GSO-RL para el problema multinivel de rutas por arcos con capacidad e instalaciones intermedias en la recogida de residuos

Juan Aday Siverio-González[1], Belén Melián-Batista[1], and J. Marcos Moreno-Vega[1]

Departamento de Ingeniería Informática y de Sistemas,
Instituto Universitario de Desarrollo Regional
Universidad de La Laguna, San Cristóbal de La Laguna, España
{alu0101503950, mbmelian, jmmoreno}@ull.edu.es

Resumen La recogida de residuos en entornos urbanos densos exige coordinar flotas heterogéneas de vehículos que operan en niveles distintos. Abordamos el problema multinivel de rutas por arcos con capacidad e instalaciones intermedias (ML-CARPIF), en el que vehículos manuales (VM) de baja capacidad recogen residuos de demanda menor y los depositan en cabañas, y vehículos compactadores (VC) de alta capacidad atienden tanto arcos de mayor demanda como las propias cabañas, descargando finalmente en estaciones de transferencia. Dado que la demanda de cada cabaña depende de las rutas de los VM, la optimización separada de ambos niveles conduce a soluciones subóptimas, por lo que se requiere una optimización integrada. Aplicamos GSO-RL (*Guided Strategic Oscillation with Reinforcement Learning*), una metaheurística que extiende el paradigma de la Oscilación Estratégica mediante una memoria adaptativa basada en regiones (GCM): agentes *Multi-Armed Bandit* independientes aprenden, a partir del historial de rendimiento acumulado en cada región, qué estrategia de búsqueda aplicar, intensificación fina, exploración equilibrada o diversificación amplia, regulando de forma adaptativa la intensidad de perturbación y la temperatura de aceptación del motor de búsqueda. Las regiones se definen en torno a soluciones guía que evolucionan a lo largo de la búsqueda. Los experimentos sobre 90 instancias derivadas del benchmark VAL muestran que GSO-RL iguala o mejora la cota proporcionada por CPLEX en el 69 % de los casos con un tiempo de cómputo doce veces inferior, obteniendo un gap medio de $-0{,}35\,\%$.

Keywords: Rutas por arcos · Recogida de residuos · Oscilación estratégica · Aprendizaje por refuerzo · Multi-Armed Bandit · Instalaciones intermedias

1. Introducción

La recogida de residuos urbanos es una tarea logística compleja, especialmente en ciudades de elevada densidad poblacional y redes viarias intrincadas.

Motivados por sistemas reales de recogida en dos niveles [2,3], consideramos una flota heterogénea formada por vehículos manuales (VM), de baja capacidad, que recogen residuos de pequeño volumen y descargan en instalaciones intermedias llamadas *cabañas*, y vehículos compactadores (VC), de alta capacidad, que atienden residuos de mayor volumen y recogen los residuos consolidados en las cabañas antes de descargar en estaciones de transferencia.

La característica que distingue este sistema es la *interacción entre niveles*: la demanda de cada cabaña no es un dato fijo, sino que depende de las rutas de los VM. Una optimización secuencial que fije primero el nivel VM sesga la búsqueda hacia óptimos locales del primer nivel [2]; la optimización integrada de ambos niveles evita este fenómeno y reduce el coste total.

Estudiamos el problema multinivel de rutas por arcos con capacidad e instalaciones intermedias (ML-CARPIF), variante NP-dura del *Capacitated Arc Routing Problem* (CARP) clásico [4]. El único algoritmo exacto disponible [3] tiene su aplicabilidad restringida a instancias de pequeña escala, lo que pone de manifiesto la necesidad de métodos heurísticos eficientes.

En este trabajo aplicamos GSO-RL (*Guided Strategic Oscillation with Reinforcement Learning*) [1], una metaheurística con memoria adaptativa basada en regiones que asigna a cada una un *Multi-Armed Bandit Agent* (MABA) para aprender qué estrategia de búsqueda resulta más eficaz en ese contexto local. Las contribuciones principales son:

- Adaptación de GSO-RL al ML-CARPIF con optimización integrada VM-VC: dado que toda modificación del nivel VM invalida el plan VC, el algoritmo usa soluciones completas (VM+VC) como puntos de referencia, evitando el sesgo hacia óptimos locales del primer nivel.
- Arquitectura de doble capa de aprendizaje: un agente MABA por región aprende qué intensidad de perturbación es más eficaz en cada zona del espacio de búsqueda, mientras un componente global compartido aprende qué operadores de destrucción resultan más productivos a lo largo de toda la búsqueda.
- Validación sobre un conjunto de instancias de la literatura (VAL): GSO-RL iguala o mejora la cota de CPLEX en el 69 % de los casos con un tiempo de cómputo 12 veces inferior, obteniendo un gap medio de $-0{,}35\,\%$ en el valor de la función objetivo.

