**INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO**





Inteligencia artificial

Proyecto 2

**Generación automatizada de horarios**

Equipo 0:

* Morante Castillo Leopoldo
* Pizaña Vela Patricio
* Ambriz Núñez Juan Manuel
* Carrasco Hernández Abel Benito

Docente: Andrés Gómez de Silva Garza

Semestre: Primavera 2024

Índice:

[**Resúmen: 3**](#_ibcg64qbbjdy)

[**Introducción 3**](#_s6bemfrpkglq)

[**El algoritmo evolutivo de asignación de horarios 4**](#_byigwhjswcot)

[El problema a resolver 4](#_hjur5g55673n)

[Las materias 4](#_qih3owm7y0qm)

[Las instalaciones 5](#_8evpl0ghg4ft)

[Los horarios 5](#_b17l7al0ci1l)

[El alumnado y profesorado 5](#_dn6338ozouup)

[Implementación 6](#_3rknazwwoudd)

[Clases 6](#_3rknazwwoudd)

[El proceso evolutivo 7](#_50ifw9sqd8p)

[La función de aptitud 8](#_ehk149ivotxr)

[Los operadores genéticos 8](#_tukrk62mcarf)

[Cruzamiento 8](#_683fmkdfwja1)

[Mutación 9](#_huol1mvmfnnq)

[**El algoritmo evolutivo de parámetros de algoritmo evolutivo 9**](#_h8mcqyz1axuj)

[El problema a resolver 9](#_5qm08dl4frbz)

[Implementación 9](#_g48uodn2k7k8)

[La función de aptitud 10](#_2do3reesr8re)

[Operadores genéticos 10](#_6w87bzwk674a)

[Cruzamiento 11](#_deo3xqej52hl)

[Mutación 11](#_jq7t1vibyn2x)

[**Resultados 11**](#_76813kq4nigo)

[**Conclusiones 16**](#_549mj1s5ta73)

[Referencias bibliográficas 17](#_fbjvtoi2bt32)

## *Resúmen:*

No todos los algoritmos evolutivos que pretenden resolver un mismo problema tienen la misma calidad. La eficiencia para encontrar una solución y la calidad de la misma dependen fuertemente de la implementación y la determinación arbitraria de algunos parámetros, como el número de generaciones, tamaño de la población o peso que tienen las características de una posible solución al momento de determinar su aptitud. Estos parámetros se suelen asignar basados en experiencia, conocimiento e intuición, pero siempre terminan limitando el rendimiento del algoritmo.

Este proyecto explora la posibilidad de utilizar un algoritmo evolutivo en un problema de optimización muy particular: el de maximizar el rendimiento de otro algoritmo evolutivo. El análisis sobre la utilidad de este enfoque está basado en pruebas realizadas sobre la implementación de ambos algoritmos evolutivos en el lenguaje de programación Python.

**Palabras clave: evolución, algoritmo, horarios.**

## *Introducción*

Los algoritmos evolutivos son una técnica computacional que busca resolver problemas de optimización y búsqueda utilizando la teoría evolutiva biológica. Consisten en seleccionar individuos, dentro de una población, que cumplen con características específicas para acercarse a la resolución de un problema. A través de un proceso iterativo, se mantienen a los más aptos por generación, los cuales participan en una modificación genética para crear a los individuos de las siguientes generaciones. Otro elemento dentro de estos algoritmos es la mutación, “pequeñas variaciones que introducen diferencias en las características físicas y tipos de respuesta de los padres y los hijos.” (Araujo & Cervigón, 2009)

El esquema general de un algoritmo evolutivo se describe en los siguientes puntos mencionados en el libro *Algoritmos Evolutivos* de Araujo y Cervigón:

1. Procesan simultáneamente todo un conjunto de soluciones potenciales a la solución del problema, que se denominan individuos. El conjunto de todos ellos forman la población con la que trabaja el algoritmo.
2. La población se modifica a través de iteraciones del algoritmo llamadas generaciones. Entre cada generación, además de variar el número de copias de un mismo individuo en la población, también aparecen nuevos individuos generados mediante operaciones de transformación sobre individuos de la población anterior. Estas operaciones se conocen como operadores genéticos.
3. En cada generación, se realiza un proceso de selección que otorga una mayor probabilidad a los mejores individuos para permanecer en la población y participar en las operaciones de reproducción. Estos mejores individuos son aquellos que producen los valores más altos en la función de adaptación del algoritmo. Es crucial que esta selección se realice de manera aleatoria para considerar también a los individuos de baja adaptación, ya que esto evita quedarse atrapado en soluciones óptimas locales.

El proyecto tiene como objetivo aplicar los principios de los algoritmos evolutivos en este contexto para resolver el problema de la creación de horarios de exámenes en instituciones educativas, optimizando la distribución de recursos y maximizando los resultados.

