TP2 - Logística centralizada de primera milla

Tecnología Digital V: Diseño de Algoritmos

Federico Giorgi Gastón Loza Montaña Tomás Curzio

Contents

Introducción

Contexto

En los últimos años, el e-commerce ha crecido de manera exponencial en el mundo, incluyendo nuestra región. Las empresas, como en cualquier otro rubro, buscan reducir sus costos operativos, con el objetivo de obtener un mayor margen de ganancia. Para vendedores con un gran volumen de ventas, se suelen planificar visitas para recolectar los productos, pero realizar esta acción con muchos vendedores pequeños (que no realizan tantas ventas) es muy costoso en terminos logísticos, por lo que se les suele encargar a ellos que entreguen los paquetes a un punto de recolección de la empresa (usualmente llamado *primera milla*).

El problema y la decisión a tomar

El trabajo nos pone en el contexto de consultores de la empresa **ThunderPack** que provee servicios logísticos y además ofrece la posibilidad a otras empresas logísticas de almacenar paquetes en depósitos (puntos) de ThunderPack para que luego las mismas pasen a recogerlos por dichos puntos (la primera milla ya no seria un depósito propiedad de la empresa si no un depósito alquilado).

ThunderPack actualmente tiene un sistema donde los vendedores eligen a que depósito mandan los paquetes y pueden cambiarlo sin previo aviso. La empresa ha realizado estudios y nota que muchas veces esto resulta en un uso ineficiente de su red de depósitos, por lo que consideran cambiarse a una modalidad centralizada donde cada vendedor tendrá su depósito asignado al cual hará envios regularmente. Estas asignaciones se harán buscando maximizar la demanda y minimizar las distancias que los vendedores recorren.

La empresa tiene tres preguntas que querría idealmente responder:

- "¿Es suficiente la capacidad de la red de depósitos, o es necesario expandir la misma para poder dar respuesta a todos los vendedores?"
- "¿Es posible encontrar una asignación donde los vendedores recorran una distancia razonable para entregar sus paquetes?"
- "¿Es factible desarrollar una herramienta que nos permita experimentar con distintos escenarios y obtener soluciones de buena calidad en unos pocos minutos?"

Al final de este informe, con modelos para el problema y análisis sobre los mismos, buscaremos responder estas preguntas.

El modelo

El problema será modelado mediante el Problema de Asignación Generalizada (GAP), el cual formularemos para el contexto de **ThunderPack** de la siguiente manera. Sea $N = \{1, \ldots, n\}$ el conjunto de vendedores y $M = \{1, \ldots, M\}$ el conjunto de depósitos, cada depósito $i \in M$ tiene una capacidad máxima de c_i. Dado un vendedor $j \in N$ y un depósito $i \in M$, d_{ij} determina la demanda y c_{ij} la distancia a recorrer del vendedor j en caso de ser asignado al depósito i. Una vez definidos los elementos, podemos representar a la solución como los conjuntos $\Gamma_1, \ldots, \Gamma_m \subseteq N$, con Γ_i el conjunto de vendedores asignados al depósito $i \in M$. Así, el problema consiste en:

1. Asignar cada vendedor $j \in N$ a exáctamente un depósito $i \in M$, es decir que $, \Gamma_i \cap \Gamma_k = \emptyset$ si $i \neq k, i, k \in M$
2. No se debe superar la capacidad de cada depósito, es decir que para cada $i \in M$:

$$\sum_{j \in \Gamma_i} d_{ij} \leq c_i$$

3. Minimizar la distancia total de la asignación:

$$\sum_{i=1}^{n} \sum_{j \in \Gamma_i} c_{ij}$$

Además, es importante notar que dependiendo las capacidades y demandas dadas, podría dificultarse encontrar una solución factible. Definimos:

$$c_{\max} = \max_{i \in M, j \in N} c_{ij}$$

como la distancia máxima posible a recorrer por cada vendedor. Podríamos tener soluciones parciales en las cuales haya vendedores que no sean asignados a ningún depósito, en aquellos casos se tomará una penalización de $3 \times c_{max}$ para cada uno.

Heuristicas Constructivas

Se pueden plantear diferentes heurísticas para resolver el problema de GAP. En principio, necesitaremos heuristicas constructivas para obtener soluciones factibles pero seguramente no óptimas en un tiempo considerablemente bajo. Para ello, proponemos tres heuristicas constructivas distintas.