2. Trabajo relacionado

ML-CARPIF y su contexto. ML-CARPIF pertenece a la familia de problemas de rutas por arcos con instalaciones intermedias [7]. Fue propuesto por Wei et al. [2], quienes formularon un modelo de programación lineal entera mixta y lo resolvieron mediante una búsqueda adaptativa de gran vecindario (ALNS) con optimización secuencial de niveles; el mismo grupo presentó posteriormente el primer algoritmo exacto basado en descomposición de Benders lógica y *branch-and-price*, que alcanza soluciones óptimas para instancias de hasta 170 arcos, pero con coste computacional creciente [3]. Ninguno de estos trabajos aborda la

optimización integrada de ambos niveles mediante metaheurísticas con memoria adaptativa.

Oscilación estratégica y memoria adaptativa. La Oscilación Estratégica [8,9] gobierna las transiciones entre vecindarios navegando deliberadamente la frontera entre regiones del espacio de soluciones, complementada por la Programación de Memoria Adaptativa [8] y generalizada en el marco multi-ola de Glover [10], donde una memoria poblacional alterna entre fases de distinta naturaleza. GSO-RL [1] instancia estos principios asignando a cada región un agente de aprendizaje por refuerzo [11] que aprende qué estrategia resulta más eficaz en ese contexto local; su adaptación al ML-CARPIF, presentada en este trabajo, constituye la primera aplicación de este paradigma a problemas multinivel de rutas por arcos.

3. Descripción del problema y formulación

El ML-CARPIF se define sobre un grafo dirigido $G = (V, A)$ que representa la red viaria. Existen tres tipos de elementos requeridos: arcos de VM (A_{VM}), arcos de VC (A_{VC}) y cabañas ($V_H \subset V$), que actúan como instalaciones intermedias donde los VM descargan antes de que los VC recojan los residuos consolidados y los transporten hasta la estación de transferencia ($V_T \subset V$).

Sean K_{VM} y K_{VC} los conjuntos de vehículos con capacidades Q_{VM} y Q_{VC}; q_{ij}^{VM}, q_{ij}^{VC} las demandas y d_{ij}^{VM}, d_{ij}^{VC} los costes de recorrido de cada arco. Cada VM realiza una ruta multi-viaje entre cabañas; mientras que cada VC realiza un único viaje circular desde la estación de transferencia atendiendo arcos de A_{VC} y recogiendo las cabañas asignadas.

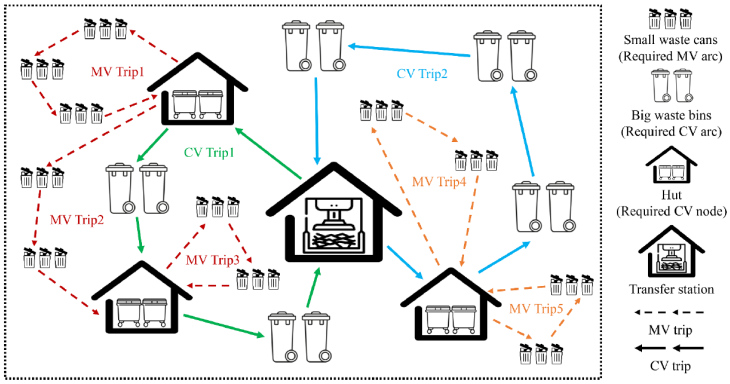


Figura 1. Sistema de recogida de residuos en dos niveles [3]. Los viajes VM (líneas punteadas) conectan arcos requeridos de pequeño volumen con las cabañas; los viajes VC (líneas sólidas) recogen los residuos consolidados en las cabañas y los transportan a la estación de transferencia central.

