UNIVERSIDAD DE GRANADA

Problema de la Diversidad Máxima

Técnicas de Soft Computing para Aprendizaje y Optimización. Redes Neuronales y Metaheurísticas, Programación Evolutiva y Bioinspirada

---

Autor: Pablo Gradolph Oliva

Granada, marzo 2025

Este trabajo está licenciado bajo Creative Commons **Attribution – Non Commercial – Non Derivatives**

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Contexto y motivación

El Problema de la Diversidad Máxima (PDM) es un problema de optimización combinatoria que consiste en seleccionar un subconjunto de elementos de un conjunto mayor de tal manera que la diversidad entre los elementos seleccionados sea máxima. Formalmente, se define como la selección de un subconjunto de tamaño fijo que maximice la suma de las distancias entre los elementos elegidos.

Este problema surge en numerosos ámbitos, incluyendo la selección de jurados diversos, la configuración de carteras de inversión con activos no correlacionados, el diseño de experimentos en investigación científica y la distribución de infraestructuras en redes de comunicación. Debido a su formulación combinatoria y su alta complejidad computacional, el PDM es un problema NP-difícil, lo que significa que no existe un algoritmo exacto eficiente para resolverlo en instancias grandes, requiriendo en su lugar el uso de técnicas heurísticas y metaheurísticas para obtener soluciones aproximadas de alta calidad.

1.2. Estado del arte

Dado que el PDM es un problema computacionalmente desafiante, se han desarrollado diversas estrategias para su resolución, incluyendo enfoques exactos y heurísticos. Los métodos exactos, como la programación entera y los algoritmos de ramificación y acotación (Branch and Bound), solo son viables para instancias pequeñas del problema debido a su alta complejidad computacional.

Para instancias de mayor tamaño, se han explorado técnicas metaheurísticas que proporcionan soluciones cercanas al óptimo en tiempos razonables. Entre las principales estrategias destacan:

- **Greedy Randomized Adaptive Search Procedure (GRASP):** Es un algoritmo iterativo que combina una fase de construcción basada en heurísticas con una fase de búsqueda local para mejorar la solución generada.
- **Búsqueda Tabú (TS):** Utiliza una memoria adaptativa para evitar ciclos en la búsqueda y escapar de óptimos locales mediante un mecanismo de penalización de movimientos previamente explorados.
- **Algoritmos Genéticos (GA):** Se basan en la evolución natural y utilizan operadores genéticos como selección, cruce y mutación para explorar el espacio de soluciones.

- **Variable Neighborhood Search (VNS):** Explora diferentes estructuras de vecindad para diversificar la búsqueda y evitar estancamientos en soluciones subóptimas.
- **Iterated Greedy (IG):** Alterna fases de destrucción y reconstrucción de soluciones para encontrar soluciones de alta calidad.

En particular, los algoritmos genéticos han demostrado ser una alternativa prometedora para el PDM, debido a su capacidad de exploración global y su flexibilidad para adaptarse a diferentes instancias del problema. El estudio comparativo entre estos métodos es un área activa de investigación, ya que el rendimiento de cada enfoque puede depender de la naturaleza específica de las instancias y de la parametrización utilizada.

1.3. Objetivos del trabajo

El objetivo de este trabajo es el desarrollo y la mejora de algoritmos genéticos (GA) para abordar el Problema de la Diversidad Máxima (PDM). Se ha partido de una implementación base de GA y se han realizado optimizaciones en dos niveles: (1) mejoras en la eficiencia computacional, con el uso de paralelización y optimización en CPU, y (2) mejoras en la calidad de las soluciones, integrando técnicas avanzadas como búsqueda local, aceptación de soluciones peores y estrategias de intensificación y diversificación.

Dado el marco temporal del trabajo, no se compararán diferentes enfoques metaheurísticos (como GRASP, VNS o Tabu Search), sino que se enfocará exclusivamente en la optimización del algoritmo genético. La comparación de resultados se realizará con los mejores valores reportados en el trabajo de Lozano et al. (2011) sobre el enfoque Iterated Greedy (IG)[3], utilizando las mismas instancias de prueba.

2. FORMULACIÓN DEL PROBLEMA

2.1. Definición matemática del PDM

El Problema de la Diversidad Máxima (PDM) se define formalmente de la siguiente manera:

Dado un conjunto de elementos $N = e_1, e_2, \ldots, e_n$ y una función de distancia $d(e_i, e_j)$ que mide la diversidad entre dos elementos e_i y e_j, el objetivo es seleccionar un subconjunto $M \subset N$ de tamaño m que maximice la suma de las distancias entre los elementos seleccionados:

$$\text{máx} \sum_{i \in M} \sum_{j \in M, j > i} d(e_i, e_j) \tag{2.1}$$

sujeto a:

$$|M| = m, \quad M \subset N. \tag{2.2}$$

Este problema es de naturaleza combinatoria y pertenece a la clase NP-difícil, lo que implica que no se puede resolver de manera exacta en tiempos razonables para instancias grandes.

