Implementación del Algoritmo Bioinspirado Bobcat para un Problema de Optimización en un contexto de Asignación de Hubs

Fabián Alexis Vidal Torres[a,*]

[a]*Escuela de Ingeniería Informática, Universidad de Valparaíso, Valparaíso, Chile.*

Abstract

El artículo propone la aplicación del Algoritmo Bioinspirado Bobcat (BOA) para resolver un problema de asignación de microhubs temporales en la última milla logística. Se modela el problema como la minimización de la suma de distancias de clientes a hubs, sujeto a restricciones de capacidad y distancia máxima. BOA se implementa con fases de exploración y explotación, evaluándose frente a un solver MILP (MiniZinc/COIN-BC) en tres instancias (simple, media y dura). Los resultados muestran que BOA alcanza el óptimo en el caso sencillo, se acerca notablemente en el medio (0,7% de desviación) y presenta mayores diferencias en el más complejo (12,2%). Se concluye que BOA es factible y de fácil implementación, aunque su precisión decae con la complejidad. Lo anterior da paso a evaluar el algoritmo en un contexto de mejora híbrida y analizar su desempeño en escenarios más complejos.

Keywords: Bobcat Optimization Algorithm, Swarm Intelligence, Last-mile logistics

1. Introducción

Con el crecimiento del comercio electrónico, la demanda de entregas rápidas y convenientes de productos se ha transformado en una parte importante dentro de los procesos logísticos en las organizaciones; y en lo que respecta también a procesos relacionados con la logística de última milla. Esta última, siendo una actividad crítica que une la eficiencia de la operación con la satisfacción del cliente [1].

En particular, el concepto de última milla se refiere a la fase final de la cadena de suministro, desde el centro de distribución hasta el destino final del producto, que puede ser el domicilio del cliente, una tienda o cualquier otro punto de entrega. Una estrategia eficiente para minimizar los costos del traslado y entrega de productos es establecer hubs temporales como puntos de pick-up, donde los clientes pueden retirar sus pedidos. En algunos trabajos se ha abordado también el adoptar la propuesta de integrar formas alternativas y eficientes de entregar productos en las estrategias de última milla.[2].

En el contexto de este trabajo, se busca implementar una solución aplicada a una problemática de última milla mediante la evaluación de microhubs temporales como una alternativa para mejorar la eficiencia del sistema logístico.

El problema principal consiste en determinar qué microhubs temporales habilitar y cómo asignar a cada cliente a uno de ellos, de forma que se minimicen los costos totales. Estos costos incluyen tanto la distancia que deben recorrer los clientes como los costos fijos asociados a la habilitación de cada microhub.

La solución propuesta consiste en la implementación del algoritmo bioinspirado Bobcat, aplicado a este problema de minimización de costos logísticos. Numerosos estudios han abordado problemas de optimización mediante algoritmos bioinspirados [3, 4, 5], incluyendo propuestas de mejora sobre sus variantes [6], y en particular, aplicaciones exitosas del algoritmo Bobcat en contextos similares [7, 8]. Estas evidencias respaldan la viabilidad de aplicar este tipo de algoritmos a problemas como el que se describe en este trabajo.

La realización de este trabajo busca contribuir al estudio y aplicación de algoritmos bioinspirados en problemas de optimización, especialmente en el ámbito de la logística. Este caso particular puede servir como referencia metodológica para futuras aplicaciones en contextos similares.

El objetivo general consiste en evaluar el desempeño de BOA para resolver el problema de asignación de microhubs temporales en la logística de última milla. Por otro lado, los objetivos específicos son: (1) Definir e identificar las restricciones asociadas al problema. (2) Representar computacionalmente el problema y BOA mediante estructuras de datos adecuadas. (3) Implementar BOA en un escenario experimental para su posterior evaluación. (4) Comparar el desempeño del algoritmo frente a múltiples instancias del problema. (5) Analizar la calidad de las soluciones encontradas en términos de costo total y factibilidad de las asignaciones.

La estructura del documento se organiza de la siguiente manera: la presente sección número 1 abarca una introducción al trabajo a realizar;la sección 2 se encarga de modelar el problema; la sección 3 describe el modelamiento de BOA; la sección 4 describe las principales tecnologías utilizadas en términos de implementación; la sección 5 muestra las pruebas, discusión y resultados de la implementación; y, por último, la sección 6 contiene la conclusión.

*Corresponding author

Email address: fabian.vidal@postgrado.uv.cl (Fabián Alexis Vidal Torres)

2. Modelamiento del Problema

La empresa logística PuertoRápido busca implementar una solución sustentable para la distribución de última milla en la ciudad de Valparaíso, la cual presenta condiciones geográficas y de movilidad complejas que dificultan la entrega directa a domicilio. Como alternativa, se propone la instalación de microhubs temporales que funcionen como puntos de retiro, permitiendo que los clientes se acerquen a recoger sus paquetes.

En este contexto, la estrategia consiste en determinar qué microhubs habilitar y cómo asignar a cada cliente a uno de ellos, con el objetivo de minimizar el costo total del sistema. A continuación, se detallan los parámetros, variables, función objetivo, restricciones e instancias del problema, que permiten formalizar este modelo de optimización.

2.1. Conjuntos y parámetros

- C: conjunto de clientes.
- J: conjunto de ubicaciones candidatas para microhubs temporales.
- d_{cj}: distancia (en minutos caminando) entre el cliente $c \in C$ y el microhub $j \in J$.
- f_j: costo fijo de habilitar el microhub $j \in J$.
- L_j: capacidad máxima de atención del microhub $j \in J$ (número máximo de clientes).
- $D_{\max}$: distancia máxima tolerada entre cliente y hub.

