ANÁLISIS DE HERRAMIENTAS INFORMÁTICAS LIBRES Y DE CÓDIGO ABIERTO PARA PROBLEMAS DE OPTIMIZACIÓN DE APILAMIENTO DE CONTENEDORES

Laidy de Armas Jacomino [*1], Danilo Valdes-Ramirez [*], Carlos Morell Pérez [**], Rafael Bello-Pérez [**]
[*]Facultad de Informática y Ciencias Exactas, Universidad de Ciego de Ávila Máximo Gómez Báez, Cuba.
[**]Centro de Investigaciones de Informática, Universidad Central "Marta Abreu" de Las Villas, Cuba.

ABSTRACT
In this study we analyze the performance of three free and open source optimization solvers: lpSolve, GLPK and Cbc. We use 4 datasets with instances corresponding to 4 configurations of a container yard located in Villa Clara, Cuba. It is analyzed if the Storage Space Allocation Problem for import containers in can be solved, in a acceptable computational time for the operations of a real container yard, using an exact optimization method. We base our evaluation on three metrics achieving an optimal solution on import containers: Computational time, number of iterations, and number of nodes. Moreover, we present a feasibility study on the impact of the exact method in the operations of container yard in Villa Clara.

KEYWORDS: Container stacking problem, free optimization solvers, storage space allocation problem.

MSC: 68U99

RESUMEN
En el presente estudio se analiza el desempeño de tres herramientas informáticas de optimización de libre acceso y de código abierto: lpSolve, GLPK y CBC. Se emplean cuatro conjuntos de datos con instancias que representan cuatro configuraciones de un patio de contenedores ubicado en Villa Clara, Cuba. Se estudia si el problema de asignación de espacios de almacenamiento a contenedores de importación puede ser resuelto en un tiempo aceptable para las operaciones de este patio, usando un método matemático exacto. La evaluación de estas herramientas informáticas de optimización se basa en tres métricas: tiempo computacional, número de iteraciones y número de nodos explorados. Además, se realiza un estudio de factibilidad sobre el impacto del método de optimización exacto en las operaciones del patio de contenedores de Villa Clara.

PALABRAS CLAVE: Apilamiento de contenedores, asignación de espacios a contenedores, herramientas de optimización libres.

1. INTRODUCCIÓN

La gestión eficiente de los patios de almacenamiento de contenedores en las terminales de contenedores ha seguido recibiendo gran atención como tema de investigación en el sector académico y empresarial. En los patios de las terminales se almacenan diversos tipos de contenedores según el flujo al que pertenecen: importación, exportación o trasbordo.
Para llevar a cabo las operaciones de apilamiento de contenedores en un patio, se utilizan varios recursos a diario, tales como los vehículos de transportación, las grúas para cargar y descargar los contenedores, y el espacio de almacenamiento [13]. El espacio de almacenamiento permanece sin cambiar incluso cuando el

[1] laidy@unica.cu

flujo de contenedores se incrementa. Para ahorrar el espacio de almacenamiento los contenedores se ubican en pilas de múltiples niveles. Las grúas que apilan los contenedores pueden acceder solamente al contenedor que está en el tope de las pilas. El acceso a contenedores ubicados en posiciones intermedias de las pilas provoca movimientos improductivos (del inglés reshuffles). Los movimientos improductivos son no deseados por los operadores y para los clientes porque provocan el deterioro de las grúas, lentitud en las operaciones de descarga y consumo adicional de combustible. Por estas razones minimizar los movimientos improductivos se ha convertido en el principal objetivo de varias investigaciones académicas cuando se resuelve el problema de optimización de asignación de espacios de almacenamiento a contenedores (del inglés Storage Space Allocation Problem o SSAP)[8, 12, 1, 5, 9].

El SSAP a contenedores consiste en encontrar la mejor asignación de contenedores a espacios de almacenamiento en un patio. Una asignación óptima es aquella en la cual se reduce el tiempo operacional de cargar y descargar los contenedores al mismo tiempo que se minimizan los movimientos improductivos. La estrategia de solución para este problema depende del flujo de contenedores que se trate, tipos de contenedores, equipamiento, informaciones disponibles sobre éstos, limitaciones físicas del patio de contenedores, entre otros aspectos. Por lo que el SSAP a contenedores se considera un problema de investigación abierto.

El presente trabajo tiene como objetivo evaluar el desempeño de tres herramientas informáticas de optimización sobre el SSAP a contenedores de importación en un patio cubano. Se describen y comparan estas herramientas de optimización, las cuales son de libre acceso y de código abierto. Los datos usados en la experimentación pertenecen a un patio de contenedores de Villa Clara. En este estudio también se analizan los métodos de optimización que implementan las herramientas informáticas para resolver problemas de programación lineal (PL) entera-mixta. Además, se presenta un estudio de factibilidad donde se analiza el impacto y el ahorro de recursos en el patio de contenedores cubano al emplear el método de optimización propuesto. Finalmente, se presentan las conclusiones del trabajo.

2. HERRAMIENTAS INFORMÁTICAS DE LIBRE ACCESO Y DE CÓDIGO ABIERTO

Existen varias herramientas informáticas tanto de libre acceso como propietarias que implementan métodos de optimización de PL. Los tipos de problemas de PL que resuelven son diversos. Estas herramientas informáticas de optimización pueden ser invocadas desde un Entorno de Desarrollo Integrado (del inglés IDE), desde bibliotecas o directamente desde un fichero escrito en un lenguaje matemático de optimización.

2.1. GLPK

GLPK[2] (GNU Linear Programming Kit) es un conjunto de rutinas programadas en ANSI C y organizadas en forma de biblioteca de software. Con esta herramienta se pueden resolver problemas de PL, PL entera-mixta, entre otros. GLPK incluye el programa *glpsol*, el cual permite que desde la consola de comandos se pueda invocar un fichero escrito en los formatos MPS, MathProg o CPLEX LP. También puede ser usada como biblioteca de software desde varios lenguajes de programación como C, Java, R, Matlab, Octave y Python. Otra forma de resolver problemas de optimización con esta herramienta es a través del IDE GUSEK (GLPK Under Scite Extended Kit).