2.2. Instancias utilizadas

En este trabajo se utilizan 30 instancias seleccionadas de los siguientes conjuntos:

- **MDG-a**: 10 instancias con distancias enteras en $[0, 10]$, $n = 500$, $m = 50$.
- **MDG-b**: 10 instancias con distancias reales en $[0, 1000]$, $n = 2000$, $m = 200$.
- **MDG-c**: 10 instancias con distancias enteras en $[0, 1000]$, $n = 3000$, m variando entre 300 y 600.

Cada instancia se representa con una matriz de distancias que contiene $n(n-1)/2$ valores, almacenados en la diagonal superior.

2.3. Criterio de evaluación de soluciones

La calidad de una solución se mide a través de la **suma de las distancias** entre los elementos seleccionados. Dado que el PDM busca maximizar la diversidad, una solución es

mejor cuanto mayor sea su valor objetivo. Además, se consideran otros criterios secundarios como el tiempo de ejecución y la estabilidad de los resultados obtenidos en múltiples ejecuciones.

3. MARCO EXPERIMENTAL

3.1. Fases de desarrollo

Para la experimentación, se ha seguido una estrategia de implementación progresiva en tres fases:

1. **Algoritmo Genético Base (GA):** Se implementó un algoritmo genético estándar con los operadores básicos de selección, cruce y mutación. Se estableció una población inicial aleatoria y se ejecutó hasta un número fijo de generaciones. Este modelo sirvió como referencia inicial para evaluar mejoras posteriores.

2. **Optimización computacional:** Se aplicaron diversas estrategias para mejorar la velocidad de ejecución del GA:

   - **Paralelización en CPU:** Uso de joblib para distribuir ejecuciones en múltiples núcleos.
   - **Aceleración con Numba y Vectorización:** Compilación JIT (just-in-time) de funciones clave para acelerar cálculos.

3. **Mejoras en la calidad de las soluciones:** Se integraron diversas estrategias para mejorar la diversidad máxima obtenida:

   - **Búsqueda Local Fast Lin-Kernighan (FastLK):** Optimización local de soluciones para evitar estancamientos en óptimos locales.
   - **Aceptación de soluciones peores con Random Walk (RW):** Permitir la exploración de soluciones subóptimas bajo ciertas condiciones.
   - **Estrategia Iterated Greedy (IG):** Uso de fases de destrucción y reconstrucción parcial para diversificar la búsqueda.
   - **Mutación Adaptativa:** Modifica dinámicamente la tasa de mutación dependiendo del progreso del algoritmo.

3.2. Instancias de prueba

Como ya se ha mencionado anteriormente, para evaluar el rendimiento del algoritmo, se utilizaron instancias estándar del PDM de los conjuntos MDG-a, MDG-b y MDG-c, idénticos a las reportados en Lozano et al. (2011)[3]. Cada instancia contiene una matriz de distancias y un número objetivo de elementos a seleccionar.

3.3. Comparación de resultados

Se comparó el desempeño del algoritmo optimizado con dos referencias:

1. **GA Base:** Se midió cuánto mejoraron las soluciones y la eficiencia computacional tras las mejoras.
2. **Resultados del paper de Lozano et al. (2011)[3]:** Se compararon los valores de diversidad máxima obtenidos con los reportados en la literatura.

Se realizaron múltiples ejecuciones de cada algoritmo sobre cada conjunto de datos obteniendo métricas clave para el problema como son la diversidad máxima, la diversidad media, la desviación y el tiempo de ejecución. El tiempo de ejecución se midió a mano con la propia herramienta de jupyter notebook que mide el tiempo de ejecución de cada celda de código.

4. RESULTADOS

4.1. Introducción a los resultados

En esta sección se presentan los resultados obtenidos tras la implementación y mejora del algoritmo genético (GA) para la resolución del Problema de la Diversidad Máxima (PDM). Se ha seguido un enfoque progresivo de optimización, en el que inicialmente se han evaluado las soluciones obtenidas con un **GA base (GA-B)** y posteriormente se han incorporado mejoras tanto en eficiencia computacional como en calidad de soluciones.

Dado que las instancias utilizadas en este estudio pertenecen a tres conjuntos de datos distintos, se ha realizado una división del análisis en tres categorías:

- **MDG-a**: Instancias con $n = 500$ y $m = 50$.
- **MDG-b**: Instancias con $n = 2000$ y $m = 200$.
- **MDG-c**: Instancias con $n = 3000$ y m variando entre 300 y 600.

Debido al alto coste computacional asociado a la ejecución del algoritmo genético sobre instancias grandes, se ha adoptado una estrategia escalonada de mejora. En primer lugar, todas las optimizaciones se han aplicado y evaluado sobre el conjunto de datos **MDG-a**. Esto permitió analizar el impacto de cada mejora de manera eficiente antes de extenderlas a instancias más grandes. Posteriormente, una vez validadas las optimizaciones, estas se han implementado sobre los conjuntos **MDG-b** y **MDG-c**.

