji}{N}$$

Donde:

- N: Número de contenedores utilizados.
- M: Número de cajas en total.
- V_j: Volumen de la caja j.
- $K_{ji:}$ Indica si la caja j se coloca en el contenedor i (Solo puede tener los valores 0 o 1).
- C_i: Capacidad del contenedor i.

Para probar la validez de la función objetivo se presenta el siguiente caso de prueba en una situación extrema:

1. Se tienen dos soluciones, la primera tiene tres contenedores con una caja dentro de cada uno de estos, la segunda tiene un contenedor con tres cajas dentro. Si consideramos que las cajas tienen un volumen de 20 unidades y los contenedores, una capacidad de 100 unidades, se tiene:
    - Primera solución: $\frac{\left(20*\frac{1}{100}\right)+\left(20*\frac{1}{100}\right)+\left(20*\frac{1}{100}\right)}{3} = 0.2$

- Segunda solución: $\frac{(20+20+20)*(\frac{1}{100})}{1} = 0.6$

Como se puede apreciar, la segunda solución es la que tiene el mayor valor para la función objetivo, por lo que sería escogida por los algoritmos.

4.3 Restricciones del problema

A continuación, se mencionarán algunas de las restricciones más importantes a considerar en el desarrollo del problema:

4.3.1 Rotaciones y posicionamiento de las cajas

Esta primera restricción es primordial para entender cómo es que las cajas serán colocadas dentro de los contenedores. Normalmente, una caja puede ser colocada de más de una manera como se puede ver en la siguiente imagen (Feng et al., 2013):

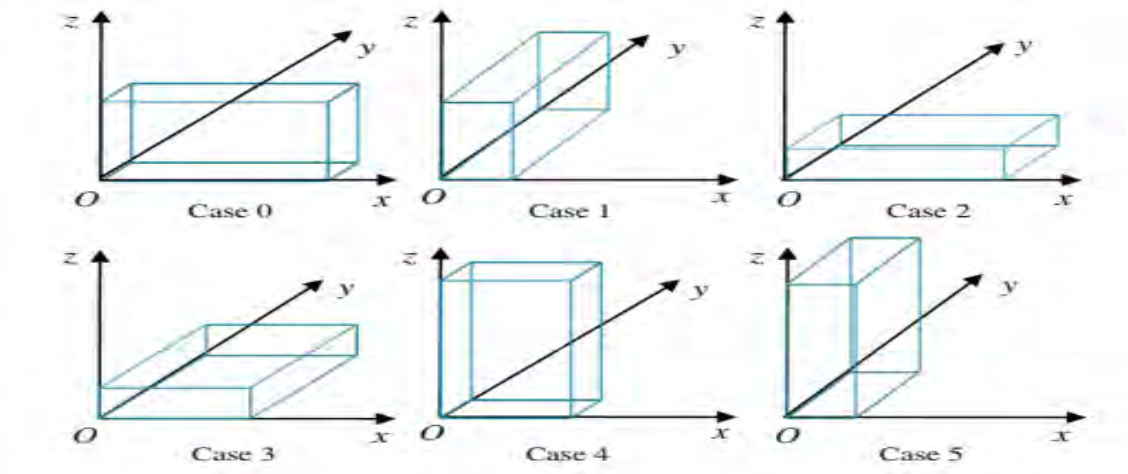


Ilustración 5: Rotaciones de una caja

4.3.2 Límite de las paredes del contenedor

Esta restricción indica que ninguna de las cajas colocadas puede sobrepasar los límites del contenedor, es decir, todas las cajas deben estar dentro de las paredes de este:

$$\nexists c_{ij} \mid X_{Cij} + Ancho_{Cij} > Ancho_{COj}$$

$$\nexists c_{ij} \mid Y_{Cij} + Largo_{Cij} > Largo_{COj}$$

$$\nexists c_{ij} \mid Z_{Cij} + Alto_{Cij} > Alto_{COj}$$

Donde:

- c_{ij}: Es la caja i colocada en el contenedor j.
- X_{Cij}: Es el punto del eje X del contenedor j en donde está colocada la caja i.
- Y_{Cij}: Es el punto del eje Y del contenedor j en donde está colocada la caja i.
- Z_{Cij}: Es el punto del eje Z del contenedor j en donde está colocada la caja i.
- $Ancho_{Cij}$: Es el ancho de la caja i colocada en el contenedor j.
- $Largo_{Cij}$: Es el largo de la caja i colocada en el contenedor j.
- $Alto_{Cij}$: Es el alto de la caja i colocada en el contenedor j.

4.3.3 Límite de peso

Esta restricción se relaciona directamente al factor de fragilidad intrínseco de las cajas. Se tiene que cada caja solo puede tener por encima de esta un límite de peso específico.

Capítulo 5. Estructura de datos para los algoritmos

En este capítulo se presentarán las estructuras de datos mencionadas en los objetivos que se utilizarán en ambos algoritmos. Cabe mencionar que estas serán las mismas para los dos, pero serán tratadas de diferente manera para cada uno de los algoritmos.

5.1 Contenedor

Dado que se trabajará con un estándar internacional, todos los contenedores usados tendrán las mismas dimensiones:

- Alto: Representa la altura del contenedor.
- Ancho: Representa el ancho del contenedor.
- Largo: Representa el largo del contenedor.
- Volumen: Representa el volumen del contenedor.

5.2 Caja

Esta estructura es la parte principal de la solución ya que no solo considerará las dimensiones de las cajas, si no que contendrá las coordenadas donde está colocada dentro del contenedor y la rotación que se ha utilizado:

- Alto: Representa la altura de la caja.
- Ancho: Representa el ancho de la caja.
- Largo: Representa el largo de la caja.
- Peso: Representa el peso de la caja.
- Volumen: Representa el volumen de la caja.
- Posición X: Representa el punto en el eje X donde se coloca la caja.
- Posición Y: Representa el punto en el eje Y donde se coloca la caja.
- Posición Z: Representa el punto en el eje Z donde se coloca la caja.
- Identificador: Representa un número único que identifica a la caja.
- Rotaciones: Representa una lista de rotaciones posibles que tiene la caja.

5.3 Rotación

Dado que cada caja se puede colocar en distintas posiciones en un mismo punto, las cuales son generadas a partir de rotar dicha caja sobre sus propios ejes, se tiene la necesidad de tener una estructura que maneje este concepto:

- Alto: Representa la altura de la caja en una determinada rotación.
- Ancho: Representa el ancho de la caja en una determinada rotación.
- Largo: Representa el largo de la caja en una determinada rotación.

5.4 Punto Extremo

Esta estructura representa un punto disponible en donde se puede colocar una caja y está representado por tres coordenadas:

- Posición X: Representa la posición en el eje X donde se coloca en punto extremo.
- Posición Y: Representa la posición en el eje Y donde se coloca en punto extremo.
- Posición Z: Representa la posición en el eje Z donde se coloca en punto extremo.

5.5 Solución

Como su propio nombre indica, esta estructura representará la solución generada, la cual contendrá una lista de todas las cajas colocadas en cada contenedor y los puntos extremos presentes en estos:

- Lista de cajas: Representa a todas las cajas colocadas en el contenedor.
- Lista de puntos extremos: Representa a todos los puntos extremos restantes luego de generar la solución.

