

PROBLEMA DE MÁXIMA DIVERSIDAD APLICANDO HEURÍSTICAS



**UNIVERSIDAD COMPLUTENSE
DE MADRID**

FACULTAD DE CIENCIAS MATEMÁTICAS
MÁSTER EN INGENIERÍA MATEMÁTICA

ASIGNATURA: Técnicas Avanzadas en Optimización

PROFESOR: Gregorio Tirado Dominguez

ALUMNAS: Ana Marta Oliveira dos Santos,
Siria Catherine Íñiguez Brito

CURSO: 2025-2026

Madrid, Diciembre de 2025

Contents

1	INTRODUCCIÓN	3
2	PROBLEMA DE MÁXIMA DIVERSIDAD	4
2.1	Descripción del Problema	4
2.2	Formulación Matemática	4
3	ALGORITMOS IMPLEMENTADOS	5
3.1	Algoritmo Constructivo	5
3.1.1	Resultados del Algoritmo Constructivo	6
3.2	Búsqueda Local	8
3.2.1	Resultados Búsqueda local	10
3.3	Temple Simulado	11
3.3.1	Calibración TS	13
3.3.2	Resultados Temple Simulado	16
3.4	Algoritmos Genéticos	18
3.4.1	Calibración AG	20
3.4.2	Resultados AG	32
3.5	Búsqueda en Entorno Variable	34
3.5.1	Calibración VNS	36
3.5.2	Resultados VNS	37
4	COMPARACIÓN DE ALGORITMOS	39
5	CONCLUSIONES	41

List of Tables

1	Resultados del Algoritmo Constructivo (Parte-1)	6
2	Resultados del Algoritmo Constructivo (Parte-2)	7
3	Resultados de la Búsqueda Local	10
4	Resultados de Calibración del Temple Simulado	14
5	Resultados de Calibración de α	15
6	Cambio en <i>iter</i>	15
7	Comparativa de Múltiples Errores de Calibración	16
8	Comparación del resultado medio y mejor resultado del Temple Simulado (SA) vs. Mejor Valor (Best Value)	17
9	Resultados de la calibracion de Probabilidad de Mutación (Parte 1) .	20
10	Resultados de la calibración de Probabilidad de Mutación (Parte 2) .	21
11	Resultados de la calibración de Probabilidad de Cruce (Parte 1) . . .	22
12	Resultados de la calibración de Probabilidad de Cruce (Parte 2) . . .	23
13	Resultados calibración combinada PM-PC (Parte-1)	24
14	Resultados completos calibración PM-PC - (Parte-2)	25
15	Resultados completos calibración PM-PC - (Parte 3)	26
16	Resultados completos calibración PM-PC - (Parte 4)	27
17	Resultados Calibración del Tamaño de la población (Parte 1)	28
18	Resultados Calibración del Tamaño de la población (Parte-2)	29
19	Resultados Calibración del Número de Generaciones	30
20	Resultados del Algoritmo Genético (AG)	32
21	Comparativa de rendimiento: Algoritmo Genético vs Temple Simulado	32
22	Resultados del Algoritmo Genético Greedy (AGG)	33
23	Impacto de la inicialización greedy en el rendimiento del algoritmo . .	33
24	Resultados de calibración de Niveles de Perturbación	36
25	Resultados del Algoritmo VNS	37
26	Comparación Mejores Resultados (Error %)	39
27	Comparación Media Resultados (Error %)	39

1 INTRODUCCIÓN

El presente trabajo aborda la implementación y el análisis comparativo de diversas técnicas algorítmicas de optimización para resolver el **Problema de Máxima Diversidad (MDP)**. Este problema representa un desafío fundamental en optimización combinatoria, clasificado como **NP-duro**, cuyo objetivo consiste en seleccionar un subconjunto de m elementos de un conjunto total de n elementos, maximizando la suma de las distancias entre los elementos seleccionados.

Este estudio se centra en el desarrollo, implementación y evaluación exhaustiva de algoritmos heurísticos y metaheurísticos. El objetivo principal es identificar estrategias capaces de obtener soluciones de alta calidad dentro de límites de tiempo computacional razonables. Para ello, se han implementado cinco categorías de algoritmos:

- **Algoritmos constructivos:** Para la generación inicial de soluciones factibles.
- **Búsqueda local:** Como método de mejora iterativa basado en entornos de vecindad.
- **Temple simulado:** Como metaheurística inspirada en procesos físicos.
- **Algoritmos genéticos:** Como técnicas evolutivas basadas en poblaciones.
- **Búsqueda en entorno variable:** Como estrategia de cambio sistemático de entorno.

El trabajo se estructura en tres partes principales: primero, la descripción de cada algoritmo y su adaptación al MDP; segundo, un proceso de calibración de parámetros para optimizar el rendimiento de cada método; y tercero, una evaluación comparativa sobre un conjunto de instancias de referencia las cuales se encuentran disponibles en <https://zenodo.org/records/13749317>. Además, cabe destacar que todos los algoritmos han sido implementados en Fortran 90, generando los siguientes archivos: `Constructivo.f90`, `Busqueda_Local.f90`, `Temple_Simulado.f90`, `Algoritmo_AG.f90` y `Algoritmo_VNS.f90`.

2 PROBLEMA DE MÁXIMA DIVERSIDAD

2.1 Descripción del Problema

El **Problema de Máxima Diversidad (MDP)** es un problema fundamental en optimización combinatoria. Su objetivo primordial radica en **seleccionar un subconjunto de m elementos** ($m < n$) de un conjunto total de n elementos disponibles, buscando que la **suma total de las distancias** entre todos los pares de elementos seleccionados sea la máxima posible. La meta intrínseca del MDP es la selección de un grupo lo más **heterogéneo y diverso** posible.

La viabilidad de esta formulación depende de la correcta definición de la **distancia** (d_{ij}) entre dos elementos i y j . Esta definición se personaliza según la aplicación específica (e.g., selección de personal, diseño experimental, ubicación de facilidades). Este valor d_{ij} cuantifica la disimilitud entre cada par de elementos y es la base para la función objetivo del modelo de optimización.

2.2 Formulación Matemática

La formulación del Problema de Máxima Diversidad (MDP) se establece como un **Problema de Programación Cuadrática Binaria (PQB)**, dada la naturaleza del objetivo (producto de variables) y las restricciones de integridad.

El modelo se fundamenta en la definición de una variable de decisión binaria, x_i , para cada elemento i del conjunto total, donde $i \in \{1, \dots, n\}$:

$$x_i = \begin{cases} 1 & \text{si el elemento } i \text{ es seleccionado en el subconjunto,} \\ 0 & \text{en caso contrario.} \end{cases} \quad \forall i \in \{1, \dots, n\}.$$

Utilizando los valores de disimilitud d_{ij} , el modelo se construye para maximizar la diversidad total sujeta a la restricción del tamaño del subconjunto (m):

$$\begin{aligned} \max \quad & Z = \sum_{i=1}^n \sum_{j=i+1}^n d_{ij} x_i x_j && \text{(Maximizar la diversidad)} \\ \text{s.a.} \quad & \sum_{i=1}^n x_i = m && \text{(Restricción de tamaño del subconjunto } m \text{ elementos)} \\ & x_i \in \{0, 1\} && \forall i = 1, \dots, n \quad \text{(Variables binarias)} \end{aligned}$$

La función objetivo, Z , es de naturaleza cuadrática debido al término $x_i x_j$. El producto $x_i x_j$ toma el valor 1 si, y solo si, ambos elementos i y j son seleccionados (es decir, $x_i = 1$ y $x_j = 1$). De esta manera, d_{ij} solo se suma a la diversidad total Z cuando el par de elementos (i, j) está presente en la solución, logrando la maximización de la distancia acumulada entre los elementos elegidos.

La restricción $\sum_{i=1}^n x_i = m$ garantiza que el número exacto de elementos seleccionados en la solución sea igual al tamaño requerido m . La restricción $x_i \in \{0, 1\}$ es la condición de integridad binaria para este problema de selección.

3 ALGORITMOS IMPLEMENTADOS

3.1 Algoritmo Constructivo

Una de las heurísticas clásicas empleadas en este trabajo es el algoritmo constructivo. Este tipo de heurísticas construye una solución factible completa de manera iterativa, añadiendo en cada paso un nuevo elemento a la solución parcial hasta completarla. La selección del elemento a incorporar se basa en un criterio específico que orienta el proceso hacia soluciones de "buena" calidad.

En general, los algoritmos constructivos por sí solos no suelen proporcionar soluciones especialmente buenas. Sin embargo, desempeñan un papel fundamental como generadores de soluciones iniciales en algoritmos de búsqueda local y en metaheurísticas. Por este motivo, es importante que sean rápidos, de baja complejidad computacional, y capaces de generar soluciones diversas de manera consistente.

Para el problema de la máxima diversidad, (MDP), cuyo objetivo es maximizar la diversidad total del subconjunto seleccionado, se ha elegido que el algoritmo añada en cada iteración el elemento que presente mayor contribución de diversidad respecto a los elementos ya incluidos.

Así, en este trabajo, se ha implementado un algoritmo constructivo *greedy* con reinicio, diseñado para tener una complejidad $O(nm)$.

Los pasos implementados son los siguientes:

Paso 0: Primero se inicializan las variables: Se pone $t = 1$, y se elige un elemento $p_1 \in \{1, \dots, N\}$. Se establece el conjunto de elementos elegidos como $P = \{p_1\}$, y el conjunto de elementos candidatos restantes como $I = \{1, \dots, n\} \setminus \{p_1\}$. Se inicializa la diversidad total a cero ($z = 0$). Se calcula la diversidad inicial $g(i) = d_{ip_1} \quad \forall i \in I$, o sea, para cada elemento restante.

Paso 1: Si $t = M$ PARAR : ya se tiene una solución factible, es decir, el número de elementos elegidos es igual al objetivo (M). Si $t < M$: ir al Paso 2.

Paso 2: Hallar el elemento k que maximiza la diversidad entre los candidatos restantes: $k = \arg \max \{g(i) \mid i \in I\}$. Se actualiza el contador $t = t + 1$, $p_t = k$, y se añade el nuevo elemento al conjunto de elegidos $P = P \cup \{p_t\}$, y se lo elimina del conjunto de candidatos $I = I \setminus \{k\}$. Por fin, se actualiza la diversidad total de la solución construida hasta el momento: $z = z + g(k)$. Finalmente, se actualizan las diversidades asociadas a cada elemento restante: $g(i) = g(i) + d_{ik} \quad \forall i \in I$. Se vuelve al Paso 1.

Algorithm 1 Greedy_con_Reinicio

```
1: Variables:  $p, z$ 
2: Estructuras: Distancias,  $g$ , Ind, Pc, sub
3:
4: 0. Inicialización
5:  $p \leftarrow \text{inicializado}$ 
6:  $\text{Ind} \leftarrow 1; \quad \text{Pc} \leftarrow 0$ 
7:  $t \leftarrow 1; \quad z \leftarrow 0.0$ 
8:  $\text{Pc}(p) \leftarrow 1; \quad \text{Ind}(p) \leftarrow 0$ 
9: for  $i \leftarrow 1$  to  $N$  do
10:    $g(i) \leftarrow \text{Distancias}(i, p)$ 
11: end for
12:
13: 2. Bucle Principal
14: for  $t \leftarrow 1$  to  $M - 1$  do
15:    $k \leftarrow \text{índice del máximo de } g$ 
16:    $\text{Pc}(k) \leftarrow 1; \quad \text{Ind}(k) \leftarrow 0$ 
17:    $z \leftarrow z + g(k)$ 
18:   for  $i \leftarrow 1$  to  $N$  do
19:     if  $\text{Ind}(i) = 1$  then
20:        $g(i) \leftarrow g(i) + \text{Distancias}(i, k)$ 
21:     else
22:        $g(i) \leftarrow 0.0$ 
23:     end if
24:   end for
25: end for
26: return  $z = 0$ 
```

3.1.1 Resultados del Algoritmo Constructivo

Para obtener los resultados experimentales, se han realizado $\lfloor \frac{N}{M} \rfloor$ ejecuciones independientes. En cada ejecución, el primer elemento p_1 seleccionado ha sido distinto $(1, 2, \dots, \lfloor \frac{N}{M} \rfloor)$, con el fin de generar soluciones iniciales variadas. Para la comparación final, se ha utilizado la media de los valores de diversidad obtenidos en todas las ejecuciones.

Los resultados obtenidos se pueden ver en las Tablas 1 y 2.

Instancia	Best Value	Constructivo	Error(%)
GKD-c_19 n500_m50.txt	19477,3281	19415,4844	0,318
GKD-c_20 n500_m50.txt	19604,8438	19542,69917	0,317
SOM-B_4 n100_m40.txt	4142	4077	1,57
SOM-B_8 n200_m80.txt	16225	16066,5	0,977
SOM-B_12 n300_m120.txt	35881	35513,5	1,02
SOM-B_16 n400_m160.txt	62487	62015	0,755
SOM-B_20 n500_m200.txt	97344	96302,5	1,07

Table 1: Resultados del Algoritmo Constructivo (Parte-1)

Instancia	Best Value	Constructivo	Error(%)
MDG-a_19 n500_m50.txt	7755,41064	7529,597	2,91
MDG-a_20 n500_m50.txt	7733,86133	7506,876	2,93
MDG-a_39 n2000_m200.txt	114201	112192,4	1,76
MDG-a_40 n2000_m200.txt	114191	112437,6	1,54
MDG-B_19 n500_m50.txt	778802,938	754375,97	3,14
MDG-B_20 n500_m50.txt	778644,813	754201,403	3,14
MDG-B_39 n2000_m200.txt	11295054,2	1119843,51	1,55
MDG-B_40 n2000_m200.txt	11307104,8	11124380,17	1,62
MDG-C_5 n3000_m300.txt	24899703	24577685,1	1,29
MDG-C_10 n3000_m400.txt	43476251	42979017,14	1,14
MDG-C_15 n3000_m500.txt	66992877	66432196,67	0,837
MDG-C_20 n3000_m600.txt	95643586	94924244,4	0,752

Table 2: Resultados del Algoritmo Constructivo (Parte-2)

Analizando los resultados, se observa que el rendimiento del algoritmo es consistentemente bueno a través de los diferentes conjuntos de datos.

Mejores Resultados (GKD-c): Las instancias del conjunto GKD-c, ambas con $n = 500$, $m = 50$, arrojaron los mejores errores, con valores de 0,318% y 0,317%.

Mayores Errores (MDG-b): Por otro lado, las instancias MDG-b, también con $n = 500$, $m = 50$, son las que presentan los mayores errores relativos, con valores que superar el 3% (3,18%).

Rendimiento General: Considerando que el algoritmo es una heurística clásica constructiva, los resultados son notablemente buenos. Al revisar la columna de Error(%), se puede constatar que casi un tercio de las instancias tienen un error inferior al 1%.

Nota: La comparación es con respecto al mejor valor conocido. El hecho de que la heurística constructiva obtenga resultados con errores tan bajos ($< 3,14\%$ en el peor caso de la tabla) indica que genera soluciones de calidad considerable, lo cual es un excelente punto de partida para una posterior mejora mediante una metaheurística.

3.2 Búsqueda Local

Otra heurística clásica utilizada en este trabajo es el algoritmo de búsqueda local. En este enfoque, los algoritmos de mejora parten de una solución inicial y la modifican sucesivamente con el objetivo de obtener soluciones factibles completas de mejor calidad. Estos algoritmos no exploran todo el espacio de soluciones, sino únicamente el entorno definido por pequeños movimientos aplicados a la solución actual. Aunque suelen ser razonablemente rápidos, presentan la desventaja de que pueden estancarse en óptimos locales.

