ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

UN MODELO DE OPTIMIZACIÓN PARA EL PROBLEMA DE REUBICACIÓN DINÁMICA DE CONTENEDORES APLICADO AL PUERTO DE ESMERALDAS.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO MATEMÁTICO

ALEXANDER ISAAC QUIÑÓNEZ MÉNDEZ

alexander.quinonez@epn.edu.ec

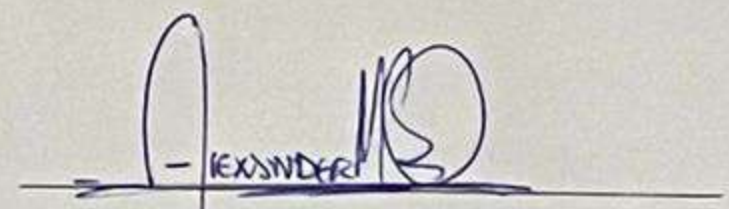
DIRECTOR: MARÍA FERNANDA SALAZAR MONTENEGRO

fernanda.salazar@epn.edu.ec

DMQ, SEPTIEMBRE 2022

CERTIFICACIONES

Yo, ALEXANDER ISAAC QUIÑÓNEZ MÉNDEZ, declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



Alexander Isaac Quiñónez Méndez

Certifico que el presente trabajo de integración curricular fue desarrollado por Alexander Isaac Quiñónez Méndez, bajo mi supervisión.

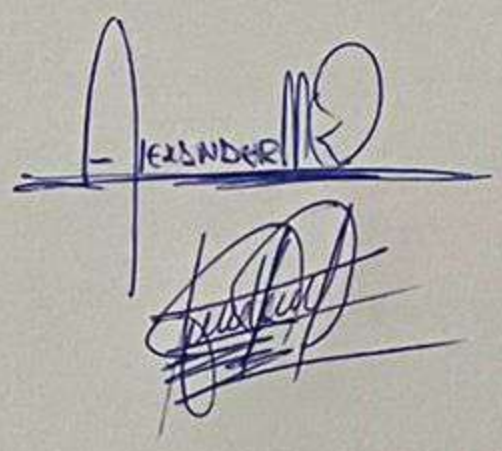
María Fernanda Salazar Montenegro

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el(los) producto(s) resultante(s) del mismo, es(son) público(s) y estará(n) a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Alexander Isaac Quiñónez Méndez



María Fernanda Salazar Montenegro

AGRADECIMIENTOS

A mis padres, Jorge y Marcia, que supieron guiarme de la mejor manera e incentivarme a alcanzar esta importante meta.

A mis hermanos, María, Bellanires, Willian y Javier, por enseñarme con acciones el verdadero significado de la fraternidad y el respaldo.

A Josselyn, por todo su apoyo durante el desarrollo de este trabajo.

A Max y Andrea, por su insuperable amistad, por las risas y las desveladas.

A María Fernanda, por acompañarme como tutora en la elaboración de este complaciente proyecto de titulación, y durante toda la carrera universitaria, compartiendo sus conocimientos y experiencias.

A Edgar Nogales y a la Autoridad Portuaria de Esmeraldas, por facilitarme los datos necesarios para la realización de este trabajo.

A todas las personas que han sido parte de mi proceso de formación, a los compañeros con quienes coincidí, a los profesores que tuve la fortuna de aprovechar, y a la Escuela Politécnica Nacional por darme la oportunidad de formarme en sus aulas y bajo su modelo de excelencia.

DEDICATORIA

A Marcia Méndez Erazo, mi madre,
quien siempre vivirá en mí.

RESUMEN

El Problema de Reubicación Dinámica de Contenedores (DCRP) estudia el vaciado y almacenamiento de un respectivo número de contenedores en una bahía de un patio de apilamiento, donde cada contenedor sigue un orden de recolección determinado para minimizar el número de reubicaciones realizadas durante el proceso de recuperación. Este trabajo de integración curricular se enfoca en analizar el desempeño del DCRP sobre instancias del puerto comercial de la ciudad de Esmeraldas, donde se conoce de antemano las secuencias de llegada y salida de los contenedores. Se presenta un modelo de programación lineal entera binaria para el problema, adaptado a las condiciones del caso. A continuación, los experimentos computacionales se realizan sobre instancias construidas según los períodos en los cuales se subdivide todo el horizonte de planificación, así con los datos compartidos por la Autoridad Portuaria de Esmeraldas se fabrican instancias bajos tres enfoques diferentes: un primer enfoque examina períodos diarios, el segundo enfoque períodos semanales y un tercer enfoque períodos mensuales. Los resultados muestran que el DCRP bajo el enfoque de períodos semanales funciona mejor que los otros enfoques estudiados para las instancias empleadas.

Palabras clave: planificación, programación lineal, programación lineal entera binaria, reubicación de contenedores, almacenamiento de contenedores.

ABSTRACT

The Dynamic Container Relocation Problem (DCRP) studies the emptying and storage of a respective number of containers in a bay in a storage yard, where each container follows a certain collection order to minimize the number of relocations carried out during the process of recovery. This curricular integration work focuses on analyzing the performance of the DCRP for instances of the commercial port of the city of Esmeraldas, where the arrival and departure sequences of the containers are known in advance. A binary integer linear programming model for the problem is presented, adapted to the conditions of the case. Next, the computational experiments are carried out on instances built according to the periods in which the entire planning horizon is subdivided, thus, with the data shared by the Port Authority of Esmeraldas, instances are constructed under three different approaches: a first approach examines daily periods, the second approach weekly periods and a third approach monthly periods. The results show that the DCRP under the weekly periods approach has a better performance than the other approaches studied for the used instances.

Keywords: scheduling, linear programming, binary integer programming, container relocation, container storage.

Índice general

Índice de figuras

Capítulo 1

Descripción del componente desarrollado

El comercio por contenedores a nivel internacional ha crecido considerablemente en los últimos años, pues se estima que para el año 2014 los porta-contenedores transportaban el 52 % del tráfico marítimo mundial en términos de valor [8]. El informe de 2021 de la UNCTAD (United Nations Conference on Trade and Development) revela que el transporte marítimo por contenedores no se vio mayormente afectado por la pandemia de COVID-19, y prevé una recuperación rápida. Dado que el transporte marítimo es un componente importante en el comercio internacional, su necesaria recuperación plantea un desafío adicional para la economía mundial [9]. Así, las terminales marítimas de contenedores juegan un papel importante en este tipo de comercio, pues su función primaria es la transferencia de contenedores de un modo de transporte a otro, así como el almacenamiento temporal de los mismos.

Según Zhang et al., la determinación de los lugares de almacenamiento de los contenedores implica dos niveles de decisiones [22]. En el primer nivel, dados los horarios de atraque de las embarcaciones, se asignan espacios de almacenamiento (es decir, bloques y bahías) para los contenedores de cada embarcación, con el objetivo de equilibrar la carga de trabajo de las grúas de patio y evitar la congestión del tráfico que puede producirse durante el traslado de contenedores. En el segundo nivel, se debe decidir la ubicación exacta de los contenedores dentro de una bahía en un bloque del patio de almacenamiento. Esto requiere la asignación

individual de cada contenedor a una ranura en una bahía. En este nivel, deben responderse dos preguntas interrelacionadas: en qué lugar se debe colocar un contenedor entrante dentro de una bahía y cómo se puede minimizar el número de reubicaciones de contenedores durante sus respectivas recuperaciones de una bahía.

El Problema de Reubicación Dinámica de Contenedores (DCRP por sus siglas en inglés) se enfoca en el segundo nivel de decisiones; donde, dado un conjunto de contenedores con sus tiempos de llegada y recuperación, un horizonte de planificación dividido en pequeños intervalos de tiempo en los cuales como máximo llega o necesita ser recuperado un contenedor; el objetivo es minimizar el número de reubicaciones de los contenedores encima del contenedor objetivo (contenedor a ser recuperado de una bahía) en todo el horizonte de planificación. Este problema es una extensión del Problema de Reubicación de Contenedores (CRP por sus siglas en inglés) que fue presentado formalmente por Kim y Hong [13], donde todos los contenedores están en el sistema en el tiempo inicial y no llegan nuevos contenedores.

El puerto comercial de Esmeraldas, debería ser el eje motor del desarrollo económico de la ciudad, generando a sus alrededores un empleo indirecto masivo, pero para el 2020 el puerto se encontraba manejando bajas tasas de carga y naves atendidas [20]. Uno de los principales costos dentro de las operaciones del puerto se localiza en el manejo del patio de contenedores, que, aunque el tránsito de contenedores ha venido a menos cada año, estos costos no se han podido abaratar, pues tampoco ha existido mayor inversión en maquinarias para la atención de buques y carga [20]. Por lo tanto, es necesario introducir mecanismos que permitan reducir los costos que conllevan estas operaciones.

1.1. Objetivo general

Formular y resolver el Problema de Reubicación Dinámica de Contenedores, mediante el uso de modelos de programación lineal entera.

1.2. Objetivos específicos

1. Formular un modelo de programación lineal entera para el Problema de Reubicación Dinámica de Contenedores (DCRP) para los datos obtenidos en el puerto de Esmeraldas.

2. Implementar computacionalmente el modelo entero para el Problema de Reubicación Dinámica de Contenedores.

1.3. Alcance

El propósito de este proyecto se centra en aportar con estudios y aplicaciones de optimización matemática para instituciones públicas de Ecuador; en el caso particular, sobre el puerto comercial de la ciudad de Esmeraldas; que tiene bajo su operatividad y responsabilidad un conjunto de procedimientos costosos que ameritan ser analizados y optimizados. Por lo expuesto, en este trabajo se plantea construir un modelo de optimización que se adapte a las condiciones del puerto comercial de Esmeraldas, orientado a las operaciones de reubicación de contenedores, y resolverlo con datos reales.

1.4. Marco teórico

En esta sección se muestra diferentes principios, ideas y varias definiciones formales necesarias para lograr un mayor entendimiento de la metodología que es empleada para alcanzar los objetivos planteados.

La administración óptima de los recursos es una parte vital en la gestión proyectos, ya que se busca asignarlos correctamente dependiendo de las necesidades, limitaciones y tiempos de disponibilidad [19]. Por su parte, la programación lineal se ha aplicado con éxito en una gran variedad de problemas para lograr una optimización de los recursos disponibles, por ejemplo, en Guerra-Olivares [10] se presenta un procedimiento heurístico en tiempo real para el problema de la reubicación de contenedores que emplea vehículos retráctiles como equipo de manipulación

de contenedores, la heurística propuesta busca buenas coordenadas de reubicación dentro de un conjunto de bahías cercanas, logrando reducir los movimientos de reubicación y limitar las distancias de viaje de los vehículos en comparación con una práctica común en las terminales de contenedores más pequeñas en Chile y México, poniendo en evidencia que una solución al problema planteado resulta ser una herramienta de gestión muy útil, en especial en las operaciones portuarias. El puerto de Esmeraldas tiene como visión "ser un puerto líder en la comunidad portuaria de los corredores de transporte de la zona de influencia, generando la excelencia en la calidad del servicio". Por este motivo, contar con un modelo de optimización para la reubicación óptima de contenedores podría ser de ayuda para la toma de decisiones de los administradores del puerto e incluso podría extenderse a la generación de políticas públicas eficientes.

1.4.1. Programación Lineal (LP)

La programación lineal (Linear Programming) es una técnica de optimización (maximización o minimización) de una función lineal o afín, denominada función objetivo, de manera que las variables de decisión de dicha función estén sujetas a un conjunto de restricciones (limitaciones o exigencias) expresadas mediante ecuaciones y/o desigualdades lineales. Las hipótesis de la programación lineal son:

Proporcionalidad.-Los efectos de una variable de decisión son proporcionales a un valor constante.

