TRABAJO FINAL

Localización de contenedores dentro de un bloque

Autores:

Javier Porcel Marí
Ernesto Arce Romero
José Nicolas Granero Moreno
Matías Risso

Índice

1. Introducción

Dado el volumen de transporte marítimo existente, es crucial para los puertos gestionar apropiadamente la carga y descarga de los contenedores, teniendo en cuenta el tiempo empleado en almacenarlos y las ubicaciones estratégicas dentro del conjunto de bloques de contenedores de la terminal portuaria. Para ilustrar esta pequeña introducción se muestra en la siguiente fotografía aérea de una terminal portuaria a un conjunto de grúas que se encuentran descargando los contenedores de una embarcación naval y que serán almacenados en los bloques de contenedores que se muestran justo detrás de las grúas.

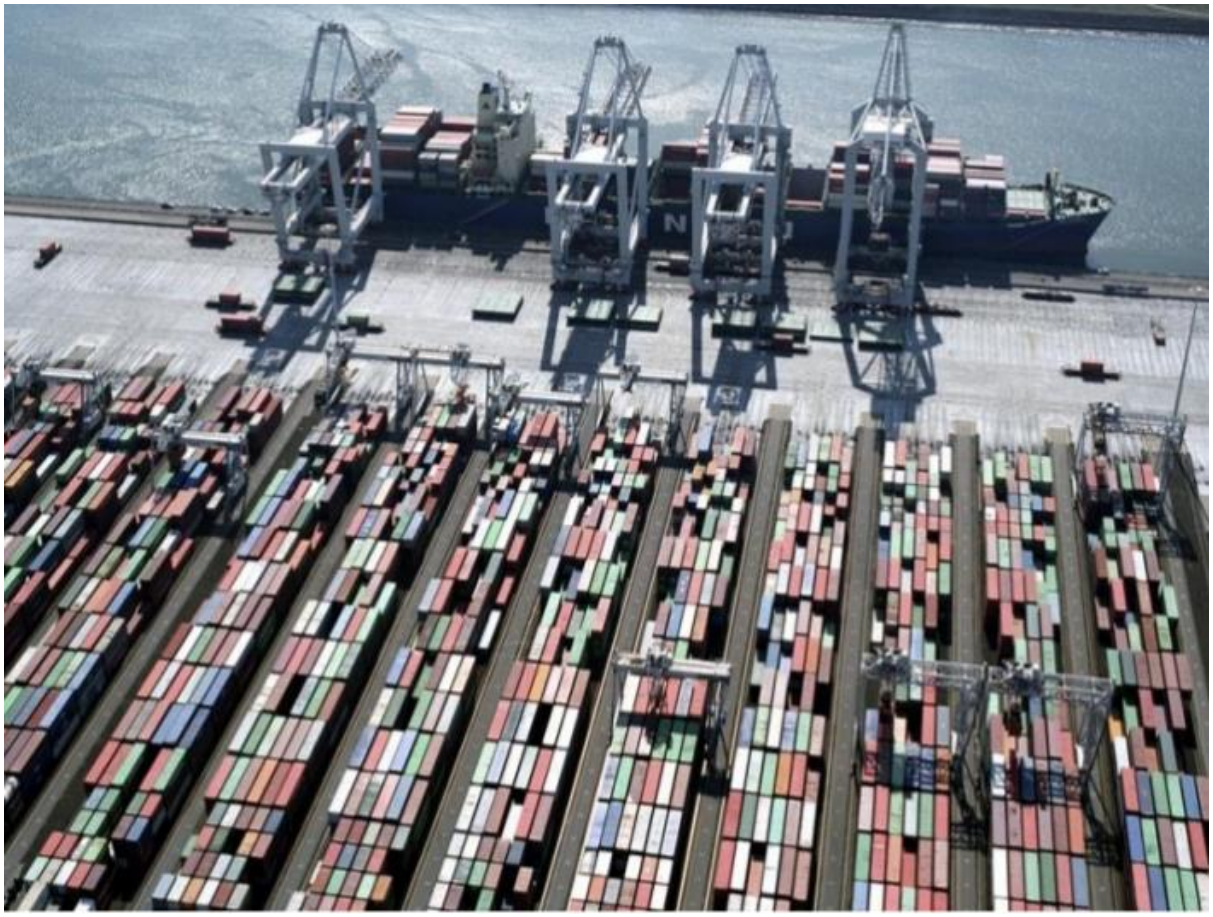


Figura 1. Fotografía aérea de una terminal portuaria.