En este trabajo se ha implementado una búsqueda local de mayor mejora. En este esquema, se evalúan todos los movimientos posibles y se selecciona aquel que proporcione la mayor mejora respecto a la solución actual.

El vecindario empleado se ha definido mediante movimientos de intercambio simple, es decir, sustituyendo un elemento del subconjunto por otro que no pertenece a él. El proceso comienza a partir de una solución inicial y, entre todos los vecinos generados, se selecciona aquel que produce la mayor mejora, entendida como el mayor incremento en el valor de la función objetivo en comparación con la solución previa.

La implementación sigue los siguientes pasos:

Paso 0: Inicialización: Leer el conjunto de elementos elegidos inicial $P = \{p_1, \dots, p_m\}$.

Calcular la diversidad inicial $z = \sum_{j=1}^{m-1} \sum_{k=j+1}^m d_{p_j p_k}$.

Paso 1: Reiniciar Búsqueda Local: Poner el máximo de mejora encontrado $g_{max} = 0$, y el indicador de óptimo local $elopt = 1$.

Paso 2: Iterar Intercambios: $\forall i \in \{1, \dots, m\}$:

Paso 2.1: Cálculo de pérdida: Calcular la pérdida de diversidad v_i al eliminar p_i :

$$v_i = \sum_{j=1, j \neq i}^m d_{p_j p_i}$$

Paso 2.2: Cálculo de ganancia e intercambio: $\forall k \in N \setminus P$:

i. Calcular la ganancia de diversidad d_k al añadir k (intercambio $p_i \leftrightarrow k$): $d_k = \sum_{j=1, j \neq i}^m d_{p_j k}$

ii. Si la ganancia $\delta = d_k - v_i > 0$ (es una mejora):

Poner $elopt = 0$; $\delta > g_{max}$ (Si esta es la mejor mejora encontrada hasta ahora):

- Actualizar los índices y la mejora máxima: $i_{max} = i$, $k_{max} = k$, $g_{max} = \delta$.

Si $d_k \leq v_i$ (No es una mejora): Continuar con el siguiente elemento k .

Paso 3:

- Si $elopt = 0$: Aplicar el mejor intercambio ($p_{i_{max}} \leftrightarrow k_{max}$). Poner $p_{i_{max}} = k_{max}$ y actualizar la diversidad total $z = z + g_{max}$. Ir al Paso 1 (reiniciar la búsqueda).

- Si $elopt = 1$: La solución actual P es óptimo local respecto a los entornos definidos. Parar.

Algorithm 2 Busqueda_Local

```
1: Variables:  $i_{\max}, k_{\max}, \text{elopt}, p, z_{\text{final}}, z, dk\_i, g_{\max}, \text{aleatorio}$ 
2: Estructuras: Distancias,  $S, P_{\text{final}}, P_{\text{cur}}, \text{enS}, v, \text{totalA}$ 
3:
4: 0. Inicialización
5:  $p \leftarrow \lfloor \text{aleatorio} \times (N - M + 1) \rfloor + 1$ 
6:  $\text{enS} \leftarrow 0$ 
7: for  $i \leftarrow 1$  to  $M$  do
8:    $S(i) \leftarrow p + i - 1$ 
9:    $\text{enS}(S(i)) \leftarrow 1$ 
10: end for
11:  $P_{\text{cur}} \leftarrow S; \quad z \leftarrow -\text{Coste}(P_{\text{cur}})$ 
12: 2.LOOP_START: {Inicio del ciclo de búsqueda local}
13:  $g_{\max} \leftarrow 0.0; \quad \text{elopt} \leftarrow 1; \quad i_{\max} \leftarrow -1; k_{\max} \leftarrow -1$ 
14: for  $i \leftarrow 1$  to  $M$  do
15:    $v(i) \leftarrow 0.0$ 
16:   for  $j \leftarrow 1$  to  $M$  do
17:     if  $j \neq i$  then
18:        $v(i) \leftarrow v(i) + \text{Distancias}(P_{\text{cur}}(j), P_{\text{cur}}(i))$ 
19:     end if
20:   end for
21: end for
22: for  $k \leftarrow 1$  to  $N$  do
23:    $\text{totalA}(k) \leftarrow 0.0$ 
24:   for  $j \leftarrow 1$  to  $M$  do
25:      $\text{totalA}(k) \leftarrow \text{totalA}(k) + \text{Distancias}(P_{\text{cur}}(j), k)$ 
26:   end for
27: end for
28: for  $i \leftarrow 1$  to  $M$  do
29:   for  $k \leftarrow 1$  to  $N$  do
30:     if  $\text{enS}(k) = 0$  then
31:        $dk\_i \leftarrow \text{totalA}(k) - \text{Distancias}(P_{\text{cur}}(i), k)$ 
32:       if  $dk\_i > v(i)$  then
33:          $\text{elopt} \leftarrow 0$ 
34:         if  $dk\_i - v(i) > g_{\max}$  then
35:            $g_{\max} \leftarrow dk\_i - v(i); \quad i_{\max} \leftarrow i; \quad k_{\max} \leftarrow k$ 
36:         end if
37:       end if
38:     end if
39:   end for
40: end for
41: if  $\text{elopt} = 0$  then
42:    $\text{enS}(P_{\text{cur}}(i_{\max})) \leftarrow 0; \quad P_{\text{cur}}(i_{\max}) \leftarrow k_{\max}; \quad \text{enS}(k_{\max}) \leftarrow 1; \quad z \leftarrow z + g_{\max}$ 
43:   GOTO LOOP_START
44: else
45:    $P_{\text{final}} \leftarrow P_{\text{cur}}; \quad z_{\text{final}} \leftarrow z$ 
46: end if
47: return  $z_{\text{final}} = 0$ 
```

3.2.1 Resultados Búsqueda local

Para obtener los resultados presentados, el algoritmo de Búsqueda Local se ha ejecutado una única vez. La solución inicial se generó mediante un procedimiento constructivo simple, donde se selecciona aleatoriamente un índice p tal que $p \in \{1, \dots, N - M + 1\}$. La solución inicial es así formada por el conjunto de elementos consecutivos $\{p, p + 1, \dots, p + M - 1\}$, donde N es el tamaño total de la población y M es el tamaño de la solución deseada.

Instance	Best Value	BusquedaLocal	Error(%)
GKD-c_19 n500_m50.txt	19477,3281	19477,32577	0,0000121
GKD-c_20 n500_m50.txt	19604,8438	19604,84356	0,000000969
SOM-B_4 n100_m40.txt	4142	4142	0
SOM-B_8 n200_m80.txt	16225	16180	0,277
SOM-B_12 n300_m120.txt	35881	35801	0,223
SOM-B_16 n400_m160.txt	62487	62235	0,403
SOM-B_20 n500_m200.txt	97344	97073	0,278
MDG-a_19 n500_m50.txt	7755,41064	7593,34	2,09
MDG-a_20 n500_m50.txt	7733,86133	7537,01	2,55
MDG-a_39 n2000_m200.txt	114201	112762	1,26
MDG-a_40 n2000_m200.txt	114191	113286	0,793
MDG-b_19 n500_m50.txt	778802,938	755611,81	2,98
MDG-b_20 n500_m50.txt	778644,813	770822,87	1
MDG-b_39 n2000_m200.txt	11295054,2	11173801,16	1,07
MDG-b_40 n2000_m200.txt	11307104,8	11158022,87	1,32
MDG-c_5 n3000_m300.txt	24899703	24697801	0,811
MDG-c_10 n3000_m400.txt	43476251	43164066	0,718
MDG-c_15 n3000_m500.txt	66992877	66766856	0,337
MDG-c_20 n3000_m600.txt	95643586	95212553	0,451

Table 3: Resultados de la Búsqueda Local

El rendimiento del algoritmo de Búsqueda Local ha sido bastante bueno, lo cual indica que la fase de mejora local fue efectiva:

Mejores Resultados: Las instancias del conjunto GKD-c siguen ofreciendo los mejores resultados, alcanzando errores extremadamente bajos (cercaos al 0,0%). De hecho, la instancia SOM-b_4 alcanza un error de 0,0%, lo que significa que el algoritmo de Búsqueda Local encontró el valor óptimo conocido.

Mayores errores: Estos se observan en las instancias MDG-a_19 (2,09%) y MDG-a_20 (2,55%), ambas con un tamaño de $N = 500$, $M = 50$.

Rendimiento Global: De las 19 instancias probadas, 12 de ellas obtuvieron un error inferior al 1%. En general, es un rendimiento destacable para una heurística. La técnica se muestra eficiente y rápida, entregando soluciones de calidad. Este buen desempeño se mantiene incluso para las instancias de mayor tamaño del conjunto MDG-c, donde los errores se mantienen consistentemente por debajo del 1%.

3.3 Temple Simulado

En esta sección se emplea la metaheurística del Temple Simulado (SA), conocida en inglés como *Simulated Annealing*.

Para implementar el algoritmo, es necesario definir varios elementos que deben adaptarse al MDP.

- **Conjunto Factible:** El conjunto factible S está formado por todos los subconjuntos de M elementos distintos seleccionados de un conjunto de tamaño N . Se trata, por tanto, de un conjunto de definición sencilla.
- **Solución inicial:** El Temple Simulado requiere una solución inicial. Dado que en las primeras fases del algoritmo la temperatura es elevada y, por tanto, la diversificación es alta, la calidad inicial no condiciona significativamente el resultado final. Por este motivo, se ha optado por generar una solución inicial completamente aleatoria, seleccionando M elementos distintos, todos aleatorios. Para ello, se utiliza la subrutina definida **ConstruirInicial**.
- **Estructura del vecindario:** A diferencia de la búsqueda local, el Temple Simulado no explora exhaustivamente el vecindario, sino que genera un único vecino por iteración. El vecindario empleado se basa en un intercambio simple: se elige aleatoriamente dos elementos: un elemento de la solución actual para eliminar y un elemento fuera de la solución para incluir. La subrutina que realiza dicho intercambio es **GenerarVecino**.
- **Función de Coste:** La función de coste implementada **Coste** corresponde a la función objetivo del MDP definida anteriormente, pero con el signo opuesto: $Coste = -Diversidad$. El objetivo sigue siendo maximizar la diversidad, sin embargo, formularlo como minimización permite aplicar directamente la lógica del SA. El cálculo del coste tras un intercambio se realiza de forma eficiente, reutilizando las ideas implementadas en la búsqueda local: se suma la aportación del elemento añadido y se resta la aportación del elemento eliminado: función **CosteVecino**.

La implementación del SA requiere también la definición de varios parámetros fundamentales: temperatura inicial, esquema de disminución de la temperatura, número de iteraciones por temperatura y criterio de parada.

Para la disminución de la temperatura en cada iteración se ha utilizado:

$$T \leftarrow \alpha T, \quad \alpha < 1 \text{ y } \alpha \approx 1.$$

El comportamiento del Temple Simulado se basa en un equilibrio entre diversificación e intensificación :

- A temperaturas altas el algoritmo permite aceptar movimientos que empeoran la solución, lo que favorece explorar zonas amplias y diversas del espacio de búsqueda. Cuanto mayor sea la temperatura y menor la diferencia de costes, mayor será la probabilidad de aceptar soluciones peores.
- A temperaturas bajas, el algoritmo se vuelve más conservador y tiende a aceptar únicamente mejoras o pequeños empeoramientos. Esto produce un comportamiento de intensificación, centrado en refinar la solución en un entorno reducido.

Los pasos que componen su funcionamiento son:

Paso 0: Construcción de la Solución Inicial: Se genera una solución factible utilizando la subrutina **ConstruirInicial**. Esta solución se establece como solución actual. Se inicializa también como la mejor solución conocida hasta el momento y se registra su coste.

Paso 1: Bucle Principal: Se mantiene activo mientras no se cumplan los criterios de parada definidos (T_{min} y número de niveles de temperatura sin mejora).

Paso 2: Dentro del bucle principal, se inicia el bucle de nivel de temperatura que realiza la exploración para la temperatura actual T . Se genera un vecino con **GenerarVecino**, y se calcula la diferencia de costes δ , mediante **CosteVecino**.

- Si $\delta < 0$: La solución vecina es mejor. Se acepta inmediatamente el vecino generado como la nueva solución actual. Además, si su coste es menor que el coste de la mejor solución encontrada hasta el momento, se pone esta como la mejor hasta ahora.
- Si $\delta \geq 0$: La solución vecina no es mejor que la solución actual. Se genera un número aleatorio $u \in [0, 1]$. La solución peor es aceptada con una probabilidad de $P = e^{-\delta/T}$, Si $u \leq P$, se acepta la solución y se establece como la nueva solución actual.

Paso 3: Al finalizar el bucle a temperatura constante, se disminuye la temperatura actual T siguiendo la regla de enfriamiento previamente definida: $T = \alpha \cdot T$. Volver al Paso 1.

Algorithm 3 Temple_Simulado

```
1: Parámetros SA (Fijos):  $T_{\text{inicial}}, T_{\text{final}}, \alpha, \text{iter}, \text{sin\_mejora}$ 
2: Variables:  $\text{niter}, p, \text{mejora}, k_{\text{salida\_val}}, \text{val\_entra}, T, C_{\text{actual}}, C_{\text{vecina}}, \delta, u$ 
3: Estructuras: Distancias ,  $x_{\text{actual}}, x_{\text{vecina}}, x_{\text{opt}}$ 
4:
5: ConstruirInicial( $N, M, x_{\text{actual}}$ )
6:  $x_{\text{opt}} \leftarrow x_{\text{actual}}; C_{\text{actual}} \leftarrow \text{Coste}(x_{\text{actual}}); T \leftarrow T_{\text{inicial}}; \text{mejora} \leftarrow 0$ 
7: while  $T > T_{\text{final}}$  AND  $\text{mejora} < \text{sin\_mejora}$  do
8:   for  $\text{niter} \leftarrow 1$  to  $\text{iter}$  do
9:     GenerarVecino( $x_{\text{actual}}, x_{\text{vecina}}, k_{\text{salida\_val}}, \text{val\_entra}$ )
10:     $\delta \leftarrow \text{CosteVecino}(x_{\text{actual}}, k_{\text{salida\_val}}, \text{val\_entra})$ 
11:     $C_{\text{vecina}} \leftarrow C_{\text{actual}} + \delta$ 
12:    if  $\delta < 0.0$  then
13:       $x_{\text{actual}} \leftarrow x_{\text{vecina}}; C_{\text{actual}} \leftarrow C_{\text{vecina}}$ 
14:      if  $C_{\text{actual}} < \text{Coste}(x_{\text{opt}})$  then
15:         $x_{\text{opt}} \leftarrow x_{\text{actual}}; \text{mejora} \leftarrow 0$ 
16:      end if
17:    else
18:       $u \leftarrow \text{numero aleatorio}$ 
19:      if  $u < \exp(-\delta/T)$  then
20:         $x_{\text{actual}} \leftarrow x_{\text{vecina}}$ 
21:         $C_{\text{actual}} \leftarrow C_{\text{vecina}}$ 
22:      end if
23:    end if
24:  end for
25:   $\text{mejora} \leftarrow \text{mejora} + 1$ 
26:   $T \leftarrow T \times \alpha$ 
27: end while
28: return -  $\text{Coste}(x_{\text{opt}}) = 0$ 
```

3.3.1 Calibración TS

El éxito del algoritmo depende en gran medida de la calibración de sus parámetros, especialmente la temperatura inicial, la tasa de enfriamiento y los criterios de parada. Esta calibración suele ser uno de los aspectos más complejos del método. Por ello, en este trabajo se realizará un estudio detallado de los parámetros del algoritmo. A continuación, se describe el proceso de calibración de los parámetros del Temple Simulado.

