Resolviendo un problema de ruteo de vehículos con capacidad a través de GRASP

García García Sacbe, Gutiérrez Salazar Carlos Cuauhtemoc

Centro de Investigación en Matemáticas
Unidad Monterrey
Monterrey, Nuevo León
Reporte de proyecto final de optimización.
Resolviendo un problema VRPPD a través de GRASP.

31 de mayo, 2023

Introducción

El problema de generación de rutas para vehículos (VRP) es una variación del problema del viajante de comercio (TSP). Este problema implica determinar el conjunto óptimo de rutas para que una flota de vehículos entregue bienes o brinde servicios a un conjunto de clientes. El objetivo es minimizar la distancia total recorrida o el costo total mientras se satisfacen varias restricciones, como las limitaciones de capacidad del vehículo o las ventanas de tiempo del cliente.

Este proyecto trabaja sobre VRPPD, donde los vehículos son responsables de recoger artículos y entregarlos a otros destinos específicos mientras minimizan la distancia total o el costo. Resolver estos problemas de manera óptima puede ser computacionalmente costoso, por lo que se han desarrollado varios algoritmos y heurísticas para encontrar soluciones casi óptimas de manera eficiente.

Este tipo de problema es clásico en el ámbito de la logística, y podemos entenderlo como el problema que podría tener una industria en Monterrey con servicio de transporte para sus trabajadores. Este servicio busca recorrer las viviendas de sus trabajadores con un número especifico de vehículos minimizando los costos asociados, como el tiempo de uso del transporte y la distancia recorrida.

El objetivo es minimizar el costo asociado a las rutas dadas las viviendas de los trabajadores utilizando un algoritmo GRASP pseudo-aleatorio para obtener soluciones al problema. El resultado a obtener son las rutas óptimas de los vehículos de la empresa.

Descripción del problema

A través de la formulación de variables, restricciones y funciones objetivo, este modelo busca minimizar los costos totales, en este caso la distancia total y maximizar la eficiencia en la distribución de carga en los autobuses. A continuación se describe la formulación matemática de un modelo VRPPD [1][4].

Parámetros:

- S: Conjunto de sitios de recogida y entrega.
- V: Conjunto de autobuses.
- d_{ij}: Distancia entre el nodo i y el nodo j; $i, j \in S$.
- q_i: Cantidad de pasajeros a recoger en el nodo i; $i \in S$.
- κ_j: Capacidad total de los autobuses; $j \in V$

Variables de decisión:

- x_{ijk}: Variable binaria que indica si el vehículo k va desde el nodo i al nodo j; $k \in V$, $i, j \in S$.
- y_{ik}: Variable binaria que indica si el vehículo k visita el nodo i; $k \in V$, $i \in S$.
- u_i: Variable de capacidad acumulada en el sitio i; $i \in S$.

Función objetivo:

Minimizar la distancia total recorrida, es decir, minimizar

$$\sum_{i,j,k} d_{ij} x_{ijk},$$

dadas las rutas obtenidas con las restricciones detallas en la siguiente sección.

Factibilidad:

En el problema los autobuses tienen que pasar por todos los nodos donde haya demanda, además se tienen que cumplir que la capacidad acumulada de pasajeros no exceda la capacidad del vehículo. También se tiene que cumplir que si el vehículo k visita el nodo i y luego el nodo j, entonces se recoge en el nodo i antes de la entrega en el nodo j, por otra parte se tiene que garantizar que el vehículo k siga un único ciclo en su ruta, por último, cada vehículo debe comenzar y terminar en el punto de partida, ya sea una escuela o centro turístico. Se muestra la formulación matemática de las restricciones.