No todos los algoritmos evolutivos destinados a resolver un mismo problema ofrecen la misma calidad en términos de rendimiento temporal, uso de memoria y calidad de la solución obtenida. Esta disparidad se debe, en parte, a que, aunque todos sean algoritmos evolutivos, ciertos elementos son tratados de manera distinta. Variables como el número total de generaciones, la probabilidad de mutación frente a la de cruzamiento, el tamaño inicial de la población y el peso otorgado a una característica específica de un individuo para determinar su aptitud, son consideraciones que el desarrollador del algoritmo debe determinar basándose en su experiencia e intuición.

El objetivo de este proyecto es diseñar e implementar un algoritmo evolutivo cuya finalidad sea encontrar valores óptimos para esas variables, maximizando así el rendimiento y la calidad de la solución de otro algoritmo evolutivo, además de analizar los resultados obtenidos. Para lograrlo, se llevaron a cabo los siguientes pasos:

1. Diseño e implementación en Python de un algoritmo evolutivo con el propósito de asignar horarios y salones para los exámenes finales de una institución educativa.
2. Diseño e implementación en Python de otro algoritmo evolutivo cuyo objetivo sea determinar los parámetros que generen los mejores resultados al ser utilizados dentro del primer algoritmo.
3. Ejecución del segundo algoritmo bajo diferentes supuestos y recopilación de información sobre esas ejecuciones.
4. Análisis de la información obtenida para determinar si el segundo algoritmo logró su objetivo.

## *El algoritmo evolutivo de asignación de horarios*

### **El problema a resolver**

El algoritmo evolutivo diseñado inicialmente se enfoca en asignar lugar y hora a los exámenes finales de una institución educativa específica. Para lograr esto, fue necesario caracterizar detalladamente la institución en la que nuestro algoritmo operaría. Los elementos clave de la institución y el proceso de caracterización son los siguientes:

#### **Las materias**

Se utilizaron las materias disponibles para el periodo de primavera 2024 en el ITAM. El número total de estas es de 431.

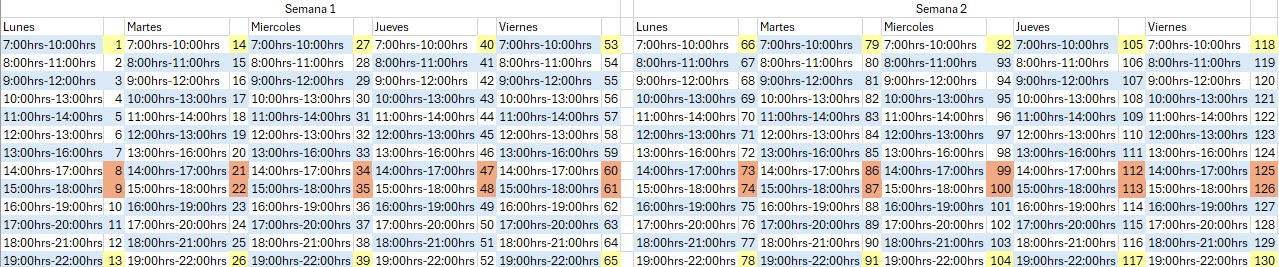
#### **Las instalaciones**

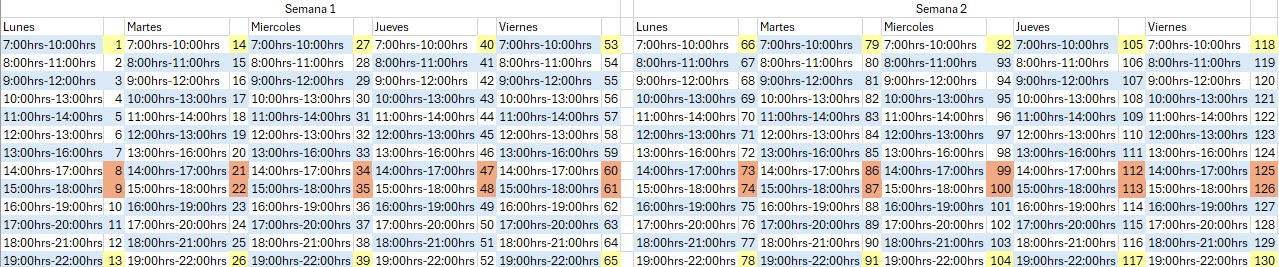
Se estableció un número fijo de salones (100) y a cada uno de ellos se le asignó una capacidad de manera aleatoria. La capacidad del salón varía entre 10 y 50 alumnos.