Capítulo 6. Pseudocódigo del algoritmo genético

En este capítulo se presentará el pseudocódigo que se utilizará para adaptar el algoritmo genético. Se utilizará un lenguaje lo más parecido al lenguaje natural para que pueda ser legible para cualquier lector.

```
Entrada: TamPoblacion, Generaciones, ProbMutacion, ProbCrossOver
Salida: MejorSolucion
1. Poblacion = GenerarPoblacionInicial()
2. MejorSolucion = EncontrarMejorSolucion(Poblacion)
3. Mientras (CondicionDeParada)
   3.1 Padres = SeleccionarPadres(Poblacion, TamPoblacion)
   3.2 Hijos = ReproducirPadres(Padres, TamPoblacion,ProbMutacion,ProbCrossOver)
   3.3 Hijos_MejorSolucion = EncontrarMejorSolucion(Hijos)
   3.4 Si ( Fitness(Hijos_MejorSolucion) > Fitness(MejorSolucion.Fitness) )
      3.4.1 MejorSolucion = Hijos_MejorSolucion
   Fin Si
   3.5 Poblacion = Hijos
Fin Mientras
4. Devolver MejorSolucion
```

Dentro de este pseudocódigo se puede apreciar en la línea 1 la inicialización de la población inicial y la función que encuentra la mejor solución dentro de una población en la línea 2. Luego, entre las líneas 3 y 4 se hace referencia a la condición de parada del algoritmo y cuatro funciones. Para este algoritmo, la condición de parada contendrá dos validaciones, la primera será la cantidad de generaciones ingresada como dato de entrada, es decir, mientras que no se generen todas las generaciones solicitadas, el algoritmo debe seguir. La segunda condición verificará las veces que la mejor solución de la población evaluada se mantiene igual o apenas varía con respecto a la mejor solución de la población anterior. Si esto sucede más de un determinado número de veces, el algoritmo se detendrá y devolverá la mejor solución encontrada hasta el momento.

A continuación, se detallan las funciones mencionas en las líneas 1, 2, 3.1 y 3.2 del pseudocódigo:

1. GenerarPoblacionInicial

   Como su propio nombre lo indica, esta función inicializa la población que utilizará el algoritmo genético para comenzar con su proceso evolutivo. A continuación, se presente el pseudocódigo de la función:

```
Entrada: TamañoPoblacion, CantidadCajas
Salida: Poblacion
1. Poblacion = { }
2. Para i = 0 hasta i = TamañoPoblacion
  2.1 Cromosoma = { }
  2.2 Para j = 0 hasta j = CantidadCajas
    2.2.1 AñadirCaja(Cajaj, Cromosoma)
  Fin Para
  2.3 AñadirSolucion(Poblacion, Cromosoma)
Fin Para
3. Devolver Poblacion
```

   Dentro de esta función cabe mencionar que no se especificará el pseudocódigo de la función “AñadirSolucion” ya que su objetivo es añadir una solución a una lista de soluciones.

2. AñadirCaja

   Esta función es la que se encarga de verificar si la caja a colocar no se traslapa con las que ya están colocadas. Además, genera nuevos puntos extremos y verifica que se cumpla la restricción de la fragilidad.

```
Entrada: Caja, Cromosoma
Salida: { }
1. Si CajaNoColocada
  1.1 Para cada PuntoExtremo en PuntosExtremos
    1.1.1 Para cada Rotacion en Caja.Rotaciones
```

```
    1.1.1.1 Si CajaDentroDelContenedor
     1.1.1.1.1 Si CajaNoTraslapa y CajaNoSobrepasaPeso
      1.1.1.1.1.1 AgregarCaja(Caja,Cromosoma.CajasColocadas)
       1.1.1.1.1.2 GenerarPuntosExtremos(Caja,PuntoExtremo)
      1.1.1.1.1.3 Salir Para
      Fin Si
    Fin Si
   Fin Para
  Fin Para
Fin Si
```

Para esta función se explican las siguientes líneas específicas:

- Línea 1: Se verifica si la caja que se intenta colocar no se encuentra dentro del contenedor.
- Línea 1.1: Se itera por cada punto extremo que contiene la solución de tal manera que se busca un lugar en donde se pueda colocar la caja.
- Línea 1.1.1: Se itera por cada rotación que tiene la caja para determinar si alguna de estas se puede colocar.
- Línea 1.1.1.1: Se verifica si la caja no sobrepasa los límites del contenedor.
- Línea 1.1.1.1.1: Se verifica si la caja en el punto extremo determinado no se traslapa con otra y además no sobrepasa el límite de peso por encima de esta.
- Línea 1.1.1.1.1.1: Se agrega la caja a la lista de cajas colocadas del cromosoma.
- Línea 1.1.1.1.1.2: Se crean los puntos extremos que genera la colocación de la caja.
- Línea 1.1.1.1.1.3: Se fuerza la salida de las iteraciones porque la caja ya fue colocada.

Cabe mencionar que la función “GenerarPuntosExtremos” sigue los lineamientos mencionados en la sección 1.3.3 del presente documento como se muestra en la ilustración 1.

3. EncontrarMejorSolucion

   Esta función itera a través de todas las soluciones de una población para encontrar la que tiene el mejor fitness, es decir, la solución que maximiza la función objetivo.

```
Entrada: Poblacion
Salida: MejorSolucion
1. MejorSolucion = { }
2. Para cada Solucion en Poblacion
  2.1 Si (Fitness(Solucion) > Fitness(MejorSolucion) Entonces
    2.1.1 MejorSolucion = Solucion
  Fin Si
Fin Para
3. Devolver MejorSolucion
```

4. SeleccionarPadres

   En esta función se seleccionarán los padres de la siguiente generación mediante la técnica del torneo. Esta técnica consiste en seleccionar un determinado número de cromosomas de la población y escoger el que tenga el mejor fitness.

```
Entrada: Poblacion, TamañoPoblacion
Salida: Padres
1. Padres = { }
2. Para i = 0 hasta i = TamañoPoblacion
  2.1 Pareja = { }
  2.2 Pareja_1 = SeleccionarPadreTorneo(Poblacion)
  2.3 Pareja_2 = SeleccionarPadreTorneo(Poblacion)
  2.4 AñadirPadre(Pareja, Pareja_1)
  2.5 AñadirPadre(Pareja, Pareja_2)
  2.6 AñadirPareja(Padres, Pareja)
```

```
Fin Para
3. Devolver Padres
```

Dentro de esta función cabe mencionar que no se especificará el pseudocódigo de las funciones “AñadirPadre” y “AñadirPareja” ya que el objetivo de ambas es añadir una solución a una lista de soluciones y una pareja (lista de dos soluciones) a una lista de parejas respectivamente.

5. ReproducirPadres

Esta función consiste en aplicar las técnicas de cruce y mutación a la lista de padres entregada por la función anterior. El cruce consiste en seleccionar la mitad de las cajas de una solución y la mitad de otra de tal manera que se genere una nueva solución a partir de estas dos mitades, mientras que la mutación consiste en seleccionar una solución e intercambiar el orden de colocación de un par de las cajas colocadas en esta. Para ambas técnicas se maneja probabilidades, es decir, no en todas las reproducciones se producirán nuevas soluciones, de tal manera que se mantengan algunas soluciones de la generación pasada.

```
Entrada: Padres, TamañoPoblacion, ProbMutacion, ProbCruce
Salida: Hijos
1. Hijos = { }
2. Para cada Pareja en Padres
 2.1 Prob = Aleatorio()
 2.2 Si ( Prob <= ProbCruce ) Entonces
  2.2.1 Hijo_1 = RealizarCruce(Pareja)
  2.2.2 Si ( prob <= ProbMutacion ) Entonces
   2.2.2.1 Hijo_2 = RealizarMutacion(Pareja[1])
   2.2.2.2 AñadirHijo(Hijos,Hijo_2)
  Fin Si
  2.2.2 AñadirHijo(Hijos,Hijo_1)
 2.3 En otro caso
```

```
    2.3.1 AñadirHijo(Hijos,Pareja[1])
  Fin Si
Fin Para
3. Devolver Hijos
```

Dentro de esta función cabe mencionar que no se especificará el pseudocódigo de la función "AñadirHijo", "RealizarCruce" y "RealizarMutacion" ya que el objetivo de la primera es añadir una solución a una lista de soluciones, mientras que los funcionamientos de las últimas han sido explicados anteriormente.

6. Fitness

   Finalmente, esta función devuelve el valor calculado mediante el uso de la función objetivo en una solución. Como se detalló anteriormente, cada solución contiene una lista de las cajas colocadas en los contenedores. Esta lista es la que se utilizará para calcular el fitness de cada solución de tal manera que se pueda calcular el volumen de cada caja colocada en los contenedores y poder simular la función objetivo. Un ejemplo del cálculo de la función objetivo se presenta en el capítulo 4 anteriormente descrito.

