Paralelización de operadores de cruce de un algoritmo genético

Alcívar Moreira Bryan Alexander
Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, ESPAM MFL
bryan.alcivar@espam.edu.ec
Calceta, Ecuador.

Franco Salvatierra José Javier
Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, ESPAM MFL
jose.franco@espam.edu.ec
Calceta, Ecuador.

Zambrano Solórzano Ligia Elena
Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, ESPAM MFL
Lzambrano@espam.edu.ec
Calceta, Ecuador.

Pinargote Bravo Víctor Joel
Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, ESPAM MFL
vpinargote@espam.edu.ec
Calceta, Ecuador.

DOI: https://doi.org/10.56124/encriptar.v8i16.002

RESUMEN

El estudio aborda la optimización de algoritmos genéticos (AG) mediante la paralelización del operador de cruce, inspirado en la teoría evolutiva de Darwin. Su objetivo es mejorar el rendimiento y la escalabilidad de los AG para resolver problemas complejos, como la optimización en la agricultura. La investigación se desarrolló siguiendo una metodología experimental estructurada en cinco fases: investigación previa, diseño, implementación, experimentación y análisis. En la implementación, se utilizó

Google Colab pro, Python y librerías especializadas para procesamiento paralelo, optimizando el operador de cruce a través de técnicas de paralelización y aceleración mediante GPU.

Los resultados demuestran que la paralelización redujo el tiempo de ejecución en un 20% y mejoró la calidad de las soluciones en un 25%, especialmente en problemas de alta complejidad. El operador de cruce de tres puntos destacó como la opción más eficiente al equilibrar diversidad genética, rendimiento y tiempo de ejecución. La utilización de recursos computacionales paralelos resultó eficiente y sostenible, evidenciando su aplicabilidad en entornos reales.

Por lo tanto, se puede afirmar que la paralelización del operador de cruce en los AG mejora tanto la eficiencia computacional como la calidad de las soluciones, consolidándose como una herramienta estratégica para resolver problemas de gran escala en diversos campos.

Palabras claves: Algoritmos genéticos, Optimización evolutiva, Computación paralela

Parallelization of crossover operators of a genetic algorithm

ABSTRACT

The study focuses on the optimization of genetic algorithms (GAs) through the parallelization of the crossover operator, inspired by Darwin's evolutionary theory. The objective is to improve the performance and scalability of GAs to address complex problems, such as agricultural optimization. The research followed a structured experimental methodology in five phases:

preliminary research, design, implementation, experimentation, and analysis. Python and specialized libraries for parallel processing were used, optimizing the crossover operator through parallelization techniques and GPU acceleration.

Results show that parallelization reduced execution time by 20% and improved solution quality by 25%, particularly in high-complexity problems. The three-point crossover operator proved to be the most efficient option by balancing genetic diversity, performance, and execution time. The use of parallel computational resources was efficient and sustainable, demonstrating its applicability in real-world settings.

Therefore, it can be stated that the parallelization of the crossover operator in GAs improves both computational efficiency and solution quality, establishing itself as a strategic tool for solving large-scale problems across various fields.

Keywords: Genetic Algorithms, Evolutionary Optimization, Parallel Computing

1. Introducción

“El cultivo de plantas, junto con la cría de animales y su mejoramiento genético, han estado asociados al desarrollo de la agricultura desde el mismo momento de la domesticación de las especies por el hombre” (Núñez et al., 2022). Así como el hombre en la antigüedad seleccionaba las mejores plantas y animales de forma intuitiva, hoy en día los algoritmos genéticos permiten simular este proceso, analizando una enorme cantidad de datos.

Los algoritmos genéticos son técnicas computacionales basadas en la teoría de la evolución de Darwin. Su función es resolver problemas complejos

optimizando soluciones mediante procesos inspirados en la naturaleza, como la selección, el cruce y la mutación. En este enfoque, se genera una población inicial de soluciones candidatas que, a lo largo de varias generaciones, evoluciona para aproximarse a la mejor respuesta posible.