La característica estructural clave es que la demanda de cada cabaña $q_h = \sum_{k,t} f^{VM}_{h,k,t}$ depende de las rutas de los VM, de modo que toda modificación del nivel VM invalida el plan VC, que debe reconstruirse desde cero. El objetivo es minimizar el coste total:

$$\text{mín} \sum_{k \in K_{VM}} \sum_{t \in T_{VM}} \sum_{(i,j) \in A} y^{VM}_{ijkt}\, d^{VM}_{ij} + \sum_{k \in K_{VC}} \sum_{(i,j) \in A} y^{VC}_{ijk}\, d^{VC}_{ij}, \tag{1}$$

donde y^{VM}_{ijkt} e y^{VC}_{ijk} representan el número de veces que cada vehículo recorre el arco (i, j). El modelo completo de restricciones se detalla en [2,3]. La Figura 1 ilustra la estructura del sistema.

4. Métodos base

El ciclo de búsqueda central sigue el esquema destroy → repair → RVND-VM → RebuildVC: en cada iteración se destruye una fracción ρ de la solución VM actual, se reparan los arcos eliminados, se aplica búsqueda local sobre el nivel VM resultante y, finalmente, se reconstruye el nivel VC desde cero. Este orden es necesario en el ML-CARPIF porque cualquier modificación del nivel VM cambia las demandas de las cabañas e invalida el plan VC vigente.

El nivel VM se construye con *PathScanningVM*, una adaptación de *Path Scanning* [5] al contexto multi-nivel con cabañas: selecciona arcos mediante una lista restringida al mejor α % ($\alpha \in [0{,}01, 0{,}30]$) y elige la cabaña de descarga con *look-ahead* sobre el arco pendiente más cercano. El nivel VC se construye con *ClarkeWrightVC*, adaptación del algoritmo de ahorros [13] que trata cabañas y arcos de VC como tareas unificadas y selecciona para cada ruta la estación de transferencia de menor coste de acceso.

La búsqueda local VM usa *Random Variable Neighborhood Descent* (RVND) con cuatro entornos explorados en orden aleatorio: reubicación de arcos, intercambio inter-ruta, 2-opt y reubicación de cabañas. Tras cada destrucción, *Repair Regret Insertion* introduce los arcos eliminados priorizando aquellos cuya postergación tiene mayor impacto relativo sobre el coste final, medido como $\textit{regret}(a) = \textit{cost}_2(a) - \textit{cost}_1(a)$, donde $\textit{cost}_1(a)$ y $\textit{cost}_2(a)$ son los costes de inserción en las posiciones óptima y segunda óptima, respectivamente.

Una vez obtenido un plan VM factible, *RebuildVC* lanza múltiples arranques en paralelo con α_{VC} aleatorizado, utilizando **8** hilos, cada uno ejecutando *Clarke & Wright* → RVND-VC → VNS-VC, y conserva la mejor solución factible encontrada.

En este trabajo, un ALNS-VM orquesta este ciclo seleccionando en cada iteración uno de los cuatro operadores de destrucción mediante ruleta proporcional sobre pesos adaptativos:

- *Destroy Proximity*: elimina los arcos geográficamente más cercanos a un arco semilla elegido al azar, concentrando la reparación en zonas locales del grafo.
- *Destroy VC Pain*: elimina los arcos VM cuya asignación de cabaña provoca mayor incremento en el coste VC, introduciendo coordinación explícita entre

niveles. Es el único operador que evalúa el impacto cruzado VM→VC en la fase de destrucción.

- *Destroy Huts Assignments*: redistribuye la carga entre cabañas mediante intercambios de asignación entre *trips* adyacentes, explorando el espacio de configuraciones de descarga sin alterar la secuencia de recogida.
- *Destroy Worst Arcs*: elimina los arcos con mayor coste relativo de desvío respecto a la ruta directa, priorizando los tramos menos eficientes de la solución actual.

La tasa de destrucción ρ y la temperatura de aceptación T no son parámetros fijos: los determina el componente de aprendizaje por regiones de GSO-RL en cada llamada al motor ALNS-VM, como se describe en la Sección 5.

5. Oscilación Estratégica Guiada con Aprendizaje por Refuerzo (GSO-RL)

5.1. Descripción y principios teóricos

GSO-RL [1] es una metaheurística con memoria adaptativa que avanza el paradigma de la Oscilación Estratégica [8,9] sustituyendo las reglas de transición predefinidas por agentes de aprendizaje por refuerzo independientes que aprenden políticas de búsqueda específicas para cada región del espacio de soluciones. En su formulación original, aplicada al SCC-VRP [1], cada agente selecciona entre tres macro-movimientos completos —construcción e intensificación, intensificación total y búsqueda tabú con múltiples entornos— evaluados con recompensa diferida a través de una *Current Context Memory* (CCM) explícita.