Aditividad.- Los efectos de dos o más variables que miden una misma magnitud son aditivos.

Continuidad.- Las variables de decisión toman valores en los números reales.

Determinismo.- Los datos se conocen con certeza, y son fijos.

Korte et al. en [14] definen el problema general de programación lineal de la siguiente manera:

Instancia: Sea una matriz $A \in \mathbb{R}^{m*n}$ y dos vectores $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

Tarea: Encontrar un vector $x \in \mathbb{R}^n$ tal que $Ax \leq b$ y $c^\top x$ sea el máximo, de modo que $x \in \mathbb{R}^n : Ax \leq b$ sea vacío, o de tal manera que $\forall \alpha \in \mathbb{R}$ exista un $x \in \mathbb{R}^n$ tal que $Ax \leq b$ y $c^\top x > \alpha$.

Un **programa lineal** es una instancia para el problema anterior, y a menudo se escribe $max\{c^\top x : Ax \leq b\}$. Si no se especifican tamaños, siempre se supone que las matrices y los vectores son compatibles. Con frecuencia se omite indicar la transposición de los vectores y simplemente se escribe, por ejemplo, $c^\top x$ para el producto escalar.

Una **solución factible** de un programa lineal $max\{c^\top x : Ax \leq b\}$ es un vector x con $Ax \leq b$. Una solución factible que alcanza el máximo se llama **solución óptima**. Como indica la formulación del problema, hay dos posibilidades cuando un programa lineal no tiene solución: El problema puede ser **no factible** (es decir, $P := \{x \in \mathbb{R}^n : Ax \leq b\} = \emptyset$) o **no acotado** (es decir, para todo $\alpha \in \mathbb{R}$ existe $x \in P$ tal que $c^\top x > \alpha$).

1.4.2. Programación Lineal Entera (IP)

La programación lineal entera (Integer Programming), es un método perteneciente a la programación lineal donde se debe cumplir la condición adicional de que todas las variables de decisión deben tomar valores enteros, y se formula como sigue:

Instancia: Sea una matriz $A \in \mathbb{Z}^{m*n}$ y dos vectores $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$.

Tarea: Encontrar un vector $x \in \mathbb{Z}^n$ tal que $Ax \leq b$ y $c^\top x$ sea el máximo, de modo que $\{x \in \mathbb{Z}^n : Ax \leq b\} = \emptyset$, o de tal manera que $sup\{c^\top x : x \in \mathbb{Z}^n,\ Ax \leq b\} = \infty$.

Prácticamente todos los problemas de optimización combinatoria se pueden formular como programas enteros [14]. El conjunto de soluciones factibles se puede escribir como $x : Ax \leq b, x \in \mathbb{Z}^n$ para alguna matriz A y algún vector b.

No se define programas enteros mixtos, es decir, programas lineales con restricciones de integralidad solo para un subconjunto de las variables, pues la mayor parte de la teoría de la programación lineal y entera se puede extender a la programación entera mixta de forma natural.

1.4.3. Programación Lineal Entera Binaria (BIP)

La programación entera binaria (Binary Integer Programming) es también un procedimiento derivado de la programación lineal, por lo que su propósito es resolver un problema que ha sido formulado mediante el uso de ecuaciones y/o inecuaciones lineales, que busca optimizar una función objetivo, donde todas sus variables de decisión pertenecen al conjunto $\{0, 1\}$.

En general, la programación binaria se utiliza en problemas de planificación, asignación o de toma de decisiones, encaminados a hacer o no una respectiva tarea. Entre los campos de aplicación más comunes para la programación lineal entera binaria se tiene el diseño de redes, la elección de locaciones, la asignación de turnos, el despacho de envíos, y la calendarización de actividades [3], que es la aplicación objeto de estudio en este trabajo de integración curricular.

1.4.4. Scheduling

El scheduling (calendarización, programación o planeación) es la actividad encargada de asignar de manera específica los recursos limitados a tareas u operaciones en un determinado tiempo. Dependiendo de la situación, los recursos y las actividades pueden tomar muchas formas diferentes. Los recursos pueden ser máquinas en una planta de ensamblaje, computadoras, memorias y dispositivos en un sistema informático, pistas en un aeropuerto, mecánicos en un taller de reparación de automóviles, etc. [17].

Los problemas de planificación de actividades o de calendarización han sido estudiados intensamente durante más de 50 años por investigadores en administración, ingeniería industrial, investigación de operaciones e informática, como se indica en Leung [17], donde además se señala que los problemas de calendarización se ocupan de la asignación de recursos escasos a actividades con el objetivo de optimizar una o más medidas de desempeño. Allí, los autores presentan una extensa investigación sobre la calendarización en diversas áreas tales como: problemas de nómina de enfermeras, horarios universitarios, problemas de progra-

mación en la industria aérea, entre otros. En Ecuador, los modelos de scheduling han sido utilizados en distintos ámbitos por instituciones públicas y privadas; por ejemplo, el trabajo presentado por Recalde et al. [18], exhibe una formulación de programación lineal entera para la calendarización de los horarios de los encuentros de la liga de fútbol profesional en Ecuador, y también un enfoque heurístico basado en tres fases para su solución.

Para la calendarización de tareas a realizarse dentro de un patio de contenedores, en una terminal marítima deben considerarse no solo las limitaciones de las maquinarias, sino también las limitaciones del personal encargado de la operación de las mismas. Adicionalmente, se debe tener en cuenta la organización del espacio, todo esto para lograr determinar una planificación óptima de las actividades de almacenamiento y recuperación de los contenedores.

1.4.5. Problema de Reubicación de Contenedores (CRP)

El patio de almacenamiento de una terminal de contenedores se divide en grandes áreas de almacenamiento llamadas bloques. La Figura 1.1 ilustra un bloque de almacenamiento típico en terminales de contenedores de puertos marítimos. La unidad de almacenamiento más pequeña es una ranura para un contenedor. Las ranuras de los contenedores se apilan una encima de la otra para formar una columna, y las columnas se colocan juntas una al lado de la otra para formar una bahía. Un bloque se forma poniendo una serie de bahías juntas. Las ranuras del mismo nivel en una bahía forman una fila.

Uno de los objetivos más importantes de las operaciones de apilamiento y recuperación de contenedores en los sistemas de almacenamiento es minimizar el número de reubicaciones durante la operación de recogida de un contenedor, dado que es imposible apilar contenedores de forma que no se bloqueen, puesto que generalmente solo se dispone de información parcial sobre las futuras recuperaciones en el momento en que se almacena un contenedor; por lo tanto, las reubicaciones no productivas acaban siendo necesarias durante el proceso de recuperación. El proce-

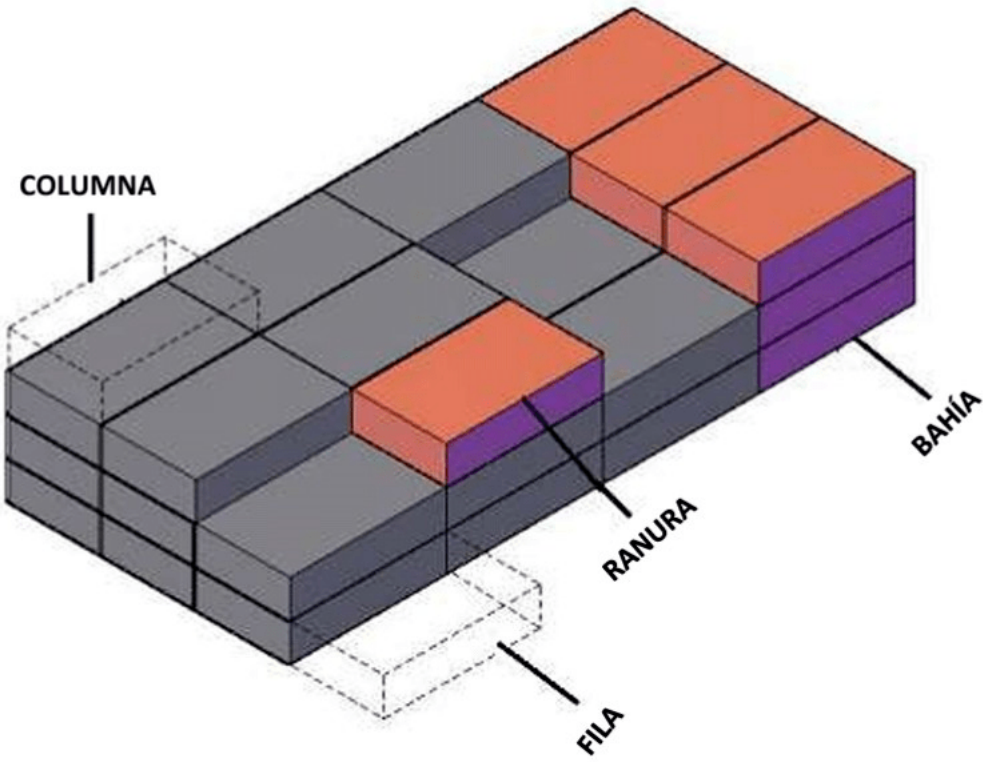


Figura 1.1: Un bloque de contenedores en una zona de apilamiento

dimiento común es realizar la reubicación en la misma bahía en que se encuentra el contenedor que necesita ser reorganizado, pues mover un contenedor de una bahía a otra puede llegar a consumir mucho tiempo, en especial si las grúas de patio empleadas no son las apropiadas. Por tal motivo, el número de reubicaciones tiene la mayor influencia en el tiempo de recuperación de un contenedor en una bahía.

El problema de asignación de la ubicación de un contenedor dentro de un bloque de almacenamiento se convierte en una colección de problemas de asignación de ranuras para cada bahía de dicho bloque, pues la mejor de las mejores ranuras para bahías individuales es de hecho la mejor ranura para todo el bloque. Por tanto, es natural estudiar el problema de asignación de la ubicación de un contenedor para una sola bahía. Es aquí donde aparece el Problema de Reubicación de Contenedores (Container Relocation Problem) donde los contenedores se cargan (llegan) y luego se descargan (salen) de una bahía en diferentes momentos y el objetivo es encontrar un plan de estiba (asignación de ubicaciones a los contenedores) para que se minimice el número de reubicaciones necesarias. El

CRP, por sus siglas en inglés, se puede modelar como un problema de programación entera binaria (BIP) y se sabe que es NP-hard [7].

El problema de la reubicación de contenedores dentro de una bahía en un bloque de almacenamiento es difícil no solo por su naturaleza dinámica, si no también por su naturaleza combinatoria y de acoplamiento de los estados y decisiones, que dificulta el problema incluso para su versión estática, es decir, para minimizar el número total de reorganizaciones al vaciar una bahía de contenedores de un bloque. El CRP es un caso especial del problema dinámico, en el que todos los contenedores de una bahía se recuperan antes de la llegada del siguiente contenedor.

Por un lado, el número de estados de este problema estático aumenta exponencialmente con el número de contenedores. Más importante aún, en el curso del vaciado de la bahía, la asignación de ubicación de un contenedor reorganizado puede causar más reorganizaciones futuras y afectar las decisiones de recuperación restantes. Representar las configuraciones de bahías futuras como funciones de decisiones pasadas no es sencillo y el problema estático de hecho tiene características dinámicas.

El objetivo del CRP es determinar una secuencia de movimientos para vaciar una bahía en un patio de almacenamiento con un número mínimo de reorganizaciones para recuperar contenedores que tienen un orden de recuperación bien definido.