GLPK no tiene limitaciones en cuanto al tamaño del modelo matemático, la cantidad de restricciones y variables permitidas es ilimitada. Además, esta herramienta de optimización cuenta con una comunidad de desarrollo que realiza mejoras continuamente y brinda nuevas versiones. Al mismo tiempo, se tiene acceso a diversos modelos matemáticos para una gran variedad de problemas de optimización y la documentación para su uso[3].

GLPK puede ser usado en diferentes plataformas como Windows, Linux y Mac OS X.

Para problemas de PL, esta herramienta implementa el método Símplex revisado primal, dual y el método de Punto Interior, y para la programación con enteros provee una implementación del método Ramas y cortes (del inglés Branch and Cut).

2.2. lpSolve

[2] http://www.gnu.org/software/glpk/

[3] https://spokuta.wordpress.com/the-gnu-programming-kit-glpk

Otra herramienta informática de libre acceso y de código abierto, la cual puede ser usada para resolver problemas de optimización es lpSolve[4]. Esta herramienta de optimización implementa el método Símplex revisado para PL y el método de Ramas y cotas (del inglés Branch and Bound) para PL entera-mixta. Se pueden resolver modelos de optimización de PL, PL entera-mixta y semicontínuos.
lpSolve no tiene límite para el tamaño del modelo matemático en cuanto al número de restricciones y variables. Esta herramienta acepta archivos de entrada en formato lp o MPS. Además permite convertir de un formato de modelo matemático a otro.
Actualmente la herramienta de optimización lpSolve puede ser usada de tres formas diferentes: como biblioteca desde los lenguajes C, VB, .NET, Delphi, Excel, Java, etc ; a partir de un fichero que contenga la implementación del modelo matemático; o se puede usar a traves de lpSolve IDE para la plataforma Windows. Con lpSolve IDE se pueden modificar los parámetros y métodos a usar para resolver el modelo de optimización y visualizar los resultados. lpSolve cuenta una comunidad de desarrollo activa, con amplia documentación y ejemplos de modelos matemáticos, similar a GLPK.

2.3. CBC

CBC (Coin Branch and Cut)[5] es una herramienta de optimización de libre acceso y de código abierto basada en el método Ramas y cortes, para resolver problemas de PL entera-mixta específicamente.
Puede ser usada al igual que GLPK y lpSolve, desde bibliotecas a partir de rutinas implementadas en C++. También puede ser usada desde un ejecutable invocando un fichero escrito en formato MPS.
Esta herramienta informática está disponible para los sistemas operativos Windows y Linux. Al igual que GLPK y lpSolve, no tiene limitaciones para el tamaño del modelo matemático.

3. ANÁLISIS DE LAS HERRAMIENTAS INFORMÁTICAS

3.1. Descripción del problema

En este trabajo se modela el problema de optimización de asignación de espacios de almacenamiento a contenedores SSAP como un problema de asignación generalizado. Se le asigna a cada contenedor que arriba nuevo al patio real una posición específica dentro de un bloque de almacenamiento. Se tiene como entrada una secuencia de contenedores que arriban $(c_1, c_2, \ldots, c_n)$, donde cada contenedor c_n tiene una fecha de arribo y una fecha de salida conocida. El costo de asignar cada contenedor a cada una de las posiciones en el patio también se conoce. La salida del método de optimización son las posiciones para cada uno de los contenedores (s, t, c), donde s representa la pila, t es el nivel de altura en la pila, y c identifica al contenedor en esta pila y nivel específico.
El patio de contenedores para el cual se realiza esta investigación cuenta con datos de sus operaciones de los últimos cuatro años (período $2014 - 2017$). De los contenedores que han sido apilados en estos cuatro años se tiene su fecha de arribo, tamaño, cliente, destino y fecha de salida. A partir de estos datos fueron creados cuatro conjuntos de datos, los cuales son usados para validar la eficiencia de los métodos de optimización. Tomando en cuenta que los métodos de optimización usados brindas una solución exacta en todos los casos que esta exista, no se considera necesario evaluar la precisión de la solución, sino la eficiencia. Los datos se dividieron en cuatro conjuntos de datos para crear cuatro casos de estudio:

- Caso de estudio 1: Contiene instancias que describen las condiciones actuales del patio de almacenamiento, 45 pilas y 3 niveles de altura en cada una. Solamente varía de una a otra instancia el número de contenedores a apilar, la cantidad de pilas en la primera fila y la configuración inicial del patio. La configuración inicial del patio representa las posiciones ocupadas y vacías que existen hasta ese momento.
- Caso de estudio 2: Contiene las instancias que describen configuraciones futuras en las cuales el patio de almacenamiento se expanda, incrementando el número de pilas a 90 y el mismo número de niveles de altura del conjunto de datos 1. El patio de contenedores de Villa Clara se encuentra en un proceso constructivo para la ampliación del mismo.
- Caso de estudio 3: Contiene instancias con la configuración futura en la cual los niveles de altura de las pilas en el patio se incremente a 4 y 5, y el número de pilas se mantiene en 45, similar al

[4] lpsolve.sourceforge.net
[5] http://www.coin-or.org/Cbc

conjunto de datos 1. Los niveles de altura en las pilas dependen de la altura máxima que alcancen las grúas empleadas en el apilamiento y de políticas que autorizan o no estos niveles en las pilas. Estas políticas pueden estar sujetas a cambios.

- Caso de estudio 4: Contiene instancias sobre configuraciones futuras donde las políticas del patio varían incrementándose los niveles de altura en las pilas a 4 y 5, y además se incrementan en 90 la cantidad posible de pilas, similar al conjunto de datos 2.

La tabla 1 presenta una descripción de las instancias por conjuntos de datos. Cada celda contiene el número de contenedores apilados en el patio, el número de pilas en la primera fila, y la cantidad de contenedores a apilar. Si contiene un asterisco adjunto a la cantidad de pilas en la primera fila significa que las posiciones en estas pilas están ocupadas. Si el número de contenedores apilados es cero entonces el patio se encuentra inicialmente vacío (Ejem. Las primeras instancias de todos los conjuntos de datos). También existen instancias como se observa en la tabla que tienen igual número de contenedores apilados, contenedores por apilar y la misma cantidad de pilas en la primera fila. Sin embargo, estas instancias difieren en la configuración inicial del patio (Ejem. las instancias cuatro y cinco del primer conjunto de datos).

