Optimization of AI algorithms applying high-performance computing techniques

Optimización de algoritmos de IA aplicando técnicas enfocadas en cómputo de alto rendimiento

TRABAJO DE FIN DE GRADO

DANIEL PIZARRO GALLEGO

Directores:
Alberto Núñez Covarrubias

Facultad de Informática
Universidad Complutense de Madrid

14 de mayo del 2024

UNIVERSIDAD COMPLUTENSE MADRID

AUTORIZACIÓN PARA LA DIFUSIÓN DEL TRABAJO FIN DE GRADO Y SU DEPÓSITO EN EL REPOSITORIO INSTITUCIONAL E-PRINTS COMPLUTENSE

Los abajo firmantes, alumno/s y tutor/es del Trabajo Fin de Grado (TFG) en el Grado en NOMBRE DEL GRADO REALIZADO de la Facultad de NOMBRE DE LA FACULTAD, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el Trabajo Fin de Grado (TFG) cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Periodo de embargo (opcional):

☐ 6 meses

☐ 12 meses

Título del TFG:

Curso académico: 20xx/20xx

Nombre del Alumno/s:

Tutor/es del TFG y departamento al que pertenece:

En Madrid, a 26 de mayo de 2024

Firma del alumno

Firma del director/es

Dedicado a quien utilice esta plantilla.

To whomever who uses this template.

Agradecimientos

Agradecer a...

Resumen

El trabajo que se presenta, se enfoca en la optimización de algoritmos de Inteligencia Artificial (IA). Mediante el uso de MPI (Message Passing Interface), una biblioteca estándar desarrollada para el cómputo de alto rendimiento. El objetivo principal, reducir el tiempo de ejecución de algoritmos de IA, mediante paralelización usando memoria distribuida. Es una tarea muy importante debido al alto costo temporal y de recursos que implica entrenar o ejecutar estos modelos.

Comenzamos con una explicación de los fundamentos teóricos de los algoritmos que se van a implementar, así como explicar el funcionamiento de la biblioteca MPI. Una vez puesto en contexto, se desarrollan en profundidad las estrategias implementadas para mejorar los algoritmos. Y para finalizar se ha realizado una fase de experimentación para analizar las mejoras desarrolladas a lo largo del proyecto. Esta evaluación incluye la ejecución de las mejoras en un superordenador con más de 1000 núcleos.

Palabras clave: IA, aprendizaje automático, MPI, speedup, memoria distribuida, redes neuronales, algoritmos evolutivos, clustering.

Abstract

The work presented focuses on the optimization of Artificial Intelligence (AI) algorithms. By using MPI (Message Passing Interface), a standard library developed for high-performance computing. The main objective, reduce the execution time of AI algorithms, through parallelization using distributed memory. It is a crucial task due to the high time and resource cost involved in training or running these models.

We initiate develving with an explanation of the theoretical foundations of the algorithms that will be implemented. Moreover, clarifying the functioning of the MPI library. Once put in context, the strategies employed to enhance the algorithms are thoroughly elaborated upon. And finally, an experimentation phase has been carried out to analyze the improvements developed throughout the project. This evaluation includes running the algorithms on a supercomputer with over 1000 cores.

Keywords: IA, machine learning, MPI, speedup, distributed memory, neural network, Evolutionary algorithm, clustering.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se presenta una perspectiva general del contexto en el que se ha llevado a cabo el proyecto. Además de los desafíos enfrentados durante su desarrollo para alcanzar las contribuciones mencionadas, se detallan cada uno de los propósitos perseguidos en él.

1.1. Introduccion LaTeX TODO QUITAR

The document is divided into `chapters`, `sections`, and `subsections`.
Some important references are [1-3].
To add paragraphs in the document, one line break is not enough,
two line breaks are needed.
An itemized list:

- An item.
- Another item.
- Final item.

An enumerated list:

1. First item.
2. Second item.
3. Third item.

A figure with an image is presented in Figura 1.1. Note that it floats away and latex places it where convenient.

Tables work in the same way, as seen in Tabla 1.1

```
# Prueba
print("Hola Mundo\n")
```

-_-% # @ º

Row	English	Español
1	One	Uno
2	Two	Dos

Tabla 1.1: Sample table

Figura 1.1: Sample figure

1.2. Definición y alcance del proyecto

El desarrollo de las Inteligencias Artificiales en nuestra sociedad ha sido un fenómeno de gran relevancia, además de popular, en los últimos años. Estas tecnologías han llegado para quedarse. Están mejorando nuestra calidad de vida, desde la automatización de tareas hasta la asistencia virtual, estas IAs desempeñan un papel cada vez más importante en nuestro día a día. Con el advenimiento del Internet de alta velocidad y la proliferación de datos, las empresas tecnológicas se enfrentan a la necesidad creciente de desarrollar servicios de alta calidad en un mercado muy competitivo. Se invierte mucho dinero y tiempo en mejorar y diseñar algoritmos, para implementar Inteligencias Artificiales para el acceso público

TODO...

1.3. Motivación

Actualmente hay muchas implementaciones de algoritmos de IA. Scikit learn es una biblioteca de python perfecta para probar cualquier técnica. Secuencialmente está perfeccionado y demuestra un alto desempeño computacional, pero tiene sus limitaciones.

TODO...

1.4. Objetivo

El objetivo principal de este trabajo es paralelizar varios algoritmos de IA, desarrollando varias implementaciones que reduzcan el tiempo de ejecución. Además de evaluar dichas mejoras para optimizarlas lo máximo posible.

1.5. Estructura del documento

El resto de este documento está organizado en los siguientes capítulos:

- Capítulo 2, Contextualización. Proporcionar información de cada algoritmo estudiado, para la correcta lectura del trabajo.
- Capítulo 3, Diseño e implementaciones. Comenzando con unos ejemplos básicos fuera del ámbito de la inteligencia artificial, seguido de las mejoras desarrolladas para las diferentes técnicas abordadas.
- Capítulo 4, Estudio empírico. Presenta el estudio realizado, el cual consiste en medir los tiempos de las mejoras así como las implementaciones secuenciales, para poder medir el speed-up y realizar comparaciones significativas.
- Capítulo 5, Conclusiones y trabajo a futuro.

Capítulo 2

Contextualización

En este capítulo se presenta una breve descripción de los algoritmos de Inteligencia Artificial que se van a profundizar a lo largo del proyecto. Así como los usos y características.

El objetivo de este capítulo es explicar rápidamente los algoritmos, y facilitar la lectura de los capítulos posteriores. Que desarrollen las mejoras realizadas y resultados obtenidos.

2.1. MPI

Message Passing Interface[8] (MPI) es un estándar para una biblioteca de paso de mensajes, diseñado para funcionar en una amplia variedad de arquitecturas informáticas paralelas. Permite la comunicación entre procesos, mandando y recibiendo mensajes de todo tipo. Comúnmente usado en informática de alto rendimiento[9] (HPC) y entornos informáticos paralelos, para desarrollar aplicaciones paralelas escalables y eficientes.

TODO...

2.2. Aprendizaje por Refuerzo

Reinforcement Learning (RL), en español Aprendizaje por Refuerzo. Es un tipo de aprendizaje automático donde el agente aprende en base a las decisiones tomadas al interactuar con el entorno. El agente aprende a llegar a una meta o maximizar un cúmulo de recompensas obtenidas al realizar un determinado número de acciones consecutivas, y observar las posibles recompensas al realizar cada acción en los estados del entorno.

TODO...

2.3. Aprendizaje No-Supervisado

Los métodos no supervisados (unsupervised methods) son algoritmos de aprendizaje automático que basan su proceso en un entrenamiento con datos sin etiquetar. Es decir, a priori no se conoce ningún valor objetivo, ya sea categórico o numérico.

TODO...

2.4. Aprendizaje Supervisado

Al contrario que el apartado anterior, este tipo de aprendizaje automático, es entrenado con un dataset categorizado con su salida correcta. El algoritmo aprende de este conjunto, para hacer predicciones sobre unos datos desconocidos.

TODO...

2.5. Algoritmos Evolutiva

La programación evolutiva es una técnica de optimización inspirada en la teoría de la evolución biológica. Se basa en el concepto de selección natural y evolución de las poblaciones para encontrar soluciones a problemas complejos.

TODO...

Capítulo 3

Diseño e Implementaciones

El objetivo de este proyecto es mejorar los algoritmos de IA mencionados anteriormente. En este capítulo se presentan los diseños e implementaciones desarrolladas a lo largo del desarrollo del mismo.

