Universidad del Valle de Guatemala

Búsqueda Tabú aplicada al Problema de Enrutamiento de Vehículos

Jorge Andrino[1,†] **y Nicolle Escobar**[2]

[1] *and21616@uvg.edu.gt*
[2] *esc20647@uvg.edu.gt*

25 de noviembre de 2024

Resumen

La búsqueda tabú es un método de optimización metaheurística desarrollado por Fred Glover en 1986, que destaca por su uso de memoria adaptativa para mejorar la búsqueda local y evitar óptimos locales. Este trabajo presenta un análisis de sus fundamentos y la implementación práctica de la búsqueda tabú, con énfasis en su aplicación al problema de enrutamiento de vehículos (VRP). Se exploran las estructuras de memoria, criterios de aspiración, y las estrategias de intensificación y diversificación que hacen de este método una herramienta útil para la optimización combinatoria.

Keywords: *búsqueda tabú, optimización, lista tabú*

1. Introducción

La búsqueda tabú es un algoritmo de optimización desarrollado por Glover (1986) con el objetivo de resolver problemas combinatorios en donde fallan los métodos tradicionales al tener un gran espacio de búsqueda o mínimos locales.

Se basa en la premisa de que la resolución de problemas, para calificar como inteligente, debe incorporar memoria adaptativa y exploración receptiva.

La búsqueda tabú se distingue de otros métodos de optimización por su capacidad de escapar de óptimos locales mediante el uso estratégico de memoria. Esta memoria permite al algoritmo aprender de la historia de la búsqueda y guiar la exploración hacia mejores regiones del espacio de soluciones. A diferencia de métodos como el recocido simulado, que utiliza aleatorización para evitar óptimos locales, la búsqueda tabú emplea un enfoque adaptativo.

2. Marco Conceptual

Sea un problema de optimización definido como:

$$\begin{aligned} &\text{Minimizar/Maximizar } f(x) \\ &\text{sujeto a: } x \in X \end{aligned} \tag{1}$$

Donde:

- $f : X \to \mathbb{R}$ es la función objetivo
- X es el espacio de soluciones factibles
- x es una solución candidata

2.1. Forma general del problema de optimización

Sea un problema de optimización definido en un espacio de búsqueda X y una función objetivo $f : X \to \mathbb{R}$. La búsqueda tabú opera sobre el par (X, f) donde:

$$\begin{aligned} &\text{optimizar } f(x) \\ &\text{sujeto a } x \in X \\ &\qquad g_i(x) \leq 0, \quad i = 1, \dots, m \\ &\qquad h_j(x) = 0, \quad j = 1, \dots, p \end{aligned} \tag{2}$$

donde g_i y h_j representan las restricciones de desigualdad e igualdad respectivamente.

2.2. Estructura topológica del espacio de búsqueda

Para cada solución $x \in X$, definimos su vecindad como una función $N : X \to \mathcal{P}(X)$:

$$N(x) = \{x' \in X : d(x, x') \leq \epsilon\} \tag{3}$$

donde $d : X \times X \to \mathbb{R}^+$ es una métrica apropiada para el espacio de soluciones, y $\epsilon > 0$ es un parámetro de vecindad.

La definición de la estructura topológica es importante para la convergencia del algoritmo, ya que determina cómo se pueden modificar las soluciones y qué soluciones son alcanzables desde un punto dado. La elección adecuada de la métrica d y el parámetro ϵ influye directamente en el tamaño y la calidad del vecindario explorado. Una métrica bien diseñada debe reflejar la estructura natural del problema y permitir transiciones suaves entre soluciones cercanas.

2.3. Mecanismos de Búsqueda

Los mecanismos de búsqueda combinan elementos de exploración local y global para lograr un balance entre la intensificación en regiones prometedoras y la diversificación hacia áreas inexploradas del espacio de soluciones. La exploración local permite mejorar soluciones existentes, mientras que la búsqueda global ayuda a escapar de óptimos locales y descubrir nuevas regiones.

2.3.1. Exploración local

Sea x_k la solución en la iteración k. La exploración local se define mediante el operador:

$$x_{k+1} = \underset{x' \in N(x_k) \setminus T_k}{\arg\min} \{f(x')\} \tag{4}$$

donde T_k representa el conjunto de soluciones tabú en la iteración k.