Para cada instancia del MDP los resultados presentados, para cada conjuntos de parámetros son una media de 5 ejecuciones independientes. Posteriormente, los resultados se compararon con los valores óptimos conocidos, obteniendo así el error presentado. Con el fin de facilitar la interpretación, en las tablas de esta sección únicamente se incluyen el nombre de las instancias, y el error asociado a cada experimento.

Los parámetros analizados fueron los siguientes:

- T : Temperatura inicial
- T_{final} : temperatura final, empleada como criterio de parada.
- α : tasa de enfriamiento
- $iter$: número de iteraciones por nivel de temperatura
- sin_mejora : número máxima de niveles de temperatura consecutivos sin mejora permitidos (criterio adicional de parada).

Variación de la temperatura inicial

Para iniciar la calibración se realizó un análisis preliminar sobre el efecto de la temperatura inicial. En esta fase no se ha tenido en cuenta el criterio de parada sin_mejora , para este apenas se ha tomado un valor bastante elevado para que no tenga gran influencia.

Se toma un valor base $T = 100$, y se comparó con los valores $T = 80$ y $T = 120$. El resto de parámetros se mantuvo constante: $T_{final} = 0.001$, $\alpha = 0.85$, $iter = 40$.

$T_{final} = 0.001, \alpha = 0.85, iter = 40$			
Instance	T=80	T=100	T=120
GKD-c_1 n500_m50.txt	0,443	0,516	0,604
GKD-c_2 n500_m50.txt	0,616	0,612	0,788
SOM-b_1 n100_m10.txt	4,5	3,604	6,01
SOM-b_2 n100_m20.txt	2,63	2,59	1,76
MDG-a_21 n2000_m200.txt	4,52	4,71	4,7
MDG-a_22 n2000_m200.txt	4,65	4,59	4,69
MDG-b_1 n500_m50.txt	4,63	4,27	4,71
MDG-c_1 n3000_m300.txt	4,05	3,84	3,92

Table 4: Resultados de Calibración del Temple Simulado

Los resultados muestran que, en general, $T = 100$ ofrece un rendimiento ligeramente mejor, aunque la diferencia con las otras temperaturas no es muy pronunciada. Se observa también que incrementar la temperatura no siempre produce mejoras, lo que sugiere que un T demasiado alto puede generar una exploración excesiva y poco eficiente. Por estos motivos, se decidió mantener $T=100$ para los siguientes experimentos.

Variación de α y número de iteraciones

El siguiente objetivo fue analizar el efecto del parámetro de enfriamiento α y del número de iteraciones por temperatura $iter$.

$T = 100, T_f = 0.001, iter = 40$		
Instance	$\alpha = 90$	$\alpha = 95$
GKD-c.1 n500_m50.txt	0,292	0,192
GKD-c.2 n500_m50.txt	0,0955	0,0845
SOM-b.1 n100_m10.txt	3,18	3,54
SOM-b.2 n100_m20.txt	1,44	1,31
MDG-a.21 n2000_m200.txt	3,82	2,9
MDG-a.22 n2000_m200.txt	4,01	3,04
MDG-b.1 n500_m50.txt	3,64	2,72
MDG-c.1 n3000_m300.txt	3,39	2,41

Table 5: Resultados de Calibración de α

$T = 100, T_f = 0.001, iter = 60$	
Instance	Error(%)
GKD-c.1 n500_m50.txt	0,0286
GKD-c.2 n500_m50.txt	0,0387
SOM-b.1 n100_m10.txt	1,74
SOM-b.2 n100_m20.txt	1,07
MDG-a.21 n2000_m200.txt	2,22
MDG-a.22 n2000_m200.txt	2,34
MDG-b.1 n500_m50.txt	2,04
MDG-c.1 n3000_m300.txt	1.86

Table 6: Cambio en $iter$

Al observar los resultados, se aprecia que un valor de α más elevado (0.95) conduce a errores significativamente mejores. Comparando con $\alpha = 0.85$ (visto anteriormente), queda claro que el algoritmo estaba enfriando demasiado rápido, reduciendo prematuramente la exploración. Un enfriamiento más lento permite explorar mejor el espacio de soluciones.

Por otro lado, la segunda tabla muestra que incrementar el número de iteraciones por nivel de temperatura mejora considerablemente el rendimiento, incluso manteniendo T y α constantes. Esto indica que el algoritmo beneficia de disponer de más tiempo para explorar el entorno local antes de reducir la temperatura.

Ajuste general

El único parámetro que no había completamente explorado era la temperatura inicial. Por ello, se decidió reducir T de manera drástica, permitiendo también disminuir la temperatura final.

Se probaron dos configuraciones:

Caso 1

- $T = 5.0$
- $T_{final} = 0.00001$
- $\alpha = 0.98$
- $iter = 200$
- $sin_mejora = 60$

Caso 2

- $T = 4.0$
- $T_{final} = 0.00001$
- $\alpha = 0.99$
- $iter = 200$
- $sin_mejora = 100$

Instance	Caso 1	Caso 2
GKD-c_1 n500_m50.txt	0,0128	0,0145
GKD-c_2 n500_m50.txt	0,00592	0,00495
SOM-b_1 n100_m10.txt	0,480	0
SOM-b_2 n100_m20.txt	0,167	0
MDG-a_21 n2000_m200.txt	1,12	1,10
MDG-a_22 n2000_m200.txt	1,28	1,18
MDG-b_1 n500_m50.txt	1,59	1,57
MDG-c_1 n3000_m300.txt	0,958	0,903

Table 7: Comparativa de Múltiples Errores de Calibración

Los resultados muestran claramente que el Caso 2 obtiene el mejor desempeño global, mejorando al Caso 1 en prácticamente todas las instancias.

Además, en dos de las instancias el algoritmo alcanzó el óptimo en todas las ejecuciones realizadas, lo que confirma que esta configuración logra un equilibrio adecuado entre diversificación inicial e intensificación final.

Resultado final de calibración

Tras analizar sistemáticamente todos los parámetros, la configuración seleccionada para el algoritmo es:

- $T = 4.0$
- $T_{final} = 0.00001$
- $\alpha = 0.99$
- $iter = 200$
- $sin_mejora = 100$

Esta combinación proporciona la mejor calidad de soluciones en la mayoría de las instancias y permite un equilibrio robusto entre exploración y explotación.

Nota: Se llevaron a cabo más pruebas donde se realizaron pequeñas variaciones en los parámetros. Sin embargo, no produjeron cambios relevantes en el rendimiento o calidad de las soluciones finales. Así, se ha optado por presentar únicamente las pruebas consideradas más relevantes y representativas para mantener claridad.

3.3.2 Resultados Temple Simulado

Para obtener los resultados relativos a esta metaheurística, han sido realizadas cinco ejecuciones independientes por cada instancia: se presenta el valor medio de las 5 ejecuciones, que representa el rendimiento promedio esperado del algoritmo, de forma a evaluar la consistencia y robustez del método. Se obtiene también el mejor valor obtenido de estas 5 ejecuciones, de forma a intentar representar el máximo potencial que el algoritmo es capaz de alcanzar.

Instance	Best Value	Temple Simulado	Error(%)	Mejor SA	Error(%)
GKD-c.19 n500_m50.txt	19477,3281	19476,25609	0,0055	19477,32577	0,0000121
GKD-c.20 n500_m50.txt	19604,84375	19604,23876	0,00309	19604,84356	0,000000969
SOM-B.4 n100_m40.txt	4142	4138,2	0,0917	4142	0
SOM-B.8 n200_m80.txt	16225	16215,6	0,0579	16225	0
SOM-B.12 n300_m120.txt	35881	35851,6	0,0819	35868	0,0362
SOM-B.16 n400_m160.txt	62487	62373,4	0,182	62442	0,072
SOM-B.20 n500_m200.txt	97344	97220,2	0,127	97255	0,0914
MDG-a.19 n500_m50.txt	7755,41064	7660,186	1,23	7710,84	0,575
MDG-a.20 n500_m50.txt	7733,86133	7621,328	1,46	7650,48	1,08
MDG-a.39 n2000_m200.txt	114201	112980,2	1,07	113081	0,981
MDG-a.40 n2000_m200.txt	114191	113221,6	0,849	113289	0,79
MDG-b.19 n500_m50.txt	778802,9375	767732,54	1,42	773582,84	0,67
MDG-b.20 n500_m50.txt	778644,8125	764914,25	1,76	778638,94	0,000754
MDG-b.39 n2000_m200.txt	11295054,2	11173161,54	1,08	11196788,01	0,87
MDG-b.40 n2000_m200.txt	11307104,8	11180662,11	1,12	11201871,06	0,931
MDG-c.5 n3000_m300.txt	24899703	24659547,8	0,964	24724380	0,704
MDG-c.10 n3000_m400.txt	43476251	43132539	0,791	43180198	0,681
MDG-c.15 n3000_m500.txt	66992877	66696887,6	0,442	66817764	0,261
MDG-c.20 n3000_m600.txt	95643586	95154304,2	0,512	95199270	0,465

Table 8: Comparación del resultado medio y mejor resultado del Temple Simulado (SA) vs. Mejor Valor (Best Value)

El Temple Simulado representa un rendimiento global bastante elevado, produciendo soluciones de alta calidad en la mayoría de las instancias, lo cual es evidenciado tanto en el resultado medio de las ejecuciones (columna Temple Simulado) como el mejor resultado (columna Mejor SA).

Relativamente al mejor resultado registrado, la calidad es excepcional. Solo una instancia presenta un error relativo superior al 1% (MDG-a.20), cuyo error es de 1,08%. En todas las demás, se han logrado soluciones con un error muy pequeño.

También el valor medio de las ejecuciones mantiene una calidad razonable, ya que solo en seis instancias supera el 1% de error.

El análisis confirma que ciertos conjuntos de datos siguen siendo más problemáticos (MDG-a y MDG-b), mientras que otros siguen presentado los menores errores relativos (GKD-c).

3.4 Algoritmos Genéticos

Los Algoritmos Genéticos (AG) son metaheurísticas poblacionales inspiradas en los procesos de evolución natural y selección genética. Los AG mantienen y evolucionan una población de soluciones candidatas a lo largo de múltiples generaciones, permitiendo una exploración más amplia del espacio de soluciones mediante operadores que simulan la reproducción y mutación biológicas.

Los AG son particularmente efectivos para problemas de optimización combinatoria complejos donde el espacio de búsqueda tiende a ser bastante grande. Su fortaleza radica en la capacidad de combinar información de múltiples soluciones prometedoras y explorar diferentes regiones del espacio de forma paralela.

Para el problema de la máxima diversidad, el Algoritmo Genético implementado sigue el esquema clásico de selección, cruce y mutación, adaptado específicamente a la representación y evaluación de soluciones para este problema. La implementación realizada presenta una complejidad computacional de:

$$O(G \cdot T \cdot m^2)$$

donde:

- G es el número de generaciones.
- T es el tamaño de la población.
- m es el tamaño del subconjunto a seleccionar.

Esta complejidad surge principalmente del cálculo repetido de la función de aptitud, que requiere $O(m^2)$ operaciones para evaluar la diversidad total de cada solución.

Los pasos implementados para AG son:

Paso 0: Inicialización. Se genera una población inicial de tamaño T compuesta por soluciones factibles. Cada solución, o individuo, se representa como un vector de enteros de longitud m , donde cada entrada corresponde al índice de un elemento seleccionado. Para generar esta población inicial, se implementaron dos estrategias distintas: por una lado, la generación inicial de una población totalmente aleatoria; y por otro lado, una versión donde el 50% de la población se genera con una estrategia voraz y el 50% restante son soluciones completamente aleatorias para balancear calidad inicial y diversidad genética. Se calcula la aptitud para cada individuo mediante la subrutina `Calcular_Fitness`, que evalúa la diversidad total del subconjunto seleccionado.

Paso 1 Selección de padres. Se emplea el método de *torneo binario* para seleccionar $T/2$ padres. Para cada torneo, se eligen aleatoriamente dos individuos de la población y se selecciona el que tenga mayor aptitud, favoreciendo así a los individuos mejores pero manteniendo cierta estocasticidad.

Paso 2 **Operadores genéticos:** Se aplican secuencialmente los operadores de cruce y mutación:

- **Cruce PMX:** Con probabilidad PC , se intercambia un segmento de genes entre dos padres para generar dos hijos. En la implementación de este operador se garantiza que las soluciones resultantes mantengan exactamente m elementos distintos.
- **Mutación por reemplazo:** Con probabilidad PM , se selecciona una posición aleatoria en el individuo y se reemplaza el elemento por otro no presente en la solución, actualizando el valor de aptitud.

Paso 3 **Selección de supervivientes:** Se aplica un torneo binario entre padres e hijos para determinar qué individuos pasan a la siguiente generación.

Paso 4 **Criterio de parada:** El algoritmo se ejecuta durante un número fijo de generaciones G .

Algorithm 4 Algoritmo Genético para el MDP

Require: G : número de generaciones, T : tamaño de población,

Require: PC : probabilidad de cruce, PM : probabilidad de mutación

Ensure: Mejor_I: mejor solución encontrada (vector de m elementos),

Ensure: Mejor_A: valor de fitness de la mejor solución

1: **1. Inicialización**

2: Poblacion_Inicial (T , Poblacion_I, Adaptacion_P,

3: Mejor_I, Mejor_A)

4: Reservar: Poblacion_Padres [$T/2 \times m$]

5: Hijos [$T \times m$], Adaptacion_H [T]

6: **2. Bucle principal (G iteraciones)**

7: **for** $gen \leftarrow 1$ **to** G **do**

8: **2.1. Selección padres:** Torneo binario

9: Seleccion_Torneo_Padres (T , Poblacion_I,

10: Adaptacion_P, Poblacion_Padres)

11: **2.2. Operadores:** Cruce PMX + Mutación

12: Cruce_PMX_ConMutacion (Poblacion_Padres,

13: Hijos, Adaptacion_H, PC , PM ,

14: Mejor_I, Mejor_A)

15: **2.3. Reemplazo:** Torneo padres vs hijos

16: Seleccion_Torneo_Final (Poblacion_I ,

17: Adaptacion_P, Hijos , Adaptacion_H)

18: **end for**

19: **3. Finalización**

20: Liberar memoria de todas las estructuras

21:

22: **return** (Mejor_I, Mejor_A) =0

3.4.1 Calibración AG

Dado que el rendimiento de los algoritmos genéticos es sensible a sus hiperparámetros, se diseñó la siguiente estrategia de calibración:

Calibración Probabilidad de Mutación. Se evaluó el impacto de la probabilidad de mutación PM en la calidad de la solución. Para la realización de pruebas centradas en la calibración de PM se mantuvieron constantes los siguientes parámetros:

- Tamaño de la población: $T = 100$
- Número de generaciones: $G = 1000$
- Probabilidad de cruce: $PC = 1.0$ (cruce total)
- Repeticiones por valor de PM : 5 ejecuciones independientes.

Los cinco valores de PM a probar son: 0.001 (muy baja), 0.01 (baja), 0.05 (moderada), 0.1 (moderada-alta) y 0.2 (alta). Los resultados completos se presentan en las siguientes Tablas 9 y 10.

