Búsqueda Tabú

Un Método de Optimización Metaheurística

Nicolle Escobar
Jorge Andrino

25 de noviembre de 2024

Contenido

¿Qué es la Búsqueda Tabú?

- **Método de optimización metaheurística**
- Desarrollado por Fred Glover (1986)
- Mejora la búsqueda local mediante memoria adaptativa
- Evita óptimos locales mediante estructuras de memoria

Formulación del Problema de Optimización

Modelo general

$$\text{Minimizar/Maximizar } f(x)$$
$$\text{sujeto a: } x \in X$$

donde:

- $f(x)$ es la función objetivo
- X es el espacio de soluciones factibles
- x es una solución candidata

Espacio de Búsqueda

Características del Espacio

- **Discreto:** Conjunto finito o numerable de soluciones
- **Continuo:** Conjunto infinito no numerable
- **Restricciones:** Definen X (espacio factible)

Estructura de la vecindad

Para cada solución x:

$$N(x) = \{x' \in X : x' \text{ es alcanzable desde } x\}$$

donde $N(x)$ es la vecindad de x

Proceso de Optimización

Exploración local

- Evaluación de $f(x')$ para $x' \in N(x)$
- Selección del mejor movimiento no tabú
- Actualización de la mejor solución

Búsqueda global

- Escape de óptimos locales
- Balance exploración/explotación
- Criterios de aspiración

Estrategias de Optimización

Movimientos en el Espacio

1. **Movimientos de Mejora:**

$$f(x') < f(x) \text{ para minimización}$$

2. **Movimientos No Mejoradores:**

$$f(x') \geq f(x) \text{ pero permitidos para escapar}$$

3. **Movimientos Prohibidos:**

$$x' \in T \text{ (lista tabú) excepto si cumple aspiración}$$

Control de la Optimización

Criterios de Parada

- Número máximo de iteraciones
- Tiempo de ejecución límite
- Estancamiento en f_{best}
- Convergencia a valor objetivo

Métricas de Calidad

- f_{best} = Mejor valor encontrado
- Δf = Mejora relativa
- Gap = Distancia al óptimo conocido

Componentes clave

Solución inicial

- Exploración de la vecindad de forma iterativa
- Se mueve a la mejor solución de la vecindad

Criterio de Aspiración

- Permite movimientos tabú prometedores
- Flexibiliza la búsqueda

Componentes clave

Intensificación y diversificación

- Intensificación $\rightarrow$ explorar áreas prometedoras del espacio de soluciones
- Diversificación $\rightarrow$ explorar nuevas regiones del espacio

Lista Tabú

- Cuando un movimiento se puede considerar prohibido
- Ayuda a determinar la solución óptima

Exploración

Movimiento en el espacio de soluciones

Sea X el espacio de soluciones factibles. Una solución $x \in X$ es un punto en ese espacio.
Explorar significa evaluar diferentes puntos x' en el vecindad $N(x)$ de la solución actual

Exploración

Evaluación de la función objetivo

Sea $f(x)$ la función objetivo. Explorar implica calcular $f(x')$ para diferentes soluciones x'.

- En minimización: x' donde $f(x') < f(x)$
- En maximización: x' donde $f(x') > f(x)$

Exploración

Distancia entre soluciones

- Se define una métrica $d(x_1, x_2)$ que mide la distancia entre dos soluciones.
- Explorar cercanía: examinar x' respecto a $d(x, x')$

Tipo de espacio

- Para vectores binarios de longitud n, entonces $d(x, x')$ es la distancia de Hamming
- Para permutaciones de n elementos, $d(x, x')$ podría ser la distancia de τ de Kendall.

Exploración

Distancia entre soluciones

- Se define una métrica $d(x_1, x_2)$ que mide la distancia entre dos soluciones.
- Explorar cercanía: examinar x' respecto a $d(x, x')$

Tipo de espacio

- Para vectores binarios de longitud n, entonces $d(x, x')$ es la distancia de Hamming
- Para permutaciones de n elementos, $d(x, x')$ podría ser la distancia de τ de Kendall.

