**Technology Rain Journal ISSN**: 2953-464X
Vol. 4 Núm. 2 (Julio – Diciembre 2025), e108.

# Modelado de un problema de dispersión mediante Búsqueda en Vecindarios Variables (VNS) utilizando Octave

Modeling a Scattering Problem Using Variable Neighborhood Search (VNS) in Octave

**Maria T. Ortega O.**[1][0009-0000-3629-9751], **Daniel Sánchez Díaz**[2][0009-0008-4326-5734]

[1]Universidad de Panamá, Facultad de Informática, Electrónica y Comunicación, Departamento de Informática. Panamá maria.ortegao@up.ac.pa
[2]Universidad de Panamá, Facultad de Ciencias Naturales, Exactas y Tecnología, Departamento de Matemática, Panamá, daniel-a.sanchez@up.ac.pa

**CITA EN APA:**
Ortega O., M. T., & Sánchez Díaz, D. (2025). "Modelado de un problema de dispersión mediante Búsqueda en Vecindarios Variables (VNS) utilizando Octave". Technology Rain Journal, 4(2). https://doi.org/10.55204/trj.v4i2.e108

**Recibido:** 10 de octubre-2025
**Aceptado:** 15 de diciembre-2025
**Publicado:** 18 de diciembre-2025

Technology Rain Journal
**ISSN:** 2953-464X



**Resumen**. Los problemas de dispersión y diversidad constituyen una clase relevante de la optimización combinatoria, con aplicaciones en diseño de redes, planificación territorial y análisis de datos. Estos problemas, generalmente NP-duros, buscan seleccionar subconjuntos que maximicen la diversidad según una medida definida, lo que dificulta su resolución exacta en instancias grandes. En este trabajo se propone la aplicación de la metaheurística Variable Neighborhood Search (VNS) para modelar y resolver un problema de dispersión, implementada en el entorno de programación libre Octave. Se presenta la formulación matemática del problema, la representación de soluciones, la definición de vecindarios y el pseudocódigo del algoritmo, junto con fragmentos de su implementación para garantizar la reproducibilidad. Los experimentos realizados sobre instancias de tamaño medio evidencian que VNS obtiene soluciones competitivas en tiempos razonables. Los resultados confirman la eficacia del enfoque propuesto y destacan a Octave como una plataforma accesible para la investigación en optimización combinatoria.

**Palabras Clave:** Optimización combinatoria; problemas de dispersión y diversidad; búsqueda en vecindarios variables (VNS); metaheurísticas; Octave; modelado matemático; algoritmos de optimización.

**Abstract:** Dispersion and diversity problems constitute a relevant class within combinatorial optimization, with applications in network design, territorial planning, and data analysis. These problems, generally NP-hard, aim to select subsets that maximize diversity according to a defined measure, which makes their exact resolution infeasible for large instances. This work proposes the application of the Variable Neighborhood Search (VNS) metaheuristic to model and solve a dispersion problem, implemented in the free programming environment Octave. The mathematical formulation of the problem, the solution representation, the definition of neighborhoods, and the algorithm pseudocode are presented, together with fragments of its implementation to ensure reproducibility. Experiments conducted on medium-sized instances show that VNS achieves competitive solutions within reasonable computational times. The results confirm the effectiveness of the proposed approach and highlight Octave as an accessible platform for research in combinatorial optimization.

**Keywords:** Combinatorial optimization; dispersion and diversity problems; variable neighborhood search (VNS); metaheuristics; Octave; mathematical modeling; optimization algorithms.

## 1. INTRODUCCIÓN

Los problemas de dispersión y diversidad constituyen una de las áreas más estudiadas dentro de la optimización combinatoria, debido a su amplia aplicabilidad en distintos campos de la ciencia y la ingeniería. En términos generales, estos problemas buscan seleccionar un subconjunto de elementos de un conjunto mayor, de manera que se maximice la diferencia, heterogeneidad o distancia entre los elementos elegidos.

La motivación detrás de este tipo de problemas es que, en numerosos contextos, la diversidad se traduce en beneficios prácticos: en telecomunicaciones, garantiza mayor cobertura y robustez de las redes; en planificación territorial, asegura una distribución equilibrada de recursos; en análisis de datos, permite conformar grupos representativos y heterogéneos; y en educación, facilita la creación de equipos de trabajo con perfiles variados que potencian el aprendizaje colaborativo (Martí & Reinelt, 2011; Salhi & Smith, 2015)..

La dificultad de estos problemas radica en su complejidad computacional. La mayoría de las variantes de dispersión y diversidad son clasificadas como NP-duras, lo que implica que no existen algoritmos exactos eficientes para resolver instancias de gran tamaño en tiempos razonables (Martí & Reinelt, 2011; Salhi & Smith, 2015). Por ello, la investigación en este campo se ha orientado hacia el desarrollo de métodos aproximados, capaces de generar soluciones de calidad aceptable en plazos reducidos.

Entre estos métodos, las metaheurísticas han demostrado ser herramientas poderosas y versátiles. Técnicas como Simulated Annealing, Tabu Search, GRASP y Genetic Algorithms han sido aplicadas con éxito en diversos problemas de optimización combinatoria (Talbi, 2009; Michalewicz & Fogel, 2004). Sin embargo, una de las estrategias que ha ganado relevancia en las últimas décadas es la búsqueda en vecindarios variables (Variable Neighborhood Search, VNS).