2.2. Variables de decisión

$$x_{cj} \in \{0,1\} \quad = \begin{cases} 1, & \text{si el cliente } c \text{ retira en el microhub } j, \\ 0, & \text{en caso contrario.} \end{cases}$$

$$y_j \in \{0,1\} \quad = \begin{cases} 1, & \text{si se habilita el microhub } j, \\ 0, & \text{en caso contrario.} \end{cases}$$

2.3. Función objetivo

$$\min \sum_{c \in C} \sum_{j \in J} d_{cj}\, x_{cj} \; + \; \sum_{j \in J} f_j\, y_j. \tag{1}$$

2.4. Restricciones

$$\sum_{j \in J} x_{cj} = 1 \quad \forall\, c \in C \quad \text{(un cliente para un solo hub)} \tag{2}$$

$$x_{cj} \le y_j \quad \forall\, c \in C,\ j \in J \quad \text{(solo a hubs habilitados)} \tag{3}$$

$$\sum_{c \in C} x_{cj} \le L_j \quad \forall\, j \in J \quad \text{(capacidad de cada hub)} \tag{4}$$

$$x_{cj} = 0 \quad \forall\, c \in C,\ j \in J \quad \text{si } d_{cj} > D_{\max} \quad \text{(distancia máx)} \tag{5}$$

Table 1: Costo fijo y capacidad de los microhubs candidatos

Hub	Costo fijo \$	Capacidad L_j
H1: Estación Metro Valparaíso	20	2
H2: Terminal de Buses Barón	25	3
H3: Universidad de Playa Ancha	15	2

Table 2: Distancias d_{cj} (minutos caminando) desde cada cliente c a cada hub j

Cliente	H1	H2	H3
C1	5	8	11
C2	7	4	10
C3	6	6	6
C4	9	5	7
C5	8	9	5
C6	4	7	12

2.5. Instancia de ejemplo

Para un piloto con 6 clientes y 3 ubicaciones candidatas, se tienen los datos que se muestran en las Tablas 1 y 2.

Se fija además $D_{\max} = 8$ minutos.

3. Bobcat Optimization Algorithm (BOA)

En esta sección se describirá el algoritmo de BOA, junto con sus parámetros y ecuaciones de movimiento, con el objetivo de poder entender el funcionamiento del mismo y su aplicación al presente contexto. Todo esto en base a lo recopilado por parte de los autores principales[8].

3.1. Pseudo-código

El algoritmo de BOA opera de la siguiente manera:

3.2. Inicialización

Inicialmente el BOA mantiene una población de N *bobcats*, cada uno representado por un vector:

$$X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m}) \in \mathbb{R}^m,$$

y la matriz población:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} \in \mathbb{R}^{N \times m}. \tag{6}$$

La posición inicial de cada dimensión d de cada bobcat i se asigna aleatoriamente dentro de sus cotas:

$$x_{i,d} \;\leftarrow\; \ell_d \;+\; r\,(u_d - \ell_d), \qquad r \sim U(0,1), \tag{7}$$

donde ℓ_d y u_d son, respectivamente, el límite inferior y superior de la variable d. A continuación se evalúa la función objetivo para cada X_i, guardando los valores en el vector

$$F = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_N) \end{bmatrix}. \tag{8}$$

Algorithm 1 Bobcat Optimization Algorithm (BOA)

Require: Problema: variables, función objetivo y restricciones
Require: Tamaño de población N, número de iteraciones T
Ensure: Mejor solución cuasi–óptima

Inicialización:
Generar población inicial $\{X_i\}_{i=1}^{N}$ al azar:

$$x_{i,d} \leftarrow l_{b_d} + r \cdot (u_{b_d} - l_{b_d}) \quad \forall\, i = 1, \ldots, N,\ d = 1, \ldots, D$$

Evaluar $F_i = F(X_i)$ para cada individuo
for $t \leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    Fase 1: Seguimiento y movimiento hacia presa (exploración)
    Construir conjunto de presas

$$CP_i \leftarrow \{\, X_k : F_k < F_i \text{ y } k \neq i\}$$

    Seleccionar al azar presas candidatas del conjunto $SP_{i,j}$
    Calcular posición exploratoria

$$x_{i,d}^{P1} \leftarrow x_{i,j} + (1 - 2\, r_{i,j})\,(SP_{i,j} - l_{i,j}\, x_{i,j})$$

    if $F_i^{P1} < F_i$ **then**
      $X_i \leftarrow X_i^{P1}$
    end if
    Fase 2: Persecución para atrapar presa (explotación)
    Calcular posición explotatoria

$$x_{i,d}^{P2} \leftarrow x_{i,d} + x_{i,j} + \frac{1 - 2\, r_{i,j}}{1 + t}\, x_{i,j}$$

    if $F_i^{P2} < F_i$ **then**
      $X_i \leftarrow X_i^{P2}$
    end if
  end for
  Guardar la mejor solución candidata hasta el momento
end for

return Mejor solución encontrada

3.3. Fase 1: Exploración (seguimiento de presa)

En esta fase, cada bobcat i "busca" presas entre aquellos con mejor desempeño:

$$CP_i \;=\; \{\, X_k : F(X_k) < F(X_i),\ k \neq i\},$$

y elige aleatoriamente una presa $SP_i \in CP_i$ del conjunto total de candidatos, se entiende por mejor candidato aquel bobcat que posea una mejor solución que el bobcat actual. Para simular el movimiento de aproximación, se calcula una nueva posición.