Tabla 1: Información general sobre las instancias [ContenedoresApilados#, Pilas#PrimeraFila, Contenedores a apilar#].

Instancia	Caso de estudio 1	Caso de estudio 2	Caso de estudio 3	Caso de estudio 4
1	[0,5,135]	[0,15,270]	[0,10,180]	[0,15,360]
2	[60, 5*, 45]	[0,15,200]	[0,10,225]	[50,15*, 180]
3	[54,5,50]	[50,15,200]	[0,10,100]	[0,15,450]
4	[25,5,100]	[50,15*, 200]	[0,10,200]	[50,15*, 300]
5	[25, 5*, 100]	[100,15*, 170]	[100,10,60]	[72,15,150]
6	[0,5,50]	[100,15*, 150]	[100,10*, 60]	[100,15*, 150]
7	[0,5,100]	[100,15*, 150]	[140,10,50]	[50,15,300]
8	[30,10*, 100]	[100,15*, 150]	[140,10*, 50]	[50,15,200]
9	[30,12*, 100]	[170,15*, 60]	[140,10*, 25]	[100,15*, 240]
10	[51,10*, 75]	[170,15*, 60]	[100,10,35]	[100,15*, 300]

3.2. Implementación de las herramientas informáticas

Las herramientas de optimización analizadas en este trabajo implementan diferentes métodos de optimización tanto para PL como para PL entera-mixta. Sin embargo, en este trabajo se hace énfasis en los métodos de optimización usados para resolver problemas de programación entera-mixta.

Una de las formas en que se pueden resolver problemas de programación entera es mediante el método de Ramas y cotas. Este método se basa en la técnica general de diseño de algoritmos Divide y vencerás. Teniendo en cuenta de que es muy costoso resolver directamente el problema original, entonces se divide en subproblemas cada vez más pequeños hasta que estos se puedan vencer. La división o ramificación se hace particionando el conjunto completo de soluciones factibles en subconjuntos más pequeños. Luego se acota la mejor solución en el subconjunto y se descartan los subconjuntos cuya cota indique que no es posible que contenga una solución óptima para el problema original [4].

Los autores Hillier y Lieberman en [4] describen un método de Ramas y cotas para programación entera binaria. El método cuenta con los pasos básicos de ramificación, acotamiento y sondeo. Puede ser empleado en un problema en particular a partir de que se definan adecuadamente procedimientos para seleccionar la variable de ramificación y el siguiente subproblema a ramificar. Para el acotamiento se resuelve un relajamiento con el método Símplex. El método Símplex empleado por las herramientas GLPK, lpSolve y CBC es la variante para computadoras del Símplex revisado. Este método es mucho más eficiente computacionalmente ya que realiza las operaciones algebraicas elementales y con renglones mediante matrices.

La herramienta lpSolve usa una implementación del método de Ramas y cotas para resolver problemas de PL entera-mixta. Los parámetros del mismo pueden ser ajustados. En este estudio se asumen los parámetros por defecto de esta herramienta.

En lpSolve se decide automáticamente qué rama o subproblema del árbol de búsqueda analizar primero

mediante la opción Branch_Automatic. Para seleccionar la variable a ramificar define la regla integrada por varias funciones: Node_Pseudononintselect, Node_Greedymode, Node_Dynamicmode y Node_Rcostfixing[6].
La convergencia del método de Ramas y cotas puede ser mucho más rápida y eficiente si se integra con técnicas de generación de cortes o cortaduras. Un plano cortante o corte es una nueva restricción funcional que reduce la región factible del relajamiento de PL sin eliminar soluciones factibles del problema de programación entera original [4]. Este procedimiento permite encontrar soluciones enteras a partir de un problema lineal, reduciendo considerablemente el tamaño del árbol de búsqueda. Los planos cortantes pueden ser incluidos en la raíz del árbol o en cada uno de los nodos del árbol, o una combinación de ambos. De forma general los métodos de Ramas y cortes son algoritmos exactos para problemas de programación entera específicamente. Estos métodos combinan los métodos de planos cortantes con el algoritmo de Ramas y cotas [7]. El autor Mitchell en [7] presenta un algoritmo de Ramas y cortes general para resolver problemas de programación entera-mixta.
Las herramientas GLPK y CBC hacen uso de estas técnicas de cortaduras, integrándolas con el método de Ramas y cotas y el preprocesado automático del problema.
La etapa de preprocesado automático del modelo matemático permite detectar reformulaciones que hacen que el problema se solucione con mayor rapidez, sin eliminar soluciones factibles. Mediante el preprocesado automático de un problema se identifican variables que se puedan fijar en uno de sus dos valores (0 o 1), se eliminan restricciones redundantes, y se estrechan algunas restricciones reduciendo la región factible para la soltura de PL, sin eliminar ninguna solución factible para el problema original.
GLPK usa un algoritmo de Ramas y cortes específico, descrito por Makhorin en [6]. En el paso de ramificación del algoritmo el subproblema actual es dividido en dos nuevos subproblemas. De esta forma todos los subproblemas son representados en forma de un árbol de búsqueda.
Los parámetros del método Ramas y cortes implementado en GLPK pueden ser ajustados, aunque en este trabajo se emplean los parámetros por defecto. GLPK incluye cuatro procedimientos de generación de cortaduras: Gomory's mixed integer cut, Mixed integer rounding cut, Mixed cover cut, y Clique cuts, los cuales son usados por defecto en esta herramienta. Para resolver las relajaciones de PL emplea el método Símplex revisado primal.
Para seleccionar la variable a ramificar y los subproblemas o nodos a explorar en GLPK se emplea la heurística de Driebeck y Tomlin en [3]. Según los autores Driebeck y Tomlin, el cálculo de esta heurística es muy costosa computacionalmente.
CBC emplea también el método Ramas y cortes, el cual se describe en su sitio web oficial[7]. Esta herramienta permite seleccionar entre dos variantes del algoritmo de Ramas y cortes, en dependencia del lugar en donde se defina que se incluirá el uso de los generadores de cortaduras dentro del algoritmo de Ramas y cotas. Si se usan los generadores de cortaduras solamente en el paso 1 del algoritmo de Ramas y cotas entonces el método se nombra Cortes y ramas (del inglés Cut and branch). De lo contrario, si se realizan los cortes para ajustar la relajación de los problemas de PL se nombra Ramas y cortes.
Para generar cortaduras, CBC implementa varios procedimientos, algunos de ellos: Gomory's mixed integer cut, Mixed integer rounding cut, Mixed cover cut, Clique cut, Knapsack Cover Cuts, Lift and project cuts, Two Phase Mixed Integer Rounding Cuts, etc [8]. En este trabajo se emplean todos menos los siguientes tres: Lift and project cuts, Reduced and split cuts y Residual capacity cuts.
En CBC también se puede configurar dónde incluir los generadores de cortaduras, en el nodo raíz antes de los pasos de Ramas y cotas, en cada uno de los nodos del árbol, o mediante la opción *ifmove.* La opción *ifmove* permite generar los planos de corte a medida que éstos tienen un buen desempeño y están permitiendo acercarse al valor objetivo. Esta opción funciona como una heurística que permite mejorar la eficiencia del método. En este trabajo se emplea la opción *ifmove*, menos en Two Phase Mixed Integer Rounding Cuts que por defecto se aplican los cortes solo en el nodo raíz.
Además, CBC permite ajustar dos parámetros que pueden determinar la eficiencia del algoritmo: *cut passes root* y *cut passes tree*. Con estas opciones se puede establecer un número fijo de veces que los generadores de planos cortantes se ejecutan en un nodo. Por ejemplo, varias veces en el nodo raíz pero pocas veces en los otros nodos del árbol. El parámetro *cut passes root* determina el número de veces que los generadores de cortaduras son aplicados en el nodo raíz, mientras que el parámetro *cut passes tree* determina el número de veces en los restantes nodos. Los valores usados por defecto del parámetro *cut passes root* es 100 si el