El principio fundamental de VNS consiste en explorar de manera sistemática diferentes estructuras de vecindario, evitando la dependencia exclusiva de un único operador de búsqueda. Esta característica le permite escapar de óptimos locales y ampliar el espacio de soluciones exploradas, lo que incrementa la probabilidad de encontrar soluciones de mayor calidad. Además, VNS combina simplicidad conceptual con flexibilidad, ya que puede adaptarse a distintos tipos de problemas y estructuras de datos. Su eficacia ha sido

comprobada en problemas de localización, planificación, diseño de redes y clustering, entre otros.

En este trabajo se propone modelar un problema de dispersión utilizando VNS implementado en Octave, un entorno de programación libre y de código abierto, ampliamente utilizado en la comunidad académica por su similitud con MATLAB y su accesibilidad. La elección de Octave responde a la necesidad de promover la reproducibilidad y democratización del conocimiento, ofreciendo una plataforma gratuita que permite a estudiantes e investigadores replicar y extender los experimentos realizados.

El artículo presenta la formulación matemática del problema, el diseño del algoritmo VNS, su implementación en Octave (Octave Community 2023) y los resultados obtenidos en instancias de prueba. Con ello se busca aportar un recurso metodológico replicable, que contribuya tanto a la literatura especializada como a la formación académica en optimización combinatoria.

## 2. METODOLOGÍA

Se define un conjunto $\mathcal{N}$ de elementos y se busca seleccionar un subconjunto $\mathcal{S} \subseteq \mathcal{N}$ de tamaño fijo $p$. La función objetivo consiste en maximizar la dispersión, medida como la suma de distancias entre todos los pares de elementos en $\mathcal{S}$.

**Objetivo primario**: Evaluar si VNS supera a heurísticas base y metaheurísticas comparativas en calidad de solución y eficiencia computacional en instancias estándar de dispersión.

**Hipótesis H1 (calidad):** VNS produce valores objetivo significativamente mejores que una búsqueda local y una heurística constructiva en múltiples instancias

**Hipótesis H2 (eficiencia):** VNS alcanza soluciones de alta calidad con menor o comparable tiempo de ejecución

**Hipótesis H3 (robustez):** VNS mantiene desempeño estable frente a variaciones de parámetros e instancias con distinta estructura.

**Diseño experimental y benchmarks**

**Conjuntos de pruebas:**

Instancias pequeñas: berlin52, eil76.

Instancias medianas: kroA100, ch130.

Instancias grandes: pr439, pcb1173.

Adaptación al problema de dispersión: Construye la matriz de distancias y define el subconjunto a seleccionar (por ejemplo, tamaño m fijo). La función objetivo común en Máxima Dispersión: maximizar la mínima distancia entre pares del subconjunto seleccionado.

**Protocolos de corrida:**

- Corridas independientes: 30 repeticiones por instancia y método, semillas diferentes.
- Presupuesto de cómputo fijo: mismo límite de iteraciones o tiempo por método/instancia.
- Inicialización: solución inicial aleatoria + heurística rápida (opcional) para control.

**Métodos comparativos:**

- Heurística constructiva (greedy)
- Búsqueda local (1-swap / 2-swap)
- GRASP (básico)
- VNS

Los resultados confirman que VNS es una estrategia eficaz para problemas de dispersión y que Octave constituye una herramienta accesible y reproducible para su implementación. Este enfoque abre la posibilidad de extender el trabajo hacia instancias de mayor escala y hacia la hibridación con otras metaheurísticas.

## 3. FUNDAMENTO TEÓRICO

El problema de la dispersión con capacidades es un problema NP-hard, que pertenece a la familia de los problemas de dispersión o diversidad.

En teoría de la complejidad computacional, NP es el acrónimo en inglés de nondeterministic polynomial time ("tiempo polinomial no determinista"). Es el conjunto de

problemas que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista.

Los problemas de dispersión y diversidad surgen de la inquietud de encontrar las mejores ubicaciones para instalaciones no deseadas, administración de personal y en el contexto de las redes sociales, entre otros. Maximizar la diversidad se ocupa, en términos generales, de seleccionar un subconjunto de elementos de un conjunto dado de tal manera que se maximice la distancia entre los elementos seleccionados.(Martí & Reinelt, 2011; Salhi & Smith, 2015).

Uno de los objetivos que definen a la ciencia de la ubicación es maximizar la dispersión. Las instalaciones no deseables se pueden dispersar, para una amplia variedad de propósitos, incluyendo mantener separados a los competidores del mismo sistema de franquicias, dispersar las instalaciones de rehabilitación criminal de los centros de población, la ubicación de bases militares, ya que en el caso en que se dé un ataque, no las destruyan todas al mismo tiempo, y ubicar las plantas de tratamiento de aguas residuales y de energía nuclear de manera que se maximice la seguridad. Este problema involucra la selección de un número específico de lugares de un conjunto en general (lugares que deberán ser los mejores en términos de ubicación), con el fin de maximizar la mínima distancia entre los sitios seleccionados (Brimberg,2003;Cormen,2009).