- Cada nodo debe ser visitado exactamente una vez por algún vehículo:

$$\sum_{k} y_{ik} = 1 \quad \forall i \in S$$

- Restricciones de capacidad:
  - Capacidad inicial en cada sitio es cero:

$$u_i = 0 \quad \forall i \in S$$

  - Capacidad acumulada en cada nodo no puede exceder la capacidad del vehículo:

$$u_i + \sum_{k} q_i y_{ik} \leq \kappa \quad \forall i \in S, k \in V$$

- Restricciones de recogida antes de entrega: Si el vehículo k visita el nodo i y luego el nodo j, entonces se recoge en el nodo i antes de la entrega en el nodo j.

$$y_{ik} \leq y_{jk} \quad \forall i, j, k$$

- Restricciones de enrutamiento: Garantizar que el vehículo k siga un único ciclo en su ruta (rutas de salida igual a las de entrada):

$$\sum_{i,j} x_{ijk} - \sum_{j} x_{jik} = 0 \quad \forall i, j, k$$

- Restricciones de inicio y fin: Cada vehículo debe comenzar y terminar en el depósito:

$$\sum_{i} y_{ik} = 2 \quad \forall k \in V$$

Algorithm 1: Construcción-RG

Require: $k, E, c(\cdot)$
$x \leftarrow \emptyset$;
$C \leftarrow E$;
Calcular el costo miope $c(e), \forall e \in C$;
for $i = 1, 2, \ldots, k$ **do**
  if $C \neq \emptyset$ **then**
    Elegir el elemento e al azar de C;
    $x \leftarrow x \cup \{e\}$;
    Actualizar el conjunto de candidatos C;
    Calcular el costo miope $c(e), \forall e \in C$;
while $C \neq \emptyset$ **do**
  $e_* \leftarrow \text{argmin}\{c(e) \mid e \in C\}$;
  $x \leftarrow x \cup \{e_*\}$;
  Actualizar el conjunto de candidatos C;
  Calcular el costo miope $c(e), \forall e \in C$;
return x;

Metodología

GRASP son las siglas de *Greedy Randomized Adaptive Search Procedure*, es un algoritmo metaheurístico utilizado para resolver problemas de optimización combinatoria. Combina elementos de algoritmos codiciosos y técnicas de búsqueda local para encontrar soluciones casi óptimas.

La idea principal detrás de GRASP es construir iterativamente una solución seleccionando elementos o componentes de manera codiciosa mientras permite la aleatorización para introducir diversidad.

Durante la fase de construcción, GRASP construye una solución de forma incremental seleccionando iterativamente elementos del espacio de búsqueda del problema en función de un criterio codicioso. El criterio codicioso normalmente evalúa los elementos en función de su potencial para contribuir a la función objetivo. Sin embargo, para promover la diversificación, se introduce un mecanismo de aleatorización que permite cierto grado de aleatoriedad en el proceso de selección. Esta aleatoriedad ayuda al algoritmo a explorar diferentes regiones del espacio de búsqueda.

Una vez que se construye una solución, comienza la fase de búsqueda local. En esta fase, el algoritmo explora la vecindad de la solución actual haciendo pequeñas modificaciones para mejorar su calidad. La búsqueda local tiene como objetivo optimizar la solución dentro de su región local mediante la aplicación de modificaciones locales.

GRASP continúa iterando entre las fases de construcción y búsqueda local hasta que se cumple un criterio de parada, como un número predeterminado de iteraciones o alcanzar un límite de tiempo. A lo largo de las iteraciones, mantiene la mejor solución encontrada hasta el momento, que representa la solución más conocida al problema de optimización.

Construcción de las soluciones

Este trabajo la construcción de soluciones se basó en el algoritmo 1 que consiste en la generación de respuestas factibles psedudo-aletorias [2]. A este algoritmo se aplicaron modificaciones para mejorar la selección de candidatos.