[6] http://lpsolve.sourceforge.net/5.5/LPBasics.htm
[7] http://www.coin-or.org/Cbc
[8] https://www.coin-or.org/Clp/userguide

modelo tiene menos de 500 variables, 100 veces igualmente si tiene menos de 5000 variables pero se detiene si el valor de la función objetivo se hace más pequeño, en otros casos se ejecuta 20 veces. Para el parámetro *cut passes tree* el valor por defecto es 1.
Para seleccionar la variable a ramificar en CBC, se establece una estrategia de costo mediante el parámetro *coststrategy*. Esta estrategia consiste en que las variables son ordenadas respecto a su costo absoluto, y la ramificación se realiza en primer lugar sobre las variables con mayor costo. Esta estrategia en forma de heurística resulta ser muy efectiva para esta herramienta específicamente y es la que se establece por defecto. El cálculo del costo de una variable está relacionado con la cantidad de veces que esta variable ha sido ramificada.
Por otro lado, la selección del próximo nodo o subproblema del árbol a explorar se ajusta mediante el parámetro *nodestrategy*. La estrategia que se emplea en este trabajo y que establece la herramienta por defecto es *fewest*. Esta estrategia consiste en seleccionar el nodo con menor cantidad de soluciones de PL infactibles.

3.3. Resultados y discusión

El SSAP a contenedores se demuestra que es NP-Completo en [2] para las condiciones actuales del patio que se describen en la primera instancia del primer conjunto de datos. Los autores demuestran la complejidad computacional de este problema para estas condiciones específicas a partir del teorema 4 que plantean en su trabajo. A continuación se presenta:
El problema factible $L, b = 3|I^{in}, I^{fix} = \emptyset, s_{ij}|$ *– es NP-Completo incluso para un área de almacenamiento vacía (*$I^{fix} = \emptyset$*) y restricciones de apilamiento transitivas* s_{ij}.
Donde:
L, Significa que es un problema de carga de contenedores.
$b = 3$ Cantidad de contenedores que pueden ser apilados en cada pila, es decir, hasta 3 contenedores cada una en este caso.
I^{in} Conjunto de todos los contenedores que arriban nuevos y deben ser apilados.
$I^{fix} = \emptyset$ Conjunto de contenedores que ya están apilados en el patio, en este caso el patio está vacío inicialmente.
s_{ij} Representa la existencia de restricciones de apilamiento transitivas. Cuando un contenedor i es apilado encima de un contenedor j, entonces el contenedor i también está apilado encima del contenedor h. El modelo matemático que se resuelve en este trabajo cuenta con dos tipos de restricciones transitivas:

1. Todos los contenedores deben ser asignados a una pila del mismo tamaño. Hay pilas en las que se apilan contenedores de 20 pies y otras en las que solo se ponen contenedores de 40 pies. Si el contenedor i tiene 20 pies, entonces el contenedor j y también el contenedor h tienen 20 pies porque los tres están en la misma pila.
2. Los contenedores deben ser apilados ordenados por su fecha de salida del patio de forma descendente. La fecha de salida del contenedor d_i tiene que ser menor o igual que la del contenedor d_j y que la fecha de salida del contenedor d_h.

Como se deduce en el teorema anterior el problema de apilamiento de contenedores que se resuelve es complejo computacionalmente, por lo que es necesario medir la eficiencia de los métodos de optimización para las diferentes instancias de los cuatro conjuntos de datos. La eficiencia computacional de las herramientas informáticas de libre acceso y de código abierto se analiza atendiendo a las siguientes tres medidas:

1. Tiempo que invierten en encontrar la solución óptima. El tiempo límite son 3600 segundos, pues más de 1 hora no se considera aceptable para las operaciones del patio de contenedores real. Las herramientas que llegaron hasta este tiempo límite en alguna instancia específica.
2. Cantidad de iteraciones que realizó el algoritmo para encontrar la solución óptima. El número de iteraciones está estrechamente relacionado con el tiempo computacional. Por ello se desea que este valor sea pequeño.
3. Cantidad de nodos analizados del árbol de búsqueda. Esta medida indica el número de nodos que el algoritmo ha tenido que analizar, para lo cual es necesario extraerlos de la estructura y comprobar si han de ser podados y, si no, expandirlos. Atendiendo a esto, se desea que este valor sea pequeño.