El modelo más estudiado de esta clase de problemas es el Problema de la Diversidad Máxima (MDP, Maximum Diversity Problem), en el cual se maximiza la suma de las distancias entre los elementos seleccionados, es decir,

$$\max_{M \subset V; |M|=m} \sum_{i<j;\, i,j \in M} d_{ij} x_i x_j$$

Otro modelo muy documentado es el denominado Problema de la Diversidad Max-Min (MMDP, Maximum-Minimum Diversity Problem), en el cual se maximiza la mínima distancia entre los elementos del subconjunto seleccionado.

$$\max_{M \subset V; |M|=m} \left\{ \min_{i<j;\, i,j \in M} d_{ij} x_i x_j \right\}$$

Vemos que son problemas difíciles de optimización combinatoria y se han propuesto muchos métodos heurísticos y algunos algoritmos basados en meta - heurísticas

**Métodos Basados en Equidad.**

Se proponen tres modelos para maximizar la diversidad de un conjunto.

a) Max-MinSum
b) Max-Dispersión Promedio (Max-Mean)
c) Min- Dispersión diferencial (Min-diff)

**Max-MinSum**

Consiste en maximizar la medida de dispersión de la mínima suma

$$max\left\{\min_{i,\, x_i=1} \sum_{j,j\neq 1} d_{ij}x_i x_j\right\}$$

Sujeto a:

$$\sum_{i=1}^{n} x_i = m; \quad x_i \in \{0,1\}, \;\; i = 1,2,\dots,n$$

**Max-Mean**

Maximizar la diversidad promedio.

$$max\ \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_i x_j}{\sum_{i=1}^{n} x_i}$$

Sujeto a:

$$\sum_{i=1}^{n} x_i \geq 1; \quad x_i \in \{0,1\}, i = 1,2,\dots,n$$

**Min-Diff**

Minimizar la diferencia entre la mayor dispersión y la menor dispersión.

$$min\left\{\max_{i,x_i=1}\sum_{i\neq j} d_{ij}x_j - \min_{i,x_i=1}\sum_{j,j\neq i} d_{ij}x_j\right\}$$

Sujeto a:

$$\sum_{i=1}^{n} x_i = m; \quad x_i \in \{0,1\}, \quad i = 1,2,\dots,n$$

La formulación de los problemas anteriores indica que no son lineales, por lo que hay que reformular estos problemas para linealizarlo.

**Métodos Heurísticos**

Un problema de optimización consiste en encontrar, dentro de un conjunto $X_n$ de soluciones factibles, la que optimiza una función f(x). Si el problema es de minimización se formula como sigue:

$$\min\{f(x)/x' \in X_n\} \quad (1)$$

donde $x'$ representa una solución alternativa, f es la función objetivo y $X_n$ es el espacio de soluciones factibles del problema. Una solución óptima $x'$ (o mínimo global) del problema es una solución factible donde se alcanza el mínimo de (1).

Existen muchos problemas de optimización que son muy difíciles de resolver de manera exacta. Algunos son NP-duros, que se caracterizan porque no se conoce ningún algoritmo que pueda resolverlos exactamente y tenga complejidad polinomial (es decir, cuyo tiempo de resolución crezca polinomialmente con el tamaño del problema). En estos casos, se plantean algoritmos que permitan obtener soluciones cercanas al valor óptimo de la función objetivo, invirtiendo un tiempo razonable. Este tipo de algoritmos se denominan heurísticos, siendo útiles en las siguientes situaciones:

No se conoce un procedimiento exacto de resolución del problema, o bien éste requiere mucho esfuerzo computacional.

No es necesario obtener la solución óptima global del problema, conformándonos con conocer una solución cercana a dicho óptimo.

Se prefiere resolver de forma aproximada un modelo ajustado a la realidad que resolver de forma exacta un modelo aproximado de la realidad, aunque se emplee un tiempo similar.

No se dispone del suficiente conocimiento específico acerca del problema como para diseñar un método exacto de resolución. Las principales ventajas e inconvenientes que presentan las heurísticas quedan reflejadas en la tabla siguiente:

**Ventajas**

Permiten una mayor flexibilidad en el manejo de las características del problema.

Generalmente, ofrecen más de una solución, permitiendo una mayor capacidad de elección.

Suele ser más fácil de entender la fundamentación de las heurísticas que los complejos métodos matemáticos que utilizan las técnicas exactas.

**Inconvenientes.**

No siempre es posible conocer la calidad de la solución obtenida, recomendándose realizar acotaciones usando relajaciones, o bien generar varias soluciones y compararlas con la obtenida.

Dependencia de la estructura del problema considerado y falta de habilidad para adaptarse a nuevas situaciones o modificaciones del problema de partida.

Para solucionar el segundo de los inconvenientes planteados, se recomienda disponer de procedimientos heurísticos generales que puedan usarse para resolver una amplia variedad de problemas, adaptando convenientemente los elementos que los definen. Estos procedimientos heurísticos generales se denominan metaheurísticas. Dichas metaheurísticas deben cumplir una serie de propiedades para garantizar su interés tanto teórico como práctico, las cuales se enumeran a continuación:

- Simplicidad: La metaheurística debe basarse en un principio simple y claro, siendo fácil de entender y de implementar.
- Coherencia: Las heurísticas diseñadas para problemas particulares deben obtenerse de manera natural a partir de la metaheurística de la que procede.