Las mejoras introducidas en el algoritmo se dividen en dos grandes categorías:

1. **Optimización del rendimiento computacional**: Se han implementado las técnicas mencionadas en la sección anterior para acelerar la ejecución del GA.
2. **Mejoras en la calidad de las soluciones**: Una vez optimizada la ejecución, se han implementado las técnicas avanzadas mencionadas anteriormente para mejorar la diversidad máxima alcanzada por el algoritmo.

Finalmente, los resultados obtenidos se han comparado en tres niveles:

- **GA Base vs. GA Mejorado**: Se ha analizado la mejora en la diversidad máxima obtenida tras la aplicación de todas las optimizaciones.
- **Comparación entre conjuntos de datos**: Se ha evaluado si el impacto de las mejoras es similar en los distintos conjuntos de instancias (MDG-a, MDG-b y MDG-c).

- **Comparación con la literatura**: Se han comparado los mejores resultados obtenidos con los reportados en *Lozano et al., 2011*, donde se utilizó el enfoque **Iterated Greedy (IG)**[3].

Este análisis nos permitirá evaluar la efectividad de las técnicas implementadas y su competitividad frente a métodos previamente estudiados en la literatura.

4.2. Evaluación del Rendimiento Computacional

Para evaluar el impacto de las optimizaciones en el rendimiento computacional, se ha medido el tiempo de ejecución del algoritmo en tres instancias representativas. En cada caso, se ha ejecutado el algoritmo genético base (GA-B), el algoritmo genético optimizado o mejorado con Numba y vectorización (GA-O) y el algoritmo memético con optimización de soluciones (MA) para comparar los tiempos de ejecución. Siempre se ha utilizando una población de 50 individuos y 250 generaciones, se ha ejecutado 10 veces cada instancia y se ha hecho una media de lo que tarda cada algoritmo en cada instancia.

Los resultados obtenidos se muestran en la siguiente tabla:

Instancia	**Tiempo GA-B (s)**	**Tiempo GA-O (s)**	**Tiempo MA (s)**
MDG-a_1_n500_m50.txt	17.1	5.1	5.5
MDG-b_21_n2000_m200.txt	260.1	134.3	166.5
MDG-c_1_n3000_m300.txt	695.9	280.5	433.6

Cuadro 4.1

Comparación de tiempos de ejecución entre GA-B, GA-O y MA.

Para visualizar mejor la reducción del tiempo de ejecución, se presentan las siguientes gráficas:

Figura 4.1

Comparación de tiempos de ejecución entre GA-B, GA-O y MA en distintas instancias.

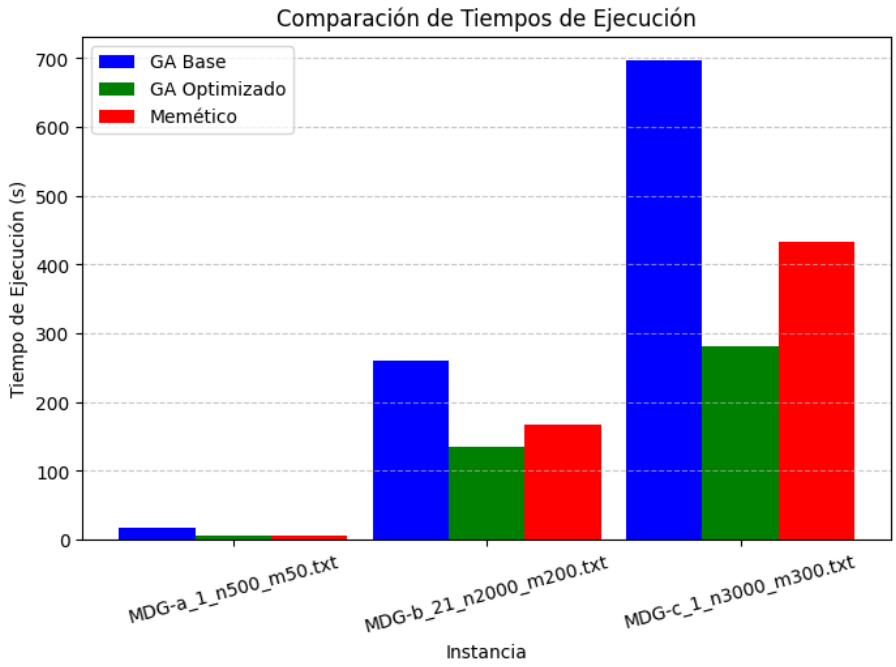
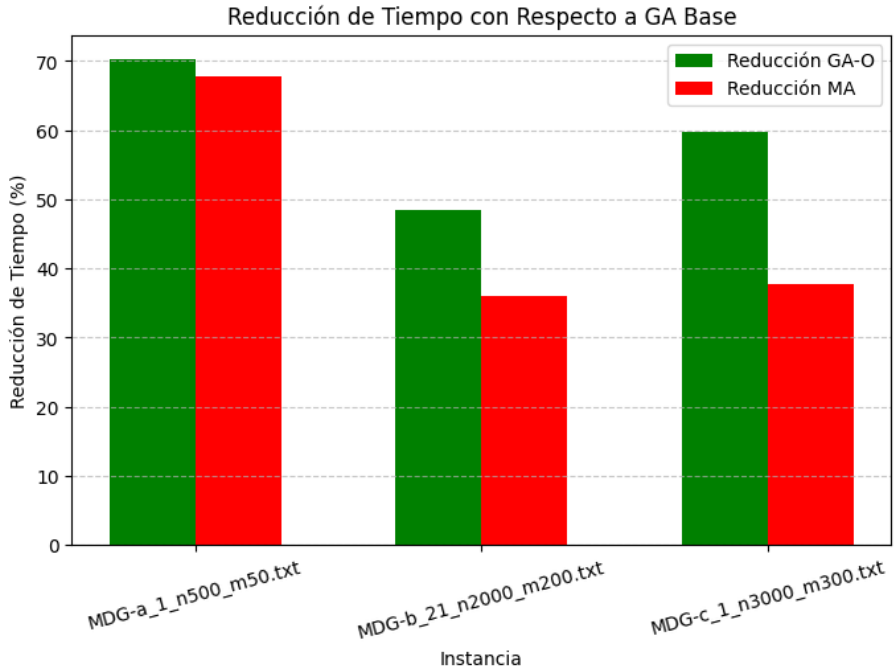