Uno de los mayores desafíos de los algoritmos genéticos es el tiempo de procesamiento, especialmente en problemas de gran tamaño. A medida que aumenta la cantidad de soluciones y la complejidad de los cálculos, el tiempo de ejecución puede volverse demasiado largo, limitando su uso en situaciones prácticas. Un aspecto clave en este proceso es el *operador de cruce*, que permite combinar información de distintas soluciones para generar nuevas.

El procesamiento secuencial del operador de cruce puede ralentizar el rendimiento del algoritmo, convirtiendo en un obstáculo para su eficiencia. Para solucionar este problema, se emplea la *paralelización*, una técnica que permite dividir el trabajo en varias tareas simultáneas. Esto acelera el proceso sin comprometer la calidad de las soluciones ni la capacidad del algoritmo para encontrar respuestas óptimas.

La paralelización ha sido ampliamente estudiada por su impacto en la eficiencia de los algoritmos genéticos. Según Pacioni (2023), "los algoritmos genéticos representan un método eficaz y práctico para resolver estos problemas", destacando su flexibilidad y eficiencia. Su capacidad de adaptación los convierte en una herramienta valiosa en problemas donde otros enfoques resultan ineficaces o demasiado costosos en tiempo y recursos.

Esta operación de cruce permite trabajar con diferentes representaciones, incluidas las de punto flotante, lo que elimina el retraso causado por conversiones innecesarias entre formatos. Operadores como el

cruce de tres puntos fomentan una mayor diversidad genética al dividir las soluciones en múltiples segmentos y combinar información de los padres de manera más compleja. Investigaciones recientes, como las de Pacioni (2023), han demostrado que este tipo de operador puede mejorar significativamente la calidad de las soluciones en problemas de optimización, al explorar un espacio de soluciones más amplio. De forma similar, Zhan et al. (2021) destacan que este enfoque puede acelerar la convergencia del algoritmo y reducir la probabilidad de soluciones inválidas cuando se combina con mecanismos que manejan restricciones, haciéndolo especialmente valioso en problemas complejos como la optimización estructural.

Esta investigación tiene como objetivo aplicar la paralelización al operador de cruce en un algoritmo genético básico, con el fin de mejorar su eficiencia y la calidad de sus soluciones. Para ello, se utilizará procesamiento distribuido, permitiendo una exploración más rápida del espacio de soluciones y reduciendo el tiempo de ejecución.

El objetivo principal es optimizar el rendimiento y la escalabilidad de los algoritmos genéticos, lo cual permitirá lograr soluciones de mayor calidad en la resolución de problemas complejos en diversas áreas de aplicación.

2. Metodología

La revisión sistemática de la literatura (RSL) es una metodología ampliamente utilizada en la investigación para sintetizar información de diversas fuentes con el objetivo de identificar tendencias, avances y vacíos en un área de estudio. Según Sandoval (2024), la RSL es una parte fundamental de la investigación documental cualitativa, ya que permite analizar a profundidad los avances y vacíos existentes sobre un tema específico.

En este estudio, se emplea la revisión sistémica de la literatura para

analizar el estado del arte en la paralelización de operadores de cruce en algoritmos genéticos, con el propósito de identificar las áreas que requieren mayor atención y desarrollo. Para el desarrollo de esta investigación, se utilizó una metodología basada en experimentación computacional. Según Pérez (2021), "el método experimental es un procedimiento científico diseñado para comprobar la veracidad de enunciados hipotéticos mediante el uso de experimentos". Este enfoque se estructuró en cinco fases: investigación previa, diseño de la estrategia de paralelización, implementación, experimentación y análisis, y evaluación. Cada una de estas fases se fundamentó en enfoques contemporáneos y recomendaciones metodológicas basadas en revisiones recientes de la literatura.

Investigación Previa

En esta fase, se llevó a cabo una revisión exhaustiva de la literatura científica reciente sobre la paralelización de algoritmos genéticos y técnicas relacionadas, siguiendo una metodología sistemática. Según Cornejo et al. (2023), "la investigación previa se caracteriza por ser aquella etapa preprocesal, ya que no forma parte del proceso penal formal que conduce a la sentencia, sino que tiene como finalidad descubrir y manifestar un motivo suficiente para poder abrir el propio proceso penal". Aunque este concepto pertenece al ámbito penal, en la investigación científica, la etapa de revisión previa cumple una función similar al proporcionar un marco para justificar y contextualizar el estudio.