A continuación, se presenta un ejemplo de ejecución para el pseudocódigo:

- Se consideran las siguientes cajas:

Identificador	Ancho	Largo	Alto	Volumen
1	20	25	15	7500
2	20	25	15	7500
3	40	50	60	120000
4	50	20	20	20000
5	30	40	50	60000

- Se considera un contenedor con las siguientes dimensiones:

Ancho	Largo	Alto	Volumen
120	150	100	1800000

- Como datos de entrada se ingresan:
  - TamPoblacion = 3
  - Generaciones = 1
  - ProbMutacion = 0.05
  - ProbCrossOver = 0.9
- Generación de la población inicial:
  - Considerando que las cajas no se colocan necesariamente en el mismo orden para cada solución. Se crean las tres soluciones de la siguiente manera:
    - Solución 1: contiene una lista con las cajas 1, 2 y 3

Caja 1	Caja 2	Caja 3

    - Solución 2: contiene una lista con las cajas 4 y 5

Caja 4	Caja 5

    - Solución 3: contiene una lista con las cajas 1, 2 y 4

Caja 1	Caja 2	Caja 4

- Se encuentra la mejor solución de la población inicial:
  - Se calcula el fitness de cada solución siguiendo los pasos del capítulo 4:
    - Solución 1: 0.075
    - Solución 2: 0.044
    - Solución 3: 0.019
  - Vemos que la mejor solución es la que tiene el mayor fitness, en este caso, la solución 1.
- Luego se seleccionan los padres en parejas:
  - Pareja 1: Solución 3 y Solución 1

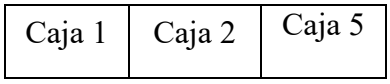
- Pareja 2: Solución 3 y Solución 2
- Pareja 3: Solución 2 y Solución 1

- A continuación, se reproducen las parejas:
  - Se reproduce la primera pareja:
    - Se genera un número aleatorio entre 0 y 1, en este caso 0.5
    - Se escoge la primera mitad de la solución 3, es decir, las cajas 1 y 2, se unen con la segunda mitad de la solución 1, en este caso, la caja 3 para generar el primer hijo:

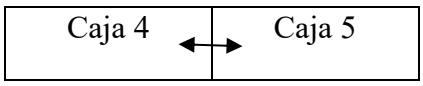
- Hijo 1: contiene las cajas 1, 2 y 3.

  - Se reproduce la segunda pareja:
    - Se genera un número aleatorio entre 0 y 1, en este caso 0.1
    - Se escoge la primera mitad de la solución 3, es decir, las cajas 1 y 2, se unen con la segunda mitad de la solución 2, en este caso, la caja 5 para generar el segundo hijo.

- Hijo 2: contiene las cajas 1, 2 y 5.



- Se reproduce la tercera pareja:
  - Se genera un número aleatorio entre 0 y 1, en este caso 0.01
  - En este caso se procede a realizar la mutación de la solución 2 de la tercera pareja y se procede a cambiar el orden de un par de sus cajas. En este caso la mutación quedaría de la siguiente manera:



- Hijo 3: contiene las cajas 5 y 4.

  - Finalmente se devuelven los hijos generados.
- Se encuentra la mejor solución entre los hijos obtenidos:
  - Se calcula el fitness de cada hijo:
    - Hijo 1: 0.075
    - Hijo 2: 0.041
    - Hijo 3: 0.019
  - Vemos que la mejor solución es el Hijo 1.
- Se compara la mejor solución encontrada hasta ahora con la mejor solución encontrada en la nueva generación. Vemos que en este caso tienen el mismo fitness por lo que la solución se mantiene.
- Finalmente se devuelve la mejor solución encontrada la cual contiene las cajas 1, 2 y 3.

Para finalizar el capítulo, es necesario resaltar que para el ejemplo de ejecución presentado no se explica cómo se manejan las restricciones de peso y fragilidad mencionadas en los primeros capítulos de este documento. Esto es debido a que se necesita estar en una fase avanzada del algoritmo para empezar a colocar cajas encima de otras. Con la finalidad de completar el ejemplo de ejecución, se presenta el siguiente pseudocódigo:

```
Entrada: CajasColocadas, CajaAColocar, PosicionAColocar

Salida: CumpleRestriccion
```

```
1. CumpleRestriccion = Falso
2. Para i = 0 hasta i = Cantidad(CajasColocadas)
  2.1 Si EstaEncima(CajasColocadas[1],CajaAColocar,PosicionAColocar)
    2.1.1 PesoEncima = SumaPesosCajasEncima(CajasColocadas[1])
    2.1.2 Si (PesoEncima + Peso(CajaAColocar)) < Fragilidad(CajasColocadas[i]))
      2.1.2.1 CumpleRestriccion = Verdadero
      2.1.2.2 Salir Para
    2.1.3 En Otro Caso
      2.1.3.1 CumpleRestriccion = Falso
    2.1.4 Fin Si
  2.2 Fin Si
3. Devolver CumpleRestriccion
```

En este pseudocódigo se debe tener en consideración que "Fragilidad(CajasColocadas[i]))" en la línea 2.1.2 hace referencia al factor de fragilidad que cada caja tendrá con respecto a su peso propio. Este factor será definido para cada caja como un porcentaje de su peso que podrá soportar por encima de esta antes de sufrir daños o romperse.

Finalmente, cabe mencionar que este pseudocódigo aplica también para el algoritmo de búsqueda tabú ya que las restricciones serán las mismas.

Capítulo 7. Pseudocódigo del algoritmo de búsqueda tabú

Al igual que el capítulo anterior, se presentará el pseudocódigo que se utilizará para diseñar el algoritmo de búsqueda tabú.

```
Entrada: Iteraciones, SolucionInicial, TamañoLista, TamañoVecindario
Salida: Solucion
1. MejorSolucion = SolucionInicial
2. Iteracion = 0
```

```
3. ListaTabu = { }
4. Mientras (CondicionDeParada)
   4.1 Vencidario = GenerarVecindario(MejorSolucion,Iteracion)
   4.2 MejorVecino = EncontrarMejorSolucion(Vecindario)
   4.3 Si ( ListaTabuNoContiene(MejorVecino) Y Fitness(MejorVecino) >
Fitness(MejorSolucion) )
      4.3.1 MejorSolucion = MejorVecino
   Fin Si
   4.4 Si ( TamañoListaTabu == TamañoLista )
      4.4.1 QuitarUltimo(ListaTabu)
   Fin Si
   4.5 AñadirListaTabu(MejorVecino)
   4.6 Iteracion = Iteracion + 1
Fin Mientras
5. Devolver MejorSolucion
```

Dentro de este pseudocódigo se pueden apreciar cuatro funciones (línea 4.1, línea 4.2, línea 4.4.1 y línea 4.5), pero solo dos de estas, “GenerarVecindario” y “QuitarUltimo”, serán especificadas ya que la función “EncontrarMejorSolucion” se detalló en el capítulo anterior y “AñadirListaTabu” solo agrega una solución a una lista de soluciones llamada “ListaTabu”:

1. GenerarVecindario

   Esta función es la principal dentro del proceso del algoritmo de búsqueda tabú ya que es la que provee de nuevas posibles soluciones hasta que se llegue a la condición de parada. Dentro de esta función se consideran dos formas de generar un nuevo vecindario, la primera es mediante la permutación del orden de colocación de las cajas de un determinado número de soluciones del vecindario inicial. Mediante las permutaciones se logra generar un vecindario no tan lejano al inicial de tal formar que se explore más detalladamente las cercanías del vecindario inicial. La segunda forma es mediante combinaciones entre las cajas colocadas y las no colocadas en una solución, es decir, se sacan algunas cajas colocadas para poner otra que aún no habían sido tomadas en cuenta, de esta

manera se logra aumentar la variedad en los vecindarios para poder escapar de óptimos locales.

```
Entrada: Solucion, Iteracion
Salida: Vecindario
1. Vecindario = { }
2. CajasNoColocadas = ObtenerCajasNoColocadas(Solucion)
3. Si (CondicionCombinacion Y Cantidad(CajasNoColocadas) > 0)
  3.1 Para i = 1 Hasta i = TamañoVecindario
    3.1.1
Intercambiar(CajaAleatoria(Solucion),CajaAleatoria(CajasNoColocadas))
    3.1.2 AñadirVecino(Vecindario, Solucion)
  Fin Para
4. En Otro Caso
  4.1 Para i = 1 Hasta i = TamañoVecindario
    4.1.1 Intercambiar(CajaAleatoria(Solucion),CajaAleatoria(Solucion))
    4.1.2 AñadirVecino(Vecindario, Solucion)
  Fin Para
Fin Si
5. Devolver Vecindario
```

Dado que las funciones utilizadas dentro de esta función se pueden comprender fácilmente, no se indicarán sus pseudocódigos, sin embargo, se procederá a indicar brevemente su funcionamiento:

- Primero, en la línea 2, “ObtenerCajasNoColocadas” encuentra las cajas que no han podido ser colocadas en la solución actual para poder realizar la combinación.
- Segundo, en la línea 3.1.1, “Intercambiar” cambiar el orden de colocación de una caja con otra.
- Tercero, en la línea 4.1.2, “AñadirVecino” adiciona el vecino o solución generada al nuevo vecindario.
- Finalmente, cabe mencionar que tanto “CondicionCombinacion” como “TamañoVecindario”, en las líneas 3 y 3.1 respectivamente, son

parámetros ingresados como variables globales para todo el algoritmo, estos parámetros indican la condición que se tiene que cumplir para realizar la combinación y la cantidad de vecinos que se generarán por vecindario respectivamente.

Para finalizar este capítulo se presenta un ejemplo de ejecución para el pseudocódigo:

- Se consideran las siguientes cajas:

Identificador	Ancho	Largo	Alto	Volumen
1	20	25	15	7500
2	20	25	15	7500
3	40	50	60	120000
4	50	20	20	20000
5	30	40	50	60000

- Se considera un contenedor con las siguientes dimensiones:

Ancho	Largo	Alto	Volumen
120	150	100	1800000

- Como datos de entrada se ingresan:
  - Iteraciones: 1
  - SolucionIncial: Será la solución obtenido en el pseudocódigo del algoritmo genético, en este caso contiene las cajas 1, 2 y 3.
  - TamañoLista: 1
  - TamañoVecindario: 3
- Generación del vecindario:
  - Se encuentran las cajas que no han sido colocadas en esta solución, en este caso, las cajas 4 y 5.
  - Se verifica si se cumple la condición de combinación. Con la finalidad de mostrar este caso particular del algoritmo se considera que sí se cumple.

Para otros casos, esta condición será indicada por el número de iteración en el que se encuentra en ese momento.

- Como se cumplen las condiciones antes mencionadas, se proceden a generar las 3 soluciones del nuevo vecindario:
  - Se genera el primer vecino, en este caso, se intercambia la caja 1 con la caja 4.
    - Vecino 1: contiene las cajas 2, 3 y 4.
  - Se genera el segundo vecino, en este caso, se intercambia la caja 2 con la 5.
    - Vecino 2: contiene las cajas 1, 3 y 5.
  - Se genera el tercer vecino, en este caso, se intercambia la caja 3 con la 5.
    - Vecino 3: contiene las cajas 1, 2 y 5.

- Se encuentra el mejor vecino del nuevo vecindario generado:
  - Vecino 1: 0.0819
  - Vecino 2: 0.1042
  - Vecino 3: 0.0417
- Vemos que el mejor vecino es el número 2 con las cajas 1, 3 y 5.
- Verificamos si la lista tabú no contiene a este vecino y si su fitness es mejor que la mejor solución encontrada hasta ahora. Vemos que, efectivamente, se cumples estas dos condiciones ya que la lista tabú está vacía y que 0.1042 es mayor que 0.075. Por lo tanto, se cambia la mejor solución por este nuevo vecino. El estado de la lista tabú se representa como una lista de soluciones vacía de tamaño 1.

Ø

- Vemos que todavía hay espacio en la lista tabú, por lo que esta nueva solución es agregada.

Caja 1	Caja 3	Caja 5

- Finalmente se aumenta la iteración y se llega a la condición de parada por lo que la mejor solución es entregada, en este caso, contiene las cajas 1, 3 y 5.

Capítulo 8. Programa e interfaz gráfica

En este capítulo se presentará la interfaz del programa que se utilizará para ejecutar los algoritmos. Esta interfaz será la que permitirá ingresar los datos de entrada necesarios para los algoritmos y mostrará los resultados devueltos por estos.

8.1 Ingreso de parámetros

Esta primera pantalla es la que permitirá ingresar los siguientes parámetros:

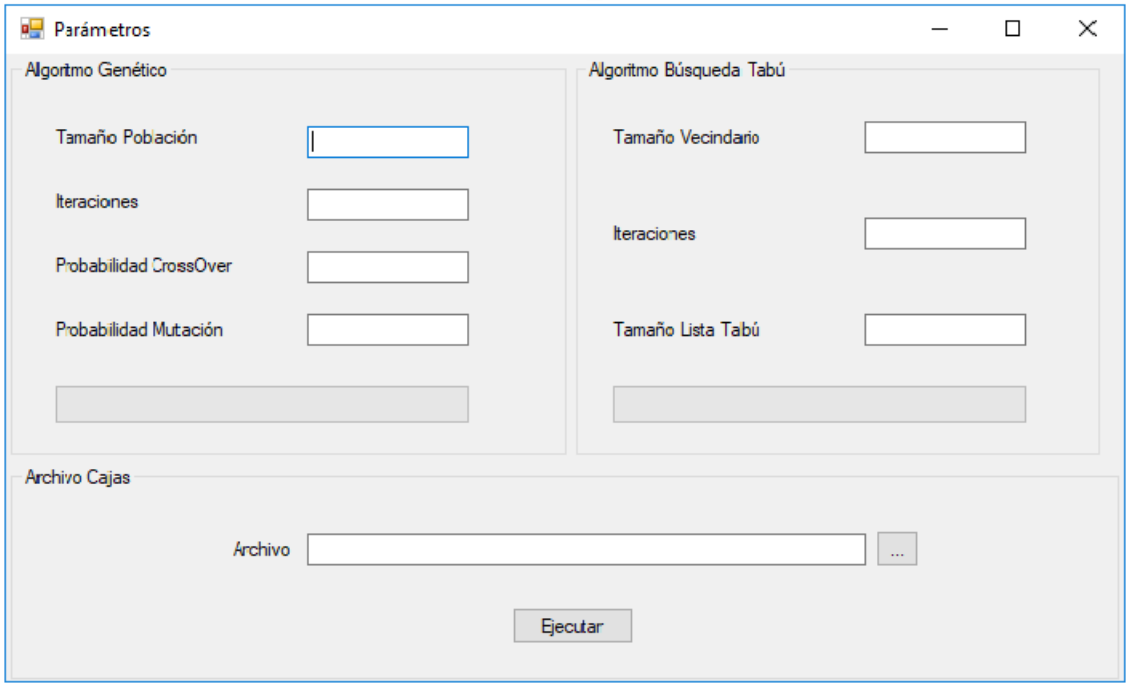


Ilustración 6. Pantalla de ingreso de parámetros

- Para el algoritmo genético:
  - Tamaño de la población: Indica la cantidad de cromosomas que tendrá la población inicial.
  - Iteraciones: Indica la cantidad de iteraciones que realizará el algoritmo.
  - Probabilidad CrossOver: Indica la probabilidad de que se realice un cruce entre cromosomas. Se representa con un número entre 0 y 1.
  - Probabilidad Mutación. Indica la probabilidad de que se realice una mutación de un cromosoma. Se representa con un número entre 0 y 1.
- Para el algoritmo de búsqueda tabú:
  - Tamaño Vecindario: Indica la cantidad de soluciones que tendrá el vecindario.