La adaptación al ML-CARPIF preserva los principios arquitectónicos pero introduce cuatro modificaciones motivadas por la naturaleza bimodal del problema, que se detallan en la Sección 5.3.

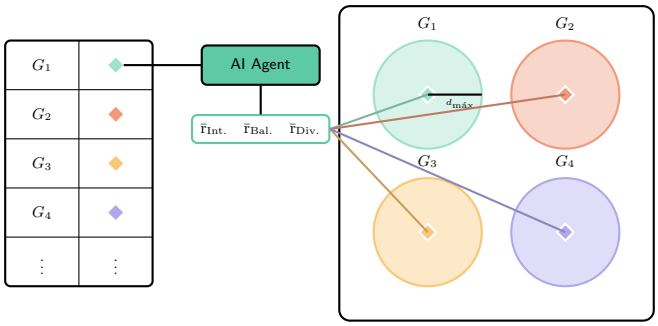


Figura 2. Estructura de la GCM en GSO-RL [1], adaptada al ML-CARPIF. *Izquierda*: tabla de guías; el agente MABA asociado a cada guía (n_{rings}=1) acumula recompensas medias $\bar{r}$ para las tres estrategias de intensidad. *Derecha*: cada guía delimita una región de radio $d_{\text{máx}}$=0,25 en el espacio de soluciones VM+VC.

El marco se sustenta en tres principios fundacionales, ilustrados en la Fig. 2:

1. Memoria basada en guías (GCM, Fig. 2 izquierda): conjunto de $|G|$ soluciones representantes organizadas en zona élite (70 %) y de diversidad (30 %), reflejando el Principio de Atracción Persistente [10].
2. Localización por similitud: una medida de disimilitud estructural identifica la región de la solución actual (Sección 5.2); el radio $d_{\text{máx}}$ delimita cada región (Fig. 2 derecha).
3. Oscilación dirigida por aprendizaje: el agente MABA de la región más cercana selecciona mediante política ε-greedy la estrategia más adecuada para ese contexto local (Fig. 2 centro).

La clave de la adaptación al ML-CARPIF es usar soluciones completas (VM+VC) como puntos de referencia de la GCM: el coste combinado guía todas las decisiones de aceptación y aprendizaje, evitando el sesgo hacia óptimos locales del primer nivel que caracteriza a los enfoques secuenciales [2]. El Algoritmo 1 resume el flujo global.

Algorithm 1 GSO-RL para el ML-CARPIF

```
1: S* ← ∅;  t_inicio ← AHORA()
2: GCM ← INICIALIZARGCM(|G|)          ▷ |G|=10 sols. VM+VC, α∼U(0,01, 0,30)
3: while AHORA() − t_inicio < t_límite do
4:     S ← SELECCIONARGUÍA(GCM)                          ▷ puntuación UCB1, ec. (3)
5:     for r = 1 to regionSearchTime do
6:         (ρ, T) ← MABA-ESTRATEGIA(S, GCM)              ▷ Tab. 1; ε-greedy
7:         S ← ALNS-VM(S, ρ, T)                  ▷ destroy → repair → RVND-VM
8:         S ← REBUILDVC(S)                      ▷ C&W → RVND-VC → VNS-VC
9:         ACTUALIZARMABA(r, ec. (4))
10:        INTENTARINSERTARGUÍA(S, GCM)          ▷ ec. (2); criterios élite/diversidad
11:        if f(S) < f(S*) then
12:            S* ← S
13:        end if
14:    end for
15:    DECREMENTARVIDAGUÍAS(GCM)
16: end while
17: return S*
```

5.2. Pasos del algoritmo

Inicialización de la GCM (Alg. 1, línea 2) La GCM (Fig. 2, izquierda) se inicializa generando $|G|$=10 soluciones completas (VM+VC) con PATHSCANNINGVM bajo $\alpha\sim U(0{,}01, 0{,}30)$, refinadas con RVND-VM y completadas con REBUILDVC. La variación de α garantiza diversidad estructural inicial. La GCM se organiza en zona élite (70 %) y de diversidad (30 %), reflejando el Principio de Atracción Persistente [10]. A cada guía se le asigna un agente MABA con estadísticas vacías y vida *liveTime*=4 000 iteraciones.