$$x_{i,d}^{P1} \;\leftarrow\; x_{i,d} \;+\; (1 - 2\, r_{i,d})\,(SP_{i,d} - I_{i,d}\, x_{i,d}), \tag{9}$$

donde $r_{i,d} \sim U(0, 1)$ y $I_{i,d} \in \{1, 2\}$ es un factor aleatorio. Si la nueva posición mejora el valor objetivo,

$$X_i \;\leftarrow\; \begin{cases} X_i^{P1}, & F(X_i^{P1}) < F(X_i), \\ X_i, & \text{en otro caso.} \end{cases} \tag{10}$$

3.4. Fase 2: Explotación (persecución)

Una vez localizada la presa, el movimiento es más local. La posición se ajusta según

$$x_{i,d}^{P2} \;\leftarrow\; x_{i,d} \;+\; x_{i,j} \;+\; \frac{1 - 2\, r_{i,d}}{1 + t}\, x_{i,j}, \tag{11}$$

donde t es el número de iteración actual. De nuevo, se acepta solo si hay mejora:

$$X_i \;\leftarrow\; \begin{cases} X_i^{P2}, & F(X_i^{P2}) < F(X_i), \\ X_i, & \text{en otro caso.} \end{cases} \tag{12}$$

El algoritmo repite estas fases para $t = 1, \ldots, T$, conservando tras cada iteración la mejor solución encontrada.

4. Implementación

4.1. Parámetros del Algorítmo

Para la elección de parámetros se especifica a continuación una tabla ilustrativa que representa la cantidad de particulares, la cantidad de iteraciones, el número de ejecuciones y el tipo de instancia del problema que se divide en 3 tipos: (1) instancia simple; (2) instancia media y (3) instancia dura, cada una de estas instancias siendo mayormente restrictivas en ese mismo orden.

Instancia	n_partículas	n_iteraciones	n_ejecuciones
Simple (6x3)	10	100	40
Media (12x5)	15	500	40
Dura (24x8)	20	1000	40

Table 3: Parámetros del algoritmo

Los parámetros fueron seleccionados a partir de pruebas experimentales, considerando tiempos de ejecución del orden de segundos.

4.2. Tecnologías utilizadas

Las tecnologías utilizadas como lenguaje de programación, librerías[9] y editor de código utilizados en este trabajo se describen a continuación:

- **Visual Studio:** Es un editor de código fuente ligero y multiplataforma desarrollado por Microsoft, que admite una amplia variedad de lenguajes de programación y cuenta con numerosas extensiones para personalizaci´ on y funcionalidad. Es conocido por su interfaz intuitiva y herramientas de desarrollo integradas.

- **Python:** es un lenguaje de programación interpretado, de alto nivel y con una sintaxis sencilla, lo que facilita su aprendizaje y uso. Se eligió debido a su extensa colección de bibliotecas, que simplifican tareas complejas en áreas como el análisis de datos y la optimización. Además, cuenta con una amplia comunidad de usuarios y desarrolladores que contribuyen activamente a su evolución y brindan soporte constante. Su capacidad de integración con otras tecnologías y sistemas lo convierte en una herramienta versátil. Python ha sido ampliamente utilizado en el campo de la inteligencia artificial, especialmente en el procesamiento del lenguaje natural y en el desarrollo de algoritmos metaheurísticos [10, 11].

- **Librería math:** Este módulo de Python proporciona acceso a funciones matemáticas comunes y a constantes. En el contexto de este trabajo, se utiliza la función *"math.floor(x)"*, cuya finalidad es redondear hacia abajo el número *"x"* al entero más cercano menor o igual que *"x"*. En este código, *"math.floor()"* se emplea durante la fase de inicialización para asegurar que los valores aleatorios generados para las posiciones iniciales de los bobcats sean enteros válidos que representen índices de hubs, es decir, asignaciones posibles a los diferentes centros de distribución.

- **Librería random:** Este módulo de Python implementa generadores de números pseudoaleatorios para varias distribuciones. En el contexto de este trabajo se utilizó *"random.random()"*, cuyo propósito es generar un número aleatorio flotante entre un intervalo de [0,1]. Particularmente, se hizo uso de esto para la implementación de un parámetro del algorítmo de BOA.

- **Librería pandas:** La biblioteca *pandas* permite la manipulación y análisis de datos de forma estructurada a través de estructuras como *Series* y *DataFrames*. En este código, se utilizó para construir una serie con los valores de fitness obtenidos a lo largo de las ejecuciones del algoritmo BOA y para generar una tabla resumen con estadísticas descriptivas como media, mediana, desviación estándar y cuartiles.

- **Librería matplotlib.pyplot:** El módulo *pyplot* de la biblioteca *matplotlib* proporciona funciones para la creación de gráficos estáticos. En este trabajo se utilizó para graficar la evolución del mejor valor de fitness a lo largo de las iteraciones (gráfico de convergencia), definiendo ejes, etiquetas y guardando la imagen en formato `.png`.

- **Librería seaborn:** *Seaborn* es una librería basada en *matplotlib* que facilita la creación de gráficos estadísticos con una estética predeterminada. En el código se utilizó para generar un *boxplot* que resume visualmente la distribución de los mejores valores de fitness obtenidos en las 40 ejecuciones del algoritmo.

4.3. Repositorio

El código fuente de la implementación del algoritmo Bobcat Optimization Algorithm (BOA) desarrollado en este trabajo para este contexto y el problema descrito, está disponible en un repositorio público de GitHub. En dicho repositorio se incluye todo el código necesario para ejecutar el algoritmo, junto con un archivo README que contiene instrucciones detalladas para la configuración y uso del software. Así, se facilita la replicabilidad y el acceso abierto al material desarrollado.