Ejemplo

Para implementar la búsqueda tabú con la τ de Kendall (distancia bubble sort), se puede proceder de la siguiente forma:

1. Definir movimientos como intercambios de elementos adyacentes
2. Usar la distancia para
   - Medir qué tan diferentes son dos soluciones
   - Definir el criterio de aspiración
   - Determinar el tamaño del vecindad a explorar

Con ello, se puede mantener una lista tabú de permutaciones cumpliendo que estén dentro de la distancia de soluciones ya visitadas.

Estructura de Memoria

Tipos de Memoria

1. **Memoria a Corto Plazo**
   - Lista tabú reciente
   - Previene ciclos

2. **Memoria a Largo Plazo**
   - Frecuencia de movimientos
   - Diversificación de la búsqueda

Memoria a Corto Plazo

Definición

Sea T la lista tabú y M un movimiento:

$T = \{m_1, m_2, ..., m_k\}$ donde k es el tamaño de la lista

M es tabú si $M \in T$

Características

- Almacena los últimos k movimientos realizados
- Previene ciclos en la búsqueda
- Estructura FIFO (First In, First Out)
- Tamaño típico: $\sqrt{n}$ donde n es el tamaño del problema

Pseudocódigo - Función principal

Procedimiento BúsquedaTabú

```
1:  Entrada: configuración_inicial, max_iteraciones
2:  mejor_solucion ← configuración_inicial
3:  solucion_actual ← configuración_inicial
4:  lista_tabu ← [ ]                          ▷ Lista de movimientos prohibidos
5:  frecuencias ← { }                         ▷ Estructura para frecuencias
6:  for iteracion = 1 hasta max_iteraciones do
7:      vecinos ← GenerarVecindad(solucion_actual)
8:      mejor_candidato ← NULL
9:      mejor_valor ← INFINITO
10:     [Continúa en siguiente diapositiva...]
11: end for
```

[1] Inicialización y estructura principal de búsqueda tabú: establece solución inicial, lista tabú para movimientos prohibidos y contador de frecuencias para intensificación/diversificación

Pseudocódigo - Evaluación de vecinos

Continuación del Bucle Principal

```
1: for cada vecino en vecinos do
2:     if vecino ∉ lista_tabu O CumpleCriterioAspiracion(vecino) then
3:         valor ← EvaluarSolucion(vecino)
4:         if valor ¡ mejor_valor then
5:             mejor_candidato ← vecino
6:             mejor_valor ← valor
7:         end if
8:     end if
9: end for
```

2

[2]Evaluación de soluciones vecinas: selecciona mejor candidato no prohibido o que cumpla criterio de aspiración

Pseudocódigo - Actualización de solución

Actualización y memoria

```
1: if mejor_candidato ≠ NULL then
2:     solucion_actual ← mejor_candidato
3:     if EvaluarSolucion(solucion_actual) ¡
   EvaluarSolucion(mejor_solucion) then
4:         mejor_solucion ← solucion_actual
5:     end if
6:     ActualizarListaTabu(lista_tabu, solucion_actual)
7:     ActualizarFrecuencias(frecuencias, solucion_actual)
8: end if
```

3

[3] Actualización de memoria adaptativa: actualiza mejor solución global, lista tabú y frecuencias de movimientos