Considere S contenedores de exportación almacenados en una bahía de C columnas y T filas de modo que, para permitir espacio para la reorganización, $S \leq (C-1)T+1$. Sin pérdida de generalidad, los contenedores se clasifican de 1 a S, con un subconjunto de contenedores recuperados previamente. Durante el proceso de vaciado de la bahía, los contenedores que se encuentran sobre los que deben recuperarse primero se reorganizan en diferentes ranuras de almacenamiento de la bahía. La meta es asignar estos contenedores reorganizados a las mejores ranuras de almacenamiento de modo que el número total de reubicaciones se minimice al vaciar la bahía. Wan et al. [21] impusieron restricciones adicionales que condujeron al primer modelo de optimización lineal para este problema.

- Restricción en reorganizaciones anticipadas. Indica que no hay nin-

guna reorganización anticipada de contenedores; es decir, un contenedor se reorganizará solo si está en un nivel superior de la misma columna del contenedor de recuperación.

- Restricción de movimientos múltiples de contenedores. Indica que para la recuperación de cualquier contenedor, cada contenedor reorganizado se mueve solo una vez durante el proceso de recuperación.

Para manejar las características dinámicas del problema, su formulación como un modelo de programación entera rastrea las etapas en el curso de la recuperación de contenedores, donde la etapa e corresponde a la recuperación del contenedor s, $1 \leq e \leq E$, donde E es el número de etapas totales en las cuales se subdivide todo el horizonte de planificación. Observe que, como se recupera o almacena solo un contenedor durante cada etapa, entonces $E = S$. Las variables en la formulación generalmente están indexadas por los siguientes cuatro subíndices: e, la etapa de recuperación de contenedores, donde el contenedor s se recupera en la etapa e y es el s-ésimo contenedor recuperado, $1 \leq s \leq S$; i, el índice del contenedor bajo consideración (que se reubica), donde los contenedores desde el $s + 1$ al S todavía están en la bahía en la etapa e cuando se recupera el contenedor s, $1 \leq i \leq S$; c, el índice de la columna en la bahía del contenedor considerado, $1 \leq c \leq C$; y p, el índice de posición dentro de la columna del contenedor considerado, contado en orden ascendente desde el contenedor más bajo hacia arriba, hasta el contenedor más alto, $1 \leq p \leq P(= T)$.

En cualquier etapa, el contenedor s es el que se recupera y el contenedor i ($> s$) se deja atrás. En general, los rangos de los índices son tales que $1 \leq c \leq C$, $1 \leq p \leq P$, $1 \leq e \leq E$ y $1 \leq s < i \leq S$. En algunas restricciones para el contenedor s, el rango puede ser de la forma de $1 \leq s \leq i \leq S$. Por otra parte, las variables de decisión se definen de la siguiente manera:

$$x_{eicp} = \begin{cases} 1, & \text{si el contenedor } i \text{ se encuentra ubicado en la posición } p \text{ de la columna } c \text{ en la etapa } e. \\ 0, & \text{en otro caso.} \end{cases} \tag{1.1}$$

$$u_{si} = \begin{cases} 1, & \text{si el índice de la columna del contenedor } i \text{ no es menor que el del contenedor } s. \\ 0, & \text{en otro caso.} \end{cases} \tag{1.2}$$

$$v_{si} = \begin{cases} 1, & \text{si el índice de la columna del contenedor } i \text{ no es mayor que el del contenedor } s. \\ 0, & \text{en otro caso.} \end{cases} \tag{1.3}$$

$$z_{si} = \begin{cases} 1, & \text{si los contenedores } i \text{ y } s \text{ están en la misma columna.} \\ 0, & \text{en otro caso.} \end{cases} \tag{1.4}$$

$$y_{si} = \begin{cases} 1, & \text{si el contenedor } i \text{ es reubicado en la recuperación del contenedor } s. \\ 0, & \text{en otro caso.} \end{cases} \tag{1.5}$$

$$w_{eij} = \begin{cases} 1, & \text{si los contenedores } i \text{ y } j \text{ se reubican en la etapa } e, \text{ y el contenedor } j \text{ está en un nivel más alto que el contenedor } i \text{ antes de reubicarse.} \\ 0, & \text{en otro caso.} \end{cases} \tag{1.6}$$

Inicialmente, se cuenta con una configuración de la bahía, que dicta los valores de x_{1icp}, $1 \leq i \leq S$, $1 \leq c \leq C$ y $1 \leq p \leq P$. Conceptualmente, el conjunto completo de valores de x_{eicp} define todas las configuraciones

de la bahía en el curso del vaciado de la bahía a medida que se recuperan los contenedores. Esta información es suficiente para que se pueda determinar el número de reorganizaciones en todo el proceso de recuperación. Por el contrario, al establecer condiciones (restricciones) sobre x_{eicp}, se puede fijar restricciones sobre la reorganización de contenedores. La forma óptima de reorganizar también se puede determinar rastreando x_{eicp}. Para una bahía de C columnas y P posiciones, hay $\frac{(S+1)SCP}{2}$ variables x_{eicp}. Para una bahía llena de contenedores, $S = [(C-1)P+1]$; es decir, el número de x_{eicp} es del orden C^3P^3, y el número posible de combinaciones (es decir, estados) es del orden $2^{C^3P^3}$.

Muchas de las combinaciones posibles son de hecho inviables de una forma u otra; por ejemplo, existen combinaciones de x_{eicp} de modo que los contenedores flotan en el aire o una única ranura de almacenamiento almacena múltiples contenedores. Las restricciones en el siguiente modelo de programación lineal entera identifican las soluciones factibles de x_{eicp}. Por definición, en cada etapa, y_{si} es el incremento de reorganizaciones debido al contenedor i, y $\sum_{s=1}^{S-1}\sum_{i=s+1}^{S} y_{si}$ es el número total de reorganizaciones durante el transcurso de la recuperación de todos los contenedores. El objetivo es minimizar el número total de reorganizaciones $\sum_{s=1}^{S-1}\sum_{i=s+1}^{S} y_{si}$ mediante programación entera, por lo que Wan et al. definieron el modelo de programación entera como MRIP (Minimize Reshuffes by Integer Program).

$$Min \sum_{s=1}^{S-1}\sum_{i=s+1}^{S} y_{si}$$

$Sujeto\ a$:

$$Cu_{si} \geq \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{eicp} - \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{escp} + 1 \quad , \ 1 \leq e \leq s < i \leq S; \tag{1.7}$$

$$Cu_{si} - C \leq \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{eicp} - \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{escp} \quad , \ 1 \leq e \leq s < i \leq S; \tag{1.8}$$

$$Cv_{si} \geq \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{escp} - \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{eicp} + 1 \quad , \ 1 \leq e \leq s < i \leq S; \tag{1.9}$$

$$Cv_{si} - C \leq \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{escp} - \sum_{c=1}^{C}\sum_{p=1}^{P} cx_{eicp} \ , \ 1 \leq e \leq s < i \leq S; \tag{1.10}$$

$$z_{si} = u_{si} + v_{si} - 1 \ , \ 1 \leq s < i \leq S; \tag{1.11}$$

$$y_{si} \leq z_{si} - 1 + (\sum_{c=1}^{C}\sum_{p=1}^{P} px_{eicp} - \sum_{c=1}^{C}\sum_{p=1}^{P} px_{escp})/P \leq 1 - y_{si} \ , \ 1 \leq e \leq s < i \leq S; \tag{1.12}$$

$$y_{si} \leq z_{si} \ , \ 1 \leq s < i \leq S; \tag{1.13}$$

$$(\sum_{c=1}^{C}\sum_{p=1}^{P} px_{escp} - \sum_{c=1}^{C}\sum_{p=1}^{P} px_{eicp})/P \leq 1 - y_{si} \ , \ 1 \leq e \leq s < i \leq S; \tag{1.14}$$

$$\sum_{c=1}^{C}\sum_{p=1}^{P} x_{eicp} = 1 \ , \ 1 \leq e < i \leq S; \tag{1.15}$$

$$\sum_{i=s}^{S} x_{eicp} \leq 1 \ , \ 1 \leq e \leq s \leq S \ , \ 1 \leq c \leq C, \ 1 \leq p \leq P; \tag{1.16}$$

$$\sum_{i=s}^{S} x_{eicp} \leq \sum_{i=s}^{S} x_{eic,p-1} \ , \ 1 \leq e \leq s \leq S \ , \ 1 \leq c \leq C, \ 2 \leq p \leq P; \tag{1.17}$$

$$\sum_{p=1}^{P} x_{e+1,icp} \leq 2 - y_{si} - \sum_{p=1}^{P} x_{escp} \ , \ 1 \leq e \leq s < i \leq S \ , \ 1 \leq c \leq C; \tag{1.18}$$

$$2 - y_{si} - y_{sj} + w_{sij} \geq (\sum_{c=1}^{C}\sum_{p=1}^{P} px_{ejcp} - \sum_{c=1}^{C}\sum_{p=1}^{P} px_{eicp})/P \ , \quad 1 \leq e \leq s < i \leq S, \ 1 \leq s < j \leq S, \ i \neq j; \tag{1.19}$$

$$y_{si} + y_{sj} + w_{sij} \leq 3(\sum_{c=1}^{C}\sum_{p=1}^{P} px_{ejcp} - \sum_{c=1}^{C}\sum_{p=1}^{P} px_{eicp})/P \ , \quad 1 \leq e \leq s < i \leq S, \ 1 \leq s < j \leq S, \ i \neq j; \tag{1.20}$$

$$w_{eij} \leq y_{si} \ , \ 1 \leq e \leq s < i \leq S, \ 1 \leq s < j \leq S, \ i \neq j; \tag{1.21}$$

$$w_{eij} \leq y_{sj} \ , \ 1 \leq e \leq s < i \leq S, \ 1 \leq s < j \leq S, \ i \neq j; \tag{1.22}$$

$$\sum_{p=1}^{P} px_{s+1,icp} - \sum_{p=1}^{P} px_{e+1,jcp} \leq -P(1 - w_{eij}) - P(1 - y_{si}) - P(1 - \sum_{p=1}^{P} x_{e+1,icp}) \ , \quad 1 \leq e \leq s < i \leq S, \ 1 \leq s < j \leq S, \ i \neq j, \ 1 \leq c \leq C; \tag{1.23}$$

$$x_{e+1,icp} - x_{eicp} \leq -y_{si} \ , \ 1 \leq e \leq s < i \leq S, \ 1 \leq c \leq C, \ 1 \leq p \leq P; \tag{1.24}$$

$$x_{eicp} - x_{e+1,icp} \leq -y_{si} \ , \ 1 \leq e \leq s < i \leq S, \ 1 \leq c \leq C, \ 1 \leq p \leq P; \tag{1.25}$$

$$x_{1icp} = X_{1icp} \ , \ 1 < i \leq S, \ 1 \leq c \leq C, \ 1 \leq p \leq P; \tag{1.26}$$

$$x_{eicp} = X_{1icp} \ , \ 2 \leq s \leq min\{i, s_i\}, \ e \leq s \leq i \leq S, \ 1 \leq c \leq C, \ 1 \leq p \leq P; \tag{1.27}$$

$$u_{si}, v_{si}, w_{eij}, y_{si}, z_{si} \in \{0,1\} \ , \ 1 \leq e \leq s < i \leq S, \ 1 \leq s < j \leq S, \ i \neq j, \ 1 \leq c \leq C, \ 1 \leq p \leq P; \tag{1.28}$$

$$x_{eicp} \in \{0,1\} \ , \ 1 \leq e < i \leq S, \ 1 \leq c \leq C, \ 1 \leq p \leq P; \tag{1.29}$$

Notemos que s_i es el rango más pequeño entre los contenedores debajo de i en la configuración inicial; x_{1icp} es el valor conocido de x_{1icp} definido por la posición del contenedor i en la configuración inicial.

Las restricciones (1.7) y (1.8) aseguran que u_{si} satisfaga su definición (1.2). Mediante un argumento similar, las restricciones (1.9) y (1.10) aseguran que v_{si} satisfaga (1.3). Las restricciones (11) aseguran que $z_{si} = 1$ si y solo si los contenedores s e i están en la misma columna. La equivalencia es clara al observar que para cada par de u_{si} y v_{si}, al menos uno de ellos es distinto de cero, y ambos son iguales a uno cuando los contenedores s_{yi} están en la misma columna.