3.1. Ejemplos básicos MPI

3.2. Algoritmos de Clustering

3.2.1. Jerárquico Aglomerativo

Este algoritmo usa una matriz para calcular las agrupaciones. Como es una matriz simétrica, podemos reducir la complejidad espacial usando solo el triángulo superior.

3.2.2. K-Medias

De las técnicas de clustering de aprendizaje no supervisado, en la cual tenemos una población inicial de individuos sin clasificar, y un valor K sujeto a una asignación flexible según nuestros criterios. Al contrario al algoritmo anterior no se usa una matriz, y solo se usa distancia por centroides.

3.2.3. K-Vecinos más cercanos (KNN)

Tenemos un valor K asignado de manera arbitraria como en el algoritmo de K-Medias. Esta técnica de clustering pertenece al aprendizaje supervisado, tenemos una población de individuos categorizados con las etiquetas de asignacion de cluster. Una población a predecir.

3.3. Aprendizaje por refuerzo

El algoritmo de Q-Learing actualiza iterativamente las estimaciones de calidad de las acciones permitidas en el entorno de desarrollo. Estos valores se almacenan en la Q-Table, representado como una matriz en la que cada fila es un estado, y las columnas son las acciones disponibles.

3.4. Algoritmos Evolutivos

Los algoritmos evolutivos son sencillos de paralelizar, debido a que son procesos que se repiten muchas veces, y se ejecutan en muchos individuos.

3.5. Redes Neuronales

Esta poderosa herramienta de aprendizaje supervisado, está diseñada para reconocer patrones complejos y realizar diversas tareas. Aprende con un proceso iterativo de entrenamiento, ajustando las conexiones entre neuronas. Este proceso secuencial es complejo de paralelizar. Al finalizar una predicción el modelo se tiene que actualizar propagando hacia atrás.

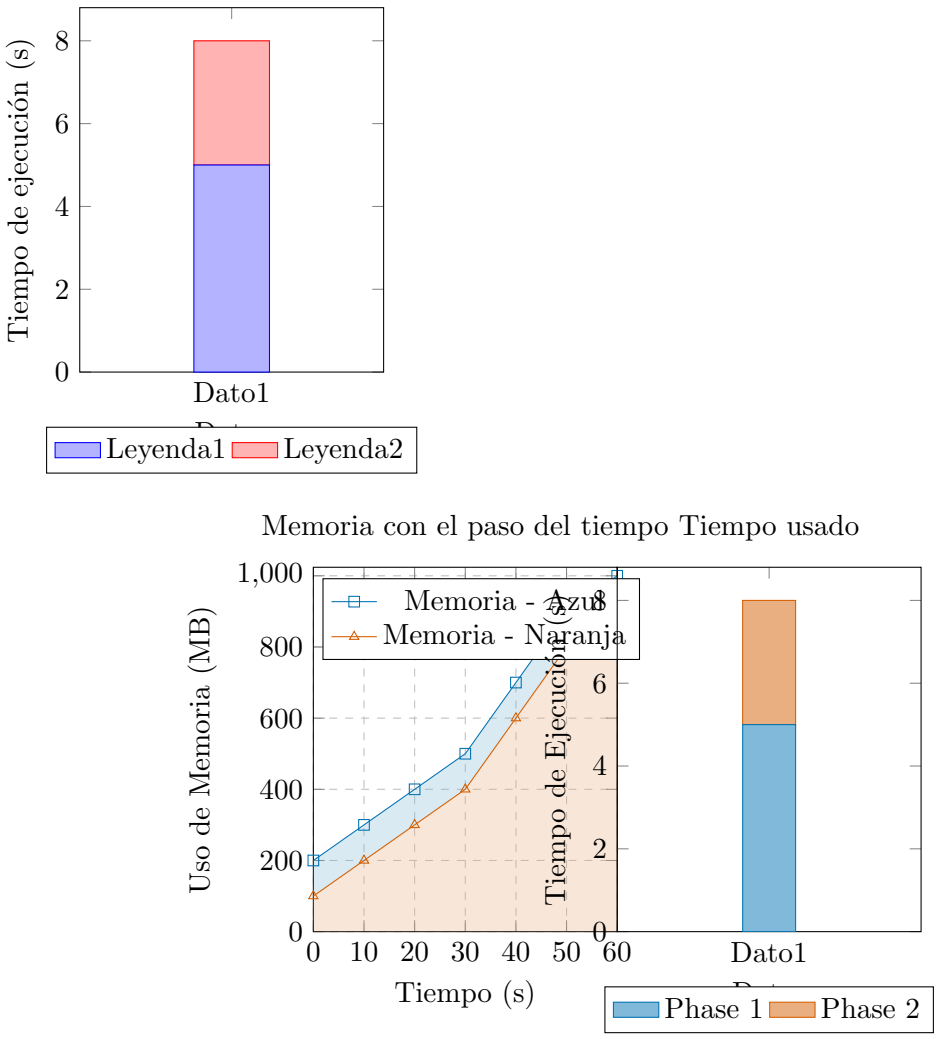
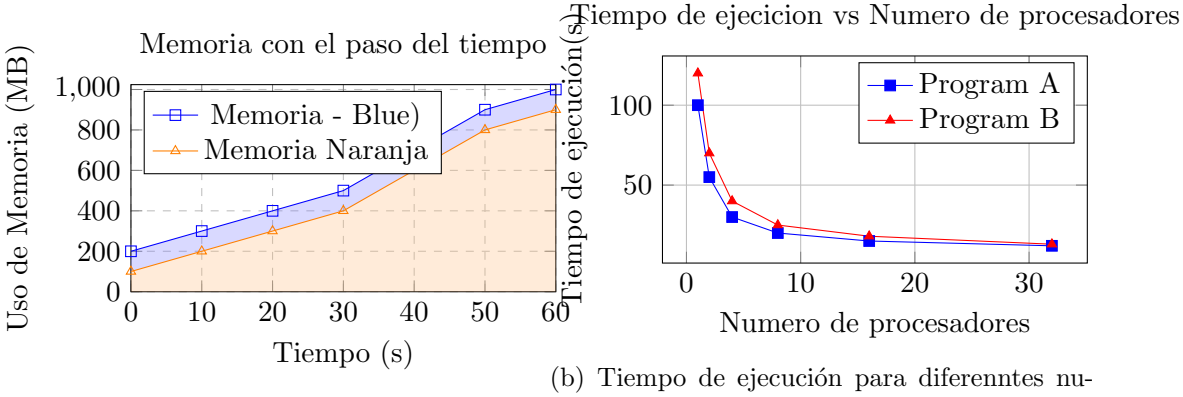


Figura 3.1: Comparacion de la memoria y tiempo de ejecución.



(a) Memoria con el paso del tiempo.

(b) Tiempo de ejecución para diferenntes numeros de procesadores

Figura 3.2: Dos gráficos al lado

```
n   # int. Tam. de la poblacion
d   # int. Numero de variables/dimensiones
K   # int. Numero de Centroides
poblacion  # float[n]. Individuos
centroides  # float[K]. Centroides
asignacion  # float[n]. Asignacion para cada individuo

indsCluster  # float[K]. Numero de individuos para cada centroide
centroidesNuevos # float[K]. Centroides calculados en la iteracion
```

Algoritmo 1: Euclid's algorithm

Data: Two nonnegative integers a and b
Result: $\gcd(a, b)$
while $b \neq 0$ **do**
    $r \leftarrow a$ mód b;
    $a \leftarrow b$;
    $b \leftarrow r$;
return a;

Capítulo 4

Estudio empírico

Después de implementar y diseñar los algoritmos y mejoras utilizando MPI, llevamos a cabo un análisis exhaustivo para evaluar los tiempos de ejecución, realizar pruebas, contrastar resultados y extraer conclusiones.

Primero se ejecutan distintas pruebas en un ordenador normal, para luego ejecutar las mejores implementaciones en un cluster con varios ordenadores y muchos procesos.

Ordenador básico: 8 núcleos, 16 procesos lógicos y 32gb de ram

Cluster:

Ordenaciones

Pruebas

Arrays de enteros, siempre en el caso peor, es decir, ordenados de forma **decreciente**, por lo que tiene que ejecutar el mayor número de comparaciones para ordenarlo de forma creciente.
Las pruebas se ejecutan sobre el mismo array de enteros, y se van aumentando el tamaño para medir los tiempos de ejecución. En cada prueba se añaden x elementos más a ordenar.

El incremento del tamaño viene dado de la siguiente forma:
- [20-1.000) → 20 elementos.
- [1.000-10.000) → 250 elementos.
- [10.000-100.000) → 1.000 elementos.

Para MergeSort, al ser un algoritmo mas eficiente, no se añade de mil en mil al llegar a 10.000 elementos, se mantiene el incremento de 250 elementos.