Resultados

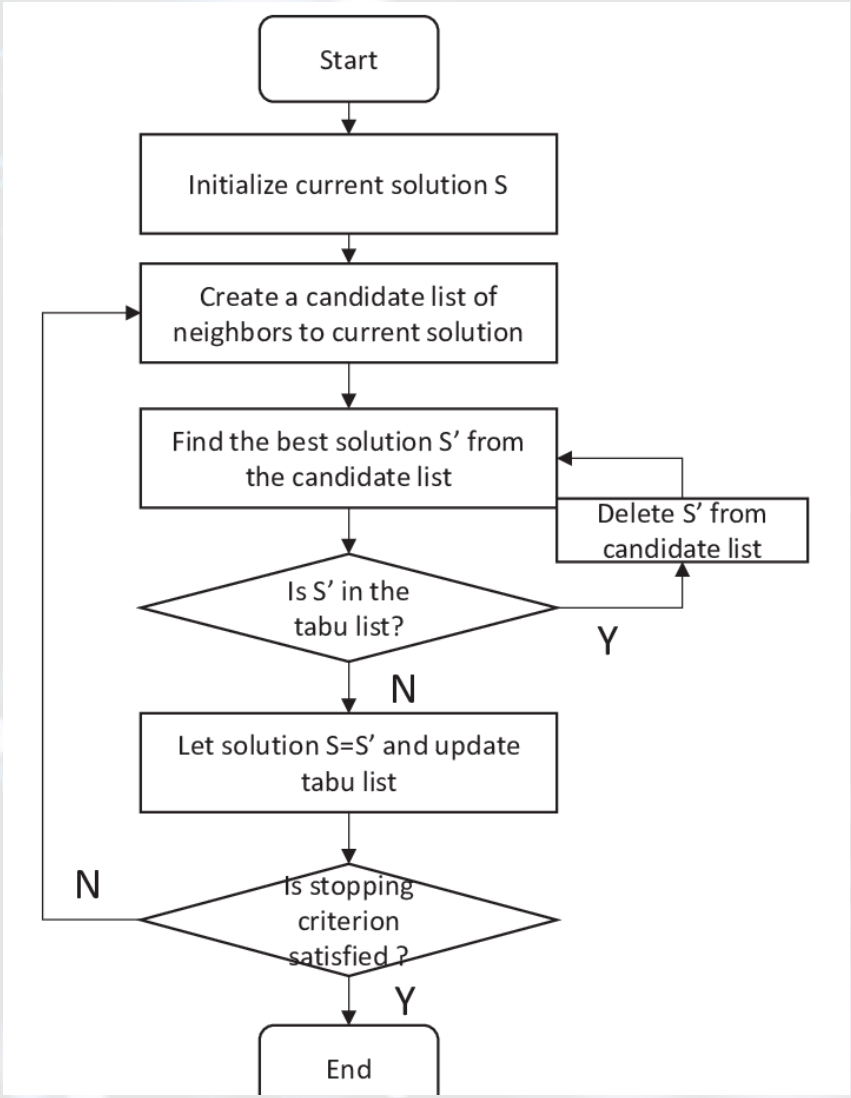


Figura: Diagrama de flujo de Búsqueda Tabú

Análisis del método

Ventajas

- Evita óptimos locales al mantener una lista tabú de soluciones visitadas recientemente
- Memoria adaptativa
- Criterios adaptables y flexibles

Limitaciones

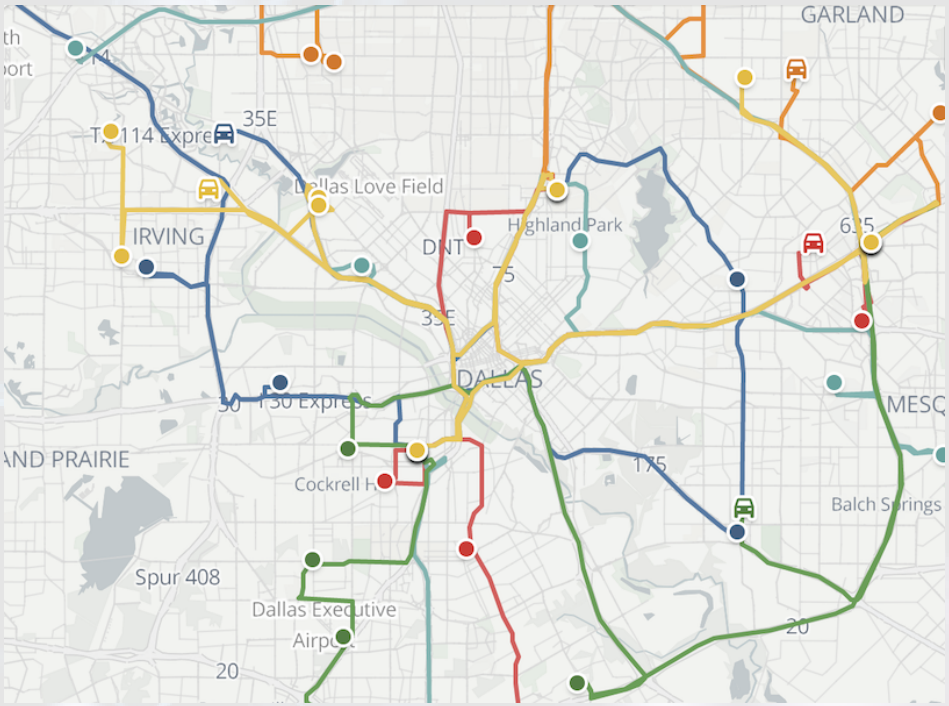
- No garantiza el óptimo global
- Diseño de estructuras (como los criterios de paro)
- Calibración de parámetros (tamaño de la lista, criterios de aspiración,...)
- Costo computacional