2.3.2. Búsqueda global

La búsqueda global incorpora mecanismos de diversificación e intensificación mediante una función de evaluación modificada:

$$\Phi(x) = f(x) + \alpha P(x) + \beta D(x) \tag{5}$$

donde:

- $P(x)$ es una función de penalización por movimientos tabú
- $D(x)$ es una medida de diversificación
- α, β son parámetros de control

2.4. Clasificación de Movimientos

2.4.1. Movimientos de Mejora

Un movimiento $x \to x'$ se considera de mejora si y solo si:

$$f(x') < f(x) - \epsilon \tag{6}$$

donde $\epsilon > 0$ es un umbral de mejora.

2.4.2. Movimientos No Mejoradores

Un movimiento $x \to x'$ se clasifica como no mejorador cuando:

$$f(x') \geq f(x) - \epsilon \text{ y } x' \notin T_k \tag{7}$$

2.4.3. Movimientos Tabú

Un movimiento $x \to x'$ es tabú si $x' \in T_k$, excepto si satisface el criterio de aspiración:

$$f(x') < f(x^*) - \delta \tag{8}$$

donde x^* es la mejor solución encontrada y $\delta > 0$ es el umbral de aspiración.

2.5. Estructuras de Memoria

2.5.1. Memoria a Corto Plazo

La lista tabú T_k en la iteración k se define como:

$$T_k = \{m_{k-\tau+1}, \ldots, m_k\} \tag{9}$$

donde:

- $\tau = \lceil \sqrt{n} \rceil$ es el tamaño de la lista
- m_i representa el i-ésimo movimiento realizado
- n es el tamaño del problema

2.5.2. Memoria a Largo Plazo

La memoria a largo plazo se implementa mediante una matriz de frecuencia F:

$$F_{ij} = \frac{\text{número de veces que } i \to j \text{ ha sido utilizado}}{\text{número total de iteraciones}} \tag{10}$$

2.6. Métricas de Distancia

Para diferentes espacios de solución, se definen las siguientes métricas:

$$d(x,y) = \begin{cases} \sum_{i=1}^{n} |x_i - y_i| & \text{para vectores binarios (Hamming)} \\ \sum_{i<j} \mathbb{1}[(x_i - x_j)(y_i - y_j) < 0] & \text{para permutaciones (Kendall)} \\ \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} & \text{para espacios continuos (Euclidiana)} \end{cases} \tag{11}$$

2.7. Criterios de Parada

El algoritmo termina cuando se cumple alguna de las siguientes condiciones:

1. Número máximo de iteraciones: $k > k_{\text{máx}}$
2. Tiempo límite: $t > t_{\text{máx}}$
3. Estancamiento: $|f(x_k) - f(x_{k-\omega})| < \epsilon$ para algún $\omega > 0$
4. Convergencia: $\|\nabla f(x_k)\| < \epsilon$ (para espacios continuos)

Los criterios de parada garantizan el manejo de los recursos computacionales y la calidad de las soluciones obtenidas. El número máximo de iteraciones proporciona un límite superior al tiempo de ejecución, mientras que los criterios basados en estancamiento y convergencia permiten detectar cuándo la búsqueda ha alcanzado un punto donde es poco probable obtener mejoras adicionales.

2.8. Estructura de vecindad

Para cada solución $x \in X$, definimos su vecindad $N(x)$ como:

$$N(x) = \{x' \in X : x' \text{ es alcanzable desde } x\} \tag{12}$$

La estructura de vecindad debe satisfacer las siguientes propiedades:

- Conectividad: Debe existir un camino entre cualquier par de soluciones
- Simetría: Si $x' \in N(x)$, entonces $x \in N(x')$

3. Estructuras de memoria

3.1. Memoria a corto plazo

La memoria a corto plazo se implementa mediante una lista tabú T que almacena los últimos k movimientos realizados:

$$T = \{m_1, m_2, ..., m_k\} \tag{13}$$

El tamaño óptimo de la lista tabú típicamente se determina como:

$$k = \lceil \sqrt{n} \rceil \tag{14}$$

donde n es el tamaño del problema.