- Iteraciones: Indica la cantidad de iteraciones que realizará el algoritmo.
- Tamaño Lista Tabú: Indica la cantidad de soluciones que podrán ser marcadas como tabú y serán añadidas en la lista tabú.

- Archivo: Permite seleccionar un archivo de texto que contenga las dimensiones de los contenedores y de las cajas que se desean colocar en estos.

Finalmente, contiene un botón "Ejecutar" para poder comenzar con la ejecución de los algoritmos.

8.2 Resultados

Esta pantalla permitirá ver los resultados otorgados por los algoritmos de tal manera que se puedan apreciar las diferencias y discernir cuál tuvo una mejor performance.

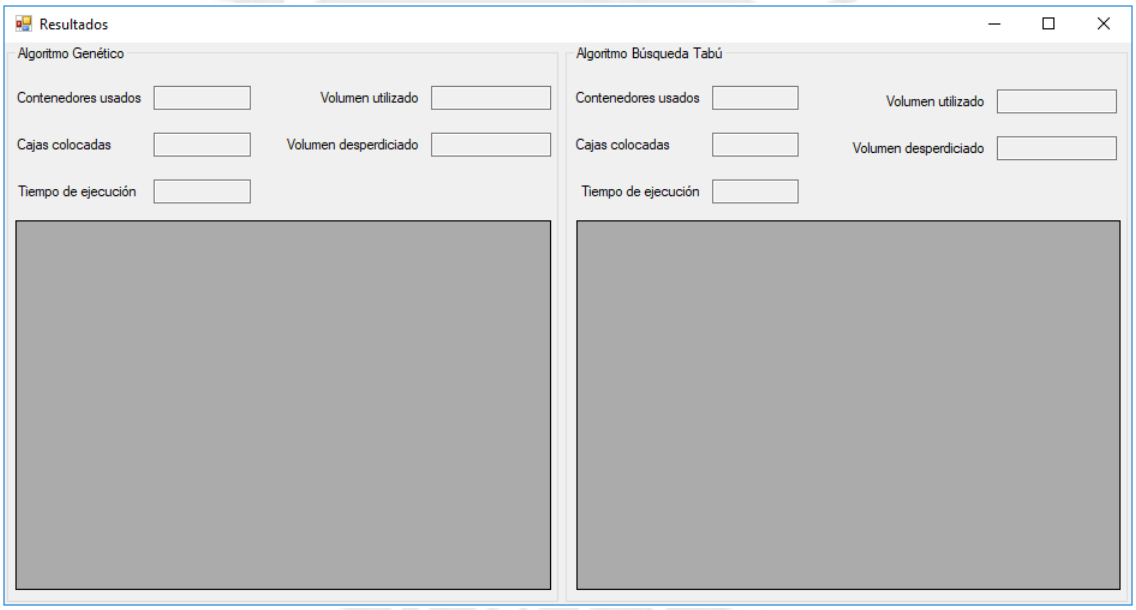


Ilustración 7. Pantalla de resultados

Los resultados que se mostrarán para ambos algoritmos son:

- Contenedores usados: Indica la cantidad total de contenedores que utiliza la solución brindada.
- Cajas colocadas: Indica la cantidad de cajas que se colocaron en total. También se muestra una grilla con el detalle de las cajas colocadas en la que se indica la posición en la que se coloca cada caja y el contenedor en la que está.

- Volumen utilizado: Indica el volumen total utilizado por las cajas dentro de los contenedores.
- Volumen desperdiciado: Indica el volumen total sin utilizar dentro de los contenedores.
- Tiempo de ejecución: Indica el tiempo que tardó el algoritmo en entregar la solución.

Capítulo 9. Experimentación numérica

9.1 Introducción

En este capítulo se presentan los resultados obtenidos mediante la experimentación numérica utilizando R Studio, la cual fue aplicada a los resultados brindados por los algoritmos de tal manera que se pueda discernir cuál de los dos brinda mejores soluciones basadas en la función objetivo. Con este fin, se realizará la prueba Z para comparar las medias de los resultados de ambos algoritmos, cabe mencionar que para poder realizar la prueba Z es necesario comprobar que los datos tengan una distribución normal y que sus varianzas sean homogéneas, para esto se realizarán las pruebas Kolmogorov-Smirnov y F de Fisher respectivamente.

Con el propósito de obtener los resultados, se ejecutaron ambos algoritmos 55 veces con diferentes muestras. Cabe mencionar que, como se trabajan con contenedores de tamaños estándar, se realizaron las pruebas con contenedores del mismo tamaño.

9.2 Estructura de datos de prueba

Los archivos de texto que se usarán como entrada para los algoritmos tienen la siguiente estructura:

- Cantidad de muestras.
- Dimensiones del contenedor: Ancho, largo y alto.
- Cantidad de tipos de cajas.
- Dimensiones de las cajas y cantidad de cajas de este tipo: Ancho, largo, alto, peso y cantidad.

Por ejemplo, se tendrán archivos de la siguiente manera:

```
100
587 233 220
70
100 76 30 10 1
110 43 25 7 1
```

92 81 55 5 1

81 33 28 5 3

9.3 Resultados de los algoritmos

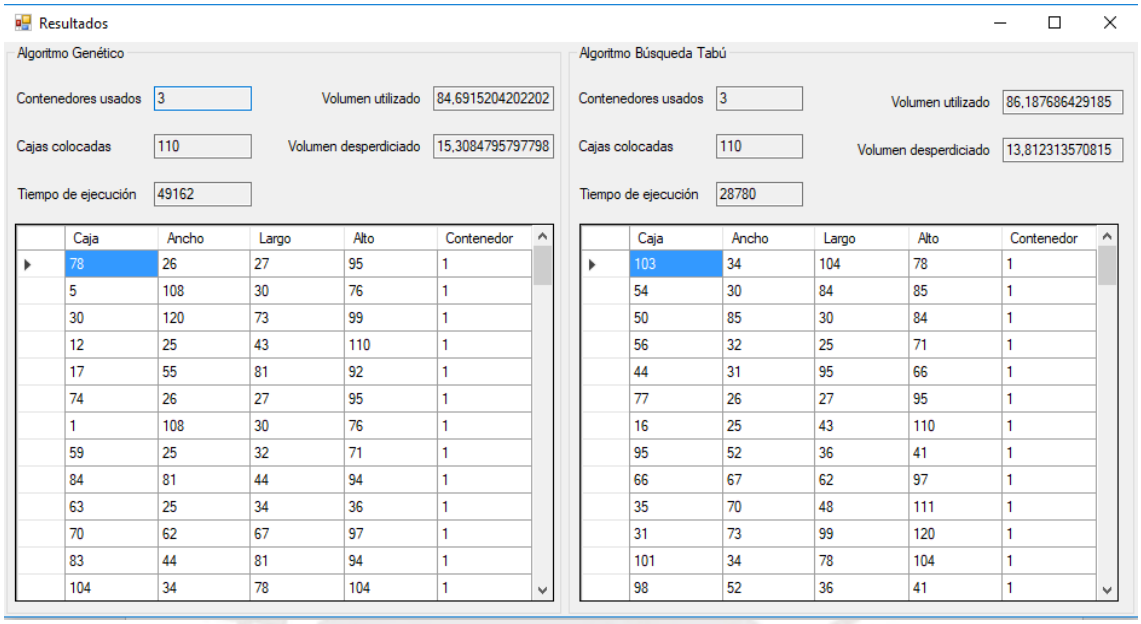
A continuación, se presentan los resultados obtenidos luego de la ejecución de los algoritmos. Estos resultados contienen los valores fitness de cada solución obtenida, el cual representa la función objetivo mencionada en capítulos anteriores.