En resumen

Nota

La estructura tabú evita que la búsqueda regrese a soluciones recientes permitiendo movimientos potenciales que pueden conducir a mejores soluciones. Equilibra explotación y exploración del espacio de soluciones.

Aplicación al Problema de Enrutamiento de Vehículos



Aplicación al Problema de Enrutamiento de Vehículos

Problema de enrutamiento de vehículos

Es un problema de optimización combinatoria en el que

- Flota de vehículos entregan bienes de un depósito central $\rightarrow$ los clientes
- Cada cliente tiene una demanda específica
- Los vehículos tienen una capacidad limitada

Objetivo: Minimizar la distancia total recorrida mientras se satisfacen todas las demandas.

Búsqueda Tabú

La Búsqueda Tabú ayuda a resolver el VRP al:

- Comenzar con una solución inicial y mejorarla iterativamente
- Usar estructuras de memoria (lista tabú) para evitar regresar a soluciones recientes
- Permitir la exploración de soluciones mediante:
  - Intercambios de clientes dentro de la misma ruta
  - Intercambios de clientes entre diferentes rutas

Búsqueda Tabú

Acepta movimientos que:

- Mejoren la solución y no estén marcados como tabú
- Estén marcados como tabú pero superen la mejor solución conocida (criterio de aspiración)

Formulación del problema

Dados los parámetros:

- V: conjunto de vehículos
- N: conjunto de clientes
- c_{ij}: distancia/costo entre los nodos i y j (en matriz de distancias)
- d_i: demanda del cliente i (en lista de demandas)
- Q: capacidad del vehículo

Variables de decisión:

- x_{ijk}: variable binaria que indica si el vehículo k viaja del nodo i al nodo j
- y_{ik}: variable binaria que indica si el cliente i es atendido

Función objetivo:

$$\text{Minimizar} \sum_{k \in V} \sum_{i \in N} c_{ij} x_{ijk}$$

Formulación del problema

Restricciones:

- Capacidad del vehículo: $\sum_{i \in N} d_i y_{ik} \leq Q, \forall k \in V$
- Cada cliente debe visitarse exactamente una vez: $\sum_{k \in V} y_{ik} = 1, \forall i \in N$
- Continuidad de ruta $\sum_{j \in N} x_{ijk} = y_{ik}, \forall i \in N, \forall k \in V$
- Todos los vehículos comienzan/terminan en el depósito (nodo 0)

Implementación

1. La solución se representa por una lista de rutas

   Ejemplo: $[[0, 1, 2, 0], [0, 3, 4, 0]]$ representa 2 rutas para 4 clientes
2. Movimiento (estructura de la vecindad)
   - Intercambiar dos clientes de la misma ruta: $[0, 1, 2, 3, 0] \rightarrow [0, 1, 3, 2, 0]$
   - Intercambiar clientes de distintas rutas:
     $[0, 1, 2, 0], [0, 3, 4, 0] \rightarrow [0, 1, 4, 0], [0, 3, 2, 0]$
   - Mover a un cliente desde una ruta a otra:
     $[0, 1, 2, 0], [0, 3, 4, 0] \rightarrow [0, 1, 0], [0, 3, 4, 2, 0]$
3. Duración tabu: 20 iteraciones (par de clientes prohibidos)