#### **Los horarios**

Se decidió que, para esta institución educativa, todos los exámenes duran un total de 3 horas. Asimismo, se establecieron horarios de inicio de exámenes fijos que impidieron el inicio de nuevos exámenes en medio de la realización de algún otro. Los horarios fueron asignados bajo el supuesto de que la institución dispone de 2 semanas para realizar sus exámenes y que en cada una de ellas dispone de 5 días.

*Tabla 1. Horarios para exámenes*





#### **El alumnado y profesorado**

Tanto los alumnos como los profesores son individuos que tienen la responsabilidad de asistir a los exámenes finales de las materias en las que están inscritos (alumnos) o que imparten (profesores). Por lo tanto, su representación en este problema es idéntica: la de una persona asociada a una lista de materias.

La lista de materias a las que una persona está relacionada (ya sea por inscripción o impartición) se generó de manera aleatoria. En primer lugar, se determinó la cantidad de materias a las que estaría relacionada la persona y luego se seleccionaron aleatoriamente entre todas las materias disponibles.

En total, se generaron 100 personas para este estudio.

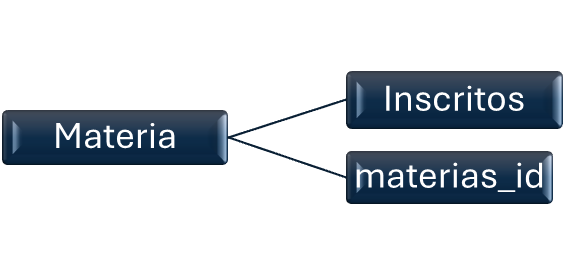
### *Implementación*

### **Clases**

Para realizar una implementación eficiente fue necesaria la construcción de clases que permiten el funcionamiento del algoritmo.

**Clase Materia:**

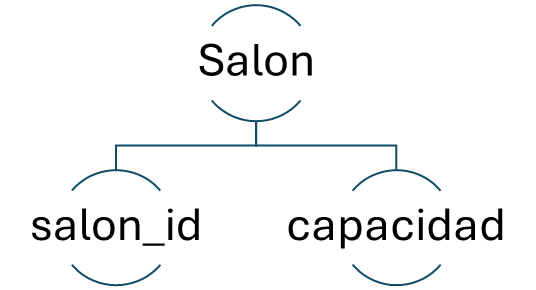
La clase Materia es una representación de las materias que se imparten en la Universidad. Cada sesión de esta clase almacena datos específicos sobre un tema, como su identificador único y el número de estudiantes inscritos.



*Figura 1. Clase Materia*

**Clase Salón:**

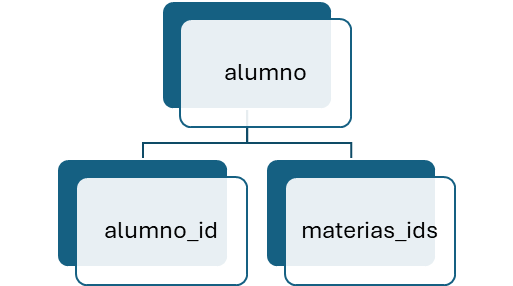
Representa los salones disponibles que tiene la Universidad. Cada sesión de esta clase almacena datos específicos sobre un salón, como su identificador único y la capacidad máxima de estudiantes. Esta clase es esencial para el proceso de asignación de horarios de exámenes finales porque garantiza que los salones sean adecuados para el número de estudiantes inscritos en cada materia.



*Figura 2. Clase Salon*

**Clase Alumno:**

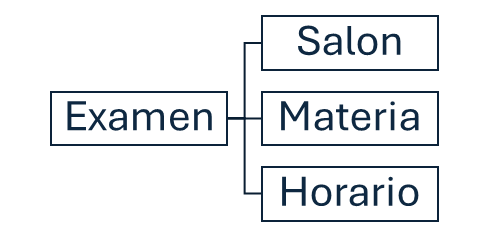
Presenta a los estudiantes que están inscritos en la Universidad. Cada instancia de esta clase almacena información específica sobre un estudiante, como su identificador único y las materias en las que está inscrito. El algoritmo de asignación de horarios de exámenes finales penaliza si hay conflictos de horarios para un solo alumno gracias a la lista de materias inscritas.



*Figura 3. Clase Alumno*

**Clase Examen:**

Representa los exámenes finales de la escuela. Las instancias de esta clase almacenan información específica sobre un examen, como el salón donde se llevará a cabo, la materia que se evaluará y el horario que se asignará para realizar dicho examen.



*Figura 4. Clase Examen*

### *El proceso evolutivo*

El proceso evolutivo del algoritmo sigue un ciclo while definido, donde se ejecutan pasos específicos en un orden determinado. Comienza generando una población inicial de soluciones aleatorias que representan posibles horarios de exámenes. Luego, se verifica si se ha alcanzado un criterio de terminación, como un número máximo de generaciones o una aptitud mínima deseada.