Medida de disimilitud (Alg. 1, línea 10) La localización de una solución respecto a las guías de la Fig. 2 (derecha) requiere una medida que capture la estructura bimodal del ML-CARPIF. Se utiliza distancia de Jaccard sobre claves de adyacencia de 64 bits key(u, v)=$u\cdot 10^7+v$, que codifican conexiones cabaña–arco y adyacencias internas de cada *trip* VM y VC:

$$d(s_1, s_2) = 1 - \frac{|K_1 \cap K_2|}{|K_1 \cup K_2|}, \tag{2}$$

calculada en $O(N \log N)$ con umbral $d_{\text{máx}}$=0,25 (Fig. 2, derecha). Al incorporar claves de ambos niveles, la medida captura simultáneamente la cobertura de arcos y la distribución de carga en cabañas, extendiendo la divergencia estructural al dominio bimodal VM+VC.

Selección de guía (Alg. 1, línea 4) Se emplea una puntuación UCB1 enriquecida:

$$\text{score}(i) = \big(q_i + C \cdot u_i \cdot s_i\big) \cdot \delta_i, \tag{3}$$

donde q_i es la calidad relativa del coste VM+VC normalizada al intervalo $[0, 1]$; $u_i = \sqrt{2\ln(N+1)/(n_i+1)}$ es el bonus UCB1 [11] que favorece guías poco visitadas; $s_i = e^{-\text{streak}_i/4}$ penaliza exponencialmente las guías cuya racha de iteraciones sin mejora es elevada, desincentivando la reexploración de regiones estancadas; y $\delta_i \in \{0{,}3,\ 1{,}0\}$ aplica una penalización adicional a las guías en *dead-zone*, es decir, aquellas visitadas repetidamente sin ningún progreso. Se selecciona la guía de mayor puntuación y su solución representante se copia como solución de trabajo S.

Gestión de la GCM (Alg. 1, línea 10) Con d definida en la ec. (2), cada solución candidata se localiza calculando $G_{\text{más cercana}} = \arg\min_{G_i} d(S, G_i)$ (Fig. 2, derecha). La candidata reemplaza una guía según tres criterios de prioridad: (1) guía expirada (*liveTime*$\leq$0); (2) mejora intra-región ($d \leq d_{\text{máx}}$ y mejor coste), aplicando un decaimiento parcial del agente —se triplica ε y se reducen a la mitad los contadores de visita— en lugar del reset total del diseño original, para no descartar el historial aprendido; (3) diversificación ($d > d_{\text{máx}}$ y mejor que la guía de menor calidad). Al final de cada ciclo externo se decrementan todos los *liveTime*.

Oscilación estratégica (Alg. 1, líneas 6–9) El agente MABA de la región activa (Fig. 2, centro) selecciona mediante política ε-greedy (ε_0=0,20, mín. 0,15, decaimiento 0,98) una de las tres estrategias de la Tabla 1:

La temperatura se escala como $T \leftarrow t_0 \cdot c^2_{\text{guía}}/\text{máx}(1, c^*)$, haciendo al algoritmo independiente de la magnitud absoluta de los costes. La recompensa al término de cada bloque ALNS-VM (línea 9) sigue:

$$r = \begin{cases} r_{\text{global}} & f(S') < f(S^*) \\ r_{\text{current}} & f(S') < f(S) \\ r_{\text{SA}} & \text{aceptada por SA, } f(S') \geq f(S) \\ 0 & \text{en caso contrario,} \end{cases} \tag{4}$$

Tabla 1. Estrategias del agente MABA (Alg. 1, línea 6).

Estrategia	ρ	t_0
Intensify	0,15	0,02
Balanced	0,30	0,05
Diversify	0,50	0,10

con valores r_{global}=6, r_{current}=3 y r_{SA}=1, calibrados para priorizar las mejoras globales sobre las locales y reconocer que las transiciones al alza pueden ser beneficiosas a largo plazo. La media de cada estrategia se actualiza de forma incremental mediante la media de Welford, garantizando estabilidad numérica con coste $O(1)$ por actualización.

La doble capa de aprendizaje factoriza el aprendizaje: ALNSLearner ajusta globalmente los pesos de los operadores de destrucción (*qué destruir*), mientras el MABA de cada región controla la intensidad (ρ, t_0) (*con qué intensidad*), evitando que ambas señales se contaminen mutuamente.

5.3. Adaptaciones respecto al GSO-RL original

Respecto al diseño canónico [1], la adaptación al ML-CARPIF introduce cuatro cambios motivados por la naturaleza bimodal del problema, sin alterar los principios arquitectónicos del marco.