Las restricciones (1.12), (1.13) y (1.14) establecen y_{si} de manera que la definición (1.5) sea consistente; es decir, y_{si} cuenta la reorganización del contenedor i si dicha reorganización ocurre al recuperar el contenedor s. Durante la recuperación de contenedores, existen relaciones físicas que las variables deben satisfacer. En la etapa e, cuando se reubica el contenedor s, las restricciones (1.15) se aseguran de que cada contenedor $i \geq s$ solo pueda estar en una ranura; por su parte las restricciones (1.16) se aseguran de que como máximo un contenedor pueda estar en una sola ranura. Las restricciones (1.17) aseguran que un contenedor no flote en el aire. Las restricciones (1.18) garantizan que un contenedor no se puede volver a reubicar en un espacio de la misma columna en una recuperación.

Las restricciones (1.19), (1.20), (1.21) y (1.22) consideran la altura relativa de los dos contenedores reorganizados a lo largo de los cuales se define la variable w_{eij}. Primero, se observa que (1.21) y (1.22) dictan que $w_{eij} = 0$ si alguno de los contenedores i,j no se reorganiza en la etapa e. Por lo tanto, si cualquiera de los y_{si} o y_{sj} es igual a cero, las restriccio-

nes (1.19) y (1.20) son redundantes. Por otro lado, si tanto y_{si} como y_{sj} son iguales a 1; es decir, ambos contenedores i y j se reorganizan en la etapa e, que es cuando se reubica el contenedor s, entonces, cuando el contenedor j está en una ranura más alta que la del contenedor i antes de la reorganización, las restricciones (1.19) dan $w_{eij} = 1$ y las restricciones (1.20) son redundantes. Cuando el contenedor j se encuentra en una ranura más baja que la del contenedor i antes de la reorganización, las restricciones (1.20) obligan a $w_{eij} = 0$ y las restricciones (1.19) son redundantes.

Las restricciones (1.23) consideran la altura relativa de los contenedores i y j en la etapa $e + 1$. Se puede comprobar que si el contenedor j no está más alto que el contenedor i en la etapa e, si alguno de los contenedores i y j no se reorganiza en la etapa e, o si el contenedor j no se reorganiza en la misma columna del contenedor i, las restricciones (1.23) son redundantes.

Luego, las restricciones (1.24) y (1.25) aseguran que los contenedores no reorganizados mantienen sus posiciones en la siguiente etapa. Por el mismo argumento, si el contenedor i se reorganiza en la etapa e, $y_{si} = 1$ y las restricciones (1.24) y (1.25) son redundantes. Como consecuencia directa de las restricciones (1.24) y (1.25), se garantiza que un contenedor reorganizado en una columna se almacenará en una ranura más alta que los contenedores que ya estaban allí.

Las restricciones (1.26) y (1.27) asignan valores conocidos a x_{eicp}. Desde la configuración inicial, el contenedor i se reorganiza primero, si es que se reorganiza, en la recuperación del contenedor más alto entre todos los contenedores debajo de él. Antes de esa etapa, el contenedor i mantiene su ranura inicial y $x_{eicp} = X_{1icp}$. Si el contenedor i no necesita ser reorganizado, mantiene su posición inicial hasta que se recupera. Las restricciones (1.28) y (1.29) corresponden a las restricciones de integralidad de las variables.

El MRIP tiene $2S(S-1)\frac{S(S+1)CP}{2} + \frac{(S-2)(S-1)S}{3}$ variables binarias en este programa de enteros, lo que se traduce en casi 7,000 variables binarias para una bahía típica de seis columnas y cuatro pilas a su máxima capacidad.

Capítulo 2

Metodología

2.1. Aplicabilidad del modelo de Reubicación Dinámica de Contenedores (DCRP)

Varios investigadores han abordado la problemática del DCRP desde diversas perspectivas. Así por ejemplo, Kim et al. [12] formulan un modelo de programación dinámica para determinar la ubicación óptima de almacenamiento de los contenedores de salida de modo que se minimice el número esperado de movimientos de reubicación, mientras que Kozan y Preston [15] presentan un enfoque integrador para determinar las ubicaciones óptimas de almacenamiento y los horarios de manipulación de los contenedores en las terminales.

Aunque el CRP ha sido ampliamente estudiado, el DCRP como tal aún no ha recibido la misma atención por parte de los investigadores. Por lo que se sabe, el primer trabajo en considerar el DCRP es el desarrollado por Wan et al. [21]; sin embargo, el foco principal de su estudio está en el CRP, visto en la sección anterior, y los autores adaptan los métodos heurísticos desarrollados para el CRP para resolver el DCRP.

Akyüz y Lee [2], proporcionan una formulación matemática para el DCRP mediante un modelo de programación lineal entera binaria, donde se considera la estructura física de la bahía para definir las variables de decisión y tiene en cuenta las horas de llegada y salida de los con-

tenedores. Adicionalmente presentan varias heurísticas de solución con resultados computacionales satisfactorios.

Por otro lado, Karpuzoglu et al. [11] proponen un enfoque heurístico basado en la búsqueda tabú para resolver el DCRP presentado por Akyüz y Lee [2], además realizan experimentos computacionales en un extenso conjunto de instancias de prueba generados aleatoriamente. Los resultados presentados muestran que el algoritmo propuesto es eficiente y produce resultados prometedores.

Akyüz [1] por su parte, aborda el problema de reubicación dinámica de contenedores tomando el modelo de programación lineal entera binaria presentado en Akyüz y Lee [2] con menos restricciones, con lo que consigue relajar la suposición que obliga a que las reubicaciones de contenedores se realicen solo para recuperar contenedores que deben abandonar la bahía. Luego, ofrece tres métodos heurísticos eficientes y realiza experimentos computacionales en un vasto conjunto de instancias de prueba estándar. Los resultados indicaron que se pueden lograr ahorros en tiempos de ejecución utilizando los métodos heurísticos sugeridos.

Borjian et al. [6] presentan un novedoso modelo en conjunto con su respectiva formulación matemática para la planificación de los movimientos en el patio de almacenamiento de terminales de contenedores enfocado en las grúas de patio. Aquí el objetivo de los autores es desarrollar una herramienta que se enfoque en las necesidades del cliente, en especial, en los tiempos de servicio y en disminuir los costos derivados de las reubicaciones. El modelo presentado incorpora varios detalles prácticos y ofrece a los operadores portuarios capacidades ampliadas que incluyen la planificación de movimientos y reubicaciones en el horizonte de planificación, el control de los tiempos de espera de cada cliente y el tiempo total de servicio, la optimización conjunta del número de reubicaciones y el tiempo de espera y la optimización simultánea del proceso de apilamiento y recuperación de contenedores. Adicionalmente se estudia un conjunto de políticas de servicio dúctiles que puedan permitir la recuperación fuera de servicio. De esta manera se logra mostrar que, con políticas flexibles, se puede disminuir el número de reubicaciones y demoras en la recuperación sin crear desigualdades.

Como se ha explicado en este trabajo, la recuperación de contenedores

de exportación de un patio de almacenamiento es una parte importante del proceso de carga de barcos durante el cual organizar la secuencia de recuperación para mejorar la eficiencia portuaria se ha convertido en un tema vital. El artículo presentado por Bian et al. [4], ofrece un algoritmo dinámico híbrido de dos fases que tiene como objetivo obtener una secuencia de carga de contenedores optimizada para que una grúa recupere todos los contenedores desde el patio de almacenamiento hasta el barco. El objetivo de optimización nuevamente es minimizar el número de operaciones de reubicación, lo que tiene un impacto directo en la eficiencia de la operación de carga de contenedores. Las dos fases del algoritmo dinámico propuesto están diseñadas de la siguiente manera: en la primera fase, se desarrolla un algoritmo heurístico para recuperar el subconjunto de contenedores que no necesita reubicación y puede cargarse directamente en el barco; en la segunda fase, se aplica una programación dinámica con reglas heurísticas para resolver el problema de la secuencia de carga del resto de contenedores. Los experimentos numéricos mostraron la efectividad y viabilidad del modelo y el algoritmo al compararlos con las propuestas de carga de una investigación existente y las reglas reales, respectivamente.

2.2. Esquema metodológico

Para encontrar una formulación del modelo DCRP para el puerto de Esmeraldas y lograr todos los objetivos planteados en este trabajo de integración curricular, se utiliza la metodología usualmente aplicada en los proyectos pertenecientes al campo de la Optimización Combinatoria, que, es una metodología investigativa que en este caso particular consta de tres etapas. La primera etapa es la formulación del modelo de programación lineal entera binaria. En la etapa dos se realizará la implementación del modelo de programación lineal entera utilizando el software propuesto. Por último, la tercera etapa consta de todas las pruebas computacionales realizadas.

2.2.1. Etapa 1: Formulación del modelo.

A diferencia del CRP, en el DCRP los contenedores también llegan a la bahía de depósito durante el tiempo de estudio. Como resultado, los supuestos del CRP presentados por Wan et al. [21] se amplían y se modifican de modo que los contenedores entrantes también se tienen en cuenta en el DCRP. Estos supuestos se describen a continuación:

- A1: La bahía es servida por una grúa de patio que puede manejar un solo contenedor a la vez.
- A2: Las secuencias de los eventos descritos por la llegada y recuperación de contenedores se conocen a priori.
- A3: Las reubicaciones se permiten solo en el período de tiempo de salida de los contenedores y un contenedor se puede reubicar como máximo una vez.
- A4: Las reubicaciones solo pueden ocurrir dentro de la bahía.

Bajo estos supuestos y tomando como base el modelo de optimización presentado por Akyüz and Lee en [2] se formula un modelo de programación lineal entera del DCRP para el problema en el puerto de Esmeraldas. El supuesto A1 implica que sólo se puede recibir un contenedor en la bahía o recuperarlo de la bahía en un período de tiempo. El supuesto A2 establece que el horizonte de planificación se puede dividir en períodos de tiempo discretos en los que un contenedor llega al patio o sale del patio. El supuesto A3 indica que cuando se debe recuperar un contenedor de destino de la bahía, la grúa del patio maneja cada contenedor que está por encima del contenedor que sale exactamente una vez. Se debe tener en cuenta que solo se puede llegar a las columnas de contenedores (pilas) desde su parte superior en una bahía de almacenamiento. Por lo tanto, para recoger un contenedor de una columna de contenedores, todos los contenedores que se encuentran arriba deben ser reubicados primero en otras columnas de la bahía y luego el contenedor objetivo puede salir de la bahía. El supuesto A3 indica que todos estos contenedores reubicados (reorganizados) pueden ser manejados solo una vez por la grúa de patio en el período de tiempo de salida del contenedor objetivo debajo de ellos.

Para decirlo de otra manera, no se permiten operaciones de clasificación previa de contenedores para reducir el número de futuras reubicaciones por adelantado. En adelante, cuando un contenedor sea referido como el "contenedor de reorganización", quiere decir que dicho contenedor se reposicionará para permitir la recuperación de otro contenedor debajo. El supuesto A4 limita el enfoque del DCRP a una sola bahía.

En la práctica, los camiones de patio se utilizan para transferir un contenedor de una bahía a otra. En tal caso, la grúa de patio primero recupera un contenedor y lo coloca en un camión de patio. Luego, el contenedor se transfiere a su bahía de destino. Una grúa de patio también se puede mover desde su bahía actual a la bahía de destino del contenedor cuando ya no hay una grúa de patio asignada para la bahía de destino. Estas reubicaciones, que necesitan camiones de patio, se evitan en la medida de lo posible en las terminales por razones de seguridad y debido a sus características que consumen mucho tiempo [16]. En consecuencia, si la bahía consta de W columnas y H filas, el supuesto A4 también establece que no hay más de $(W-1)H+1$ contenedores que permanezcan en la bahía durante todo el horizonte de planificación en un período de tiempo dado. El supuesto A5 indica que no se tienen en cuenta diferentes tipos de contenedores y que una bahía puede contener un solo tipo de contenedor.