Una de las modificaciones fue el criterio de parada, ya que además de tomar en cuenta que el conjunto de nodos factibles no sea vacío también se consideró que las cargas de estos nodos sea menor que la carga total establecida. Otra modificación fue la selección de candidatos, ya que no se tomó necesariamente el de mínima distancia sino que se ponderó la distancia como se muestra en la ecuación 1

$$\epsilon = \frac{d_{ij}}{q_j}, \tag{1}$$

donde ϵ es la ponderación obtenida de cada nodo, d_{ij} es la distancia del nodo i al nodo j y q_j es la carga de pasajeros en el nodo j. De esta manera se toman en cuenta la relación distancia - pasajeros. De esta ponderación se definió una *lista restringida de candidatos* (RLC) basada en calidad, es decir, sea $0 \leq \alpha \leq 1$ se seleccionan los candidatos tales que satisfacen la restricción 2

$$\epsilon_{\text{mín}} \leq \epsilon_i \leq \epsilon_{\text{mín}} + \alpha(\epsilon_{\text{máx}} - \epsilon_{\text{mín}}) \tag{2}$$

donde $\alpha = 0$ corresponde a una implementación puramente codiciosa, y $\alpha = 1$ es completamente aleatoria [3][5]. En este reporte se utilizó un valor de $\alpha = 0{,}3$.

Búsqueda local

A partir de las soluciones factibles generadas se realiza una búsqueda en la región factible, utilizando el método $2-opt$ que consiste en Sustituir cualquier par de aristas no adyacentes de la solución S por el único par de aristas que recrea un ciclo hamiltoniano [3], dicho de otra manera, supongamos que tenemos una solución inicial representada por la permutación $P = [p_1, p_2, ..., p_n]$, donde n es el número total de sitios. El método $2-opt$ busca intercambiar dos aristas de la ruta, es decir, encontrar dos índices i y j (con $i < j$) y construir una nueva solución P' donde se invierte el orden de los sitios entre los índices i y j.
La nueva solución P' se puede representar matemáticamente de la siguiente manera:

$$P' = \{p_1, ..., p_{i-1}, p_j, p_{j-1}, ..., p_{i+1}, p_i, \\ p_{i+1}, ..., p_{j-1}, p_{j+1}, ..., p_n\} \tag{3}$$

En esta representación, los sitios desde p_1 hasta p_{i-1} se mantienen en el mismo orden, seguidas por las ciudades desde p_j hasta p_{i+1} en orden inverso, y luego las ciudades desde p_{j+1} hasta p_n se mantienen en el mismo orden. La imagen 1 muestra de forma gráfica el intercambio de rutas para la obtención de mejores costos.

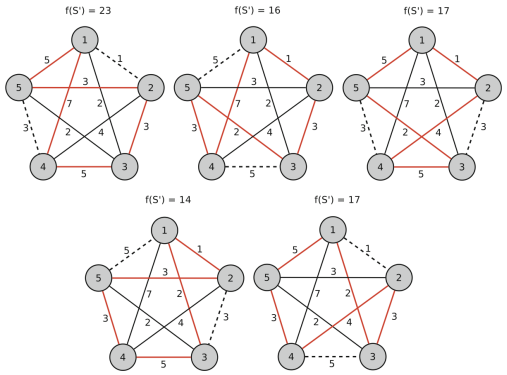

Figura 1. Cinco vecinos de una solución inicial junto con sus costes, indicando para cada uno de ellos las aristas eliminadas mediante líneas discontinuas usando $2-opt$.

Algoritmo GRASP

Un GRASP es un método multi-arranque, en el cual cada iteración GRASP consiste en la construcción de una solución miope aleatorizada seguida de una búsqueda local usando la solución construida como el punto inicial de la búsqueda local. Este procedimiento se repite varias veces y la mejor solución encontrada sobre todas las iteraciones GRASP se devuelve como la solución aproximada [2].

En el algoritmo 2, se define un parámetro $i_{\text{máx}}$ que determina el número máximo de iteraciones permitidas. El algoritmo comienza inicializando el valor de la mejor solución encontrada, f^*, como infinito.

A continuación, se ejecuta un bucle en el que se generan soluciones utilizando una combinación de estrategias de construcción voraz y aleatorias. La función $GreedyRandomized()$ genera una solución inicial utilizando una estrategia voraz aleatorizada, mientras que la función $LocalSearch()$ mejora iterativamente la solución generada mediante la exploración de vecindarios locales.