Espacio de acciones. En el diseño original, cada agente selecciona entre tres *macro-moves* completos que cambian cualitativamente el tipo de búsqueda: construcción de una solución nueva, intensificación iterativa y búsqueda tabú. En el ML-CARPIF, dado que toda modificación del nivel VM invalida el plan VC y obliga a su reconstrucción completa, sustituir la solución de trabajo por una de nueva construcción introduciría un coste computacional prohibitivo en cada llamada al agente. El espacio de acciones se redefine por ello como tres *estrategias de intensidad* (ρ, t_0) para el motor ALNS-VM (Tabla 1), que modulan el grado de perturbación y la temperatura de aceptación sin abandonar la solución de trabajo.

CCM implícita. La CCM explícita se sustituye por la ventana implícita del bucle ALNS, preservando el Principio de Validez Condicional Marginal [10]: el efecto de una estrategia solo se evalúa tras completar el bloque de *regionSearchTime* iteraciones, evitando señales de crédito prematuras.

Granularidad regional. La estructura de anillos concéntricos del diseño original se simplifica a n_{rings}=1 agente por guía (Fig. 2), pues el coste combinado VM+VC ya proporciona una señal diferenciadora suficiente para distinguir regiones sin necesidad de subdivisiones adicionales.

Doble capa de aprendizaje. Se añade ALNSLearner, un componente global compartido por todas las guías que aprende qué operadores de destrucción son más productivos a lo largo de toda la búsqueda. Esta segunda capa desacopla las decisiones de *qué destruir* (nivel global) de *con qué intensidad* (nivel regional),

evitando que ambas señales se contaminen mutuamente y permitiendo que cada nivel converja a su propia política óptima.

6. Experimentos computacionales

GSO-RL se implementó en C++ y se ejecutó en un equipo con procesador Intel Core i7-10700KF a 3,80 GHz y 32 GB de RAM bajo Windows 11 de 64 bits, empleando **8** hilos para los arranques paralelos de RebuildVC, con un límite de 300 segundos por instancia. Como referencia exacta se emplean las cotas reportadas en [3] para el modelo de programación matemática resuelto con CPLEX bajo un límite de 3 600 segundos; en la práctica, CPLEX agota este límite en la gran mayoría de las instancias, de modo que las cotas del modelo representan la mejor solución que el *solver* exacto fue capaz de encontrar antes de que se agotara el tiempo, no necesariamente el óptimo global.

6.1. Instancias de prueba y ajustes de parámetros

Se emplean las 90 instancias del ML-CARPIF introducidas en [3], derivadas del benchmark VAL [6] bajo tres escenarios de capacidad vehicular (1,200, 2,400 y 3,300 unidades), distribuidas en tres conjuntos de 30 (Sets 1, 2 y 3).

Los parámetros se ajustaron con *irace* [12], sobre un subconjunto de instancias con propiedades dispersas, distinto del conjunto de test, obteniendo: $|G|$=10 guías con vida *liveTime*=4 000 iteraciones, razón calidad–diversidad de 0,70, umbral de disimilitud $d_{\text{máx}}$=0,25, ε_0=0,20 con mínimo 0,15 y decaimiento 0,98, constante de exploración C=1,0 y límite de tiempo de 300 s por instancia.

6.2. Resultados

La Tabla 2 presenta los estadísticos agregados del gap relativo entre el mejor coste hallado por GSO-RL (*BEST COST*) y la cota del modelo (*COST MODELO*) sobre las 90 instancias VAL, agrupadas en tres conjuntos de 30:

$$\%\text{GAP} = \frac{\textit{BEST COST} - \textit{COST MODELO}}{\textit{COST MODELO}} \times 100. \tag{5}$$

Un valor negativo indica que GSO-RL obtiene un coste estrictamente mejor que la cota alcanzada por CPLEX en 3 600 s, esto es, en doce veces más tiempo de cómputo.

La mediana de 0,00 % en los tres conjuntos indica que el comportamiento típico de GSO-RL es igualar exactamente la cota de CPLEX: más de la mitad de las instancias presentan un gap nulo o negativo. La divergencia entre media y mediana en el Set 3 (−1,00 % frente a 0,00 %) no refleja inestabilidad del método, sino la presencia de casos donde GSO-RL supera ampliamente a CPLEX, en particular `val9_3_1200` (−15,89 %) y `val10_3_1200` (−23,02 %): instancias de gran escala donde CPLEX agota su límite de 3 600 s atrapado en regiones de baja calidad, mientras GSO-RL, gracias a su memoria adaptativa, continúa progresando dentro de su límite de 300 s.