3.2. Criterio de aspiración

El criterio de aspiración $A(s)$ permite aceptar un movimiento tabú si:

$$f(x') < f(x_{best}) \tag{15}$$

donde x_{best} es la mejor solución encontrada hasta el momento.

4. Algoritmo

4.1. Pseudocódigo

Sea x una solución, $f(x)$ la función objetivo, T la lista tabú, y $N(x)$ el vecindario de x.

Procedimiento: BúsquedaTabú(x_0, k_{max}, τ)
Entrada:

- x_0: Solución inicial
- k_{max}: Número máximo de iteraciones
- τ: Tamaño de la lista tabú

Inicio:

1. $x_{mejor} \leftarrow x_0$ ▷ Mejor solución encontrada
2. $x_{actual} \leftarrow x_0$ ▷ Solución actual
3. $T \leftarrow \emptyset$ ▷ Lista tabú inicial vacía
4. $k \leftarrow 0$ ▷ Contador de iteraciones
5. **mientras** $k < k_{max}$ **hacer**:
   a) $x_{candidata} \leftarrow$ MejorVecinoPermitido(x_{actual}, T)
   b) **si** $f(x_{candidata}) < f(x_{mejor})$ **entonces**:
      1) $x_{mejor} \leftarrow x_{candidata}$
   c) $x_{actual} \leftarrow x_{candidata}$
   d) ActualizarListaTabú(T, x_{actual}, τ)
   e) $k \leftarrow k + 1$
6. **retornar** x_{mejor}

4.2. Procedimientos auxiliares

Procedimiento: MejorVecinoPermitido(x, T)
Inicio:

1. $mejor\_valor \leftarrow \infty$
2. $mejor\_vecino \leftarrow null$
3. **para cada** $x' \in N(x)$ **hacer**:
   a) **si** $(x' \notin T$ **O** CumpleAspiración$(x'))$ **Y** $f(x') < mejor\_valor$ **entonces**:
      1) $mejor\_valor \leftarrow f(x')$
      2) $mejor\_vecino \leftarrow x'$
4. **retornar** $mejor\_vecino$

5. Aplicación al Problema de Enrutamiento de Vehículos

5.1. Formulación de VRP

El problema de enrutamiento de vehículos se puede formular de la siguiente manera:

$$\text{Minimizar} \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{16}$$

sujeto a:

$$\sum_{i \in N} d_i y_{ik} \leq Q, \qquad \forall k \in V \text{ (Capacidad)} \tag{17}$$

$$\sum_{k \in V} y_{ik} = 1, \qquad \forall i \in N \text{ (Asignación única)} \tag{18}$$

$$\sum_{j \in N} x_{ijk} = y_{ik}, \qquad \forall i \in N, k \in V \text{ (Continuidad)} \tag{19}$$

donde:

- V: conjunto de vehículos
- N: conjunto de clientes
- c_{ij}: distancia entre los nodos i y j
- d_i: demanda del cliente i
- Q: capacidad del vehículo
- x_{ijk}: variable binaria que indica si el vehículo k viaja de i a j
- y_{ik}: variable binaria que indica si el cliente i es atendido por el vehículo k

El VRP es un problema NP-hard lo que significa que su complejidad crece exponencialmente con el tamaño del problema.

5.2. Implementación

En la implementación para el VRP se utilizaron las siguientes estructuras y operadores:

5.2.1. Representación de soluciones

Una solución se representa como un conjunto de rutas:

$$S = \{R_1, R_2, ..., R_m\} \tag{20}$$

donde cada ruta R_k es una secuencia de clientes:

$$R_k = (0, i_1, i_2, ..., i_{n_k}, 0) \tag{21}$$

5.2.2. Operadores de movimiento

Se definieron tres tipos principales de movimientos:
1. Intercambio intra-ruta: Este intercambia la posición de dos clientes dentro de la misma ruta:

$$(...i, j, ...) \rightarrow (...j, i, ...) \tag{22}$$

2. Intercambio inter-ruta: Este intercambia clientes entre dos rutas diferentes:

$$\begin{aligned} R_1 &: (...i...) \rightarrow (...j...) \\ R_2 &: (...j...) \rightarrow (...i...) \end{aligned} \tag{23}$$