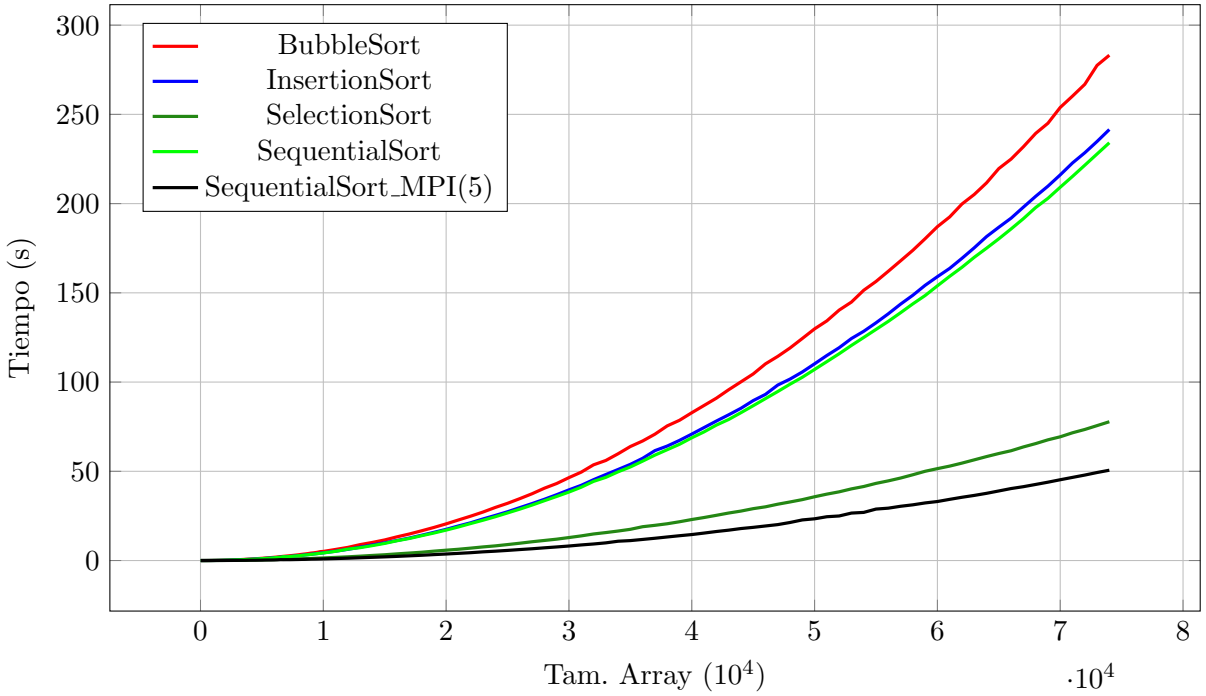


Figura 4.1: Tiempo de ejecución de los algoritmos de ordenación cuadráticos

La implementación aplicando MPI sobre SequentialSort tiene un speed-up proporcional al número de procesos ejecutándose al llegar a un cierto número de elementos a ordenar. Al paralelizar el trabajo de SequentialSort, se pueden obtener mejores resultados que SelectionSort al usar cinco o más procesos workers.

De las ordenaciones básicas con coste cuadrático (O(N2)) comentadas anteriormente, SelectionSort es la que mejores resultados obtiene, y BubbleSort la peor, siendo aproximadamente 3.5 veces más lenta con 70.000 elementos. La ordenación diseñada SequentialSort es incluso más rápida que dos de las más famosas, debido a la simpleza de las operaciones aplicadas en la ordenación. Esta ordenación hace N2 comparaciones, pero al no modificar elementos del array sigue siendo más rápida que las otras.

```
def selection_sort(a):
n = len(a)
minE = 0
pos = 0

for i in range(n-1):
minE = a[i]
pos = i
for j in range(i+1, n):
if minE > a[j]:
minE = a[j]
pos = j
b[cont]=val
tmp = a[i]
a[i] = a[pos]
a[pos] = tmp
```

```
def sequential_sort(a):
  INF=sys.maxsize
  n=len(a)
b=[(INF) for i in range(n)]

  for i in range(n):
  cont=0
  val=a[i]
  for i in range(n):
if a[i]<val: cont+=1
  while b[cont]!=INF:
cont+=1
```

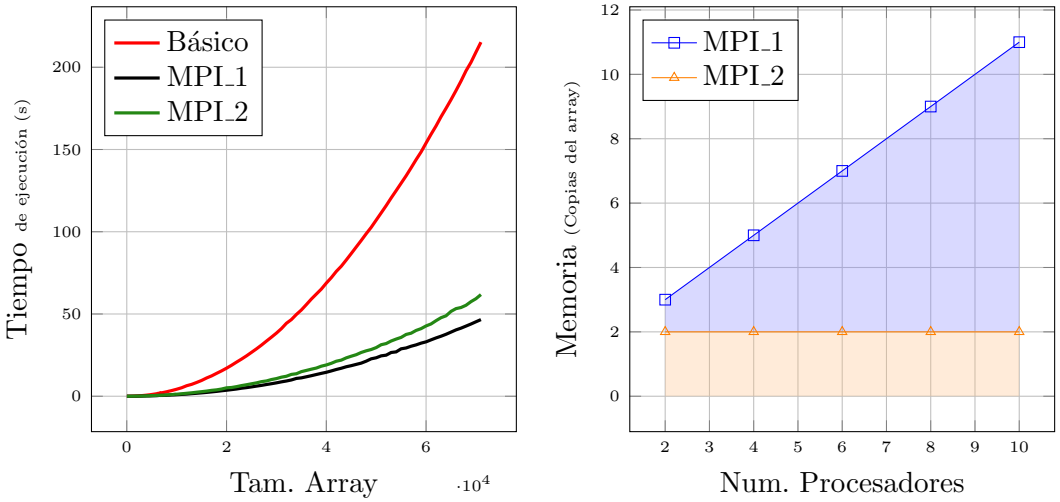


Figura 4.2: Mejoras MPI SequentialSort + Memoria

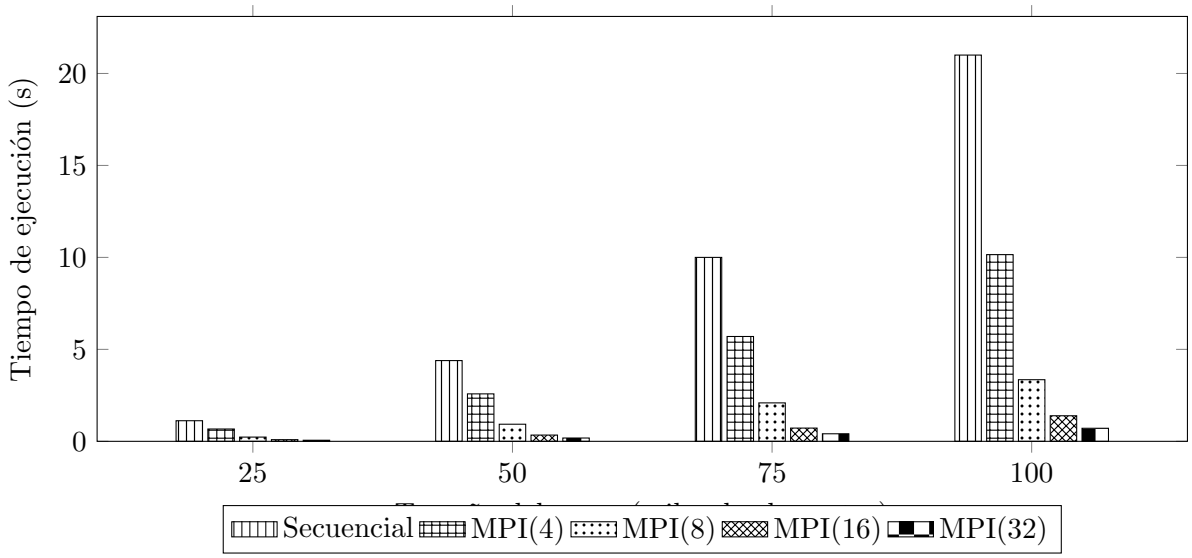


Figura 4.3: Tiempo de ejecución MergeSort + MPI

Algoritmos de Agrupación

Pruebas

Las poblaciones que se usan en cada algoritmo se han generado previamente de manera aleatoria, delimitando un intervalo [-10, 10] para todas las dimensiones disponibles. Para crear los gráficos se ejecutó muchas veces con diferentes tamaños para guardar los tiempos de ejecución con poblaciones de diferentes tamaños. Se añaden x elementos a las poblaciones.