Los experimentos computacionales fueron realizados sobre una computadora personal Intel(R) Core(TM) i7-3520M, CPU 2.90 GHz con 8GB de RAM. Las herramientas informáticas de libre acceso y de código abierto sobre las que se corrieron todas las instancias del patio real fueron las siguientes:

- lpSolve IDE versión 5.5.2.5
- GUSEK versión 0.2.23 actualizado para GLPK 4.64
- CBC 2.7.6

El gráfico de la figura 1 se muestran los tiempos de ejecución de lpSolve, GLPK y CBC para las 40 instancias. Las instancias $1-10$ pertenecen al caso de estudio 1, las instancias $11-20$ al caso de estudio 2, las instancias $21-30$ al caso de estudio 3 y las restantes al caso de estudio 4.
La herramienta de optimización lpSolve excedió el tiempo computacional aceptable en 19 del total de instancias. Para el caso de estudio 1, el cual representa las dimensiones actuales del patio, excedió el tiempo límite en tres instancias. Además, si se observan los resultados de las iteraciones realizadas por esta herramienta en el gráfico de la figura 1 se puede apreciar que realizó muchas más iteraciones que las herramientas de optimización GLPK y CBC en todas las instancias de este caso de estudio. De forma similar se comportó con el resto de los conjuntos de datos que representan condiciones de crecimiento futuro del patio.
A partir de estos resultados se determina que la herramienta informática lpSolve, así como el método de optimización que implementa Ramas y cotas no son apropiados para resolver las condiciones actuales ni futuras del patio de contenedores real. El método Ramas y cotas, no integra el uso de planos cortantes, los cuales se ha demostrado que son mucho más eficientes.

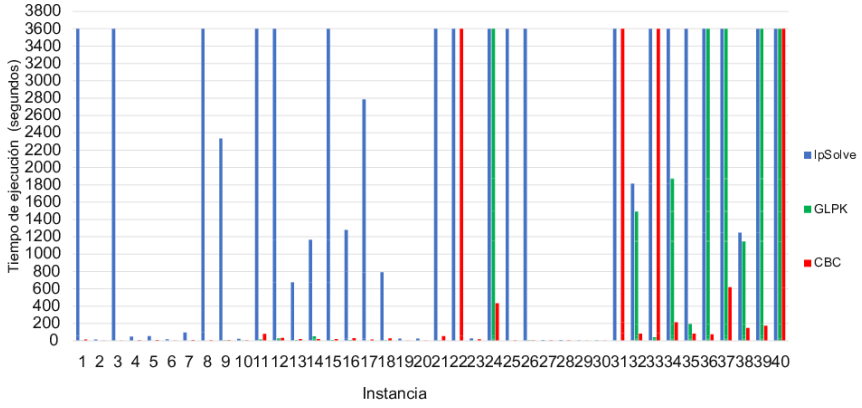

Figura 1: Resultados del tiempo computacional que invierte lpSolve, GLPK y CBC en las 40 instancias. Las instancias 1-10 pertenecen al caso de estudio 1, las instancias $11-20$ al caso de estudio 2, las instancias $21-30$ al caso de estudio 3 y las restantes al caso de estudio 4.

A diferencia de lpSolve, las herramientas de optimización GLPK y CBC que implementan el método Ramas y cortes, no excedieron el tiempo límite establecido para las condiciones reales del caso de estudio 1, ni las condiciones de crecimiento futuras del caso de estudio 2. Para las instancias que describen condiciones futuras de crecimiento del caso de estudio 3 tanto GLPK como CBC excedieron el tiempo permitido. Sin embargo, se consideran aceptables para estas condiciones futuras del patio, porque las instancias en las que excedieron el tiempo permitido no representan situaciones reales que ocurren en el patio. Tanto en la instancia 2 como la 4 del caso de estudio 3 el patio se encuentra inicialmente vacío, y la cantidad de contenedores a apilar es igual al número de espacios de almacenamiento vacíos.
Para el último caso de estudio, el cual representa condiciones futuras de crecimiento del patio tanto GLPK como CBC excedieron el tiempo límite. La herramienta de optimización GLPK excedió el tiempo límite en 4 de las 10 instancias de este caso de estudio, mientras que la herramienta CBC excedió el tiempo en 3 de las instancias
CBC realizó menos iteraciones que GLPK para apilar los contenedores. Solo para 14 instancias necesitó realizar una cantidad mayor de iteraciones que GLPK. En el gráfico de la figura 2 se observan la cantidad de iteraciones que necesitó cada herramienta para resolver las instancias de cada conjuntos de datos.
La cantidad de nodos analizados por CBC y por GLPK para cada instancia y conjunto de datos se presenta en la figura 3. Para resolver las instancias del primer caso de estudio CBC necesitó analizar menos nodos que GLPK para 6 de las 10 instancias. Sin embargo, para las instancias del segundo caso de estudio al crecer el espacio de soluciones CBC necesitó analizar muchos más nodos que GLPK.

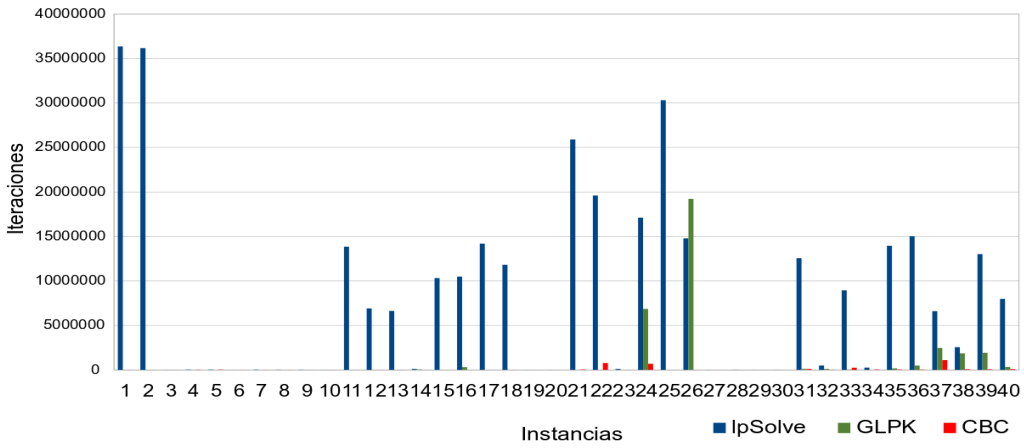

Figura 2: Iteraciones realizadas por lpSolve, GLPK y CBC.