Tabla 2. Estadísticos del %GAP por conjunto (90 instancias VAL, 300 s GSO-RL vs. 3 600 s CPLEX).

Conjunto	n	Media	Mediana	Desv. típ.	Mín	Máx
Set 1	30	−0,17 %	0,00 %	0,79 %	−3,33 %	+1,38 %
Set 2	30	+0,11 %	0,00 %	0,98 %	−2,33 %	+3,55 %
Set 3	30	−1,00 %	0,00 %	5,17 %	−23,02 %	+2,39 %
Global	90	−0,35 %	0,00 %	3,08 %	−23,02 %	+3,55 %

6.3. Análisis de convergencia

La Figura 3 muestra las curvas de convergencia de GSO-RL sobre cuatro instancias VAL representativas de distinto tamaño y configuración de capacidad. El eje temporal se representa en escala logarítmica para hacer visible tanto la fase de mejora inicial, que ocurre en décimas de segundo, como las mejoras tardías que se extienden hasta el límite de 300 s. Cada curva adopta forma escalonada porque el coste solo se actualiza cuando se encuentra una solución estrictamente mejor; el punto relleno marca el instante en que se alcanza el mejor coste registrado.

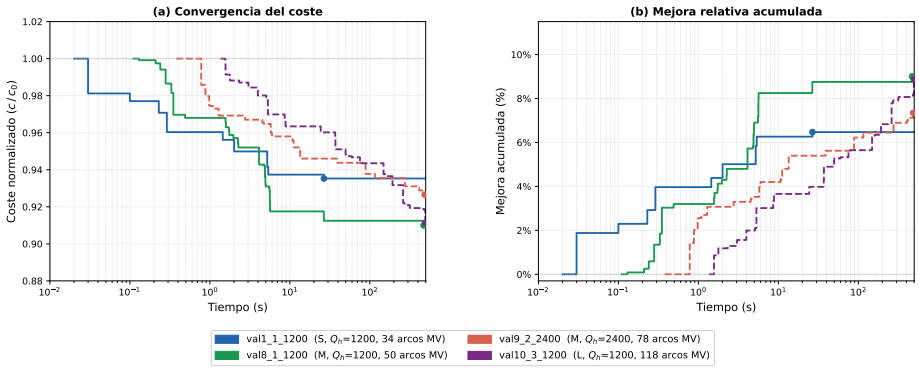


Figura 3. Curvas de convergencia de GSO-RL sobre cuatro instancias VAL. Panel (a): coste normalizado respecto al coste de la primera solución construida (c/c_0). Panel (b): mejora relativa acumulada respecto a c_0. Q_h denota la capacidad vehicular en unidades. Eje temporal en escala logarítmica; el punto relleno indica el instante del mejor coste. S/M/L: tamaño relativo de la instancia dentro del conjunto VAL.

El comportamiento es consistente en todas las instancias y responde a un patrón de dos fases bien diferenciadas. En la **primera fase**, que abarca aproximadamente el primer segundo de ejecución, GSO-RL registra las reducciones de

coste más pronunciadas: entre el 2 % y el 5 % del coste inicial. Esta caída refleja la acción del motor ALNS-VM sobre las guías recién generadas, que parten de soluciones constructivas de calidad moderada y convergen rápidamente hacia mínimos locales de buena calidad. En la **segunda fase**, las mejoras se vuelven más esporádicas y de menor magnitud individual, pero la memoria adaptativa de la GCM permite que el algoritmo continúe progresando durante intervalos prolongados en lugar de estancarse.

Este patrón de segunda fase es especialmente visible en las instancias de mayor complejidad. Para caracterizarlo con mayor detalle, estas instancias se ejecutaron adicionalmente con un límite extendido de 600 s, exclusivamente con fines de análisis de convergencia; los resultados competitivos de la Tabla 2 corresponden siempre al límite estándar de 300 s. La instancia `val8_1_1200` (M, 50 arcos VM) acumula una mejora total del 9,0 % y registra su último salto a los 470 s, bien pasado el punto en que un método sin memoria a largo plazo habría agotado su capacidad de exploración. La instancia `val10_3_1200` (L, 118 arcos VM) muestra el comportamiento más relevante: mejora de forma continua hasta los 495 s, con varios saltos distribuidos a lo largo de todo el horizonte de búsqueda y una reducción acumulada del 8,9 %. Este es precisamente el patrón que subyace a los gaps negativos extremos del Set 3: instancias donde CPLEX, limitado a 3 600 s, se queda atrapado en regiones de baja calidad mientras GSO-RL, en solo 300 s, sigue explotando su historial regional para escapar de ellas.