1.1. El Problema

Se requiere colocar contenedores que llegan al puerto, tanto por mar como por tierra en localizaciones que estén disponibles en los diferentes bloques, tratando de obtener una combinación que tenga una reducida utilización de movimientos (y tiempo asociado) en comparación a las otras soluciones posibles.

Un proceso de disposición de contenedores que no esté regulado podría provocar ineficiencias y gastos excesivos en el desplazamiento de la grúa, además de cuellos de botella.

Concretamente el problema se ilustra mediante el siguiente esquema:

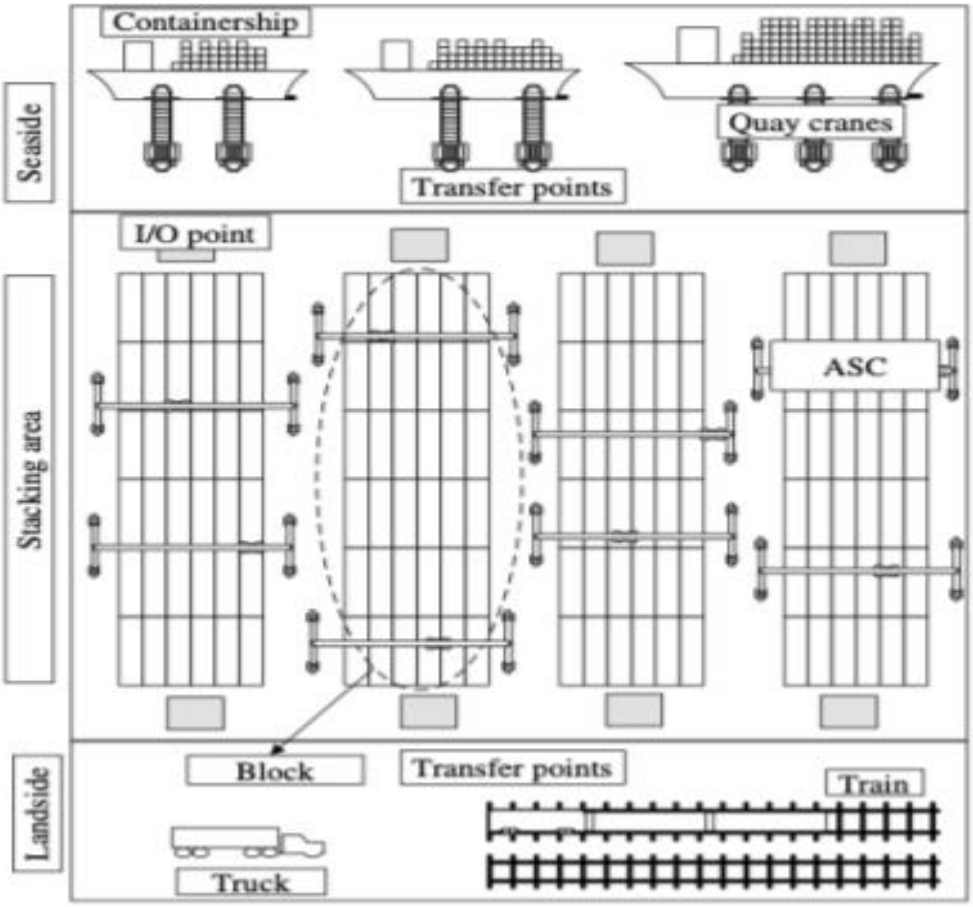


Figura 2. Esquema ilustrativo del funcionamiento de la terminal portuaria.

En el esquema se observa en la parte superior a los barcos que se encuentran en el mar transfiriendo y cargando los contenedores en el I/O point. Posteriormente los contenedores descargados se colocan en los bloques que se encuentran en el área de apilamiento o “Stacking área”.

1.2. Datos iniciales

Se dispone de instancias en las que la información proporcionada consiste en los siguientes parámetros:

P_m: Ubicación de recogida para los contenedores de mar.

