PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UN ALGORITMO BÚSQUEDA TABÚ PARA LA MINIMIZACIÓN DEL DESPERDICIO DE ESPACIO EN ALMACENES DE EMPRESAS COMERCIALIZADORAS DE TUBERÍAS

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Daniel Alberto Rodríguez Sánchez

ASESOR: Ing. Rony Cueva Moscoso

Lima, Octubre de 2014

Tabla de contenido

Índice de Imágenes

CAPÍTULO 1

1 Problemática

Las empresas industriales se dedican a la transformación de materias primas a través de un conjunto de procesos y actividades en productos terminados, los cuales deben estar siempre disponibles para cubrir la demanda de sus clientes. Este tipo de empresas no solo deben mantener permanentemente ejecutando dichos procesos productivos sino que también deben considerar como un factor importante el almacenamiento de los productos terminados dentro de la empresa.

El almacenamiento de productos terminados debe considerar, entre otros, los siguientes criterios:

- Tamaño: En base al tamaño, los productos ocuparan un determinado espacio en el almacén. Es necesario asegurarse que, al momento de apilarse, se aproveche al máximo el espacio donde están siendo apilados.
- Forma: Los productos deben apilarse con el objetivo de no desperdiciar espacio debido a las formas de los mismos. Por lo general, los productos son de forma rectangular debido a que en su mayoría son cajas; sin embargo, también se puede dar el caso de contar con productos de formas especiales. Por ejemplo, las tuberías son de forma circular. Una buena práctica es agrupar los productos en base a su forma.
- Peso: Los productos deben apilarse buscando que los más livianos se apoyen sobre los más pesados. Esto se hace con el objetivo de evitar que se dañen productos por tener un peso excesivo encima.

Otros criterios que pueden considerarse al momento de realizar el almacenamiento de productos son si el material en base al cual están hechos es perecible, así como la rotación y el tiempo de los mismos en el almacén.

Por otro lado, la ubicación de los productos terminados dentro del almacén también es un elemento a considerar. Los productos terminados pueden colocarse en las siguientes ubicaciones:

- Ruma de productos: Es la forma más simple de almacenar productos. Los productos se apilan unos encima de otros sobre la superficie del almacén.
- Estanterías: Son estructuras metálicas diseñadas con la finalidad de almacenar todo tipo de productos. Las estanterías pueden ser desde estructuras simples para almacenar productos sueltos hasta grandes obras de ingeniería en las que las propias estanterías forman parte del sistema constructivo del almacén.

A pesar de tener en cuenta todos los factores mencionados anteriormente, productos de diversos tipos terminan apilándose unos encima de otros. Esto se debe al gran movimiento que se realiza de los productos en el almacén. Estos productos de diversos tamaños y formas apilados sin ninguna restricción generan diversos problemas como el deterioro de los mismos, la perdida de la ubicación dentro del almacén y el desperdicio de espacio de almacenamiento.

La falta de atención al problema de desperdicio de espacio se ve reflejada a través de espacios mal utilizados, apilamientos excesivos y congestionamientos constantes. Como consecuencia se generan costos extras como el mantenimiento de productos que nunca son usados y la perdida por deterioro de productos ya sea porque nunca fueron usados o por haber sido apilados debajo de productos muchos más grandes y pesados. Asimismo, el apilamiento sin restricciones de los productos en el almacén también puede generar accidentes; ya que, no habrá estabilidad en la ruma de productos o estanterías donde se encuentren ubicados. Esto no solo generará costos extras por la pérdida de los productos que ocasionaron el accidente sino también de los productos cercanos que pudieron haberse visto afectados en el mismo.

En las empresas de tuberías, el almacenamiento de los productos terminados tendrá una mayor complejidad. Esto se debe a que sus productos y servicios están basados principalmente en tuberías, las cuales tienen una forma circular. En base a esto, evitar el desperdicio de espacio será más complicado debido a la forma especial que poseen los mismos al momento de apilarse.

Por lo referido anteriormente, el almacenamiento de tuberías debe considerar los siguientes escenarios:

- Se generarán inevitablemente espacios considerables entre los mismos a diferencia de los clásicos productos que se guardan en cajas generando un espacio mínimo entre estos al momento de almacenarlos.

- Apilar tuberías de diversos tamaños y con forma circular en una determinada área del almacén aumenta el riesgo de que estos productos puedan deslizarse y ocasionar accidentes.

En la actualidad, existen diversas formas de resolver el problema de desperdicio de espacio en el almacén. A pesar de esto, estas soluciones pueden requerir un tiempo de ejecución considerable o no cubrir con todas las necesidades del negocio.

Durante el presente proyecto de fin de carrera se plantea el desarrollo de un algoritmo metaheurístico el cual brinde una buena solución para el problema planteado. Esto se debe a diversos factores como la ubicación del almacén donde se colocaran los productos (rumas o estanterías) y los criterios en base a los cuales se apilaran los productos (tamaño, forma y peso) que deben considerarse al momento de realizar el almacenamiento de productos terminados. Además, como se mencionó anteriormente, estos factores se vuelven más complejos cuando se trata de tuberías, debido a que estas poseen una forma circular. Finalmente, este algoritmo no tiene como objetivo resolver el problema de forma exacta dada la complejidad de tiempo y recursos que presenta; sin embargo, permite obtener una buena solución que pueda cubrir con las necesidades de almacenamiento y que pueda ser ejecutada en un tiempo comprensible a las necesidades del negocio.

1.1 Objetivo general

Desarrollar un algoritmo de búsqueda Tabú que permita minimizar el desperdicio de espacio en almacenes de empresas comercializadoras de tuberías.

1.2 Objetivos específicos

1. Diseñar las estructuras de datos necesarias para soportar los algoritmos GRASP y de búsqueda Tabú.
2. Diseñar la función objetivo a ser utilizada en los algoritmos GRASP y de búsqueda Tabú.
3. Diseñar el pseudocódigo del algoritmo metaheurístico GRASP construcción con doble relajación que permita resolver el problema planteado con la finalidad de poder realizar una comparación con el algoritmo de búsqueda Tabú en la experimentación numérica.
4. Diseñar el pseudocódigo del algoritmo metaheurístico de búsqueda Tabú que permita resolver el problema planteado.
5. Desarrollar una herramienta de software que permita generar datos aleatorios pero configurables a las variables propias del problema como son el tamaño, la forma y el peso de las tuberías. Esta herramienta será utilizada por ambos algoritmos.
6. Desarrollar la experimentación numérica que permita, mediante el uso de herramientas estadísticas, analizar e interpretar los resultados de los algoritmos GRASP y de búsqueda Tabú para poder determinar cuál es el algoritmo más óptimo para resolver el problema planteado.
7. Desarrollar un prototipo funcional que muestre los resultados obtenidos de la ejecución de los algoritmos GRASP y de búsqueda Tabú.

1.3 Resultados esperados

1. Resultado 1 para el objetivo 1: Documento de arquitectura de información presentando las estructuras de datos a utilizar por los algoritmos GRASP y de búsqueda Tabú.
2. Resultado 2 para el objetivo 2: Función objetivo a ser utilizada en los algoritmos GRASP y de búsqueda Tabú.
3. Resultado 3 para el objetivo 3: Pseudocódigo del algoritmo metaheurístico GRASP.
4. Resultado 4 para el objetivo 4: Pseudocódigo del algoritmo metaheurístico de búsqueda Tabú.

5. Resultado 5 para el objetivo 5: Herramienta de software desarrollada que permite la generación de data a ser utilizada por ambos algoritmos.
6. Resultado 6 para el objetivo 6: Documento con la selección y el planteamiento de los métodos estadísticos utilizados para el desarrollo de la experimentación numérica.
7. Resultado 7 para el objetivo 7: Prototipo funcional que muestra los resultados de la ejecución realizada con los algoritmos GRASP y de búsqueda Tabú.

2 Herramientas, métodos, metodologías y procedimientos

2.1 Introducción

A continuación se presenta una tabla con las herramientas, métodos y procedimientos utilizados en cada uno de los resultados esperados del presente proyecto de fin de carrera.

Resultados Esperados	Herramientas a usarse
RE1: Documento de arquitectura de información presentando las estructuras de datos a utilizar por los algoritmos GRASP y de búsqueda Tabú.	**Método del algoritmo GRASP** es utilizado como base para la implementación del mismo. **Método del algoritmo de búsqueda Tabú** es utilizado como base para la implementación del mismo. **Programación Extrema** es una metodología de desarrollo de ingeniería de software.
RE2: Función objetivo a ser utilizada en los algoritmos GRASP y de búsqueda Tabú.	**Neatbeans** es una herramienta que permite a los programadores escribir, compilar, depurar y ejecutar programas. **Java** es un lenguaje de programación orientado a objetos. **Método del algoritmo GRASP** es utilizado como base para la implementación del mismo. **Método del algoritmo de búsqueda Tabú** es utilizado como base para la implementación del mismo.

RE3: Pseudocódigo del algoritmo metaheurístico GRASP.	**Neatbeans** es una herramienta que permite a los programadores escribir, compilar, depurar y ejecutar programas. **Java** es un lenguaje de programación orientado a objetos. **Método del algoritmo GRASP** es utilizado como base para la implementación del mismo. **Programación Extrema** es una metodología de desarrollo de ingeniería de software.
RE4: Pseudocódigo del algoritmo metaheurístico de búsqueda Tabú.	**Neatbeans** es una herramienta que permite a los programadores escribir, compilar, depurar y ejecutar programas. **Java** es un lenguaje de programación orientado a objetos. **Método del algoritmo de búsqueda Tabú** es utilizado como base para la implementación del mismo. **Programación Extrema** es una metodología de desarrollo de ingeniería de software.
RE5: Herramienta desarrollada que permite la generación de data a ser utilizada por ambos algoritmos.	**Neatbeans** es una herramienta que permite a los programadores escribir, compilar, depurar y ejecutar programas. **Java** es un lenguaje de programación orientado a objetos. **Programación Extrema** es una metodología de desarrollo de ingeniería de software.
RE6: Documento con la selección y el planteamiento de los métodos estadísticos utilizados para el desarrollo de la experimentación numérica.	**Microsoft Excel** es un software que maneja hojas de cálculo y formulas estadísticas. **Comparación de dos tratamientos** es un método que permite determinar cuál tiene la media más óptima entre dos

	muestras desconocidas e independientes entre sí.
RE7: Prototipo funcional que muestra los resultados de la ejecución realizada con los algoritmos GRASP y de búsqueda Tabú.	**Neatbeans** es una herramienta que permite a los programadores escribir, compilar, depurar y ejecutar programas. **Java** es un lenguaje de programación orientado a objetos. **Programación Extrema** es una metodología de desarrollo de ingeniería de software.

2.2 Herramientas

2.2.1 IDE Neatbeans

Neatbeans es un entorno de desarrollo realizado principalmente para el lenguaje de programación Java. Esta herramienta permite a los programadores escribir, compilar, depurar y ejecutar programas.

Su principal característica es la modularidad. Todas las funciones del IDE son provistas por módulos. Cada módulo ofrece una función bien definida como el soporte de Java o el sistema de control de versiones.

A continuación, se presentan las principales ventajas de este IDE [NEATBEANS, 2013]:

- Edición de código más rápida y sencilla.
- Manejo de proyectos fácil y eficiente.
- Interfaz de desarrollo practica para el usuario.
- Herramientas para la identificación y corrección de código con errores.

El principal motivo para su elección se debe a que es una herramienta que permite una gestión sencilla y practica de sus proyectos. Esto permite optimizar los tiempos de desarrollo del código y enfocarse más en el diseño de los algoritmos GRASP y de búsqueda Tabú que resolverán el problema planteado.

2.2.2 Lenguaje de programación Java

Java es un lenguaje de programación orientado a objetos que fue desarrollado originalmente por James Gosling. Su sintaxis está basada principalmente en C y C++ pero posee menos facilidades de bajo nivel a diferencias de ellos.

El objetivo principal de este lenguaje de programación es permitir a los desarrolladores implementar una aplicación que pueda ser ejecutada en cualquier dispositivo. En otras palabras, el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

Entre sus principales características se tiene que es un lenguaje interpretado, distribuido, solido, seguro, de arquitectura neutral, portable, de multihilos y dinámico [ORACLE, 2013].

La principal razón de su elección se debe a que es un lenguaje de programación sencillo y práctico para el desarrollo de aplicaciones. Además, es el lenguaje de programación ideal para trabajar en el IDE Neatbeans.

2.2.3 Microsoft Excel

Microsoft Excel es un software (desarrollado y distribuido por Microsoft Office) que permite el manejo de hojas de cálculo. Este software permite a los usuarios elaborar tablas y formatos que incluyan cálculos matemáticos mediante fórmulas. Además, permite agrupar, ordenar y filtrar la información. Entre sus principales características se tiene el manejo de plantillas, el análisis rápido de datos y el control del formato de los gráficos [MICROSOFT EXCEL, 2013].

El principal motivo para su elección se debe a que cuenta con una gran variedad de fórmulas estadísticas que permitirán el desarrollo de la experimentación numérica.

2.3 Métodos y Procedimientos

2.3.1 Algoritmo GRASP

El método para el desarrollo del algoritmo GRASP a implementar durante el presente proyecto de fin de carrera seguirá el lineamiento establecido para el algoritmo GRASP descrito por Feo y Resende como se ha indicado en el marco conceptual.

Este método está basado en la construcción de una solución miope aleatorizada seguida de una búsqueda local usando la solución construida como el punto inicial de la búsqueda local. Este procedimiento se itera varias veces y la mejor solución obtenida de todas estas iteraciones se devuelve como la solución aproximada [FEO, RESENDE 1995].

El motivo principal de su elección es debido a la necesidad de realizar una comparación con el algoritmo de búsqueda Tabú. Además, este algoritmo también generará la población inicial a ser utilizada como solución base para el algoritmo de búsqueda Tabú. Del mismo modo, es necesario resaltar que solo se desarrollará la fase de construcción del algoritmo GRASP, ya que no es el objetivo del presente proyecto desarrollar este algoritmo.

2.3.2 Algoritmo de búsqueda Tabú

El método para el desarrollo del algoritmo de búsqueda Tabú a implementar durante el presente proyecto de fin de carrera seguirá el esquema básico del algoritmo de búsqueda Tabú descrito por Fred Glover como se ha indicado en el marco conceptual.

Este método tiene una filosofía basada en derivar y explotar una colección de estrategias inteligentes para la resolución de problemas, basadas en procedimientos explícitos y explícitos de aprendizaje. El marco de memoria adaptativa de la búsqueda Tabú no solo explota la historia del proceso de resolución del problema, sino que también exige la creación de estructuras para hacer posible tal explotación. De esta forma, los elementos prohibidos en la búsqueda Tabú reciben este estatus por la confianza en una memoria evolutiva, que permite alterar este estado en función del tiempo y las circunstancias [GLOVER, LAGUNA 1997].

La razón principal de la elección de este método es que el tema del presente proyecto de fin de carrera es el desarrollo de un algoritmo de búsqueda Tabú para el almacenamiento de tuberías. Por lo tanto, el uso de este método es indispensable. También es necesario especificar que se utilizará la memoria a corto plazo.

2.3.3 Comparación de dos tratamientos

Un problema frecuente que se presenta es comparar la media de dos procesos o dos tratamientos. Por ejemplo comparar dos proveedores, dos materiales, dos maquinarias o dos equipos de trabajo. La comparación de dos tratamientos es un método que permite comparar dos muestras que son resultados de la ejecución de una operación

específica realizada por distintos elementos. En base a este método, se determina cuál de los dos elementos es el que genera una muestra más significativa para una operación específica.

Durante el presente proyecto de fin de carrera, se realizará una comparación de dos tratamientos para determinar si es el algoritmo GRASP o el algoritmo de búsqueda Tabú el que presenta una solución más óptima para el almacenamiento de tuberías. En base a lo referido anteriormente, los algoritmos vendrían a ser los distintos elementos que ejecutaran una misma operación, en este caso, el almacenamiento de tuberías [GUTIERREZ, DE LA VARA 2004].

A continuación, se presentan las tres pruebas que se realizarán para ejecutar este método:

2.3.3.1 Prueba de Kolmogorov-Smirnov

La Prueba de Kolmogorov-Smirnov es una prueba no paramétrica que se emplea para probar el grado de concordancia entre la distribución de datos empíricos de la muestra y alguna distribución teórica especifica. El objetivo de esta prueba de bondad de ajuste es señalar y determinar si los datos estudiados o mediciones muéstrales provienen de una población que tiene una distribución teórica determinada.

El principal motivo de la elección de esta prueba es que permitirá determinar si los resultados obtenidos de la ejecución de los algoritmos GRASP y de búsqueda Tabú se comportan de manera normal.

2.3.3.2 Prueba F de Fisher

La Prueba F de Fisher fue creada por Ronald Fisher. Es una prueba estadística que sirve para comparar varianzas mediante una distribución de probabilidad continua F en la cual si la hipótesis es nula no puede ser rechazada. Es aplicable cuando se busca una comparación entre poblaciones para determinar cual tiene una mayor variación sobre otra o cuando se busca una comparación simultanea entre varias medias poblacionales.

El principal motivo de la elección de esta prueba es que permitirá asegurar la calidad de los resultados obtenidos de la ejecución de los algoritmos GRASP y de búsqueda Tabú.

2.3.3.3 Prueba Z

La Prueba Z es una prueba estadística utilizada para determinar si la media de dos poblaciones es diferente cuando las varianzas son conocidas y el tamaño de la muestra es lo suficientemente grande. Se asume que la prueba tiene una distribución normal y que los parámetros como la desviación estándar deben ser conocidos para que se pueda llevar a cabo una Prueba Z exacta.

El principal motivo de la elección de esta prueba es que permitirá determinar cuál de los dos algoritmos es el que presenta la muestra con la media más óptima.

2.4 Metodologías

2.4.1 Programación Extrema

La Programación Extrema es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck. Es el más destacado de los procesos agiles. Esto se debe a que el enfoque se desarrolló llevando a niveles “extremos” las practicas reconocidas, como el desarrollo iterativo.

En la Programación Extrema, los requerimientos se expresan como escenarios (llamados historias de usuario) que se implementan directamente con una serie de tareas y antes de escribir el código desarrollan pruebas para cada tarea. Todas las pruebas deben ejecutarse con éxito una vez que el nuevo código se integre en el sistema.

La programación extrema incluye algunas prácticas, las cuales reflejan los principios de los métodos agiles:

- El desarrollo incremental se apoya en pequeñas y frecuentes liberaciones del sistema. Los requerimientos se fundamentan en simples historias de clientes, o bien, en escenarios usados como base para decidir que funcionalidad debe incluirse en un incremento del sistema.
- La inclusión del cliente se apoya a través de un enlace continuo con el cliente en el equipo de desarrollo. El representante del cliente anticipa en el desarrollo y es responsable de definir las pruebas de aceptación para el sistema.
- El cambio se acepta mediante liberaciones regulares del sistema a los clientes, desarrollo de primera prueba, refactorización para evitar degeneración del código e integración continua de nueva funcionalidad.

- Mantener la simplicidad se logra mediante la refactorización constante, que mejora la calidad del código, y con el uso de diseños simples que no anticipan innecesariamente futuros cambios al sistema.

En un proceso de Programación Extrema, los clientes intervienen estrechamente en la especificación y priorización de los requerimientos del sistema. Estos últimos no se especifican como listas de actividades requeridas del sistema. En cambio, el cliente del sistema forma parte del equipo de desarrollo y discute los escenarios con otros miembros del equipo. En conjunto, desarrollan una “tarjeta de historia” que encapsula las necesidades del cliente. Entonces, el equipo de desarrollo implementa dicho escenario en una liberación futura del software [BECK, 2000].

El motivo principal de su elección es debido a su adaptabilidad con el mismo. El presente proyecto tiene como objetivo el desarrollo de un algoritmo; por lo tanto, se debe hacer un mayor énfasis en el desarrollo del algoritmo que en la documentación del mismo. La Programación Extrema busca un trabajo en conjunto más cercano entre los desarrolladores y el cliente. En este sentido, la programación Extrema brinda avances con cierta regularidad y para ser evaluados por el cliente. Aplicándolo al presente proyecto de fin de carrera, se presentarán semanalmente avances del proyecto, los cuales son evaluados. Esto es directamente compatible con la metodología de la Programación Extrema.