En contraste, la instancia `val1_1_1200` (S, 34 arcos VM) alcanza su mejor coste a los 27 s y no registra mejoras posteriores, lo que es coherente con el limitado tamaño de su espacio de búsqueda: una vez que las principales regiones han sido exploradas, la memoria no aporta señal adicional relevante. Este resultado no contradice la utilidad de la GCM, sino que la delimita: su beneficio crece con la escala del problema, en línea con las conclusiones del estudio de ablación para el SCC-VRP [1].

7. Conclusiones y trabajo futuro

Hemos adaptado GSO-RL al ML-CARPIF logrando una optimización integrada VM+VC que evita el sesgo hacia óptimos locales del primer nivel, mediante una arquitectura de doble capa de aprendizaje: agentes MABA por región controlan la intensidad de perturbación, mientras ALNSLearner ajusta globalmente los operadores de destrucción.

Los experimentos sobre 90 instancias VAL muestran que GSO-RL iguala o mejora la cota de CPLEX en el 69 % de los casos con doce veces menos tiempo de cómputo, obteniendo un gap medio de $-0{,}35\,\%$ y el 80 % de las instancias dentro de $\pm 1\,\%$. Los casos más destacados, `val9_3_1200` ($-15{,}89\,\%$) y `val10_3_1200` ($-23{,}02\,\%$), ilustran cómo la memoria adaptativa de la GCM permite escapar de regiones donde CPLEX se queda atrapado.

Como trabajo futuro cabe explorar el tamaño dinámico del conjunto de guías, regiones de anchura no uniforme y retención parcial del historial MABA al reemplazar una guía. En cuanto al problema, resulta de interés extender el modelo

a variantes con horarios irregulares, costes fijos de despliegue o decisiones de localización de las cabañas.

Referencias

1. Siverio-González, J.A., Melián-Batista, B., Moreno-Vega, J.M.: Guided strategic oscillation with reinforcement learning for the shared customer collaboration vehicle routing problem. SSRN preprint (2026). https://doi.org/10.2139/ssrn.6804414
2. Wei, C., Wøhlk, S., Che, A.: A multi-level capacitated arc routing problem with intermediate facilities in waste collection. Computers & Operations Research **167**, 106671 (2024). https://doi.org/10.1016/j.cor.2024.106671
3. Wei, C., Che, A., Xue, L.: A hierarchical integrated exact algorithm for multi-level capacitated arc routing problem in waste collection. Transportation Research Part E **204**, 104441 (2025). https://doi.org/10.1016/j.tre.2025.104441
4. Golden, B.L., Wong, R.T.: Capacitated arc routing problems. Networks **11**(3), 305–315 (1981). https://doi.org/10.1002/net.3230110308
5. Golden, B.L., DeArmon, J.S., Baker, E.K.: Computational experiments with algorithms for a class of routing problems. Computers & Operations Research **10**(1), 47–59 (1983). https://doi.org/10.1016/0305-0548(83)90026-6
6. Benavent, E., Campos, V., Corberán, Á., Mota, E.: The capacitated Chinese postman problem: Lower bounds. Networks **22**(7), 669–690 (1992). https://doi.org/10.1002/net.3230220706
7. Ghiani, G., Improta, G., Laporte, G.: The capacitated arc routing problem with intermediate facilities. Networks **37**(3), 134–143 (2001). https://doi.org/10.1002/net.3
8. Glover, F.: Tabu search and adaptive memory programming — advances, applications and challenges. In: Interfaces in Computer Science and Operations Research, pp. 1–75. Springer, Boston (1997)
9. Glover, F., Hao, J.-K.: The case for strategic oscillation. Annals of Operations Research **183**(1), 163–173 (2011)
10. Glover, F.: Multi-wave algorithms for metaheuristic optimization. Journal of Heuristics **22**(3), 331–358 (2016). https://doi.org/10.1007/s10732-016-9312-y
11. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2–3), 235–256 (2002). https://doi.org/10.1023/A:1013689704352
12. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002
13. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research **12**(4), 568–581 (1964). https://doi.org/10.1287/opre.12.4.568