P_t: Ubicación de recogida para los contenedores de tierra.

T_i: Tiempo de llegada del contenedor i.

O_i: Origen de llegada del contenedor i (valdrá 1 para tierra y 2 para mar).

L_j: Ubicación del lugar libre j.

Se trabajará con dos instancias de 8 contenedores y 16 ubicaciones posibles, y una instancia de 40 contenedores y 120 ubicaciones disponibles.

1.3. Función Objetivo

Se necesita minimizar el tiempo destinado al movimiento más la espera correspondiente para cada contenedor.

$$Min\ Z = \sum_i M_i + \sum_j E_j$$

donde M_i representa la cantidad de movimientos de la grúa y E_j corresponde a la cantidad de tiempo de espera de la grúa.

1.4. Motivación para la utilización de Heurísticas

Este tipo de problemas es NP-Hard porque lo que se busca es obtener dos vectores solución, uno con el orden en el que son recogidos los contenedores, y otro con la ubicación de cada contenedor dentro de las ubicaciones disponibles. Es posible notar que al aumentar el número de contendores y ubicaciones la cantidad de combinaciones no aumenta en forma lineal, al punto que tratar este tipo de problemas mediante las técnicas de optimización tradicionales es una medida muy demandante de tiempo y recursos computacionales, recurriendo entonces al desarrollo de heurísticas que permitan la obtención de una solución buena en un tiempo de procesamiento adecuado.

Debido a que este problema es un problema difícil de optimización combinatoria, una vez obtenida la heurística se procede con una metaheurística para mejorar la solución mediante un algoritmo genético y una búsqueda local.

2. Soluciones Iniciales

Para inicializar la población del algoritmo genético se introducirán soluciones no-aleatorias que han sido obtenidas mediante dos procesos heurísticos distintos y, de este modo, se consigue que la población inicial tenga soluciones con cierta calidad, las cuales se cruzarán con soluciones aleatorias para explorar así el espacio de soluciones.

La primera de estas soluciones heurísticas, y la más simple, se alcanza tomando las siguientes directrices:

- El orden de recogida de los contenedores coincide con el orden de llegada a la terminal portuaria.
- La localización en la que se almacena cada contenedor corresponde a la localización libre más cercana al punto de recogida.

La segunda de las soluciones se obtendrá de la siguiente manera:

- El orden de cogida de los contenedores se decidirá dependiendo si el contenedor ha llegado o no, calculándose todos los contenedores que no han sido colocados. Si los movimientos que se han hecho más el tiempo esperado más los movimientos que se tienen que hacer para llegar al contenedor son menores que el tiempo de llegada del contenedor, se calculará la diferencia de tiempo de la llegada del contenedor y los movimientos que se han hecho contabilizándose también la espera. En caso contrario, se calcularán los movimientos necesarios para llegar al contenedor. Finalmente, se escogerá el que minimice esa cantidad.
- La localización en la que se coloque cada contenedor será la localización más cercana al punto de recogida que esté libre.

3. Algoritmo Genético

3.1. Selección de Soluciones

El algoritmo genético trabajará con un tiempo definido. Siendo éste, 3 segundos por cada contenedor a colocar, es decir, en las instancias con 8 contenedores el algoritmo tendrá 24 segundos para realizar los debidos cruces y mutaciones y en las instancias con 40 contenedores dispondrá de 120 segundos. Se dividirá en dos grupos las soluciones, dos con las dos soluciones de heurísticas ya resueltas previamente y otras 98 soluciones generadas al azar (estas posteriormente se irán sustituyendo por soluciones con una mejor función objetivo).

Se selecciona al azar una solución del subconjunto de soluciones no-aleatorias y otra solución al azar del subconjunto de soluciones aleatorias para realizar el cruce. Es decir, se elige una de entre las dos soluciones "buenas" y otra de entre las 98 restantes.

Una vez el algoritmo genético alcanza 100 iteraciones sin realizar ningún cambio en la población, es decir, sin que entre ninguna solución nueva en el grupo de las soluciones buenas (las dos que tienen una mejor función objetivo) ni que entre en el grupo de las otras 98 soluciones (las que eran previamente aleatorias), se generarán un nuevo grupo de soluciones aleatorias de tamaño 98 dejando el grupo de las dos mejores soluciones intacto.