Sean J, K y T el conjunto de contenedores, el conjunto de ranuras en la bahía y el conjunto de períodos de tiempo que se indican con los índices j, k y t, respectivamente. Se define $T_j = \{\tau_j^a, \tau_j^{a+1}, ..., \tau_j^d\}$ como el conjunto de períodos de tiempo que un contenedor j permanece en la bahía donde $T = \bigcup_{j \in J} T_j$; es decir, que τ_j^a y τ_j^d son períodos de tiempo específicos que denotan los períodos de tiempo de llegada y salida del contenedor j. Sin pérdida de generalidad, se supone que las ranuras en la bahía están numeradas en orden creciente desde la parte inferior izquierda a la parte superior derecha de la bahía como se muestra en la Figura 2.1.

Sea O el conjunto de ranuras que están en la parte superior de cada columna. También definimos $W(k)$ como el índice de columna de la ranura k. El parámetro $\delta_k^{k'}$ describe las relaciones físicas entre las ranuras de la bahía. Si la ranura k' está ubicada físicamente por encima de la ranura k, entonces $\delta_k^{k'}$ es igual a 1 y 0 en caso contrario.

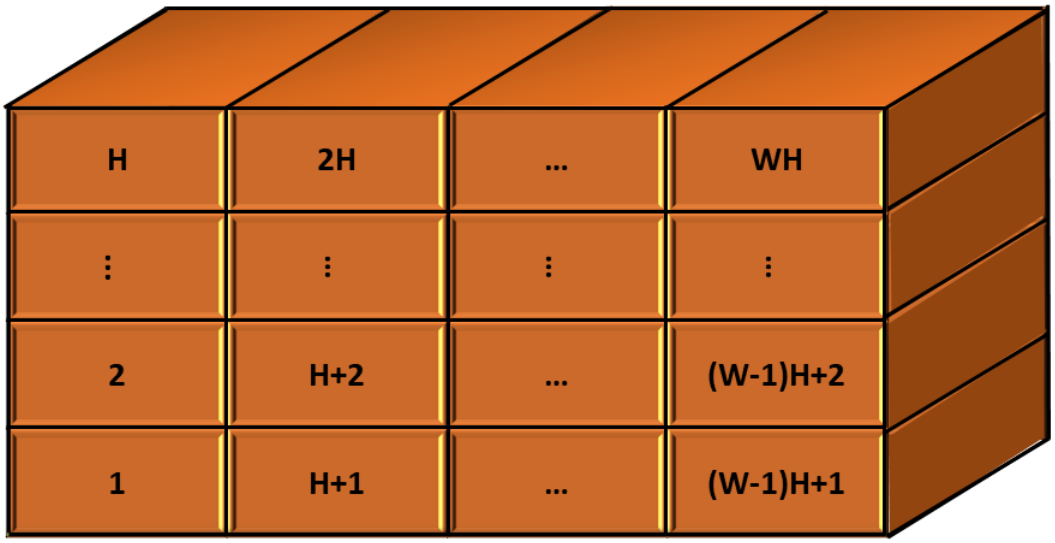


Figura 2.1: Enumeración de ranuras en una bahía con W columnas y H filas

Las variables de decisión binarias x_{jk}^t y y_j^t se establecen de la siguiente manera:

$$x_{jk}^t = \begin{cases} 1, & \text{si el contenedor } j \text{ está ubicado en la ranura } k \text{ en el período de tiempo } t. \\ 0, & \text{en caso contrario.} \end{cases}$$

$$y_j^t = \begin{cases} 1, & \text{si y solo si el contenedor } j \text{ se reubica en el período de tiempo } t. \\ 0, & \text{en caso contrario.} \end{cases}$$

Entonces, la formulación entera binaria del DCRP se puede establecer como sigue:

$$Min\ Z = \sum_{t \in \{T_j \setminus \tau_j^d\}} \sum_{j \in J} y_j^t \tag{2.1}$$

Sujeto a :

$$\sum_{\{j\in J: t\in T_j\}} x_{jk}^t \leq 1 \ , \ \forall k \in K; \ \forall t \in T \tag{2.2}$$

$$\sum_{\{k\in K\}} x_{jk}^t = 1 \ , \ \forall t \in T_j; \ \forall j \in J \tag{2.3}$$

$$\delta_k^{k+1}(\sum_{\{j\in J: t\in T_j\}} x_{j,k+1}^t - \sum_{\{j\in J: t\in T_j\}} x_{j,k}^t) \leq 0 \ , \ \forall k \in \{K \setminus O\}; \ \forall t \in T \tag{2.4}$$

$$\delta_k^{k+1}(x_{j,k+1}^t + x_{j'k}^t) - y_j^t \leq 1 \ , \ \forall k \in \{K \setminus O\}; t = \tau_j^d \ \forall j = j' \in J \tag{2.5}$$

$$x_{jk}^t - \sum_{\{j'\in\{J\setminus j\}: t=\tau_{j'}^d\}} \sum_{k'\in K} \delta_{k'}^k x_{j',k'}^t \leq 1 - y_j^t \ , \ \forall k \in K; \ \forall t \in T_j; \ \forall j \in J \tag{2.6}$$

$$\begin{aligned} x_{jk}^t - x_{jk}^{t+1} \leq y_j^t \ , \ \forall k \in K; \ \forall t \in \{T_j \setminus \tau_j^d\}; \ \forall j \in J \\ x_{jk}^{t+1} - x_{jk}^t \leq y_j^t \ , \ \forall k \in K; \ \forall t \in \{T_j \setminus \tau_j^d\}; \ \forall j \in J \end{aligned} \tag{2.7}$$

$$\begin{aligned} x_{jk}^t + x_{jk}^{t+1} + \sum_{k'\in K} \delta_{k'}^k x_{jk'}^{t+1} + \sum_{k'\in K} \delta_k^{k'} x_{jk'}^{t+1} \leq 2 - y_j^t \ , \\ \forall k \in K; \ \forall t \in \{T_j \setminus \tau_j^d\}; \ \forall j \in J \end{aligned} \tag{2.8}$$

$$\begin{aligned} x_{jk}^t + \sum_{k''\in K} \delta_k^{k''} x_{j'k''}^t + x_{jk'}^{t+1} + \sum_{k''\in K} \delta_{k'}^{k''} x_{j'k''}^{t+1} + y_j^t + y_{j'}^t \leq 5 \ , \\ \forall k, k', k'' \in \{K \setminus O\}; \ W(k) \neq W(k'); \ t \in \{T_j \setminus \tau_j^d\} \bigcap \{T_{j'} \setminus \tau_{j'}^d\}; \\ \forall j \neq j' \in J \end{aligned} \tag{2.9}$$

$$x_{jk}^t \in \{0,1\} \quad \forall k \in K; \ \forall t \in T_j; \ \forall j \in J \tag{2.10}$$

$$y_j^t \in \{0,1\} \quad \forall t \in \{T_j \setminus \tau_j^d\}; \ \forall j \in J \tag{2.11}$$

$$\begin{aligned} x_{jk}^1 = 0 \quad \forall j \in J^1; \ \forall k \in K^1 \\ x_{jk}^1 = 1 \quad \forall j \in J^1; \ \forall k \notin K^1 \end{aligned} \tag{2.12}$$

La función objetivo (2.1) consiste en el número total de reubicaciones realizadas a lo largo del horizonte de planificación. Las restricciones (2.2) permiten ubicar como máximo un contenedor en una ranura k para cada período de tiempo t. Las restricciones (2.3) garantizan que cada contenedor j debe estar ubicado en una ranura k mientras permanezca en la bahía, es decir, $t \in T_j$. Las restricciones (2.4) establecen que un contenedor solo puede ubicarse en una ranura $k+1$ si y solo si la ranura k ya está ocupada. El parámetro δ_k^{k+1} impone que la ranura k esté debajo

de la ranura $k+1$ cuando $\delta_k^{k+1} = 1$, y las restricciones (2.4) se vuelven redundantes en caso contrario.

Las restricciones (2.5) y las restricciones (2.6) definen la relación entre las variables de asignación de ubicación x_{jk}^t y las variables de reubicación y_j^t. Las restricciones (2.5) implican que, si el contenedor j está ubicado en una ranura superior del contenedor j', que sale antes que el contenedor j en el momento $\tau_{j'}^d$, entonces el contenedor j se reubica y y_j^t se obliga a tomar el valor 1. Por el contrario, las restricciones (2.6) obligan a y_j^t a tomar un valor cero cuando no hay ningún contenedor para recoger en una ranura más baja que el contenedor j en el período de tiempo t. El segundo término en el lado izquierdo de las restricciones (2.6) suma el número de contenedores debajo de la ranura k que salen antes del contenedor j en el período de tiempo $t = \tau_{j'}^d$. No obstante, podría haber como máximo uno de estos contenedores j' en un período de tiempo t. Si no existe tal contenedor j', entonces y_j^t es igual a cero. De lo contrario, las restricciones (2.6) son redundantes. Las restricciones (2.7) aseguran que los contenedores, que no se reubican en el período de tiempo t, mantengan su ubicación para el próximo período de tiempo $t+1$. Esto significa que continúan ocupando el mismo espacio k en el período de tiempo $t+1$ que en el período de tiempo t; es decir, $x_{jk}^t = x_{jk}^{t+1}$.

Las restricciones (2.8) imponen que un contenedor j reubicado en el período de tiempo t no se puede colocar en la misma columna para el siguiente período de tiempo $t+1$. Cuando $y_j^t = 0$, el contenedor j continúa ocupando su espacio actual k para el cual la restricción (2.8) es vinculante siguiendo las restricciones (2.7). Para los espacios restantes, es decir, $k' \in K \setminus k$, que no están ocupados por el contenedor j, todos los términos del lado izquierdo son cero y las restricciones (2.8) se vuelven redundantes. Cuando $y_j^t = 1$ y $x_{jk}^t = 1$ para una ranura $k \in K$, el contenedor j se reubica en otra columna que no sea la de la ranura k. Los últimos tres términos en el lado izquierdo de las restricciones (2.8), $x_{jk}^{t+1} + \sum_{k' \in K} \delta_{k'}^k x_{jk'}^{t+1} + \sum_{k' \in K} \delta_k^{k'} x_{jk'}^{t+1}$, representan los espacios ocupados por el contenedor j en la misma columna con el espacio k en el siguiente período de tiempo $t+1$. Observe que $x_{jk}^{t+1} + \sum_{k' \in K} \delta_{k'}^k x_{jk'}^{t+1} + \sum_{k' \in K} \delta_k^{k'} x_{jk'}^{t+1} \leq 1$ ya está satisfecho por las restricciones (2.3). Claramente, si $x_{jk}^t = 1$ e $y_j^t = 1$, todos los términos restantes en el lado izquierdo de las restricciones (2.8)

se hacen cero para garantizar que el contenedor j se reubique en otra columna que no sea la de la ranura k en el período de tiempo $t+1$. Cuando $x_{jk}^t = 0$ e $y_j^t = 1$, las restricciones (2.8) son redundantes para las ranuras que tienen la misma columna con la ranura k y son vinculantes para el resto de las ranuras.

Las restricciones (2.9) tienen como objetivo satisfacer el supuesto A3. Recuerde que las columnas son atendidas por una grúa de un solo patio que puede recoger un solo contenedor de la parte superior de una columna. Por lo tanto, para recuperar un contenedor de destino, que se encuentra en la ranura debajo de algunos otros contenedores, se deben limpiar los contenedores de arriba del contenedor de destino. Esto se logra reubicando esos contenedores de reorganización uno por uno, con una secuencia que comienza desde las ranuras superiores hasta las ranuras inferiores de la columna.