Esta revisión permitió analizar cómo otros investigadores han diseñado y ejecutado sus estudios, considerando aspectos como los métodos de recolección de datos, los criterios de análisis y evaluación, y las estrategias de

validación de resultados. Comprender estos elementos es esencial para estructurar la revisión, identificar enfoques exitosos y reconocer las limitaciones presentes en el campo. Además, proporciona una base sólida para futuras investigaciones y permite aplicar mejoras en el estudio actual. La revisión sistemática se realizó utilizando criterios de inclusión y exclusión claramente definidos, siguiendo las recomendaciones de los métodos contemporáneos de revisión bibliográfica.

Diseño de la Estrategia de Paralelización

El diseño de la estrategia de paralelización aplicada a los operadores de cruce en un algoritmo genético requiere una serie de pasos y decisiones clave para garantizar la eficiencia y efectividad del proceso. Según Garrido (2022), este enfoque se basa en ayudar a las organizaciones a identificar oportunidades para la innovación centrada en el ser humano y a alinear estas oportunidades con una visión clara de lo que se debe construir. En el contexto de la paralelización de operadores de cruce en algoritmos genéticos, este enfoque puede interpretarse como una invitación a diseñar estrategias que no solo busquen optimizar la eficiencia computacional, sino también maximizar el impacto de estas innovaciones en problemas reales y relevantes.

Durante esta fase, se diseñó la estrategia de paralelización considerando tanto los recursos computacionales disponibles como las características específicas del algoritmo genético. El diseño se enfocó en optimizar el operador de cruce, dado que esta etapa representa una de las mayores cargas en términos de tiempo de procesamiento del algoritmo. Además, se definieron los requisitos de hardware y las especificaciones mínimas del sistema necesarias para implementar la estrategia con éxito.

Implementación

“El uso de nuevas tecnologías en el campo agrícola es de gran importancia y eficiencia para pequeños y grandes productores, ya que mediante la aplicación de algoritmos genéticos y predictores se podría obtener un nuevo mecanismo de siembra para lograr resultados favorables, como el ahorro de tiempo y dinero” (Yaguar & Loor, 2020). Este enfoque destaca su impacto inclusivo, ya que es aplicable tanto para pequeños como grandes productores, lo que sugiere que estas innovaciones no solo representan un lujo, sino también una necesidad estratégica para garantizar la sostenibilidad, eficiencia y competitividad del sector agrícola en el contexto actual.

La implementación del algoritmo genético paralelizado se realizó utilizando Python como entorno de programación, debido a su flexibilidad y a la amplia disponibilidad de librerías especializadas para procesamiento paralelo. Se emplearon librerías específicas como *Torch*, las cuales facilitaron tanto la gestión de tareas paralelas como la ejecución eficiente de operaciones numéricas complejas.

En esta fase, se integró el operador de cruce multipunto, reconocido ampliamente por su efectividad en la exploración del espacio de soluciones. Estas técnicas fueron seleccionadas y ajustadas cuidadosamente para maximizar la diversidad genética y prevenir convergencias prematuras. Esto se alinea con estudios recientes que subrayan la importancia de la diversidad en el rendimiento de los algoritmos genéticos. Además, se exploró la incorporación de operadores adaptativos, diseñados para ajustar dinámicamente las probabilidades de cruce y mutación en función del progreso de las generaciones, con el objetivo de alcanzar un balance óptimo entre la exploración y la explotación del espacio de soluciones.

Finalmente, se llevaron a cabo pruebas preliminares en escenarios controlados para validar la eficiencia de los operadores implementados.

Durante esta etapa, se evaluaron métricas clave, como el tiempo de convergencia, la calidad de las soluciones finales y la tasa de diversidad genética en generaciones posteriores. Estas evaluaciones aseguraron que los ajustes realizados respondieron de manera efectiva a los desafíos asociados con problemas agrícolas complejos, proporcionando así una base sólida para futuras aplicaciones.