Best Value	Instancia	Opción	Media Diversidad	Error (%)
19485.1875	GKD-c.1.n500_m50.txt	0.001	17875.0836	8.263%
19485.1875	GKD-c.1.n500_m50.txt	0.01	18846.71965	3.277%
19485.1875	GKD-c.1.n500_m50.txt	0.05	19377.43068	0.553%
19485.1875	GKD-c.1.n500_m50.txt	0.1	19428.14054	0.230%
19485.1875	GKD-c.1.n500_m50.txt	0.2	19457.00584	0.144%
19701.5371	GKD-c.2.n500_m50.txt	0.001	18141.12999	7.920%
19701.5371	GKD-c.2.n500_m50.txt	0.01	19067.79488	3.217%
19701.5371	GKD-c.2.n500_m50.txt	0.05	19558.65932	0.242%
19701.5371	GKD-c.2.n500_m50.txt	0.1	19625.77184	0.385%
19701.5371	GKD-c.2.n500_m50.txt	0.2	19665.16382	0.083%
333	SOM-b.1.n100_m10.txt	0.001	286.2	14.054%
333	SOM-b.1.n100_m10.txt	0.01	315.4	5.295%
333	SOM-b.1.n100_m10.txt	0.05	312.2	5.845%
333	SOM-b.1.n100_m10.txt	0.1	317.9	4.565%
333	SOM-b.1.n100_m10.txt	0.2	319	4.204%
1195	SOM-b.2.n100_m20.txt	0.001	1091.8	8.636%
1195	SOM-b.2.n100_m20.txt	0.01	1128.6	5.556%
1195	SOM-b.2.n100_m20.txt	0.05	1162.2	2.328%
1195	SOM-b.2.n100_m20.txt	0.1	1160.2	2.912%
1195	SOM-b.2.n100_m20.txt	0.2	1172	1.925%
7833.83252	MDG-a.1.n500_m50.txt	0.001	6613.764	13.021%
7833.83252	MDG-a.1.n500_m50.txt	0.01	7182.62	8.310%
7833.83252	MDG-a.1.n500_m50.txt	0.05	7463.62	4.726%
7833.83252	MDG-a.1.n500_m50.txt	0.1	7514.544	4.076%
7833.83252	MDG-a.1.n500_m50.txt	0.2	7564.692	3.436%
7771.66162	MDG-a.2.n500_m50.txt	0.001	6841.854	11.964%
7771.66162	MDG-a.2.n500_m50.txt	0.01	7154.31	7.944%
7771.66162	MDG-a.2.n500_m50.txt	0.05	7410.198	4.651%
7771.66162	MDG-a.2.n500_m50.txt	0.1	7523.426	4.058%
7771.66162	MDG-a.2.n500_m50.txt	0.2	7523.422	3.078%
778030.625	MDG-b.1.n500_m50.txt	0.001	686039.008	11.824%
778030.625	MDG-b.1.n500_m50.txt	0.01	715688.232	8.013%
778030.625	MDG-b.1.n500_m50.txt	0.05	740677.266	4.801%
778030.625	MDG-b.1.n500_m50.txt	0.1	753595.684	3.141%
778030.625	MDG-b.1.n500_m50.txt	0.2	756090.956	2.820%
779963.688	MDG-b.2.n500_m50.txt	0.001	677046.528	13.195%
779963.688	MDG-b.2.n500_m50.txt	0.01	716247.658	8.169%
779963.688	MDG-b.2.n500_m50.txt	0.05	744851.662	4.502%
779963.688	MDG-b.2.n500_m50.txt	0.1	748304.414	3.931%
779963.688	MDG-b.2.n500_m50.txt	0.2	756787.602	2.971%

Table 9: Resultados de la calibracion de Probabilidad de Mutación (Parte 1)

Best Value	Instancia	Opción	Media Diversidad	Error (%)
24884110	MDG-c.1_n3000_m300.txt	0.001	22873807	8.079%
24884110	MDG-c.1_n3000_m300.txt	0.01	23265465	6.505%
24884110	MDG-c.1_n3000_m300.txt	0.05	23639293	5.002%
24884110	MDG-c.1_n3000_m300.txt	0.1	23779526.6	4.439%
24884110	MDG-c.1_n3000_m300.txt	0.2	23970622.8	3.671%
24905330	MDG-c.2_n3000_m300.txt	0.001	22836412.6	8.307%
24905330	MDG-c.2_n3000_m300.txt	0.01	23225133.8	6.746%
24905330	MDG-c.2_n3000_m300.txt	0.05	23660509	4.998%
24905330	MDG-c.2_n3000_m300.txt	0.1	23854637.8	4.219%
24905330	MDG-c.2_n3000_m300.txt	0.2	23949233.4	3.839%

Table 10: Resultados de la calibración de Probabilidad de Mutación (Parte 2)

Análisis de Resultados:

- **Tendencia clara y consistente:** Se observa una relación positiva monótona entre el valor de PM y la calidad de la solución en todas las instancias evaluadas. Sin excepción, el mayor valor probado ($PM = 0.2$) obtuvo el mejor rendimiento, presentando el menor porcentaje de error en cada caso. La secuencia de mejora es sistemática: $PM : 0.2 > 0.1 > 0.05 > 0.01 > 0.001$.
- **Errores elevados:** Llama la atención los elevados porcentajes de error de hasta el 14.05% en SOM-b_1 con $PM = 0.001$. Esto puede deberse a que aún no se han optimizado otros parámetros críticos como la probabilidad de cruce, el tamaño de población o el número de generaciones. La configuración actual ($PC = 1.0$, $T = 100$, $G = 1000$) es intencionalmente genérica para aislar el efecto de PM .
- **Efecto de la mutación en diversidad genética:** Con $PC = 1.0$ (cruce total), los resultados muestran que valores muy bajos de mutación ($PM \leq 0.01$) resultan en convergencia prematura a óptimos locales de baja calidad, con errores sistemáticamente superiores al 5%. En contraste, valores más altos ($PM \geq 0.1$) permiten mantener suficiente variabilidad para explorar mejor el espacio de búsqueda.
- **Escalabilidad del efecto:** El patrón se mantiene consistente a través de instancias de diferente tamaño y complejidad:
 - **Pequeñas:** SOM-b ($n=100$) - error máximo 14.05%, mínimo 1.92%
 - **Medianas:** GKD-c, MDG-a/b ($n=500$) - error máximo 13.20%, mínimo 0.08%
 - **Grandes:** MDG-c ($n=3000$) - error máximo 8.31%, mínimo 3.67%
- **Punto óptimo identificado:** $PM = 0.2$ demostró ser claramente superior, reduciendo el considerablemente el error en muchas instancias (ej: GKD-c.2: 7.92% vs 0.08%). Esto sugiere que, con $PC = 1.0$, se requiere una tasa de mutación relativamente alta para contrarrestar la homogenización causada por el cruce total.

Conclusión y siguiente paso: en base a estos resultados, se selecciona $PM = 0.2$ como el valor óptimo para la probabilidad de mutación en esta etapa de la calibración secuencial. Este valor será mantenido fijo durante la siguiente fase:

calibración de la probabilidad de cruce (PC). Es importante destacar que estos resultados son preliminares y podrían refinarse tras considerar las interacciones con otros parámetros en etapas posteriores de calibración o la introducción de modificaciones en el algoritmo que permitan mejorar significativas.

Calibración Probabilidad de Cruce. Tras fijar la probabilidad de mutación en $PM = 0.2$, se procedió a calibrar la probabilidad de cruce PC . Esta fase evalúa el impacto de la recombinación genética en la calidad de las soluciones, manteniendo los siguientes parámetros constantes:

- Probabilidad de mutación: $PM = 0.2$
- Tamaño de la población: $T = 100$
- Número de generaciones: $G = 1000$
- Repeticiones por valor de PC : 5 ejecuciones independientes.

Se evaluaron cuatro valores de PC : 0.6 (moderada-baja), 0.75 (moderada), 0.9 (alta) y 1.0 (cruce total). Los resultados completos se presentan en las siguientes Tablas 11 y 12.

Best Value	Instancia	Opción	Media Diversidad	Error (%)
19485.1875	GKD-c_1_n500_m50.txt	0.6	19471.42	0.076%
19485.1875	GKD-c_1_n500_m50.txt	0.75	19462.75	0.115%
19485.1875	GKD-c_1_n500_m50.txt	0.9	19464.8	0.104%
19485.1875	GKD-c_1_n500_m50.txt	1	19470.28	0.076%
19701.5371	GKD-c_2_n500_m50.txt	0.6	19682.73	0.095%
19701.5371	GKD-c_2_n500_m50.txt	0.75	19690.41	0.056%
19701.5371	GKD-c_2_n500_m50.txt	0.9	19685.78	0.079%
19701.5371	GKD-c_2_n500_m50.txt	1	19685.86	0.079%
333	SOM-b_1_n100_m10.txt	0.6	315.6	5.225%
333	SOM-b_1_n100_m10.txt	0.75	320.2	3.844%
333	SOM-b_1_n100_m10.txt	0.9	316.4	4.984%
333	SOM-b_1_n100_m10.txt	1	317	4.805%
1195	SOM-b_2_n100_m20.txt	0.6	1167.2	2.326%
1195	SOM-b_2_n100_m20.txt	0.75	1164.8	2.527%
1195	SOM-b_2_n100_m20.txt	0.9	1181.8	1.105%
1195	SOM-b_2_n100_m20.txt	1	1178.8	1.356%
7833.83252	MDG-a_1_n500_m50.txt	0.6	7567.34	3.402%
7833.83252	MDG-a_1_n500_m50.txt	0.75	7544.822	3.693%
7833.83252	MDG-a_1_n500_m50.txt	0.9	7548.144	3.647%
7833.83252	MDG-a_1_n500_m50.txt	1	7537.82	3.782%
7771.66162	MDG-a_2_n500_m50.txt	0.6	7522.472	3.206%
7771.66162	MDG-a_2_n500_m50.txt	0.75	7516.526	3.283%
7771.66162	MDG-a_2_n500_m50.txt	0.9	7538.622	2.998%
7771.66162	MDG-a_2_n500_m50.txt	1	7532.426	3.078%
778030.625	MDG-b_1_n500_m50.txt	0.6	755469.8	2.899%
778030.625	MDG-b_1_n500_m50.txt	0.75	754837.4	2.981%
778030.625	MDG-b_1_n500_m50.txt	0.9	751675.9	3.387%
778030.625	MDG-b_1_n500_m50.txt	1	750489.4	3.539%
779963.6875	MDG-b_2_n500_m50.txt	0.6	755153.6	3.181%
779963.6875	MDG-b_2_n500_m50.txt	0.75	753660.9	3.373%
779963.6875	MDG-b_2_n500_m50.txt	0.9	751959.6	3.590%
779963.6875	MDG-b_2_n500_m50.txt	1	752127.6	3.569%

Table 11: Resultados de la calibración de Probabilidad de Cruce (Parte 1)

Best Value	Instancia	Opción	Media Diversidad	Error (%)
24884110	MDG-c.1_n3000_m300.txt	0.6	23984187.4	3.616%
24884110	MDG-c.1_n3000_m300.txt	0.75	2397117.4	3.668%
24884110	MDG-c.1_n3000_m300.txt	0.9	23958792.6	3.719%
24884110	MDG-c.1_n3000_m300.txt	1	23979522	3.635%
24905330	MDG-c.2_n3000_m300.txt	0.6	24005118.6	3.615%
24905330	MDG-c.2_n3000_m300.txt	0.75	23964524.2	3.778%
24905330	MDG-c.2_n3000_m300.txt	0.9	23942589.8	3.866%
24905330	MDG-c.2_n3000_m300.txt	1	23961006.8	3.792%

Table 12: Resultados de la calibración de Probabilidad de Cruce (Parte 2)

Análisis de Resultados:

- **No hay un valor dominante claro:** Los resultados para PC muestran una variabilidad significativa. En algunas instancias (GKD-c.1, MDG-b.1, MDG-b.2) el mejor resultado se obtiene con $PC = 0.6$, mientras que en otras (SOM-b.1, MDG-a.2, MDG-c.1, MDG-c.2) es $PC = 0.75$ o $PC = 0.9$. En GKD-c.2, $PC = 0.75$ obtiene el mejor resultado, pero $PC = 0.6$ está muy cercano.
- **Error reducido significativamente:** En comparación con la primera etapa de calibración (cuando PC estaba fijo en 1.0), en esta segunda se observa un descenso de los errores. Por ejemplo, en instancias GKD-c los errores han bajado de 8.263-0.083% a 0.115-0.056%, y en MDG-c de 8.307-3.671% a 3.866-3.615%.
- **$PC=1.0$ generalmente no óptimo:** En la mayoría de las instancias, el cruce total ($PC = 1.0$) no produce los mejores resultados, especialmente en instancias MDG donde muestra un rendimiento consistentemente inferior a valores más moderados de PC .
- **Interacción PM-PC balanceada:** Con $PM = 0.2$ relativamente alta, parece que valores moderados de PC (0.6-0.75) crean un mejor balance entre exploración (mutación) y explotación (cruce) que un cruce total.

Conclusión final: Siguiendo una estrategia de calibración secuencial se decide $PC = 0.6$ (con $PM = 0.2$) por obtener mejor resultado en 4/8 instancias, parece ser más robusto en instancias más difíciles y proporcionar un balance óptimo con mutación.

Calibración combinada de PM y PC: Dado que $PM = 0.2$ se decidió realizar una calibración conjunta de los parámetros PM y PC para capturar sus interacciones. Para ello se evaluaron todas las combinaciones de los siguientes valores:

- Probabilidades de Mutación: $PM = [0.001, 0.01, 0.05, 0.1, 0.2]$
- Probabilidades de Cruce: $PC = [0.6, 0.75, 0.9, 1.0]$
- Total de combinaciones: 20 por instancia
- Parámetros fijos: $T = 100$, $G = 1000$, 5 repeticiones por combinación

Los resultados completos de la calibración conjunta de los 20 parámetros combinados (5 valores de PM \times 4 valores de PC) se presentan en las siguientes Tablas 13, 14, 15 y 16:

Best Value	Instancia	Combinación (PM, PC)	Media Diversidad	Ranking	Error (%)
19485.1875	GKD-c.1n500_m50.txt	(0.001, 0.600)	17713.88393	20	9.091%
		(0.001, 0.750)	17769.50969	19	8.805%
		(0.001, 0.900)	17941.58002	17	7.922%
		(0.001, 1.000)	17802.40446	18	8.636%
		(0.01, 0.600)	18739.85146	16	3.825%
		(0.01, 0.750)	18823.49336	14	3.396%
		(0.01, 0.900)	18823.11198	15	3.398%
		(0.01, 1.000)	18897.94970	13	3.014%
		(0.05, 0.600)	19351.55120	12	0.686%
		(0.05, 0.750)	19374.77661	11	0.567%
		(0.05, 0.900)	19381.21157	10	0.534%
		(0.05, 1.000)	19388.74985	9	0.495%
		(0.1, 0.600)	19438.95705	7	0.237%
		(0.1, 0.750)	19438.27399	8	0.241%
		(0.1, 0.900)	19445.07175	5	0.206%
		(0.1, 1.000)	19443.48840	6	0.214%
		(0.2, 0.600)	19464.68021	4	0.105%
		(0.2, 0.750)	19468.50584	1	0.086%
		(0.2, 0.900)	19467.80033	2	0.089%
		(0.2, 1.000)	19465.86559	3	0.099%
19701.53711	GKD-c.2n500_m50.txt	(0.001, 0.600)	17841.98723	19	9.439%
		(0.001, 0.750)	17786.95693	20	9.718%
		(0.001, 0.900)	18121.80921	17	8.018%
		(0.001, 1.000)	18095.23264	18	8.153%
		(0.01, 0.600)	19071.35802	14	3.199%
		(0.01, 0.750)	18929.98056	16	3.916%
		(0.01, 0.900)	18997.29506	15	3.575%
		(0.01, 1.000)	19083.85825	13	3.135%
		(0.05, 0.600)	19564.69387	11	0.695%
		(0.05, 0.750)	19576.83078	9	0.633%
		(0.05, 0.900)	19573.85654	10	0.648%
		(0.05, 1.000)	19553.48054	12	0.751%
		(0.1, 0.600)	19653.13736	7	0.246%
		(0.1, 0.750)	19658.38710	6	0.219%
		(0.1, 0.900)	19635.44454	8	0.335%
		(0.1, 1.000)	19659.09498	5	0.215%
		(0.2, 0.600)	19684.50545	2	0.086%
		(0.2, 0.750)	19681.27334	4	0.103%
		(0.2, 0.900)	19683.19806	3	0.093%
		(0.2, 1.000)	19685.45929	1	0.082%
333	SOM-b.1n100_m10.txt	(0.001, 0.750)	292.6	20	12.132%
		(0.001, 0.900)	294.2	19	11.652%
		(0.001, 1.000)	299.2	17	10.150%
		(0.01, 0.600)	307.4	13	7.688%
		(0.01, 0.750)	306.2	15	8.048%
		(0.01, 0.900)	306.8	14	7.868%
		(0.01, 1.000)	300.6	16	9.730%
		(0.05, 0.600)	308.8	12	7.267%
		(0.05, 0.750)	315.4	7	5.285%
		(0.05, 0.900)	322.4	1	3.183%
		(0.05, 1.000)	315.8	5	5.165%
		(0.1, 0.600)	312.4	11	6.186%
		(0.1, 0.750)	318.6	3	4.324%
		(0.1, 0.900)	314.2	9	5.646%
		(0.1, 1.000)	321.4	2	3.483%
		(0.2, 0.600)	315.2	8	5.345%
		(0.2, 0.750)	312.6	10	6.126%
		(0.2, 0.900)	317.8	4	4.565%
		(0.2, 1.000)	315.8	5	5.165%