Mejor ajuste (?

Distancia mínima

La primera heuristica construcvitva que planteamos consiste en asignar a cada vendedor con el almacén que menos distancia supone, sin excedernos de su respectiva capacidad. El procedimiento de la heurística consiste en los siguientes pasos:

1. Tomar un vendedor $j \in N$.
2. Asignarlo al depósito $i \in M$ más cercano factible, es decir, cuya capacidad c_i sea mayor o igual a la demanda d_{ij}.
3. Actualizar la capacidad c_j del depósito disminuyendole la demanda d_{ij} del vendedor asignado.
4. Repetir para todos los vendedores.
5. Para todos los vendedores no asignados (en el caso de que hayan), sumo la penalidad de no poder asignarlos.

Esta estrategia es totalmente válida. Sin embargo, es posible que existan casos donde tener que recurrir a otro depósito debido a que el más cercano no es factible sea muy costoso y por lo tanto aumente considerablemente el valor de nuestra función objetivo. Por este motivo presentamos un enfoque similar pero ajustado en la próxima heurística.

Martello & Toth

La idea de esta heurística fue tomada del paper "A class of greedy algorithms" donde se mencioa la heuristica idea por los autores Martello y Toth. En este caso, para cada vendedor $j \in N$, bucasmos los dos depósitos $l, k \in M$ más cercanos, y además, computamos la diferencia entre estas distancias, es decir, calculamos $dif = d_{kj} - d_{lj}$ siendo l el depósito más cercano al vendedor j y k el segundo más cercano.

Luego, ordenamos los vendedores en base a las diferencias calculadas y los asignamos en este orden, es decir, aquellos vendedores con mayor diferencia serán asignados a su mejor depósito primero (si es que este es aún factible).

La lógica detrás de esta heurística es que al calcular los dos mejores depósitos (en base a menor distancia) para cada vendedor y luego obtener la diferencia, podemos tener en cuenta aquellos vendedores cuyo costo de no poder ser asignados en su mejor opción (desición local) sea mayor asignándolos primero. Esto puede suponer importantes mejoras, debido a que nos aseguramos que aquellos vendedores que por capacidad se vean forzados a ir a su segundo depósito más cercano, no causarán un aumento significativo en el valor objetivo.

Operadores de búsqueda local

Una vez obtenidas nuestras soluciones inciales factibles, buscaremos mejorarlas con operadores de búsqueda local. En este caso planteamos el operador **Swap** y **Relocate**. Para ambos, la condición para movernos de una solución a otra será el criterio de aceptación de best improvement. Es decir, sea s la solución, tomaremos la s' $\in N(s)$ tal que:

$$\text{s'} = \arg\max_{s \in \mathbb{N}(\sim)} f(s)$$

y

$$f(s') < f(s)$$

Swap

El operador swap toma dos vendedores de distintos depósitos y evalúa si intercambiándolos (siempre y cuando sea factible) conseguimos una mejora en el valor objetivo. Formalmente, siendo s el conjunto solución tomamos $j_1, j_2 \in s$ asignados a los depósitos i_1 e i_2 respectivamente, con $i_1 \neq i_2$. Posteriormente, removeremos j_1 de i_1 para insertarlo en i_2, y a j_2 de i_2 para insertarlo en i_1 en caso de que se cumplan las siguientes condiciones:

$$c_{i_1 j_2} + c_{i_2 j_1} < c_{i_1 j_1} + c_{i_2 j_2}$$
$$d_{i_1 j_2} \leq \bar{c}_{i_1} + d_{i_1 j_1}$$
$$d_{i_2 j_1} \leq \bar{c}_{i_2} + d_{i_2 j_2}$$

siendo $\bar{c}_k$ la capacidad restante del depósito k.

Relocate

El operador relocate toma un vendedor asignado a un depósito y evalúa los demás depósitos para ver si se puede asignar a algún otro depósito que mejore el valor objetivo, es decir, que distancia total recorrida por los vendedores sea menor. Formalmente, tomamos un vendedor $j \in s$ asignado al depósito i, consideramos todos los posibles depósitos $k \neq i$ y removemos j del depósito i para insertarlo en k en caso de que se cumplan las siguiente condiciones:

$$c_{kj} < c_{ij}$$
$$dkj \leq \bar{c}_k$$

con $\bar{c}_k$ la capacidad restante del depósito k. Luego, repetimos estos pasos para todos los vendedores de la solución.