5. Pruebas y discusión de resultados

5.1. Pruebas Comparativas

Para la ejecución de pruebas comparativas, se utilizó la herramienta SOLVER con MiniZinc. Esta herramienta es un lenguaje de modelado de restricciones de alto nivel que permite expresar y resolver fácilmente problemas de optimización discretos. Para este trabajo se utilizó el motor COIN-BC de la herramienta mencionada, el cual es utilizado para programación lineal entera mixta (MILP). Adecuado para modelos que usan muchas variables y restricciones lineales, como el problema que se describe en este trabajo.

Para realizar la comparación se tendrá en cuenta la desviación relativa del óptimo encontrado de BOA con respecto a cada instancia del problema. Dicha desviación la podemos especificar como:

$$\text{Desviación Relativa (\%)} = \left(\frac{\text{Costo}_{\text{BOA}} - \text{Costo}_{\text{Solver}}}{\text{Costo}_{\text{Solver}}} \right) \times 100$$

Por lo visto y en base a los resultados, se puede decir que BOA requiere de una población de agentes adecuada para alcanzar soluciones factibles. Dentro de esas soluciones factibles, existe 1 instancia que es la simple, la cual alcanza el óptimo conocido. Para la instancia media, se acerca bastante a la solución, mostrando un resultado 0,6 desviado de la mejor conocida. Para la instancia dura y mediante el uso de solo 20 agentes, presenta una desviación mayor. Esto conduce a la hipótesis de que un mayor número de agentes podría incrementar la probabilidad de obtener soluciones óptimas o de mayor calidad.

Dicho lo anterior, a continuación se muestra la tabla comparativa con los resultados.

Instancia	**Óptimo Solver**	**Óptimo BOA**	**Desv menor**	**Desv mayor**
Simple (6x3)	89	89	0%	6,3%
Media (12x5)	151	152	0,7%	7,3%
Dura (24x8)	271	309	12,2%	21,6%

Table 4: Tabla comparativa: SOLVER VS BOA

5.2. Métricas de desempeño

En esta sub-sección, se dará paso a explicar las métricas de desempeño empleadas para la evaluación del algoritmo en cuestión.

5.2.1. Análisis de Medidas de Tendencia Central

Para el caso de las métricas se utilizarán medidas de tendencia central, en particular la media, mediana y la desviación estándar para medir el comportamiento de las variables de estudio.

- **Media aritmética:** Es la suma de todos los valores dividida entre el número total de observaciones[12]. Se calcula mediante:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{13}$$

  Donde N es la cantidad total de valores, y $\sum_{i=1}^{N} x_i$ representa la suma de todos los valores.

- **Mediana:** Es el valor central en un conjunto de datos ordenados[12].

  – Si n es impar:

$$Me = \frac{n+1}{2} \tag{14}$$

  – Si n es par:

$$Me = \frac{x_{n/2} + x_{(n/2)+1}}{2} \tag{15}$$

- **Mejor valor:** Representa el mejor valor de todos los n experimentos. Para problemas de maximización, siendo este el más alto de todos y para problemas de minimización, siendo este el más bajo de todos.

- **Peor valor:** Representa el peor valor de todos los experimentos. Para problemas de maximización, siendo este el más bajo de todos y para problemas de minimización, siendo este el más alto de todos.

- **Desviación estándar:** Representa la variación o dispersión de un conjunto de datos respecto a la media[12]. Se calcula mediante:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{16}$$

5.2.2. Ánalisis de Outliers y Rangos Intercuartílicos

Los resultados o valores atípicos son aquellos que se encuentran lejos del cuerpo principal de datos, pueden surgir por errores de registro, fallas en el equipo o diferencias válidas pero extremas dentro del conjunto de datos. Aunque pueden distorsionar medidas como la media (x) y la desviación estándar (s), también pueden contener información valiosa[13]. Por ello, identificar estos valores es crucial en el análisis preliminar de datos, y una herramienta útil para esto es la gráfica de caja.

Dicho lo anterior, a continuación se da paso a la definición de esta métrica de desempeño y sus variantes.

- **Rango intercuartílico (IQR):** Representa el rango de valores que incluye el 50% de una distribución. Este se obtiene de la siguiente manera:

$$IQR = Q3 - Q1 \tag{17}$$

- **Primer cuartil ($Q1$):**

$$Q1 = \frac{N+1}{4} \tag{18}$$

- **Tercer cuartil ($Q3$):**

$$Q3 = \frac{3(N+1)}{4} \tag{19}$$

- **Outlier leve:**

$$VM < Q1 - 1.5 \cdot IQR \tag{20}$$
$$VM > Q3 + 1.5 \cdot IQR \tag{21}$$

- **Outlier extremo:**

$$VM < Q1 - 3 \cdot IQR \tag{22}$$
$$VM > Q3 + 3 \cdot IQR \tag{23}$$

Donde:

- VM: es el valor de la muestra.
- N: es la cantidad total de valores.
- $Q1$: es el primer cuartil.
- $Q3$: es el tercer cuartil.
- IQR: es el rango intercuartil, calculado como $Q3 - Q1$.

Con estas métricas ya definidas se puede dar paso a la siguiente sección correspondiente al análisis estadístico.

5.2.3. Análisis Estadístico

Para el análisis estadístico se utilizará el test de Mann-Whitney U. Este test es una prueba no paramétrica que sirve para evaluar si dos muestras independientes provienen de la misma distribución sin asumir normalidad ni igualdad de varianzas; bajo la hipótesis nula se plantea que ambas muestras comparten distribución, mientras que la alternativa sugiere diferencias en la tendencia de sus valores. Para llevarlo a cabo se combinan los datos de ambos grupos, se ordenan y se asignan rangos, y a partir de la suma de rangos de cada grupo se calcula la estadística U, cuya significación se determina mediante tablas o aproximación normal. Sus principales ventajas son la robustez ante valores atípicos y la capacidad de trabajar con datos ordinales, aunque pierde potencia cuando hay muchos empates o varianzas muy desiguales y asume que las distribuciones tienen formas similares[14].