Algorithm 2: GRASP

Require: $i_{\text{máx}}$
$f^* \leftarrow \infty$;
for $i \leq i_{máx}$ **do**
  $x \leftarrow$ GreedyRandomized();
  $x \leftarrow$ LocalSearch(x);
  if $f(x) < f^*$ **then**
    $f^* \leftarrow f(x)$;
    $x^* \leftarrow x$;
return x^*;

Después de generar una solución, se verifica si su valor objetivo, $f(x)$, es mejor que el mejor valor objetivo encontrado hasta el momento, f^*. Si es así, se actualiza f^* con el nuevo valor y se guarda la solución en x^*.

Una vez que se completa el bucle de iteraciones, se devuelve la mejor solución encontrada, x^*, que se considera la solución óptima aproximada.

Experimentación computacional

Se tomaron los parámetro de $\alpha = 0{,}3$, $max\_iterations = 1000$, número máximo de pasajeros en los vehículos $\kappa = 60$ y número de vehículos $num\_aut = 4$.

La instancia para el modelo se define con 'd' la matriz de distancias y 'q' el arreglo de pasajeros asociado a cada sitio.

Para la instancia analizada, 'q' es un arreglo generado de forma aleatoria.

El número de sitios que se consideró fue de 15, los cuales fueron generados aleatoriamente utilizando una distribución uniforme conjunta sobre el polígono que incluye los municipios de San Nicolás de los Garza, Apodaca, Monterrey, San Pedro, Santa Catarina y Guadalupe, en el estado de Nuevo León.

La figura 2 es un ejemplo del resultado de este proceso aleatorio de generación de sitios

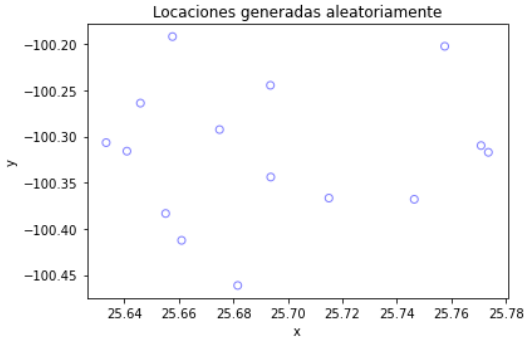

Figura 2. Los sitios fueron generados con una distribución uniforme conjunta, estos puntos se consideraron al estar dentro del polígono de la localidad elegida. Estos puntos generados aleatoriamente no representan los utilizados en la instancia analizada

Una vez generados los sitios, se utilizo la API de Google Maps para generar la matriz de distancias 'd'.

La API de Google Maps (interfaz de programación de aplicaciones en ingles) es un conjunto de herramientas y servicios proporcionados por Google que permite desarrollar sobre funcionalidad de Google Maps en aplicaciones propias. Proporciona una forma de acceder, mostrar e interactuar con mapas, datos de geolocalización, información de rutas y otros servicios geográficos. La utilización de esta API es gratuito pero limitado a un uso especifico por mes.

A continuación se describen las especificaciones de los equipos donde fue probado el algoritmo:

Macboock Air, M1 2020 Procesador *Apple M1 chip* que tiene 8 núcleos, sistema operativo y 16 GB de memoria *Ventura 13.3.1*

XPS 15 Laptop Procesador *Intel Core i7-127000H* de 12.ª generación con 16 GB de memoria, sistema operativo *Windows 11*

Sobre cada uno de estos equipos se probo el algoritmo 5 veces con lo que obtuvimos los resultados mostrados en la tabla **1**.

Conclusiones

En conclusión, este reporte presentó un enfoque GRASP constructivo para resolver el desafiante problema de rutas de

Equipo	XPS 15 Laptop	Macbook Air, M1
Mejor valor	46.89	46.89
Valor promedio	49.42	52.54
Peor valor	54.97	56.68
Mejor tiempo	0.170	0.141
Tiempo promedio	0.201	0.142
Peor tiempo	0.283	0.145

Tabla 1. Se tomaron las semillas 2023, 1023, 4023, 7023 y 53711, para la obtención de estos resultados

vehículos con recogida y entrega (VRPPD).