Tabla 2. Resultados de los algoritmos

Muestra	Resultados del algoritmo genético	Resultados del algoritmo de búsqueda tabú
1	0.862351701350831	0.868905212716766
2	0.859659925914651	0.859659925914651
3	0.852777447283593	0.852777447283593
4	0.84968569449088	0.84968569449088
5	0.848484693392605	0.848484693392605
6	0.847555743475657	0.853234030539435
7	0.846907023086367	0.852388747348753
8	0.846289894433141	0.846289894433141
9	0.843971608813937	0.843971608813937
10	0.841955653367064	0.841955653367064
11	0.841003121120617	0.845155704857688
12	0.840696957954271	0.843458541516975
13	0.840477014997198	0.840477014997198
14	0.840178107932237	0.840178107932237
15	0.840013001161198	0.841342529417121
16	0.839739263351725	0.843318415675129
17	0.839335901660883	0.872697777505997
18	0.839181274694286	0.844430039329177
19	0.839106010644202	0.839106010644202
20	0.838821737640201	0.838821737640201
21	0.838743194496973	0.848570362470513
22	0.838610380146598	0.838610380146598
23	0.838360825649067	0.841123372999282
24	0.838320390885628	0.838320390885628
25	0.837857149741339	0.845669846722336

26	0.837607010878392	0.842349033764246
27	0.837453369855341	0.837453369855341
28	0.836870289488535	0.836971786283775
29	0.836569676408897	0.839018173044392
30	0.836426227161836	0.836426227161836
31	0.836105817665139	0.843431533310601
32	0.836059301070159	0.836059301070159
33	0.835525462490609	0.83833268748492
34	0.835370126530899	0.849511575752701
35	0.83512035047302	0.844078445213554
36	0.834956152099406	0.834956152099406
37	0.834297508576047	0.836090009334337
38	0.83361432724419	0.83361432724419
39	0.832916179510852	0.832916179510852
40	0.832780850450532	0.848074186380553
41	0.832726822959767	0.834537790773031
42	0.832638619785383	0.837562300000244
43	0.83252694229217	0.844192205374035
44	0.832470699197929	0.851079142907089
45	0.83239575641035	0.835965758291398
46	0.832312405407579	0.83611291867428
47	0.83033504577326	0.839219194747779
48	0.829997609363849	0.829997609363849
49	0.828858899957305	0.831294823043074
50	0.828111499358693	0.836875662326964
51	0.827500723948436	0.836880691746855
52	0.826167717195055	0.832074139409759
53	0.823679361853024	0.824405160317744
54	0.821020272107125	0.835046149912606
55	0.820310913420198	0.832740327062954

Estos resultados se visualizan dentro de la interfaz creada de la siguiente manera:



Caja	Ancho	Largo	Alto	Contenedor
78	26	27	95	1
5	108	30	76	1
30	120	73	99	1
12	25	43	110	1
17	55	81	92	1
74	26	27	95	1
1	108	30	76	1
59	25	32	71	1
84	81	44	94	1
63	25	34	36	1
70	62	67	97	1
83	44	81	94	1
104	34	78	104	1

Caja	Ancho	Largo	Alto	Contenedor
103	34	104	78	1
54	30	84	85	1
50	85	30	84	1
56	32	25	71	1
44	31	95	66	1
77	26	27	95	1
16	25	43	110	1
95	52	36	41	1
66	67	62	97	1
35	70	48	111	1
31	73	99	120	1
101	34	78	104	1
98	52	36	41	1

Ilustración 8. Interfaz de resultados de los algoritmos

Donde los volúmenes están representados en porcentaje y el tiempo de ejecución en milisegundos. Así mismo, se indican todas las cajas y sus respectivas medidas junto con el contenedor en el que se colocaron.

9.4 Prueba Kolmogorov-Smirnov

Como se mencionó en la introducción, esta prueba se utilizará para probar que los resultados de los algoritmos, es decir, los fitness de cada solución encontrada siguen una distribución normal ya que es requisito indispensable para poder realizar la prueba Z. Normalmente, las hipótesis para esta prueba son que los datos analizados siguen una distribución determinada, para este caso se tienen las siguientes hipótesis:

- H_0: Los resultados del algoritmo siguen una distribución normal.
- H_1: Los resultados del algoritmo no siguen una distribución normal.

Estas hipótesis se plantearán para ambos algoritmos de tal manera que se tenga prueba suficiente para concluir que alguna de las dos hipótesis es cierta.

9.4.1 Prueba para el algoritmo genético

Con la finalidad de comprobar que los resultados del algoritmo genético siguen una distribución normal se calcularon los siguientes valores:

- Media de los resultados: 0.8373238

- Desviación estándar de los resultados: 0.008056261

Luego, con ayuda de R Studio, considerando una significancia de 5% se obtuvo el resultado de la prueba Kolmogorov-Smirnov:

- P-valor: 0.1968

Como el P-valor obtenido es mayor que 0.05, se acepta la hipótesis nula y se concluye que los resultados del algoritmo genético siguen una distribución normal.

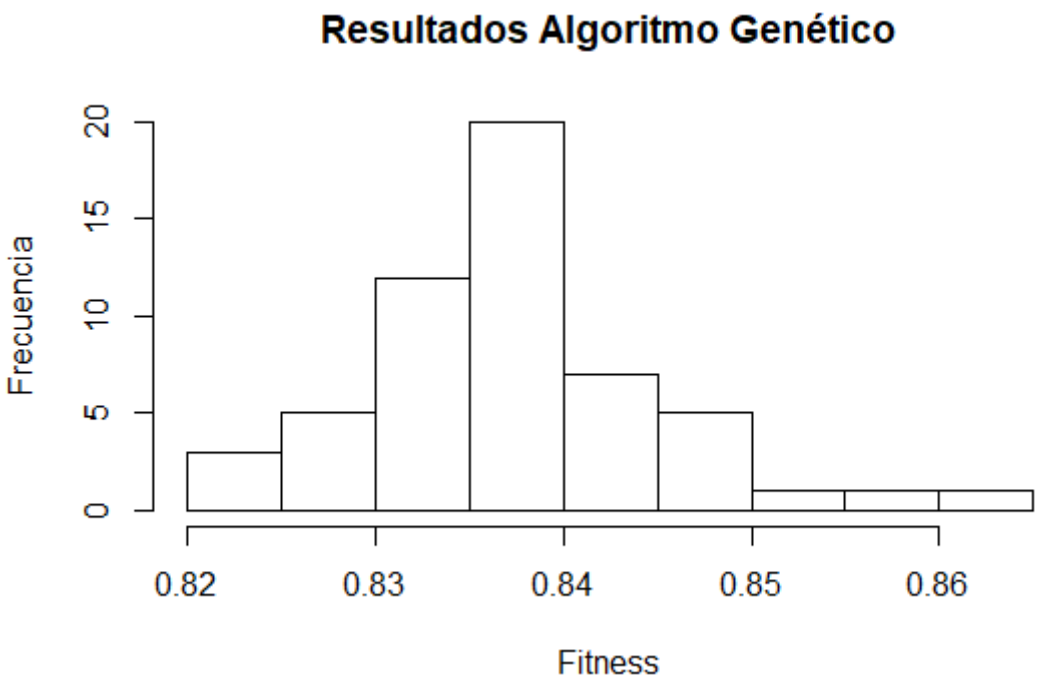


Ilustración 9. Resultados del Algoritmo Genético

9.4.2 Prueba para el algoritmo de búsqueda tabú

Con la finalidad de comprobar que los resultados del algoritmo genético siguen una distribución normal se calcularon los siguientes valores:

- Media de los resultados: 0.8419255
- Desviación estándar de los resultados: 0.008761028

Luego, considerando una significancia de 5% se obtuvo el resultado de la prueba Kolmogorov-Smirnov:

- P-valor: 0.4315

Como el P-valor obtenido es mayor que 0.05, se acepta la hipótesis nula y se concluye que los resultados del algoritmo de búsqueda tabú siguen una distribución normal.