Figura 4.2

Reducción del tiempo de ejecución con respecto a GA-B.



Los resultados muestran que la optimización ha permitido reducir significativamente el tiempo de ejecución en todas las instancias, aunque, especialmente en aquellas con menor número de elementos (MDG-a). Además, la optimización de las soluciones (algoritmo MA) no supone mucha carga computacional extra con respecto a GA-O y si que el

rendimiento es mayor con respecto a GA-B.

Sin embargo, el problema plantea ejecutar 10 veces cada una de las instancias y tenemos 30 instancias, no solamente 3, por lo que los tiempos de ejecución seguirían siendo demasiado altos sobre todo para MDG-b y MDG-c. Por ello, se ha planteado la parelización utilizando la librería *joblib* de Python y optimizar así el uso de la CPU.

El código lo que hace es ejecutar de manera simultánea tantos archivos como núcleos en el procesador haya, de esta manera, se optimiza el rendimiento.

Debido a la gran carga computacional en los problemas MDG-b y MDG-c se ha reducido el número de la población inicial y el número de generaciones del algoritmo MA. La estructura de las ejecuciones es la siguiente:

- **MDG-a**: population_size=100, generations=750.
- **MDG-b:** population_size=50, generations=200.
- **MDG-c:** population_size=25, generations=50.

Los cálculos se han hecho calculando, para la columna sin paralelizar, cuanto tarda una instancia individual en ejecutarse con las condiciones anteriores y multiplicando por 10 vueltas a cada instancia y por 10 instancias en cada tipo de problema MDG-a, MDG-b y MDG-c. Y luego ejecutando el algorimto paralelizado para la columna correspondiente.

Aquí se muestra una tabla de lo que tardarían los algorimtos sin la ejecución en paralelo, y lo que tardan aplicando esta nueva técnica:

Instancia	Tiempo MA sin joblib (s)	Tiempo MA con joblib (s)	Reducción %
MDG-a	3600	264.3	92.66 %
MDG-b	13760	2540.4	81.54 %
MDG-c	6330	2606.6	58.82 %

Cuadro 4.2

Comparación de tiempos de ejecución entre algoritmos sin paralelizar y algoritmos paralelizados.

4.3. Evaluación de la Calidad de Soluciones

Además del tiempo de ejecución, es fundamental analizar la mejora en la calidad de las soluciones obtenidas. En la siguiente tabla se presentan los valores de diversidad máxima obtenidos por el GA-O y el MA en cada instancia. Al igual que antes se han hecho 10 ejecuciones de cada instancia con las siguientes condiciones:

- **MDG-a**: population_size=100, generations=750.

- **MDG-b:** population_size=50, generations=200.
- **MDG-c:** population_size=25, generations=50.