Table 13: Resultados calibración combinada PM-PC (Parte-1)

Best Value	Instancia	Combinación (PM, PC)	Media Diversidad	Ranking	Error (%)
1195	SOM-b_2_n100_m20.txt	(0.001, 0.600)	1075.2	19	10.025%
		(0.001, 0.750)	1066.4	20	10.762%
		(0.001, 0.900)	1078.2	18	9.774%
		(0.001, 1.000)	1081.2	17	9.523%
		(0.01, 0.600)	1122.6	16	6.059%
		(0.01, 0.750)	1142.4	14	4.402%
		(0.01, 0.900)	1144.8	13	4.201%
		(0.01, 1.000)	1132.2	15	5.255%
		(0.05, 0.600)	1157.0	12	3.180%
		(0.05, 0.750)	1164.4	10	2.561%
		(0.05, 0.900)	1161.0	11	2.845%
		(0.05, 1.000)	1169.2	4	2.159%
		(0.1, 0.600)	1166.4	7	2.393%
		(0.1, 0.750)	1174.6	1	1.707%
		(0.1, 0.900)	1169.2	4	2.159%
		(0.1, 1.000)	1165.0	9	2.510%
		(0.2, 0.600)	1166.4	7	2.393%
		(0.2, 0.750)	1172.2	3	1.908%
		(0.2, 0.900)	1166.8	6	2.360%
		(0.2, 1.000)	1172.4	2	1.891%
7833.83252	MDG-a_1_n500_m50.txt	(0.001, 0.600)	6746.592	19	13.879%
		(0.001, 0.750)	6840.242	18	12.683%
		(0.001, 0.900)	6851.046	17	12.545%
		(0.001, 1.000)	6734.052	20	14.039%
		(0.01, 0.600)	7160.002	15	8.602%
		(0.01, 0.750)	7220.770	13	7.826%
		(0.01, 0.900)	7133.998	16	8.933%
		(0.01, 1.000)	7183.018	14	8.308%
		(0.05, 0.600)	7413.532	10	5.365%
		(0.05, 0.750)	7403.250	12	5.496%
		(0.05, 0.900)	7442.148	9	5.000%
		(0.05, 1.000)	7407.470	11	5.443%
		(0.1, 0.600)	7510.672	6	4.125%
		(0.1, 0.750)	7547.394	3	3.656%
		(0.1, 0.900)	7494.358	8	4.333%
		(0.1, 1.000)	7499.236	7	4.271%
		(0.2, 0.600)	7524.598	5	3.947%
		(0.2, 0.750)	7527.328	4	3.913%
		(0.2, 0.900)	7562.570	2	3.463%
		(0.2, 1.000)	7584.130	1	3.187%
7771.66162	MDG-a_2_n500_m50.txt	(0.001, 0.600)	6802.534	18	12.470%
		(0.001, 0.750)	6737.600	20	13.306%
		(0.001, 0.900)	6774.972	19	12.825%
		(0.001, 1.000)	6832.796	17	12.081%
		(0.01, 0.600)	7160.882	14	7.859%
		(0.01, 0.750)	7181.720	13	7.591%
		(0.01, 0.900)	7148.726	15	8.015%
		(0.01, 1.000)	7144.476	16	8.070%
		(0.05, 0.600)	7401.774	11	4.759%
		(0.05, 0.750)	7403.446	10	4.738%
		(0.05, 0.900)	7457.264	9	4.045%
		(0.05, 1.000)	7400.660	12	4.774%
		(0.1, 0.600)	7490.994	6	3.611%
		(0.1, 0.750)	7508.272	5	3.389%
		(0.1, 0.900)	7463.508	8	3.965%
		(0.1, 1.000)	7482.520	7	3.720%
		(0.2, 0.600)	7532.240	2	3.081%
		(0.2, 0.750)	7539.552	1	2.987%
		(0.2, 0.900)	7525.832	3	3.163%
		(0.2, 1.000)	7511.652	4	3.346%

Table 14: Resultados completos calibración PM-PC - (Parte-2)

Best Value	Instancia	Combinación (PM, PC)	Media Diversidad	Ranking	Error (%)
778030.625	MDG-b_1_n500_m50.txt	(0.001, 0.600)	673908.020	19	13.383%
		(0.001, 0.750)	671011.988	20	13.755%
		(0.001, 0.900)	676079.068	18	13.104%
		(0.001, 1.000)	680662.954	17	12.515%
		(0.01, 0.600)	713522.406	16	8.291%
		(0.01, 0.750)	715618.866	15	8.022%
		(0.01, 0.900)	716764.590	13	7.875%
		(0.01, 1.000)	716154.902	14	7.953%
		(0.05, 0.600)	737590.578	12	5.198%
		(0.05, 0.750)	746022.696	8	4.114%
		(0.05, 0.900)	743123.004	11	4.487%
		(0.05, 1.000)	745171.046	10	4.223%
		(0.1, 0.600)	749635.684	7	3.650%
		(0.1, 0.750)	749953.618	6	3.609%
		(0.1, 0.900)	745196.960	9	4.220%
		(0.1, 1.000)	753834.178	5	3.110%
		(0.2, 0.600)	754628.626	4	3.008%
		(0.2, 0.750)	756866.894	3	2.720%
		(0.2, 0.900)	757497.898	1	2.639%
		(0.2, 1.000)	756923.764	2	2.713%
779963.6875	MDG-b_2_n500_m50.txt	(0.001, 0.600)	670480.932	20	14.037%
		(0.001, 0.750)	679463.384	17	12.885%
		(0.001, 0.900)	674036.828	19	13.581%
		(0.001, 1.000)	678962.364	18	12.949%
		(0.01, 0.600)	718587.644	13	7.869%
		(0.01, 0.750)	714961.396	16	8.334%
		(0.01, 0.900)	716698.834	15	8.111%
		(0.01, 1.000)	718411.764	14	7.892%
		(0.05, 0.600)	742006.240	10	4.867%
		(0.05, 0.750)	741400.358	12	4.944%
		(0.05, 0.900)	741440.330	11	4.939%
		(0.05, 1.000)	746320.936	9	4.313%
		(0.1, 0.600)	748429.230	8	4.043%
		(0.1, 0.750)	750935.810	6	3.722%
		(0.1, 0.900)	750557.504	7	3.770%
		(0.1, 1.000)	751459.808	5	3.655%
		(0.2, 0.600)	756869.952	2	2.961%
		(0.2, 0.750)	754627.358	4	3.248%
		(0.2, 0.900)	757092.544	1	2.932%
		(0.2, 1.000)	756752.060	3	2.976%
24884110	MDG-c_1_n3000_m300.txt	(0.001, 0.600)	22803610.0	19	8.361%
		(0.001, 0.750)	22800075.2	20	8.375%
		(0.001, 0.900)	22865221.4	17	8.113%
		(0.001, 1.000)	22847852.0	18	8.183%
		(0.01, 0.600)	23238931.6	14	6.611%
		(0.01, 0.750)	23231949.6	15	6.639%
		(0.01, 0.900)	23243383.0	13	6.593%
		(0.01, 1.000)	23224957.0	16	6.668%
		(0.05, 0.600)	23649715.0	12	4.961%
		(0.05, 0.750)	23683208.6	10	4.826%
		(0.05, 0.900)	23706122.4	9	4.734%
		(0.05, 1.000)	23673217.6	11	4.866%
		(0.1, 0.600)	23857145.8	5	4.127%
		(0.1, 0.750)	23844479.2	6	4.178%
		(0.1, 0.900)	23807574.8	8	4.326%
		(0.1, 1.000)	23829814.0	7	4.237%
		(0.2, 0.600)	23980190.4	1	3.633%
		(0.2, 0.750)	23960759.2	3	3.711%
		(0.2, 0.900)	23973742.0	2	3.658%
		(0.2, 1.000)	23946476.0	4	3.768%

Table 15: Resultados completos calibración PM-PC - (Parte 3)

Best Value	Instancia	Combinación (PM, PC)	Media Diversidad	Ranking	Error (%)
24905330	MDG-c.2.n3000.m300.txt	(0.001, 0.600)	22820789.2	18	8.370%
		(0.001, 0.750)	22799502.8	20	8.455%
		(0.001, 0.900)	22871384.2	17	8.167%
		(0.001, 1.000)	22813180.8	19	8.400%
		(0.01, 0.600)	23227475.0	16	6.737%
		(0.01, 0.750)	23240950.2	14	6.683%
		(0.01, 0.900)	23237411.8	15	6.697%
		(0.01, 1.000)	23255666.0	13	6.624%
		(0.05, 0.600)	23676478.2	10	4.934%
		(0.05, 0.750)	23633875.4	12	5.105%
		(0.05, 0.900)	23695246.4	9	4.859%
		(0.05, 1.000)	23651879.0	11	5.033%
		(0.1, 0.600)	23822617.6	8	4.347%
		(0.1, 0.750)	23853583.2	5	4.223%
		(0.1, 0.900)	23836963.6	7	4.290%
		(0.1, 1.000)	23850810.0	6	4.234%
		(0.2, 0.600)	23960924.8	3	3.792%
		(0.2, 0.750)	23971672.4	2	3.749%
		(0.2, 0.900)	23958881.6	4	3.800%
		(0.2, 1.000)	23977929.2	1	3.724%

Table 16: Resultados completos calibración PM-PC - (Parte 4)

Análisis de Resultados:

Si bien los resultados muestran que las combinaciones con $PM = 0.2$ obtienen los menores porcentajes de error relativo en la mayoría de las instancias, un análisis más profundo revela importantes consideraciones prácticas y teóricas que justifican la selección de valores más conservadores.

1. Consideraciones prácticas sobre PM=0.2:

- **Compromiso de explotación:** Con tanta mutación, el algoritmo puede perder su capacidad para explotar y refinar soluciones prometedoras, comportándose más como una búsqueda aleatoria que como un algoritmo evolutivo dirigido.
- **Desviación de prácticas comunes:** La investigación en algoritmos genéticos establece que valores de probabilidad de mutación superiores al 10% son inusuales. Estos valores extremos generalmente se justifican solo en problemas con propiedades específicas que no caracterizan al MDP, haciendo cuestionable su adopción en nuestro contexto.

2. Selección PM=0.1, PC=0.75: La combinación ($PM = 0.1, PC = 0.75$) demuestra un rendimiento sólido y consistente en todos los tipos de instancias:

- **GKD-c:** Ranking 4-5 con errores entre 0.2408% y 0.2154%
- **SOM-b:** Ranking 3-5 con errores entre 4.3243% y 1.7071%
- **MDG-a:** Ranking 3-5 con errores entre 3.3891% y 3.6564%
- **MDG-b:** Ranking 5 con errores entre 3.1099% y 3.6545%
- **MDG-c:** Ranking 5 con errores entre 4.1270% y 4.2230%

PM=0.1: Proporciona suficiente diversidad genética para evitar convergencia prematura, manteniendo la integridad de buenas soluciones. Además, PM=0.1 reduce a la mitad las operaciones de mutación respecto a PM=0.2, disminuyendo el tiempo de ejecución sin sacrificar significativamente la calidad de la solución. En problemas de gran escala como MDG-c ($n=3000$), esta reducción computacional podría ser particularmente valiosa.

PC=0.75: Permite suficiente recombinación para combinar soluciones prometedoras (75% de la población) mientras mantiene cierto grado de elitismo natural al no cruzar el 25% restante, crucial para preservar soluciones ya optimizadas.

Conclusión final: Se selecciona la configuración ($PM = 0.1, PC = 0.75$) como parámetros óptimos para el algoritmo genético, fundamentada en:

1. Ofrece mayor eficiencia computacional y menor variabilidad entre ejecuciones
2. Demuestra robustez y adaptabilidad a diferentes tipos de instancias
3. Se alinea con prácticas estándar en la literatura de algoritmos genéticos

Esta configuración será utilizada en las siguientes fases de calibración (tamaño de población T y número de generaciones G) y en la evaluación final del algoritmo.

Calibración del Tamaño de la Población. Para determinar el tamaño óptimo de población (T) del algoritmo genético, se evaluaron cuatro configuraciones manteniendo fijo el resto de parámetros:

- Tamaños de población: $T = [50, 100, 150, 200]$
- Parámetros fijos: $PM = 0.1, PC = 0.75, G = 1000$
- Repeticiones por valor de T: 5 ejecuciones independientes.