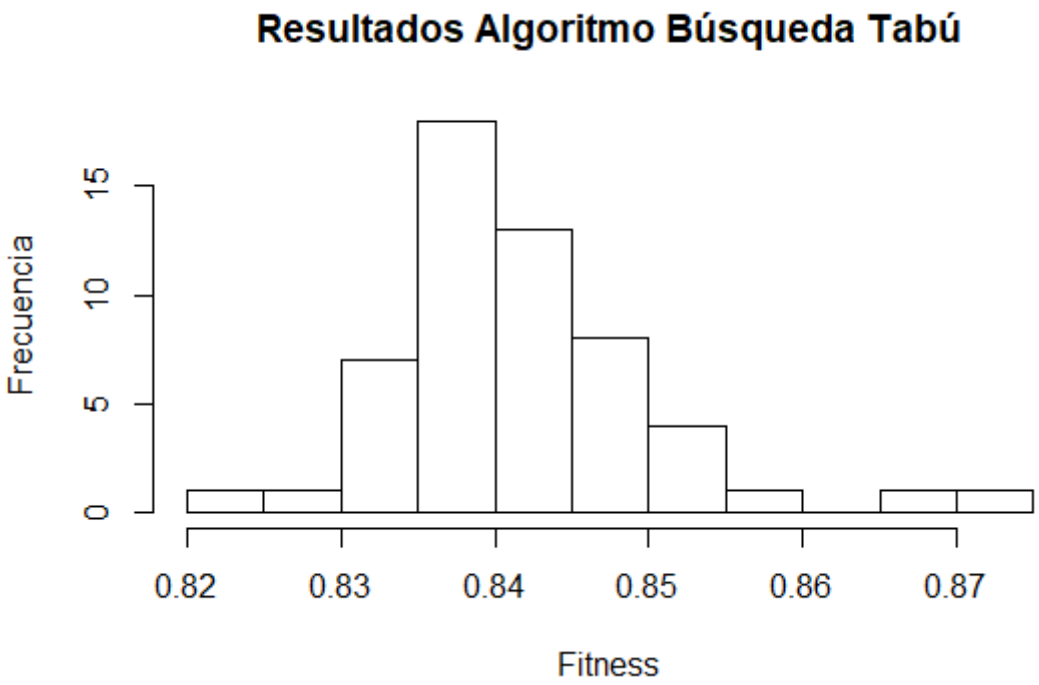


Ilustración 10. Resultados del Algoritmo de Búsqueda Tabú

9.5 Prueba F de Fisher

La finalidad de esta prueba estadística es verificar que las varianzas de los resultados de los algoritmos sean homogéneas ya que es necesario asegurar esto para poder realizar la prueba Z. Esta prueba puede ser aplicada a diferentes estadísticos, sin embargo, su uso más común es para verificar que las varianzas de las muestras tomadas, en este caso, de los resultados de los algoritmos sean considerablemente homogéneas o en su defecto, considerablemente diferentes. Por lo tanto, se tienen las siguientes hipótesis para ambos algoritmos:

- H_0: Las varianzas de los resultados de los algoritmos son considerablemente homogéneas.
- H_1: Las varianzas de los resultados de los algoritmos son considerablemente diferentes.

Con la finalidad de comprobar que las varianzas de los resultados de los algoritmos sean homogéneas se realizó la prueba F de Fisher. En esta prueba se obtuvo el siguiente resultado:

- P-valor: 0.5398

Como el P-valor obtenido es mayor que 0.05 se acepta la hipótesis nula, concluyendo que las varianzas de los resultados de los algoritmos son considerablemente homogéneas.

9.6 Prueba Z

Dado que esta prueba se utilizará para comparar las medias de los resultados de ambos algoritmos, es ideal para poder discernir cuál de los dos brinda mejores soluciones y llegar a una conclusión.

Para poder comparar ambas medias, primero se requiere demostrar que estas son considerablemente diferentes. Luego de comprobarlo, se puede hacer la prueba para verificar cuál de las dos medias es menor que la otra y así, finalmente, concluir cuál algoritmo tuvo un mejor desempeño. Para la primera prueba (de dos colas) se tienen las siguientes hipótesis:

- H_0: Las medias de los resultados de los algoritmos son considerablemente homogéneas.
- H_1: Las medias de los resultados de los algoritmos son considerablemente diferentes.

Para la segunda prueba (de una cola) se tienen las siguientes hipótesis:

- H_0: La media de los resultados del algoritmo de búsqueda tabú es menor que la media de los resultados del algoritmo genético.
- H_1: La media de los resultados del algoritmo de búsqueda tabú es mayor que la media de los resultados del algoritmo genético.

Tabla 3. Resultados de la prueba Z

	Algoritmo Genético	Algoritmo Búsqueda Tabú
Media	0,837323812	0,841925455
Varianza (conocida)	6,49034E-05	7,67556E-05
Observaciones	55	55
Diferencia hipotética de las medias	0	
Z	-2,86729349	
P(Z<=z) una cola	0,002069994	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,004139989	
Valor crítico de z (dos colas)	1,959963985	

9.6.1 Resultados para la prueba de dos colas

Con la finalidad de aceptar o rechazar la hipótesis nula de la prueba de dos colas, se comprueba si el valor de "Z" hallado está entre -1,959963985 y +1,959963985 (valores que representan el intervalo fuera de la zona crítica). En este caso, el valor "Z" se encuentra en la zona crítica, por lo que se puede afirmar que se tiene prueba suficiente para rechazar la hipótesis nula, en otras palabras, las medias de los resultados de los algoritmos genético y búsqueda tabú son considerablemente diferentes.

9.6.2 Resultados para la prueba de una cola

Gracias a los resultados de la prueba de dos colas, se puede continuar con la prueba de una cola. Para esta prueba, el valor de "Z" se debe comparar con el valor crítico de una cola, en este caso, -1,644853627. Ya que el valor de "Z" hallado es -2,86729349, este se encuentra en la zona crítica por que le se puede concluir que se cuenta con prueba suficiente para rechazar la hipótesis nula, en otras palabras, la media de los resultados del algoritmo de búsqueda tabú es mayor a la del algoritmo genético.

Finalmente se puede concluir que el algoritmo de búsqueda tabú brinda mejores resultados que el algoritmo genético para optimizar el espacio utilizado en el llenado de contenedores.

Capítulo 10. Conclusiones y trabajos futuros

10.1 Conclusiones

En este capítulo se presentan las conclusiones a las que se llegaron luego de finalizar todos los objetivos específicos y sus respectivos resultados esperados para brindar una solución al problema planteado.

En primer lugar, la función objetivo planteada se pudo acoplar fácilmente al desarrollo de los algoritmos ya que las soluciones brindadas por estos contienen todos los elementos que forman para de la función. Así mismo, se pudo observar que la función objetivo cumplió con su función de discernir si una solución es mejor que otra.

En segundo lugar, ambos algoritmos se pudieron adaptar correctamente al problema de optimización del espacio utilizado en los contenedores. Tanto el algoritmo genético como el algoritmo de búsqueda tabú brindaron las soluciones esperadas.

En tercer lugar, si bien la interfaz gráfica no permite una visualización gráfica del resultado final, cumple con su función de señalar cuál caja y en cuál contenedor debe ser colocada.

En cuarto lugar, gracias a los resultados de la experimentación numérica, se puede observar que el algoritmo de búsqueda tabú brinda mejores resultados que el algoritmo genético. Esto gracias a que el modo de trabajo de del algoritmo de búsqueda tabú hace énfasis en seguir buscando siempre por el óptimo global evitando reincidir en óptimos locales. Por otro lado, el algoritmo genético solo se basa en heredar las mejores soluciones y tratar de mejorarlas iteración tras iteración, lo cual puede terminar en un óptimo local. Sin embargo, como se puede apreciar en los resultados mostrados en la tabla 2 (página 53), los resultados del algoritmo de búsqueda tabú no son diferentes a los del algoritmo genético en gran medida, esto es debido a que las medidas de las cajas que se introducen en los contenedores no varían considerablemente en tamaño por lo que las soluciones halladas difícilmente serán muy distantes.