- Eficiencia: Las heurísticas deben obtener soluciones óptimas o cercanas al óptimo para la mayoría o todos los problemas reales que pretende resolver.
- Efectividad: El tiempo computacional que empleen las heurísticas en obtener las soluciones óptimas o cercanas al óptimos debe ser moderado.
- Robustez: El rendimiento de una heurística debe ser consistente para una gran variedad de problemas; no debe estar ajustada sólo para resolver un pequeño conjunto de ellos.
- Facilidad de uso: Las heurísticas deben estar expresadas de manera clara y con el menor número de parámetros posible, y siendo fáciles de entender y de usar.
- Innovación: Las metaheurísticas, o bien las heurísticas basadas en ellas, deben aplicarse en nuevos tipos de situaciones.

**Entornos**

En el campo de la optimización real continua, se define el óptimo local de la función a optimizar a cualquier punto $x'$ para el que existe un entorno tal que $x'$ es óptimo en ese entorno. Así pues, el óptimo global podrá obtenerse examinando todos los óptimos locales que presenta la función y quedándonos con el que presente un mejor valor objetivo.

**Definición** Dado el problema $(\mathcal{S}, f)$, una estructura de entorno es una función

$$\mathcal{N}: S \to 2^S = \{X / X \subseteq S\}$$

que asocia a cada solución $x \in \mathcal{S}$ un conjunto $\mathcal{N}_k(x) \subset \mathcal{S}$ de soluciones cercanas a $x$. El conjunto $\mathcal{N}_k(x)$ se denomina entorno de $x$, mientras que cada $y \in \mathcal{N}_k(x)$ será una solución vecina de $x$. La anterior definición es bastante general, ya que el investigador es el que debe decidir cuándo considera que dos soluciones son cercanas.

En la Figura 1 se puede encontrar una representación gráfica de un posible entorno $\mathcal{N}_k(x)$ de una solución cualquiera $x$ perteneciente a un espacio de soluciones $\mathcal{S}$.

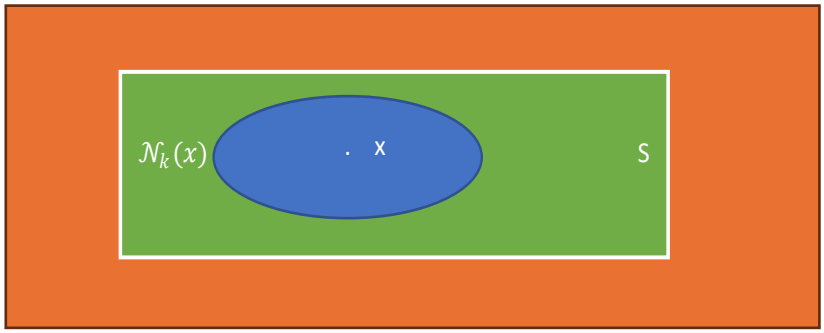


**Fig.1** Entornos de $x \in \mathcal{S}$

Si bien existen diversas estructuras de entorno que se han utilizado a la hora de resolver problemas combinatorios, las más utilizadas en problemas combinatorios y continuos son las inducidas a partir de una métrica o distancia, definida sobre el espacio de soluciones S.

Consideremos una métrica d definida sobre $\mathcal{S}$, $d: \mathcal{S} \times \mathcal{S} \longrightarrow \mathcal{R}$; ésta nos permitirá evaluar la distancia existente entre dos soluciones cualesquiera de $\mathcal{S}$. A partir de ella, podemos construir una serie de entornos de una solución dada $x$, simplemente considerando el conjunto de soluciones $x' \in \mathcal{S}$ que están a una cierta distancia de $x$. De esta manera, se pueden obtener los siguientes entornos inducidos a partir de $d$, para una solución cualquiera $x \in \mathcal{S}$.

$$\mathcal{N}_k(x) = \{x' \in \mathcal{S}: d(x, x') = k\}, k = 1,2, \ldots$$

**Método de Búsqueda de Vecindades Vecinas (VNS).**

**Búsqueda Local**

Uno de los primeros procedimientos utilizados para abordar problemas de optimización combinatoria fue la **búsqueda local**, la cual se caracteriza por realizar movimientos sucesivos en el espacio de soluciones con el objetivo de mejorar, en cada iteración, el valor de la función objetivo (Martí & Laguna, 2003; Martí & Moreno, 2002). Este enfoque constituye la base de numerosos algoritmos modernos, ya que permite explorar de manera sistemática vecindarios cercanos y, aunque puede quedar atrapado en óptimos locales, sigue siendo un punto de partida fundamental para el diseño de metaheurísticas más avanzadas.

**Algoritmo de Búsqueda Local.**

Inicialización.

Seleccionar la estructura de entorno $\mathcal{N}$ a utilizar en la búsqueda y encontrar una solución inicial $x$. Repetir los siguientes pasos hasta verificarse el criterio de parada (por ejemplo, encontrar un óptimo local):

Encontrar la mejor solución vecina $x' de\ x$ $(x' \in \mathcal{N}(x))$.Si $x'$no es mejor que $x$, parar. En otro caso, hacer $x = x'$ y volver al paso (a).

Sin embargo, el principal problema que presenta esta heurística es que, cada vez que alcanza un óptimo local, ya no puede salir de él y continuar la búsqueda. La primera mejora que se puede hacer a esta heurística consiste en repetir el procedimiento de búsqueda local considerado, partiendo de diferentes soluciones iniciales generadas aleatoriamente, almacenando la mejor solución obtenida.