2.4.2 Análisis de Varianza

El Análisis de Varianza es una colección de modelos estadísticos y procedimientos asociados, en el cual la varianza esta particionada en ciertos componentes debidos a diferentes variables explicativas.

El Análisis de Varianza determina si diversas muestras tomadas en diferentes situaciones son significativamente diferentes desde un punto de vista estadístico. En base a lo descrito anteriormente, se puede decir que permite contrastar una hipótesis. El Análisis de Varianza descompone la variación observada en varias muestras para determinar en que muestra se produce una variación aleatoria. El objetivo es determinar si ciertas variables pueden explicar una parte significativa de la variación, siendo la variación aleatoria pequeña frente a la variación explicable o determinista [SPIEGEL, SCHILLER, SRINIVASAN, 2007].

La razón principal de su elección es debido a que es una metodología que permite la comparación de dos o más medias de alguna población a través de diversas pruebas estadísticas. Por lo tanto, en el presente trabajo, permitirá el análisis de los resultados de la ejecución de los algoritmos GRASP y de búsqueda Tabú para poder determinar cuál es el más óptimo.

2.4.3 PMBOK

PMBOK es una guía desarrollada por el Project Management Institute (PMI) que reúne un conjunto de conocimientos reconocidos como buenas prácticas que se constituyen como estándares de la gestión de proyectos.

PMBOK es un término inclusivo que describe la suma de conocimiento dentro de la gestión de proyectos. Al igual que en otras profesiones como derecho, medicina y contabilidad, el cuerpo del conocimiento corresponde a los profesionales y académicos que se aplican y avanzan en él.

El cuerpo de conocimiento de la gestión de proyectos incluye el conocimiento de buenas prácticas comprobadas tradicionalmente que son ampliamente aplicables, así como el conocimiento de las prácticas innovadoras y avanzadas que han visto un uso más limitado [PMI, 2013].

Los procesos de la gestión de proyecto pueden ser organizados en 5 grupos:

- Procesos de Iniciación.
- Procesos de Planificación.
- Procesos de Ejecución.
- Procesos de Seguimiento y Control.
- Procesos de Cierre.

Por otro lado, PMBOK cuenta con 10 áreas de conocimiento:

- Gestión de la Integración del proyecto.
- Gestión del Alcance del proyecto.
- Gestión del Tiempo del proyecto.
- Gestión del Costo del proyecto.
- Gestión de la Calidad del proyecto.
- Gestión de los Recursos Humanos del proyecto.

- Gestión de las Comunicaciones del proyecto.
- Gestión de los Riesgos del proyecto.
- Gestión de las Adquisiciones del proyecto.
- Gestión de los Interesados del proyecto.

La razón principal de su elección se debe a la necesidad de una gestión de proyecto. En este sentido, el PMBOK ofrece una metodología generalizada que permite ser adaptada a diversos tipos de proyectos.

Cabe resaltar que durante el presente proyecto de fin de carrera no se aplicará todo lo presentado por el PMBOK sino solo aquellos puntos que sean necesarios para el funcionamiento del mismo.

A continuación, se presentan las áreas del conocimiento que serán utilizadas:

- Gestión de la Integración del proyecto. Se establecerá el acta de constitución del proyecto. Además, se dirigirán y gestionarán distintas etapas del presente proyecto. De este modo, se podrá seguir un orden determinado con la finalidad de llevar el proyecto de la mejor manera.
- Gestión del Alcance del proyecto. Se desarrollará el EDT (Estructura de Desglose del Trabajo) del proyecto. Además, se definirá el alcance con la finalidad de que todas las actividades que se realicen permitan alcanzar el objetivo del proyecto. Este alcance deberá ser verificado y controlado durante las distintas etapas del proyecto con la finalidad de que las actividades no superen el alcance del mismo.
- Gestión del Tiempo del proyecto. Se desarrollará un cronograma de actividades donde se indique su duración y restricciones en el proyecto. Para poder realizar esto, previamente se debe realizar un análisis donde se definan las actividades a realizar y se estime su tiempo de duración. Además, se deberá realizar un seguimiento del cronograma de modo que se actualice el mismo cuando surjan cambios.
- Gestión de las Comunicaciones del proyecto. Se deberán definir las comunicaciones que se usarán para comunicarse con los interesados (profesor del curso y asesor) del proyecto en base a sus necesidades. Se presentarán básicamente informes del estado del proyecto y del desempeño del mismo.

- Gestión de los Riesgos del proyecto. Se definirán los riesgos que puedan afectar el presente proyecto. Además, se hará el análisis respectivo de los riesgos identificados. Finalmente, se establecerán controles para la reducción de amenazas del proyecto.

3 Alcance

Durante el presente proyecto de fin de carrera, se deben tener en cuenta las siguientes consideraciones:

- El proyecto esta aplicado únicamente a las empresas comercializadoras de tuberías. Esto se debe a que la forma circular y el tipo de material de las tuberías dan un enfoque para resolver diferente a cualquier otro tipo de producto, ya que se generan espacios considerables entre los mismos y se corre el riesgo de que puedan deslizarse y ocasionar accidentes.
- Las variables que permiten realizar el correcto almacenamiento de tuberías son: tamaño, forma y peso de las tuberías. Esto se debe a que son factores indispensables en el almacenamiento de tuberías. El tamaño y la forma de las tuberías permitirán verificar que se aproveche al máximo el espacio donde están siendo apilados mientras el peso busca evitar que los productos se dañen por tener un peso excesivo encima.
- El algoritmo GRASP solo será implementado en la fase de construcción y con un valor alfa que será calibrado durante la experimentación numérica. Esto se debe a que está orientado al apoyo del algoritmo de búsqueda Tabú.
- El algoritmo de búsqueda Tabú será implementado usando una memoria a corto plazo. Esto se debe a que se busca evitar costos computacionales altos como en los que se podría incurrir si se usase una memoria a largo plazo en el desarrollo del algoritmo de búsqueda Tabú.
- La solución devuelta por los algoritmos GRASP y de búsqueda Tabú no necesariamente será la más óptima pero si lo bastante buena. Esto se debe a que estos algoritmos presentan cierto grado de aleatoriedad. Durante la experimentación numérica se evaluará si la solución resulta optima o no.

3.1 Limitaciones

Las limitaciones y obstáculos del presente proyecto de fin de carrera son las siguientes:

- Dependiendo del hardware o software del equipo en el que se ejecuten los algoritmos GRASP y de búsqueda Tabú, el tiempo de ejecución puede variar en base a la complejidad de los algoritmos que sean ejecutados.
- El número de iteraciones que realizara cada algoritmo. Esto se debe a que los algoritmos deben ser definidos con un tiempo máximo definido. Por lo tanto, el número de iteraciones que se realicen al momento de ejecutar los algoritmos no debe hacer que el tiempo de ejecución sobrepase el tiempo máximo definido.

3.2 Riesgos

A continuación se presenta una tabla con los principales riesgos identificados en el presente proyecto de fin de carrera. Además de los impactos y medidas correctas de los mismos.

Riesgo Identificado	**Impacto en el proyecto**	**Medidas correctivas para mitigar**
Ineficiente planificación del proyecto.	- No se pueden presentar entregables a tiempo. - No se pueden realizar entregables.	Realizar una revisión exhaustiva de los plazos y tiempo del proyecto para poder definir una planificación real.
Perdida de información relevante del proyecto.	- No se pueden presentar entregables a tiempo. - Se pierde tiempo en rehacer los documentos perdidos.	Realizar backups de la información del proyecto de forma constante.
Dificultad para desarrollar el algoritmo debido a la complejidad del mismo.	- No se pueden presentar entregables a tiempo. - No se pueden realizar entregables.	Realizar una planificación del proyecto en la cual se dé un tiempo considerable para la realización del algoritmo.
Deficientes datos a ser	- No se puede obtener una	Desarrollar una

utilizados como entradas para los algoritmos.	solución óptima para el problema. - No se puede realizar una experimentación numérica valida.	herramienta que genere datos de entrada para los algoritmos GRASP y de búsqueda Tabú.

4 Justificativa y viabilidad del proyecto

4.1 Justificativa

En la actualidad, las empresas industriales no solo deben enfocarse en los procesos productivos que están ejecutando sino también en el almacenamiento de sus productos terminados. Esto se debe a que la demanda de un producto no siempre ira acorde en tiempo y calidad con la oferta del mismo. Por lo tanto, se deben manejar ciertos stocks que permitan cumplir con la demanda del cliente. Además, este almacenamiento también debe optimizar el espacio de los almacenes en base a criterios como el tamaño, forma y peso de los productos. En el caso de las empresas de tuberías, estos factores deben analizarse con un mayor detalle debido a que la forma circular de este tipo de productos permite que se generen espacios más grandes entre los mismos al momento de apilarse uno sobre otro, además existe el riesgo de que estos productos puedan deslizarse y ocasionar accidentes.

En base a lo referido anteriormente, el presente proyecto de fin de carrera tiene como objetivo disminuir los espacios que se puedan generar entre las tuberías y evitar que se produzcan accidentes por deslizamiento de los mismos. Esto no solo se verá reflejado en una mejora del espacio de utilización del almacén, sino que permite al área de producción de la empresa generar más productos terminados en vista que tendrá espacio donde almacenarlos. Como consecuencia, los interesados ajenos a la empresa como los clientes o proveedores también se verán beneficiados, ya que contaran con una mayor cantidad de productos a su disposición o se requerirá de una mayor cantidad de insumos para producir las tuberías, lo cual finalmente se verá reflejado en utilidades para la empresa.

4.2 Viabilidad

A continuación se presentan los criterios que influyen en la viabilidad del proyecto:

- **Viabilidad Técnica.** No se presenta una gran limitante respecto a la parte técnica. Esto se debe, a que la mayoría de herramientas, métodos,

procedimientos y metodologías a emplear ya han sido estudiadas y aplicadas durante el desarrollo de la carrera universitaria.

En lo referente a las herramientas, IDE Neatbeans es un entorno de desarrollo que ha sido usado para la realización de diversos trabajos en el lenguaje de programación Java. Además, la página del proveedor ofrece un soporte amplio en base a consultas directas por correo o a través de un foro de discusión [NEATBEANS-FORUMS, 2013].

Por otro lado, Microsoft Excel es una herramienta que ha sido usada durante la carrera universitaria. Además, también se ha usado esta herramienta para el desarrollo de la experimentación numérica en trabajos pasados; por lo que, no presentará curva de aprendizaje. Del mismo modo, el método de comparación de dos tratamientos que será usado para la experimentación numérica también ha sido usado en trabajos pasados.

Finalmente, los métodos para la elaboración de los algoritmos GRASP y de búsqueda Tabú serán los únicos que presenten nuevo conocimiento; sin embargo, esta curva de aprendizaje no será de gran magnitud debido a que se contará con la guía del asesor.

Por lo tanto, en base a todo lo referido anteriormente, el proyecto es viable desde el punto de vista técnico.

- **Viabilidad Temporal.** Se tiene una gran limitante respecto al tiempo del proyecto, el cual es al término del próximo ciclo. Sin embargo, el proyecto no solo está planificado para ser desarrollado durante el tiempo del calendario académico sino que también se desarrollará durante el verano y el curso Proyecto de Tesis 2. Por lo tanto, si se tiene una buena planificación y se realizan las medidas correctivas para mitigar los riesgos identificados anteriormente, el proyecto será viable.

- **Viabilidad Económica.** No se presenta una gran limitante en este aspecto. Esto se debe a que las herramientas a usarse provienen de proveedores gratuitos o fueron adquiridas con anterioridad. En base a esto, se puede afirmar que el proyecto es viable desde el punto de vista económico.

CAPÍTULO 2

1 Marco conceptual

1.1 Introducción

En esta sección se mostraran los conceptos necesarios para entender el presente proyecto de fin de carrera. Esto comprenderá el problema de almacenamiento de tuberías y los conceptos necesarios para comprender el desarrollo de un algoritmo metaheurístico como solución para el presente problema.

1.2 Objetivo del marco conceptual

El objetivo del marco conceptual es profundizar en la problemática a través de la presentación de diversos conceptos relacionados tanto con el problema que se plantea resolver como con la solución que se propone.

1.3 Conceptos relacionados al problema

1.3.1 Almacenamiento de Productos Terminados

El almacenamiento de productos terminados tiene como finalidad cubrir los desequilibrios entre la oferta y la demanda. Esto se basa en la premisa de que la demanda de un producto no siempre ira acorde en tiempo y cantidad con la oferta del mismo. Por lo tanto, se deben manejar ciertos stocks para poder cumplir con la demanda del cliente. En caso contrario, se producirán pérdidas que afecten no solo en los ingresos de la empresa sino también en su imagen.

El almacenamiento de productos terminados se realiza mediante el apilamiento de los mismos unos encima de otros en determinado lugar del almacén. El apilamiento solo deja espacio entre los productos cuando el contacto entre los mismos no es perfecto. Además, deben considerarse ciertos criterios como el tamaño, la forma y el peso de los productos que se almacenan. Esto puede realizarse a mano o por medio de elevadores de carga [HOMPEL, SCHMIDT 2007].

El almacenamiento de productos terminados puede darse de las siguientes formas:

- Almacenamiento Ordenado. Cada tipo de producto tiene asignado a un lugar único. Por lo tanto, los espacios destinados para alojar cada tipo de producto

serán acondicionados para cubrir las características particulares de los mismos. Entre sus ventajas podemos apreciar la facilidad de control y manipulación de los productos. Sin embargo, la limitación de almacenamiento a los espacios ocasiona que no se pueda aprovechar al máximo el almacén.

- Almacenamiento Caótico. Cada producto va siendo asignado sin contemplar ningún orden predeterminado. A pesar de esto, contempla ciertas reglas como la optimización de recorridos, condiciones medioambientales y el aprovechamiento de los huecos que se puedan generar para colocar los nuevos productos que entren. Su principal ventaja es que permite el aprovechamiento del espacio; sin embargo, la gestión de los productos almacenados es más complicada.

Durante el presente proyecto de fin de carrera se trabajará en base a un almacenamiento caótico.

1.3.2 Tuberías

Las tuberías son productos de forma circular que transportan agua y otros fluidos. Son utilizadas principalmente en el desarrollo y la ejecución de obras para domicilios, infraestructura, electricidad, riego e industria. En la actualidad, el sector de tuberías se encuentra en crecimiento impulsado por la inversión en redes públicas de agua y alcantarillado [TIGRE 2013; EL COMERCIO 2013].

Imagen 1: Tuberías

1.3.3 Problema de Bin-Packing

El problema de Bin-Packing consiste en tener n objetos, cada uno con un tamaño en particular. Estos objetos deben ser asignados a varios contenedores con la finalidad usar la menor cantidad posible. Por supuesto, el tamaño total de los objetos asignados a un contenedor no debe exceder su capacidad. Asumiendo que la capacidad de los

contenedores es 1, esto se puede representar en términos más matemáticos de la siguiente forma [KORTE, VYGEN 2005]:

> Para: Una lista de números no negativos $a_1, \dots, a_n \leq 1$.
> Necesito: Encontrar un $k \in \mathbb{N}$ y una asignación $f: \{1, \dots, n\} \to \{1, \dots, k\}$ con $\sum_{i:f(i)=j} a_i \leq 1$ para todo $j \in \{1, \dots, k\}$ tal que el k usado sea mínimo.

El problema de Bin-Packing tiene diversas aplicaciones relacionadas con la optimización de espacio. Por lo tanto, es adaptable para el problema de desperdicio de espacio que se está buscando resolver. En el caso de las empresas de tuberías, los objetos mencionados en el enunciado del problema vienen a ser las tuberías mientras que los contenedores son las diversas locaciones del almacén donde se pueden colocar.

1.4 Conceptos relacionados a la propuesta de solución

1.4.1 Optimización Combinatoria

Durante el transcurso de la vida universitaria o profesional, es muy probable que se escuchen expresiones como:

- “Es una solución hecha a la medida”
- “La solución hallada cumple con las expectativas”
- “No será la solución exacta, pero es la mejor solución que se puede dar”

Al momento de resolver un problema, habrá ciertos casos para los cuales se pueda encontrar soluciones exactas. En otros casos, bastará con encontrar soluciones que abarquen ciertas expectativas o rangos de calidad [TUPIA, 2009].

La optimización combinatoria es una rama de la Ciencia de la Computación que busca obtener las posibles combinaciones y ordenamientos que se realicen entre elementos o estructuras de datos con la finalidad de obtener la mejor solución entre soluciones posibles [PAPADIMITRIOU, STEIGLITZ 1998].

Los problemas que pretende resolver la optimización combinatoria tienen como objetivo optimizar (maximizar o minimizar) uno o más criterios con la finalidad de llegar a un objetivo.

La forma más general en la que puede ser expresado un problema de optimización es la siguiente [TUPIA, 2009]:

Optimizar: $f(x)$
Sujeto a: $g_i(x) \geq 0$, i = 1,…, m
$h_j(x) \geq 0$, j = 1,…, m

Dónde:

- $f(x)$ es la función objetivo y representa el valor a optimizar.
- $g_i(x)$ y $h_j(x)$ son las restricciones del problema y representan las condiciones que debe poseer una solución viable.

1.4.2 Complejidad Algorítmica del Problema

La complejidad algorítmica de un problema se determina por el grado de dificultad que presenta el algoritmo que lo resuelve. A continuación, se presenta la clasificación de los problemas según su complejidad algorítmica [TUPIA, 2009]:

- Clase P. En esta clase se encuentran los algoritmos que poseen una complejidad polinomial.
- Clase NP. En esta clase se encuentran los algoritmos que generan soluciones aproximadas. Un problema NP no puede ser resuelto por un algoritmo de complejidad polinomial. Además, dentro de esta clasificación, también se puede definir un problema como NP-Hard. Un problema NP es “Hard” (difícil) respecto a un conjunto de problemas NP, si el algoritmo que resuelve el problema NP-Hard puede resolver cualquiera de los problemas NP del conjunto descrito anteriormente.

1.4.3 Heurísticas y Metaheurísticas

Las técnicas heurísticas y metaheurísticas son métodos aproximados que permiten encontrar soluciones lo suficientemente buenas para resolver problemas de optimización combinatoria. No se obtendrán soluciones exactas pero si aproximadas que permitan solucionar el problema con costos computacionales relativamente bajos [TUPIA, 2009].

1.4.3.1 Algoritmos Heurísticos

La heurística es un método que busca reducir el costo computacional sacrificando la precisión y la calidad de los resultados con la finalidad de ahorrar tiempo y espacio. En una definición más simple, es un proceso de búsqueda que resuelve problemas difíciles mediante la obtención de una solución no exacta [BROWNLEE, 2011; ZANAKIS, EVANS 1981].

Existen características para considerar la aplicación de un método heurístico en la resolución de un problema dado [MOSCATO, 1996]:

- No se necesite una solución exacta.
- No existe un método exacto de la resolución o de existir, requiere mucho tiempo de procesamiento.
- Datos poco fiables o poco variados.
- Aparecen limitaciones de tiempo y de proceso computacional (hardware).
- Es un paso intermedio en la aplicación de otro tipo de algoritmos.

Finalmente, un algoritmo heurístico es aquel que puede encontrar soluciones lo suficientemente buenas para un problema sin la necesidad de haber revisado todas las posibles soluciones [BROWNLEE, 2011].

1.4.3.2 Algoritmos Metaheurísticos

La metaheurística se define como un proceso de generación iterativo que, usando como base una heurística, busca combinar de forma inteligente diferentes conceptos para explorar y explotar el espacio de búsqueda. Además, utiliza estrategias de aprendizaje para manejar la información de la estructura de información con la finalidad de encontrar de manera eficiente soluciones óptimas [OSMAN, LAPORTE 1996].

A diferencia de la heurística, busca mejorar la solución mediante métodos diversos como la combinación de métodos heurísticos y buscando por encima del optimo local [BROWNLEE, 2011].

Las características fundamentales de los metaheurísticos son las siguientes [BLUM, ROLI, 2003]:

- Son estrategias que “guían” el proceso de búsqueda.