Los resultados completos de la calibración del tamaño de población se presentan en las Tablas 17 y 18:

Best Value	Instancia	Tamaño (T)	Media Diversidad	Error (%)
19485.1875	GKD-c.1.n500.m50.txt	50	19445.65991	0.203%
19485.1875	GKD-c.1.n500.m50.txt	100	19465.55062	0.101%
19485.1875	GKD-c.1.n500.m50.txt	150	19475.07663	0.052%
19485.1875	GKD-c.1.n500.m50.txt	200	19479.80213	0.028%
19701.53711	GKD-c.2.n500.m50.txt	50	19579.94914	0.617%
19701.53711	GKD-c.2.n500.m50.txt	100	19662.30315	0.199%
19701.53711	GKD-c.2.n500.m50.txt	150	19673.82555	0.141%
19701.53711	GKD-c.2.n500.m50.txt	200	19673.44007	0.143%
333	SOM-b.1.n100.m10.txt	50	312.2	6.246%
333	SOM-b.1.n100.m10.txt	100	316.8	4.865%
333	SOM-b.1.n100.m10.txt	150	320.4	3.784%
333	SOM-b.1.n100.m10.txt	200	319.6	4.024%
1195	SOM-b.2.n100.m20.txt	50	1165.0	2.510%
1195	SOM-b.2.n100.m20.txt	100	1161.2	2.828%
1195	SOM-b.2.n100.m20.txt	150	1170.2	2.075%
1195	SOM-b.2.n100.m20.txt	200	1173.6	1.791%

Table 17: Resultados Calibración del Tamaño de la población (Parte 1)

Best Value	Instancia	Tamaño (T)	Media Diversidad	Error (%)
7833.83252	MDG-a_1_n500_m50.txt	50	7421.712	5.261%
7833.83252	MDG-a_1_n500_m50.txt	100	7495.644	4.317%
7833.83252	MDG-a_1_n500_m50.txt	150	7571.118	3.354%
7833.83252	MDG-a_1_n500_m50.txt	200	7508.210	4.157%
7771.66162	MDG-a_2_n500_m50.txt	50	7370.174	5.166%
7771.66162	MDG-a_2_n500_m50.txt	100	7483.766	3.704%
7771.66162	MDG-a_2_n500_m50.txt	150	7540.852	2.970%
7771.66162	MDG-a_2_n500_m50.txt	200	7522.550	3.205%
778030.625	MDG-b_1_n500_m50.txt	50	743121.500	4.487%
778030.625	MDG-b_1_n500_m50.txt	100	750007.466	3.602%
778030.625	MDG-b_1_n500_m50.txt	150	751048.296	3.468%
778030.625	MDG-b_1_n500_m50.txt	200	754972.152	2.964%
779963.6875	MDG-b_2_n500_m50.txt	50	745040.880	4.477%
779963.6875	MDG-b_2_n500_m50.txt	100	752917.626	3.468%
779963.6875	MDG-b_2_n500_m50.txt	150	751469.532	3.653%
779963.6875	MDG-b_2_n500_m50.txt	200	755049.316	3.194%
24884110	MDG-c_1_n3000_m300.txt	50	23674259.4	4.862%
24884110	MDG-c_1_n3000_m300.txt	100	23839389.4	4.198%
24884110	MDG-c_1_n3000_m300.txt	150	23907239.4	3.926%
24884110	MDG-c_1_n3000_m300.txt	200	23965718.6	3.691%
24905330	MDG-c_2_n3000_m300.txt	50	23821976.6	4.350%
24905330	MDG-c_2_n3000_m300.txt	100	23965130.4	3.775%
24905330	MDG-c_2_n3000_m300.txt	150	24029101.4	3.518%
24905330	MDG-c_2_n3000_m300.txt	200	24073144.8	3.341%

Table 18: Resultados Calibración del Tamaño de la población (Parte-2)

Análisis de Resultados:

El análisis de los resultados muestra que los tamaños de población pequeños ($T = 50$ y $T = 100$) nunca produjeron el mejor resultado en ninguna de las 10 instancias evaluadas, concentrándose el rendimiento óptimo en $T = 150$ (4 instancias) y $T = 200$ (6 instancias). A continuación, se detalla la comparativa de error entre estas dos configuraciones:

- **Instancias GKD-c:** $T = 200$ fue superior en GKD-c_1 (0.028% vs 0.052%), pero $T = 150$ fue ligeramente mejor en GKD-c_2 (0.141% vs 0.143%).
- **Instancias SOM-b:** Hubo un comportamiento mixto. $T = 150$ resultó claramente superior en SOM-b_1 (3.784% vs 4.024%), mientras que $T = 200$ fue mejor en SOM-b_2 (1.791% vs 2.075%).
- **Instancias MDG-a:** $T = 150$ fue consistentemente superior, obteniendo el menor error en ambas instancias (MDG-a_1: 3.354% vs 4.157%; MDG-a_2: 2.970% vs 3.205%).
- **Instancias MDG-b:** $T = 200$ fue el tamaño dominante, logrando el menor error en ambas instancias (MDG-b_1: 2.964% vs 3.468%; MDG-b_2: 3.194% vs 3.653%).
- **Instancias MDG-c:** $T = 200$ fue superior en ambas instancias (MDG-c_1: 3.691% vs 3.926%; MDG-c_2: 3.341% vs 3.518%).

En resumen, aunque $T = 200$ obtuvo el menor error en 6 de 10 casos, $T = 150$ demostró una mejor robustez en el grupo MDG-a y GKD-c_2, e incluso fue superior en SOM-b_1.

Conclusión final: Aunque $T=200$ obtiene más ‘mejores resultados’ (6 vs 4 instancias), se selecciona como configuración óptima **$T=150$** por ofrecer un mejor equilibrio calidad-eficiencia, sabiendo que a mayor tamaño de la población, el coste computacional también sera mayor. Tomaremos este valor como fijo en la siguiente calibración.

Calibración del Número de Generaciones. Para determinar el valor óptimo del parámetro G (número de generaciones), se evaluaron cuatro configuraciones manteniendo fijos los parámetros previamente calibrados:

- Número de generaciones: $G = [500, 1000, 1500, 1700]$
- Parámetros fijos: $T = 150$, $PM = 0.1$, $PC = 0.75$
- Repeticiones por valor de G : 5 ejecuciones independientes.

Los resultados completos de la calibración del número de generaciones se presentan en la Tabla 19:

Best Value	Instancia	Generaciones (G)	Media Diversidad	Error (%)
19485.1875	GKD-c.1.n500.m50.txt	500	19406.59	0.403%
19485.1875	GKD-c.1.n500.m50.txt	1000	19456.83	0.146%
19485.1875	GKD-c.1.n500.m50.txt	1500	19472.44	0.065%
19485.1875	GKD-c.1.n500.m50.txt	1700	19474.79	0.053%
19701.53711	GKD-c.2.n500.m50.txt	500	19570.57	0.665%
19701.53711	GKD-c.2.n500.m50.txt	1000	19673.02	0.145%
19701.53711	GKD-c.2.n500.m50.txt	1500	19693.24	0.042%
19701.53711	GKD-c.2.n500.m50.txt	1700	19694.17	0.037%
333	SOM-b.1.n100.m10.txt	500	316.20	5.045%
333	SOM-b.1.n100.m10.txt	1000	318.60	4.324%
333	SOM-b.1.n100.m10.txt	1500	320.40	3.784%
333	SOM-b.1.n100.m10.txt	1700	319.40	4.084%
1195	SOM-b.2.n100.m20.txt	500	1167.00	2.343%
1195	SOM-b.2.n100.m20.txt	1000	1153.60	3.464%
1195	SOM-b.2.n100.m20.txt	1500	1173.80	1.774%
1195	SOM-b.2.n100.m20.txt	1700	1175.80	1.607%
7833.83252	MDG-a.1.n500.m50.txt	500	7431.75	5.133%
7833.83252	MDG-a.1.n500.m50.txt	1000	7553.27	3.581%
7833.83252	MDG-a.1.n500.m50.txt	1500	7592.03	3.087%
7833.83252	MDG-a.1.n500.m50.txt	1700	7622.06	2.703%
7771.66162	MDG-a.2.n500.m50.txt	500	7419.69	4.529%
7771.66162	MDG-a.2.n500.m50.txt	1000	7492.89	3.587%
7771.66162	MDG-a.2.n500.m50.txt	1500	7535.00	3.045%
7771.66162	MDG-a.2.n500.m50.txt	1700	7580.97	2.454%
778030.625	MDG-b.1.n500.m50.txt	500	743626.11	4.422%
778030.625	MDG-b.1.n500.m50.txt	1000	751129.72	3.458%
778030.625	MDG-b.1.n500.m50.txt	1500	757772.22	2.604%
778030.625	MDG-b.1.n500.m50.txt	1700	756298.48	2.793%
779963.6875	MDG-b.2.n500.m50.txt	500	743886.97	4.625%
779963.6875	MDG-b.2.n500.m50.txt	1000	751772.61	3.614%
779963.6875	MDG-b.2.n500.m50.txt	1500	759833.51	2.581%
779963.6875	MDG-b.2.n500.m50.txt	1700	758612.56	2.737%
24884110	MDG-c.1.n3000.m300.txt	500	23548413.40	5.368%
24884110	MDG-c.1.n3000.m300.txt	1000	23904991.20	3.935%
24884110	MDG-c.1.n3000.m300.txt	1500	24095666.20	3.168%
24884110	MDG-c.1.n3000.m300.txt	1700	24169696.40	2.871%
24905330	MDG-c.2.n3000.m300.txt	500	23519060.20	5.566%
24905330	MDG-c.2.n3000.m300.txt	1000	23916990.80	3.968%
24905330	MDG-c.2.n3000.m300.txt	1500	24116650.80	3.167%
24905330	MDG-c.2.n3000.m300.txt	1700	24122379.80	3.144%

Table 19: Resultados Calibración del Número de Generaciones

Análisis de Resultados:

El análisis de los resultados muestra que las configuraciones con menos iteraciones ($G = 500$ y $G = 1000$) no produjeron el mejor resultado en ninguna de las 10 instancias. El rendimiento óptimo se concentra en los valores más altos, con $G = 1700$ logrando el menor error en 6 instancias y $G = 1500$ en 4 instancias. A continuación, se detalla el desempeño entre estas dos configuraciones:

- **Instancias GKD-c:** $G = 1700$ fue superior en ambas instancias (GKD-c_1: 0.053% vs 0.065%; GKD-c_2: 0.037% vs 0.042%).
- **Instancias SOM-b:** Hubo un comportamiento mixto. $G = 1500$ fue superior en SOM-b_1 (3.784% vs 4.084%), pero $G = 1700$ fue mejor en SOM-b_2 (1.607% vs 1.774%).
- **Instancias MDG-a:** $G = 1700$ fue consistentemente superior, obteniendo el menor error en ambas instancias (MDG-a_1: 2.703% vs 3.087%; MDG-a_2: 2.454% vs 3.045%).
- **Instancias MDG-b:** $G = 1500$ obtuvo el menor error en ambas instancias (MDG-b_1: 2.604% vs 2.793%; MDG-b_2: 2.581% vs 2.737%).
- **Instancias MDG-c:** $G = 1700$ fue superior en MDG-c_1 (2.871% vs 3.168%), mientras que $G = 1500$ fue ligeramente mejor en MDG-c_2 (3.167% vs 3.144%).

En resumen, $G = 1700$ obtuvo el menor error en 6 de 10 casos, demostrando una clara superioridad general. Aunque $G = 1500$ mostró buen rendimiento en algunas instancias, particularmente en MDG-b, la configuración con $G = 1700$ proporciona mejoras consistentes y significativas en la mayoría de categorías de problemas.

Conclusión Final: Dado que $G = 1700$ logra los mejores resultados en la mayoría de instancias (60% de los casos) y ofrece mejoras sustanciales en categorías clave como GKD-c y MDG-a, se selecciona como configuración óptima **G=1700**, apostando esta vez por el valor del parámetro más alto con la esperanza de encontrar mejores soluciones aumentando el número de iteraciones.

Configuración final del Algoritmo Genético: Tras completar el proceso exhaustivo de calibración de parámetros, se ha determinado la siguiente configuración óptima para el algoritmo genético:

- **Tamaño de población (T):** 150 individuos
- **Número de generaciones (G):** 1700 generaciones
- **Probabilidad de cruce (PC):** 0.75 (75%)
- **Probabilidad de mutación (PM):** 0.1 (10%)

Esta configuración representa el mejor equilibrio encontrado entre calidad de solución y eficiencia computacional para el problema de diversidad máxima. Los parámetros han sido calibrados mediante experimentación sistemática sobre 10 instancias representativas de diferentes tipos y tamaños, asegurando su validez en un amplio espectro de casos.

3.4.2 Resultados AG

Una vez determinada la configuración óptima mediante el proceso de calibración ($T = 150$, $G = 1500$, $PC = 0.75$, $PM = 0.1$), se procedió a evaluar el rendimiento del algoritmo genético sobre un conjunto de 19 instancias representativas que no fueron utilizadas durante la fase de calibración. Cada instancia se ejecutó 5 veces de forma independiente, registrándose tanto el valor medio obtenido, así como el mejor resultado individual. Los resultados completos se presentan en la Tabla 20.

Instancia	Best Value	Resultado Medio		Mejor Resultado	
		Valor	Error (%)	Valor	Error (%)
GKD-c_19_n500_m50.txt	19477.3281	19465.49918	0.061%	19471.61825	0.029%
GKD-c_20_n500_m50.txt	19604.84375	19590.2779	0.074%	19602.00706	0.015%
SOM-b_4_n100_m40.txt	4142	4123.2	0.454%	4134	0.193%
SOM-b_8_n200_m80.txt	16225	16177.8	0.291%	16220	0.031%
SOM-b_12_n300_m120.txt	35881	35731.8	0.416%	35819	0.173%
SOM-b_16_n400_m160.txt	62487	62080.6	0.650%	62148	0.543%
SOM-b_20_n500_m200.txt	97344	96856.6	0.501%	96957	0.398%
MDG-a_19_n500_m50.txt	7755.41064	7549.954	2.649%	7577.51	2.294%
MDG-a_20_n500_m50.txt	7733.86133	7536.71	2.549%	7562.81	2.212%
MDG-a_39_n2000_m200.txt	114201	110584.4	3.167%	110780	2.996%
MDG-a_40_n2000_m200.txt	114191	110555.8	3.183%	110688	3.068%
MDG-b_19_n500_m50.txt	778802.9375	758263.224	2.637%	762006	2.157%
MDG-b_20_n500_m50.txt	778644.8125	756219.408	2.880%	762741.41	2.042%
MDG-b_39_n2000_m200.txt	1129505.42	10963803.33	2.933%	10972563.94	2.855%
MDG-b_40_n2000_m200.txt	1130710.48	10944691.96	3.205%	10962181.09	3.051%
MDG-c_5_n3000_m300.txt	24899703	24103052.6	3.199%	24124511	3.110%
MDG-c_10_n3000_m400.txt	43476251	42245586.2	2.831%	42333594	2.628%
MDG-c_15_n3000_m500.txt	66992877	65454681.2	2.296%	65500821	2.227%
MDG-c_20_n3000_m600.txt	95643586	93503486.6	2.238%	93575057	2.163%

Table 20: Resultados del Algoritmo Genético (AG)

Los resultados obtenidos muestran que el algoritmo genético presenta un rendimiento aceptable para instancias pequeñas (GKD-c y SOM-b), manteniendo errores generalmente inferiores al 0.7%. Sin embargo, para instancias de mayor tamaño y complejidad (MDG-a, MDG-b, MDG-c), los errores aumentan significativamente, alcanzando valores entre el 2.2% y 3.2%. Por otro lado, al comparar estos resultados con los obtenidos mediante Temple Simulado (Véase Tabla 8), se observa una diferencia notable en la calidad de las soluciones. A continuación, en la Tabla 21 se presenta un análisis comparativo por categorías de instancias:

Tipo de Instancia	Algoritmo Genético (AG)		Temple Simulado (TS)	
	Error Medio	Error Mejor	Error Medio	Error Mejor
GKD-c	0.0675%	0.0219%	0.0043%	0.0000065%
SOM-b	0.4623%	0.2674%	0.1083%	0.0399%
MDG-a	2.8872%	2.6422%	1.1523%	0.8567%
MDG-b	2.9138%	2.5262%	1.3450%	0.6134%
MDG-c	2.6409%	2.5319%	0.6775%	0.5278%
Promedio Total	1.7943%	1.5979%	0.6575%	0.4075%

Table 21: Comparativa de rendimiento: Algoritmo Genético vs Temple Simulado

Principales hallazgos:

1. **Diferencia significativa en rendimiento:** El temple simulado supera consistentemente al algoritmo genético en todas las categorías de instancias, presentando errores promedio que son menos de la mitad de los obtenidos por el AG.

2. **Comportamiento diferenciado por tamaño de instancia:** Para instancias pequeñas (GKD-c y SOM-b), ambas técnicas ofrecen resultados aceptables, aunque el TS mantiene una clara ventaja cuantitativa. En contraste, para instancias grandes (MDG), la diferencia se acentúa notablemente.
3. **Consistencia en la calidad de soluciones:** El TS no solo obtiene mejores resultados promedio, sino que también logra soluciones individuales de mayor calidad en todas las instancias evaluadas.