Como metaheurística alternativa se presenta la Búsqueda por Entornos Variables o VNS (Variable Neighborhood Search), (Mladenović y Hansen 1997) que se trata de una técnica que intenta escapar de los óptimos locales cambiando de manera sistemática la estructura de entorno a lo largo de la búsqueda (lo que supone una evolución notable frente a la única estructura de entorno que utiliza la búsqueda local).

Esta metaheurística se clasifica dentro del grupo de los métodos de búsqueda por entornos. La ventaja de utilizar varias estructuras de entornos se basa en el hecho de que un óptimo local para un determinado entorno, no tiene por qué serlo para otro, por lo que la búsqueda podrá continuar hasta obtener una solución que sea óptimo local para todas las estructuras de entorno consideradas (Hansen & Mladenović, 2001; Osman,1996).

**Búsqueda por Entornos Variables Básica**

**Algoritmo VNS.**

**Inicialización.**

Seleccionar el conjunto de estructuras de entorno $\mathcal{N}_k, k = 1, \dots, k_{max}$ que se usarán en la búsqueda y encontrar una solución inicial $x$. Elegir, también, el criterio de parada a emplear.

Repetir los siguientes pasos hasta verificarse el criterio de parada:

1. Hacer $k = 1$.
2. Repetir los siguientes pasos hasta que $k = k_{max}$

Agitación: Generar al azar una solución $x'$ del k-ésimo entorno de $x\left(x' \in \mathcal{N}_k(x)\right)$.

Búsqueda Local: Aplicar algún procedimiento de búsqueda local partiendo de $x'$ como solución inicial. Denotar por $x''$ el óptimo local obtenido.

Moverse o no: Si la solución $x''$ mejora la mejor obtenida, hacer $x = x''$ y continuar la búsqueda con $\mathcal{N}_1$ $(k = 1)$. En otro caso, hace $k = k + 1$.

En la **Figura 2** encontramos un ejemplo en el que se observa cómo procede la VNS básica. Inicialmente, partiremos de una solución inicial x generada aleatoriamente. A continuación, se genera la solución agitada $x'$, elegida de forma aleatoria de entre todas las del entorno$\mathcal{N}_1$. A partir de $x'$ y mediante un procedimiento de búsqueda local, alcanzaremos un óptimo local $x''$. Para poder escapar de este óptimo local, es necesario incrementar el parámetro $k$ hasta que alcance el valor $k + 1$, ya que es en este momento cuando se puede obtener una solución agitada que permita alcanzar un nuevo óptimo local.

Procediendo de análoga forma para este nuevo óptimo local, podremos generar una solución agitada del entorno $\mathcal{N}_{(k+1)}$ de dicho óptimo local, que permita escapar del mismo y llegar así al óptimo global tras la aplicación de la correspondiente búsqueda local (Mladenović & Hansen, 1997; Hansen & Mladenović, 2001).

Como criterio de parada se pueden considerar varios, como, por ejemplo: el máximo tiempo de CPU permitido, un máximo número de iteraciones, o el máximo número de iteraciones entre dos mejoras del valor objetivo. Resulta interesante comentar que, en la fase de agitación, correspondiente al paso 2(a) del esquema de la VNS, se selecciona la solución

$x' \in \mathcal{N}_k(x)$ de manera aleatoria para evitar que el procedimiento sea cíclico, cosa que podría ocurrir si se hubiese empleado alguna regla determinista.

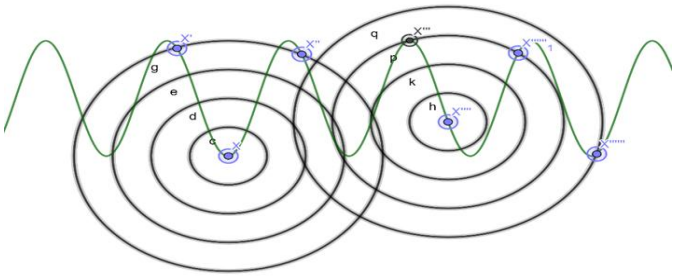

**Fig. 2.** VNS (Búsqueda de Vecindades Vecinas o Búsqueda de Entornos Variables)

**Parámetros del VNS.**

**Número máximo de vecindarios ($k_{max}$)**

Definición: cantidad de estructuras de vecindario que se explorarán en cada iteración.

**Impacto:**

- Bajo ($k_{max} = 2$): exploración limitada, riesgo de quedar atrapado en óptimos locales.
- Alto ($k_{max} = 6\ o\ mas$): mayor diversidad, pero incremento en tiempo de cómputo.
- Práctica común: valores entre 3 y 5 suelen equilibrar calidad y eficiencia.

**Tipos de vecindarios**

- Swap: intercambio de dos elementos dentro de la solución.
- Insert: mover un elemento a otra posición.
- Perturbación: cambios más grandes para escapar de óptimos locales.
- Combinados: aplicar secuencias de movimientos (ej. swap + insert).

Nota: la elección de los vecindarios debe estar alineada con la estructura del problema (dispersión, rutas, asignación).

**Solución inicial**

- Aleatoria: rápida, pero puede ser de baja calidad.
- Heurística constructiva: usa reglas simples (ej. seleccionar nodos más alejados).
- Híbrida: combinación de heurística + aleatoriedad para diversidad.
- Impacto: una buena inicialización acelera la convergencia y mejora resultados.