5.2.4. *Resumen Descripitvo*

En este apartado se presenta una tabla de resumen descriptivo con las diferentes instancias del problema.

Instancia	μ	σ	$x_{\min}$	Q_1	$\tilde{x}$	Q_3	$x_{\max}$
S (6×3)	90.2	1.90	89	89	89	92	95
M (12×5)	156.1	3.0	152	154	156	158	163
D (24×8)	328.4	8.1	309	324	328	334	346

Table 5: Resumen descriptivo de resultados por instancia

De acuerdo a lo ilustrado en **tabla 5**, se puede observar que el comportamiento del algoritmo de BOA para una instancia simple llega al mejor valor posible, presenta una media cercana al valor mínimo con una desviación estándar minúscula, lo cual representa que en la mayor parte de las ejecuciones de BOA este alcanza los valores esperados.

Por su parte, el comportamiento en la instancia media presenta una baja desviación y una media bastante acercada al valor real óptimo, los rangos intercuartílicos Q1 y Q2 no se alejan de la media y el mejor valor encontrado presenta una diferencia de una unidad en comparación con el del SOLVER.

Por último, la instancia dura muestra una media más alejada del valor esperado, el mejor valor alcanzado fue de 309 el cual está mayormente más alejado del ópitmo, y el peor valor presenta una desviación relativa sobre los 50 puntos, lo cual sugiere que el algoritmo para problemas de dimensionalidad más complejas o costosas suele ser menos preciso.

5.2.5. *Resumen Estadístico*

En esta sección se presenta el análisis estadístico realizando una comparación del algoritmo BOA vs PSO, se analiza cada instancia del mismo problema para presentar los resultados del test de Mann Whitney U.

Instancia	**Valor Mann W**	**p-value**
Simple (6x3)	435	0,334
Media (12x5)	412	0
Dura (24x8)	248	0003

Table 6: Resumen Estadístico de resultados por instancia

En la instancia simple (6×3), con U=435 y p=0,334, no se rechaza la hipótesis nula, lo que indica que BOA y PSO rinden de manera similar en problemas de baja complejidad; en cambio, para la instancia media (12×5) con U=412 y p=0,000, así como para la dura (24×8) con U=248 y p=0,0003, el p-value es muy inferior a 0,05, por lo que existe una diferencia estadísticamente significativa entre las distribuciones de resultados de ambos algoritmos en problemas de complejidad media y alta—esto significa que uno de ellos (el que presente la mediana de fitness menor) supera de forma consistente al otro conforme aumenta la dificultad del problema.

5.2.6. *Gráficos de convergencia*

Se presentan los gráficos de convergencia del BOA, con su correspondiente interpretación y discusión.

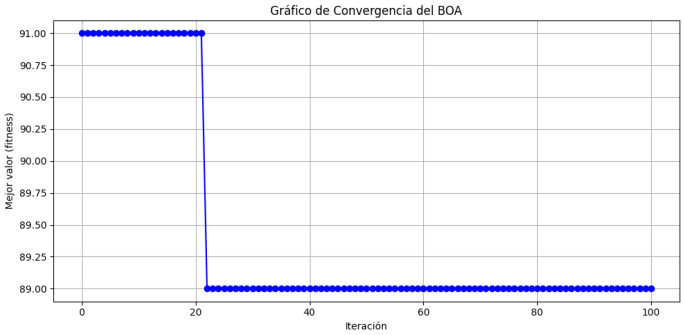

Figure 1: Convergencia Instancia Simple

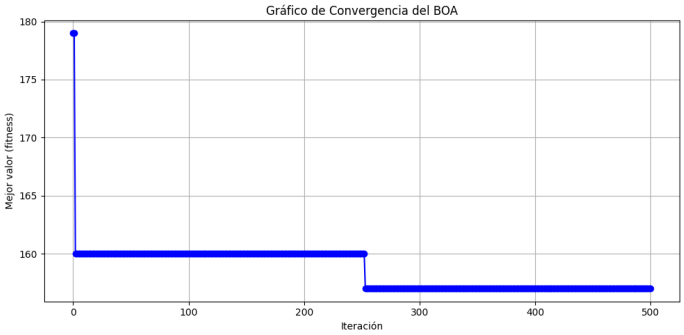

Figure 2: Convergencia Instancia Media

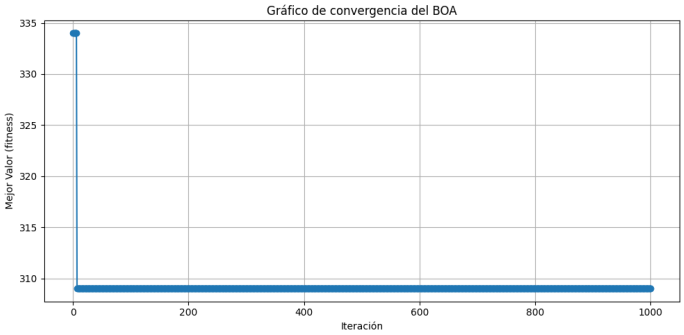

Figure 3: Convergencia Instancia Dura

En base a lo recopilado en cada gráfico correspondiente a cada instancia, se puede observar que el algoritmo presenta una convergencia temprana en las primeras iteraciones cuando la dimensionalidad es pequeña. Por ejemplo, en una instancia con 100 iteraciones, alcanzó el mejor resultado en la número 22, lo que indica una buena precisión del algoritmo BOA en problemas de bajo costo. No se evidencia un estancamiento significativo.