Claramente, según el supuesto A3, cada contenedor puede reubicarse solo una vez en cada período de tiempo. Como resultado, las nuevas posiciones de los espacios de cualquier par de contenedores de reorganización deben estar en el orden inverso de su secuencia de recuperación en caso de que se coloquen en la misma columna después de su reubicación, es decir los que están en una ranura superior deben estar en una ranura inferior y viceversa después de que se reubiquen con respecto a sus posiciones de ranura anteriores.

Para lograr esto, las restricciones (2.9) se vuelven activas cuando dos contenedores j y j' cualesquiera se reubican en el período de tiempo t. Observe que cuando y_j^t o $y_{j'}^t$ es cero, las restricciones (2.9) son redundantes. Para el caso en el que j y j' se reubican, es decir, $y_j^t = y_{j'}^t = 1$, las restricciones (2.9) pueden ser vinculantes y verifican las ranuras de los contenedores j y j' en los períodos de tiempo t y $t+1$. Si en el período de tiempo t, el contenedor j está ubicado debajo del contenedor j', entonces las restricciones (2.9) imponen que el contenedor j no se puede reubicar en una ranura k' debajo del contenedor j' en el siguiente período de tiempo $t+1$. La suma de los dos primeros términos de las restricciones (2.9), $x_{jk}^t + \sum_{k'' \in K} \delta_k^{k''} x_{j'k''}^t$, es igual a 2 si el contenedor j está ubicado en la ranura k que está debajo del contenedor j' en el período de tiempo t. Observe que la desigualdad $\sum_{k'' \in K} \delta_{k'}^{k''} x_{j'k''}^{t+1}$ ya se deriva de las restricciones

(2.3). Entonces, el tercer término o el cuarto término de las restricciones (2.9) se hacen cero y esto garantiza que el orden de reorganización de los contenedores satisface el supuesto A3 en el modelo mencionado. En resumen, para satisfacer el supuesto A3, no todos los términos del lado izquierdo son iguales a 1 al mismo tiempo. Las restricciones (2.9) logran esto restringiendo la suma de su lado izquierdo para que sea menor o igual a 5. Para todos los demás casos, estas restricciones ya son redundantes. En las restricciones (2.9), $W(k) \neq W(k')$ implica que la ranura k y la ranura k' están en columnas diferentes de la bahía.

Las restricciones (2.10) y las restricciones (2.11) representan la restricción de integralidad de las variables de decisión x_{jk}^t y y_j^t, respectivamente. Observe que el contenedor j se recupera de la bahía al final de su estancia cuando $t = \tau_j^d$. Esto implica que el contenedor j no se reubica en su último período de tiempo $t = \tau_j^d$ sino que se recupera de la bahía. De lo contrario, el contenedor j debe permanecer un período de tiempo más hasta $\tau_j^d + 1$ en la bahía. Por lo tanto, las variables de reubicación y_j^t se pueden establecer directamente en cero en los períodos de tiempo de salida τ_j^d de cada contenedor j y no se consideran en la formulación.

Las asignaciones iniciales de espacios de contenedores, que ya están en la bahía, pueden imponerse con las restricciones (2,12). Donde J^1 denota el conjunto de contenedores que ya existen en la bahía y K^1 representa el conjunto de ranuras que no están ocupadas por el contenedor j. Por tanto, $x_{jk}^1 = 1$ con $k \notin K^1$ indica la ranura k en la que se asigna inicialmente el contenedor j. Por lo tanto, la formulación de DCRP se convierte en una IP binaria denotada por (1)-(12). Es posible pensar que definir las variables x_{jk}^t solo en sus horas de llegada y salida será suficiente y podría salvarnos de muchas variables binarias a primera vista; sin embargo, es posible que sea necesario reubicar un contenedor en cualquier período de tiempo en el que deba permanecer en el patio. Es decir, no hay garantía de que un contenedor permanezca en la misma ranura hasta que salga. De hecho, algunos contenedores requieren ser reubicados más de una vez dentro de los períodos de tiempo que permanecen en la bahía.

Hay como máximo $(\mathrm{K}+1)\sum_{j\in J}|T_j| - \mathrm{J}$ variables binarias en la formulación DCRP donde $\mathrm{K} = WH = |K|$ es el número total de ranuras y $\mathrm{J} = |J|$ es el número de contenedores que se servirán en la bahía. Co-

mo, por ejemplo, dada una bahía con seis columnas de altura cuatro $(\mathrm{K} = 24)$ que alberga 10 contenedores, cada uno pasando cinco períodos de tiempo, el número de variables binarias es 1200 y 40 para x_{jk}^{t} e y_{j}^{t}, respectivamente. Por lo tanto, para este ejemplo de tamaño pequeño, el número total de variables binarias es más de 1200. Desafortunadamente, para los problemas que tienen horizontes de planificación más largos y más contenedores, resolver la formulación DCRP de manera óptima puede ser computacionalmente exigente. En consecuencia, los métodos heurísticos resultan más razonables que resolver un DCRP completo para obtener soluciones aproximadas suficientemente buenas. Por el alcance de este trabajo de integración curricular se implementa el modelo de programación lineal entera para el DCRP completo, motivo por el cual no se implementan heurísticas.

2.2.2. Etapa 2: Implementación del modelo.

La implementación del modelo de programación lineal entera binaria y del método de solución diseñado en la etapa anterior se realizó en el lenguaje Python, mediante el uso del software libre Anaconda (Anaconda 4.10.3, 2021), conectado con el solver especializado de optimización Gurobi (Gurobi 9.5.0, 2021).

Se debe señalar también, que los experimentos se realizan en una computadora portátil modelo HP Pavilion 14 Notebook PC con un procesador Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz, 2601 Mhz, que cuenta con 2 procesadores principales y 4 procesadores lógicos de cuarta generación. Una memoria RAM instalada de 8.00GB y un sistema operativo Windows de 64 bits, edición Windows 10 Home Single Language y versión 21H2.

2.2.3. Etapa 3: Pruebas computacionales.

En esta etapa se encuentran las pruebas computacionales, constituidas por varias instancias con datos reales, obtenidas de las bases de datos del flujo de contenedores proporcionadas por APE. Se analiza la validez de las soluciones y su aplicabilidad, y se generan reportes con los

resultados finales.

El patio de contenedores de esta locación cuenta con un área de depósito de contenedores, dividida en dos zonas, adecuadas para contenedores llenos y para contenedores vacíos. El patio exclusivo para contenedores llenos cuenta con una capacidad de almacenamiento de 3000 contenedores y el patio exclusivo para contenedores vacíos cuenta con una capacidad de almacenamiento de 1800 contenedores. Es preciso señalar que dentro de las instalaciones de APE existen patios alternativos que pueden ser ocupados en caso de completar la capacidad de almacenamiento en los patios principales. Adicionalmente se cuenta con 5 porta-contenedores para las operaciones de contenedores llenos y 2 para las operaciones de contenedores vacíos. En el área de contenedores llenos, figuran varios bloques de almacenamiento, donde dichos contenedores son organizados en bahías de 3 columnas y 2 filas, generalmente, aunque en los días de mayor afluencia, los bloques se organizaban hasta en bahías de 3 columnas y 4 filas.

En la Figura 2.2, se presenta un extracto del registro del flujo de contenedores del puerto de Esmeraldas para el año 2019. Aquí se observa varias características relevantes, como las dimensiones de ancho, alto y profundidad de cada uno de los contenedores; sus fechas y horas de entrada y salida, lo que nos indica el número de días de permanencia de los contenedores en el patio de almacenamiento, entre otras. Para la construcción de las instancias usadas en los experimentos computaciones y debido a que el modelo DCRP no soporta contenedores con diferentes medidas, se toman únicamente los contenedores de 40 pies DRY HC, que, junto al contenedor de 20 pies, es la unidad de transporte más utilizada en transporte marítimo, según la UNCTAD [9].

El flujo de contenedores para el año 2019 no mostró mayor movimiento en el puerto de Esmeraldas, de hecho se consideró como flujo moderado bajo [20]. En la Figura 2.3 se muestra un breve resumen del tráfico de contenedores en el patio de almacenamiento para los meses de enero, febrero y marzo. Al observar las bases cedidas por APE fue posible percibir que un porcentaje mayor de los movimientos se dan en el patio de almacenamiento para contenedores de importación, en vista de esto, el análisis se centró en el patio de almacenamiento para contenedo-

res de importación y las instancias usadas fueron construidas bajo estos registros.

NÚMERO DE CONTENEDOR	CLIENTE	TAM	TIPO	ALTO	EXPORT / IMPORT CONT. FULL	FECHA Y HORA DE ENTRADA	FECHA Y HORA DE SALIDA	DÍAS DE PERMANENCIA	DÍAS A FACTURARSE
SUDU8199325	MAERSK	40	DRY	HC	Exp / Mty	07/02/19 / 09:18	30/03/19 / 06:00	51	41
MNBU3249198	MAERSK	40	DRY	HC	Exp / Mty	07/02/19 / 09:29	30/03/19 / 06:00	51	41
MNBU3596366	MAERSK	40	DRY	HC	Exp / Mty	07/02/19 / 09:33	30/03/19 / 06:00	51	41
MWCU6879572	MAERSK	40	DRY	HC	Exp / Mty	08/02/19 / 10:25	30/03/19 / 06:00	50	40
PONU4977332	MAERSK	40	DRY	HC	Exp / Mty	08/02/19 / 10:45	30/03/19 / 06:00	50	40
PONU4500632	MAERSK	40	DRY	HC	Exp / Mty	11/02/19 / 09:45	30/03/19 / 06:00	47	37
MNBU4043935	MAERSK	40	DRY	HC	Exp / Mty	12/02/19 / 20:51	30/03/19 / 06:00	45	35
MWCU6727800	MAERSK	40	DRY	HC	Exp / Mty	14/02/19 / 20:27	30/03/19 / 06:00	43	33
MNBU3111228	MAERSK	40	DRY	HC	Exp / Mty	17/02/19 / 08:41	30/03/19 / 06:00	41	31
MWCU6869872	MAERSK	40	DRY	HC	Exp / Mty	17/02/19 / 10:20	30/03/19 / 06:00	41	31
MWCU6880022	MAERSK	40	DRY	HC	Exp / Mty	20/02/19 / 17:56	30/03/19 / 06:00	38	28
MWCU5288662	MAERSK	40	DRY	HC	Exp / Mty	27/02/19 / 12:49	30/03/19 / 06:00	31	21
GATU8685163	MAERSK	40	DRY	HC	Exp / Mty	27/02/19 / 19:27	15/03/19 / 14:10	16	6
FCIU4265271	MAERSK	20	DRY	STD	Exp / Mty	01/02/19 / 11:58	16/02/19 / 00:30	15	5
MSKU7161770	MAERSK	20	DRY	STD	Exp / Mty	01/02/19 / 12:00	16/02/19 / 00:30	15	5
SUDU7718771	MAERSK	20	DRY	STD	Exp / Mty	01/02/19 / 12:05	16/02/19 / 00:30	15	5
MSKU4499320	MAERSK	20	DRY	STD	Exp / Mty	01/02/19 / 12:07	16/02/19 / 00:30	15	5
MRKU6177825	MAERSK	40	DRY	HC	Exp / Mty	28/02/19 / 08:44	15/03/19 / 14:10	15	5
MSKU1371721	MAERSK	40	DRY	HC	Exp / Mty	28/02/19 / 08:59	15/03/19 / 14:10	15	5
MRKU4933570	MAERSK	40	DRY	HC	Exp / Mty	28/02/19 / 09:14	15/03/19 / 14:10	15	5
MSKU0239500	MAERSK	40	DRY	HC	Exp / Mty	28/02/19 / 09:24	15/03/19 / 14:10	15	5
MRKU3730710	MAERSK	40	DRY	HC	Exp / Mty	28/02/19 / 09:38	15/03/19 / 14:10	15	5
SUDU8725029	MAERSK	40	DRY	HC	Exp / Mty	28/02/19 / 10:03	15/03/19 / 14:10	15	5

Figura 2.2: Detalle de registro del flujo de contenedores del puerto de Esmeraldas en 2019.