Para las implementaciones de K-Medias y Jerárquico Aglomerativo se aplican el mismo incremento que en las ordenaciones: El incremento del tamaño viene dado de la siguiente forma:
- [20-1.000) → 20 elementos.
- [1.000-10.000) → 250 elementos.
- [10.000-100.000) → 1.000 elementos.

Para K-Vecinos más cercanos (KNN), al ser un algoritmo lineal, se guarda el tiempo de ejecución cada veinte nuevos individuos categorizados.

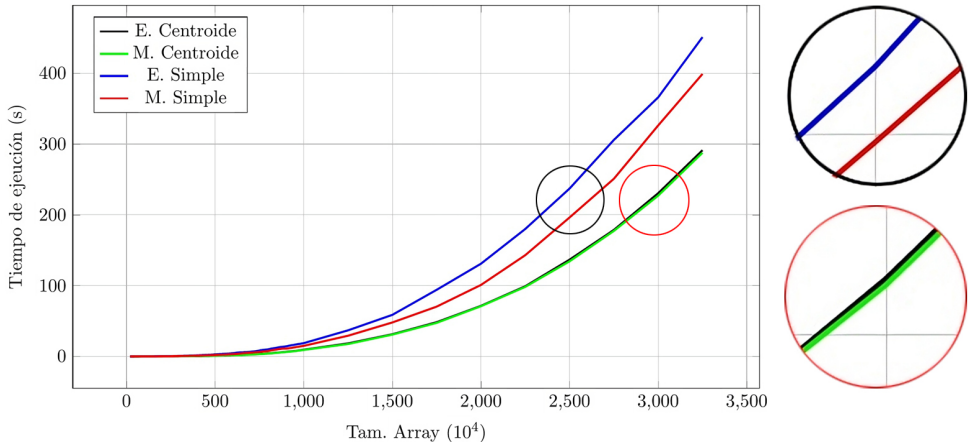
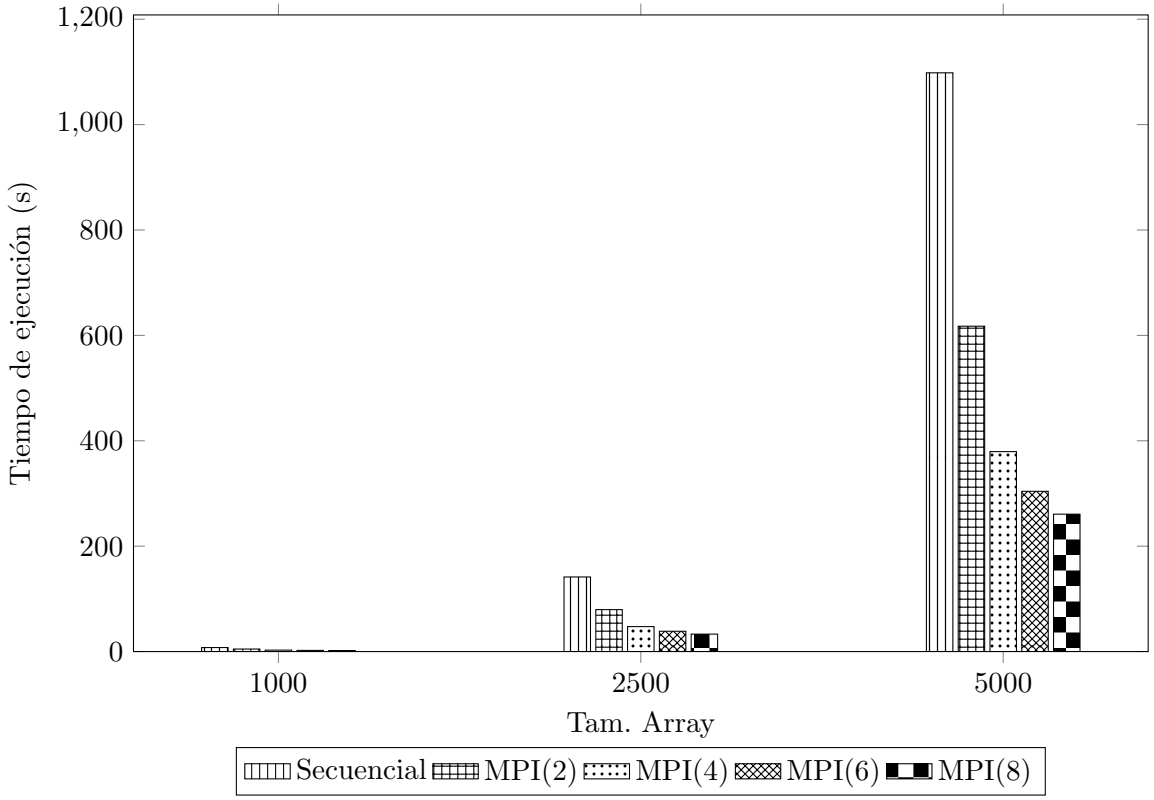