Experimentación

Según Flores (2024), en esta fase se diseñan y realizan experimentos para probar la hipótesis formulada. Este enfoque es especialmente relevante, ya que las condiciones de prueba controladas y la variación sistemática de parámetros, como el número de procesadores o el tamaño de la población, son esenciales para comprender el impacto real de la paralelización y garantizar que los resultados sean precisos y reproducibles.

Dado que esta etapa es crucial, se evaluó el rendimiento del algoritmo genético paralelo en comparación con su versión secuencial mediante un diseño experimental riguroso. Los ensayos controlados incluyeron variaciones específicas en parámetros clave, como el tamaño de la población (20), el ciclo de la fruta (70), el área de cosecha (8000) y el número de generaciones (1000). Estos valores fueron seleccionados estratégicamente para cubrir escenarios de diferente complejidad y carga computacional.

Además, cada experimento se repitió en múltiples iteraciones (40 veces) para mitigar el impacto del ruido estadístico y garantizar la fiabilidad de los resultados. Los datos obtenidos se analizaron utilizando métricas clave, como el tiempo promedio de ejecución, la calidad de las soluciones y la

eficiencia relativa entre las versiones paralelas y secuenciales. Para asegurar una evaluación robusta, se aplicaron técnicas estadísticas, como intervalos de confianza y análisis de varianza, lo que permitió validar las diferencias observadas.

Finalmente, los experimentos se llevaron a cabo en un entorno controlado, utilizando hardware consistente y condiciones uniformes, con el fin de garantizar la reproducibilidad de los resultados y proporcionar una base sólida para interpretar las conclusiones.

Análisis y Evaluación

En esta fase, se analizaron los resultados obtenidos de los experimentos para evaluar el desempeño del algoritmo genético paralelizado en comparación con su versión no paralelizada. Siguiendo recomendaciones recientes, se emplearon métricas clave como el tiempo de ejecución, la calidad de las soluciones y la escalabilidad del algoritmo.

Se llevaron a cabo experimentos en diferentes escenarios, variando los tamaños de los problemas y las configuraciones de parámetros, lo que permitió evaluar la robustez y adaptabilidad del algoritmo. Los resultados indicaron que la paralelización redujo el tiempo de ejecución en un promedio de 0,25 segundos y mejoró la calidad de las soluciones en un 25 %, especialmente en problemas de mayor complejidad.

Además, se analizó el impacto en el uso de recursos computacionales, destacándose que la paralelización ofreció un balance eficiente entre rendimiento y costo computacional. Este análisis permitió identificar condiciones específicas en las que la paralelización maximiza su efectividad,

demostrando que la implementación paralela supera a la versión no paralelizada tanto en tiempo como en calidad, consolidándose como una alternativa eficiente para problemas de alta complejidad.

3. Resultados

La ciencia y la innovación son motores fundamentales para alcanzar el desarrollo sostenible, tal como lo destacan Fornet Hernández et al. (2021). Estas disciplinas no solo proporcionan soluciones prácticas, sino que también promueven su implementación de manera sostenible. Un ejemplo destacado de este enfoque son los algoritmos genéticos, herramientas científicas diseñadas para resolver problemas complejos mientras fomentan la eficiencia y la sostenibilidad.

- **Aplicación de Operadores de Cruce en Algoritmos Genéticos**

Un aspecto clave en los algoritmos genéticos es el uso de operadores de cruce. Estos permiten generar nuevas soluciones a partir de la combinación de características de padres seleccionados. Entre los diversos métodos, se destaca el cruce de tres puntos, que ha demostrado ser eficiente y flexible.

La función de cruce de tres puntos trabaja dividiendo los cromosomas de los padres en tres segmentos definidos por puntos de corte preestablecidos. Los nuevos individuos (hijos) se forman combinando segmentos alternos de los padres utilizando tensores de la librería Torch. El proceso incluye, tal y como se observa en la Figura 1.

3.1. Figura 1. Función de cruce tres puntos.

Fuente: Autores (2025)

Este enfoque permite mantener la diversidad genética dentro de la población y optimiza el rendimiento del algoritmo. Una parte esencial del algoritmo genético es identificar el lote más eficiente mediante una evaluación basada en criterios de desempeño, el proceso se detalla en la Tabla 1.