Por otro lado, en la instancia de dificultad media, se observa un mayor grado de estancamiento. En las primeras iteraciones, el algoritmo converge hacia una solución cercana al óptimo sin llegar a alcanzarlo, y posteriormente le cuesta encontrar mejoras hasta aproximadamente la iteración 250. Luego, muestra una nueva fase de convergencia, acercándose a la solución óptima global, aunque sin lograr alcanzarla, obteniendo un

valor de fitness de 152.

Finalmente, en la instancia más difícil, el algoritmo presenta un estancamiento más marcado. De las 1000 iteraciones, se observa que entre la iteración 20 y la 30 comienza a estancarse, sin lograr encontrar soluciones significativamente mejores. Esta situación sugiere que BOA, al basarse en movimientos simples pero factibles, tiene dificultades para escapar de óptimos locales en problemas más complejos. Esto podría deberse a la naturaleza de su mecanismo de exploración dentro del espacio de búsqueda.

5.2.7. Boxplot

En el siguiente apartado se presentan los gráficos de caja y bigote mejor conocidos como boxplot del BOA, con su correspondiente interpretación y discusión.

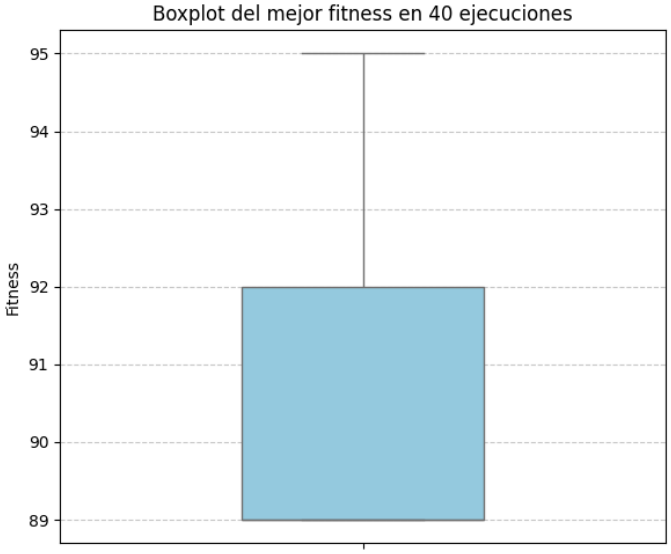

Figure 4: Boxplot Instancia Simple

En cuanto a los resultados del gráfico de caja y bigotes (boxplot), el algoritmo BOA no presenta valores atípicos en ninguno de los tres casos o instancias analizadas.

Para la instancia simple, el extremo superior muestra una diferencia de solo 3 puntos respecto al tercer cuartil (Q3), lo que indica una dispersión poco significativa. Esto sugiere que la mayoría de los valores están concentrados cerca de la solución óptima.

En la instancia media, se observa un boxplot que incluye un límite inferior, el cual no estaba presente en la instancia simple. Este límite inferior presenta una diferencia de 2 puntos respecto al primer cuartil (Q1) y, de hecho, coincide con el valor óptimo encontrado por el BOA. Por otro lado, el límite superior se aleja un poco más de la media, con una diferencia de 6 puntos, lo que refleja una mayor dispersión de los datos hacia valores menos óptimos.

Finalmente, en la instancia difícil, el boxplot presenta una notable simetría entre los rangos superior e inferior respecto a la caja del gráfico. Esto sugiere una variabilidad estándar de

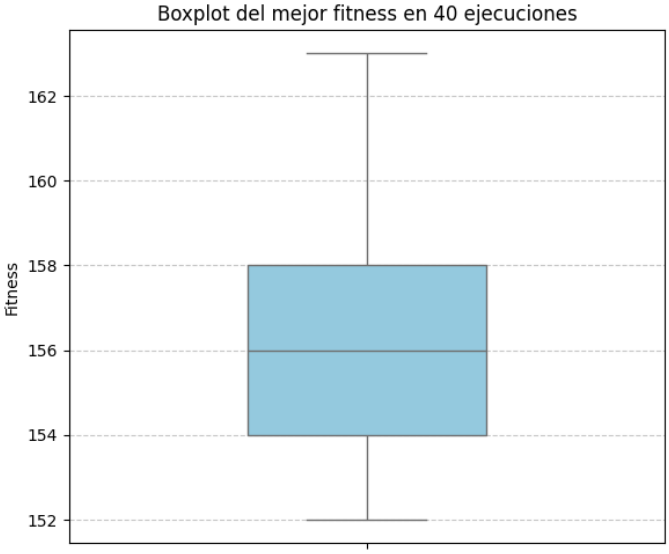

Figure 5: Boxplot Instancia Media

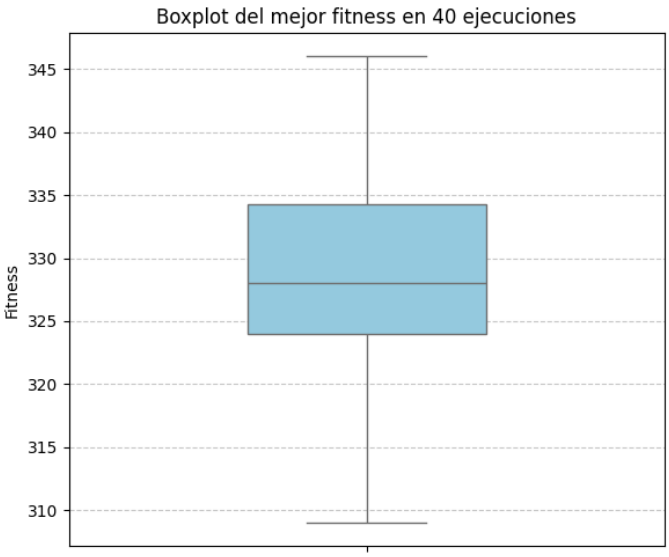

Figure 6: Boxplot Instancia Dura

los datos, donde el algoritmo BOA encontró soluciones tanto precisas como imprecisas en proporciones similares.