Metaheurística: VND

La metaheurística que utilizaremos será VND (Variable Neighborhood Descent), cuya idea principal es tomar una solución inicial y buscar una mejora en la solución en diferentes vecindarios. Definimos vecindario como todas las soluciones que podemos alcanzar aplicando una vez el operador de busqueda local. La busqueda se realiza en el vecindario actual hasta que se encuentra una solución que mejore la actual. Luego, se pasa al siguiente vecindario y se repite el proceso hasta que no podamos encontrar mejoras.

Implementación

En nuestra solución, utilizaremos como operadores de busqueda local a swap y relocate. La implementación de VND realizada permite elegir el orden de aplicación de los operadores y si se utilizarán ambos o solo uno. (aca me está costando explicarlo la verdad).

Evaluación comparativa de las estrategias en intancias de Benchmark

Dado que la principal motivación de la formulación de diferentes alternativas de heurísticas es la reducción de las distancia recorrida por los clientes, nos proponemos, a través de experimentaciones con instancias de Benchmark para GAP, evaluar cómo se comportan en base a valor objetivo y tiempo.

Experimentación

Para comparar las distintas estrategias entre sí definimos las siguientes métricas:

- %cost_gap: diferencia porcentual relativa de la estrategia batching sobre la greedy en base a los costos.
- %time_gap: diferencia porcentual relativa de la estrategia batching sobre la greedy en base al tiempo de ejecución.

Hipotesis

Planteamos como hipótesis que el operador relocate por si solo probablemente nunca genere ninguna mejora en soluciones generadas por la heurística de distancia mínima, debido a que su función a grandes rasgos es revisar si hay algún depósito factible más cercano al asignado, pero esto no ocurrirá debido a que el funcionamiento de la heurística constructiva es exactamente ese, por lo que si existiese dicho depósito, ya se habría asignado previamente. El operador relocate es especialmente útil para poder asignar los vendedores que no fue posible asignar a ningún depósito previamente, teniendo la posibilidad de evitar la alta penalización que estos suponían.

Discusión y análisis de resultados

Comparativa de constructivas:

{bla, bla}

type_instance	ov_mincost_vs_bestfit%	ov_mincost_vs_mt%	ov_mt_vs_bestfit%
a	-1.15347	0.00995446	-1.17576
b	-0.813656	0.0433551	-0.928853
e	0.514431	-0.0180081	0.523833

Swap vs. Relocate

{bla, bla}

Cada constructiva vs Relocate:

type_instance	mincost_vs_relocate%	bestfit_vs_relocate%	mt_vs_relocate%
a	0	0.53162	0
b	0	0	0
e	0	0.00233594	0

validar hipostesis !!

Cada constructiva vs Swap:

type_instance	mincost_vs_swap%	bestfit_vs_swap%	mt_vs_swap%
a	0.00867928	0.496591	0.00026878
b	0.069682	0.296073	0.0558688
e	0.00904424	0.0290887	0.00861152

Swap vs. VND_1 (relocate, swap)

type_instance	mincost_swap_vs_VND1%	bestfit_swap_vs_VND1%	mt_swap___vsVND1%
a	0.000395507	0.0722756	0
b	0.2837	0.562937	0.247701
e	0.0574338	0.077536	0.0460814

Swap vs. VND_2 (swap, relocate)

type_instance	mincost_swap_vs_VND2%	bestfit_swap_vs_VND2%	mt_swap___vsVND2%
a	0.000395507	0.0724913	0
b	0.195097	0.343229	0.167047
e	0.00286099	0.0028813	0.000521941

Tiempos de ejecución

Para instancias con $m = 1600$.

MinCostHeuristic:

m	st_mincost (s)	st_mincost_swap (s)	st_mincost_relocate (s)	st_mincost_relocate_swap (s)	st_mincost_swap_relocate (s)
20	0.0026151	120.906	0.0065021	102.888	116.098
40	0.00502786	108.86	0.0126091	98.9035	108.904
80	0.0102733	115.616	0.0247985	105.5	107.972

BestFitHeuristic:

m	st_bestfit (s)	st_bestfit_swap (s)	st_bestfit_relocate (s)	st_bestfit_relocate_swap (s)	st_bestfit_swap_relocate (s)
20	0.00584707	686.884	0.00958234	647.991	680.107
40	0.0113951	666.598	0.0188384	649.511	659.547
80	0.0241176	653.314	0.0386273	654.542	651.38