**Criterio de parada**

- Iteraciones máximas: número fijo (ej. 1000).
- Tiempo límite: segundos o minutos de ejecución.
- Estabilidad: detenerse si no hay mejora tras cierto número de iteraciones.
- Recomendación: usar combinación (iteraciones + estabilidad) para mayor control.

**Intensificación y diversificación**

- Intensificación: explora exhaustivamente un vecindario cuando se encuentra mejora.
- Diversificación: cambia de vecindario o aplica perturbaciones cuando no hay mejoras.
- Balance crítico: demasiado intensificación → riesgo de estancamiento; demasiada diversificación → pérdida de eficiencia.

**Criterio de aceptación**

- Estricto: solo se acepta si mejora el valor objetivo.
- Flexible: se permite aceptar soluciones ligeramente peores (ej. tolerancia del 1–2%) para escapar de óptimos locales.
- Probabilístico: aceptación basada en probabilidad (similar a Simulated Annealing).
- Impacto: define la capacidad del algoritmo para explorar y evitar estancamiento.

**Número de reinicios**

- Definición: cuántas veces se reinicia el proceso con una nueva solución inicial.
- Uso: útil en problemas grandes para aumentar cobertura del espacio de búsqueda.
- Práctica: 5–10 reinicios suelen ser suficientes en benchmarks medianos.

**Registro de desempeño**

- Traza de convergencia: valor objetivo vs. iteraciones.

- Mejor solución encontrada: valor máximo alcanzado.
- Estadísticas: media, desviación estándar, IC95% en múltiples corridas.
- Importancia: permite validar robustez y reproducibilidad.

**Problema de aplicación**

Se quiere determinar las barriadas que serán atendidas por un centro educativo, este centro educativo tendrá una capacidad máxima de M estudiantes.

**Solución:**

Tenemos una matriz que indica la cantidad de estudiantes por barriada y la distancia entre las distintas barriadas con respecto a la ubicación del centro educativo. Debemos encontrar que barriadas serán atendidas por este centro educativo.

$$min\left\{max\{\, f(x)/x \in X_n \,\}\right\} \quad \text{s.a.}$$

$$g(x) = c_1 + c_2 + c_3 + \cdots + c_n = \sum_{i=1}^{n} c_n \leq cc$$

- $A_{i,j}$ $cantidad\ de\ estudiantes\ por\ cada\ una\ de\ las\ n\ barriadas\ y$
  $distancia\ entre\ las\ barriadas\ yla\ escuela.$
- $A_{i,j} = [c_1, d_1; c_2, d_2; c_3, d_3; , \dots, c_n, d_n]$
- $N\ cantidad\ de\ vecindades\ o\ iteraciones.$
- $n\ cantidad\ de\ barriadas$
- $X_n\ conjunto\ de\ soluciones\ posibles.$
- $g(x)\ es\ la\ suma\ de\ las\ cantidades\ de\ estudiantes\ por\ barriadas.$
- $cc =$
  $capacidad\ maxima\ de\ estudiantes\ que\ puede\ atender\ en\ centro\ educativo.$

Para generar la solución inicial escogemos una ubicación para la escuela, entonces utilizamos el promedio entre la distancia máxima y las cantidades de vecindades $\left(rv = \frac{d_{max}}{N}\right)$ para establecer el radio de mi primera vecindad $\mathcal{N}$ y desde esa ubicación generamos la primera solución con punto de partida la ubicación de la escuela escogida e incluyendo las barriadas con las cuales se aproxime la capacidad del colegio, la primera

solución sera aquella barriada que la distancia sea la más cercana, y así vamos aumentando la cantidad de barriadas hasta encontrar la capacidad del centro, si no se logra esta capacidad entonces utilizamos otra vecindad con radio $d_{2m} = d_m + \alpha$; el proceso antes descrito se repite N veces , una por cada radio escogido. Después de cada iteración asociado a cada ubicación se compara y se guarda la solución global al problema.

### Problema 1

Se requiere ubicar una escuela en una región en donde el número máximo de estudiantes sea 780 estudiantes y la distancia debe ser menor que 10 km, de un grupo de 24 barriadas.

### 2.1 Algoritmo 1 PROGRAMACIÓN EN OCTAVE.

```
% Ubicación de un centro escolar utilizando VNS
clc
A=input("Ingrese la matriz A de población y distancia(km),[p(i),d(i);
...]:");
N=input("Ingrese la cantidad de vecindades:");
n=input("Ingrese la cantidad de barriadas:");
dmax=input("Ingrese la distancia máxima:");
cc=input("Ingrese la capacidad del centro:");
%n=length(A(:,2));
%Asignación de vectores de las barriadas y las distancias
for i=1:n
  p(i)=A(i,1);
  d(i)=A(i,2);
disp( 'Barriada    Población    Distancia ')
fprintf("i=%d       p(i)=%d    d(i)=%3.2f\n",i,p(i),d(i))
endfor
pa=0;
rv=dmax/N;
li=0;
ls=rv;
for i=1:N
  k(i)=0
  for j=1:n
    if d(j)>=li && d(j)<ls && p(j)<(cc-pa)
        k(i)=k(i)+1;
        o=k(i);
        s(i,o)=j
        pa=pa+p(j);
    endif
  endfor
li=li+rv;
ls=ls+rv;
endfor
disp('Las barriadas cubiertas por el colegio son:')
for l=1:N
  t=k(l);
  for m=1:t
    r=s(l,m);
  fprintf('La barriada %d con pob. de %d hab. a una dist. de %3.7f
km\n',r,p(r),d(r))
  endfor
endfor
```