5.2.8. Gráficos de Q-Metrics

Dada una ejecución de un algoritmo de optimización sobre un benchmark f, cuya mejor solución alcanzó un valor de función objetivo f_{ach}, se define una medida de error absoluto de fitness como:

$$\text{error} = f_{\text{ach}} - f^*$$

donde f^* representa el valor óptimo conocido o estimado del problema. Para transformar este error en una métrica de calidad relativa y normalizada en el intervalo [0, 1], se utiliza la métrica denominada **QMetric**, definida en dos pasos:

1. Primero, se normaliza el error respecto a un rango estimado del problema, utilizando un valor máximo estimado $\hat{f}$:

$$q = \frac{\hat{f} - f_{\text{ach}}}{\hat{f} - f^*}$$

2. Luego, se escala esta proporción mediante un exponente dependiente de la cantidad de dígitos significativos d, obteniendo así la QMetric:

$$\text{QMetric} = 2q^{10d} - 1$$

Finalmente, el resultado se restringe al intervalo $[0, 1]$:

$$\text{QMetric} \in [0, 1]$$

Esta formulación permite interpretar la QMetric como una medida de calidad de la solución: valores cercanos a 1 indican soluciones altamente cercanas al óptimo, mientras que valores cercanos a 0 reflejan soluciones alejadas del valor óptimo conocido.

El parámetro d actúa como un control de exigencia: a mayor valor de d, más estricta es la penalización a soluciones que se alejan del óptimo. Por lo tanto, el exponente define qué tan sensible es la métrica a pequeñas diferencias de calidad.

A continuación, se presentan los resultados de aplicar la métrica QMetric en las tres instancias del problema. Cabe destacar que se utilizaron diferentes niveles de penalización en relación con el óptimo, ajustando la fórmula de QMetric según la dificultad de cada instancia:

- Para la **instancia simple**, se utilizó una penalización estricta con la expresión:

$$\text{QMetric} = 2q^{10d} - 1, \quad \text{con } d = 4.$$

- Para la **instancia media**, se optó por una penalización intermedia empleando:

$$\text{QMetric} = 2q^{d} - 1, \quad \text{con } d = 4.$$

- Para la **instancia dura**, se utilizó una penalización suave representada simplemente como:

$$\text{QMetric} = q.$$

Estas variantes permiten adaptar la sensibilidad de la métrica a la dificultad del problema, favoreciendo una interpretación más representativa de la calidad relativa de las soluciones obtenidas.

Los resultados obtenidos muestran un comportamiento diferenciado del algoritmo según la dificultad de la instancia y el nivel de exigencia impuesto por la QMetric.

En la **instancia simple**, utilizando una penalización exigente (QMetric $= 2q^{10d} - 1$ con $d = 4$), el algoritmo alcanza el valor máximo de la métrica (QMetric $= 1$) en pocas iteraciones. Este comportamiento es coherente con los gráficos de convergencia,

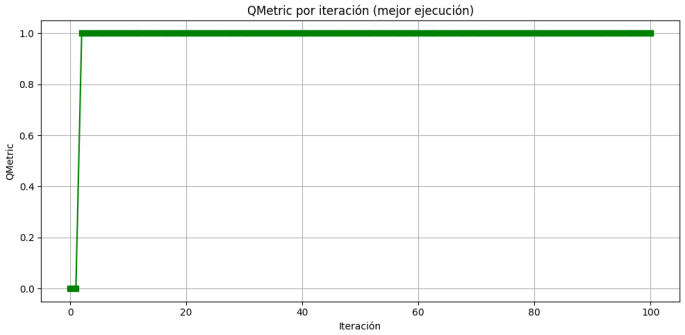

Figure 7: QMetrics Instancia Simple

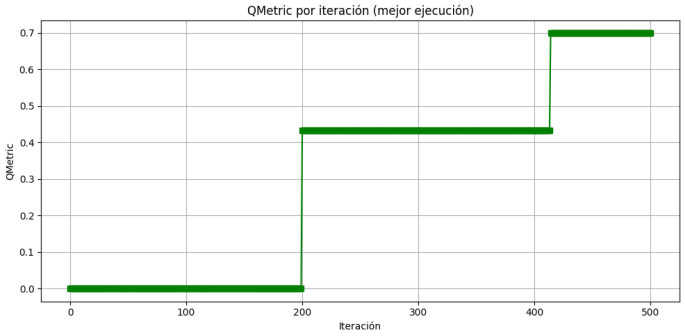

Figure 8: QMetrics Instancia Media

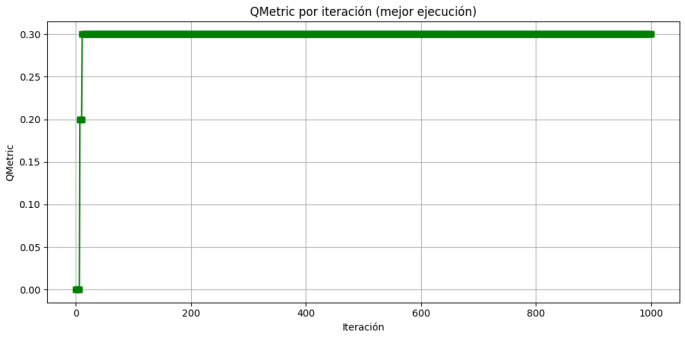

Figure 9: QMetrics Instancia Dura

en los que se observa que el óptimo es alcanzado de forma temprana.