3. Inserción: Este operador mueve un cliente de una ruta a otra, insertándolo en la mejor posición posible:

$$\begin{aligned} R_1 &: (...i...) \rightarrow (...) \\ R_2 &: (...) \rightarrow (...i...) \end{aligned} \tag{24}$$

5.3. Lista Tabú

La lista tabú se gestiona dinámicamente durante la búsqueda con las siguientes características:

- **Estructura**: Lista de movimientos recientes representados como pares de clientes
- **Tamaño**: Determinado dinámicamente como $\tau = \lceil \sqrt{n} \rceil$
- **Actualización**: FIFO (First In, First Out) con tenure variable
- **Criterio de aspiración**: Permite movimientos tabú si mejoran la mejor solución conocida

6. Procedimiento

La generación de la solución inicial emplea un algoritmo basándose en el vecino más cercano en consideración de las restricciones de capacidad. Este proceso sigue una estrategia en donde se construyen las rutas secuencialmente, comenzando desde el depósito (nodo 0) y agregando en cada paso el cliente no asignado más cercano que cumpla la restricción de capacidad del vehículo.

El procedimiento comienza con un conjunto de clientes no asignados y va construyendo rutas hasta que todos los clientes son asignados o se alcanza el número máximo de vehículos disponibles. Para cada ruta:

Información

1. Se inicia en el depósito (nodo 0)
2. Se selecciona iterativamente el cliente más cercano al último nodo visitado
3. Se verifica la factibilidad respecto a la capacidad del vehículo
4. Se continúa hasta que no sea posible agregar más clientes
5. Se cierra la ruta regresando al depósito

La implementación utiliza las siguientes estructuras de datos principales:

Información

- **Matriz de distancias**: Almacena las distancias entre todos los pares de nodos, incluyendo el depósito.
- **Vector de demandas**: Contiene la demanda de cada cliente, con el depósito teniendo demanda cero.
- **Lista tabú**: Implementada como una lista de pares de clientes (i, j) que han sido intercambiados recientemente.
- **Solución**: Representada como una lista de rutas, donde cada ruta es una secuencia de clientes comenzando y terminando en el depósito.

Clase: VRPSolver
Atributos:

- *distances*: Matriz de distancias entre nodos
- *demands*: Lista de demandas de clientes
- *vehicle_capacity*: Capacidad de cada vehículo
- *num_vehicles*: Número de vehículos disponibles
- *num_customers*: Número de clientes
- *max_iterations*: Número máximo de iteraciones

Procedimiento: solve_tabu_search(*tabu_tenure*)
Inicio:

1. *current_solution* ← generate_initial_solution()
2. *best_solution* ← *current_solution*
3. *tabu_list* ← ∅
4. **para** *iteration* ← 1 **hasta** *max_iterations* **hacer**:
   a) *moves* ← get_neighborhood_moves(*current_solution*)
   b) *best_move* ← null
   c) *best_cost* ← ∞
   d) **para cada** *move* **en** *moves* **hacer**:
      1) *neighbor* ← apply_move(*current_solution*, *move*)
      2) **si** is_feasible(*neighbor*) **entonces**:
         a′ *cost* ← calculate_total_cost(*neighbor*)
         b′ **si** (*move* ∉ *tabu_list* **o** *cost* < *best_cost*) **entonces**:
         c′ *best_move* ← *move*
         d′ *best_cost* ← *cost*
   e) **si** *best_move* = null **entonces**: **terminar**
   f) Actualizar *current_solution* y *tabu_list*
   g) Actualizar *best_solution* si corresponde
5. **retornar** *best_solution*

Procedimiento: generate_initial_solution()
Inicio:

1. *unassigned* ← [1, ..., *num_customers*]
2. *routes* ← []
3. **mientras** *unassigned* ≠ ∅ **hacer**:
   a) Iniciar nueva ruta con depósito
   b) **mientras** existan clientes factibles **hacer**:
      1) Seleccionar cliente más cercano que cumpla restricción de capacidad
      2) Agregar cliente a la ruta actual
   c) Cerrar ruta en depósito
   d) **si** número de rutas = *num_vehicles* **entonces**:
      1) Asignar clientes restantes a última ruta
      2) **terminar**
4. **retornar** *routes*