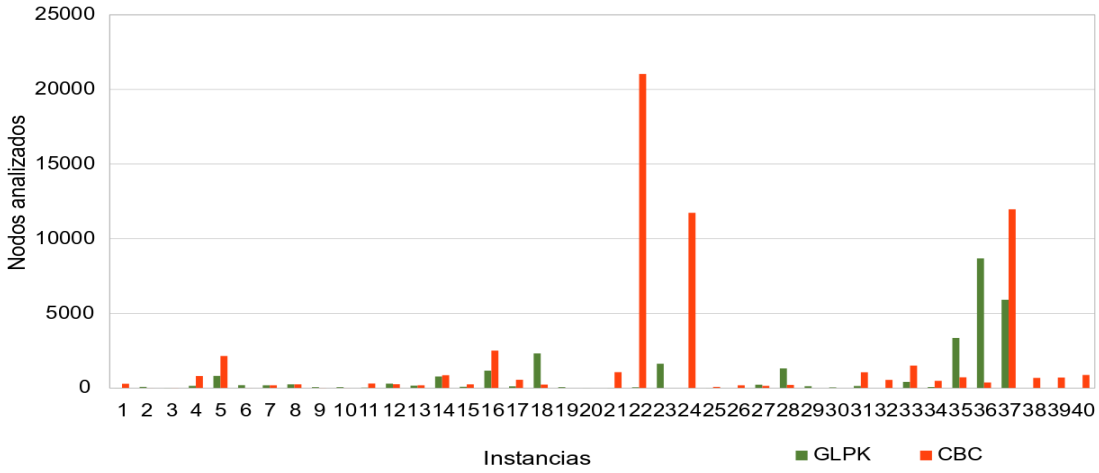

Figura 3: Nodos analizados por GLPK y CBC para encontrar el óptimo.

A partir de los resultados obtenidos se llega a la conclusión que para resolver el SSAP a contenedores de importación en las condiciones actuales, así como las futuras representadas en los casos de estudio 2 y 3 del patio de Villa Clara, tanto GLPK como CBC pueden ser usadas. Estas herramientas de optimización brindan planes de apilamiento de los contenedores óptimos de acuerdo al número de movimientos improductivos y sin exceder el tiempo límite permitido. Sin embargo, para las condiciones futuras de crecimientos analizadas en el último caso de estudio, es conveniente analizar un método aproximado que permita obtener soluciones sin exceder el tiempo permitido.

4 EVALUACIÓN DE LA FACTIBILIDAD DEL MÉTODO EXACTO PROPUESTO

El método Ramas y cortes implementado en GLPK y CBC, permite al operador logístico gestionar un plan de apilamiento de contenedores óptimo y en el tiempo límite, para las condiciones actuales del patio. El apilamiento se considera óptimo porque se minimizan los movimientos improductivos de los contenedores. Minimizar los movimientos improductivos no solo permite retardar el deterioro de la grúa y agilizar las operaciones en el patio, sino que garantiza que el consumo adicional de combustible de la grúa sea el mínimo. El ahorro de combustible se traduce en ahorro de dinero.

En el patio de contenedores de Villa Clara se emplea una grúa tipo Reachstacker para apilar, descargar y transportar los contenedores dentro de la zona de almacenamiento. El Reachstacker es usado generalmente en terminales de contenedores pequeñas. Aunque el Reachstacker ofrece ventajas para patios de contenedores como el de Villa Clara, se encuentra entre los equipamientos con más alto consumo de diesel [11].
El diesel es aun el combustible más usado en los equipamiento de las terminales en América Latina y el Caribe como se observa en el gráfico de la figura 4. En un estudio realizado por los autores Wilmsmeier y Spengler [11] hasta 2015, se presenta la cantidad de litros de diesel que consume el Reachstacker por cada contenedor que manipula. Hasta 2015 el consumo promedio de diesel por contenedor manipulado del Reachstacker es de 4.7 litros (ver figura 5).

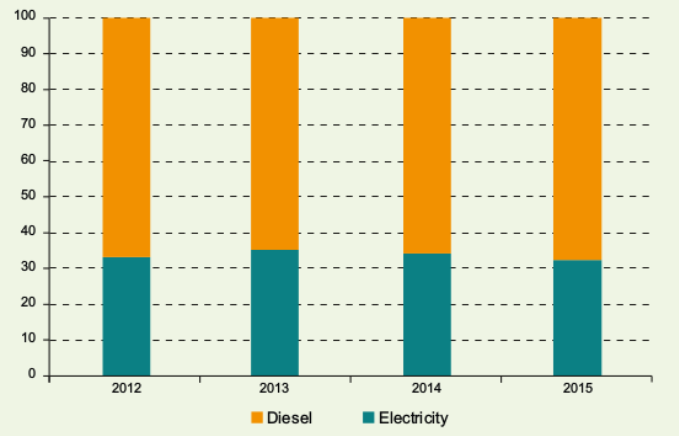

Figura 4: Tendencias en el uso de fuentes energéticas en la terminales de contenedores de América Latina y el Caribe (Fuente [11]).

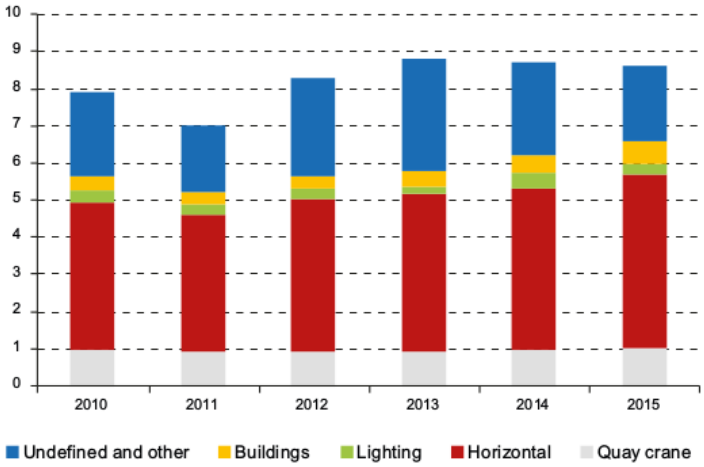

Figura 5: Litros aproximados de diesel consumidos por grupos de actividad en los años 2012 – 2015 (Fuente [11]). El Reachstacker se encuentra en el grupo de equipamientos de tipo Horizontal.
En la gráfica de la figura 6 se presenta el consumo de litros de diesel por movimientos improductivos que realiza el Reachstacker para cada una de las 40 instancias empleando el método de optimización propuesto. En 19 de las 40 instancias no se consume combustible diesel por movimiento improductivo, ya que el método encontró una asignación de contenedores a posiciones que no genera movimientos improductivos.