Implementación

1. Criterio de aspiración: se permite un movimiento si lleva a una mejor solución que el mejor global
2. Estrategia de búsqueda:
   - Solución inicial: greedy
   - Proceso de iteración
     - Se generan los posibles movimientos en la vecindad
     - Se evalúan todos los movimientos permitidos (o que cumplen con el criterio de aspiración)
     - Se selecciona el mejor movimiento permitido
     - Se actualiza la lista tabu y de duración
     - Actualizar la mejor solución si mejora
3. Se verifica que se cumplan las capacidades de rutas y de vehículos

En general

1. Generación de todos los posibles movimientos para la solución actual
2. Usa memoria a corto plazo (lista tabu) para evitar ciclos
3. El criterio de aspiración permite movimientos tabu si estos llevan a una mejor solución
4. Continuamente chequea las restricciones de capacidad
5. Termina si no hay mejores movimientos
6. Movimientos interruta e intrarruta

Resultados

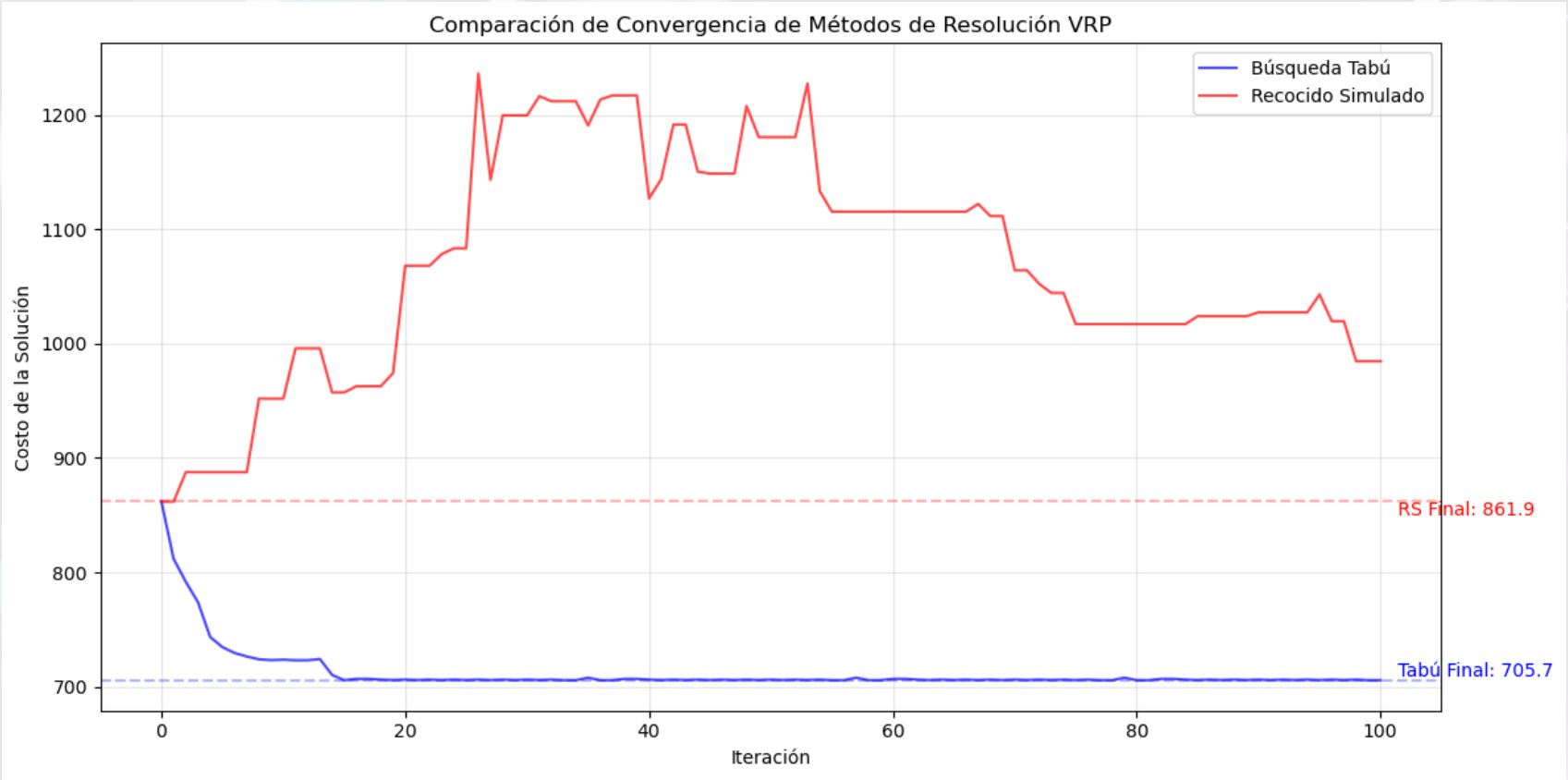


Figura: Comparación de costos por método

Resultados

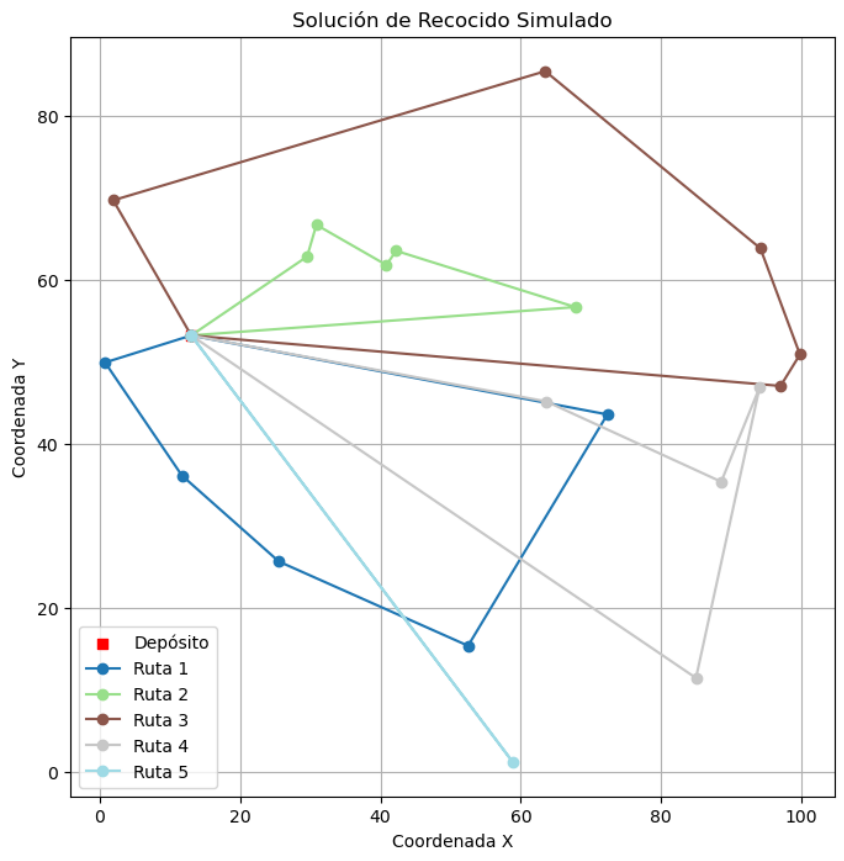


Figura: Solución de Recocido Simulado

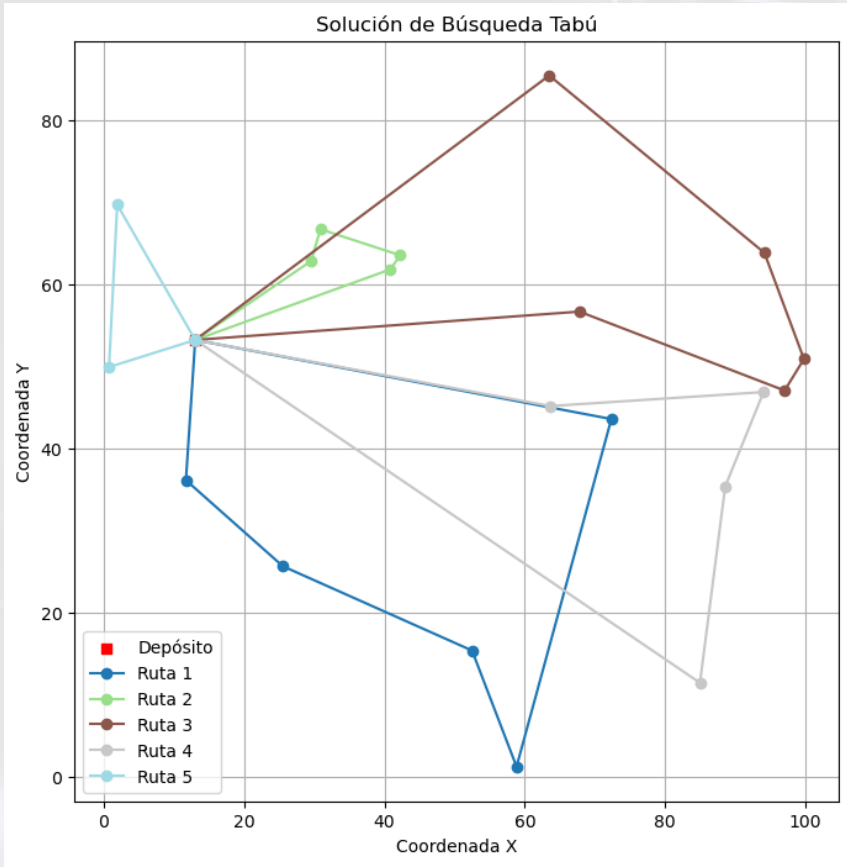


Figura: Solución de Búsqueda Tabú

Estadísticas de Demanda

- Demanda total: **404.36**
- Demanda promedio por cliente: **20.22**
- Demanda mínima: **11.90**
- Demanda máxima: **27.78**

Comparación de Resultados

Método	Costo	Tiempo (s)	Rutas
Búsqueda Tabú	705.71	1.73	5
Recocido Simulado	861.89	0.01	5

Rutas de Búsqueda Tabú

Ruta 1: [0, 3, 7, 18, 10, 11, 0]
Demanda: 90.8, Costo: 183.4

Ruta 2: [0, 1, 20, 16, 14, 0]
Demanda: 79.1, Costo: 66.3

Ruta 3: [0, 13, 9, 15, 2, 12, 0]
Demanda: 98.7, Costo: 201.8

Ruta 4: [0, 4, 6, 19, 17, 0]
Demanda: 92.4, Costo: 201.8

Ruta 5: [0, 8, 5, 0]
Demanda: 43.4, Costo: 52.4

Rutas de Recocido Simulado

Ruta 1: [0, 8, 3, 7, 18, 11, 0]
Demanda: 99.3, Costo: 171.1

Ruta 2: [0, 14, 16, 1, 20, 12, 0]
Demanda: 94.3, Costo: 117.8

Ruta 3: [0, 5, 13, 9, 15, 2, 0]
Demanda: 99.1, Costo: 224.0

Ruta 4: [0, 17, 6, 19, 4, 0]
Demanda: 92.4, Costo: 210.5

Ruta 5: [0, 10, 0]
Demanda: 19.2, Costo: 138.6

Gracias por su atención

Referencias I

- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence.
- Glover, F. (1989). Tabu Search - Part I.
- Gendreau, M. (2003). An Introduction to Tabu Search.