Instancia	Diversidad GA-O	Diversidad MA	Mejora (%)
MDG-a_1_n500_m50.txt	6610.84999	7551.83000	14.23 %
MDG-a_2_n500_m50.txt	6652.70000	7538.50000	13.31 %
MDG-a_3_n500_m50.txt	6616.45999	7494.49000	13.27 %
MDG-a_4_n500_m50.txt	6648.31999	7506.10999	12.90 %
MDG-a_5_n500_m50.txt	6624.63999	7504.01000	13.27 %
MDG-a_6_n500_m50.txt	6620.64000	7537.49000	13.85 %
MDG-a_7_n500_m50.txt	6603.86000	7487.00000	13.37 %
MDG-a_8_n500_m50.txt	6617.64999	7501.1500	13.35 %
MDG-a_9_n500_m50.txt	6648.89999	7564.38999	13.77 %
MDG-a_10_n500_m50.txt	6578.1400	7519.82999	14.32 %
MDG-b_21_n2000_m200.txt	10118935.68	11044884.94	9.15 %
MDG-b_22_n2000_m200.txt	10133191.59	11086845.82	9.41 %
MDG-b_23_n2000_m200.txt	10143284.16	11040744.15	8.85 %
MDG-b_24_n2000_m200.txt	10129099.26	10993878.76	8.54 %
MDG-b_25_n2000_m200.txt	10165418.40	11021439.54	8.42 %
MDG-b_26_n2000_m200.txt	10129004.65	11070528.90	9.30 %
MDG-b_27_n2000_m200.txt	10124686.61	11040704.00	9.05 %
MDG-b_28_n2000_m200.txt	10141944.40	11038220.59	8.84 %
MDG-b_29_n2000_m200.txt	10109625.71	11037724.66	9.18 %
MDG-b_30_n2000_m200.txt	10118115.35	11040484.13	9.12 %
MDG-c_1_n3000_m300.txt	22659313	24396028	7.66 %
MDG-c_2_n3000_m300.txt	22660728	24467123	7.97 %
MDG-c_8_n3000_m400.txt	40226358	42499241	5.65 %
MDG-c_9_n3000_m400.txt	40202671	42508675	5.74 %
MDG-c_10_n3000_m400.txt	40177941	42667460	6.20 %
MDG-c_13_n3000_m500.txt	62832027	66059017	5.14 %
MDG-c_14_n3000_m500.txt	62761168	66239118	5.54 %
MDG-c_15_n3000_m500.txt	62811829	65780843	4.73 %
MDG-c_19_n3000_m600.txt	90329559	94308384	4.40 %
MDG-c_20_n3000_m600.txt	90332251	93835573	3.88 %

Cuadro 4.3

Comparación de la diversidad máxima obtenida por GA-O y MA.

Los resultados muestran una mejora consistente en la diversidad máxima obtenida, confirmando que las optimizaciones han impactado positivamente en la calidad de las soluciones generadas por el algoritmo.

4.4. Comparación entre Conjuntos de Datos

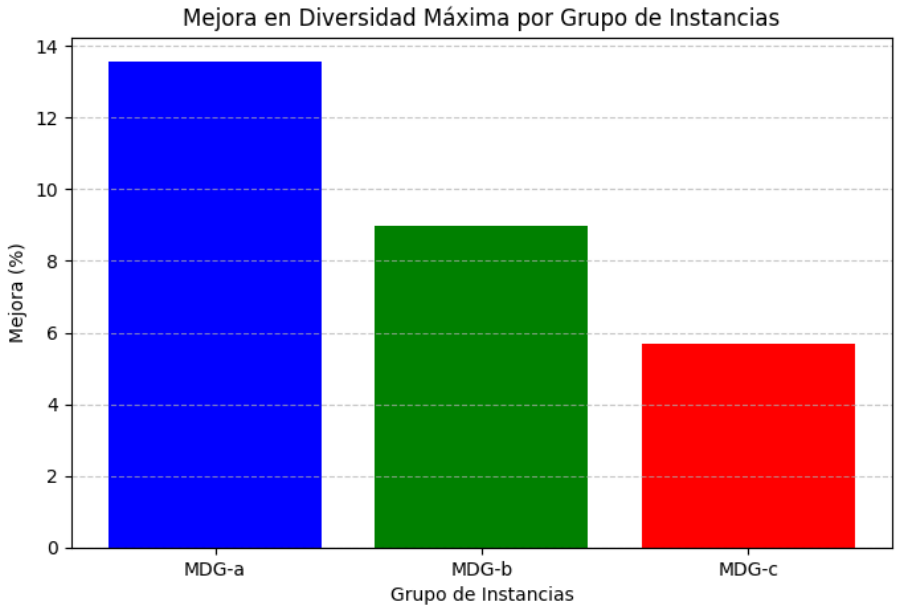
Para evaluar la consistencia de las mejoras en distintos conjuntos de datos, se ha calculado el porcentaje de mejora relativa de MA con respecto a GA-O en cada instancia. La siguiente tabla resume estos resultados:

Conjunto de datos	Mejora Relativa (%)
MDG-a	13.57 %
MDG-b	8.98 %
MDG-c	5.69 %

Cuadro 4.4
Comparación de mejora relativa en cada conjunto de datos.