- Tienen como objetivo explorar eficientemente el espacio de búsqueda con la finalidad de encontrar soluciones óptimas.
- Sus técnicas van desde procedimientos de búsquedas locales hasta procesos de aprendizaje complejos.
- Son aproximados y usualmente no determinísticos.
- Incorporan mecanismos que evitan ser confinados en algún espacio de búsqueda.
- Las más avanzadas metaheurísticas mantienen un historial de sus búsquedas (mediante el uso de memoria) para así poder usarlas como base en las siguientes búsquedas.

A pesar de todas las ventajas que poseen los algoritmos metaheurísticos sobre los algoritmos heurísticos, es necesario señalar que para algunos criterios no se obtienen buenas soluciones debido a las siguientes características [TUPIA, 2009]:

- Voracidad. Siempre selecciona al mejor candidato para formar parte de la solución en base al valor de la función objetivo.
- Miopía. No analiza los efectos de haber seleccionado un elemento como parte de la solución. El elemento seleccionado permanece en la solución hasta el final. Por lo tanto, esta elección es única e inmodificable.

1.4.4 Algoritmo GRASP

El algoritmo GRASP es una técnica que fue desarrollada por Thomas Feo y Mauricio Resende a finales de los años 80. Consiste en la construcción de una solución miope aleatorizada seguida de una búsqueda local usando la solución construida como el punto inicial de la búsqueda local. Este procedimiento se itera varias veces y la mejor solución obtenida de todas estas iteraciones se devuelve como la solución aproximada [FEO, RESENDE 1995].

La siguiente figura muestra la estructura básica del algoritmo GRASP [TUPIA, 2009]:

Procedimiento GRASP (Instancia del Problema)
1. Leer (Instancia)
2. Mientras (no se cumpla condición de parada) hacer
   a. Procedimiento Construcción (S)
   b. Procedimiento Mejoría (S)
3. Fin Mientras
4. Retornar (Mejor S)

Fin GRASP

Imagen 2: Estructura Básica del algoritmo GRASP

Dónde:

- En el punto 1, se ingresan los datos que conforman la instancia del problema.
- En el punto 2, se realizan los procesos de construcción y de mejoría de la solución hasta que se cumpla el criterio de parada.
- En el punto 3, se devuelve el resultado de la aplicación del algoritmo luego de haber cumplido con el criterio de parada.

A continuación se explicara con mayor detalle los dos componentes más importantes del algoritmo GRASP: La fase de construcción GRASP y la fase de mejoría de la solución GRASP.

1.4.4.1 Fase de Construcción GRASP

La fase construcción GRASP construye una solución a través de la incorporación de elementos, uno a la vez. Durante cada paso del proceso de construcción se tiene a la mano una solución parcial. Este elemento seleccionado recibe el nombre de candidato. El elemento seleccionado para ser un elemento candidato se determina mediante el uso de una función miope. Esta función miope mide el aporte que cada elemento da a la solución parcial. La aleatoriedad se introduce mediante una lista restringida de candidatos (RCL), la cual contiene un conjunto de elementos candidatos con los mejores valores de la función miope. El siguiente candidato a ser introducido a la solución se selecciona al azar de la lista restringida de candidatos [RESENDE, GONZÁLEZ 2003].

La siguiente figura muestra la estructura básica del algoritmo de la fase de construcción [TUPIA, 2009]:

Procedimiento GRASP Construcción (N, c, E, F, S, a)
Inicio
    Inicializar c, E, a
    $S = \emptyset$
    N = E
    1. Mientras <no se cumpla condición de parada> hacer
    Inicio
      1.1 $RCL = \emptyset$
      1.2 $\beta = Mejor\ \{c(x): x \epsilon N\}$
      1.3 $\tau = Peor\ \{c(x): x \epsilon N\}$
      1.4 $RCL = \{\forall x \epsilon N: \beta \leq c(x) \leq \beta + \alpha(\tau - \beta)\}$
      1.5 $a = Aleatorio(RCL)$
      1.6 $Si\ S \cup \{a\}\ \epsilon\ F \rightarrow S = S \cup \{a\}$
      1.7 $N = N - \{a\}$
      1.8 $Adaptar\ c$
    Fin Mientras
Fin GRASP Construcción.

Imagen 3: Algoritmo de la fase de construcción GRASP

Dónde:

- En el punto 1, el proceso se repetirá hasta que se cumpla con el criterio de parada.
- Del punto 1.1 al 1.3, se inicializa el RCL y se hallan el mejor y peor valor de c para N.
- En el punto 1.4, se forma la lista RCL mediante el uso del parámetro de relajación α. Este parámetro se encarga de que la solución sea menos voraz. Considerar que si α es igual a 0, el criterio a emplearse será aleatorio; mientras que si es igual a 1, el criterio será goloso.
- En el punto 1.5, se realiza una selección aleatoria de un elemento de RCL.
- En el punto 1.6, se determina si el conjunto solución es factible luego de agregarle el elemento seleccionado anteriormente.
- En el punto 1.7, se elimina el elemento seleccionado de N.
- En el punto 1.8, se configura la función c para las siguientes iteraciones.

1.4.4.2 Fase de Mejoría de la solución GRASP

La fase de mejoría de la solución GRASP explora repetidamente la vecindad de una solución en busca de una solución que mejore la actual. Cuando no se encuentra esta

nueva solución, se dice que la solución es localmente óptima. La fase de mejoría es muy importante en el algoritmo GRASP; ya que, sirve para buscar soluciones óptimas en regiones prometedoras del espacio de soluciones [RESENDE, GONZÁLEZ 2003].

La siguiente figura muestra la estructura básica del algoritmo de la fase de mejoría del algoritmo GRASP [TUPIA, 2009]:

Procedimiento GRASP Mejoría (X)
Mientras (no es posible mejorar solución) hacer
Inicio
1. $N_k(X) = \{z_1, z_2, z_3, \dots z_N\}\ tq\ \sum_j^N |z_j - x_j| \leq k, z_j \epsilon \{0,1\}$
2. $Best_k(N_k) = \{C_i^{i+1} z_i\}$
3. $Si\ Best_k(N_k) es\ mejor\ que\ X \rightarrow X = Best_k(N_k)$

Fin Mientras
Retornar(X)
Fin GRASP Mejoría

Imagen 4: Algoritmo de la fase de mejoría GRASP

Dónde:

- En el punto 1, se forma la vecindad k-ésima en base a la solución X en la cual cada elemento N_k tiene un grado de separación máximo igual a k.
- En el punto 2, se busca el mejor elemento de la vecindad en base a la aplicación de componentes de los elementos.
- En el punto 3, si se logra mejor X, se actualiza la solución para que las siguientes iteraciones sean sobre la nueva solución encontrada.

Cabe resaltar que a pesar de las ventajas que ofrece esta fase del algoritmo GRASP, su aplicación depende de las restricciones del problema. Esto es debido a que la mayoría de problemas tienen restricciones que imposibilitan hacer combinaciones entre elementos que forman parte de la solución final.

1.4.5 Algoritmo Búsqueda Tabú

La abundancia de problemas de optimización combinatoria, encontrados en diversas áreas como telecomunicaciones, logística, finanzas, transporte y producción, ha motivado al desarrollo de poderosas técnicas de optimización. El objetivo es que estas técnicas de optimización sean eficientes y capaces de manejar la complejidad de este tipo de problemas. En base a esta motivación es que se desarrolló el método de búsqueda tabú [GLOVER, LAGUNA 1997].

La Búsqueda Tabú fue desarrollada por Fred Glover en 1988. Es un procedimiento adaptativo que tiene la habilidad de hacer uso de otros métodos como algoritmos de programación lineal y heurísticas especializadas con la finalidad de sobrepasar las limitaciones del óptimo local [GLOVER, 1989].

La Búsqueda Tabú está basada en la premisa que para resolver un problema se debe incorporar memoria adaptativa y exploración responsable. Para poder entender mejor este concepto se presentará una analogía. (Ver imagen 5) Se puede observar a un hombre escalando una montaña. Este hombre debe recordar los elementos claves del camino recorrido (usando memoria adaptativa) y debe estar en la capacidad de realizar elecciones estratégicas a lo largo del camino (usando exploración responsable). En base a esto, la memoria adaptativa permite la implementación de procedimientos capaces de realizar la búsqueda de soluciones efectivas y de bajo costo mientras la exploración responsable (haciendo uso de la memoria) contempla que una mala elección basada en la estrategia permite proporcionar pistas útiles sobre qué elementos cambiar para mejorar la estrategia [GLOVER, LAGUNA 1997].

Imagen 5: Memoria Adaptativa y Exploración Responsable

La siguiente figura muestra la estructura básica del algoritmo de Búsqueda Tabú [JAZIRI, 2008]:

Procedimiento Búsqueda Tabú
Inicio
$T = [\,]$;
$s = solución\ inicial$
$s^* = s$
1. Repetir
   1.1 $Encontrar\ Mejor\ Candidato\ s' \epsilon\ N(s)$
   1.2 $Si\ f(s') > f(s^*)\ entonces\ s^* = s'$
   1.3 $s = s'$
   1.4 $Actualizar\ Lista\ Tabu\ T$
2. Hasta <Condición de Parada>

Fin

Imagen 6: Estructura Básica del algoritmo Búsqueda Tabú

Dónde:

- En el punto 1, el proceso se repetirá hasta que se cumpla con el criterio de parada.
- En el punto 1.1, se halla al mejor candidato s para N.
- En el punto 1.2, se evalúa si s' (candidato encontrado) es mejor que s^* (solución actual). De ser así, se actualiza el s^* con la nueva mejor solución encontrada.
- En el punto 1.3, se actualiza la solución encontrada para trabajar con esta en las siguientes iteraciones.
- En el punto 1.4, se actualiza la lista tabú.

A continuación se explicaran algunos conceptos con la finalidad de brindar un mejor entendimiento sobre cómo funciona la búsqueda tabú.

1.4.5.1 Uso de Memoria

Las estructuras de memoria de la búsqueda tabú funcionan en base a cuatro dimensiones principales. Estas se basan en la propiedad de ser reciente, en frecuencia, en calidad y en influencia. Las dimensiones basadas en lo reciente y en la frecuencia se complementan para lograr un balance entre intensificación y diversificación, lo cual se ve reflejado mediante un seguimiento a las propiedades que han cambiado recientemente y la frecuencia con la que ha sucedido. La dimensión basada en la calidad se desarrolla haciendo uso del aprendizaje con incentivos. Las

acciones que conducen a soluciones óptimas se refuerzan mientras las acciones que derivan en soluciones ineficientes reciben una penalización. Finalmente, la dimensión basada en la influencia busca evaluar el impacto de las decisiones tomadas en el proceso de búsqueda, no solo considerando la calidad sino también la estructura de las soluciones.

El uso de memoria en la búsqueda tabú es tanto explicita como basada en atributos. La memoria explicita almacena en memoria soluciones elites completas las cuales fueron visitadas durante el proceso de búsqueda con la finalidad de expandir los entornos de búsqueda. La memoria basada en atributos almacena información de determinados atributos que cambian al pasar de una solución a otra con la finalidad de prohibir determinados movimientos en procesos de búsqueda futuros [GLOVER, LAGUNA 1997].

1.4.5.2 Intensificación y Diversificación

Las estrategias de intensificación y de diversificación son componentes altamente importantes. Las estrategias de intensificación modifican las reglas de selección con la finalidad de encontrar soluciones en base a buenas combinaciones de movimientos. Esto hace necesario seleccionar soluciones elites de las cuales se tomaran buenos atributos que puedan ser incorporados luego a las nuevas soluciones. Por otro lado, las estrategias de diversificación conducen la búsqueda a zonas del espacio de soluciones que no han sido exploradas anteriormente con la finalidad de generar soluciones que difieran considerablemente con las ya encontradas [GLOVER, LAGUNA 1997].

1.4.5.3 Memorias a corto y largo plazo

La búsqueda tabú hace uso de la memoria explicita y basada en atributos para determinar nuevas soluciones desde la base de una solución actual. Esto lo puede realizar mediante el uso de memorias a corto y largo plazo.

Las memorias a corto plazo actúan en base a un atributo llamado tabú-activo. Un tabú-activo es un atributo de una solución que ha sido modificado en un pasado reciente. En base a lo referido anteriormente, si una solución contiene atributos tabú-activos o combinaciones de los mismos, esta se convierte en una solución prohibida. De esta forma, se garantiza que la búsqueda tabú no visite soluciones encontradas en un pasado reciente. El objetivo es impulsar al descubrimiento de nuevas soluciones de alta calidad.

Por otro lado, las memorias a largo plazo actúan de la misma forma que las memorias a corto plazo pero complementándolas con la información de la frecuencia con que se realizaron determinados movimientos. Esto permite ampliar el criterio que se tendrá al momento de seleccionar el siguiente movimiento. La frecuencia es una proporción donde el numerador representa el número de ocurrencia de un evento en particular mientras el denominador puede representar uno de los siguientes valores: el número total de ocurrencias de todos los eventos ocurridos, la suma de los numeradores o el máximo valor del numerador. El objetivo es realizar una evaluación más profunda antes de producirse un movimiento [GLOVER, LAGUNA 1997].

1.5 Conclusión

El problema del desperdicio de espacio es un problema de optimización combinatoria adaptado mediante el problema de Bin-Packing. En base a esto, se puede afirmar que no habrá una solución exacta para el mismo o de haberla no estará en tiempos adecuados a las necesidades del negocio. Por lo tanto, la solución para este problema es el desarrollo de un algoritmo metaheurístico; ya que, este algoritmo busca encontrar una buena solución que cumpla con las necesidades del negocio en base a los métodos de aprendizaje que realiza durante el proceso de búsqueda.

2 Estado del arte

2.1 Introducción

La optimización del espacio al momento a apilar los productos es un problema que se presenta en el día a día de muchas empresas. En base a esto, se han planteado diversas formas de resolverlo. En la presente sección, se muestran algunos métodos a través de los cuales se desarrolló una solución.

2.2 Objetivos de la revisión del estado del arte

El objetivo de la revisión del estado del arte es recuperar y transcender reflexivamente el conocimiento acumulado sobre el problema del desperdicio de espacio durante el almacenamiento de productos.

2.3 Método usado en la revisión del estado del arte

El estado del arte se realizó en base a una revisión sistemática. Esta elección se debe a la ventaja que ofrece la misma para minimizar la parcialidad de la investigación mediante una investigación bibliográfica exhaustiva.

En primer lugar, se planteó una pregunta de investigación: ¿Cuáles son las iniciativas que han sido llevadas a cabo para la optimización de espacio en los almacenes y que al mismo tiempo presentan un caso de estudio real? En base a esta pregunta, se elaboró una lista de términos usados para resolver la pregunta de investigación. La lista de términos está conformada por los siguientes elementos: investigación, optimización de espacio, apilamiento, tuberías, almacén, Bin-Packing, algoritmo, software.

En segundo lugar, se realizó búsquedas en base a cadenas elaboradas a partir de los elementos de la lista de términos. En la presente revisión sistemática, estas cadenas fueron usadas directamente en páginas de institutos de investigación reconocidos como AICIT. Además, también se usaron las librerías digitales de Scopus, IEEE y Springer.

Una vez encontrado una gran cantidad de estudios primarios relacionados con el tema, el análisis de su inclusión dentro del estudio estuvo basado en el título, “abstract” y palabras claves de los artículos. Los objetivos principales de estos artículos debían ser la optimización de espacio y el apilamiento eficiente de elementos

en un espacio determinado. Por ejemplo, se encontraron muchos artículos sobre el Bin-Packing pero que trataban el problema de forma general; por lo que, no aportaban debido a que este problema no solo se resuelve optimizando el espacio dentro del contenedor sino también considerando que hay productos que no pueden ser apilados sobre otros debido características como su peso o forma.

Finalmente, como resultado de la revisión sistemática se puede apreciar que la mayoría de investigaciones están dedicadas a la optimización de espacio enfocándose en el volumen de la carga. Mientras que, en menor pero considerable proporción, se tienen investigaciones que buscan evaluar también otros factores como la estabilidad de la carga cuando terminen de ser almacenados todos los productos.

2.4 Formas Aproximadas de resolver el problema

2.4.1 A simple Meta-Heuristic Approach for the Multiple Container Loading Problem

En este documento, el problema de la carga de múltiples contenedores se resuelve mediante un enfoque metaheurístico utilizando un algoritmo voraz, el cual se basará en el concepto de primer ajuste. Esto se realiza con la finalidad de reducir al minino los espacios que puedan generarse entre las cajas al momento de colocarse en los contenedores. También se tienen en cuenta las limitaciones prácticas como la estabilidad de la carga cuando estén dentro de los contenedores.

El algoritmo tiene dos fases:

- En la primera fase, se genera una solución inicial usando heurísticas simples. Esta solución inicial es una secuencia de cajas que indica el orden en el cual deben entrar a los contenedores.
- En la segunda fase, se optimiza la solución inicial usando procedimientos de búsqueda local y de recocido simulado.

La eficacia del enfoque propuesto se muestra mediante la comparación de los resultados obtenidos con respecto a otros enfoques para el problema de carga de múltiples contenedores [TAKAHARA, 2006].

2.4.2 A Hybrid Genetic Approach for Container Loading in Logistic Industry

En el presente documento se propone un enfoque genético hibrido para resolver el problema de carga de contenedores. El problema de carga de contenedores se modelo como un problema Bin-Packing en tres dimensiones, debido a que busca minimizar el desperdicio de espacio y garantizar que cada caja sea estable después de su colocación en el interior del contenedor.

Mediante la combinación de un algoritmo genético con un método de colocación heurístico, el problema de Bin-Packing complicado y con muchas restricciones se transforma en un problema de permutación simple con un dominio de búsqueda más pequeño.

A continuación se muestra el diagrama del flujo para el algoritmo genético hibrido:

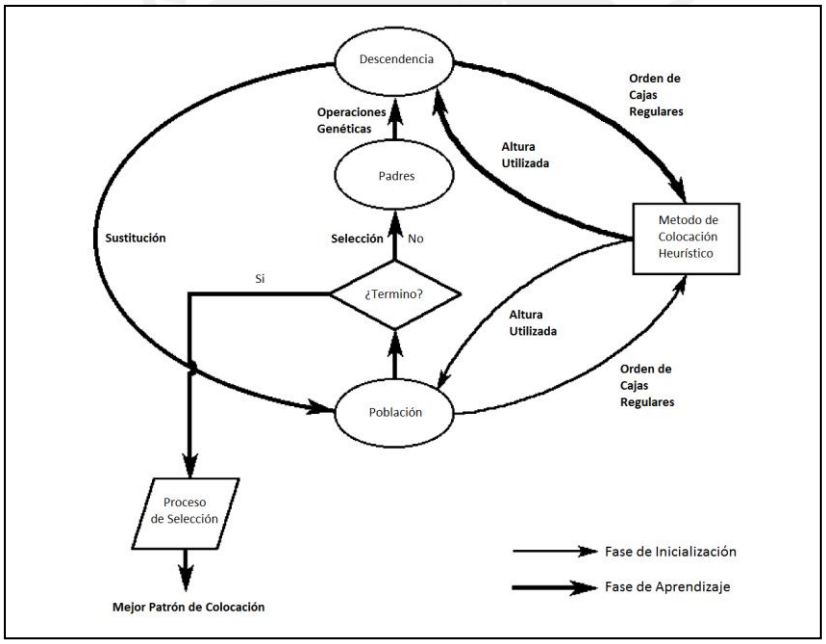


Imagen 7: Diagrama de Flujo para algoritmo Genético Hibrido

Finalmente, se realizan pruebas usando el algoritmo propuesto que demuestran que se pueden obtener resultados óptimos en periodos relativamente cortos [HO, TANG 2005].

2.4.3 *Ant Colony Optimization Algorithm Based on Space Division for Container Loading Problem*

En este documento, se tiene un conjunto de cajas que deben ser cargadas en contenedores rectangulares con dimensiones fijas. Esto se realiza con la finalidad de aprovechar al máximo la relación de utilización del contenedor.