3.2. Cruce

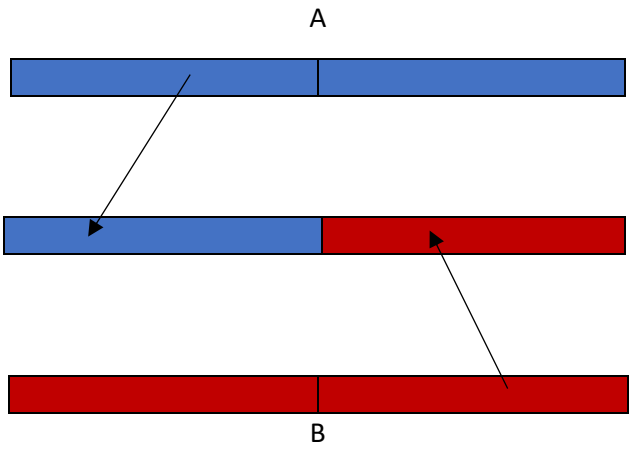
Se selecciona el punto de cruce en el vector de orden de recogida y se replica lo mismo en el vector de localización asignada. El punto de corte se genera de forma aleatoria entre los cuartiles centrales del vector, es decir, entre el primer cuartil y el tercer cuartil del vector. Por ejemplo, en un vector de longitud 8, en corte estará entre 3 y 6. De esta forma se asegura que uno de los progenitores no domine tanto al descendiente generado.

1	2	3	4	5	6	7	8

Posibles puntos de corte

Para obtener las nuevas soluciones se junta la primera parte de un progenitor y la segunda parte del otro progenitor para un descendiente y para el otro descendiente de manera inversa.

Por ejemplo, si se corta por la mitad se obtendrían los siguientes descendientes de los padres A y B:



3.3. Mutación

Cada descendiente tiene una probabilidad del 50% de ser mutado. De ser así, se realiza solo una mutación de entre dos tipos:

- Rotación del orden de recogida de dos contenedores. Ambos seleccionados de forma aleatoria.
- Cambio del lugar libre a ubicar un contenedor. En caso de estar ocupado por otro contenedor, se realiza una rotación. El nuevo lugar es seleccionado aleatoriamente.

Ambos tipos de mutación tienen la misma probabilidad de ocurrir.

3.4. Aceptación de Descendientes

Todas las soluciones almacenadas están ordenadas de mejor a peor. Luego de cada cruce se compara la función objetivo de los descendientes con la mejor obtenida, antes de mutarla y perder una mejor solución de ser el caso. Si es superior que todas las soluciones, se incluye en el modelo y se quita la peor, al final de la lista.

Luego, para todos los descendientes (tanto mutados como no), se busca si ya existe esa solución en la población. De ser así, se descarta y si no, se acepta.

Finalmente, si un descendiente es mejor que la peor solución almacenada, la reemplazará en la población.

3.5. Búsqueda local dentro del Algoritmo genético

Dentro del algoritmo genético, se le aplicará la búsqueda local, explicada a continuación, con una probabilidad del 1% a los descendientes resultantes del cruce y mutación (en algunos casos) propios de un algoritmo genético, para intentar mejorar su desempeño.

4. Búsqueda Local

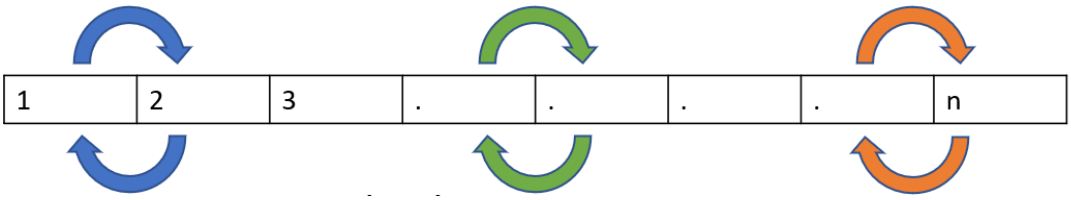
Una vez ha finalizado el algoritmo genético y, por lo tanto, el algoritmo ha devuelto la mejor solución que ha sido capaz de encontrar, se procede a realizar una búsqueda local para estudiar si en el entorno de la solución obtenida existe una solución mejor.