Figura 4.4: Tiempo de ejecución del algoritmo básico Jerarquíco Aglomerativo



K-MEDIAS

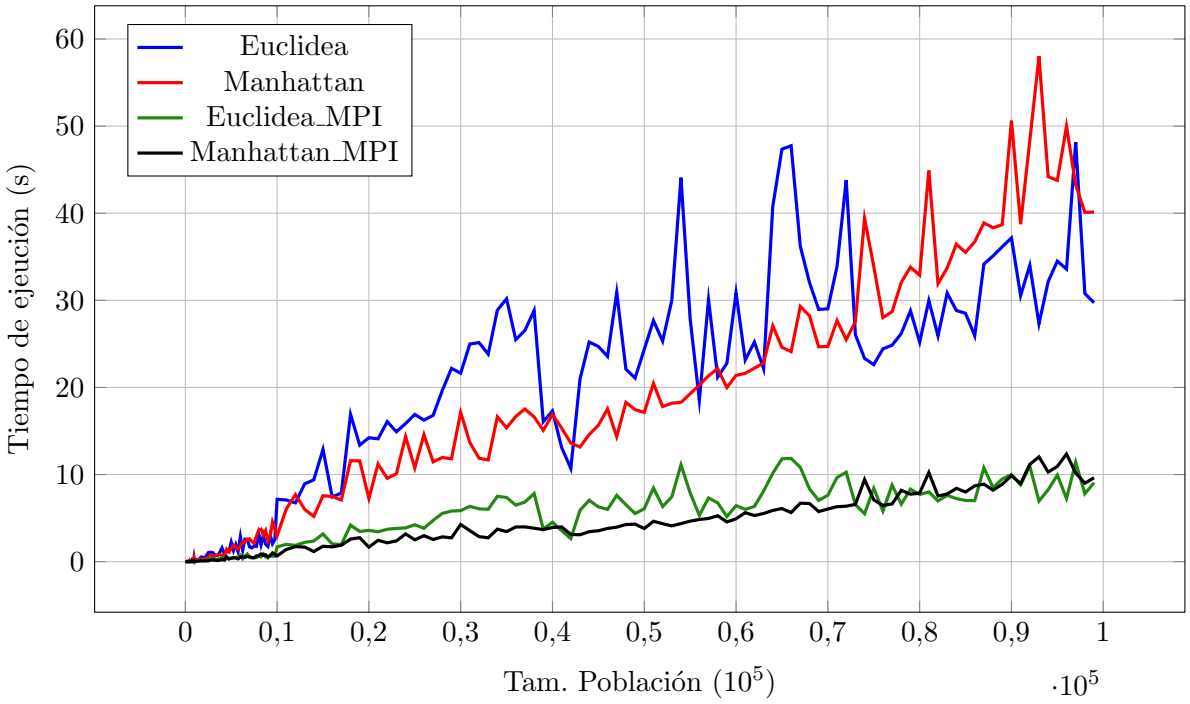


Figura 4.5: Tiempo de ejecución para KMedias

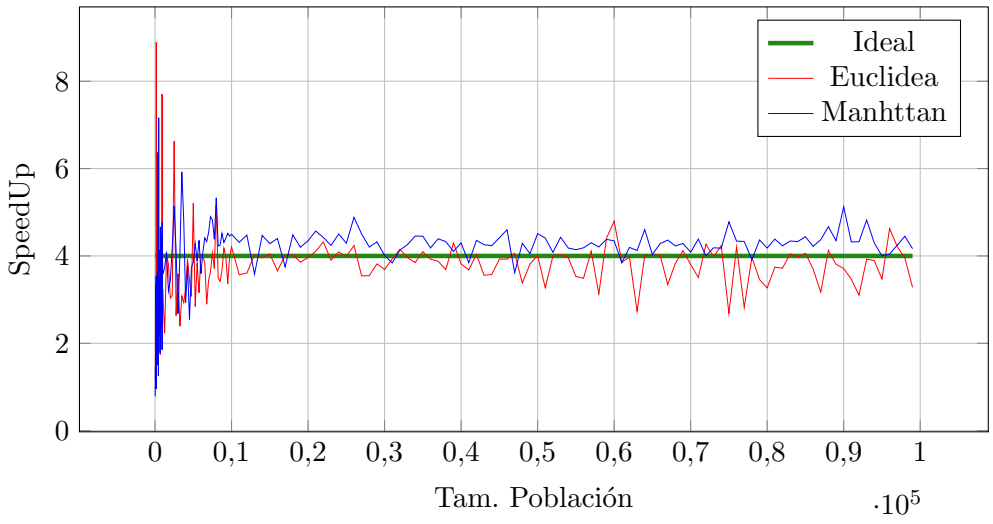


Figura 4.6: SpeedUp - KMedias

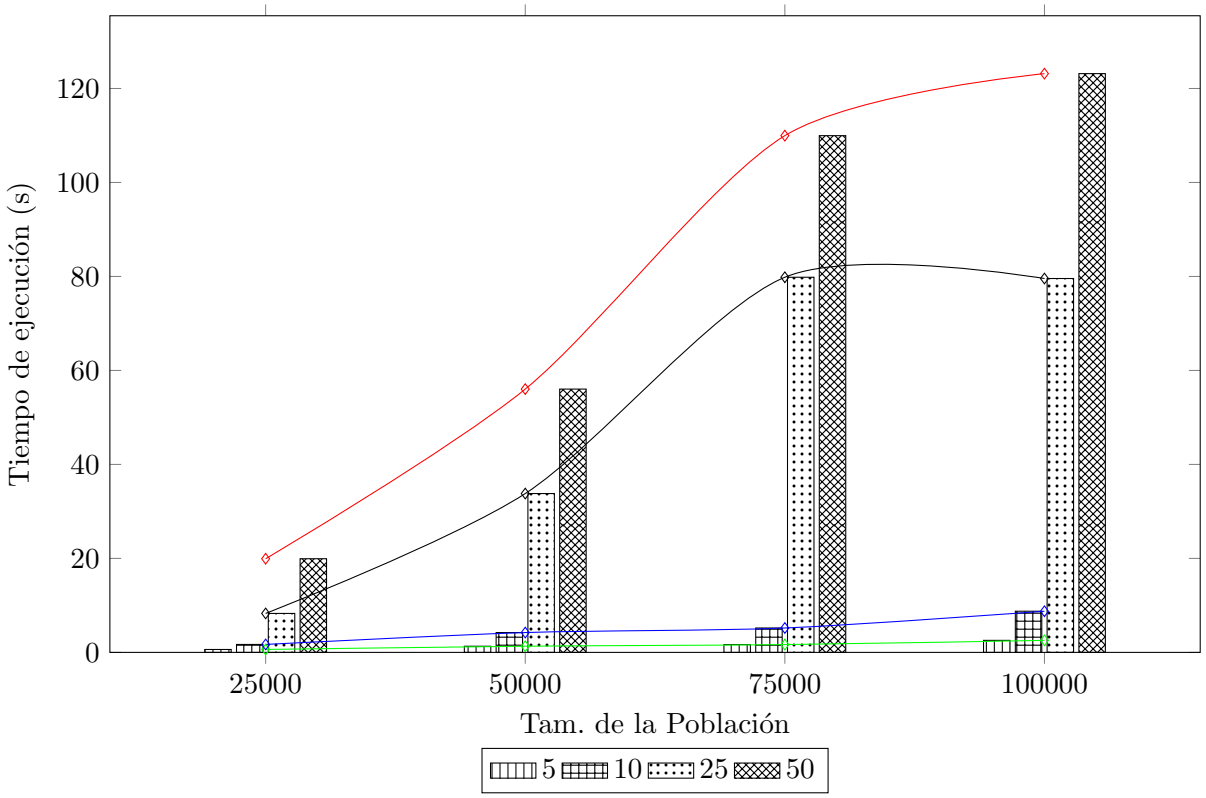


Figura 4.7: KMedias variando K

KNN

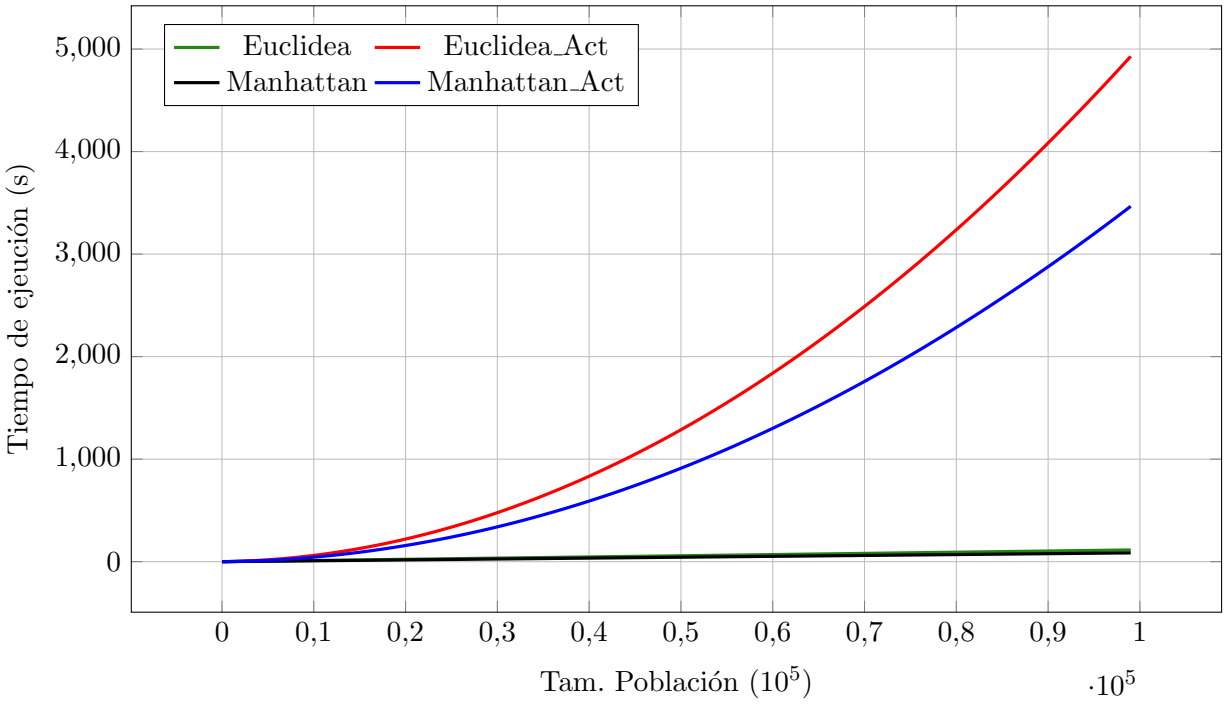


Figura 4.8: Tiempo de ejecución para KNN

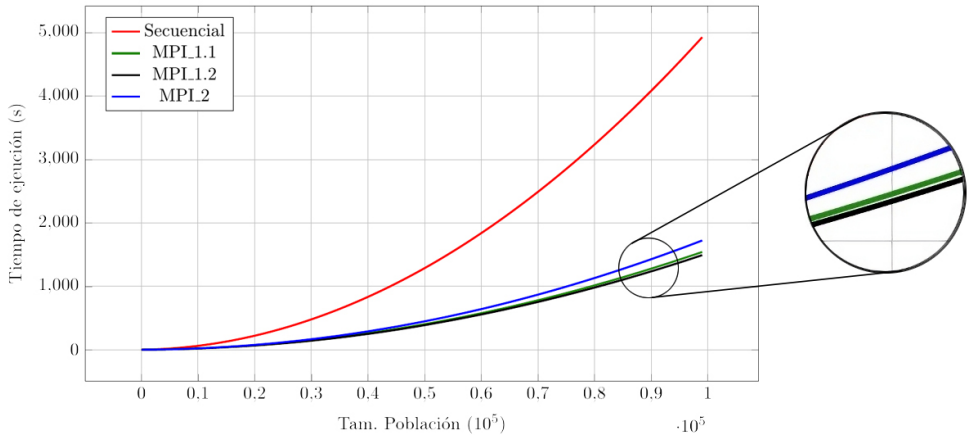


Figura 4.9: Tiempo de ejecución del algoritmo básico Jerarquíco Aglomerativo

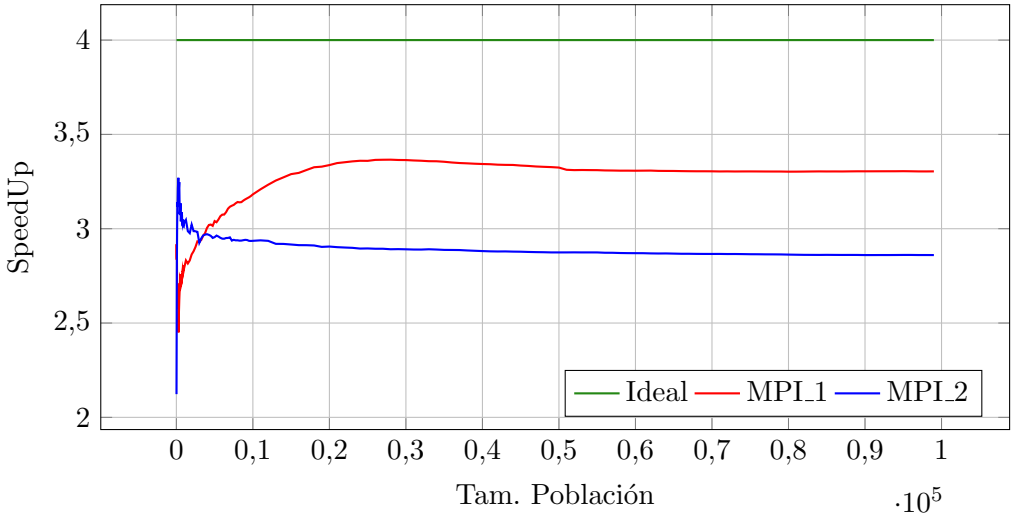


Figura 4.10: SpeedUp - KNN

RL

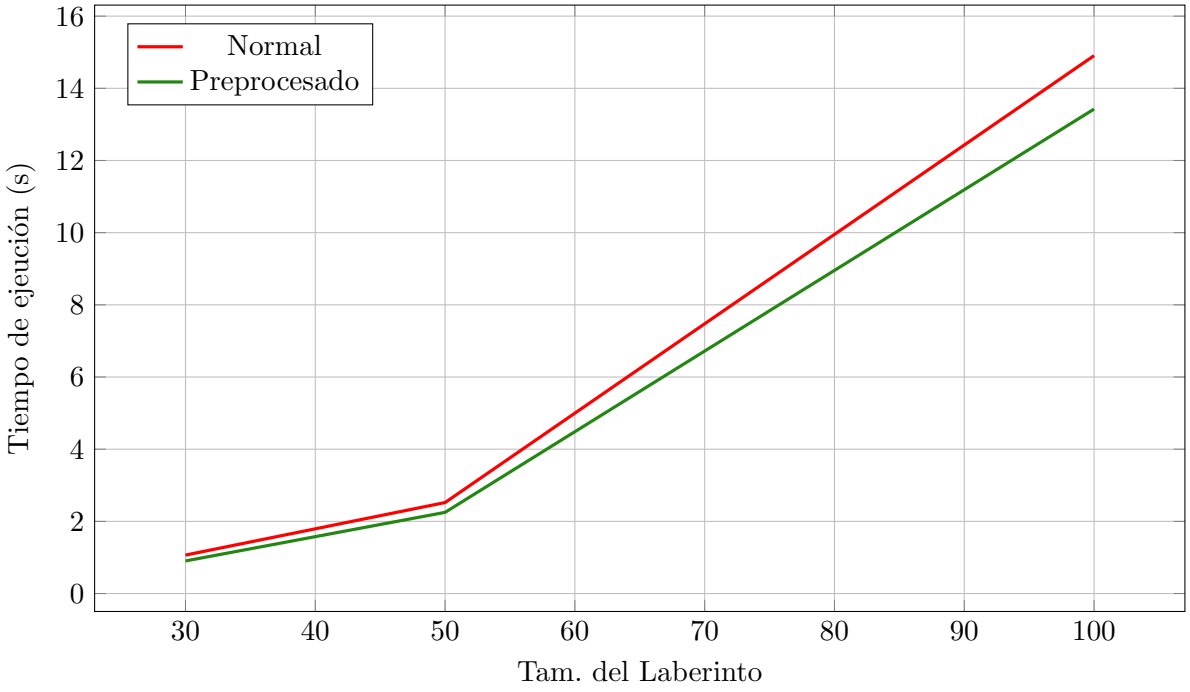


Figura 4.11: Tiempo de ejecución para RL

PEV

Pruebas

Para el algoritmo se ha aplicado un elitismo de 5 % conservando los mejores individuos de cada generación. Para el problema de árboles se aplica un método de control de bloating, para intentar reducir la alutra de los individuos. Para mejorar la aptitud de los individuos se aplica un desplazamiento, cuya finalidad es todos los individuos tengan valores positivos. Además de aplicar un escalado lineal, controlando la diversidad de las aptitudes.

El método de evaluación depende del tipo de individuo.
- Si es binario se calcula su valor real y se aplica una función matemática.
- Si es real, el problema del aeropuerto.
- Si es árbol, el problema del cortacésped.