Para visualizar esta comparación, se presenta la siguiente gráfica:

Figura 4.3
Comparación de la mejora relativa en cada conjunto de datos.



Los resultados indican que la mejora relativa es consistente en los diferentes conjuntos de datos, aunque el impacto puede variar en función del tamaño de la instancia, de hecho, la mejora es mayor cuanto menor es el conjunto de datos.

4.5. Comparación con la Literatura

Finalmente, se ha comparado el mejor valor de diversidad máxima obtenido por el MA con los valores reportados en *Lozano et al., 2011*, donde se empleó el enfoque **Iterated Greedy (IG)**[3]. En la siguiente tabla se presentan los resultados:

Instancia	Literatura	MA	Diferencia (%)
MDG-a_1_n500_m50.txt	7833.83252	7551.83000	3.60 %
MDG-a_2_n500_m50.txt	7771.66162	7538.50000	3.00 %
MDG-a_3_n500_m50.txt	7759.35986	7494.49000	3.41 %
MDG-a_4_n500_m50.txt	7770.2417	7506.10999	3.40 %
MDG-a_5_n500_m50.txt	7755.23096	7504.01000	3.24 %
MDG-a_6_n500_m50.txt	7773.70996	7537.49000	3.04 %
MDG-a_7_n500_m50.txt	7771.73096	7487.00000	3.66 %
MDG-a_8_n500_m50.txt	7750.88135	7501.1500	3.22 %
MDG-a_9_n500_m50.txt	7770.0708	7564.38999	2.65 %
MDG-a_10_n500_m50.txt	7780.35059	7519.82999	3.35 %
MDG-b_21_n2000_m200.txt	11299894.86	11044884.94	2.26 %
MDG-b_22_n2000_m200.txt	11286775.59	11086845.82	1.77 %
MDG-b_23_n2000_m200.txt	11299940.87	11040744.15	2.29 %
MDG-b_24_n2000_m200.txt	11290874.16	10993878.76	2.63 %
MDG-b_25_n2000_m200.txt	11296066.66	11021439.54	2.43 %
MDG-b_26_n2000_m200.txt	11292295.89	11070528.90	1.96 %
MDG-b_27_n2000_m200.txt	11305676.85	11040704.00	2.34 %
MDG-b_28_n2000_m200.txt	11279916.06	11038220.59	2.14 %
MDG-b_29_n2000_m200.txt	11297188.33	11037724.66	2.30 %
MDG-b_30_n2000_m200.txt	11296414.93	11040484.13	2.27 %
MDG-c_1_n3000_m300.txt	24884110	24396028	1.96 %
MDG-c_2_n3000_m300.txt	24905330	24467123	1.76 %
MDG-c_8_n3000_m400.txt	43437261	42499241	2.16 %
MDG-c_9_n3000_m400.txt	43437861	42508675	2.14 %
MDG-c_10_n3000_m400.txt	43476251	42667460	1.86 %
MDG-c_13_n3000_m500.txt	67014051	66059017	1.43 %
MDG-c_14_n3000_m500.txt	66979606	66239118	1.11 %
MDG-c_15_n3000_m500.txt	66992877	65780843	1.81 %
MDG-c_19_n3000_m600.txt	95633549	94308384	1.39 %
MDG-c_20_n3000_m600.txt	95643586	93835573	1.89 %

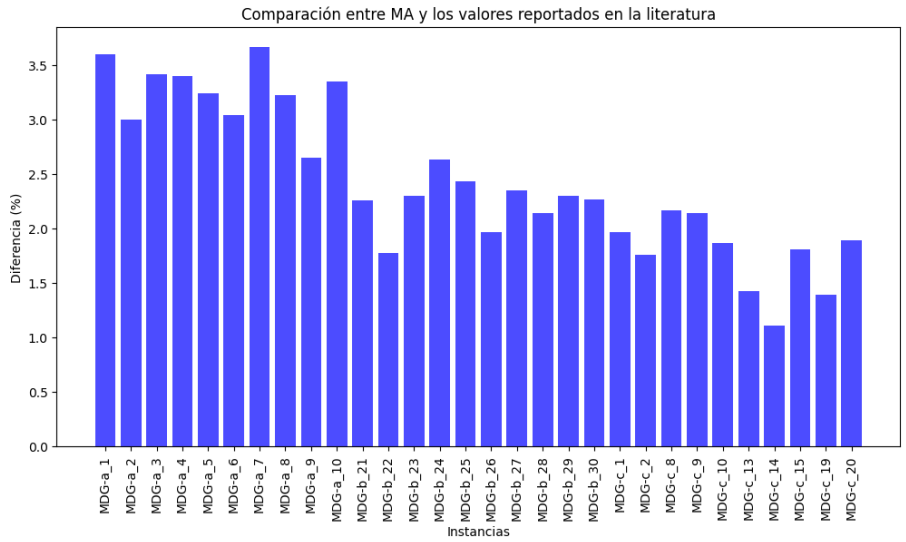
Cuadro 4.5

Comparación de resultados entre MA y la literatura.