La búsqueda local que se empleará consistirá en dos pasos. En el primero, se actuará sobre los dos vectores de la solución obtenida, aunque de un modo distinto en cada uno de ellos (como ya se ha comentado anteriormente, una solución consta de dos vectores: el vector que posee el orden de recogida de los contenedores y el vector que dispone de las localizaciones ordenadas que ocuparán dichos contenedores).

El proceso de búsqueda sigue el siguiente ciclo, donde se empieza con i=0:

- Se permuta el valor de la posición i en el primer vector correspondiente al orden de recogida de los contenedores con el valor de la posición inmediatamente posterior, es decir, la posición i+1. Después se realiza exactamente lo mismo con el segundo vector correspondiente a las localizaciones.
- Si dichas permutaciones resultan en una mejora de la función objetivo se validan y se consolidan y si no mejoran la función objetivo se deshace la permutación en el segundo vector para evaluar solamente la permutación en el primer vector.
- Si esta permutación mejora la función objetivo se consolida y si no, se deshace.
- Se prosigue con el procedimiento, a menos de haber llegado al final del primer vector (alcanzar i=n-1), comenzando nuevamente con el primer paso, pero aumentado en n en una unidad, es decir, prosiguiendo con la siguiente posición de los dos vectores.

Este procedimiento acabará recorriendo ambos vectores evaluando y consolidando aquellas permutaciones que sean beneficiosas (tal y como se muestra en la figura).



Finalmente, la búsqueda local nos proporciona la solución definitiva (la mejor que ha sido posible encontrar) a la instancia seleccionada y la cual nos proporciona los datos iniciales.

Una vez acabado el proceso anterior, se procederá al segundo paso el cual consistirá en probar las localizaciones que queden libres sustituyendo una por una las localizaciones ya asignadas, es decir, si la primera localización está libre se sustituye una por una en las localizaciones ya ocupadas.

5. Resultados

Los resultados obtenidos para cada instancia y en cada paso del algoritmo de optimización heurística se muestran en la siguiente tabla:

Instancia	8 containers Nº2	8 containers Nº5	40 containers Nº1
Heurística 1 (F.O)	333 (22% más grande)	264 (13% más grande)	1580 (81% más grande)
Heurística 2 (F.O)	275 (0,7% más grande)	264 (13% más grande)	879 (0,7% más grande)
Genético (F.O)	275 (0,7% más grande)	235 (0,9% más grande)	879 (0,7% más grande)
Genético + Busq. Local (F.O)	273 (0% más grande)	233 (0% más grande)	873 (0% más grande)
Genético con Busq. Local + Busq.Local (F.O)	273	233	873

donde el porcentaje de mejora es fruto de comparar la solución proporcionada en cada paso con la mejor solución que nos proporciona el algoritmo completo para cada instancia. Concretamente la expresión del porcentaje de mejora es la siguiente:

$$\%\, mejora = \frac{F.O\ de\ la\ solución - F.O\ de\ la\ mejor\ solución}{F.O\ de\ la\ mejor\ solución} x100$$

6. Conclusiones

A partir de los resultados obtenidos se puede concluir que se ha conseguido optimizar cada una de las instancias proporcionadas correctamente dando así una solución de calidad al problema de localización de contenedores que se planteaba. Esto se ha logrado mediante la utilización de un algoritmo genético con búsqueda local que se ha detallado paso a paso en este estudio y cuyo código en Python se encuentra en el Apéndice 1 de este trabajo.

Para finalizar hace falta mencionar que la segunda solución heurística inicial ya se aproxima muchísimo a lo que finalmente nos devuelve el algoritmo, pues esta solución ya posee mucha calidad y es difícilmente optimizable. Concretamente el algoritmo genético en si mismo solo consigue optimizar la instancia nº5 de 8 contenedores y es mediante la búsqueda local que ya sí se consigue optimizar todas las instancias aunque en todos los casos la mejora es muy pequeña.

Apéndice 1. Código en Python

El código desarrollado es posible verlo en el siguiente enlace: https://bit.ly/3p89aKs