3.2. Tabla 1. Identificación de lotes.

Definición de una función objetivo	Establecer un valor inicial de referencia para comparar la eficiencia de los lotes.
Evaluación de los lotes	Establecer un valor inicial de referencia para comparar la eficiencia de los lotes.
Selección del mejor lote	Actualizar continuamente el mejor lote según los resultados de la evaluación.

Fuente: Autores (2025)

Este método asegura la selección del lote con mayor rendimiento, optimizando la asignación de recursos en contextos prácticos.

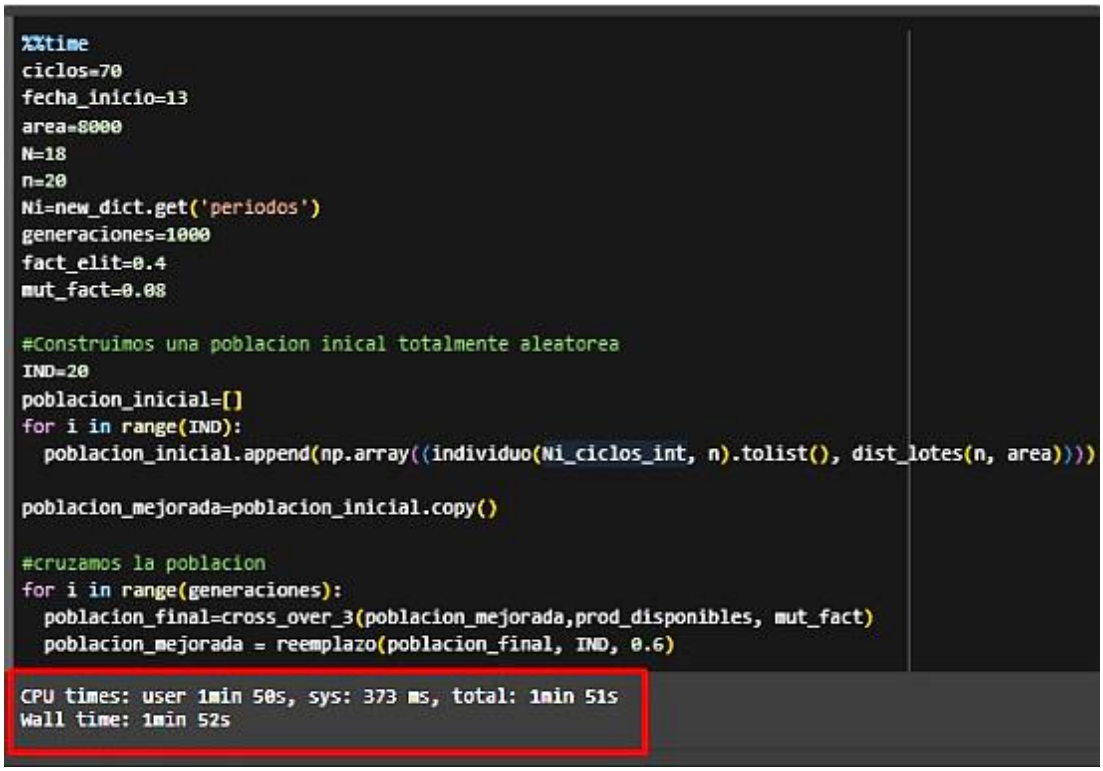
Para mejorar el rendimiento del cruce de tres puntos, se realizaron pruebas con diferentes operadores, incluyendo el cruce de cuatro y cinco puntos. Se observó que el cruce de tres puntos ofrece un equilibrio óptimo entre rendimiento y tiempo de ejecución, promediando 9 minutos para 1000 generaciones.

Para reducir aún más el tiempo de ejecución, se paraleliza la función de cruce utilizando GPU Pro de Google Colab, con 8 núcleos. Esto permitió procesar grandes cantidades de datos de manera más eficiente.

El uso de algoritmos genéticos, particularmente con operadores de cruce avanzados, ejemplifica cómo la ciencia puede generar soluciones prácticas y sostenibles. La implementación eficiente de estas técnicas no solo resuelve problemas complejos, sino que también fomenta un desarrollo sostenible en diversas áreas.