En primer lugar se muestra un modelo matemático. Este modelo matemático presenta una función objetivo buscando maximizar la relación de utilización del contenedor para el problema. Luego, se presentan las restricciones del problema:

- Restricción de Volumen. La suma de los volúmenes de las cajas cargadas no debe exceder al volumen de los contenedores.
- Restricción de Orientación. La forma de colocar una caja en un contenedor está relacionada con las propiedades de la caja en sí misma y la relación de utilización del contenedor.
- Restricción de Rodamiento. La suma de los pesos de las cajas cargadas no debe exceder el volumen efectivo del contenedor.
- Restricción de Capa. El número máximo de apilamientos de cajas que puede haber sobre un determinado tipo de caja recibe el nombre de capa del tipo de caja. En base a esto, se busca que no hayan apilamientos excesivos sobre un determinado tipo de caja y así evitar el deterioro de la misma.

Una vez definida la función objetivo y las restricciones del problema, se desarrolla un algoritmo hibrido integrado con un algoritmo de colonia de hormigas para resolver el problema. Finalmente, los resultados de la simulación demuestran la eficacia del modelo matemático y el algoritmo desarrollado [WANG, et al 2010].

2.4.4 *Two GRASP Metaheuristic for the Capacitated Vehicle Routing Problem Considering Split Delivery and Solving the Three Dimensional Bin Packing Problem*

A diferencia de los documentos anteriores, en este documento se busca resolver el problema de Bin-Packing en tres dimensiones enfocándose en el uso de vehículos repartidores como contenedores. Se plantea resolver este problema mediante la elaboración de un algoritmo GRASP.

El algoritmo GRASP tiene una función objetivo que recibe como parámetros de entrada el ancho, la longitud y la altura tanto de los paquetes como del vehículo donde

se cargaran. La función objetivo busca maximizar el espacio usado en los vehículos repartidores y se usa en la fase de construcción del algoritmo GRASP.

Además, también contara con una función de costo que se calcula en base al volumen de los paquetes cargados a un determinado vehículo, el volumen del espacio de carga del vehículo, la suma de los pesos de todos los paquetes cargados al vehículo y el peso de carga que soporta el vehículo. La función de costo se usa en la fase de mejoramiento del algoritmo GRASP.

A continuación se presenta la estructura básica del algoritmo GRASP [GALLART, TUPIA 2010]:

Procedimiento GRASP
1. $Para\ i = 0\ a\ NumIteraciones\ hacer$
   1.1 $NuevoVehiculo =$
   $FaseConstruccion\ (NuevoVehiculo, ListaPaquetes)$
   1.2 $Si\ i = 0\ entonces$
   1.2.1 ActualizarPaquete (Vehiculo, NuevoVehiculo)
   1.3 Caso Contrario
   1.3.1 $Si\ costo(Vehiculo) < costo(Nuevo\ Vehiculo) entonces$
   $ActualizarPaquete\ (Vehiculo, NuevoVehiculo$
   1.4 $Fin\ Si$
2. $Fin\ Para$
3. $Retornar\ Vehiculo$

Imagen 8: Estructura Básica de algoritmo GRASP Gallart-Tupia

Dónde:

- En la línea 1.1, se procede a ejecutar la fase de construcción del algoritmo GRASP, el cual nos proporciona un vehículo con paquetes que fueron cargados al mismo.
- De la línea 1.2 a la 1.4, la solución obtenida de la fase de construcción es comparada con la mejor solución hasta el momento. Esto se determina en base a la función de costo mencionada anteriormente.
- En la línea 3, se retorna como resultado del procedimiento GRASP la mejor solución obtenida.

2.5 Productos comerciales para resolver el problema

2.5.1 Cube-IQ

Cube-IQ es un software de planeación y optimización de carga. Fue desarrollado por MagicLogic Optimization Inc. Su función principal es la carga de artículos en uno o más contenedores. Esto es realizado teniendo como base la utilización de forma óptima del volumen y del peso de los elementos a introducir en los contenedores.

MagicLogic desarrollo su propio motor de carga para el desarrollo de Cube-IQ, por lo que tiene la ventaja de poder seguir agregando funciones todo el tiempo. En base a esta característica, realiza y tiene varias investigaciones en curso con la finalidad de obtener una mejora continua de su software.

A continuación se mostraran las principales ventajas del software [CUBE-IQ, 2013]:

- Genera planes de carga 3D usando diversos criterios como la secuencia de carga, cargas parciales y la distribución del peso.
- Optimiza cargas bajo apilamiento integral y normas de orientación de caja.
- Almacena y recupera los planes de carga generados por el software.
- Distribuye las instrucciones de carga de los planes de carga generados por el software en diagramas claros y en tres dimensiones.
- Trabaja de forma integral permitiendo usar los planes de carga en otras instalaciones del Cube-IQ.

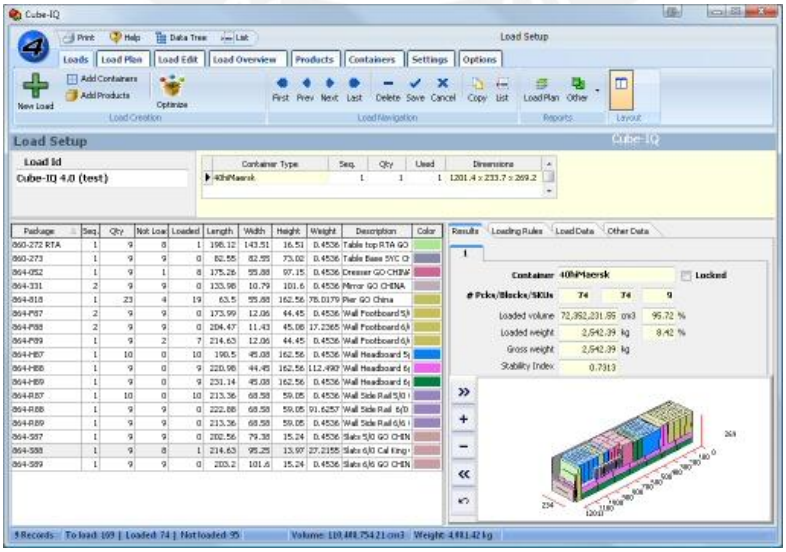


Imagen 9: Software Cube-IQ

2.5.2 PackVol

PackVol es un software de planeación y optimización de carga con una moderna interfaz gráfica. Escrita en C++ posee grandes ventajas en términos de velocidad y capacidad de respuesta de la aplicación. Su objetivo es reducir el desperdicio de espacio con la finalidad de reducir los costos de envió.

Entre sus principales características se pueden encontrar:

- Fácil adaptabilidad a diversos tipos de carga: contenedores, paletas, bastidores, camiones y vagones de ferrocarril.
- Permite al usuario crear y modificar planes de carga ya existentes.
- Fácil integración con sistemas de almacenamiento externos.
- Permite trabajar con la información generada en los planes de carga mediante la elaboración de reportes que servirán para el análisis posterior de las empresas.

La razón principal del éxito de este software es que permite al usuario introducir sus propias reglas para la planeación de la carga. Entre los criterios que permite manejar el software se contempla: generación del plan de carga en varias etapas, contenedores irregulares, peso de carga máximo, priorización de carga y posiciones permitidas [PACKVOL, 2013].

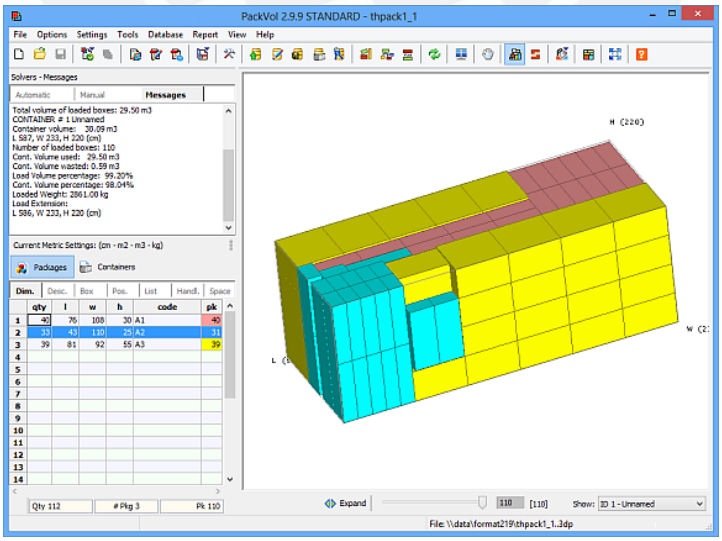


Imagen 10: Software PackVol

2.5.3 CubeMaster

CubeMaster es un software de planeamiento y optimización de carga. Fue desarrollado por Logen Solutions. Su función principal es calcular como cargar productos iguales o mixtos en contenedores, camiones, paletas, cajas y vagones de ferrocarril. Mediante una planificación de carga inteligente busca reducir costos de envió y transporte.

CubeMaste utiliza un algoritmo de optimización de carga publicado en JORS (Journal of Operational Research). Además, posee un para el análisis posterior a la generación del planeamiento de carga que se verán reflejados mediante la elaboración de reportes.

Su principal característica es el manejo de tipos de carga. Estos están clasificados en [LOGEN SOLUTIONS, 2013]:

- Carga Simple. Para productos con la misma forma.
- Carga Mixta. Para productos de diversas formas.
- Carga en conjunto. Se usa para la carga de contenedores especiales que tienen varios espacios adentro del mismo.
- Carga en múltiples conjuntos. Se usa para la carga de varios contenedores especiales.

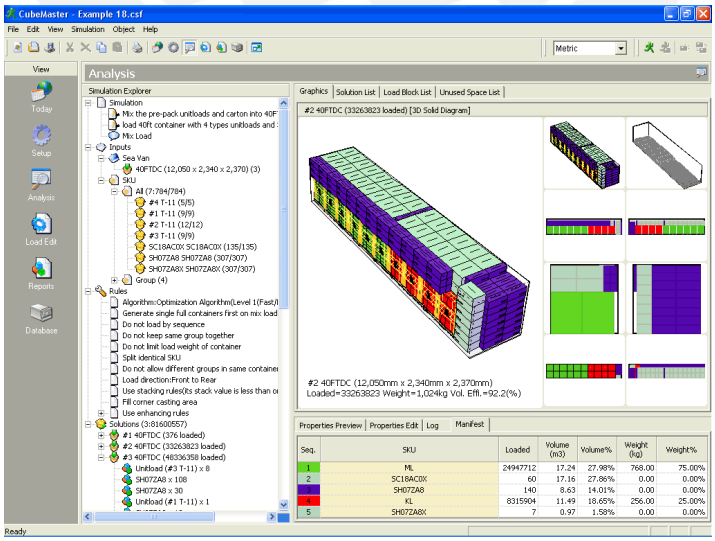


Imagen 11: Software CubeMaster

2.6 Productos de investigación para resolver el problema

2.6.1 Research on solution to complex Container Loading Problem based on Genetic Algorithm

En el presente documento, se estudió el algoritmo genético como solución para el problema de carga de contenedores. Los objetivos que se buscan obtener son los siguientes:

- Maximizar el volumen utilizado en el contenedor.
- Maximizar el peso utilizado en el contenedor.
- Minimizar la altura del centro de gravedad con la finalidad de estabilizar la carga.

En base a lo referido anteriormente, se justificó el uso de un algoritmo genético como un enfoque eficaz para la solución de un problema NP-Completo como el de carga de contenedores. Esto se debe a la robustez, el paralelismo y la variedad de aplicaciones que posee este algoritmo en particular. Finalmente, el resultado de la investigación determinó que el algoritmo genético es efectivo para resolver problemas de carga de contenedores [HE, CHA 2002].

2.6.2 Parallelization of the Multi-Objective Container Loading Problem

El presente estudio realiza un enfoque multi-objetivo para resolver el problema de carga de contenedores. Por lo general, en este tipo de problemas, el desarrollo de la solución se enfoca en optimizar el volumen de total de los productos introducidos en un contenedor. Sin embargo, un factor a considerar en el desarrollo de este problema es el límite de peso que los contenedores pueden soportar para estar en equilibrio.

En base a lo referido anteriormente, el estudio se ha centrado en una formulación multi-objetivo que busca maximizar el volumen de los productos en el contenedor y, al mismo tiempo, no superar el límite de peso señalado anteriormente.

El estudio desarrolla como solución al problema un algoritmo evolutivo. Esto se debe a la gran efectividad de este tipo de algoritmos con otros problemas multi-objetivos del mundo real [DE ARMAS, et al 2012].

2.7 Conclusiones sobre el estado del arte

Una vez presentados algunos métodos para resolver el problema de desperdicio de espacio se observa que se deben considerar muchos factores en el desarrollo de una solución para el mismo. Además, es necesario resaltar que la complejidad del problema aumenta cuando se trata de productos con formas especiales como las tuberías. Por lo tanto, los métodos mostrados durante el estado del arte deben mejorarse si se busca resolver este problema en particular. Por otro lado, cabe resaltar que algunos de los productos comerciales presentados dan la opción de resolver este problema; sin embargo, finalmente, no se tratan de productos desarrollados especialmente para brindar una solución a la medida de las empresas de tuberías. Estos productos agrupan las tuberías dentro de productos de diversas formas sin considerar que su forma circular aumenta el riesgo de que los productos a su alrededor puedan deslizarse y ocasionar accidentes.

En base a esto, el presente proyecto de fin de carrera aporta un método sencillo que permitirá almacenar eficientemente tuberías en las diversas locaciones del almacén. Esto se realizará teniendo como base un apilamiento eficiente de productos en base a criterios como el tamaño, la forma y el peso de los productos. De este modo, se aprovechará al máximo el espacio de almacenamiento y se evitaran posibles pérdidas de productos por mal apilamiento de los mismos, lo cual finalmente generará ganancias a la empresa.

CAPÍTULO 3

En este capítulo se presentan las estructuras de datos y variables involucradas en el desarrollo de los algoritmos GRASP y de búsqueda Tabú.

1 Estructuras de Datos

Para el desarrollo del presente proyecto de fin de carrera se tendrán las siguientes estructuras de datos como la base para el desarrollo de los algoritmos:

- En lo referente a las tuberías, se tiene una estructura de datos contiene la información de una determinada tubería que desea almacenarse. Esta estructura presentará los siguientes campos:

Nombre	Representación	Descripción
Material	M	Este campo es un texto que indica el tipo de material en base al cual ha sido desarrollada la tubería. Por ejemplo.- Hierro, Polietileno y PVC (Cloruro de Polivinilo).
Radio	RT	Este campo es un número que indica el radio externo que maneja la circunferencia de la tubería.
Largo	LT	Este campo es un número que indica el largo de la tubería.
Peso	P	Este campo es un número que indica el peso de la tubería.
Fragilidad	F	Este campo es un número que indica el máximo peso que puede soportar la tubería.

Además, el número de tuberías utilizadas está representado como B. En base a esto, para representar las tuberías con las que se trabajará se tendrá la siguiente **Matriz T [T_{Bx6}]**:

$\boldsymbol{T}$	1	2	3	…	B
$\boldsymbol{M}$	PVC	Hierro	Polietileno	…	Hierro
$\boldsymbol{RT}$	10	5	8	…	7
$\boldsymbol{LT}$	120	250	180	…	200
$\boldsymbol{P}$	6	4	10	…	5
$\boldsymbol{F}$	12	7	25	…	8

- En lo referente a los contenedores, se tiene estructura de datos contiene la información de un determinado espacio de almacenamiento de tuberías. Esta estructura presentará los siguientes campos:

Nombre	Representación	Descripción
Alto	ALC	Este campo es un número que indica la altura del contenedor.
Ancho	ANC	Este campo es un número que indica el ancho del contenedor.
Largo	LC	Este campo es un número que indica el largo del contenedor.
Factor de Uso	FU	Este campo es un número que indica si el contenedor ya se encuentra siendo usado en una determinada solución. Inicialmente, tendrá el valor de 1 pero cuando empiece a ser usado para una solución su valor cambiara a 500.

Además, el número de contenedores utilizado está representado como A. En base a esto, para representar los contenedores con los que se trabajará se tendrá la siguiente **Matriz C [$\boldsymbol{C_{Ax5}}$]**:

$\boldsymbol{C}$	1	2	3	…	A
$\boldsymbol{ALC}$	200	500	600	…	150
$\boldsymbol{ANC}$	200	400	600	…	150
$\boldsymbol{LC}$	400	700	1000	…	300
$\boldsymbol{FU}$	1	1	1	…	1

- En lo referente a la solución, una vez definidas las estructuras con las que se trabajará la información de las tuberías y los contenedores es necesario definir una estructura con la que se podrá manejar la relación entre estas; es decir, la asignación de una tubería a un contenedor como solución. Esta estructura presentará los siguientes campos:

Nombre	Representación	Descripción
Contenedor	C	Este campo es un número que indica un determinado contenedor.
Tubería	T	Este campo es un número que indica una determinada tubería.
Ubicación X	UX	Este campo es un número que indica la posición de la tubería en el contenedor asignado respecto a su eje X (ancho).
Ubicación Y	UY	Este campo es un número que indica la posición de la tubería en el contenedor asignado respecto a su eje Y (alto).
Ubicación Z	UZ	Este campo es un número que indica la posición de la tubería en el contenedor asignado respecto a su eje Z (largo).

En base a esto, para representar en que contenedores han sido asignadas las tuberías se tendrá la siguiente **Matriz S [S_{Bx5}]**:

$\boldsymbol{C}$	1	1	1	…	2
$\boldsymbol{T}$	1	2	3	…	B
$\boldsymbol{UX}$	10	25	8	…	60
$\boldsymbol{UY}$	10	5	8	…	60
$\boldsymbol{UZ}$	0	0	120	…	80

- En lo referente al algoritmo de búsqueda Tabú, el manejo de los movimientos realizados será analizado no solo entre tuberías sino también entre contenedores. Esto se debe a que la solución muestra la relación entre los contenedores y las tuberías que han sido asignadas a los mismos.

No es lo mismo hacer un movimiento entre dos tuberías que se encuentran en un mismo contenedor que entre dos tuberías que se encuentran en distintos contenedores. En el primer caso, la función objetivo no se ve afectada ya que el desperdicio de espacio no varía en el contenedor en el que se encuentran las dos tuberías, mientras que esto si puede suceder en el segundo caso.

En base a esto, el algoritmo de búsqueda Tabú dispondrá de tres estructuras que le permitan el manejo de los movimientos entre las tuberías ubicadas en distintos contenedores. Además, el número de combinaciones entre tuberías y contenedores (*AxB*) está representado como *C*.

En primer lugar, el manejo de la **memoria basada en lo reciente** durante la ejecución del algoritmo será realizado mediante la siguiente **Matriz MR [MR_{CxC}]**:

	C	**1**	**1**	**1**	**1**	**2**	**2**	**2**	**2**	**…**	**…**	**…**	**…**	**A**	**A**	**A**	**A**
	T	**1**	**2**	**…**	**B**	**1**	**2**	**…**	**B**	**…**	**…**	**…**	**…**	**1**	**2**	**…**	**B**
C	**T**		**1**	**2**	**..**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**C**
1	**1**	**1**					0	0	0	0	0	0	0	0	0	0	0	0
1	**2**	**2**					0	1	0	10	0	0	5	0	0	0	9	0
1	**…**	**…**					0	0	0	0	0	0	0	0	0	0	0	0
1	**B**	**…**					0	0	0	0	0	0	0	8	0	0	0	0
2	**1**	**…**									0	0	0	0	0	0	0	6
2	**2**	**…**									0	0	0	0	0	0	0	0
2	**…**	**…**									0	4	0	7	0	0	0	0
2	**B**	**…**									0	0	0	0	0	2	0	0
…	**…**	**…**													0	0	0	0
…	**…**	**…**													0	0	0	0
…	**…**	**…**													0	0	0	3
…	**…**	**…**													0	0	0	0
A	**1**	**…**																
A	**2**	**…**																
A	**…**	**…**																
A	**B**	**C**																

El manejo de la matriz es el siguiente: La posición 1 de una de las dimensiones del arreglo representa la tubería 1 colocada en el contenedor 1. Del mismo modo, la posición 2 de una de las dimensiones del arreglo representa la tubería

2 colocada en el contenedor 2. Esto se repite hasta completar todas las posibles combinaciones de tuberías con el contenedor 1. Luego, la siguiente posición de una de las dimensiones de la matriz será para la tubería 1 pero ahora colocada en el contenedor 2. Esto se repetirá hasta ubicar todas las posibles combinaciones de tuberías y contenedores.