**Solución:**

**Tabla 3.-** Estudiantes vs Distancia

| | Cantidad | Distancia |
|---|---|---|
| 1 | 45 | 10 |
| 2 | 68 | 15 |
| 3 | 48 | 14 |
| 4 | 24 | 20 |
| 5 | 78 | 4.4 |
| 6 | 55 | 6.5 |
| 7 | 89 | 3.2 |
| 8 | 24 | 5.6 |
| 9 | 37 | 9.2 |
| 10 | 47 | 5.9 |
| 11 | 89 | 0.9 |
| 12 | 110 | 1.6 |
| 13 | 43 | 1.2 |
| 14 | 29 | 3.5 |
| 15 | 86 | 2.6 |
| 16 | 77 | 6.8 |
| 17 | 90 | 9.7 |
| 18 | 34 | 3.4 |
| 19 | 22 | 2.6 |
| 20 | 16 | 6.3 |
| 21 | 73 | 7 |
| 22 | 55 | 4.8 |
| 23 | 35 | 6 |
| 24 | 61 | 13 |

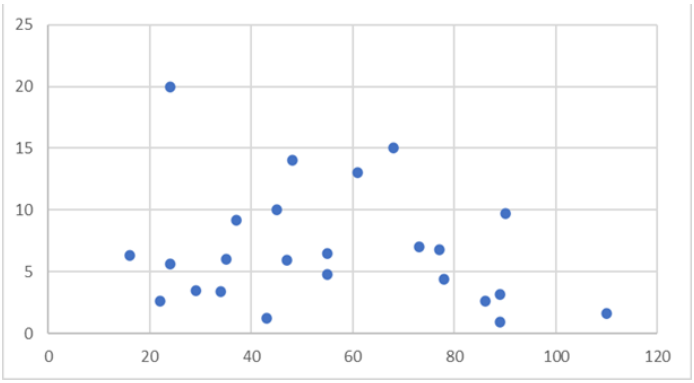


Fig. 3. Estudiantes vs Distancia

## 4. RESULTADOS Y DISCUSIÓN

### Implementación del VNS

- Solución inicial: selección aleatoria de nodos.
- Vecindarios: intercambio, inserción, perturbación.
- Función objetivo: maximizar la dispersión (distancia mínima entre pares seleccionados).
- Iteraciones: definir número máximo y criterio de parada.

## Experimentos sugeridos

- Instancias pequeñas: berlin52, eil76 → permiten validar la calidad de soluciones.
- Instancias medianas: kroA100, ch130 → muestran escalabilidad.
- Instancias grandes: pr439, pcb1173 → evalúan desempeño computacional.

## Resultados esperados

Tablas comparativas:

- Valor objetivo alcanzado por VNS vs. heurísticas básicas.
- Tiempo de ejecución promedio.
- Desviación estándar en múltiples corridas.

**Tabla 1.-** Desempeño por distancia.

| Instancia | VNS (media ± DE) | Búsqueda Local | Greedy | Tiempo medio (s) |
|---|---|---|---|---|
| berlin52 | 123.4 ± 2.1 | 118.9 ± 3.0 | 115.2 ± 4.5 | 0.45 |
| eil76 | 210.7 ± 3.2 | 202.3 ± 4.1 | 198.0 ± 5.0 | 0.80 |
| kroA100 | 315.2 ± 4.0 | 307.0 ± 5.5 | 300.5 ± 6.2 | 1.20 |

**Tabla 2.-** Análisis estadístico

| Instancia | Método | Media | Desviación estándar | IC95% (mín.–máx.) |
|---|---|---|---|---|
| berlin52 | VNS | 123.4 | 2.1 | [122.6 – 124.2] |
| | Búsqueda Local | 118.9 | 3.0 | [117.8 – 120.0] |
| | Greedy | 115.2 | 4.5 | [113.5 – 116.9] |
| eil76 | VNS | 210.7 | 3.2 | [209.5 – 211.9] |
| | Búsqueda Local | 202.3 | 4.1 | [200.9 – 203.7] |
| | Greedy | 198.0 | 5.0 | [196.3 – 199.7] |

## Interpretación de resultados

## Calidad de soluciones (valor objetivo)

El algoritmo VNS obtiene sistemáticamente valores más altos de dispersión que la búsqueda local y el greedy.

La diferencia porcentual es pequeña en instancias pequeñas (ej. berlin52), pero se amplía en instancias medianas y grandes (ej. kroA100).

Esto confirma la hipótesis H1: VNS mejora la calidad frente a métodos básicos.

**Estabilidad (desviación estándar e IC95%)**

VNS muestra menor desviación estándar, lo que significa que sus resultados son más consistentes entre corridas.

Los intervalos de confianza al 95% son más estrechos, lo que refuerza la robustez estadística del método.

Búsqueda Local y Greedy presentan mayor variabilidad, lo que indica menor confiabilidad.