Las pruebas realizadas para todas las gráficas se han ejecutado con las siguientes características:

Tam. Población = 100
Núm. Generaciones = {25,50,100,250,500,1000,2000} **(Eje X)**
Met. Selección: Torneo Determinístico, con un valor k=5.

- Individuo Binario:
Met. Cruce (p=0.6): Básica
Met. Mutación (p=0.05): Básica
P(x)=precision: {P2: 30 bits, P10: 76 bits}

- Individuo Real:
Met. Cruce (p=0.6): PMX
Met. Mutación (p=0.3): Inserción
AER(x)=aeropuerto: {AER1: 10 vuelos, 3 pistas, AER1: 25 vuelos, 5 pistas, AER3: 100 vuelos, 10 pistas}

- Individuo Binario:
Met. Cruce (p=0.6): Intercambio
Met. Mutación (p=0.3): Terminal
M(x)X(y)=matriz: {M8X8: 8 filas, 8 columnas y 100 ticks; M100X100: 100 filas, 100 columnas y 10000 ticks}

Datos	Funciones	Init(1)	Evaluación(1)	Selección(1)	Cruce(2)	Mutación(1)
Precision: 2	Binario	2.56e-05	4.4e-06	8.56e-06	1.36e-05	1.53e-05
Precision: 10	Binario	3.33e-05	5.44e-06	8.9e-06	1.71e-05	1.88e-05
aviones: 12 pistas: 3	Aeropuerto 1	7.04e-06	2.55e-05	4.12e-06	1.48e-05	2.764e-06
aviones: 25 pistas: 5	Aeropuerto 2	1.36e-05	6.55e-05	4.62e-06	2.4e-05	3.43e-06
aviones: 100 pistas: 10	Aeropuerto 3	3.97e-05	4.3e-04	8.05e-06	4.18e-05	1.04e-05
M10x10 ticks: 150	Árbol	6.12e-05	6.47e-05	7.92e-05	2.33e-05	3.47e-07
M25x25 ticks: 400	Árbol	6.16e-05	1.65e-04	7.88e-05	2.32e-05	3.7e-07
M100x100 ticks: 800	Árbol	6.41e-05	3.66e-04	8.07e-05	2.09e-05	3.23e-07