Procedimiento: is_feasible(*solution*)
Inicio:

1. **para cada** *route* **en** *solution* **hacer**:
   a) *total_demand* ← ∑ demandas de clientes en *route*
   b) **si** *total_demand* > *vehicle_capacity* **entonces**:
      1) **retornar** falso
2. **retornar** verdadero

6.1. Evaluación de Soluciones

En la evaluación de soluciones se consideraron los siguientes aspectos:

- **Costo total**: Suma de las distancias recorridas por todos los vehículos
- **Factibilidad**: Verificación de restricciones de capacidad y número de vehículos
- **Eficiencia**: Cálculo incremental de cambios en la función objetivo

7. Resultados

7.1. Configuración de parámetros

Los parámetros usados para este problema fueron los siguientes:

- Tamaño de lista tabú: $\tau = 20$
- Criterio de parada: $k_{max} = 100$ iteraciones
- Instancias: Generadas aleatoriamente con n clientes en área 100×100
- Capacidad de vehículos: $Q = 100$
- Demandas: Distribuidas uniformemente entre $[10, 30]$
- Área de distribución: 100×100 unidades

7.2. Comparación con Recocido Simulado

El recocido simulado se implementó con los siguientes parámetros:

- **Temperatura inicial**: $T_0 = 1000$
- **Tasa de enfriamiento**: $\alpha = 0{,}95$
- **Número de iteraciones**: $k_{max} = 1000$

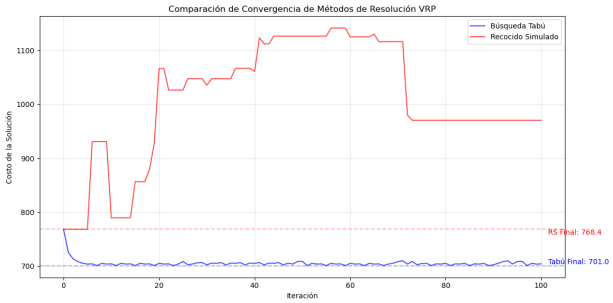

Figura 1. Comparación de convergencia de Métodos de Resolución para VRP

Los resultados muestran que la búsqueda tabú supera al recocido simulado por su calidad de solución:

Método	Costo	Tiempo (s)	Rutas
Búsqueda Tabú	701.02	2.49	4
Recocido Simulado	768.35	0.02	4

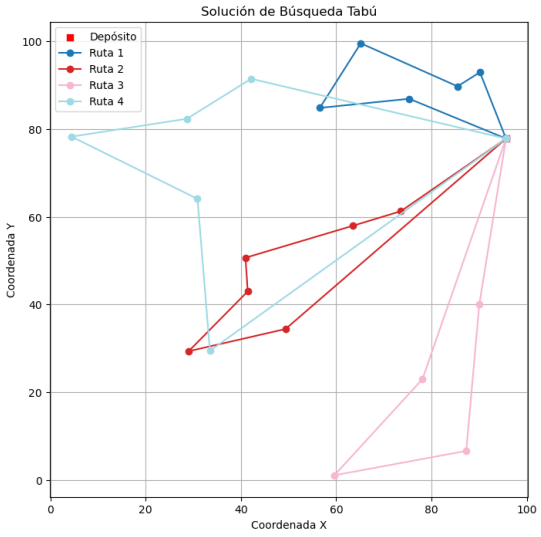

Figura 2. Solución de Búsqueda Tabú

Cuadro 1. Rutas obtenidas mediante Búsqueda Tabú

Ruta	Secuencia	Demanda	Costo
1	[0, 10, 18, 15, 3, 19, 0]	97.6	102.6
2	[0, 7, 4, 20, 9, 1, 16, 0]	97.9	172.5
3	[0, 5, 11, 13, 12, 0]	80.8	186.3
4	[0, 6, 8, 17, 2, 14, 0]	94.5	239.6