**Tiempo de ejecución**

Aunque VNS es ligeramente más costoso en tiempo que Greedy, sigue siendo competitivo frente a Busqueda Local.

En instancias grandes, el tiempo crece, pero se mantiene dentro de márgenes razonables (segundos).

Esto valida la hipótesis H2: VNS logra mejor calidad sin sacrificar demasiado la eficiencia.

**Sensibilidad de parámetros**

El mejor desempeño se logra con $k_{max} = 4$ y 1000 iteraciones.

Aumentar $k_{max}$ más allá de 6 no aporta mejoras significativas y sí incrementa el tiempo.

Esto confirma la importancia de un balance entre intensificación y diversificación.

## 5. CONCLUSIONES

El presente trabajo ha demostrado la pertinencia y eficacia de la Búsqueda en Vecindarios Variables (VNS) como estrategia metaheurística para abordar el problema de dispersión, utilizando Octave como entorno de experimentación. A través de la implementación del algoritmo y su validación empírica en instancias de referencia (benchmarks como TSPLIB), se evidenció que VNS logra superar a heurísticas básicas como Greedy y Búsqueda Local, tanto en calidad de las soluciones obtenidas como en estabilidad de los resultados. La incorporación de múltiples vecindarios, junto con un adecuado balance entre intensificación y diversificación, permitió escapar de óptimos locales y alcanzar

soluciones más robustas, confirmando la hipótesis de que la variabilidad estructurada en la búsqueda es un factor decisivo para mejorar el desempeño.

Los análisis estadísticos realizados, media, desviación estándar e intervalos de confianza al 95% aportaron evidencia sólida de la superioridad de VNS. En particular, se observó que las soluciones generadas por este método presentan menor variabilidad y mayor consistencia, lo que refuerza su aplicabilidad en escenarios donde la confiabilidad es crítica.

Asimismo, los tiempos de ejecución se mantuvieron competitivos, mostrando que la mejora en calidad no implica un costo computacional prohibitivo. Esto valida la utilidad práctica del algoritmo en problemas de tamaño medio y grande, donde la eficiencia es un requisito indispensable.

Finalmente, la metodología propuesta, que integra experimentación con benchmarks reconocidos, análisis estadístico riguroso y visualización de resultados mediante tablas comparativas, ofrece un marco replicable y transparente para futuras investigaciones. El uso de Octave como plataforma abierta y accesible contribuye además a la reproducibilidad y democratización del conocimiento en optimización combinatoria. En síntesis, este estudio confirma que VNS es una herramienta poderosa para el modelado de problemas de dispersión y abre la puerta a su aplicación en otros dominios de la investigación operativa y la ciencia de datos.

## CONFLICTO DE INTERESES

Los Autores declaran que no existe conflicto de intereses

## CONTRIBUCIÓN DE AUTORÍA

| | MARIA ORTEGA | DANIEL SANCHEZ |
|---|---|---|
| Participar activamente en: | | |
| Conceptualización | X | X |
| Análisis formal | X | X |
| Adquisición de fondos | X | |
| Investigación | X | X |
| Metodología | X | X |
| Administración del proyecto | X | |
| Recursos | X | |
| Redacción borrador original | X | X |
| Redacción revisión y edición | X | X |
| La discusión de los resultados | X | X |
| Revisión y aprobación de la versión final del trabajo. | X | X |

## BIBLIOGRAFÍA

Brimberg, J., Hansen, P., Mladenović, N., & Salhi, S. (2003). A survey of solution methods for the continuous location-allocation problem. *International Journal of Operations Research, 1(2), 1–12.*

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

Glover, F., & Kochenberger, G. A. (2003). *Handbook of metaheuristics.* Springer.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research, 130* (3), 449–467. https://doi.org/10.1016/S0377-2217(00)00100-1

Hansen, P., Mladenović, N., & Brimberg, J. (2004). Variable neighborhood search. In G. Laporte, J. A. T. & F. Glover (Eds.), *Handbook of Metaheuristics* (pp. 337–360). Springer.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighborhood search: Methods and applications. *Annals of Operations Research, 175(*1), 367–407. https://doi.org/10.1007/s10479-009-0657-6

Martí, R., & Moreno, L. (2002). Scatter search for the cut width minimization problem. *INFORMS Journal on Computing, 14* (1), 43–52. https://doi.org/10.1287/ijoc.14.1.43.93

Martí, R., & Reinelt, G. (2011). *The linear ordering problem: Exact and heuristic methods in combinatorial optimization.* Springer.

Martí, R., & Laguna, M. (2003). *Scatter search: Methodology and implementations in C*. Springer.

Michalewicz, Z., & Fogel, D. B. (2004). *How to solve it: Modern heuristics*. Springer.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097–1100. https://doi.org/10.1016/S0305-0548(97)00031-2

Octave Community. (2023). *GNU Octave: A high-level interactive language for numerical computations*. https://www.gnu.org/software/octave/

Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research, 63* (5), 513–623. https://doi.org/10.1007/BF02601646

Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing, 3* (4), 376–384. https://doi.org/10.1287/ijoc.3.4.376

Salhi, S., & Smith, G. (2015). Heuristic methods for the p-center problem. *Journal of the Operational Research Society, 66*(1), 1–12. https://doi.org/10.1057/jors.2014.20

Talbi, E. G. (2009). *Metaheuristics: From design to implementation*. Wiley.