Además, es necesario considerar que la matriz no permite movimientos en los que no varié el contenedor. Esto se debe a que, como se explicó anteriormente, si se mueven dos tuberías en un mismo contenedor, el espacio aprovechado por el contenedor no cambiará.

Finalmente, en base a esta matriz, se podrá penalizar los últimos movimientos realizados durante la ejecución del algoritmo de búsqueda Tabú.

En segundo lugar, el manejo de la **memoria basada en frecuencia** durante la ejecución del algoritmo será realizado mediante la siguiente **Matriz MF [MF_{CxC}]:**

	C	1	1	1	1	2	2	2	2	…	…	…	…	A	A	A	A
	T	1	2	…	B	1	2	…	B	…	…	…	…	1	2	…	B
C	T		1	2	..	…	…	…	…	…	…	…	…	…	…	…	…	C
1	1	1					21	0	0	0	16	0	0	25	0	9	0	8
1	2	2					0	1	0	17	0	0	20	0	7	0	1	0
1	…	…					0	0	1	0	0	15	0	0	0	0	18	0
1	B	…					1	0	0	15	0	0	0	16	0	0	0	0
2	1	…									0	21	0	0	0	24	0	1
2	2	…									0	0	5	6	0	0	0	0
2	…	…									0	2	0	8	0	0	27	0
2	B	…									0	0	0	0	0	19	0	0
…	…	…													23	0	0	1
…	…	…													0	0	0	30
…	…	…													12	0	0	1
…	…	…													0	19	0	17
A	1	…																
A	2	…																
A	…	…																
A	B	C																

El manejo de esta matriz es igual a la anterior pero su funcionamiento es distinto. Esto se debe a que, en base a esta matriz, se podrá tener un historial de la cantidad de veces que se realizó un determinado movimiento durante la ejecución del algoritmo de búsqueda Tabú.

Finalmente, **para determinar qué movimiento se realizará** en una determinada iteración se dispondrá de una estructura que permita determinar cuál es el mejor movimiento para la solución actual. Esta estructura presentará los siguientes campos:

Nombre	Representación	Descripción
Primera Tubería	PT	Este campo es un número que indica una determinada tubería.
Primer Contenedor	PC	Este campo es un número que indica un determinado contenedor.
Segunda Tubería	ST	Este campo es un número que indica una determinada tubería.
Segundo Contenedor	SC	Este campo es un número que indica un determinado contenedor.
Espacio Aprovechado	V	Este campo es un número que indica el valor de la función objetivo luego de aplicar un determinado movimiento. Inicialmente es -1.
Espacio Aprovechado Penalizado	VP	Este campo es un número que indica el valor de la variable Espacio Aprovechado luego de aplicarle una penalización en base a la frecuencia con que se ha realizado el movimiento seleccionado. Inicialmente es -1.
Factor Tabú	FT	Este campo es un número que indica si un movimiento es tabú o no. Inicialmente tendrá el valor de 0 pero cuando se realice el movimiento su valor pasará a 1. Solo cuando termine su tiempo de penalización es que este valor regresará a ser 0.

Además, el número de movimientos posibles está representado como D. En base a esto, para determinar qué movimiento se realizara en una determinada iteración se tendrá la siguiente **Matriz MM [MM_{Dx7}]**:

PT	1	10	1	…	A
PC	1	6	3	…	B
ST	3	5	3	…	A
SC	7	1	8	…	B
V	0.7	0.8	0.6	…	-1
VP	0.4	0.3	0.6	…	-1
FT	1	0	1	…	0

2 Variables

Para el desarrollo del presente proyecto de fin de carrera se tendrán en cuenta las siguientes variables para el desarrollo de los algoritmos.

En el caso de las variables iniciales se tendrán:

- **Número de iteraciones.** Esta variable es un número que indica el número de iteraciones que realizan los algoritmos. Esto permitirá manejar el tiempo de ejecución de los algoritmos a desarrollar. El número de iteraciones que se utiliza está representado como *N*.
- **Lista de Tuberías.** Esta variable es una **Matriz T** que contiene las tuberías que se desean almacenar. La lista de tuberías está representada como *T*.
- **Lista de Contenedores.** Esta variable es una **Matriz C** que contiene los contenedores con los espacios disponibles dentro del almacén para almacenar las tuberías que vienen del área de producción. La lista de contenedores está representada como *C*.

Para las variables a utilizar en el algoritmo GRASP se tendrán:

- **Contenedores Disponibles.** Esta variable es una **Matriz C** que contiene los contenedores que dispone el algoritmo GRASP para almacenar las tuberías. La lista de contenedores disponibles está representados como *C*.

- **Tuberías Disponibles.** Esta variable es una **Matriz T** que contiene las tuberías que se disponen en cada iteración del algoritmo GRASP. La lista de tuberías disponibles está representada como *T*.
- **Constante de Relajamiento.** Esta variable es un número decimal que contiene la constante de relajamiento utilizada para la elaboración del RCL. Este número se encontrará entre 0 y 1, y su valor dependerá de si se busca un enfoque aleatorio (valor de alfa cercano a uno) o goloso (valor de alfa cercano a cero). La constante de relajamiento está representada como α.
- **Mejor Valor.** Esta variable es un número que contiene el valor de la tubería o contenedor con las mejores capacidades de almacenamiento. Estas capacidades de almacenamiento serán evaluadas en base a la función objetivo correspondiente. Esto se debe a que se tendrá dos variables de mejor valor: una para determinar el mejor valor de la tubería y otra, para el mejor valor del contenedor. El mejor valor está representada como β.
- **Peor Valor.** Esta variable es un número que contiene el valor de la tubería o contenedor con las peores capacidades de almacenamiento. Estas capacidades de almacenamiento serán evaluadas en base a la función objetivo correspondiente. Esto se debe a que se tendrá dos variables de peor valor: una para determinar el peor valor de la tubería y otra, para el peor valor del contenedor. La peor tubería está representada como τ.
- **Lista Restringida de Candidatos (RCL)**. Esta variable es una **Matriz T** que contiene todas las tuberías o contenedores disponibles que se encuentren en un rango determinado por el peor y el mejor valor (evaluadas por la función objetivo correspondiente) y la constante de relajamiento descrita anteriormente. La lista restringida de candidatos está representada como *RCL*.
- **Contenedor Seleccionado.** Esta variable es un número que indica un contenedor seleccionado aleatoriamente del RCL de contenedores. El contenedor seleccionado está representado como *CS*.
- **Tubería Seleccionada**. Esta variable es un número que indica una tubería seleccionada aleatoriamente del RCL de tuberías. La tubería seleccionada está representada como *TS*.
- **Solución Implementada**. Esta variable es una **Matriz S** que contiene las relaciones entre los contenedores y las tuberías asignadas en una solución desarrollada durante una de las iteraciones del algoritmo GRASP. Estos contenedores contienen las tuberías que fueron colocadas en cada uno de

ellos como propuesta de esa solución. La solución implementada en cada iteración está representada como *S*.

- **Espacio Aprovechado.** Esta variable es un número decimal entre 0 y 1 que indica el espacio aprovechado por una solución implementada en una iteración del algoritmo GRASP en base a las tuberías y contenedores usados en la misma. El espacio aprovechado está representado como *Espacio_Aprovechado*.

Para las variables a utilizar en el algoritmo de búsqueda Tabú se tendrán:

- **Solución Implementada.** Esta variable es una **Matriz S** que contiene las relaciones entre los contenedores y las tuberías asignadas en la solución implementada en una iteración del algoritmo de búsqueda Tabú. La solución implementada está representada como *S*.
- **Espacio Aprovechado Implementado**. Esta variable es un número decimal entre 0 y 1 que indica el espacio aprovechado por una solución implementada en una iteración del algoritmo de búsqueda Tabú. El espacio aprovechado implementado está representado como *Espacio_Aprovechado*.
- **Iteración de Frecuencia**. Esta variable es un número que indica a partir de que iteración del algoritmo de búsqueda Tabú se empezará a hacer uso de la memoria basada en frecuencia. La iteración de frecuencia está representada como *N_Frecuencia*.
- **Periodo Tabú**. Esta variable es un número que indica cuantas iteraciones estará un movimiento considerado como Tabú luego de ser realizado. El periodo tabú está representado como *PT*.
- **Memoria basada en lo Reciente**. Esta variable es una **Matriz MR** que contiene los movimientos entre las tuberías y los contenedores que son considerados Tabú para las iteraciones del algoritmo. La memoria basada en lo reciente está representada como *MR*.
- **Memoria basada en frecuencia.** Esta variable es una **Matriz MF** que contiene la frecuencia con la que fueron realizados los movimientos entre las tuberías y contenedores durante las iteraciones del algoritmo. La memoria basada en frecuencia está representada como *MF*.
- **Vecindario de Soluciones**. Esta variable es una **Matriz MM** que contiene todos los posibles movimientos que pueden realizar el algoritmo para mejorar la solución actual y sus características: valor del espacio aprovechado,

penalización en base a frecuencia y si es un movimiento tabú o no. El vecindario de soluciones está representado como *MM*.

- **Movimiento Seleccionado**. Esta variable es un número que indica el mejor movimiento seleccionado en una determinada iteración del algoritmo. El movimiento seleccionado está representado como *MS*.
- **Mejor Solución.** Esta variable es una **Matriz S** que contiene las relaciones entre los contenedores y las tuberías asignadas en la mejor solución obtenida por el algoritmo de búsqueda Tabú. La mejor solución está representada como *S_Mejor*.
- **Mejor Espacio Aprovechado.** Esta variable es un número decimal entre 0 y 1 que indica el espacio aprovechado por la mejor solución implementada del algoritmo de búsqueda Tabú. El mejor espacio aprovechado está representado como *Espacio_Aprovechado_Mejor*.

CAPÍTULO 4

1 Función Objetivo

En esta sección se presenta la función objetivo a ser utilizada por los algoritmos GRASP y de búsqueda Tabú. La función objetivo permite evaluar la calidad de la solución en base a la optimización de la misma. Esta optimización puede ser minimización o maximización dependiendo del problema a resolver.

El presente proyecto de fin de carrera tiene como objetivo minimizar el desperdicio de espacio de los espacios de almacenamiento disponibles, lo cual se puede interpretar también como maximizar el espacio ocupado por las tuberías dentro de los contenedores disponibles. Por lo tanto, la función objetivo es la siguiente:

$$Max\ (\frac{\sum_{j=1}^{B} 2 * \pi * RT_{ij} * LT_{ij}}{ALC_i * ANC_i * LC_i})$$

Dónde:

- ALC_i: Alto del contenedor i.
- ANC_i: Ancho del contenedor i.
- LC_i: Largo del contenedor i.
- RT_{ij}: Radio de la tubería j ubicada dentro del contenedor i.
- LT_{ij}: Largo de la tubería j ubicada dentro del contenedor i.

La función objetivo maneja en el numerador la suma de todos los espacios ocupados por cada una de las tuberías que fueron asignadas a un contenedor mientras que en el denominador se tiene el espacio total del cual dispone el contenedor. En base a esta función objetivo se podrá determinar cuál es solución la más óptima.

En el caso del algoritmo de búsqueda Tabú, durante cada iteración se irá implementando una nueva solución en base a una solución inicial que será alterada mediante un determinado movimiento. Para evaluar si la nueva solución es mejor que la mejor solución, se usará la función objetivo mencionada.

En el caso del algoritmo GRASP, durante cada iteración se irá generando una solución la cual será evaluada respecto a la mejor solución actual para determinar si es mejor o no, para esta evaluación será necesario el uso de la función objetivo mencionada. Sin embargo, dentro del algoritmo GRASP, para escoger qué combinación de tuberías y contenedores forman parte de la solución, deben diseñarse otras dos funciones objetivo.

Esto se debe a que el algoritmo trabajará con dos tipos diferentes de elementos en base a los cuales forma la solución: las tuberías y los contenedores. Además, si solo se utiliza el espacio de almacenamiento como criterio de selección, no se está considerando que las tuberías que se encuentran ubicadas en la base deban ser más largas. Sin este criterio de selección, se puede dar el caso que se escoja una tubería con un radio muy grande pero un largo pequeño como base y si luego se coloca encima tuberías con largos más grandes inevitablemente estos terminaran perdiendo el equilibrio.

Para solucionar este problema, se le multiplicará el largo de la tubería y el contenedor nuevamente en la función objetivo de cada uno. Este valor extra del largo de la tubería y del contenedor será un peso que se le dará para que el criterio de selección elija una tubería o un contenedor que no solo ocupe un considerable espacio de almacenamiento sino también un considerable largo. De este modo, se tendrá una base sólida para las tuberías que sean colocadas en los niveles más altos y a su vez se dispondrá de los contenedores con mayores largos para almacenar estas tuberías.

Además, como se le debe dar prioridad a los contenedores que ya están siendo usados, en la función objetivo de los contenedores se le multiplicará además una variable llamada Factor de Uso (*FU*). Esta variable tendrá el valor de 1 cuando el contenedor no haya sido utilizado en la solución. Del mismo modo, tendrá el valor de 500 cuando si sea usado. De este modo, se asegura el uso de un contenedor hasta el momento en que ya no se puedan introducir más tuberías.

Por lo tanto, la función objetivo para las tuberías en el algoritmo GRASP es la siguiente:

$$Max\ (2 * \pi * RT_i * {LT_i}^2)$$

Dónde:

- RT_i: Radio de la tubería i.
- LT_i: Largo de la tubería i.

Del mismo modo, la función objetivo para los contenedores en el algoritmo GRASP es la siguiente:

$$Max\ (ALC_i * ANC_i * {LC_i}^2 * FU_i)$$

Dónde:

- ALC_i: Alto del contenedor i.
- ANC_i: Ancho del contenedor i.
- LC_i: Largo del contenedor i.
- FU_i: Factor de Uso del contenedor i.

Finalmente, es necesario tener una consideración. Si bien es cierto que el manejo del peso que soportará una tubería no será considerado en la función objetivo, si será considerado como una restricción para determinar si una solución propuesta es viable o no. En otras palabras, cada tubería no debe tener encima tantas tuberías que la suma del peso de estas supere la fragilidad de la tubería.

CAPÍTULO 5

En este capítulo se presenta el algoritmo GRASP desarrollado en el presente proyecto de fin de carrera. Este algoritmo ha sido desarrollado con la finalidad de generar una solución inicial que pueda ser utilizada por el algoritmo de búsqueda Tabú. Además, también se utiliza como punto de comparación con el algoritmo propuesto a través de la experimentación numérica.

1 Variables del Algoritmo GRASP

Las variables a utilizar en el algoritmo GRASP son:

Variable	Descripción
N	Número de iteraciones del algoritmo.
C	Tuberías disponibles para el algoritmo.
T	Contenedores disponibles para el algoritmo.
$\propto$	Constante de Relajamiento utilizada en la elaboración del algoritmo.
$\boldsymbol{\beta_1}$	Mejor valor de contenedores.
$\boldsymbol{\tau_1}$	Peor valor de contenedores.
$\boldsymbol{RCL_1}$	Lista restringida de candidatos de contenedores.
$\boldsymbol{\beta_2}$	Mejor valor de tuberías.
$\boldsymbol{\tau_2}$	Peor valor de tuberías.
$\boldsymbol{RCL_2}$	Lista restringida de candidatos de tuberías.
CS	Contenedor seleccionado del RCL de contenedores.
TS	Tubería seleccionada del RCL de tuberías.
S	Solución implementada por el algoritmo.
Espacio_Aprovechado	Valor de la función objetivo de la solución implementada por el algoritmo.

2 Presentación del Algoritmo GRASP

El algoritmo GRASP que se desarrollará solo estará en fase de construcción, la cual será iterada *N* veces para conseguir la solución inicial que recibirá el algoritmo de búsqueda Tabú. También es necesario resaltar que este algoritmo GRASP presenta doble relajación ya que maneja dos listas de candidatos: una lista de tuberías y una lista de contenedores.

Inicio Algoritmo GRASP Construcción Tuberías (C, T, S, α)

1. $Inicializar\ Matrices\ (C,T)$
2. $Espacio\_Aprovechado = \emptyset$
3. $Inicializar\ Matriz\ Solución\ (S)$
4. $Mientras\ exitanTuberiasSinAlmacenar(T,S)\ hacer$
   $Inicio$
   4.1 $CS = \emptyset$
   4.2 $TS = \emptyset$
   4.3 $\beta_1 = Max\ (ALC_i * ANC_i * {LC_i}^2 * FU_i\ \forall C_i)$
   4.4 $\tau_1 = Min\ (ALC_i * ANC_i * {LC_i}^2 * FU_i\ \forall C_i)$
   4.5 $RCL_1 = \{C_i\ /\ \beta_1 - \alpha * (\beta_1 - \tau_1) \leq ALC_i * ANC_i * {LC_i}^2 * FU_i \leq \beta_1\}$
   4.6 $\beta_2 = Max\ (2 * \pi * RT_i * {LT_i}^2\ \forall T_i \notin S)$
   4.7 $\tau_2 = Min\ (2 * \pi * RT_i * {LT_i}^2\ \forall T_i \notin S)$
   4.8 $RCL_2 = \{T_i \notin S\ /\ \beta_2 - \alpha * (\beta_2 - \tau_2) \leq 2 * \pi * RT_i * {LT_i}^2 \leq \beta_2\}$
   4.9 $Mientras\ noSePuedeUbicarTuberia(TS,CS,T,C,S)\ y$
   $afectaFragilidad(TS,CS,T,C,S)hacer$
   $Inicio$
   4.9.1 $CS = Aleatorio(RCL_1)$
   4.9.2 $TS = Aleatorio(RCL_2)$
   $Fin\ Mientras$ 5.9
   4.10 $AgregarElementosSeleccionados(TS,CS,T,C,S)$
   $Fin\ Mientras$ 3.3
5. $Espacio\_Aprovechado = \sum_{i=1}^{A} \frac{\sum_{j=1}^{B} 2*\pi*RT_{ij}*LT_{ij}}{ALC_i*ANC_i*LC_i}$
6. $Retornar\ S, Espacio\_Aprovechado$

Fin GRASP Construcción Tuberías

Dónde:

- En el punto 1, se inicializan las variables C (Matriz C) y T (Matriz T) con la información de las tuberías y contenedores a utilizar. Esta información se obtendrá de la lectura de un archivo.

- En los puntos 2 y 3, se inicializan las variables que contendrán la solución que devolverá el algoritmo.
- En el punto 4, se ejecuta un bucle el cual terminará cuando todas las tuberías sean colocadas en un determinado contenedor.
    - En los puntos 4.1 y 4.2, se inicializan las variables que manejarán la tubería y el contenedor seleccionados de las listas de candidatos respectivas.
    - Del punto 4.3 al 4.5, se obtienen el mejor y el peor valor de los contenedores en base a la función objetivo para los contenedores. Con estas variables se formará el RCL de contenedores.
    - Del punto 4.6 al 4.8, se obtienen el mejor y el peor valor de las tuberías en base a la función objetivo para las tuberías. Con estas variables se formará el RCL de tuberías.
    - En el punto 4.9, se ejecuta un bucle el cual terminará cuando la tubería seleccionada pueda ser ubicada dentro del contenedor seleccionado. Además, también se verifica que la tubería seleccionada no afecte la fragilidad de las otras tuberías que ya se encuentren dentro del contenedor seleccionado.
        - En el punto 4.9.1, se selecciona aleatoriamente un contenedor del RCL de contenedores.
        - En el punto 4.9.2, se selecciona aleatoriamente una tubería del RCL de tuberías.
    - En el punto 4.10, la tubería seleccionada y el contenedor seleccionado que lo contendrá son agregados a la solución que se está desarrollando. Además, esta solución también comprende que se colocará la ubicación (UX, UY, UZ) de la tubería seleccionada dentro del contenedor seleccionado.
- En el punto 5, se calcula el espacio que ha sido aprovechado por la solución en base al espacio aprovechado por las tuberías que hay en cada contenedor que se usó en la solución.
- En el punto 6, se obtiene la solución que representa todos los contenedores y tuberías utilizados en la solución. Además, también se obtiene el espacio aprovechado por la solución.

CAPÍTULO 6

En este capítulo se presenta el algoritmo de búsqueda Tabú desarrollado con la finalidad de desarrollar el presente proyecto de fin de carrera.