MTHeuristic:

m	st_mt (s)	st_mt_swap (s)	st_mt_relocate (s)	st_mt_relocate_swap (s)	st_mt_swap_relocate (s)
20	0.00338484	77.4777	0.00739598	69.5559	78.8022
40	0.00637205	73.6666	0.0138893	67.6196	74.313
80	0.0123502	88.5029	0.0271206	83.8109	93.7482

Evaluación comparativa de las estrategias en intancia real

Experimentación

Discusión y análisis de resultados

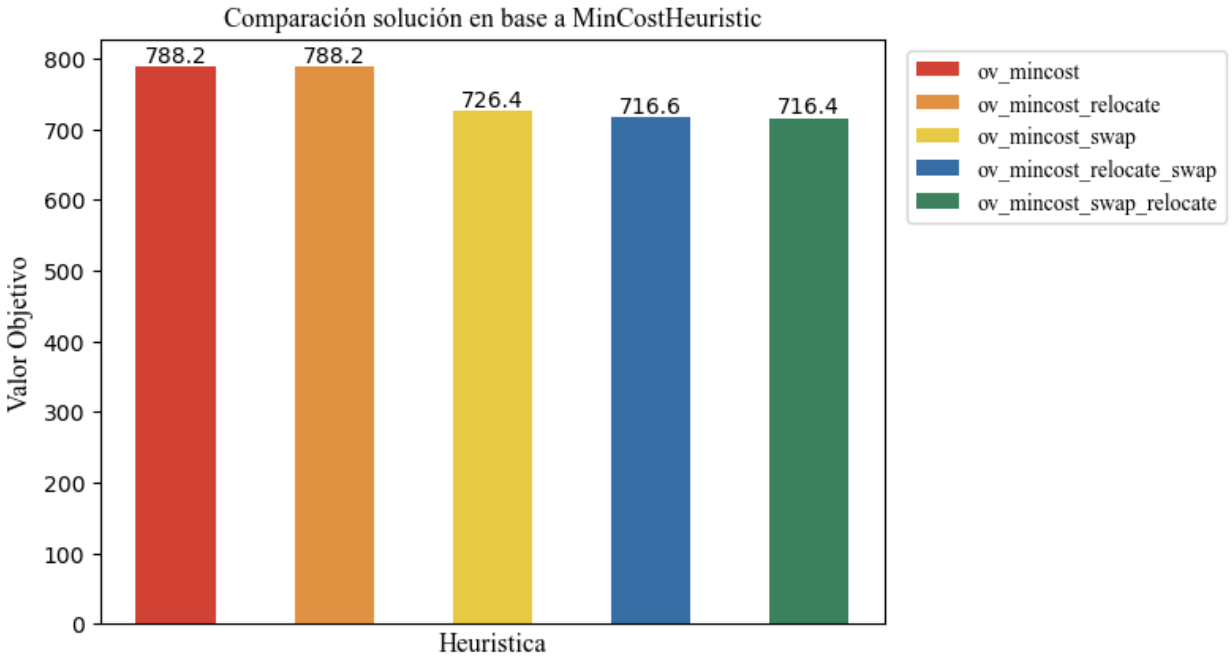

Figure 1: Comparación instancia real partiendo de MinCostHeuristic

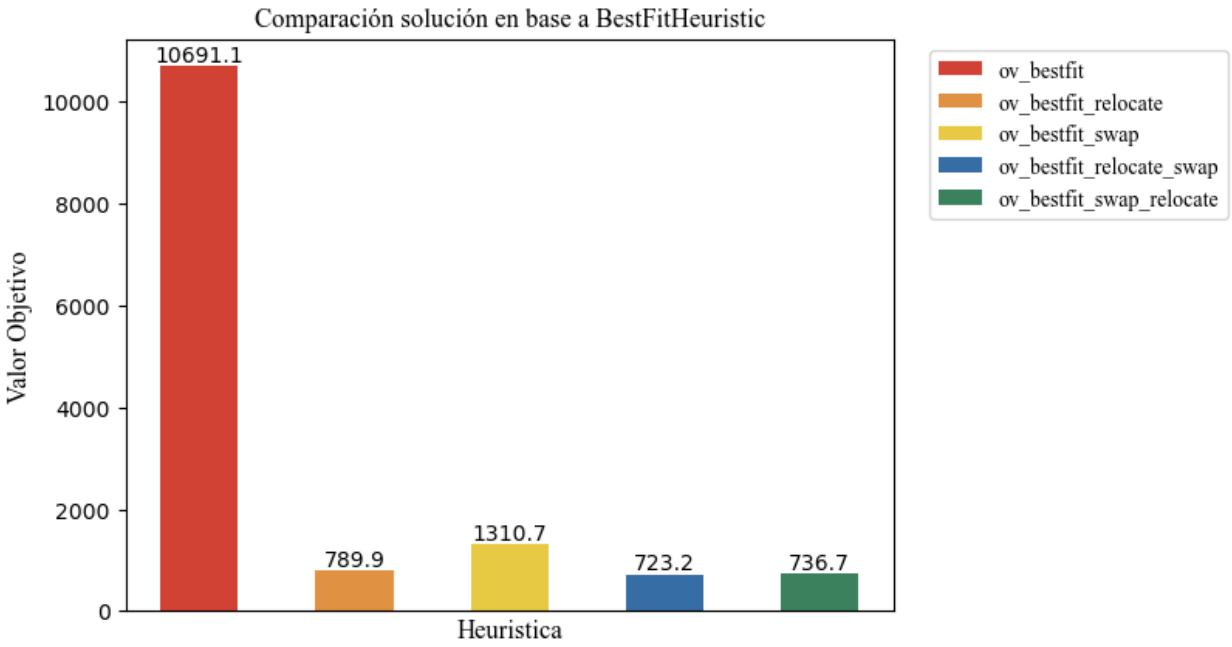

Figure 2: Comparación instancia real partiendo de BestFitHeuristic

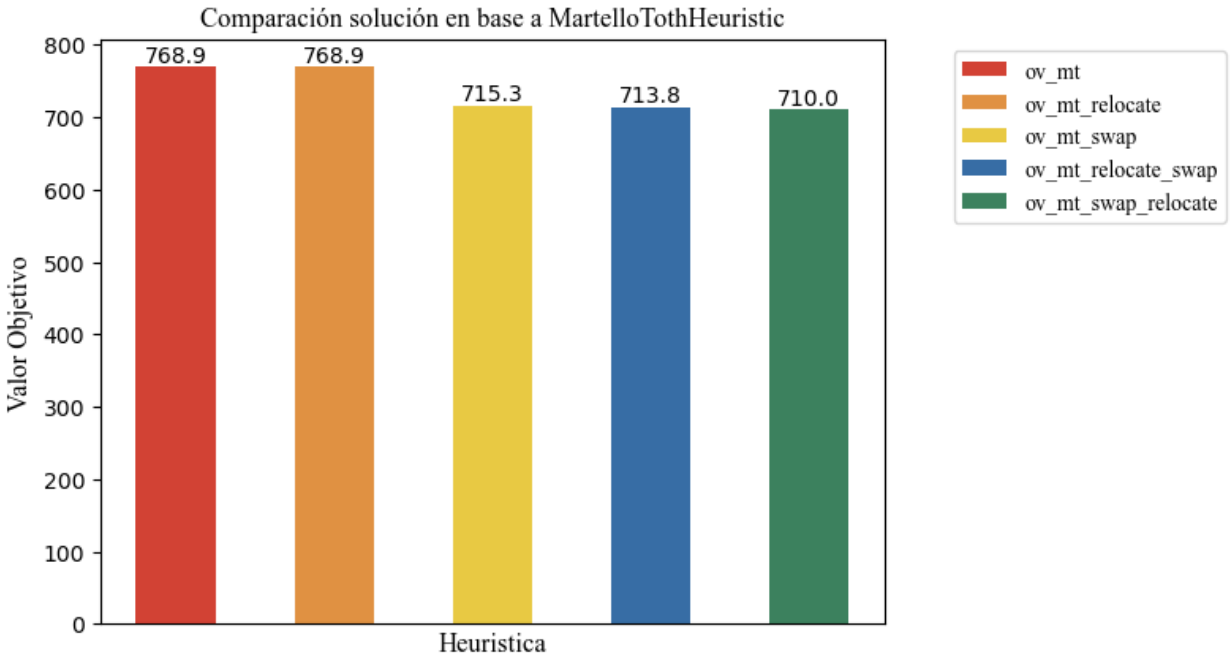

Figure 3: Comparación instancia real partiendo de MTHeuristic