A través de iteraciones de las fases de construcción y búsqueda local, el algoritmo GRASP equilibró de manera efectiva la exploración y la explotación, refinando gradualmente la solución y convergiendo hacia soluciones casi óptimas. Los resultados obtenidos en nuestros experimentos demostraron la capacidad del algoritmo para encontrar soluciones de alta calidad.

El método GRASP constructivo abordó con éxito las complejidades del problema VRPPD y proporcionó resultados prometedores en términos de calidad de solución y eficiencia computacional. El algoritmo no solo abordó el desafío de encontrar rutas óptimas, sino que también consideró las limitaciones de recogida y entrega, lo que garantiza que las personas se transporten de manera eficiente dentro de los plazos especificados.

En la figura 3 se pueden observar una de las soluciones óptimas generadas por este algoritmo. Cada color representa un vehículo en ruta. En general, el método GRASP constructivo

Figura 3. Mapa de rutas resultantes obtenidas por el algoritmo GRASP. Cada color representa la el circuito de un vehículo dado, donde estos recorren la misma calle se combinan los colores a fin de distinguir. En este caso el valor obtenido fue de 46.89

presentado en este informe ofrece un enfoque práctico y efectivo para resolver el problema VRPPD. Su capacidad para manejar las limitaciones del mundo real y generar soluciones casi óptimas lo convierte en una herramienta valiosa para las empresas de logística y transporte que buscan optimizar sus operaciones de rutas de vehículos. La investigación futura puede explorar mejoras adicionales al algoritmo, como la incorporación de técnicas de optimización adicionales o la evaluación de su rendimiento en instancias de problemas a mayor escala.

Más adelante se podría considerar explorar diferentes estrategias para la fase de construcción de GRASP como con diferentes métodos de inicialización, mecanismos de diversificación y reglas miopes para seleccionar vehículos. Así como buscas otras técnicas para la fase de búsqueda local. Se podría considerar también combinar esta implementación del algoritmo GRASP con otros algoritmos o técnicas metaheurísticas para explotar sus puntos fuertes y superar las limitaciones. Por ejemplo, puede explorar la integración de búsqueda Tabú, recocido simulado, algoritmos genéticos u otros con GRASP para mejorar la calidad de la solución o para simplemente comparar el desempeño del algoritmo con otros realizando estudios comparativos para identificar sus fortalezas y debilidades en este tipo especifico de problemas. Las variaciones mencionadas pueden afectar la calidad y diversidad de las soluciones generadas.

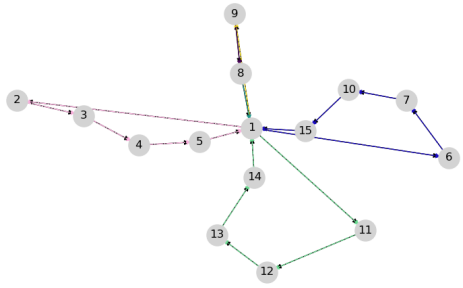

Figura 4. Grafo óptimo obtenido por la solución del método GRASP.

Bibliografía

[1] García C. Irma Delia, (2010). El problema de ruteo de vehículos, Universidad Autónoma de Coahuila, FC-UNAM, 21(33).
[2] Resende, M. G., Velarde, J. L. G. (2003). GRASP: Procedimientos de búsqueda miopes aleatorizados y adaptativos. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 7(19), 0.
[3] M. G. C. Resende y C. C. Ribeiro (2016). Optimization by GRASP: Greedy Randomized Adaptive Search Procedures. New York, NY: Springer New York.
[4] P. Toth y D. Vigo, Eds. (2002). The vehicle routing problem. en SIAM monographs on discrete mathematics and applications. Philadelphia, Pa: Society for Industrial and Applied Mathematics.
[5] M. J. Hirsch, C. N. Meneses, P. M. Pardalos, y M. G. C. Resende (2007), "Global optimization by continuous grasp", Optimization Letters, vol. 1, núm. 2, pp. 201–212, doi: 10.1007/s11590-006-0021-6.