Tabla 4.1: PEV - Tiempos de cada método

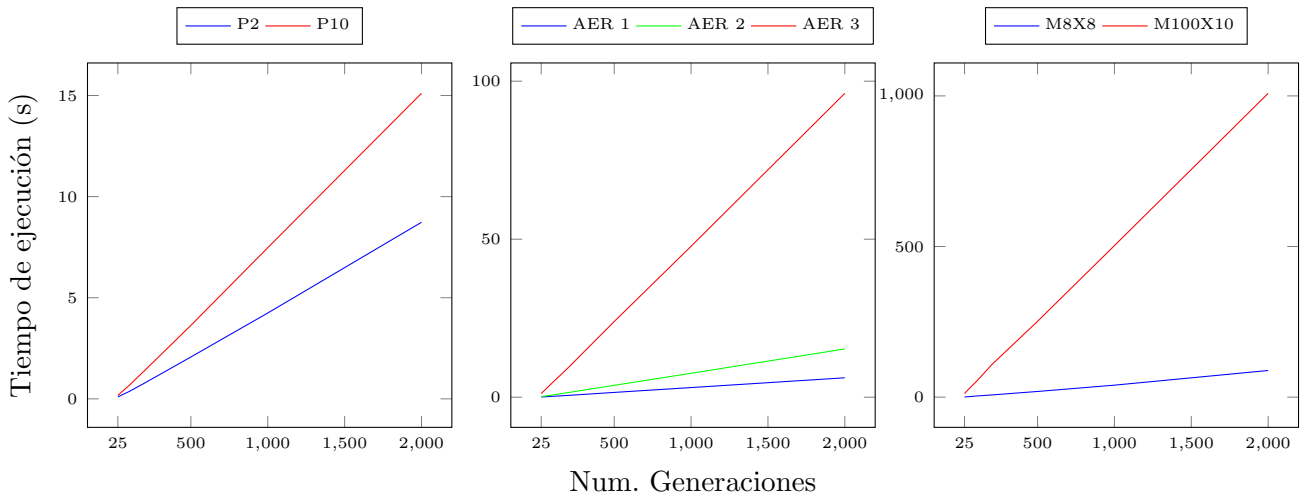


Figura 4.12: PEV Secuencial

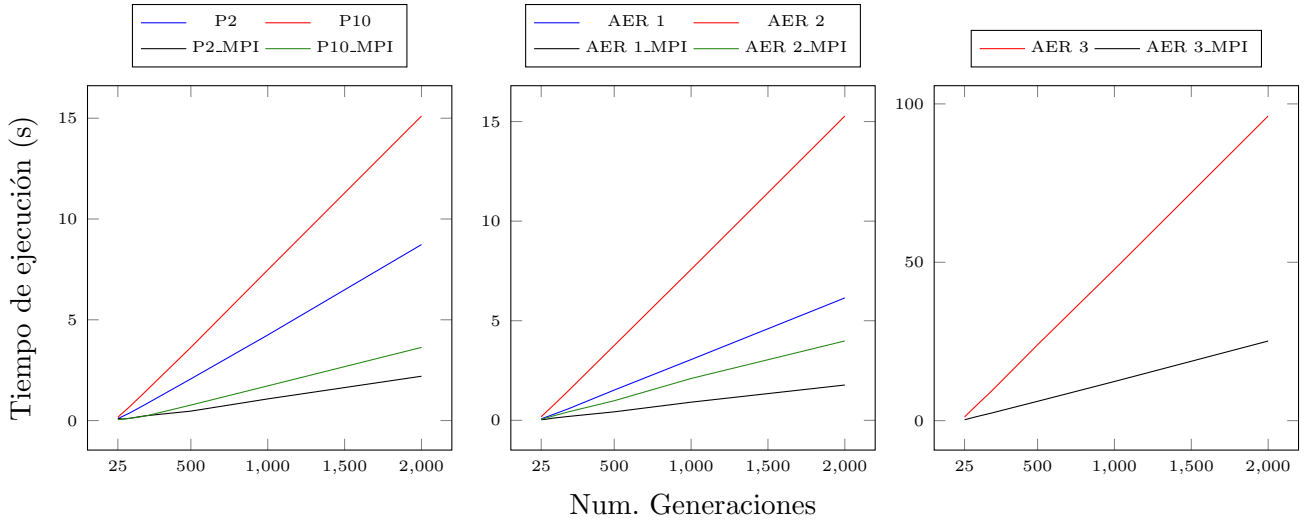


Figura 4.13: MPI - Modelo de Islas

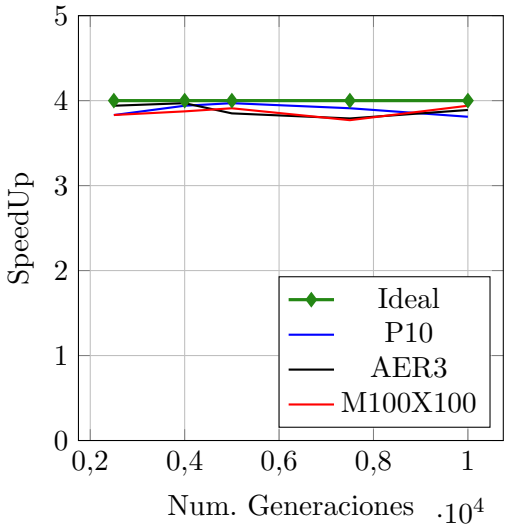


Figura 4.14: SpeedUp - Modelo en Islas

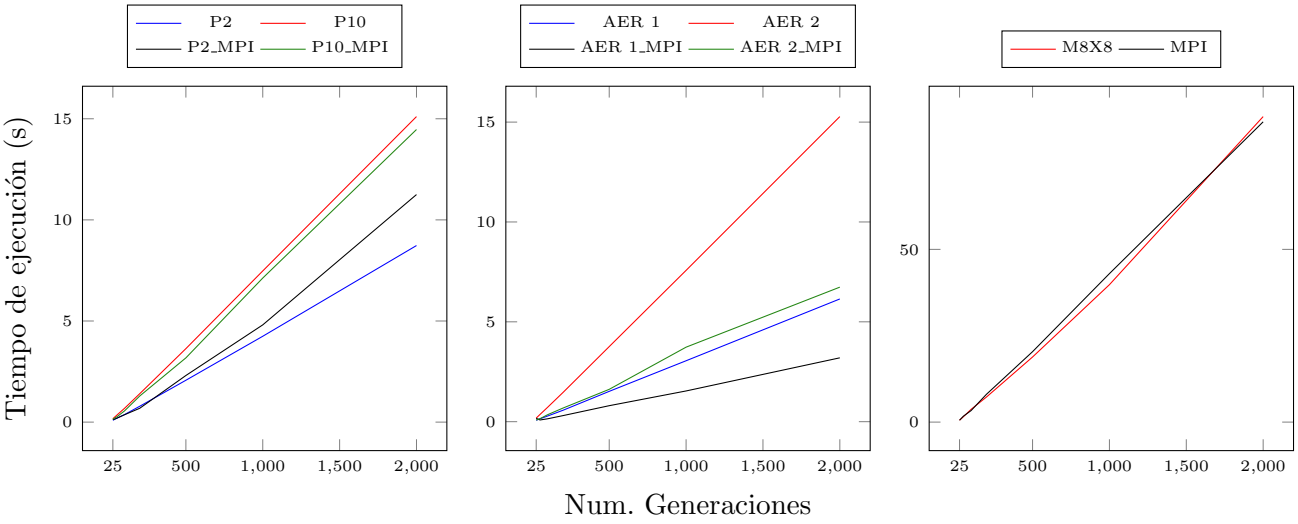


Figura 4.15: MPI1.1 - Dividir Poblacion

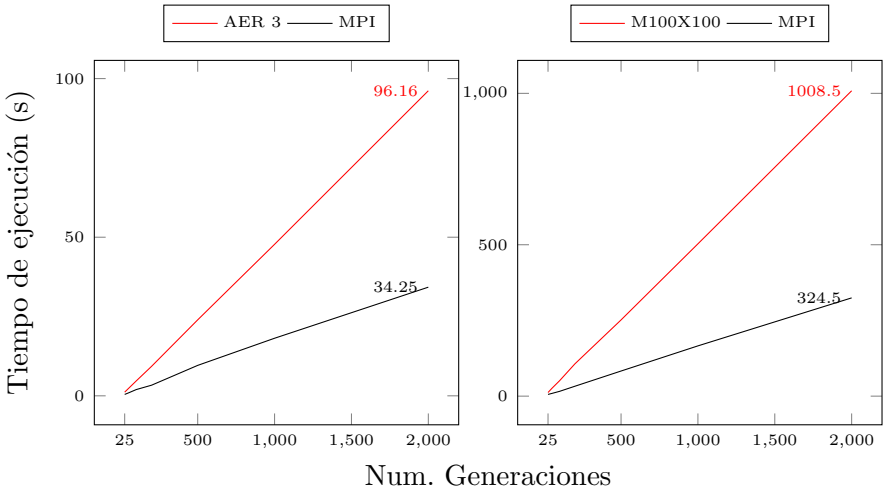


Figura 4.16: MPI1.2 - Dividir Poblacion

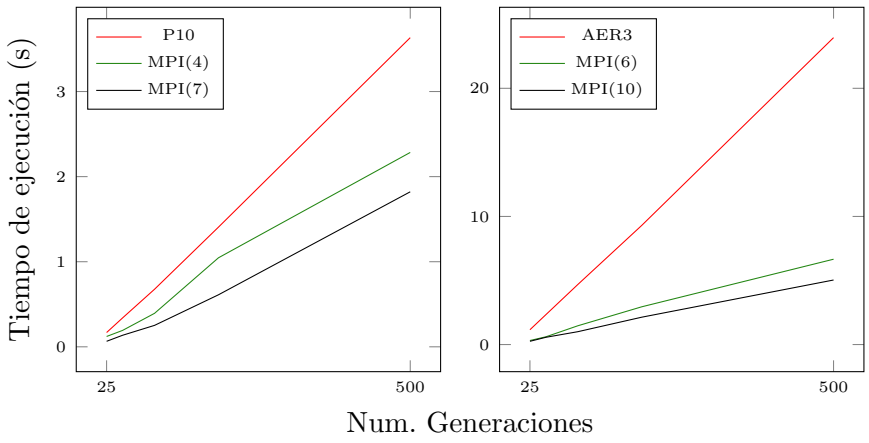


Figura 4.17: MPI3 - PipeLine

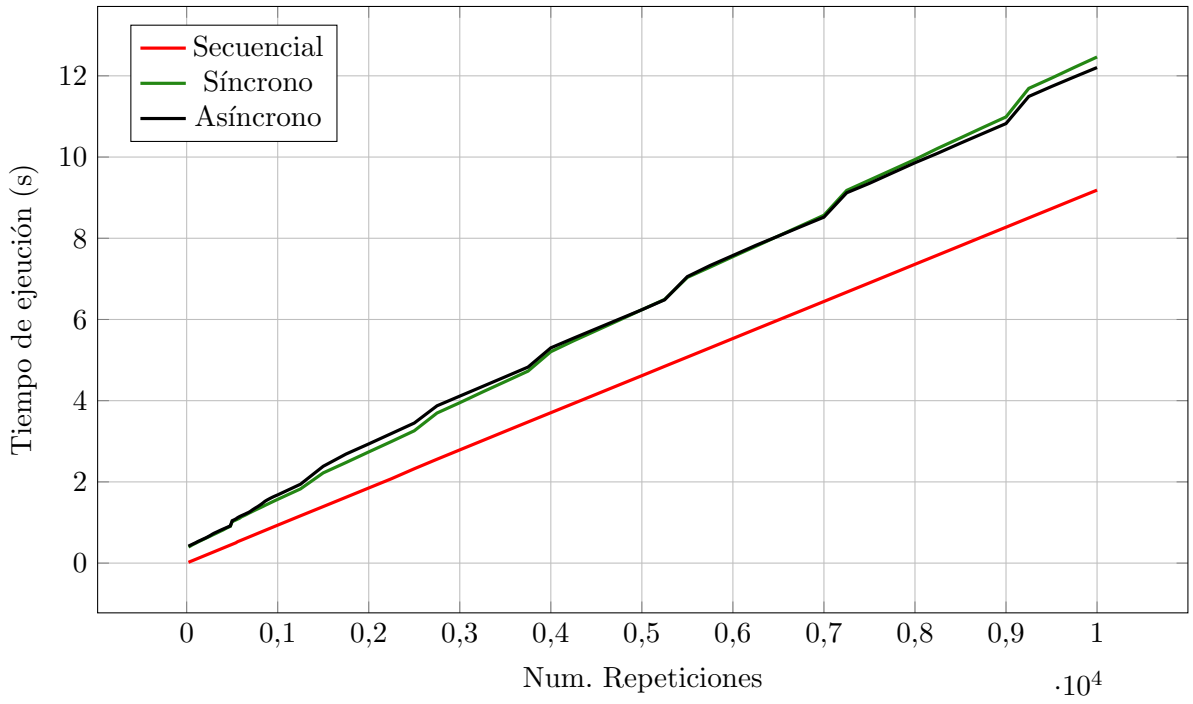


Figura 4.18: MPI1 - Red Neuronal

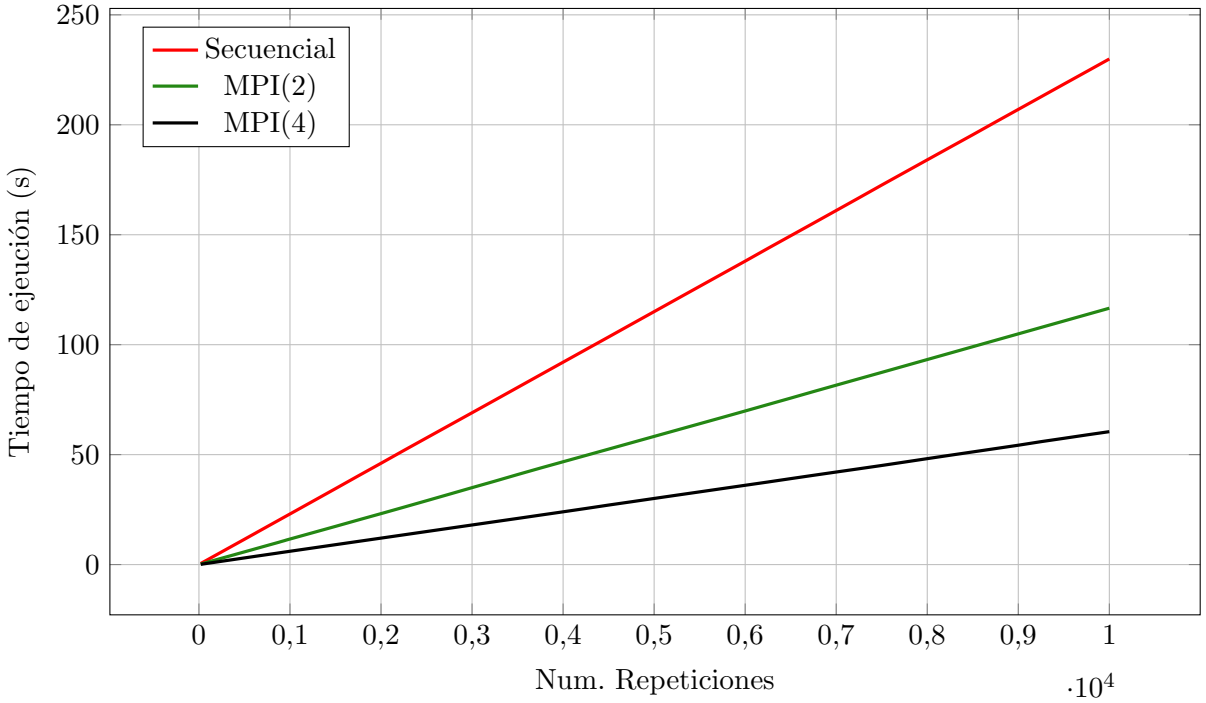


Figura 4.19: MPI2 - Red Neuronal

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo se han desarrollado varias mejoras en distintos algoritmos de IA, a través de la biblioteca estándar de paso de mensajes MPI. Los desafíos encontrados durante su desarrollo resultaron ser más complejos de lo que se había previsto inicialmente. Los problemas de configuración de la biblioteca MPI en windows, y adaptar la gestión de bibliotecas de Python usando Anaconda, fueron unos problemas completamente imprevistos. El desconocimiento general de MPI, se debe a la ausencia de asignaturas específicas de programación distribuidas en el grado de Ingeniería Informática, únicamente ofreciendo fundamentos teóricos, sin profundizar en la práctica. La escasa implementación práctica en las asignaturas de IA, en el itinerario Tecnología Específica de Computación del tercer curso ha derivado en tener