El rendimiento inferior del Algoritmo Genético convencional se atribuye a su **población inicial aleatoria** y a la **convergencia prematura** asociada a soluciones de baja calidad. Para mejorar el desempeño, se implementó una versión que usa un **procedimiento greedy con reinicio** para generar la población inicial de forma híbrida (50% greedy, 50% aleatoria). Esta modificación garantiza que la evolución comience con soluciones de mayor calidad, manteniendo los parámetros óptimos calibrados ($T = 150$, $G = 1700$, $PC = 0.75$, $PM = 0.1$).

La nueva implementación produjo una mejora dramática en el rendimiento, como se evidencia en la Tabla 22, haciendo a esta nueva versión competitiva.

Instancia	Best Value	Resultado Medio		Mejor Resultado	
		Valor	Error (%)	Valor	Error (%)
GKD-c_19_n500_m50.txt	19477.3281	19476.24642	0.0000555	19477.32577	0.0000001
GKD-c_20_n500_m50.txt	19604.84375	19604.84356	0.0000000	19604.84356	0.0000000
SOM-b_4_n100_m40.txt	4142	4127.4	0.3525	4137	0.1207
SOM-b_8_n200_m80.txt	16225	16187.6	0.2305	16212	0.0801
SOM-b_12_n300_m120.txt	35881	35821	0.1672	35844	0.1031
SOM-b_16_n400_m160.txt	62487	62275	0.3393	62346	0.2256
SOM-b_20_n500_m200.txt	97344	97079.4	0.2718	97164	0.1849
MDG-a_19_n500_m50.txt	7755.41064	7655.98	1.2821	7672.97	1.0630
MDG-a_20_n500_m50.txt	7733.86133	7661.128	0.9405	7689.63	0.5719
MDG-a_39_n2000_m200.txt	114201	112759.8	1.2620	112933	1.1103
MDG-a_40_n2000_m200.txt	114191	113002.2	1.0411	113014	1.0307
MDG-b_19_n500_m50.txt	778802.9375	769626.032	1.1783	770856.53	1.0203
MDG-b_20_n500_m50.txt	778644.8125	766910.526	1.5070	768688.79	1.2786
MDG-b_39_n2000_m200.txt	11295054.2	11167721.16	1.1273	11178685.15	1.0303
MDG-b_40_n2000_m200.txt	11307104.8	11174072.36	1.1765	11184595.7	1.0835
MDG-c_5_n3000_m300.txt	24899703	24654541	0.9846	24667562	0.9323
MDG-c_10_n3000_m400.txt	43476251	43113794	0.8337	43133097	0.7893
MDG-c_15_n3000_m500.txt	66992877	66609140	0.0573	66630770	0.5405
MDG-c_20_n3000_m600.txt	95643586	95028378	0.6432	95032648	0.6388

Table 22: Resultados del Algoritmo Genético Greedy (AGG)

Para cuantificar el impacto de la mejora implementada, la Tabla 23 presenta un análisis comparativo entre el AG convencional y el AGGreedy:

Tipo de Instancia	Error Medio AG (%)	Error Medio AGG (%)	Reducción de Error (%)
GKD-c	0.0675	0.0000278	99.96
SOM-b	0.4623	0.2722	41.12
MDG-a	2.8872	1.1314	60.82
MDG-b	2.9138	1.2473	57.19
MDG-c	2.6409	0.6292	76.17
Promedio Total	1.7943	0.6560	63.44

Table 23: Impacto de la inicialización greedy en el rendimiento del algoritmo

Nota: Reducción de Error = $(1 - \frac{Error\ AGG}{Error\ AG}) \times 100$, representa el porcentaje del error de AG que es eliminado por AGG.

3.5 Búsqueda en Entorno Variable

La Búsqueda en Entorno Variable es una metaheurística de búsqueda local que explora sistemáticamente diferentes niveles de perturbación alrededor de una solución actual. Su principio fundamental radica en la alternancia estructurada entre fases de **perturbación** (mediante movimientos de intercambio a diferentes escalas) y fases de **intensificación** (mediante Búsqueda en Entorno Variable con Descenso, VND).

Los pasos principales para VNS son:

Paso 0: Inicialización. Se genera una solución inicial factible mediante una heurística constructiva. Para el MDP, se emplea el procedimiento `Construir_Solucion_Inicial` que selecciona iterativamente los elementos que maximizan la diversidad marginal. La solución se representa como un vector de enteros S_{global} de longitud m que contiene los índices de los elementos seleccionados. Inicialmente se calcula la diversidad total mediante `Calcular_Diversidad` y se aplica una primera mejora con VND.

Paso 1 Definición de niveles de perturbación. Se establece una jerarquía de K_{max} niveles de perturbación, donde cada nivel k (con $k = 1, \dots, K_{\text{max}}$) está asociado a un número de intercambios calculado como:

$$k_{\text{shake_actual}} = \max(1, \lfloor \text{Frac_Perturbacion}(k) \times m \rfloor)$$

Los valores $\text{Frac_Perturbacion}(k)$ determinan la fracción de elementos a intercambiar en cada nivel, permitiendo perturbaciones desde cambios sutiles hasta modificaciones estructurales significativas.

Paso 2 Ciclo principal VNS. Se ejecuta un bucle controlado por dos criterios de parada: número máximo de iteraciones (`Max_Iteraciones`), Iteraciones consecutivas sin mejora (`Max_Sin_Mejora`). Dentro de cada iteración, se exploran secuencialmente todos los niveles de perturbación:

- **Perturbación k -swap:** Dada la solución actual S_{global} y el nivel actual k , se genera una solución perturbada S_{local} aplicando $k_{\text{shake_actual}}$ intercambios simultáneos mediante la rutina `Perturbacion_kSwap`.
- **Búsqueda local VND:** Sobre la solución perturbada, se aplica el procedimiento VND que explora sistemáticamente diferentes entornos de mejora local para obtener una solución mejorada $S_{\text{temp_vnd}}$.
- **Criterio de aceptación:** Si la diversidad de $S_{\text{temp_vnd}}$ supera a la mejor conocida (con tolerancia 10^{-8}), se actualiza la solución global y se reinicia la exploración desde el primer nivel ($k = 1$). En caso contrario, se avanza al siguiente nivel de perturbación.

Paso 3 Control de convergencia. Se mantienen contadores para:

- Iteraciones totales realizadas
- Iteraciones consecutivas sin mejora significativa
- Mejora relativa respecto a la iteración anterior

El algoritmo termina cuando se satisface alguno de los criterios de parada establecidos.

Nomenclatura

$S_g = S_{\text{global}}$	$z_g = z_{\text{global}}$
$S_l = S_{\text{local}}$	$z_a = z_{\text{anterior}}$
$S_t = S_{\text{temp_vnd}}$	$z_i = z_{\text{vnd_init}}$
$\text{Iter} = \text{Iteraciones}$	$z_f = z_{\text{vnd_final}}$
$\text{ism} = \text{iter_sin_mejora}$	$\text{hubo} = \text{hubo_mejora}$

Algorithm 5 Algoritmo VNS para el MDP

Require: K_{\max} : niveles máx. perturbación,
Require: $\text{Frac_P}[K_{\max}]$: fracciones perturbación,
Require: Max_Iter : iteraciones máx.,
Require: Max_Sin_Mej : iteraciones sin mejora
Ensure: Iter : iteraciones ejecutadas,
Ensure: Mejor_Z : diversidad mejor solución

- 1: **0. Inicialización**
- 2: $mloc \leftarrow m$
- 3: Construir_Solucion_Inicial(S_g, z_g)
- 4: $z_a \leftarrow z_g$
- 5: Reservar: $S_l[mloc], S_t[mloc]$
- 6: $z_i \leftarrow z_g$, VND(S_g, S_l, z_i, z_g), $S_g \leftarrow S_l$
- 7: $\text{Iter} \leftarrow 0$, $\text{ism} \leftarrow 0$, $\text{hubo} \leftarrow \text{VERDAD}$
- 8: **1. Niveles de perturbación**
- 9: $k_s = \max(1, \lfloor \text{Frac_P}[k] \times m \rfloor)$, $k = 1, \dots, K_{\max}$
- 10: **2. Ciclo principal VNS**
- 11: **while** ($\text{Iter} < \text{Max_Iter}$) \wedge ($\text{ism} < \text{Max_Sin_Mej}$) **do**
- 12: $k_v \leftarrow 1$, $\text{hubo} \leftarrow \text{FALSO}$
- 13: **while** $k_v \leq K_{\max}$ **do**
- 14: **2.1 Perturbación k -swap**
- 15: $k_s \leftarrow \max(1, \lfloor \text{Frac_P}[k_v] \times mloc \rfloor)$
- 16: Perturbacion_kSwap(S_g, S_l, k_s)
- 17: **2.2 Búsqueda local VND**
- 18: $z_i \leftarrow \text{Calcular_Diversidad}(S_l)$
- 19: VND(S_l, S_t, z_i, z_f)
- 20: **2.3 Criterio de aceptación**
- 21: **if** $z_f > z_g + 10^{-8}$ **then**
- 22: $S_g \leftarrow S_t$, $z_g \leftarrow z_f$, $k_v \leftarrow 1$, $\text{hubo} \leftarrow \text{VERDAD}$
- 23: **else**
- 24: $k_v \leftarrow k_v + 1$
- 25: **end if**
- 26: **end while**
- 27: **3. Control de convergencia**
- 28: $\text{Iter} \leftarrow \text{Iter} + 1$
- 29: **if** hubo **then**
- 30: $\text{ism} \leftarrow 0$
- 31: **else**
- 32: $\text{ism} \leftarrow \text{ism} + 1$
- 33: **end if**
- 34: **if** $|z_g - z_a| \geq 10^{-8}$ **then**
- 35: $z_a \leftarrow z_g$
- 36: **end if**
- 37: **end while**
- 38: **4. Finalización**
- 39: $\text{Mejor_Z} \leftarrow z_g$
- 40: Liberar memoria: S_g, S_l, S_t
- 41: **return** ($\text{Iter}, \text{Mejor_Z}$) = 0

Características clave de la implementación VNS para el MDP:

- **Perturbación escalonada:** Los niveles de perturbación permiten desde ajustes finos (k pequeño) hasta reestructuraciones profundas (k grande), facilitando tanto la explotación local como la exploración global.
- **Integración con VND:** El uso de Búsqueda en Entorno Variable con Descenso como procedimiento de intensificación asegura una mejora exhaustiva de las soluciones perturbadas antes de su evaluación.

- **Criterios de parada adaptativos:** La combinación de límites absolutos (iteraciones máximas) y relativos (iteraciones sin mejora) proporciona un balance entre tiempo de ejecución y calidad de solución.
- **Tolerancia numérica:** El uso de una tolerancia 10^{-8} en las comparaciones evita ciclos infinitos debido a diferencias numéricas insignificantes.
- **Reinicio estratégico:** Al encontrar una mejora, se reinicia la exploración desde el primer nivel de perturbación, aprovechando que la nueva solución puede beneficiarse de ajustes más sutiles.

3.5.1 Calibración VNS

Calibración de los Niveles de Perturbación.

Para determinar la configuración óptima del parámetro de fracción de perturbaciones en algoritmos de búsqueda en entorno variable, se evaluaron cuatro configuraciones diferentes que varían en el número de niveles de perturbación (k_{\max}) y los valores de las fracciones aplicadas.

- Fracciones de perturbación: [0.10, 0.25], [0.15, 0.35], [0.08, 0.20, 0.40] y [0.12, 0.25, 0.5].
- Máximo número de iteraciones 15
- Máximo número de iteraciones sin mejora 8
- Repeticiones por conjunto de fracciones de perturbación : 3 ejecuciones independientes

Los resultados completos se presentan en la Tabla 24.

Best Value	Instancia	k_{\max}	Niveles de Perturbación	Diversidad	Error (%)
19485.1875	GKD-c_1_n500_m50.txt	2	[0.10, 0.25]	19485.18697	0.00000272
		2	[0.15, 0.35]	19485.18697	0.00000272
		3	[0.08, 0.20, 0.40]	19485.18697	0.00000272
		3	[0.12, 0.25, 0.5]	19485.18697	0.00000272
19701.5371	GKD-c_2_n500_m50.txt	2	[0.10, 0.25]	19701.53477	0.00001188
		2	[0.15, 0.35]	19701.53477	0.00001188
		3	[0.08, 0.20, 0.40]	19701.53477	0.00001188
		3	[0.12, 0.25, 0.5]	19701.53477	0.00001188
333	SOM-b_1_n100_m10.txt	2	[0.10, 0.25]	323	3.003
		2	[0.15, 0.35]	329.33333	1.101
		3	[0.08, 0.20, 0.40]	331.33333	0.501
		3	[0.12, 0.25, 0.5]	329.66667	1.001
1195	SOM-b_2_n100_m20.txt	2	[0.10, 0.25]	1195	0.000
		2	[0.15, 0.35]	1188.33	0.558
		3	[0.08, 0.20, 0.40]	1191.66667	0.279
		3	[0.12, 0.25, 0.5]	1195	0.000
7833.8325	MDG-a_1_n500_m50.txt	2	[0.10, 0.25]	7734.37333	1.270
		2	[0.15, 0.35]	7710.57	1.573
		3	[0.08, 0.20, 0.40]	7700.76	1.700
		3	[0.12, 0.25, 0.5]	7746.08	1.120
7771.6616	MDG-a_2_n500_m50.txt	2	[0.10, 0.25]	7700.75	0.912
		2	[0.15, 0.35]	7729.076	0.548
		3	[0.08, 0.20, 0.40]	7670.82667	1.297
		3	[0.12, 0.25, 0.5]	7713.06333	0.754
778030.625	MDG-b_1_n500_m50.txt	2	[0.10, 0.25]	773549.8633	0.576
		2	[0.15, 0.35]	771820.0233	0.798
		3	[0.08, 0.20, 0.40]	771918.0166	0.789
		3	[0.12, 0.25, 0.5]	769731.0166	1.061
779963.688	MDG-b_2_n500_m50.txt	2	[0.10, 0.25]	770675.2833	1.191
		2	[0.15, 0.35]	769388.8366	1.356
		3	[0.08, 0.20, 0.40]	772303.83	0.982
		3	[0.12, 0.25, 0.5]	774203.1833	0.739

Table 24: Resultados de calibración de Niveles de Perturbación

Análisis de resultados:

El análisis de los resultados muestra que los conjuntos de perturbación más agresivos (valores más altos de fracción) no siempre produjeron los mejores resultados, concentrándose el rendimiento óptimo en configuraciones moderadas. A continuación, se detalla la comparativa de error entre las configuraciones evaluadas:

- **Instancias GKD-c:** Todas las configuraciones alcanzaron errores extremadamente bajos (menos de 0.00002%), demostrando que estas instancias son fáciles de resolver y que la calibración de perturbación tiene un impacto mínimo en su desempeño.
- **Instancias SOM-b:** Se observó la mayor sensibilidad a la configuración. Para SOM-b_1, la configuración **[0.08, 0.20, 0.40]** ($k_{\max} = 3$) obtuvo el menor error (0.501%), mientras que **[0.10, 0.25]** tuvo el peor (3.003%). Para SOM-b_2, tanto **[0.10, 0.25]** como **[0.12, 0.25, 0.50]** alcanzaron el óptimo (0% error).
- **Instancias MDG-a:** La configuración **[0.10, 0.25]** ($k_{\max} = 2$) mostró el mejor desempeño en MDG-a_1 (1.270% error) y fue competitiva en MDG-a_2 (0.912% vs 0.548% del segundo mejor). Las configuraciones más agresivas obtuvieron errores mayores, hasta 1.700%.
- **Instancias MDG-b:** El comportamiento fue mixto. Para MDG-b_1, **[0.10, 0.25]** obtuvo el menor error (0.576%), mientras que para MDG-b_2, **[0.12, 0.25, 0.50]** fue superior (0.739%). Las configuraciones con perturbaciones mayores tendieron a tener peores resultados.