1 Variables del Algoritmo de búsqueda Tabú

Las variables a utilizar en el algoritmo de búsqueda Tabú son:

Variable	Descripción
N	Número de iteraciones del algoritmo.
C	Contenedores disponibles para el algoritmo.
T	Tuberías disponibles para el algoritmo.
S	Solución implementada por el algoritmo.
Espacio_Aprovechado	Valor de la función objetivo de la solución implementada por el algoritmo.
N_Frecuencia	Número de iteración a partir de la cual se aplica la memoria basada en frecuencia.
PT	Periodo que permanece un movimiento como prohibido luego de ser realizado.
MR	Lista con los movimientos entre tuberías que son considerados tabú.
MF	Lista con la frecuencia con la que se han realizado los movimientos entre tuberías.
MM	Lista con todos los posibles movimientos que puede realizar la solución actual.
MS	Movimiento seleccionado de la lista de posibles movimientos.
S_Mejor	Mejor solución implementada por el algoritmo.
Espacio_Aprovechado_Mejor	Valor de la función objetivo de la mejor solución implementada por el algoritmo.

2 Lineamiento Generales del Algoritmo de Búsqueda Tabú

Antes de presentar el algoritmo, es necesario explicar los principales conceptos y como serán manejados por el algoritmo de búsqueda Tabú desarrollado.

2.1 Memoria basada en lo Reciente

El algoritmo de búsqueda Tabú desarrollado trabajará la memoria basada en lo reciente mediante el uso de la **Matriz MR**. Esta matriz permitirá el control de los movimientos recientes que se hayan producido durante la ejecución del algoritmo. Este control se realiza mediante la asignación de un periodo Tabú a estos movimientos realizados.

El periodo Tabú es un número que indica cuantas iteraciones se prohíbe realizar el movimiento nuevamente. Cuando en determinada iteración se realice un movimiento, la matriz actualiza este movimiento colocándole el valor del periodo Tabú. Este valor ira decreciendo en uno en cada iteración del algoritmo y cuando llegue a cero recién se permite su uso nuevamente.

El valor del periodo Tabú se determinará calculando la raíz cuadrada de todas las combinaciones posibles de tuberías y contenedores. El periodo Tabú se representa de la siguiente forma:

$$PT = \sqrt{A * B}$$

Dónde:

- *A*: Número de Contenedores.
- *B*: Número de Tuberías.

2.2 Memoria basada en frecuencia

El algoritmo de búsqueda Tabú desarrollado trabajará la memoria basada en frecuencia mediante el uso de la **Matriz MF**. Esta matriz permitirá saber cuántas veces fue realizado determinado movimiento durante la ejecución del algoritmo. Este control promueve la búsqueda de nuevos espacios de soluciones mediante la penalización al valor de la función objetivo aplicada a una determinada solución.

Cada solución tiene una variable Espacio Aprovechado que es el valor de la función objetivo sobre esa solución. En base a la memoria basada en la frecuencia se dispondrá además de una variable llamada Espacio Aprovechado Penalizado que es el valor del espacio aprovechado luego de penalizar por ejecutar cierto movimiento que fue realizado un determinado número de veces.

El valor de penalización se determinará multiplicando el espacio aprovechado de la solución por la razón entre el número de veces que fue ejecutado un movimiento determinado y cantidad de movimientos realizados durante la ejecución del algoritmo (en otras palabras, la cantidad de iteraciones realizadas hasta el momento).

El cálculo de este valor se representa de la siguiente forma:

$$VP = (1 - \frac{MF_{ij}}{n}) * V$$

Dónde:

- *VP*: Espacio Aprovechado Penalizado.
- MF_{ij}: Número de veces que fue realizado el movimiento entre la tubería y el contenedor ubicados en el punto i y la tubería y el contenedor ubicados en el punto j de la matriz MF.
- *n*: Número de iteraciones realizadas hasta el momento.
- *V*: Espacio Aprovechado de la solución.

También es necesario precisar que la memoria basada en frecuencia solo será ejecutada a partir de una determinada iteración. El número de iteración en el que comenzará a realizarse se asigna al inicio del algoritmo.

2.3 Criterio de Aspiración

Si bien es cierto que la matriz MR permite manejar que movimientos serán tabú en cada iteración, se debe tener en cuenta que esta restricción no es inviolable bajo todas las circunstancias. A esto se le denomina Criterio de Aspiración.

El algoritmo de búsqueda Tabú desarrollado trabajará con el siguiente criterio de aspiración: **Cuando un movimiento tabú resulta en una solución mejor que cualquiera visitada hasta el momento, se elimina su clasificación como tabú**.

Esto se debe a que será un movimiento de alta influencia que conducirá a una mejora significativa de la solución.

3 Presentación del Algoritmo de búsqueda Tabú

3.1 Procedimiento principal del Algoritmo de Búsqueda Tabú

Este procedimiento muestra la estructura principal del algoritmo de búsqueda Tabú.

Inicio Algoritmo de Búsqueda Tabú Tuberías (*C, T, N*)

1. $Inicializar\ Variables\ Iniciales\ (C, T, N)$
2. $S, Espacio\_Aprovechado = Generar\ Solución\ GRASP\ (C, T, N)$
3. $Inicializar\ Variables\ Tabú\ (MM, MR, MF, N\_Frecuencia, PT)$
4. $S\_Mejor = S$
5. $Espacio\_Aprovechado\_Mejor = Espacio\_Aprovechado$
6. $Para\ n = 1\ a\ N$

   $Inicio$

   6.1 $GenerarVecindario(MM, MR, MF, N\_Frecuencia, n, C, T, S)$

   6.2 $MS = EscogerMejorMovimiento(MM)$

   6.3 $RealizarMovimiento(S, Espacio\_Aprovechado, MM, MS)$

   6.4 $ActualizarMejorSolucion(S, Espacio\_Aprovechado, S\_Mejor, Espacio\_Aprovechado\_Mejor)$

   6.5 $ActualizarEstructurasTabu(MM, MR, MF, PT, MS)$

   $Fin\ Para\ 6$
7. $Retornar\ S\_Mejor, Espacio\_Aprovechado\_Mejor$

Fin Algoritmo de búsqueda Tabú Tuberías

Dónde:

- En el punto 1, se inicializan las variables iniciales del problema. Las variables C (Matriz C) y T (Matriz T) reciben la información de las tuberías y los contenedores a utilizar mediante la lectura de un archivo. El número de iteraciones (N) que tendrá el algoritmo se define también en esta parte.

- En el punto 2, se genera una solución inicial para el algoritmo de búsqueda Tabú en base al uso del algoritmo GRASP definido en la sección anterior.
- En el punto 3, se inicializan las variables que permiten el desarrollo del algoritmo de búsqueda Tabú durante las iteraciones del mismo.
- En los puntos 4 y 5, se inicializa la mejor solución del algoritmo como la solución generada por el algoritmo GRASP. Además, también se almacena el espacio aprovechado por esta solución.
- En el punto 6, se ejecuta un proceso que se repetirá hasta que se hayan completado las N iteraciones definidas para el algoritmo.
    - En el punto 6.1, se genera el vecindario de soluciones con todos los posibles movimientos que puede realizar la solución actual.
    - En el punto 6.2, se realiza la elección del mejor movimiento en base al vecindario de soluciones generado en el punto 6.1.
    - En el punto 6.3, se procede a actualizar los contenedores de las tuberías que fueron intercambiados en base al movimiento seleccionado (MS) en el punto 6.2. Además, una vez actualizada la solución actual se procede a calcular nuevamente su espacio aprovechado. Este se debe a que el movimiento realizado puede haber cambiado su valor.
    - En el punto 6.4, se actualiza la mejor solución si el espacio aprovechado por la nueva solución es mejor que el espacio aprovechado por la mejor solución encontrada hasta el momento.
    - En el punto 6.5, se actualizan las variables que sirven para el desarrollo del algoritmo en base al movimiento seleccionado en el punto 6.2.
- En el punto 7, se obtiene la solución del algoritmo de búsqueda Tabú que representa todos los contenedores y tuberías utilizados. Además, también se obtiene el espacio aprovechado por la solución.

3.2 Procedimiento GenerarVecindario

GenerarVecindario (*MM,MR,MF,N_Frecuencia,n,C,T,S*)
Inicio

1. $Para\ i = 1\ a\ D$
   $Inicio$
   1.1 $S\_Aux = S$
   1.2 $GenerarSolucionesFactibles(MM, i, T, C, S\_Aux)$
   1.3 $AplicarMemoriaBasadaEnLoReciente(MM, i, MR)$
   1.4 $AplicarMemoriaBasadaEnFrecuencia(MM, i, MF, N\_Frecuencia, n)$
   1.5 $AplicarCriterioDeAspiracion(MM, i, Espacio\_Aprovechado\_Mejor)$
   $Fin\ Para\ 1$

Fin

Dónde:

- En el punto 1, se ejecuta un proceso que se repetirá hasta que se hayan recorrido todos los posibles movimientos (D) que se encuentran en la matriz MM.
  - En el punto 1.1, se asigna la solución actual a una solución auxiliar. Esta variable servirá para analizar un determinado movimiento (i) y determinar si es factible o no realizarlo.
  - En el punto 1.2, se evalúa si se puede realizar un determinado movimiento (i) en la solución actual y si no afecta la fragilidad de la misma.
  - En el punto 1.3, se aplica la memoria basada en lo reciente. Se evalúa si un determinado movimiento (i) se encuentra en periodo Tabú.
  - En el punto 1.4, se aplica la memoria basada en frecuencia. Se aplica una penalización al valor del espacio aprovechado por la solución en base al número de ocasiones en que fue utilizado un determinado movimiento (i).
  - En el punto 1.5, se aplica el criterio de aspiración. Si se cumple con este criterio, se elimina su restricción tabú y la penalización que pueda haber recibido.

3.3 Procedimiento GenerarSolucionesFactibles

GenerarSolucionesFactibles (*MM,MS,T,C,S_Aux*)

Inicio

1. $Espacio\_Aprovechado\_Aux = \emptyset$
2. $RealizarMovimiento(S\_Aux, Espacio\_Aprovechado\_Aux, MM, MS)$
3. $Si\ sePuedeRealizarMovimiento(S\_Aux, C, T)\ y$
   $noAfectaFragilidad(S\_Aux, C, T)\ entonces$

   $Inicio$

   3.1 $MM_{MS}.V = Espacio\_Aprovechado\_Aux$

   3.2 $MM_{MS}.VP = Espacio\_Aprovechado\_Aux$

   $Fin\ Si\ 3$

Fin

Dónde:

- En el punto 1, se define una variable auxiliar para almacenar el espacio aprovechado de la solución actual.
- En el punto 2, se procede a actualizar los contenedores de las tuberías que fueron intercambiados en base a un movimiento seleccionado (MS). Además, una vez actualizada la solución (S_Aux) se procede a calcular nuevamente su espacio aprovechado (Espacio_Aprovechado_Aux). Este se debe a que el movimiento realizado puede haber cambiado su valor.
- En el punto 3, se evalúa si el movimiento realizado en el punto 2 es factible y si no afecta la fragilidad de la solución. En otras palabras, se está realizando el control de aberraciones que se realizó en el algoritmo GRASP.
  - En el punto 3.1, se actualiza el valor del espacio aprovechado con el espacio aprovechado calculado en el punto 2.
  - En el punto 3.2, se inicializa el valor del espacio aprovechado penalizado con el espacio aprovechado calculado en el punto 2.

3.4 Procedimiento AplicarMemoriaBasadaEnLoReciente

AplicarMemoriaBasadaEnLoReciente (*MM,MS,MR*)
Inicio

1. $i = (MM_{MS}.PC - 1) * B + MM_{MS}.PT$
2. $j = (MM_{MS}.SC - 1) * B + MM_{MS}.ST$
3. $Si\ MR_{ij} > 0\ y\ MM_{Ms}.V \neq -1\ entonces\ MM_{MS}.FT = 1$

Fin

Dónde:

- En los puntos 1 y 2, se calcula la ubicación del movimiento seleccionado (MS) en la matriz de memoria basada en lo reciente (MR). El número de tuberías está representado como B.
- En el punto 3, se evalúa si un movimiento factible está en periodo Tabú. De ser así, se actualiza su factor Tabú en la matriz de movimientos (MM).

3.5 Procedimiento AplicarMemoriaBasadaEnFrecuencia

AplicarMemoriaBasadaEnFrecuencia (*MM,MS,MF,N_Frecuencia,n*)
Inicio

1. $i = (MM_{MS}.PC - 1) * B + MM_{MS}.PT$
2. $j = (MM_{MS}.SC - 1) * B + MM_{MS}.ST$
3. $Si\ n \geq N\_Frecuencia\ y\ MM_{Ms}.V \neq -1\ entonces$
   $Inicio$
   3.1 $MM_{MS}.VP = \left(1 - \frac{MF_{ij}}{n}\right) * MM_{MS}.V$
   $Fin\ Si\ 3$

Fin

Dónde:

- En los puntos 1 y 2, se calcula la ubicación del movimiento seleccionado (MS) en la matriz de memoria basada en frecuencia (MF). El número de tuberías está representado como B.
- En el punto 3, se evalúa si un movimiento factible se encuentra en una iteración donde esté permitido el uso de memoria basada en frecuencia.
    - En el punto 3.1, se actualiza el valor del espacio aprovechado penalizado aplicando una penalidad en base al número de veces que fue utilizado el movimiento.

3.6 Procedimiento AplicarCriterioDeAspiracion

AplicarCriterioDeAspiracion (*MM,MS,Espacio_Aprovechado_Mejor*)
Inicio

1. $Si\ MM_{MS}.V \geq Espacio\_Aprovechado\_Mejor\ y\ MM_{MS}.V \neq -1\ entonces$
   $Inicio$
   1.1 $MM_{MS}.VP = MM_{MS}.V$
   1.2 $MM_{MS}.FT = 0$
   $Fin\ Si\ 1$

Fin

Dónde:

- En el punto 1, se evalúa si el espacio aprovechado por la solución luego de aplicar el movimiento seleccionado (MS) es mejor que el espacio aprovechado por la mejor solución encontrada hasta el momento.
    - En el punto 1.1, se actualiza el valor del espacio aprovechado penalizado con el valor de su espacio aprovechado original.
    - En el punto 1.2, se actualiza el factor tabú del movimiento seleccionado como 0. De modo, que es un movimiento factible para ser elegido.

3.7 Procedimiento EscogerMejorMovimiento

EscogerMejorMovimiento (*MM*)

Inicio

1. $MS = \emptyset$
2. $VP\_Mejor = \emptyset$
3. $V\_Mejor = \emptyset$
4. $Para\ i = 1\ a\ D$

   $Inicio$

   4.1 $Si\ MM_i.V \neq -1\ y\ MM_i.FT = 0\ entonces$

   $Inicio$

   4.1.1 $Si\ (MM_i.VP > VP\_Mejor)\ o\ (MM_i.VP = VP\_Mejor\ y\ MM_i.V > V\_Mejor)\ entonces$

   $Inicio$

   4.1.1.1 $MS = i$

   4.1.1.2 $V\_Mejor = MM_i.V$

   4.1.1.3 $VP\_Mejor = MM_i.VP$

   $Fin\ Si$ 4.1.1

   $Fin\ Si$ 4.1

   $Fin\ Para$ 4
5. $Retornar\ MS$

Fin

Dónde:

- En el punto 1, se inicializa la variable que contendrá cual es el mejor movimiento de la matriz de movimientos (MM).
- En los puntos 2 y 3, se inicializan las variables que contendrán el valor del espacio aprovechado y su valor penalizado del mejor movimiento que se encontrará luego de evaluar todos los movimientos de la matriz de movimientos (MM).
- En el punto 4, se ejecuta un proceso que se repetirá hasta que se hayan recorrido todos los posibles movimientos (D) que se encuentran en la matriz de movimientos (MM).

- En el punto 4.1, se evalúa si un determinado movimiento es factible y no se encuentra en periodo Tabú.
    - En el punto 4.1.1, se evalúa si valor penalizado de realizar un determinado movimiento es mejor que el mejor valor penalizado encontrado hasta el momento o, en caso de ambos valores sean igual, si el valor de realizar un determinado movimiento es mejor que el mejor valor encontrado hasta el momento.
        - ✓ En el punto 4.1.1.1, se actualiza el movimiento seleccionado (MS) como el movimiento actual (i).
        - ✓ En el punto 4.1.1.2, se actualiza el mejor valor encontrado como el valor del espacio de aprovechamiento del movimiento actual (i).
        - ✓ En el punto 4.1.1.3, se actualiza el mejor valor penalizado encontrado como el valor penalizado del espacio de aprovechamiento del movimiento actual (i).

3.8 Procedimiento RealizarMovimiento

RealizarMovimiento (*S,Espacio_Aprovechado,MM,MS*)
Inicio

1. $Para\ i = 1\ a\ B$
   $Inicio$
   1.1 $Si\ S_i.T = MM_{MS}.PT\ entonces\ S_i.C = MM_{MS}.SC$
   1.2 $Si\ (S_i.T = MM_{MS}.ST)\ y\ (MM_{MS}.PT \neq MM_{MS}.ST)$
   $entonces\ S_i.C = MM_{MS}.PC$
   $Fin\ Para\ 1$
2. $Espacio\_Aprovechado = \sum_{i=1}^{A} \frac{\sum_{j=1}^{B} 2*\pi*RT_{ij}*LT_{ij}}{ALC_i*ANC_i*LC_i}$

Fin

Dónde:

- En el punto 1, se ejecuta un proceso que se repetirá hasta que se hayan recorrido todas las tuberías (B) que se encuentran en la matriz de solución (S).
    - En el punto 1.1, se evalúa si una determinada tubería es la primera tubería del intercambio de movimientos. De ser así, se actualiza su

contenedor con el valor del segundo contenedor del intercambio de movimientos.

- En el punto 1.2, se evalúa si una determinada tubería es la segunda tubería del intercambio de movimientos y que el intercambio no sea solo de contenedores. De ser así, se actualiza su contenedor con el valor del primer contenedor del intercambio de movimientos.

- En el punto 2, se calcula el espacio aprovechado luego de haber actualizado la ubicación de las tuberías en el punto 1.

3.9 Procedimiento ActualizarMejorSolucion

ActualizarMejorSolucion (*S,Espacio_Aprovechado,S_Mejor,Espacio_Aprovechado_Mejor*)
Inicio

1. $Si\ Espacio\_Aprovechado > Espacio\_Aprovechado\_Mejor\ entonces$
   $Inicio$
   1.1 $S\_Mejor = S$
   1.2 $Espacio\_Aprovechado\_Mejor = Espacio\_Aprovechado$
   $Fin\ Si\ 1$

Fin

Dónde:

- En el punto 1, se evalúa si el espacio aprovechado de una solución (S) es mejor que el espacio aprovechado de la mejor solución encontrada hasta el momento (S_Mejor).
  - En el punto 1.1, se actualiza la mejor solución encontrada hasta el momento.
  - En el punto 1.2, se actualiza el espacio de aprovechamiento de la mejor solución encontrada hasta el momento.

3.10 Procedimiento ActualizarEstructurasTabu

ActualizarEstructurasTabu (*MM,MR,MF,PT,MS*)

Inicio

1. $i = (MM_{MS}.PC - 1) * B + MM_{MS}.PT$
2. $j = (MM_{MS}.SC - 1) * B + MM_{MS}.ST$
3. $MF_{ij} = MF_{ij} + 1$
4. $Si\ MR_{ij} > 0\ entonces\ MR_{ij} = MR_{ij} - 1\ caso\ contrario\ MR_{ij} = PT$

Fin

Dónde:

- En los puntos 1 y 2, se calcula la ubicación del movimiento seleccionado (MS) en las matrices de memoria basada en frecuencia (MF) y de memoria basada en lo reciente (MR). El número de tuberías está representado como B.
- En el punto 3, se actualiza la matriz de memoria basada en frecuencia aumentado en uno el número de veces que fue utilizado el movimiento seleccionado.
- En el punto 4, se evalúa si el movimiento seleccionado se encuentra en un periodo tabú. De ser así, se reduce en uno su periodo Tabú. Por otro lado, en caso no se encuentre en un periodo Tabú, se actualiza su valor asignando el valor del periodo Tabú.