que invertir más tiempo en investigar e implementar los algoritmos, proceso que he encontrado satisfactorio. La teoría vista en clase fue muy útil para el desarrollo del trabajo, pero usar exclusivamente la librería sklearn, de scikit-learn, provocó un desconocimiento de código para implementar estos algoritmos.

Una vez finalizadas las implementaciones, se ha llevado a cabo una fase de experimentación. Consistiendo en analizar los tiempos de ejecución, variando todos los parámetros disponibles, además de variar los conjuntos de poblaciones para cada tipo de algoritmo. Las ejecuciones de las pruebas requieren un coste computacional alto, además de mucho tiempo para finalizar. Para pruebas pequeñas no se consiguen apreciar reducciones significativas. Normalmente se pierde tiempo al paralelizar. Pero conforme aumentan los parámetros introducidos, mejora notablemente el speedup de las mejoras.

Cabe destacar .... TODO

Como trabajo a futuro se propone investigar otros algoritmos de las técnicas desarrolladas. Además de investigar y mejorar otras técnicas de IA, como puede ser el procesamiento del lenguaje natural.

Bibliografía

[1] Albert Einstein. «Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]». En: *Annalen der Physik* 322.10 (1905), págs. 891-921.

[2] Michel Goossens, Frank Mittelbach y Alexander Samarin. *The LaTeX Companion*. Reading, Massachusetts: Addison-Wesley, 1993.

[3] Donald Knuth. *Knuth: Computers and Typesetting*. URL: `http://www-cs-faculty.stanford.edu/\~{}uno/abcde.html`.