Se observa que MA obtiene peores valores a los reportados en la literatura, aunque son soluciones competitivas, lo que sugiere que la combinación de técnicas utilizadas está resultado efectiva y que con trabajos futuros podríamos superar el estado del arte. La siguiente gráfica muestra la diferencia porcentual en cada instancia:

Figura 4.4

Comparación entre GA-M y los valores reportados en Lozano et al., 2011.



Finalmente, me gustaría mostrar una tabla final con datos de mi algoritmo final MA y se comparará brevemente con la de la literatura también. El tiempo se calcula de forma aproximada dividiendo lo que tarda el algoritmo en ejecutar el problema MDG-a, MDG-b o MDG-c y dividiendo entre 10, ya que al ejecutar de forma paralelizada no puedo llevar un registro de lo que tarda cada instancia.

Instancia	Diversidad Media	Desv	Tiempo (s)
MDG-a_1_n500_m50.txt	7449,41199	2.06	26.43
MDG-a_2_n500_m50.txt	7471.13600	0.89	26.43
MDG-a_3_n500_m50.txt	7433.31100	0.82	26.43
MDG-a_4_n500_m50.txt	7433.65799	0.97	26.43
MDG-a_5_n500_m50.txt	7421.34400	1.10	26.43
MDG-a_6_n500_m50.txt	7446.94000	1.20	26.43
MDG-a_7_n500_m50.txt	7446.46500	0.54	26.43
MDG-a_8_n500_m50.txt	7433.83400	0.90	26.43
MDG-a_9_n500_m50.txt	7474.28799	1.19	26.43
MDG-a_10_n500_m50.txt	7471.542999	0.64	26.43
MDG-b_21_n2000_m200.txt	10991871.78	0.48	254.04
MDG-b_22_n2000_m200.txt	10972059.75	1.04	254.04
MDG-b_23_n2000_m200.txt	10960141.90	0.73	254.04
MDG-b_24_n2000_m200.txt	10933043.80	0.55	254.04
MDG-b_25_n2000_m200.txt	10966513.12	0.50	254.04
MDG-b_26_n2000_m200.txt	10988511.81	0.74	254.04
MDG-b_27_n2000_m200.txt	10971838.99	0.62	254.04
MDG-b_28_n2000_m200.txt	10961322.18	0.70	254.04
MDG-b_29_n2000_m200.txt	10940835.82	0.88	254.04
MDG-b_30_n2000_m200.txt	10982425.31	0.53	254.04
MDG-c_1_n3000_m300.txt	24091654.7	1.25	260.66
MDG-c_2_n3000_m300.txt	24104018.1	1.48	260.66
MDG-c_8_n3000_m400.txt	42293711.6	0.48	260.66
MDG-c_9_n3000_m400.txt	42303160.8	0.48	260.66
MDG-c_10_n3000_m400.txt	42292206.5	0.88	260.66
MDG-c_13_n3000_m500.txt	65666133.9	0.59	260.66
MDG-c_14_n3000_m500.txt	65632156.7	0.92	260.66
MDG-c_15_n3000_m500.txt	65559236.8	0.34	260.66
MDG-c_19_n3000_m600.txt	93954351.4	0.38	260.66
MDG-c_20_n3000_m600.txt	93593260.2	0.26	260.66

Cuadro 4.6
Resultados MA.

- La desviación media recogida es de 0.77, menor que en la literatura donde el valor de la desviación media es de 3.65.
- El tiempo medio es de 180.38 segundos, mayor que en la literatura donde el valor del tiempo medio es de 133.33 segundos.

5. DISCUSIÓN GENERAL

En esta sección se analizan los resultados obtenidos tras la implementación del Algoritmo Genético (GA) y su posterior mejora con estrategias avanzadas, dando lugar a un Algoritmo Memético (MA) para la resolución del Problema de la Diversidad Máxima (PDM).

5.1. Comparación del rendimineto computacional

El primer aspecto evaluado fue la optimización del tiempo de ejecución del algoritmo. Se implementaron diversas estrategias de optimización computacional, incluyendo:

- **Vectorización y uso de Numba:** Mejora el cálculo de la diversidad.
- **Paralelización con Joblib:** Permite ejecutar las instancias en paralelo y reducir significativamente los tiempos de ejecución.

Los resultados muestran que la optimización logró una reducción del tiempo de ejecución de hasta un 92.66 % en las instancias de menor tamaño (MDG-a), mientras que en instancias más grandes (MDG-c) la reducción alcanzó un 58.82 %. Esto indica que la paralelización es especialmente beneficiosa para instancias con menor carga computacional, mientras que en problemas más grandes la sobrecarga de gestión de procesos limita la ganancia en eficiencia. En cualquier caso, siempre es necesario aplicar estas técnicas para reducir costo computacional.