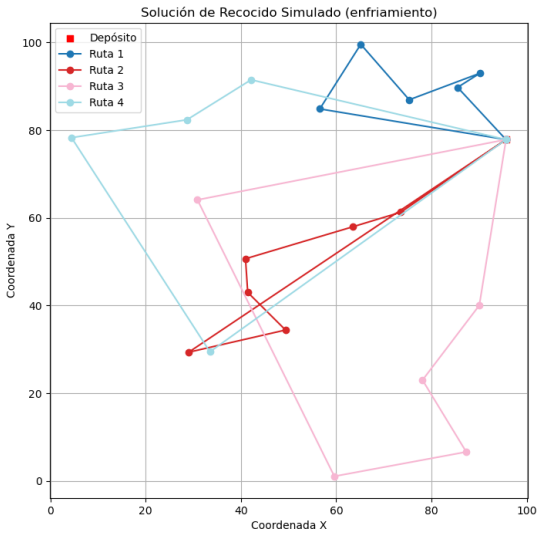

Figura 3. Solución de Recocido Simulado

Cuadro 2. Rutas obtenidas mediante Recocido Simulado

Ruta	Secuencia	Demanda	Costo
1	[0, 18, 10, 19, 15, 3, 0]	97.6	110.4
2	[0, 7, 4, 20, 9, 16, 1, 0]	97.9	184.7
3	[0, 5, 12, 11, 13, 2, 0]	93.4	241.6
4	[0, 6, 8, 17, 14, 0]	82.0	231.7

7.3. Análisis de demanda

Las estadísticas de demanda muestran:

- Demanda total: 370.89
- Demanda promedio por cliente: 18.54
- Demanda mínima: 10.62
- Demanda máxima: 28.85

8. Discusión

Se implementaron y compararon dos metaheurísticas usadas en el Problema de Ruteo de Vehículos (VRP): la Búsqueda Tabú y el Recocido Simulado. Los resultados que se obtuvieron muestran diferencias en la búsqueda de mínimos locales y la eficiencia computacional.

La Búsqueda Tabú demostró mejores resultados al optimizar costos, alcanzando una solución con un costo total de 701.02 unidades, siendo un 9,6 % menor que la solución dada por el Recocido Simulado (768.35 unidades). Además, la Búsqueda Tabú denotó un comportamiento más estable y consistente en la búsqueda de soluciones óptimas, mientras que el Recocido Simulado exhibió fluctuaciones pronunciadas por sus fundamentos probabilísticos.

Ambos métodos demostraron eficiencia al emplear 4 vehículos de los 6 disponibles. La Búsqueda Tabú también logró una mejor distribución de costos entre rutas.

9. Conclusiones

- En términos de eficiencia de recursos, el Recocido Simulado destacó por su velocidad de ejecución (0.02 segundos) en comparación con la Búsqueda Tabú (2.49 segundos), siendo aproximadamente 124 veces más rápido. Sin embargo, esta ventaja en velocidad viene con un costo en la calidad de la solución.
- La diferencia en el costo total entre ambos métodos sugiere que la estrategia de memoria adaptativa de la Búsqueda Tabú es más efectiva para este tipo de problemas de ruteo que el enfoque probabilístico del Recocido Simulado.

Referencias

[1] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533–549.

[2] Glover, F. (1989). Tabu Search - Part I. *ORSA Journal on Computing*, 1(3), 190–206.

[3] Gendreau, M. (2003). An Introduction to Tabu Search. *Handbook of Metaheuristics*, 37–54.

[4] Glover, F., & Melián Batista, B. (2006). *Introducción a la Búsqueda Tabú*. Retrieved from https://leeds-faculty.colorado.edu/glover/fred%20pubs/329%20-%20Introduccion%20a%20la%20Busqueda%20Tabu%20TS_Spanish%20w%20Belen(11-9-06).pdf

[5] Universidad de Sonora. (2006). *Capítulo 2*. Retrieved from http://tesis.uson.mx/digital/tesis/docs/18920/Capitulo2.pdf

[6] Morales Manzanares, E. (2004). *5.5 The Reactive Tabu Search*. Retrieved from https://ccc.inaoep.mx/~emorales/Cursos/Busqueda/node72.html

[7] Kuo, M. (2024). *Solving the Vehicle Routing Problem*. Retrieved from https://www.routific.com/blog/what-is-the-vehicle-routing-problem

[8] Göbel, N. (2022). *The Vehicle Routing Problem Explained*. Retrieved from https://conundra.eu/blog/the-vehicle-routing-problem-explained