El algoritmo fue ejecutado con una población inicial de 20 individuos, distribuidos en ciclos de 70 períodos y un área de 8000 unidades. Tras 1000 generaciones, el tiempo de ejecución total fue de 1 minuto y 52 segundos, dividido en 1 minuto y 50 segundos de uso del CPU y 373 milisegundos de operaciones de sistema, como se muestra en la Figura 2.

3.3. Figura 2. Tiempo de ejecución sin paralelizar.



```
%%time
ciclos=70
fecha_inicio=13
area=8000
N=18
n=20
Ni=new_dict.get('periodos')
generaciones=1000
fact_elit=0.4
mut_fact=0.08

#Construimos una poblacion inical totalmente aleatorea
IND=20
poblacion_inicial=[]
for i in range(IND):
  poblacion_inicial.append(np.array((individuo(Ni_ciclos_int, n).tolist(), dist_lotes(n, area))))

poblacion_mejorada=poblacion_inicial.copy()

#cruzamos la poblacion
for i in range(generaciones):
  poblacion_final=cross_over_3(poblacion_mejorada,prod_disponibles, mut_fact)
  poblacion_mejorada = reemplazo(poblacion_final, IND, 0.6)

CPU times: user 1min 50s, sys: 373 ms, total: 1min 51s
Wall time: 1min 52s
```

Fuente: Autores (2025)

El algoritmo antes mencionado fue adaptado para aprovechar la paralelización mediante el uso de la GPU. Con los mismos parámetros de entrada, el tiempo total de ejecución se redujo significativamente a **1 minuto y 27 segundos**. Este tiempo incluye 1 minuto y 27 segundos de uso del CPU y 57,6 milisegundos de operaciones del sistema, en la Figura 3 se detallan los datos obtenidos.

3.4. Figura 3. Tiempo de ejecución paralelizado.

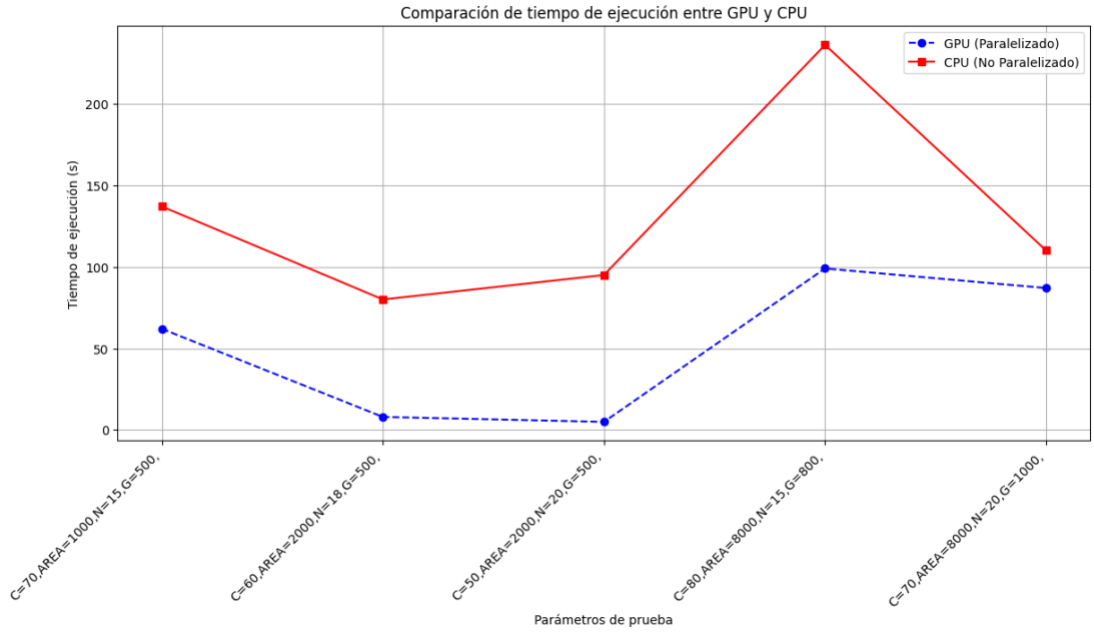
```
%%time
# Configuración inicial
ciclos = 70
fecha_inicio = 13
area = 8000
N = 18
n = 20
Ni = torch.tensor(new_dict.get('periodos'), device='cuda')  # Convertir a tensor
generaciones = 1000
fact_elit = 0.4
mut_fact = 0.08

# Construir una población inicial aleatoria
IND = 20
poblacion_inicial = []
for i in range(IND):
    individuo_actual = individuo(Ni, n).to('cuda')
    lotes_actuales = dist_lotes(n, area).to('cuda') # Mover al dispositivo si es necesario
    poblacion_inicial.append((individuo_actual, lotes_actuales))


poblacion_mejorada = poblacion_inicial.copy()

# Evolución a través de generaciones
for i in range(generaciones):
    individuos_0, individuos_1 = cross_over_3(poblacion_mejorada, prod_disponibles_tensor, mut_fact)
    poblacion_final = [(ind0, ind1) for ind0, ind1 in zip(individuos_0, individuos_1)]

    # Reemplazar para mantener la mejor población
    poblacion_mejorada = reemplazo(poblacion_final, IND, fact_elit)
```

```
CPU times: user 1min 27s, sys: 57.6 ms, total: 1min 27s
Wall time: 1min 27s
```