En resumen, aunque no hay un claro ganador único en todas las instancias, la configuración **[0.10, 0.25]** ($k_{\max} = 2$) demostró la mayor consistencia, siendo la mejor o igual a la mejor en 5 de las 8 instancias evaluadas.

Conclusión final: Aunque algunas configuraciones con 3 niveles obtuvieron resultados ligeramente mejores en instancias específicas, se selecciona como configuración óptima **Frac_Perturbacion = [0.10, 0.25]** con $k_{\max} = 2$ por ofrecer el mejor equilibrio entre simplicidad, consistencia y eficiencia computacional. Esta configuración proporciona perturbaciones moderadas que equilibran adecuadamente la exploración y explotación del espacio de búsqueda.

3.5.2 Resultados VNS

El algoritmo VNS configurado con **Frac_Perturbacion = [0.15, 0.35]** y $k_{\max} = 2$ demuestra un desempeño notablemente consistente y eficaz sobre las 17 instancias del conjunto de test, que abarcan desde problemas pequeños ($n=100$) hasta muy grandes ($n=3000$).

Instancia	Best Value	Resultado Medio		Mejor Resultado	
		Valor	Error (%)	Valor	Error (%)
GKD-c_19_n500_m50.txt	19477.3281	19477.3257	1.23×10^{-5}	19477.32577	1.20×10^{-5}
GKD-c_20_n500_m50.txt	19604.84375	19604.84356	9.69×10^{-7}	19604.84356	9.69×10^{-7}
SOM-b_4_n100_m40.txt	4142	4142	0.00000	4142	0.00000
SOM-b_8_n200_m80.txt	16225	16225	0.00000	16225	0.00000
SOM-b_12_n300_m120.txt	35881	35859.67	0.05946	35866	0.04180
SOM-b_16_n400_m160.txt	62487	62337.67	0.23898	62446	0.06561
SOM-b_20_n500_m200.txt	97344	97188.66	0.15958	97295	0.05034
MDG-a_19_n500_m50.txt	7755.41064	7682.26	0.94322	7710.65	0.57715
MDG-a_20_n500_m50.txt	7733.86133	7633.433	1.29855	7687.91	0.59416
MDG-a_39_n2000_m200.txt	114201	112560	1.43694	112832	1.19876
MDG-a_40_n2000_m200.txt	114191	112731.33	1.27827	112836	1.18661
MDG-b_19_n500_m50.txt	778802.9375	772319.41	0.83250	774893.11	0.50203
MDG-b_20_n500_m50.txt	778644.8125	769925.903	1.11975	773085.66	0.71395
MDG-b_39_n2000_m200.txt	11295054.2	11152466.24	1.26239	11179563.4	1.02249
MDG-b_40_n2000_m200.txt	11307104.8	11148672.85	1.40117	11158556.85	1.31376
MDG-c_5_n3000_m300.txt	24899703	24621708.66	1.11646	24642462	1.03311
MDG-c_10_n3000_m400.txt	43476251	43052591	0.97446	43066888	0.94158
MDG-c_15_n3000_m500.txt	66992877	66462328	0.79195	66550448	0.66041
MDG-c_20_n3000_m600.txt	95643586	94909248	0.76779	94942765	0.73274

Table 25: Resultados del Algoritmo VNS

A continuación se presentan los hallazgos clave:

- **Excelencia en instancias GKD-c:** El algoritmo alcanza soluciones prácticamente idénticas al óptimo conocido, con errores mínimos del orden de $10^{-5}\%$ a $10^{-7}\%$, confirmando la efectividad de la configuración de perturbación para este tipo de problemas.
- **Óptimos exactos en SOM-b:** Para las instancias SOM-b_4 y SOM-b_8 se obtienen soluciones con 0% de error, mientras que en las instancias más grandes (SOM-b_12 a SOM-b_20) los errores se mantienen por debajo del 0.24%, demostrando una excelente capacidad de escalabilidad.
- **Consistencia en instancias MDG:** El algoritmo muestra un comportamiento predecible y robusto:
 - Para tamaños moderados ($n=500$): errores entre 0.50-1.30%
 - Para tamaños grandes ($n=2000$): errores entre 1.02-1.44%
 - Para tamaños muy grandes ($n=3000$): errores entre 0.66-1.12%

Notablemente, el desempeño en las instancias MDG-c ($n=3000$) es superior al observado en MDG-a y MDG-b de menor tamaño.

- **Escalabilidad probada:** El algoritmo mantiene errores inferiores al 1.5% incluso en las instancias más grandes ($n=3000$, $m=600$), demostrando que la configuración de perturbación [15%, 35%] maneja efectivamente el crecimiento de la dimensionalidad.

Conclusión: La configuración VNS con perturbaciones del 15% y 35% establece un balance óptimo entre exploración y explotación, logrando precisiones excepcionales en instancias pequeñas/medianas (0-0.24% error) y manteniendo robustez en instancias muy grandes (0.66-1.44% error). La consistencia entre réplicas y la escalabilidad demostrada validan esta configuración como efectiva para el problema MDP en un amplio rango de tamaños y tipos de instancias.

4 COMPARACIÓN DE ALGORITMOS

Para realizar una evaluación justa y comparable entre los diferentes algoritmos, se estableció la siguiente metodología experimental: dado que los algoritmos AG y VNS tienen un mayor coste computacional, se limitó su tiempo de ejecución a 30 y 60 segundos. Es posible que para instancia de tamaño pequeño no gaste el tiempo límite, pero para otras instancias de gran tamaños nos aseguramos de no sobrepasar ese límite. En contraste, los métodos más simples (Constructivo, Búsqueda Local y Temple Simulado) se ejecutaron sin límite temporal debido a su menor coste ya que su ejecución es prácticamente instantánea. Cada configuración se ejecutó 3 veces independientes, reportándose tanto la media de las ejecuciones (indicador de consistencia) como el mejor resultado obtenido (indicador de potencial). Todos los resultados se expresan como porcentaje de error relativo respecto al valor óptimo conocido para cada instancia, permitiendo una comparación directa entre algoritmos.

Instance	Constructivo	Búsqueda Local	Temple Simulado	30 seg - AG	60 seg - AG	30 seg - VNS	60 seg - VNS
GKD-c_19 n500_m50.txt	0,289	1,21e-5	1,21e-5	0,013	0,000042	1,23e-5	1,59e-5
GKD-c_20 n500_m50.txt	0,154	9,69e-7	9,69e-7	0,000000	9,69e-7	1,28e-6	1,91e-5
SOM-b_4 n100_m40.txt	1,04	0,121	0,000	0,169	0,121	0,000	0,000
SOM-b_8 n200_m80.txt	0,814	0,629	0,000	0,049	0,117	0,000	0,000
SOM-b_12 n300_m120.txt	0,805	0,042	0,022	0,125	0,125	0,086	0,000
SOM-b_16 n400_m160.txt	0,634	0,384	0,134	0,262	0,224	0,162	0,112
SOM-b_20 n500_m200.txt	0,569	0,206	0,103	0,219	0,242	0,379	0,013
MDG-a_19 n500_m50.txt	1,86	1,44	0,952	0,792	0,333	8,25e-6	0,652
MDG-a_20 n500_m50.txt	2,37	1,97	0,652	0,668	0,541	0,471	0,414
MDG-a_39 n2000_m200.txt	1,79	1,15	0,871	1,074	1,001	1,424	1,570
MDG-a_40 n2000_m200.txt	1,51	0,713	0,997	0,922	0,866	1,287	1,089
MDG-b_19 n500_m50.txt	2,33	1,21	0,566	0,902	0,750	0,382	0,659
MDG-b_20 n500_m50.txt	2,62	2,43	1,06	1,266	1,060	1,04e-4	2,09e-5
MDG-b_39 n2000_m200.txt	1,53	1,18	0,788	0,200	0,858	1,287	1,270
MDG-b_40 n2000_m200.txt	1,61	0,936	0,750	1,084	0,943	1,266	1,242
MDG-c_5 n3000_m300.txt	1,19	0,888	0,851	0,949	0,856	1,060	1,120
MDG-c_10 n3000_m400.txt	1,11	0,494	0,705	0,821	0,777	0,997	1,003
MDG-c_15 n3000_m500.txt	0,83	0,261	0,432	0,592	0,557	0,786	0,743
MDG-c_20 n3000_m600.txt	0,737	0,394	0,434	0,670	0,614	0,739	0,798

Table 26: Comparación Mejores Resultados (Error %)

La Tabla 26 presenta los mejores resultados individuales obtenidos por cada algoritmo, revelando una clara especialización según el tipo de instancia. Para problemas GKD-c, la Búsqueda Local y el Temple Simulado alcanzan los menores errores (10^{-5} - 10^{-7} %). En instancias SOM-b, el algoritmo VNS con 60 segundos logra errores nulos en tres de cinco casos (SOM-b_4, SOM-b_8 y SOM-b_12), lo que sugiere que ha alcanzado el óptimo conocido o soluciones extremadamente cercanas al mismo. El Temple Simulado también muestra excelente rendimiento en tamaños pequeños de esta categoría. Para las categorías MDG, ningún algoritmo domina consistentemente, aunque VNS-60 obtiene buenos resultados en varias instancias MDG-a y MDG-b, con errores que oscilan entre 0.414% y 1.570%.

Instance	Constructivo	Búsqueda Local	Temple Simulado	30 seg - AG	60 seg - AG	30 seg - VNS	60 seg - VNS
GKD-c_19 n500_m50.txt	0,325	0,0000121	0,00501	0,017	0,014	0,0000123	0,0000121
GKD-c_20 n500_m50.txt	0,361	0,000000969	0,00482	0,000000969	0,005	0,0000191	0,000000969
SOM-b_4 n100_m40.txt	1,58	0,193	0	0,338	0,295	0,032	0,004
SOM-b_8 n200_m80.txt	0,922	0,727	0,0657	0,238	0,293	0,002	0,002
SOM-b_12 n300_m120.txt	1,05	0,186	0,0985	0,197	0,221	0,117	0,050
SOM-b_16 n400_m160.txt	0,852	0,402	0,163	0,391	0,277	0,229	0,164
SOM-b_20 n500_m200.txt	0,903	0,294	0,115	0,296	0,305	0,506	0,206
MDG-a_19 n500_m50.txt	2,55	1,68	1,13	1,017	0,994	0,503	0,502
MDG-a_20 n500_m50.txt	3,15	2,25	1,26	0,896	0,846	0,560	0,556
MDG-a_39 n2000_m200.txt	1,81	1,29	1,12	1,237	1,092	1,662	1,404
MDG-a_40 n2000_m200.txt	1,54	0,988	1,18	1,097	0,957	1,392	1,185
MDG-b_19 n500_m50.txt	2,76	1,74	1,55	1,153	0,930	0,664	0,502
MDG-b_20 n500_m50.txt	3,17	2,68	1,65	1,434	1,060	0,917	0,714
MDG-b_39 n2000_m200.txt	1,61	1,23	0,989	1,017	1,155	1,445	1,022
MDG-b_40 n2000_m200.txt	1,66	1,12	1,15	1,145	1,058	1,583	1,314
MDG-c_5 n3000_m300.txt	1,29	0,967	0,936	0,949	0,961	1,176	1,033
MDG-c_10 n3000_m400.txt	1,16	0,634	0,777	0,862	0,814	1,106	0,942
MDG-c_15 n3000_m500.txt	0,847	0,362	0,447	0,607	0,583	0,846	0,660
MDG-c_20 n3000_m600.txt	0,774	0,444	0,494	0,677	0,643	0,767	0,733

Table 27: Comparación Media Resultados (Error %)

La Tabla 27, que muestra los errores medios tras tres ejecuciones, revela patrones diferenciales de rendimiento entre los algoritmos evaluados. El algoritmo Constructivo presenta los mayores

errores en todas las categorías (0.29-3.17%), confirmando su limitación como método independiente. La Búsqueda Local muestra excelente rendimiento en instancias GKD-c con errores mínimos ($< 10^{-5}\%$) y buen desempeño en algunas instancias MDG-c, pero su efectividad disminuye notablemente en otras categorías. El Temple Simulado emerge como una alternativa sólida, obteniendo los menores errores medios en 8 de 19 instancias, particularmente destacando en categorías SOM-b (3/5 casos) y varias instancias MDG, con un equilibrio notable entre simplicidad y efectividad.

Analizando los algoritmos con límite temporal, VNS-60 segundos obtiene los menores errores medios en 6 instancias, mostrando fortaleza en categorías MDG-a y MDG-b, mientras que AG-60 segundos alcanza este logro en solo 2 instancias. Sin embargo, la comparación entre versiones de 30 y 60 segundos revela comportamientos distintos: AG experimenta mejoras al aumentar el tiempo en 12 de 19 instancias, mientras que VNS mejora en 11 instancias con mejoras más moderadas. Esto sugiere que AG se beneficia más del tiempo adicional, aunque parte de una base de calidad inferior.

La robustez de los algoritmos, medida por la diferencia entre mejores resultados individuales (Tabla 26) y medias, proporciona *insights* adicionales. VNS exhibe la mayor consistencia, con diferencias típicas de 0.05-0.3 puntos porcentuales entre sus mejores y medias ejecuciones. En contraste, AG muestra mayor variabilidad (0.1-0.5 puntos). El Temple Simulado demuestra sorprendente consistencia en instancias donde es efectivo, con diferencias mínimas entre ejecuciones. Para los algoritmos sin límite temporal, la Búsqueda Local presenta variabilidad moderada, mientras que el Constructivo, aunque con los peores resultados, muestra poca variación debido a su naturaleza determinística.

En conclusión, la elección del algoritmo óptimo para el MDP requiere considerar múltiples factores: tipo de instancia, tiempo disponible y prioridades de calidad vs consistencia. Para máxima calidad general con tiempo suficiente, VNS-60 segundos ofrece el mejor equilibrio. Cuando el tiempo es limitado (30 segundos), tanto VNS como AG proporcionan soluciones competitivas, con VNS mostrando ventaja en la mayoría de casos. Para problemas específicos, algoritmos especializados ofrecen alternativas valiosas: la Búsqueda Local es óptima para instancias GKD-c, el Temple Simulado para problemas SOM-b y MDG de tamaño moderado, y AG puede considerarse para instancias MDG-c muy grandes donde su exploración más amplia resulta beneficiosa. El algoritmo Constructivo, aunque con limitaciones en calidad, mantiene utilidad como inicializador para métodos más sofisticados.

5 CONCLUSIONES

A lo largo de este trabajo se han implementado y evaluado diversas técnicas heurísticas y meta-heurísticas para el problema de máxima diversidad. Los algoritmos constructivos resultaron rápidos pero de calidad limitada, mientras que la búsqueda local mostró eficacia en instancias pequeñas con tendencia al estancamiento. El temple simulado ofreció un buen equilibrio entre exploración y explotación, y los algoritmos genéticos mejoraron significativamente al incorporar inicialización greedy. La búsqueda en entorno variable parece ofrecer un buen rendimiento en líneas generales.

Como trabajos futuros, se propone recalibrar el algoritmo genético utilizando inicialización greedy con reinicio en toda la población, así como realizar una calibración exhaustiva de los parámetros de parada en VNS (`max_iter` y `max_iter_sin_mejora`). También sería valioso estudiar la interacción entre el número de generaciones y el tamaño de población en métodos evolutivos, además de explorar técnicas de hibridación y paralelización para mejorar el rendimiento en instancias a gran escala.