CAPÍTULO 7

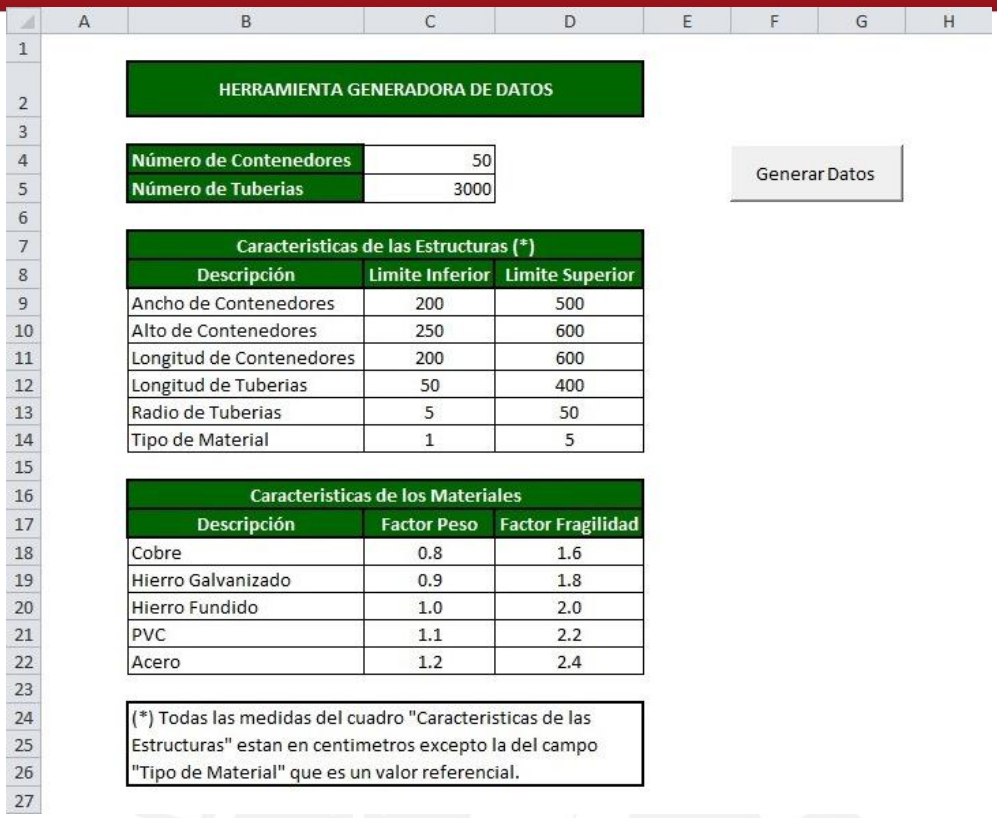
1 Herramienta para la generación de datos

En esta sección se presenta el funcionamiento de la herramienta de software desarrollada para el presente proyecto de fin de carrera que permite la generación de datos a ser utilizados por los algoritmos GRASP y de búsqueda Tabú.

Esta herramienta es un archivo de Microsoft Excel que permite la generación de datos en base a las variables iniciales configurables involucradas con el problema. Estas son el número de contenedores, el número de tuberías, el rango de valores del ancho, el largo y el alto de los contenedores, el rango de valores del largo y el radio de las tuberías, y finalmente el rango de valores para el tipo de material de las tuberías.

La herramienta generadora de datos trabaja de la siguiente manera:

- Para cada contenedor, genera aleatoriamente su alto, ancho y largo en base a los rangos configurables que se han colocado en la herramienta.
- Para cada tubería, genera aleatoriamente su largo y su radio en base a los rangos configurables en la herramienta. Luego, genera un valor aleatorio entre 1 y 5 para escoger que tipo de material se usará: Cobre, Hierro Galvanizado, Hierro Fundido, PVC o Acero. Luego, se genera un peso referencial en base al largo y el radio generados aleatoriamente. Finalmente, para calcular el peso y la fragilidad de cada tubería, se multiplica el peso referencial por el factor peso y el factor fragilidad del tipo de material asignado.



HERRAMIENTA GENERADORA DE DATOS

Número de Contenedores	50
Número de Tuberias	3000

Generar Datos

Caracteristicas de las Estructuras (*)		
Descripción	Limite Inferior	Limite Superior
Ancho de Contenedores	200	500
Alto de Contenedores	250	600
Longitud de Contenedores	200	600
Longitud de Tuberias	50	400
Radio de Tuberias	5	50
Tipo de Material	1	5

Caracteristicas de los Materiales		
Descripción	Factor Peso	Factor Fragilidad
Cobre	0.8	1.6
Hierro Galvanizado	0.9	1.8
Hierro Fundido	1.0	2.0
PVC	1.1	2.2
Acero	1.2	2.4

(*) Todas las medidas del cuadro "Caracteristicas de las Estructuras" estan en centimetros excepto la del campo "Tipo de Material" que es un valor referencial.

Imagen 12: Herramienta generadora de datos

El proceso para la generación de datos inicia colocando los valores que usará la herramienta generadora de datos. Una vez actualizados estos valores, se procede a presionar el botón “Generar Datos”. Esto generará dos archivos: uno para guardar los datos de los contenedores y otro, para los datos de las tuberías.

En el caso del archivo para los datos de los contenedores, este tiene como nombre “input_contenedores.csv”. En la primera fila se muestra el número de contenedores que se maneja. A partir de ahí, durante las siguientes filas, se muestran los datos con los que cuenta cada contenedor: alto, ancho y largo respectivamente.

	A	B	C
1	50		
2	497	360	432
3	351	291	510
4	255	428	526
5	498	214	366
6	552	437	349
7	587	461	222
8	582	309	410
9	518	216	437
10	414	289	449
11	477	279	312
12	540	447	436
13	595	473	291
14	493	494	298
15	437	232	600
16	487	205	430
17	285	231	520
18	350	214	318

Imagen 13: Archivo "input-contenedores.csv"

En el caso del archivo para los datos de las tuberías, este tiene como nombre “input_tuberias.csv”. En la primera fila se muestra el número de tuberías que se maneja. A partir de ahí, durante las siguientes filas, se muestran los datos con los que cuenta cada tubería: material, largo, radio, peso y fragilidad respectivamente.

	A	B	C	D	E
1	3000				
2	PVC	205	17	184	367
3	PVC	313	17	281	561
4	Cobre	140	9	47	94
5	Hierro Galvanizado	163	41	293	587
6	Hierro Galvanizado	132	27	156	311
7	Hierro Fundido	169	7	54	108
8	Hierro Fundido	122	44	262	524
9	Hierro Galvanizado	314	47	650	1300
10	PVC	240	9	111	222
11	Cobre	194	48	365	730
12	Hierro Galvanizado	373	33	539	1078
13	Hierro Galvanizado	102	27	121	241
14	Cobre	398	11	166	331
15	Acero	171	30	299	598
16	PVC	238	23	292	583
17	PVC	339	35	636	1272
18	Hierro Fundido	399	20	385	770
19	Hierro Galvanizado	195	36	308	616
20	PVC	198	29	307	614
21	Hierro Fundido	239	24	278	556
22	Hierro Fundido	130	33	209	418
23	Hierro Galvanizado	288	45	571	1141
24	Hierro Galvanizado	156	18	122	243
25	Hierro Fundido	235	15	168	336

Imagen 14: Archivo "input-tuberias.csv"

Finalmente, dentro del proceso explicado anteriormente, se está ejecutando un control adicional:

- Una vez terminada la generación de los datos de los contenedores y antes de la generación de los datos de las tuberías se obtiene el mayor largo de los contenedores generados. Luego, se procede a compararlo con el límite superior del largo de las tuberías. Si el límite superior del largo de las tuberías es mayor que el mayor largo de los contenedores generados, se actualiza el límite superior de las tuberías como el mayor largo de los contenedores generados. Esto se realiza para evitar tener el caso de una tubería con un mayor largo que cualquier contenedor disponible, lo cual no permitiría resolver el problema.

CAPÍTULO 8

En este capítulo se presenta la experimentación numérica realizada a los algoritmos GRASP y de búsqueda Tabú implementados para el presente proyecto de fin de carrera. Esto se realizó con la finalidad de demostrar que el algoritmo de búsqueda Tabú propuesto desarrolla una solución eficiente para el problema presentado.

1 Definición del problema de investigación

La experimentación numérica se realiza con la finalidad de determinar cuál es el mejor algoritmo para resolver el problema del desperdicio de espacio para el almacenamiento de tuberías. En este caso, se han implementado los siguientes algoritmos:

- Algoritmo GRASP con doble relajación.
- Algoritmo de búsqueda Tabú.

Estos algoritmos fueron adaptados e implementados para resolver el problema mencionado anteriormente.

2 Datos de las muestras

La experimentación numérica se ha realizado sobre un conjunto de cuarenta muestras de resultados de la función objetivo de las combinaciones de contenedores y tuberías por cada algoritmo. Los datos de los contenedores y las tuberías utilizados en cada muestra de la experimentación numérica se han generado en base a la herramienta generadora de datos.

La herramienta generadora de datos genero todas las muestras de datos en base a la siguiente información:

Características de las Estructuras		
Descripción	**Límite Inferior**	**Limite Superior**
Número de Contenedores	10	14
Número de Tuberías	60	80
Ancho de Contenedores	150 cm	350 cm
Alto de Contenedores	150 cm	350 cm
Longitud de Contenedores	150 cm	350 cm
Longitud de Tuberías	75 cm	150 cm
Radio de Tuberías	20 cm	40 cm

Características de los Materiales		
Descripción	**Factor Peso**	**Factor Fragilidad**
PVC	0.60	2.00
Cobre	0.65	2.30
Acero	0.90	2.90
Hierro Galvanizado	1.00	3.20
Hierro Fundido	1.40	4.00

Una vez generadas todas las muestras de datos, se procedió a la ejecución de los algoritmos para cada muestra. A continuación, se presenta una tabla con los resultados por cada muestra evaluada:

N° de Muestra	GRASP	BUSQUEDA TABU
1	0.02701	0.03030
2	0.02994	0.03272
3	0.02612	0.03112
4	0.02792	0.02956
5	0.02793	0.02905
6	0.02752	0.03092
7	0.02647	0.03145
8	0.02862	0.03124
9	0.02793	0.03000
10	0.02748	0.03100
11	0.02737	0.03054
12	0.02775	0.02958

13	0.02840	0.03072
14	0.02809	0.03223
15	0.02700	0.02966
16	0.02714	0.03300
17	0.02681	0.03048
18	0.02958	0.03269
19	0.02832	0.03098
20	0.02703	0.02969
21	0.02866	0.03152
22	0.02787	0.03090
23	0.02563	0.02760
24	0.02784	0.03073
25	0.02607	0.02992
26	0.02725	0.02979
27	0.02752	0.03135
28	0.02669	0.02831
29	0.02778	0.03067
30	0.02725	0.03095
31	0.02821	0.03165
32	0.02830	0.03250
33	0.02624	0.03109
34	0.02648	0.03037
35	0.02655	0.03038
36	0.02636	0.02717
37	0.02908	0.03213
38	0.02875	0.03283
39	0.02928	0.03240
40	0.02849	0.03165

3 Experimentación Numérica

La experimentación numérica realizada esta basada en la comparación de dos tratamientos. En base a este método se comparó dos muestras realizadas para la optimización del almacenamiento de tuberías en base a dos métodos diferentes: el algoritmo GRASP y el algoritmo de búsqueda Tabú. De este modo, se determinó cuál

de los dos métodos es el que genero la muestra más significativa para el almacenamiento de tuberías.

3.1 Prueba de Kolmogorov-Smirnov

La primera prueba a realizarse en la experimentación numérica es la prueba de Kolmogorov-Smirnov. Esta prueba evaluó el grado de concordancia entre la distribución acumulada de las frecuencias teóricas y la distribución acumulada de las frecuencias observadas con la finalidad de determinar si las muestras estudiadas provienen de una población con una distribución teórica determinada.

Para poder realizar esta prueba se necesitó de los siguientes datos (los cuales fueron obtenidos de las muestras presentadas anteriormente):

Descripción	GRASP	BUSQUEDA TABU
Media	0.02762	0.03077
Desviación Estándar	0.0010108	0.0013407
Mínimo Valor de la Muestra	0.02563	0.02717
Máximo Valor de la Muestra	0.02994	0.03300
Cantidad de Muestras (N)	40	40

3.1.1 Resultados del Algoritmo GRASP

Para el algoritmo GRASP, se obtuvo los siguientes resultados:

Rango	0.00431
Número de Intervalos	6.32455532
Tamaño del Intervalo	0.00068126

N° de Intervalo	Limite Inferior	Limite Superior	F	Fo Acumulada	Ft Acumulada	Estadístico D
1	0.02563	0.02631	4	0.1000	0.0978	0.0022
2	0.02631	0.02699	6	0.2500	0.2675	0.0175
3	0.02699	0.02767	10	0.5000	0.5214	**0.0214**
4	0.02767	0.02835	11	0.7750	0.7665	0.0085
5	0.02835	0.02903	5	0.9000	0.9195	0.0195
6	0.02903	0.02972	3	0.9750	0.9810	0.0060

7	0.02972	0.03040	1	1.0000	0.9970	0.0030
8	0.03040	0.03108	0	1.0000	0.9997	0.0003
9	0.03108	0.03176	0	1.0000	1.0000	0.0000
10	0.03176	0.03244	0	1.0000	1.0000	0.0000

Dónde:

- **F**: Es la frecuencia con que se encontraron muestras para un determinado intervalo.
- **Fo Acumulada**: Es la frecuencia observada acumulada para un determinado intervalo.
- **Ft Acumulada**: Es la frecuencia teórica acumulada para un determinado intervalo.
- **Estadístico D**: Es el valor absoluto de la diferencia entre la frecuencia observada acumulada y la frecuencia teórica acumulada para un determinado intervalo.

Además, se debe calcular el estadístico de la tabla de Kolmogorov-Smirnov (K-S). El estadístico de la tabla K-S se obtuvo en base a la cantidad de muestras (N) y el nivel de significancia usado en la experimentación numérica. Para la experimentación numérica se trabajó con un nivel de significancia de 0.05.

En base a esta premisa se obtuvo el siguiente estadístico de la tabla K-S:

Nivel de Significancia	0.05
Grados de libertad	40
Estadístico de la tabla K-S	0.215035

Finalmente, en base a los resultados obtenidos, se comparó el máximo estadístico D obtenido de los intervalos con el estadístico de la tabla K-S. Al ser el primero menor que el segundo, se concluye que las frecuencias observadas y teóricas no difieren significativamente. Por lo tanto, las muestras del algoritmo GRASP tienen una distribución normal.

Estadístico D	0.0214
Estadístico de la tabla K-S	0.215035

3.1.2 Resultados del Algoritmo de búsqueda Tabú

Para el algoritmo de búsqueda Tabú, se obtuvo los siguientes resultados:

Rango	0.00583
Número de Intervalos	6.32455532
Tamaño del Intervalo	0.00092149

N° de Intervalo	Limite Inferior	Limite Superior	F	Fo Acumulada	Ft Acumulada	Estadístico D
1	0.02717	0.02809	2	0.0500	0.0230	0.0270
2	0.02809	0.02902	1	0.0750	0.0954	0.0204
3	0.02902	0.02994	7	0.2500	0.2673	0.0173
4	0.02994	0.03086	9	0.4750	0.5264	**0.0514**
5	0.03086	0.03178	13	0.8000	0.7744	0.0256
6	0.03178	0.03270	5	0.9250	0.9252	0.0002
7	0.03270	0.03362	3	1.0000	0.9833	0.0167
8	0.03362	0.03454	0	1.0000	0.9976	0.0024
9	0.03454	0.03547	0	1.0000	0.9998	0.0002
10	0.03547	0.03639	0	1.0000	1.0000	0.0000

Dónde:

- **F**: Es la frecuencia con que se encontraron muestras para un determinado intervalo.
- **Fo Acumulada**: Es la frecuencia observada acumulada para un determinado intervalo.
- **Ft Acumulada**: Es la frecuencia teórica acumulada para un determinado intervalo.
- **Estadístico D**: Es el valor absoluto de la diferencia entre la frecuencia observada acumulada y la frecuencia teórica acumulada para un determinado intervalo.

Además, el estadístico de la tabla S-K mantiene el mismo valor que el utilizado para las muestras del algoritmo GRASP.

Finalmente, en base a los resultados obtenidos, se comparó el máximo estadístico D obtenido de los intervalos con el estadístico de la tabla K-S. Al ser el primero menor que el segundo, se concluye que las frecuencias observadas y teóricas no difieren significativamente. Por lo tanto, las muestras del algoritmo de búsqueda Tabú tienen una distribución normal.

Estadístico D	0.051390
Estadístico de la tabla K-S	0.215035

3.2 Prueba F de Fisher

Una vez comprobado que ambas muestras presentan una distribución normal, se procede a realizar la prueba F de Fisher. Esta prueba evaluó ambas muestras para poder determinar cual tiene una mayor variación sobre otra o si presentan una variación homogénea. Esto permite asegurar la calidad de los resultados obtenidos por los algoritmos GRASP y de búsqueda Tabú.

En primer lugar, se establecen las siguientes hipótesis:

- H_0: Las varianzas de las muestras son homogéneas.
- H_1: Las varianzas de las muestras son diferentes.

Para poder realizar esta prueba se necesitó de los siguientes datos (los cuales fueron obtenidos de las muestras presentadas anteriormente):

Descripción	GRASP	BUSQUEDA TABU
Media	0.02762	0.03077
Varianza	0.00000102172	0.00000179755
Cantidad de Muestras	40	40
Grados de Libertad	39	39

Además, debe tenerse en consideración esta prueba ha sido realizada utilizando un nivel de significancia de 0.05. El resultado de la prueba F de Fisher es el siguiente:

F	0.56839462
P(F<=f) una cola	0.040808778
Valor crítico para F (una cola)	0.58669434

Finalmente, en base a los resultados obtenidos, se comparó el valor obtenido de F con el valor crítico para F. Si un F es mayor que el valor critico de F, entonces no presenta muestras homogéneas. En base a esto, al ser el valor de F menor que el valor crítico para F, se acepta la hipótesis H_0. Por lo tanto, las muestras de los algoritmos GRASP y de búsqueda Tabú presentan varianzas homogéneas.

3.3 Prueba Z

A diferencia de las pruebas anteriores, la prueba Z no busca evaluar que las muestras presenten una distribución normal o que presenten varianzas homogéneas. Esta prueba evaluó si las muestras presentan medias diferentes y cual, finalmente, es la mejor media.

En primer lugar, se establecen las siguientes hipótesis para evaluar si los algoritmos presentan medias iguales:

- H_0: La media del algoritmo GRASP es igual a la media del algoritmo de búsqueda Tabú.
- H_1: La media del algoritmo GRASP es diferente a la media del algoritmo de búsqueda Tabú.

También se establecen hipótesis para evaluar que algoritmo presenta la mejor media:

- H_0: La media del algoritmo GRASP es igual a la media del algoritmo de búsqueda Tabú.
- H_1: La media del algoritmo GRASP es menor que la media del algoritmo de búsqueda Tabú.

Para poder realizar esta prueba se necesitó de los siguientes datos (los cuales fueron obtenidos de las muestras presentadas anteriormente):

Descripción	GRASP	BUSQUEDA TABU
Media	0.02762	0.03077
Varianza	0.00000102172	0.00000179755
Cantidad de Muestras	40	40

Además, debe tenerse en consideración esta prueba ha sido realizada utilizando un nivel de significancia de 0.05. El resultado de la prueba Z es el siguiente:

Z	-11.87468
P(Z<=z) una cola	0.00000
Valor crítico de z (una cola)	1.64485
P(Z<=z) dos colas	0.00000
Valor crítico de z (dos colas)	1.95996

En base a los resultados obtenidos, se procedió a comprobar las hipótesis planteadas:

- Para el primer grupo de hipótesis, se comparó el valor de z con el valor crítico de z para dos colas. En caso el valor absoluto de z sea mayor que el valor crítico de z para dos colas entonces las medias del algoritmo GRASP y el algoritmo de búsqueda Tabú son diferentes. Por lo tanto, se rechaza la hipótesis H_0 del primer grupo.