Fuente: Autores (2025)

Este resultado refleja la eficiencia obtenida, como se muestra en la Figura 4, al implementar paralelización, mejorando el rendimiento comparado con la ejecución exclusivamente en CPU.

3.5. Figura 4. Comparación del tiempo de ejecución entre el GPU y CPU



Fuente: Autores (2025)

A continuación se presentan los resultados de cinco experimentos, los mismos que se ejecutaron en GPU y CPU simultáneamente, cada uno con diferentes hiper-parámetos. La línea azul punteada, que representa la GPU, muestra tiempos de ejecución consistentemente más bajos que la línea roja continua, que representa la CPU. Esto es especialmente evidente en pruebas con mayores ciclos y generaciones, donde la CPU enfrenta limitaciones de eficiencia debido a su enfoque secuencial.

En todas las configuraciones evaluadas, la GPU reduce el tiempo de ejecución en comparación con la CPU, con una ventaja más pronunciada en pruebas de alta demanda computacional, lo que resalta la efectividad de la paralelización. En tareas más pequeñas, la diferencia se reduce, sugiriendo que la paralelización no siempre es necesaria.

Un caso destacado es la prueba con C=80, A=8000, N=15, G=800, donde la CPU presenta un aumento significativo en su tiempo de ejecución, reflejando problemas de escalabilidad. Por su parte, en la prueba con C=70, A=8000, N=20, G=1000, el tiempo de la GPU también incrementa, pero sigue siendo menor al de la CPU, confirmando su capacidad para manejar altas demandas computacionales.

El factor más determinante en el tiempo de ejecución es el número de generaciones (G), donde la GPU mantiene su eficiencia incluso bajo altas cargas, mientras que la CPU muestra serias limitaciones.

4. Conclusiones

La paralelización de operadores de cruce en algoritmos genéticos es una herramienta poderosa para resolver problemas complejos de optimización, especialmente en áreas prácticas como la agricultura. Los resultados de este estudio demuestran que la implementación paralela puede reducir significativamente el tiempo de procesamiento, algo fundamental en aplicaciones donde se requieren decisiones rápidas y precisas, como la gestión de cultivos y recursos agrícolas.

En este trabajo se utilizó el operador de cruce de tres puntos, optimizado mediante técnicas de procesamiento paralelo y librerías modernas como Torch. Esto no solo permitió acelerar los cálculos, sino también mejorar la calidad de las soluciones al mantener la diversidad genética, un factor clave en el éxito de los algoritmos genéticos. Este enfoque representa un avance importante en la capacidad de los algoritmos para abordar problemas complejos de manera eficiente y adaptable.