5.2. Evaluación de la calidad de las soluciones

La calidad de las soluciones obtenidas por el algoritmo mejorado (MA) también fue analizada en comparación con la versión base del GA y con los valores reportados en la literatura (Lozano et al., 2011)[3].

Los resultados muestran que el MA mejora significativamente las soluciones obtenidas en todas las instancias, con incrementos de hasta 14.32 % en la diversidad máxima en el conjunto MDG-a. Sin embargo, en las instancias más grandes (MDG-c), la mejora relativa disminuye a 5.69 %. Esto sugiere que el impacto de las optimizaciones en la calidad de las soluciones es más significativo en problemas de menor escala, mientras que en instancias mayores se puede requerir ajustes adicionales en los parámetros del algoritmo para mantener la competitividad.

5.2.1. Comparación con la literatura

Uno de los objetivos clave de este trabajo fue evaluar la competitividad del MA con respecto al enfoque "Iterated Greedy" (IG) reportado en Lozano et al. (2011)[3]. Los resultados muestran que, aunque el MA no supera los valores de diversidad máxima reportados en la literatura, se encuentra en un margen competitivo. La diferencia promedio entre los valores obtenidos por el MA y los de la literatura se sitúa en torno al 2-3 % en la mayoría de las instancias, lo que indica que el enfoque propuesto tiene potencial para seguir mejorando con refinamientos adicionales.

Además, se observó que la variabilidad en los resultados del MA es menor que en la literatura, con una desviación estándar promedio de 0.77 frente a 3.65. Esto sugiere que el MA es más estable y menos sensible a variaciones en la ejecución, lo que podría llegar a ser una ventaja en aplicaciones prácticas.

6. TRABAJO FUTURO Y CONCLUSIONES

6.1. Trabajo Futuro

Para mejorar los resultados obtenidos, se proponen las siguientes líneas de investigación futuras:

1. **Migración a GPU:** Implementar el algoritmo en CUDA u otro framework de procesamiento paralelo en GPU podría reducir significativamente los tiempos de ejecución, especialmente en instancias grandes. Esto permitiría explorar más soluciones, por ejemplo, incrementando el número de generaciones, encontrando mejores resultados.

2. **Optimización de parámetros:** Se podría explorar un ajuste fino de los parámetros del MA utilizando algoritmos de búsqueda de hiperparámetros como Bayesian Optimization o Grid Search.

3. **Exploración de nuevas estrategias de cruce y mutación:** Incluir operadores más avanzados de recombinación podría aumentar la diversidad de las soluciones y mejorar la convergencia.

4. **Incorporación de aprendizaje por refuerzo:** Se podría emplear técnicas de Aprendizaje Reforzado para ajustar dinámicamente la estrategia de búsqueda durante la ejecución del algoritmo.

5. **Hibridación con otras metaheurísticas:** La combinación del MA con enfoques como GRASP o Variable Neighborhood Search podría ayudar a mejorar los resultados y hacer el algoritmo más robusto.

6.2. Conclusiones

Finalmente, en este trabajo se ha desarrollado un Algoritmo Memético para la resolución del Problema de la Diversidad Máxima, logrando buenos resultados en términos de calidad de soluciones y eficiencia computacional. Si bien los resultados obtenidos son competitivos con la literatura, queda margen para futuras mejoras, especialmente en la optimización del tiempo de ejecución y en la exploración de nuevas estrategias híbridas para mejorar la calidad de las soluciones generadas.

BIBLIOGRAFÍA

[1] Alfonso Cevallos, Friedrich Eisenbrand y Sarah Morell. «Diversity maximization in doubling metrics». En: *arXiv preprint arXiv:1809.09521* (2018). URL: https://arxiv.org/abs/1809.09521.

[2] Alfonso Cevallos, Friedrich Eisenbrand y Rico Zenklusen. «Max-sum diversity via convex programming». En: *arXiv preprint arXiv:1511.07077* (2015). URL: https://arxiv.org/abs/1511.07077.

[3] Manuel Lozano, Daniel Molina y Carlos García-Martínez. «Iterated greedy for the maximum diversity problem». En: *European Journal of Operational Research* 214.1 (2011), págs. 31-38. DOI: 10.1016/j.ejor.2011.04.018.

[4] Sepideh Mahabadi y Shyam Narayanan. «Improved Diversity Maximization Algorithms for Matching and Pseudoforest». En: *arXiv preprint arXiv:2307.04329* (2023). URL: https://arxiv.org/abs/2307.04329.

[5] Francisco Parreño, Ramón Álvarez-Valdés y Rafael Martí. «Measuring diversity: A review and an empirical analysis». En: *arXiv preprint arXiv:2401.12365* (2024). URL: https://arxiv.org/abs/2401.12365.