REGISTRO DE ARRIBO Y SALIDA DE CONTENEDORES 2019				
FECHAS DE ARRIBO	IMPORTACION	EXPORTACION VACIO	EXPORTACION LLENO	TOTAL MOVIMIENTOS
04/01/2019	99	0	0	198
11/01/2019	212	0	0	212
18/01/2019	69	301	5	508
25/01/2019	106	0	0	318
07/02/2019	248	548	5	1292
15/02/2019	84	84	4	168
22/02/2019	152	109	2	261
28/02/2019	107	89	0	196
08/03/2019	70	0	0	210
14/03/2019	119	88	2	207
21/03/2019	65	95	1	225
29/03/2019	93	220	1	499

Figura 2.3: Resumen de arribo y salida de contenedores del puerto de Esmeraldas en 2019.

Fundamentado en las instancias generadas por Akyüz y Lee [2], se construyó instancias de prueba con los datos proporcionados por la Autoridad Portuaria de Esmeraldas, correspondientes al año 2019. Las instancias elaboradas encajaron con las instancias descritas como del Grupo

I, para las cuales en su construcción se crea un suceso (llegada o salida de contenedores) en cada período de tiempo. Luego, se decide aleatoriamente si el suceso es para un contenedor que llega o un contenedor que sale de la bahía. Este esquema utiliza unidades de tiempo discretas para generar instancias y asocia un período de tiempo distinto para cada evento. Luego, se asocia un contenedor con cada intervalo de tiempo de llegada y salida para generar las instancias.

Para las instancias del Grupo I, un contenedor puede permanecer dentro de la bahía durante cualquier período de tiempo hasta el final del horizonte de planificación. En el caso particular de este trabajo de integración curricular, al contar con los registros de entrada y salida de los contenedores a analizar, fue suficiente realizar esta asignación a los contenedores correspondientes dentro del horizonte de planificación.

Capítulo 3

Resultados, conclusiones y recomendaciones

3.1. Resultados

A continuación se exhibe en cuadros, un resumen de los resultados obtenidos de las pruebas computacionales realizadas. Se debe señalar que en las cuadros presentados, "Número de contenedores" se refiere al número de contenedores recuperados de las bahías, "Dimensiones" indica el tamaño de las bahías, donde "W" es el número de columnas y "H" el número de filas. La columna "Objetivo" señala el número de reubicaciones reportado por el modelo como solución para una respectiva instancia, el "gap(%)" es la brecha relativa de minimización del modelo de programación entera binaria, "# Tiempo(s)" representa el tiempo en segundos que tardó en ejecutarse el modelo y reportar un resultado para la instancia dada, "# Nodos" es el número de nodos recorridos por el modelo, "# Variables" indica la cantidad de variables del modelo y, "# Restricciones" es el número de restricciones del modelo para la instancia correspondiente.

Las pruebas computaciones fueron dividas en tres enfoques a la hora de su realización. El primer enfoque estudió el problema subdividiendo el horizonte de planificación en días, logrando así 365 períodos a los largo del año. En el segundo enfoque, donde se subdividió el horizonte de planificación en semanas, se obtuvo 52 períodos. En el tercer y último enfoque se subdividió el horizonte de planificación en meses, resultando

12 períodos. Cabe recalcar que para cada instancia dentro de cada enfoque de estudio el número de períodos del modelo equivale al número de períodos (días, semanas o meses) que transcurrieron desde la llegada del primer contenedor hasta la salida del último contenedor de dicha instancia. En el cuadro 3.1 se muestra el resumen del desempeño del DCRP para los enfoques estudiados, aquí se presentan los promedios de las reubicaciones y de los tiempos de ejecución para cada uno de los enfoques según el número de contenedores recuperados. Los cuadros 3.2, 3.3 y 3.4 muestran a detalle los resultados obtenidos para los enfoques 1, 2 y 3, respectivamente.

De esta manera, y acorde con las soluciones obtenidas de la implementación del modelo de programación entera binaria del DCRP aplicado a instancias del puerto comercial de Esmeraldas se han obtenido los siguientes resultados:

Cuadro 3.1: Resumen del desempeño del DCRP

Número de contenedores	**Instancias diarias**		**Instancias semanales**		**Instancias mensuales**	
	Objetivo promedio	**# Tiempo(s)**	**Objetivo promedio**	**# Tiempo(s)**	**Objetivo promedio**	**# Tiempo(s)**
5	3	135.83	1.33	0.73	1.33	2.68
10	2	728.53	2.33	362.1	3.67	739
12	6	993.85	3	1035.41	3	2722.22

Cuadro 3.2: Desempeño del DCRP bajo el Enfoque 1

Número de contenedores	**Dimensiones (W,H)**	**Reporte de Gurobi - Instancias diarias**					
		Objetivo	**Gap(%)**	**# Tiempo(s)**	**# Nodos**	**# Variables**	**# Restricciones**
5	(3,2)	4	0.0%	0.51	256	26580	5354
	(3,3)	0	0.0%	0.37	1	10475	39720
	(3,4)	2	50.0%	406.62	2483	110125	41204
10	(3,2)	3	60.3%	808.67	631	2097	30720
	(3,3)	2	70.1%	530.93	670	26560	37640
	(3,4)	1	75.0%	846	1079	101269	38488
12	(3,2)	5	88.9%	976.87	2332	2184	33660
	(3,3)	8	63.6%	1422.45	4563	23446	31870
	(3,4)	5	0.0%	582.23	5681	83695	32934

Cuadro 3.3: Desempeño del DCRP bajo el Enfoque 2

Número de contenedores	Dimensiones (W,H)	Reporte de Gurobi - Instancias semanales					
		Objetivo	Gap(%)	# Tiempo(s)	# Nodos	# Variables	# Restricciones
5	(3,2)	4	83.3%	0.44	176	1164	2177
	(3,3)	0	0.0%	0.38	1	5180	1596
	(3,4)	0	0.0%	1.38	1	2028	9551
10	(3,2)	2	88.3%	755.51	3740	2246	8140
	(3,3)	3	92.7%	153.1	1141	2796	5187
	(3,4)	2	75.0%	177.7	3191	3652	26083
12	(3,2)	2	55.5%	165.7	2489	1078	21770
	(3,3)	2	81.2%	867.53	4991	4142	23640
	(3,4)	5	50.0%	2073	5392	16225	34603

Cuadro 3.4: Desempeño del DCRP bajo el Enfoque 3

Número de contenedores	Dimensiones (W,H)	Reporte de Gurobi - Instancias mensuales					
		Objetivo	Gap(%)	# Tiempo(s)	# Nodos	# Variables	# Restricciones
5	(3,2)	4	0.0%	1.73	186	1008	2177
	(3,3)	0	0.0%	1.47	1	1440	5180
	(3,4)	0	0.0%	4.84	1	1872	9551
10	(3,2)	3	91.7%	361.55	1478	2680	12064
	(3,3)	6	55.5%	837.6	4605	4508	31280
	(3,4)	2	0.0%	1017.86	5911	4225	31603
12	(3,2)	1	73.7%	401.31	1146	3620	22960
	(3,3)	0	0.0%	41.34	1	5425	63766
	(3,4)	8	87.3%	7724	25463	4225	35063

Se pudo observar que el modelo DCRP funciona bastante bien y es veloz a la hora de reportar resultados cuando el número de contenedores a recuperar en las bahías es menor, pues en estos casos se obtuvo mejores soluciones en cuanto a tiempos de ejecución y número de reubicaciones. Claramente cuando el número de contenedores recuperados fue 5, el DCRP trabajó mejor y más rápido, en los tres enfoques propuestos, pues halló la solución óptima en un gran número de casos, y sus tiempos de ejecución fueron bajos la mayoría de veces.

Por otra parte, se debe notar que mientras más contenedores se debían recuperar, a pesar de que las bahías pudieran o no ser más grandes,

existe la tendencia de no alcanzar la solución óptima y de reportar tiempos de ejecución muchos mayores a los presentados con el número de contenedores recuperados más bajo. Además, se advierte que el modelo reportó menos reubicaciones y menores tiempos de ejecución para las instancias semanales. Asimismo, considerando el gap, se logra evidenciar que el mejor enfoque fue el tercero (para instancias mensuales), pues se alcanza la solución óptima en 5 de las 9 instancias estudiadas.

Cabe resaltar que al no contar con los registros detallados de los movimientos realizados por cada contenedor que ingresó al patio de almacenamiento de APE, no se puede realizar una comparación minuciosa y real en cuanto a número de reubicaciones se refiere. Sin embargo los resultados mostraron que el número de reubicaciones realizadas en bastante bajo en la mayoría de los casos para las instancias tratadas.

3.2. Conclusiones y recomendaciones

En este trabajo, se presentó un modelo basado en los tiempos de llegada y salida de los contenedores, puesto que se contaba con dicha información de antemano, esto permitió explorar una formulación de programación entera binaria que explota dicha información.

Se revisaron diversos artículos donde se proponen y estudian heurísticas utilizando enfoques diferentes para el CRP y DCRP, dado que por la complejidad del modelo, era extremadamente difícil lograr la solución óptima para las instancias generadas. Debido a que la construcción de las instancias usadas en los experimentos computacionales de este trabajo estaban ligadas a la realidad del puerto de la ciudad Esmeraldas, el cual es un puerto relativamente pequeño, estas instancias resultaron bastante reducidas en cuanto a complejidad, lo que desembocó en tiempos razonables de trabajo computacional salvo para la instancia que tardó 7724 segundos. En cuanto a soluciones, en 10 de las 27 instancias (37 %) se alcanzó la solución óptima, asimismo, en 9 instancias (33.3 %), el gap fue de al menos el 75 %, mientras que en las instancias restantes el gap resultó menor o igual al 73.7 %.

La literatura nos indica que la gestión de los movimientos de reubi-

cación de contenedores es uno de los principales desafíos en el patio de almacenamiento de las terminales y tiene un efecto directo en los costos y la eficiencia de las operaciones del patio. Al investigar sobre las operaciones del patio de almacenamiento de APE, se pudo notar que el tiempo de salida de los contenedores no se conoce de antemano en la gran mayoría de casos. De esta manera, considerar esta información como un dato a priori sería poco realista. Por lo tanto, relajar la suposición de que el orden de recuperación de todos los contenedores se conoce inicialmente sería de mucha utilidad. En el modelo DCRP presentado, se asume que el orden de recuperación de un subconjunto de contenedores presentes en la bahía al inicio del horizonte de planificación se conoce originalmente tanto como el orden de recuperación de los contenedores que arriben a lo largo de este tiempo. Se debe notar que esta suposición es particularmente engañosa y poco práctica en puertos sin un sistema de citas como el de APE.

Adicionalmente, al caso particular de este trabajo de integración curricular, se debe tener en cuenta que en un caso real al conocerse los tiempos de entrada y salida de los contenedores, los operadores portuarios tendrían la posibilidad de gestionar arbitrariamente el reposicionamiento y la planificación de los movimientos de los contenedores, en especial si estos llegan y salen continuamente. Por lo tanto, para futuras formulaciones se recomienda tener en cuenta elementos relacionados con el servicio al cliente, como la demora en el servicio de recuperación de un contenedor. Este enfoque fue tratado por Borjian et al. en [6], donde presentan una política de planificación de recuperación flexible para el CRP y estudian el impacto de ésta en el número de reubicaciones y en el retraso en las recuperaciones. Allí presentan que dicha política mejora las operaciones al disminuir o mantener sin cambios el número de reubicaciones mientras que el tiempo de espera de los camiones externos se acorta o no se ve afectado.