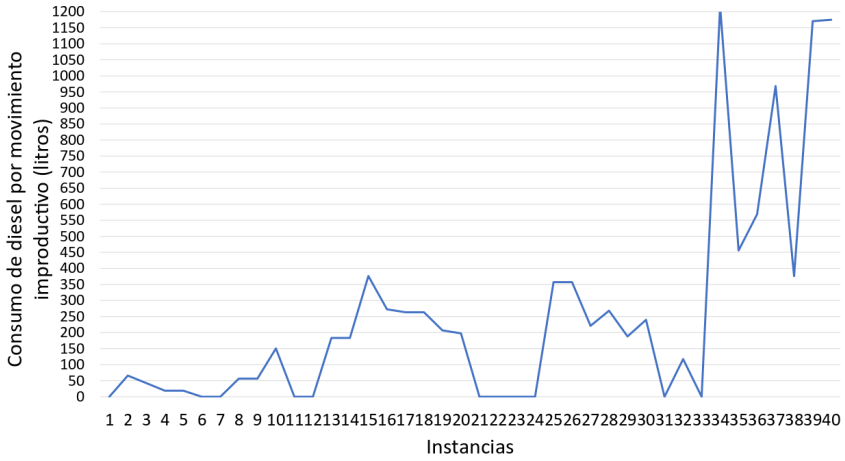

Figura 6: Consumo de diesel por movimiento improductivo del Reachstacker empleando el método de optimización.

En la actualidad los operadores logísticos del patio de contenedores en Villa Clara crean el plan de apilamiento para los contenedores que arriban basándose en su experiencia y en información parcial de los contenedores. Los operadores logísticos registran varias informaciones de los contenedores como tamaño, cliente, destino, fecha de arribo al país, fecha de arribo al patio, etc. Pero no registran la información histórica sobre las posiciones que le han asignado a los contenedores en el patio. Por ello, en este trabajo se emplea el proceso que los operadores siguen para apilar los contenedores que arriban (ver figura 7) para apilar los contenedores de la instancia 3 del caso de estudio 1 y se compara con el resultado del método de optimización para esta instancia. El caso de estudio 1 representa las condiciones actuales del patio de contenedores y la instancia 3 cuenta con 50 contenedores a apilar.

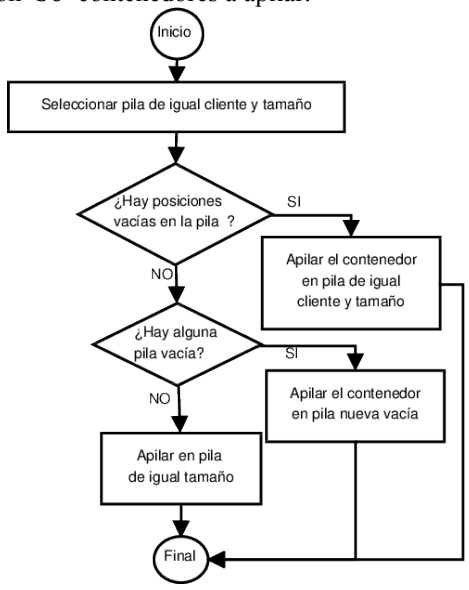

Figura 7: Proceso que siguen los operadores para apilar los contenedores en el patio de Villa Clara.

En el gráfico de la figura 8 se presenta el consumo de diesel para cada uno de los 50 contenedores que se apilaron en el patio con el método de optimización y siguiendo el proceso del operador. Para apilar el primer contenedor tanto con el método como siguiendo el proceso del operador se consumen 4.7 litros, porque la grúa solo realiza un movimiento improductivo. Mientras que para apilar el contenedor 11 con el método la grúa consume 4.7 igualmente, pero siguiendo el proceso consume el doble de litros de diesel.

El método de optimización brinda un plan de apilamiento para los 50 contenedores donde la grúa consume en total 42 litros de diesel al realizar 9 movimientos improductivos. Mientras que con el proceso que sigue

el operador para apilar estos mismos 50 contenedores la grúa debe realizar en total 35 movimientos improductivos y consumir 164.5 litros de diesel.

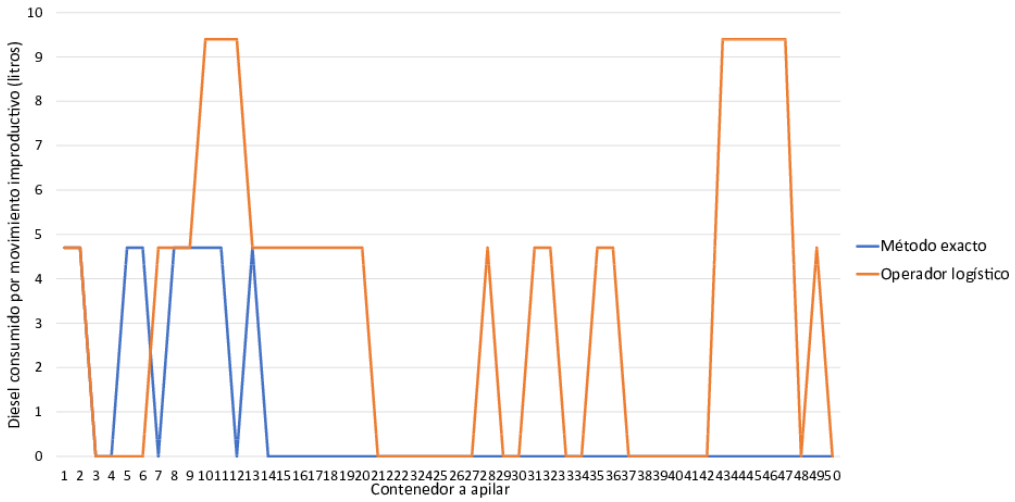

Figura 8: Consumo de diesel por movimiento improductivo al apilar 50 contenedores con el método exacto y siguiendo el proceso del operador logístico del patio de Villa Clara.