Finalmente, es necesario mencionar que, si bien este proyecto de fin de carrera orienta la solución del problema solamente a los contenedores, los algoritmos propuestos se

podrían aplicar a otro tipo de depósitos como almacenes, estanterías, vagones de tren, etc.

10.2 Trabajos futuros

En este apartado se plantean algunos trabajos futuros que puedan complementar o mejorar el presente proyecto de fin de carrera.

En primer lugar, se plantea la posibilidad de desarrollar una interfaz gráfica que muestre de mejor manera cómo sería el resultado final propuesto por el algoritmo. Esta interfaz serviría a las personas de tal manera que tengan un apoyo visual de cómo y dónde colocar cada caja dentro del contenedor.

En segundo lugar, se propone que la solución inicial del algoritmo de búsqueda tabú no sea la generada por el algoritmo genético, si no que se plantee una fase previa en la que se genere según una heurística nueva.

Finalmente, se propone que se añada una fase en la generación de las soluciones en donde se acomoden las cajas colocadas de tal manera que se reduzcan los espacios vacíos generados luego de su colocación. Una manera de lograr esto puede ser desplazando las cajas hacía la parte trasera del contenedor.

Capítulo 11. Referencias

[1] Tiba Group. (2016). Historia del contenedor marítimo. Tibagroup.com. Recuperado el 5 de setiembre de 2017 a partir de: http://www.tibagroup.com/mx/mclean-y-la-caja-que-cambio-la-historia-del-comercio.

[2] Organisation for EconomicCo-operation and Development. (2002). OECD Glossary of Statistical Terms - Twenty foot equivalent unit (TEU) Definition. Stats.oecd.org. Recuperado el 5 de setiembre de 2017 a partir de: https://stats.oecd.org/glossary/detail.asp?ID=4313.

[3] Glover, F. (1989). Tabu Search-- Part I. ORSA Journal On Computing, 1(3), 190.

[4] Garey, M.R., & Johnson, D.S. (1979). Computers and Intractability: a Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York.

[5] Afza, N., Zulkipli, F., Sarah, S., & Radiah, S. (2014). An Alternative Heuristics for Bin Packing Problem. Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management, Bali, Indonesia, Enero 7–9.

[6] Kacprzak, Ł., Rudy, J., & Żelazny, D. (2015). Multicriteria 3-dimension Bin packing Problem. Research in logistics and production, Vol. 5.

[7] UNECE (1972). Convenio Aduanero Sobre Contenedores. Recuperado el 22 de agosto de 2017 a partir de https://www.unece.org/fileadmin/DAM/trans/conventn/ccc_1972s.pdf.

[8] Guia_Transporte_Acuatico_13072015.pdf. (2015). Recuperado el 22 de agosto de 2017 a partir de https://www.mincetur.gob.pe/wp-content/uploads/documentos/comercio_exterior/facilitacion_comercio_exterior/Guia_Transporte_Acuatico_13072015.pdf.

[9] Why Logistics is So Important to Supply Chains. (2016, julio 19). Recuperado el 4 de septiembre de 2017, a partir de https://www.aacb.com/why-logistics-is-so-important-to-supply-chains/.

[10] Martello, S., Pisinger, D., & Vigo, D. (2000, 04). The Three-Dimensional Bin Packing Problem. Operations Research, 48(2), 256-267. doi:10.1287/opre.48.2.256.12386

[11] Cook, S. A. (1971, May). The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing . ACM.

[12] Holland, J. H. (1975). Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. Ann Arbor, MI: University of Michigan Press.

[13] Glover, F. (1989). Tabu search—part I. ORSA Journal on computing, 1(3), 190-206.

[14] Papadimitriou, C. H., & Steiglitz, K. (1998). Combinatorial optimization: algorithms and complexity. Courier Corporation.

[15] Brownlee, J. (2011). Clever Algorithms: Nature-Inspired Programming Recipes

[16] Aapa-ports.org. (2017). Port Industry Statistics. Recuperado el 1 de setiembre de 2017 a partir de: http://www.aapa-ports.org/unifying/content.aspx?ItemNumber=21048.

[17] Pino, R. & Fuente, D. & Quesada, I. & García, N. (2017). Optimización del Llenado de Contenedores para Transporte Multimodal.

[18] Hifi, M., Negre, S., & Wu, L. (2014). Hybrid greedy heuristics based on linear programming for the three-dimensional single bin-size bin packing problem. International Transactions In Operational Research, 21(1), 59-79. doi:10.1111/itor.12035

[19] Wang, Hongfeng & Chen, Yanjie. (2010). A hybrid genetic algorithm for 3D bin packing problems. Proceedings 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010. 703 - 707. 10.1109/BICTA.2010.5645211.

[20] 3DBinPacking. (2012). Recuperado el 10 de setiembre de 2017 a partir de: https://www.3dbinpacking.com/products

[21] MagicLogic Optimization Inc. - Cube-IQ Load Planning & Optimization Software » Products. (1996). Recuperado el 10 de setiembre de 2017 a partir de: http://magiclogic.com/

[22] iContainers. (2012). Precio del transporte marítimo para contenedor completo - iContainers. Recuperado el 10 de setiembre de 2017 a partir de: http://www.icontainers.com/es/2012/03/05/precio-del-transporte-maritimo-para-contenedor-completo/

[23] Liang, Shyi-Ching; Lee, Chi-Yu; and Huang, Shih-Wei (2007) "A Hybrid Meta-heuristic for the Container Loading Problem.

[24] SUNAT. (2012). RESOLUCIÓN DE SUPERINTENDENCIA NACIONAL ADJUNTA DE ADUANAS.

[25] Rodríguez, D. (2014) Diseño de un algoritmo de búsqueda tabú para la minimización del desperdicio de espacio en almacenes de empresas comercializadoras de tuberías.

[26] Michalewicz, Z., & Fogel, D. B. (2013). How to solve it: modern heuristics. Springer Science & Business Media.

[27] Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR), 35(3).

[28] LeBlanc, R. (2016). What is a pallet. Recuperado el 11 de setiembre de 2017 a partir de: https://www.thebalance.com/what-is-a-pallet-2877860.

[29] Jiménez Sánchez, J. and Jiménez Castillo, J. (2015). Cubicaje: distribución a bajo costo. Enfasis. Recuperado el 12 de setiembre de 2017 a partir de: http://www.logisticamx.enfasis.com/articulos/72752-cubicaje-distribucion-costo.

[30] WERRA, D., & Hertz, A. (1989). Tabu search techniques. Operations-Research-Spektrum, 11(3)

[31] Microsoft (2017). IDE de Visual Studio. Recuperado el 22 de octubre de 2017 a partir de: https://msdn.microsoft.com/library/dn762121(v=vs.140).aspx

[32] RStudio (2017). RStudio IDE features. Recuperado el 22 de octubre de 2017 a partir de: https://www.rstudio.com/products/rstudio/features/

[33] Microsoft (2017) C#. Recuperado el 22 de octubre de 2017 a partir de: https://www.microsoft.com/net/

[34] Crainic T., Perboli G. & Tadei R. (2007) Extremo Point-Based Heuristics for Three-Dimensional Bin Packing

[35] Fisher, R. A. (1992). Statistical methods for research workers. In *Breakthroughs in Statistics*. Springer New York.

[36] Iversen, G. R., & Norpoth, H. (1987). *Analysis of variance* (No. 1). Sage.

[37] SCRUM (2016). What is Scrum. Recuperado el 27 de octubre de 2017 a partir de: https://www.scrum.org/resources/what-is-scrum.

[38] Xuehao Feng, Ilkyeong Moon, Jeongho Shin (2013). Hybrid genetic algorithms for the three-dimensional multiple container packing problem.

[39] Lopes R.H.C. (2011) Kolmogorov-Smirnov Test. In: Lovric M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg.

[40] Sprinthall, R. C. (2011). Basic Statistical Analysis (9th ed.). Pearson Education.