- Para el segundo grupo de hipótesis, se comparó el valor de z con el valor crítico de z para una cola. En caso el valor de z sea menor que el valor critico de z para una cola entonces la media del algoritmo GRASP es menor que la media del algoritmo de búsqueda Tabú. Por lo tanto, se rechaza la hipótesis H_0 del segundo grupo.

4 Conclusión de la Experimentación Numérica

En el presente proyecto de fin de carrera se propone solucionar el problema del desperdicio de espacio en el almacenamiento de tuberías mediante el desarrollo de un algoritmo de búsqueda Tabú. No obstante, para comprobar la eficiencia de este algoritmo, fue necesaria la implementación de un algoritmo GRASP. Esto se debe a que se necesitó generar una solución inicial para el algoritmo de búsqueda Tabú. Además, también permitió ser una medida para evaluar si el algoritmo de búsqueda Tabú efectivamente genera soluciones eficientes.

En base a esto, se desarrolló la experimentación numérica de ambos algoritmos en base a un conjunto de cuarenta muestras. En primer lugar, se evaluó que las muestras resultantes presenten una distribución normal, varianzas homogéneas y medias diferentes. Una vez evaluada la calidad de las muestras, se procedió a determinar que algoritmo tiene la mejor media, siendo el algoritmo de búsqueda Tabú el que desarrolla las mejores soluciones.

Por lo tanto, se puede afirmar que el algoritmo de búsqueda Tabú propuesto en el presente proyecto de fin de carrera es una solución eficiente para resolver el problema del desperdicio de espacio en el almacenamiento de tuberías.

CAPÍTULO 9

1 Prototipo Funcional

En esta sección se presenta el funcionamiento del prototipo funcional desarrollado para el presente proyecto de fin de carrera. Esta herramienta permite resolver el problema del desperdicio de espacio para el almacenamiento de tuberías mediante el uso de los algoritmos GRASP y de búsqueda Tabú implementados.

El prototipo funcional presenta una estructura simple y de fácil manejo con la finalidad de que el usuario pueda generar una solución para el problema planteado. A continuación, se presenta la vista inicial del prototipo funcional.

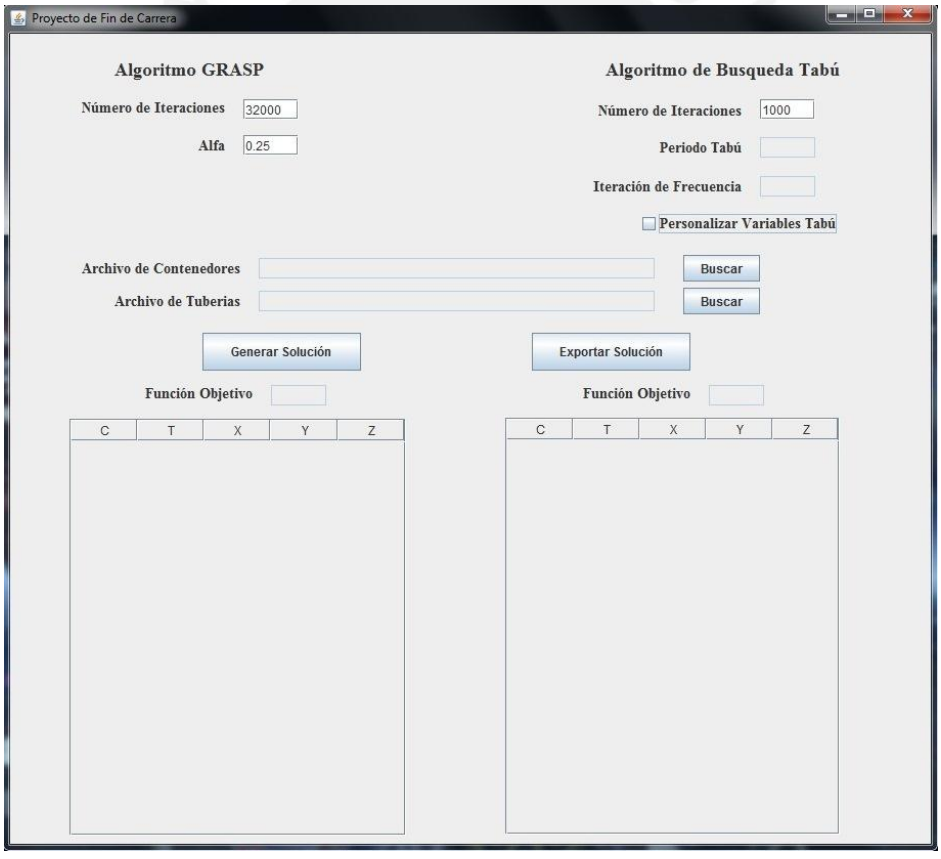


Imagen 15: Prototipo Funcional

En base a esta vista, se deben tener en cuenta las siguientes consideraciones:

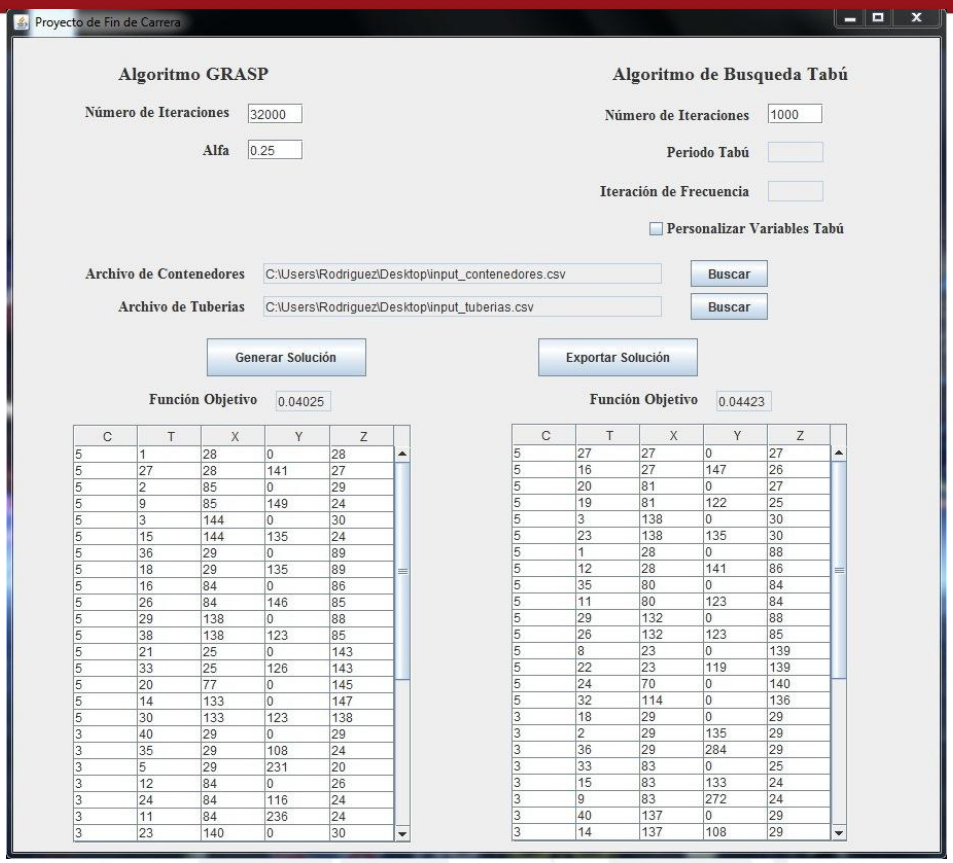
- El número de iteraciones y el valor de la constante de relajamiento del algoritmo GRASP se inicializan con los valores resultantes de la calibración del algoritmo mientras el número de iteraciones del algoritmo de búsqueda Tabú se inicializa en mil.
- El periodo tabú y el número de iteración de frecuencia a partir de la cual se aplica la memoria basada en frecuencia pueden ser modificados. Si bien es cierto que el algoritmo maneja fórmulas para calcular estos valores, el prototipo permite al usuario cambiar sus valores por los que se coloquen en los cuadros de texto respectivos. Para poder realizar esto, el usuario debe presionar el botón de comprobación “Personalizar Variables Tabú”. De no presionarlo, el algoritmo usará las formulas preestablecidas.
- Los archivos con la información de los contenedores y las tuberías podrán ser ubicados desde cualquier ubicación dentro del equipo donde se esté ejecutando el prototipo funcional.

El funcionamiento del prototipo es el siguiente:

- El usuario debe modificar las variables de los algoritmos de acuerdo al tipo de prueba que se desee realizar.
- El usuario debe seleccionar los archivos con los que trabajarán los algoritmos.
- El usuario debe presionar el botón “Generar Solución”.

Una vez generadas las soluciones de los algoritmos, estas se muestran a través del prototipo funcional, el cual maneja un cuadro de texto y una tabla para mostrar los resultados de cada algoritmo. En el cuadro de texto, se muestra el valor de la función objetivo del algoritmo, mientras que en la tabla se muestra en cada fila una tubería, el contenedor al que fue asignada y la posición que tiene dentro del contenedor.

A continuación, se presenta una vista del prototipo funcional luego de haberse generado una solución:



C	T	X	Y	Z
5	1	28	0	28
5	27	28	141	27
5	2	85	0	29
5	9	85	149	24
5	3	144	0	30
5	15	144	135	24
5	36	29	0	89
5	18	29	135	89
5	16	84	0	86
5	26	84	146	85
5	29	138	0	88
5	38	138	123	85
5	21	25	0	143
5	33	25	126	143
5	20	77	0	145
5	14	133	0	147
5	30	133	123	138
3	40	29	0	29
3	35	29	108	24
3	5	29	231	20
3	12	84	0	26
3	24	84	116	24
3	11	84	236	24
3	23	140	0	30

C	T	X	Y	Z
5	27	27	0	27
5	16	27	147	26
5	20	81	0	27
5	19	81	122	25
5	3	138	0	30
5	23	138	135	30
5	1	28	0	88
5	12	28	141	86
5	35	80	0	84
5	11	80	123	84
5	29	132	0	88
5	26	132	123	85
5	8	23	0	139
5	22	23	119	139
5	24	70	0	140
5	32	114	0	136
3	18	29	0	29
3	2	29	135	29
3	36	29	284	29
3	33	83	0	25
3	15	83	133	24
3	9	83	272	24
3	40	137	0	29
3	14	137	108	29

Imagen 16: Resultados del Prototipo Funcional

Finalmente, el prototipo funcional permite al usuario guardar los resultados de la ejecución de los algoritmos mediante el botón “Exportar Solución”. Este botón crea un archivo de texto donde se almacena la información de los resultados de los algoritmos.

CAPÍTULO 10

1 Conclusiones

En esta sección se presentan las conclusiones del presente proyecto de fin de carrera. Estas conclusiones se han realizado luego de haber completado todos los objetivos trazados para resolver el problema del desperdicio de espacio para el almacenamiento de tuberías.

En primer lugar, se observó que el algoritmo de búsqueda Tabú permite manejar muestras de hasta veinte contenedores y cien tuberías. Esto se debe a la limitación que presenta el hardware utilizado durante el presente proyecto de fin de carrera. En base a esto, se puede concluir que si bien el algoritmo presenta una solución eficiente para el problema planteado, su ejecución con equipos más sofisticados, como los de una empresa, permitiría su ejecución con muestras más grandes.

También se pudo comprobar que el tiempo promedio que le toma al algoritmo de búsqueda Tabú generar una solución, en base a mil iteraciones, es de cuarenta segundos mientras que el tiempo promedio que le toma al algoritmo GRASP, con sus valores calibrados, es de diez segundos. Si bien es cierto que el tiempo del algoritmo GRASP es menor que el tiempo del algoritmo de búsqueda Tabú, el algoritmo de búsqueda Tabú siempre ofrece una mayor variedad de soluciones eficientes mientras que el algoritmo GRASP ya no puede mejorar más. Además, como se muestra en la experimentación numérica, las soluciones del algoritmo de búsqueda Tabú son mejores que las soluciones del algoritmo GRASP. Por lo tanto, se puede concluir que el algoritmo de búsqueda Tabú presenta una mejor solución que el algoritmo GRASP para el presente problema.

Por otro lado, si bien es cierto que el objetivo del algoritmo es la optimización del espacio, es necesario considerar que las tuberías solo pueden soportar un determinado peso encima. En base a esto, el algoritmo de búsqueda Tabú fue desarrollado implementando un control de aberración que permita manejar el peso de las tuberías que se colocan encima.

Finalmente, en base a todo lo mencionado anteriormente, se concluye que el algoritmo de búsqueda Tabú soluciona el problema del desperdicio de espacio en el almacenamiento de tuberías en tiempos comprensibles a las necesidades del negocio.

2 Recomendaciones y trabajos futuros

En esta sección se presentan las recomendaciones y trabajos a futuros con la finalidad de proponer mejoras a futuro para solucionar el problema planteado.

En primer lugar, como un trabajo a futuro, se propone el desarrollo de un sistema de información que maneje internamente el algoritmo de búsqueda Tabú. Si bien es cierto que el algoritmo ya demostró su eficiencia para el presente problema, es necesario que sea utilizado en un escenario real. A través de un sistema de información, el algoritmo sería utilizado frecuentemente para asignar la ubicación de las tuberías que provengan del área de producción. Además, esto también permitiría evaluar su eficiencia a través de los reportes que manejaría el sistema donde se vería reflejado como aumento la capacidad de almacenamiento de la empresa y como se ve esto reflejado en sus ganancias.

Por otro lado, también se propone la mejora del algoritmo de búsqueda Tabú dándole un enfoque más físico. Si bien es cierto que el manejo de los pesos que soportan las tuberías es manejable mediante el uso de los pesos de las tuberías que han sido colocadas encima; el uso de conceptos de la física, como la presión que ejerce una tubería sobre otra, haría al algoritmo más preciso.

Además, enfocándose más en la mejora interna del algoritmo, se propone que el algoritmo sea implementado para que maneje todos los contenedores disponibles. Esto se debe a que el algoritmo de búsqueda Tabú solo maneja los contenedores que han sido asignados por el algoritmo GRASP ya que es la solución inicial. En base a esto, el manejo de todos los contenedores disponibles sería una buena opción para implementar a futuro ya que permitiría trabajar con un mayor espacio de soluciones.

Finalmente, enfocándose en el ámbito de la investigación, se propone comparar el algoritmo de búsqueda Tabú con otros algoritmos que han resuelto este problema como el algoritmo de colonia de hormigas para determinar que algoritmo es el que se adapta mejor para el problema de optimización de espacios.

Referencias bibliográficas

BECK, Kent.
2000 Extreme Programming explained. Primera Edición. Addison-Wesley.

BLUM, Christian y ROLI, Andrea.
2003 “Metaheuristics in combinatorial optimization: Overview and conceptual comparison”. ACM Computing Survey. No. 35, pp. 268 – 308.

BROWNLEE, Jason.
2011 Clever Algorithms: Nature-Inspired Programming Recipes. Primera Edición. USA: Lulu Enterprises.

CUBE-IQ – Load Planning & Optimization Software
2013 Página Web de MagicLogic Optimizaton Inc., última consulta 20-09-2013 <http://www.magiclogic.com>

DE ARMAS, Jesica, Yanira GONZALEZ, Gara MIRANDA y Coromoto LEON.
2012 “Parallelization of the multi-objective container loading problem”. Evolutionary Computation (CEC), 2012 IEEE Congress on. pp. 1 – 8.

EL COMERCIO – Sector de tuberías plásticas prevé un crecimiento de hasta 10% en el 2013.
2013 Página Web de El Comercio, última consulta 20-09-2013 <http://elcomercio.pe/economia/1592686/noticia-sector-tuberias-plasticas-preve-crecimiento-hasta-10-2013>

FEO, Thomas y RESENDE, Mauricio.
1995 “Greedy Randomized Adaptative Search Procedures”. Journal of Global Optimization. pp 109 - 134.

GALLART, Josehp y TUPIA, Manuel.
2010 “Two GRASP Metaheuristic for the Capacitated Vehicle Routing Problem Considering Split Delivery and Solving the Three Dimensional Bin Packing Problem”. AISS. Vol. 2, No. 2, pp. 42 – 50.

GLOVER, Fred.
1989 “Tabu Search”. Journal on Computing. Vol. 1, No. 3, pp. 190 – 206.

GLOVER, Fred y LAGUNA, Manuel.
1997 Tabu Search. Primera Edición. USA: Kluwer Academic Publishers.

GUTIERREZ, Humberto y DE LA VARA, Román.
2004 “Análisis y diseño de experimentos”. Primera Edición. McGrawl-Hill.

HE, D. y CHA, J.
2002 “Research on solution to complex container loading problem based on genetic algorithm”. Machine Learning and Cybernetics. 2002 International Conference on. Vol. 1, pp. 78 – 92.

HO WAY YEUNG, Leo y TANG, Wallace.
2005 “A Hybrid Genetic Approach for Container Loading in Logistics Industry”. Industrial Electronics, IEEE Transaction on. Vol. 52, pp. 617 – 627.

HOMPEL, Michael y SCHMIDT, Thorsten.
2007 Warehouse Management. Primera Edición. Alemania: Springer.

JAZIRI, Wassim.
2008 Local Search Techniques: Focus on Tabu Search. Primera Edición. Croacia: In-Teh.

KORTE, Bernhard y VYGEN, Jens.
2005 Combinatorial Optimization: Theory and Algorithms. Tercera Edición. USA: Prentice Hall.

LOGEN SOLUTIONS – Cargo Loading Software
2013 Página Web de Logen Solutions, última consulta 20-09-2013 <http://www.logensolutions.com>

MICROSOFT EXCEL – Software de hoja de cálculo
2013 Página Web de Microsoft Office, última consulta 09-11-2013 <http://office.microsoft.com/es-mx/excel/>

MOSCATO, P.
1996 Optimización heurística y Redes Neuronales. España: Editorial Paraninfo.

NEATBEANS
2013 Página Web de Neatbeans IDE, última consulta 09-11-2013 <https://netbeans.org/features/index.html>

NEATBEANS-FORUMS
2013 Página Web de Neatbeans IDE, última consulta 16-11-2013 <http://forums.netbeans.org/>

ORACLE – Oracle Technology Network for Java Developers
2013 Página Web de Oracle, última consulta 09-11-2013 <http://www.oracle.com/technetwork/java/index.html>

OSMAN, L. y LAPORTE, G.
1996 “Metaheuristics: A bibliography. Annals of Operations Research”. No. 63, pp. 513 – 623.

PACKVOL – Container Loading Optimization Software
2013 Página Web de PackVol, última consulta 20-09-2013 <http://www.packvol.com>

PAPADIMITRIOU, Christos y STEIGLITZ, Kenneth.
1998 Combinatorial Optimization: Algorithms and Complexity. USA: Prentice Hall.

PROJECT MANAGEMENT INSTITUTE
2013 A Guide to the Project Management Body of Knowledge. Pennsylvania: PMI Publications.

RESENDE, Mauricio y GONZÁLEZ, José.
2003 “GRASP: Procedimientos de búsqueda miope aleatorizado y adaptativo”. Inteligencia Artificial. No. 19, pp. 61 – 76.

SPIEGEL, Murray, John SCHILLER y Alu SRINIVASAN
2007 Probabilidad y Estadística. Segunda Edición. México: McGraw-Hill.

TAKAHARA, Shigeyuki.
2006 “A Simple Meta-Heuristic Approach for the Multiple Container Loading Problem”. System, Man and Cybernetics. SMC ’06. IEEE International Conference on. Vol. 3.

TIGRE S.A.
2013 Página Web del Grupo Tigre S.A., última consulta 20-09-2013 <http://www.tigre.pe>

TUPIA, Manuel.
2009 Fundamentos de Inteligencia Artificial. Primera Edición. Lima: Tupia Consultores y Auditores S.A.C.

WANG, Li, Hui ZHANG, Yan XIONG y Dawei LI.
2010 “Ant colony optimization algorithm based on space division for container loading problem”. Control and Decision Conference (CCDC). pp. 3448 – 3451.

ZANAKIS, Stelios y EVANS, James.
1981 “Heuristic Optimization: Why, when and how to use it. Interfaces”. Vol. 11, No. 5, pp. 84-90.