En la **instancia media**, con una penalización intermedia (QMetric $= 2q^{d} - 1$), se aprecia un progreso gradual: en la iteración 200 la métrica alcanza un valor cercano a 0.4, y a partir de la iteración 410 supera la mitad del valor máximo, llegando hasta aproximadamente 0.7.

Finalmente, en la **instancia dura**, donde se aplicó una penalización menos estricta (QMetric $= q$), se observa un claro estancamiento. El valor máximo alcanzado por la métrica es de aproximadamente 0.3, sin mejoras posteriores, lo que evidencia la dificultad del algoritmo para aproximarse al valor óptimo en esta configuración más compleja.

6. Conclusiones

En este trabajo presentado se realizó una implementación del algoritmo BOA para un problema de optimización de asignación de hubs, en un contexto logístico de última milla. El trabajo evidencia a grandes rasgos la factibilidad que generan las soluciones de BOA para problemas de optimización y particularmente de minimización para el caso mostrado. BOA también se destaca por su simplicidad de implementación gracias a sus ecuaciones de movimiento directas y a la baja cantidad de parámetros requeridos. Estas características lo hacen especialmente atractivo para su aplicación en entornos industriales. No obstante, se reconoce que su desempeño puede mejorar aún más al ser evaluado en instancias más complejas, con una configuración adecuada en cuanto al número de agentes y parámetros, lo que abre la puerta a futuras investigaciones orientadas a su mejora y validación mediante algoritmos híbridos en escenarios reales de mayor escala.

References

[1] N. Oviedo, E. López Hincapié, Evolución de la logística de la última milla. revisión de la literatura 44 (2023) 216–229.

[2] N. Pourmohammadreza, M. R. A. Jokar, T. Van Woensel, Last-mile logistics with alternative delivery locations: A systematic literature review, Results in Engineering 25 (2025) 104085. doi:https://doi.org/10.1016/j.rineng.2025.104085.
URL https://www.sciencedirect.com/science/article/pii/S2590123025001732

[3] G. Han, Bio-inspired swarm intelligence for enhanced real-time aerial tracking: integrating whale optimization and grey wolf optimizer algorithms, Discover Artificial Intelligence 5 (1) (2025) 18. doi:10.1007/s44163-025-00237-5.
URL https://doi.org/10.1007/s44163-025-00237-5

[4] M. Shaabani, J. Bagherinejad, Optimizing flexible multi-compartment location routing problem for waste collection with priority of service using a hyper-heuristic algorithm and –constraint method, International Journal of Engineering 39 (1) (2026) 244–264. arXiv:https://www.ije.ir/article_217277_6236d3d13df63ecfcae335f5999510c1.pdf, doi:10.5829/ije.2026.39.01a.19.
URL https://www.ije.ir/article_217277.html

[5] S. Lin, J. Wang, B. Huang, X. Kong, H. Yang, Bio particle swarm optimization and reinforcement learning algorithm for path planning of automated guided vehicles in dynamic industrial environments, Scientific Reports 15 (1) (2025) 463. doi:10.1038/s41598-024-84821-2.
URL https://doi.org/10.1038/s41598-024-84821-2

[6] Y. Cai, H. Chen, An improved salp swarm algorithm for permutation flow shop vehicle routing problem, Scientific Reports 15 (1) (2025) 6704. doi:10.1038/s41598-025-86054-3.
URL https://doi.org/10.1038/s41598-025-86054-3

[7] M. Prajul, P. Subramanian, R. Surendran, Air pollution monitoring system using stacked attentional vectormap convolutional bidirectional network with bobcat optimization and iot-cloud, Global NEST Journal 27 (3) (2025). doi:10.30955/gnj.06937.
URL https://doi.org/10.30955/gnj.06937

[8] Z. Benmamoun, K. Khlie, G. Bektemyssova, M. Dehghani, Y. Gherabi, Bobcat optimization algorithm: an effective bio-inspired metaheuristic algorithm for solving supply chain optimization problems, Scientific Reports 14 (1) (2024) 20099. doi:10.1038/s41598-024-70497-1.
URL https://doi.org/10.1038/s41598-024-70497-1

[9] Python Software Foundation, Python 3 documentation, accessed: 2025-06-18 (2024).
URL https://docs.python.org/3/

[10] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.

[11] N. Van Thieu, E. H. Houssein, D. Oliva, N. D. Hung, Intelelm: A python framework for intelligent metaheuristic-based extreme learning machine, Neurocomputing 618 (2025) 129062. doi:https://doi.org/10.1016/j.neucom.2024.129062.
URL https://www.sciencedirect.com/science/article/pii/S0925231224018332

[12] M. F. Triola, Estadística, 12th Edition, Pearson Educación, Madrid, España, 2014.

[13] W. Mendenhall, R. J. Beaver, B. M. Beaver, Introducción a la probabilidad y estadística, 13th Edition, Cengage Learning Editores, S.A. de C.V., México, D.F., 2010.
URL https://www.fcfm.buap.mx/jzacarias/cursos/estad2/libros/book5e2.pdf

[14] N. Nachar, The mann-whitney u: A test for assessing whether two independent samples come from the same distribution, Tutorials in Quantitative Methods for Psychology 4 (03 2008). doi:10.20982/tqmp.04.1.p013.