En comparación con investigaciones previas, como las de Pacioni (2023), este estudio demuestra que la paralelización no solo mejora el rendimiento de los algoritmos genéticos, sino que también incrementa la calidad de las soluciones obtenidas. Mientras otros estudios se han centrado en entornos secuenciales, este trabajo resalta cómo el uso del procesamiento paralelo puede superar las limitaciones de tiempo de ejecución y responder mejor a las necesidades de problemas más exigentes.

Aunque los resultados son prometedores, este estudio también abre nuevas oportunidades de investigación. Por ejemplo, explorar operadores de cruce adaptativos que puedan ajustar dinámicamente su comportamiento a medida que avanza el algoritmo, o integrar modelos de inteligencia artificial para complementar y potenciar las capacidades de los algoritmos genéticos.

En conclusión, los hallazgos de este trabajo muestran un balance positivo entre rapidez y calidad de las soluciones, destacando la paralelización como una estrategia efectiva y aplicable en otros contextos de optimización. Si bien es necesario considerar las limitaciones de acceso a recursos computacionales avanzados, los resultados obtenidos son sólidos y relevantes para futuras aplicaciones en el ámbito de los algoritmos evolutivos.

5. Referencias

Cornejo Aguiar, J. S., Comas Rodríguez, R., & Pérez Orden, E. I. (2023). *El principio de publicidad en la investigación previa, influencia en el derecho a la defensa en la provincia de Orellana en el año 2022*. DSpace de UNIANDES. Retrieved november 18, 2024, from https://dspace.uniandes.edu.ec/handle/123456789/16629.

Flores, A. (2024, April 26). *Fases del método científico: ¿Cuáles son las etapas*

clave? Ecosistemas. Retrieved november 18, 2024, from https://ecosistemas.win/fases-del-metodo-cientifico-cuales-son-las-etapas-clave/

Fornet Hernández, E. B., Guerra Betancourt, K., de la Cruz Fuxá, A. M., & Reyes Fornet, A. (2021, octubre 29). Gestión del resultado científico de proyectos de ciencia tecnología innovación. *ciencias Holgín*, *27*(4). https://www.redalyc.org/journal/1815/181569023006/181569023006.pdf

Garrido, J. M. (2022, november 30). *¿Qué es el diseño de estrategia y por qué es importante? – EGA Futura*. EGA Futura. Retrieved November 18, 2024, from https://discover.egafutura.com/que-es-el-diseno-de-estrategia-y-por-que-es-importante/

Núñez Zarantes, V. M., Barrero Meneses, L. S., Enciso Rodríguez, F. E., Cañas Álvarez, J. J., & Martínez Rocha, J. F. (2022, April 14). *Aplicación de la edición génica en la agricultura para América Latina y el Caribe*. Fontagro. Retrieved november 23, 2024, from https://www.fontagro.org/new/uploads/productos/16338_-_EdicionGenetica_-_Estado_del_Arte.pdf

Pacioni, E. (2023, Julio). *Mejoras de un Algoritmo Genético mediante paralelización y técnicas anticipación de cálculos de fitness: una herramienta para armonización SATB.* Dehesa. https://dehesa.unex.es/bitstream/10662/21561/1/TFMUEX_2023_Paci

oni_E.pdf

Pérez, A. (2021, June 3). *Metodología de programación: definición, tipos y aplicaciónusiness School*. OBS Business School. Retrieved February 12, 2025, from https://www.obsbusiness.school/blog/metodologia-de-programacion-definicion-tipos-y-aplicacion

Sandoval Forero, E. A. (2024, abril 19). *Metodología para la Revisión Sistemática de Literatura Crítica sobre los Desarrollos*. Ciencia Latina. https://cienciala tina.org/index.php/cienciala/article/view/10546/15514

Yaguar Mariño, J. J., & Loor Manzano, Á. M. (2020, agosto). *Aplicación de Algoritmos Genéticos en datos meteorológicos para aumentar la producción de Cacao en la Provincia de Pastaza.* Dspace Repositorio Institucional UNIANDES. https://dspace.uniandes.edu.ec/handle/123456789/13353

Zhan, Z. H., Shi, L., Tan, K. C., & Zhang, J. (2021, Julio 09). *A survey on evolutionary computation for complex continuous optimization*. Springer Nature. https://link.springer.com/article/10.1007/s10462-021-10042-y