El gasto innecesario de diesel en la manipulación de contenedores se traduce para la empresa y el país en dinero. Hasta el 2 de abril del presente año el precio del diesel en el Cuba es 1.06 USD[9]. En el patio de contenedores se recibe un tren con aproximadamente 25 contenedores como mínimo cada semana. Si en aproximadamente 50 semanas del año un tren con 25 contenedores y en la asignación de los mismos se realizan solo 10 movimientos improductivos el gasto anual podría ser de 2,350 litros de diesel y por consiguiente 2,491 USD. En realidad en el patio de contenedores pueden existir más de 10 movimientos improductivos y la cantidad de contenedores que arriban cada semana también ha sido superior. Esto indica que el consumo adicional de diesel y el gasto monetario puede ser aun mayor que la estimación presentada. Por lo que el empleo del método de optimización propuesto permite disminuir el consumo adicional de diesel y de dinero. Además la propuesta puede ser extendida a otros patios de contenedores del país.

Además de los beneficios tangibles por concepto de ahorro de diesel y dinero, a partir de minimizar los movimientos improductivos de la grúa en el patio de Villa Clara, se derivan otros beneficios como son:

- El patio de contenedores tendrá soberanía tecnológica pues la propuesta se basa en el uso de tecnologías libres y de código abierto. Por lo tanto no tendría ningún riesgo legal por emplear tecnologías propietarias y la posibilidad de realizar mejoras al método. La seguridad de la información y el empleo de tecnologías de software libre sigue siendo en Cuba una política a seguir.
- Entre los desafíos que hoy enfrentan los puertos y terminales de contenedores se encuentra la disminución de emisiones de gases de efecto invernadero a la atmósfera procedentes de motores de diesel [10]. Por ello se apuesta por el uso de fuentes de energía renovables. Sin embargo aun el diesel como se mostró en la gráfica de la figura 3 sigue siendo hoy el principal combustible en países de América Latina y el Caribe, específicamente en Cuba. Por ello lo que se busca es disminuir el uso innecesario de las grúas.

5. CONCLUSIONES

En este trabajo se analizaron tres herramientas informáticas de libre acceso y de código abierto: lpSolve, GLPK y CBC. El desempeño computacional de las mismas se evaluó atendiendo al tiempo computacional, la cantidad de iteraciones y cantidad nodos que invierten para dar una solución óptima al problema de asignación de espacios a contenedores en un patio real en Cuba. Además, se presentaron los métodos de PL entera-mixta que usan, así como los parámetros que se ajustan en estas herramientas.

[9] https://es.globalpetrolprices.com/Cuba/diesel_prices/

Para resolver el SSAP a contenedores de importación en las condiciones actuales, así como las futuras representadas en los casos de estudio 2 y 3 del patio de Villa Clara, tanto GLPK como CBC pueden ser usadas. Tanto GLPK como CBC brindan planes de apilamiento de los contenedores óptimos y sin exceder el tiempo límite. Sin embargo, para las condiciones futuras de crecimientos analizadas en el caso de estudio 4, sería conveniente analizar un método aproximado que permita obtener soluciones sin exceder el tiempo permitido, como trabajo futuro de investigación.
Por último, en este trabajo se evaluó la factibilidad de emplear el método exacto Ramas y cortes (implementado en GLPK y CBC) en las operaciones del patio de contenedores de Villa Clara. Se comparó el método exacto con el proceso que normalmente sigue el operador logístico para apilar los contenedores que arriban. A partir del análisis realizado se demostró que emplear el plan de apilamiento que propone el método exacto permite ahorrar diesel y dinero, al minimizar los movimientos improductivos que realiza el Reachstacker.

AGRADECIMIENTOS: Los autores agradecen por su valiosa ayuda en el desarrollo de esta investigación al Centro de Carga y Descarga de Contenedores de Almacenes Universales SA. en Villa Clara, Cuba.

RECEIVED: APRIL, 2018
REVISED: DECEMBER, 2019

REFERENCIAS

[1] BOYSEN, N. and EMDE, S. (2016): The parallel stack loading problem to minimize blockages. **European Journal of Operational Research**, 249, 618–627.
[2] BRUNS, F., KNUST, S. and SHAKHLEVICH, N. V. (2015): Complexity results for storage loading problems with stacking constraints. **European Journal of Operational Research**, 000, 1-8.
[3] DRIEBEEK, N. J: (1966): An Algorithm for the Solution of Mixed Integer Programming Problems. **Management Science**, 7, 576-587.
[4] HILLIER, F. S. and LIEBERMAN, G. J. (2010***):*** **Introduction to operations research.** ninth edition,. The McGraw-Hill Companies, Inc, N. York.
[5] KIM, K. H. and KIM, H. B. (1999): Segregating space allocation models for container inventories in port container terminals. **Int. J. Production Economics**, 59, 415–423.
[6] MAKHORIN, A: (2017): **GNU Linear Programming Kit**. Free Software Foundation, For GLPK Version 4.64 edition. Boston, MA.
[7] MITCHELL, J. E. (2000): **Handbook of Applied Optimization**, chapter Branch-and-Cut Algorithms for Combinatorial Optimization Problems, pages 19. Oxford University Press, Oxford.
[8] PARK, T. , CHOE, RI. , YOUNG H., K. and KWANG, R. R. (2011): Dynamic adjustment of container stacking policy in an automated container terminal. **International Journal of Production Economics,** 133,
[9] SAURI, S. and MARTIN, E. (2011): Space allocating strategies for improving import yard performance at marine terminals. **Transportation Research Part E: Logistics and Transportation Review**, 47, 1038–1057.
[10] UNCTAD (2014): Review of Maritime Transport 2014. **Technical report, UNCTAD/RMT/2014, UNCTAD**, Nueva York y Ginebra,.
[11] WILMSMEIER, G. and SPENGLER, T. (2016): Energy consumption and container terminal efficiency. **Bulletin FAL**, 350, 1-10.
[12] YANG, X., ZHAO, N., BIAN, Z., CHAI, J. and CHAO, M. (2015): An Intelligent Storage Determining Method for Inbound Containers in Container Terminals. **Journal of Coastal Research**, 73, 197–204.
[13] ZHEN, L., JIANG, X., HAY L. and CHEW, E. P. (2013): A Review on Yard Management in Container Terminals. **Industrial Engineering & Management Systems**, 12, 289–305.