Además, parece que usar tiempos de servicio en el modelo es más conveniente cuando se resuelve el DCRP con información incompleta (por ejemplo, cuando tenemos ventanas de tiempo como una estimación de los tiempos de salida y llegada de los contenedores, o alguna distribución probabilística sobre los tiempos de servicio), así lo indican Borjian

et al. en [5]. Por lo tanto, estudiar el CRP o el DCRP con información incompleta en un marco basado en el tiempo es otra dirección importante para posteriores investigaciones, así los esfuerzos para diseñar heurísticas basadas en el tiempo son una temática interesante para el trabajo futuro.

Otros sistemas de almacenamiento, como el apilamiento de placas de acero y los sistemas de almacenamiento de bloques, enfrentan el problema de la reubicación con información incompleta o incluso nula, de donde se genera otro problema interesante de estudiar para el DCRP, así lo es el caso en el que no se conoce de antemano información de los tiempos de llegada ni de recuperación de ningún contenedor. Intuitivamente, parece que la política de colocar el contenedor en la columna más vacía debería ser la solución óptima; sin embargo, esto queda por demostrar.

Capítulo A

Código Python del Problema de Reubicación Dinámica de Contenedores

```python
#Llamado de Gurobi y de las librerías a usar.
from gurobipy import *
import random
import sys
import math
import re
```

```python
# Conjuntos del modelo.

# Conjunto de ranuras.
w =      # número de columnas en la bahía.
h =      # número de contenedores posibles en una columna (filas-altura).
wh = w*h  # número de ranuras en la bahía.
K = set(range(1, wh+1)) # ranuras enumeradas.

# Conjunto de ranuras superiores.
# Es el conjunto de ranuras superiores en cada columna.
O=set()
j=h
i=1
while i <= w:
    O.add(j)
    j=j+h
    i+=1

# Conjunto de períodos de tiempo.
per =  # Número de períodos en que se divide el horizonte de planificación.
T = tuplelist(range(1, per+1))
```

```
# Parámetos del modelo

# Parámetro de columna de las ranuras.
# Definimos W(k) como el índice de columna de la ranura k.
Wk = tupledict({})
for j in K:
    d = (j/h)
    for k in range(1, w+1, 1):
        if k-1 < d <= k:
            for i in range(1, j+1):
                Wk[j] = k

# Parámetro de fila de las ranuras.
Hk = tupledict({})
i = int(1)
for j in K:
    Hk[j] = i
    if i == h:
        i = int(1)
    else:
        i = i+1

# Parámetro de referencia entre dos ranuras.
delta = tupledict({})
for i in K:
    for j in K:
        delta[i,j] = 0 if Hk[i] <= Hk[j] else 1
```

```
# Conjunto de contenedores.
num_cont =     # contenedores que llegan a la bahía
total_cont = cont_0 + num_cont  # total de contenedores en la bahía
J2 = set(range(cont_0+1,total_cont+1))  # contenedores enumerados
J = J1 | J2

# Conjunto de tiempos de llegada y salida de cada contenedor.
TJ = tupledict({})
# Conjunto de tiempos internos (El conjunto de tiempos sin incluir el tiempo de salida de cada contenedor).
TI = tupledict({})
# Conjunto de tiempos internos en notación de conjunto de python.
TIs = tupledict({})
# Cada clave equivale a un contenedor y su valor al conjunto de períodos de tiempo dentro de la bahía de dicho contenedor.
```

```
# Definición del modelo y sus variables de decisión.
m = Model('DCRP')

# Variable de ubicación de contenedores en ranuras en los períodos de tiempo.
x = m.addVars(J, K, T, vtype = GRB.BINARY, name="x")

# Variable de reubicación de los contenedores en ranuras
y = m.addVars(J, TI, vtype = GRB.BINARY, name="y")
```

```
#Definición de la función objetivo
m.setObjective(quicksum(y[j,t]
                        for j in J for t in TI[j]),
               GRB.MINIMIZE)
```

```
# Restricciones 2.2
# Permiten ubicar como máximo un contenedor en una ranura k en un período de tiempo.
m.addConstrs((quicksum(x[j,k,t] for j in J if t in TJ[j]) <= 1 for k in K for t in T), name="(2)")
```

```
# Restricciones 2.3
# Garantizan que cada contenedor j debe estar ubicado en una ranura k mientras permanezca en la bahía.
m.addConstrs((quicksum(x[j,k,t] for k in K) == 1 for j in J for t in TJ[j]), name="(3)")
```

```
# Restricciones 2.4
# Establecen que un contenedor solo puede ubicarse en una ranura k+1 si y solo si la ranura k ya está ocupada.
m.addConstrs ((delta[k+1,k]*(quicksum(x[j,k+1,t] for j in J if t in TJ[j]) - quicksum(x[j,k,t] for j in J if t in TJ[j]))  <= 0
                                for k in K-O for t in T), "(4)")
```

```
#Restricciones 2.5
# Implican que, si el contenedor j está ubicado en una ranura superior del contenedor j', que sale antes que el contenedor j
# en el momento τ_j'^d, entonces el contenedor j se reubica y y_j^t se obliga a tomar el valor 1.

m.addConstrs((delta[k+1,k]*(x[j,k+1,t] + x[jj,k,t])-y[j,t] <= 1
            for k in K-O for j in J for jj in J if j != jj for t in T if t == TJ[jj][-1]), "(5)")
```

```
# Restricciones 2.6
# Obligan a y_j^t a tomar un valor cero cuando no hay ningún contenedor para recoger en una ranura más baja que el contenedor j
# en el período de tiempo t.
m.addConstrs((
    x[j,k,t] - quicksum(quicksum(delta[k,kk]*x[jj,kk,t] for kk in K) for jj in J-{j} if t==TJ[jj][-1]) <= 1 - y[j,t]
    for k in K for j in J for t in TJ[j]), "(6)")
```

```
#Restricciones 2.7
# Aseguran que los contenedores que no se reubican en el período de tiempo t, mantengan su ubicación para el próximo período
# de tiempo t+1.
m.addConstrs((x[j,k,t] - x[j,k,t+1] <= y[j,t]  for k in K for j in J for t in TI[j]), "(7)")
m.addConstrs((x[j,k,t+1] - x[j,k,t] <= y[j,t]  for k in K for j in J for t in TI[j]), "(7.1)")
```

```
# Restricciones 2.8
# Imponen que un contenedor j reubicado en el período de tiempo t no se puede colocar en la misma columna
# para el siguiente período de tiempo t+1.
m.addConstrs((
    x[j,k,t]+x[j,k,t+1]+quicksum(delta[k,kk]*x[j,kk,t+1] for kk in K)+quicksum(delta[kk,k]*x[j,kk,t+1] for kk in K) <= 2-y[j,t]
    for k in K for j in J for t in TI[j]), "(8)")
```

```
# Restrcciones 2.9
# Estas restricciones satisfacen A3: Las reubicaciones se permiten solo en el período de tiempo de salida
# de los contenedores y un contenedor se puede reubicar como máximo una vez en un período de tiempo.
m.addConstrs(( x[j,k,t] + quicksum(delta[kkk,k]*x[jj,kkk,t] for kkk in K) + x[j,kk,t+1]
             + quicksum(delta[kkk,kk]*x[jj,kkk,t+1] for kkk in K) + y[j,t] + y[jj,t] <= 5
             for k in K-O for kk in K-O if Wk[k]!=Wk[kk] for j in J for jj in J if j!=jj for t in TIs[j]&TIs[jj]), "(9)")
```

```
# Restricciones 2.12
# Imponen las asignaciones iniciales de ranuras de contenedores que ya están en la bahía.
m.addConstrs((x[j,k,1] ==0 for j in J1 for k in K1), "(12)")
```

Referencias bibliográficas

[1] M. Hakan Akyüz. *Heuristic Methods for the Unrestricted Dynamic Container Relocation Problem*. International MultiConference of Engineers and Computer Scientists, IMECS, 2017.

[2] M. Hakan Akyüz et al. *A Mathematical Formulation and Efficient Heuristics for the Dynamic Container Relocation Problem*. Naval Research Logistics & Wiley Periodicals, Inc., 2014.

[3] Mokhtar Bazaraa et al. *Programación lineal y flujo en redes*. Volumen 2 & Limusa, 1994.

[4] Zhan Bian et al. *Optimization on the container loading sequence based on hybrid dynamic programming*. Vilnius Gediminas Technical University (VGTU) Press & Taylor and Francis Group, 2015.

[5] Setareh Borjian et al. *Container Relocation Problem: Approximation, Asymptotic, and Incomplete Information*. arXiv preprint arXiv:1505.04229v2, 2015.

[6] Setareh Borjian et al. *Managing Relocation and Delay in Container Terminals with Flexible Service Policies*. arXiv preprint arXiv:1503.01535, 2015.

[7] Marco Caserta et al. *A mathematical formulation and complexity considerations for the blocks relocation problem*. European Journal of Operational Research & Elsevier, 2009.

[8] Laidy De Armas et al. *Revisión bibliográfica sobre funciones objetivo para el apilamiento de contenedores*. Inteligencia Artificial & IBERAMIA, 2017.

[9] Conferencia de las Naciones Unidas sobre Comercio y Desarrollo. *Informe sobre el transporte marítimo 2021*. Naciones Unidas & UNCTAD, 2021.

[10] Roberto Guerra-Olivares et al. *A Heuristic Procedure for the Outbound Container Relocation Problem during Export Loading Operations*. Mathematical Problems in Engineering & Hindawi Publishing Corporation, 2015.

[11] Osman Karpuzoglu et al. *A Tabu Search Based Heuristic Approach for the Dynamic Container Relocation Problem*. Operations Research Proceedings & Springer-Verlag, 2017.

[12] Kap Hwan Kim et al. *Deriving decision rules to locate export containers in container yards*. European Journal of Operational Research & Elsevier, 2000.

[13] Kap Hwan Kim et al. *A heuristic rule for relocating blocks*. Computers and Operations Research & Elsevier, 2006.

[14] Bernhard Korte et al. *Combinatorial Optimization, Theory and Algorithms*. Springer, 2018.

[15] Erhan Kozan et al. *Mathematical modelling of container transfers and storage locations at seaport terminals*. REGULAR ARTICLE & Springer-Verlag, 2006.

[16] Yusin Lee et al. *An optimization model for the container premarshalling problem*. Computers and Operations Research & Elsevier, 2007.

[17] Joseph Y-T. Leung. *Handbook of SCHEDULING: Algorithms, Models, and Performance Analysis*. COMPUTER and INFORMATION SCIENCE SERIES & Chapman and Hall/CRC, 2017.

[18] Diego Recalde et al. *Scheduling the professional Ecuadorian football league by integer programming*. Computers and Operations Research & Elsevier, 2013.

[19] Zacarías Torres. *Administración Estratégica*. Grupo Editorial Patria, 2014.

[20] Paúl Vera Morán. *Análisis de la problemática actual del puerto comercial de Esmeraldas, para obtener conclusiones y proponer posibles soluciones*. Escenarios y oportunidades del comercio exterior & Autoridad Portuaria Esmeraldas, 2020.

[21] Yat-wah Wan et al. *The Assignment of Storage Locations to Containers for a Container Stack*. Naval Research Logistics & Wiley Periodicals, Inc., 2009.

[22] Chuqian Zhang et al. *Storage space allocation in container terminals*. Transportation Research Part B & Elsevier, 2003.