En cada iteración del ciclo *while*, se evalúa la aptitud de cada solución utilizando una función de aptitud que considera diversos criterios, como la programación de exámenes en horas no deseadas y la capacidad de los salones asignados. Posteriormente, se seleccionan los individuos más aptos mediante técnicas de selección y se aplican operadores genéticos como la mutación y la recombinación para generar una nueva población.

Este ciclo se repite hasta que se cumple la condición de finalización, permitiendo que la población evolucione gradualmente hacia soluciones más óptimas en la organización de los horarios de exámenes. Al final del proceso, el algoritmo devuelve información sobre los resultados obtenidos. Esto incluye la mejor solución alcanzada, el número de generaciones transcurridas antes de detenerse, la varianza final y el número de errores presentes en la mejor solución encontrada por el algoritmo.

### *La función de aptitud*

La función de aptitud en el algoritmo evolutivo evalúa la calidad de un horario considerando varios aspectos clave. Penaliza la programación de exámenes en horas indeseadas, como muy temprano, muy tarde o a la hora de la comida, para asegurar condiciones ideales para los estudiantes. Además, tiene en cuenta la capacidad de los salones asignados, penalizando la asignación de salones con capacidad insuficiente para el número de estudiantes inscritos. También penaliza la superposición de exámenes en un mismo salón, evitando conflictos logísticos, y la superposición de exámenes para los alumnos, garantizando que puedan realizar todos sus exámenes sin problemas de horarios.

### *Los operadores genéticos*

Stuart Russell y Peter Norvig describen que los algoritmos genéticos funcionan con la premisa de que efectuando una adecuada serie de pequeñas mutaciones a un programa de código máquina se podría generar un programa con buen rendimiento aplicable en cualquier tarea sencilla (2004, p.25).

Tras haber evaluado a todos los individuos de la población mediante la función de aptitud, se realiza un proceso de selección de los individuos más aptos. Decidimos utilizar un criterio simple para determinar quienes son los más aptos. Aquellos individuos cuya aptitud sea mayor a la del promedio de su generación, será seleccionado y participará en la formación de la nueva generación mediante la aplicación de los operadores genéticos de cruzamiento y mutación.

#### **Cruzamiento**

Esta función es un paso importante en nuestro algoritmo evolutivo porque implica unir dos personas de la población para crear una nueva. Esta función selecciona aleatoriamente características de cada uno de los dos individuos (padres) para generar un descendiente, similar al cruce genético en la reproducción biológica. El cruzamiento implica la combinación de horarios o salones de uno de los padres con los del otro, lo que significa que el atributo compartido entre las personas en nuestro algoritmo es el de materia.

El propósito fundamental de esta función es explorar el espacio de búsqueda de soluciones y mezclar características para promover una mayor diversidad genética en las generaciones futuras.

#### **Mutación**

La función de mutación toma un miembro de la población y hace una mutación en cada examen de ese miembro. Cada examen tiene dos formas posibles de realizar la mutación, en ambos casos se mantiene el mismo curso:

1. Cambiar el horario del examen, pero mantener el salón.
2. Cambiar el salón, pero mantener el horario.

Tomar un individuo (que representa un horario) como entrada y crear una versión mutada del mismo se realiza a través de un algoritmo evolutivo para introducir cambios aleatorios en los individuos; así, descubrir nuevas soluciones a los problemas.

## *El algoritmo evolutivo de parámetros de algoritmo evolutivo*

### **El problema a resolver**

El segundo algoritmo evolutivo, pretende determinar qué parámetros hacen que el algoritmo evolutivo de organización de horarios entregue los mejores resultados posibles.

### *Implementación*

En el caso de este algoritmo evolutivo, no se hizo uso de clases. En su lugar, se creó una población donde cada individuo es un vector de 8 números que representan los parámetros de entrada necesarios para ejecutar el algoritmo evolutivo de horarios para exámenes finales. Estos parámetros son el tamaño de la población, el número de generaciones, la varianza máxima (criterio de convergencia), la probabilidad de que ocurra una mutación, y cuatro valores enteros: pesoH, pesoC, pesoE y pesoA, que corresponden al impacto que tiene en la evaluación de una solución cometer los diferentes errores penalizables (asignar salones con capacidad insuficiente para un examen, programar exámenes en horarios inadecuados, etc.).

El proceso del algoritmo evolutivo mantiene la misma estructura general que el otro algoritmo. Se realizan varias iteraciones y en cada una de ellas se verifica la aptitud de cada individuo y se aplican los operadores de cruzamiento y mutación sobre los aptos para generar una nueva población. Las principales diferencias entre el primer y segundo algoritmo radica en la implementación de las funciones de aptitud, cruzamiento y mutación.

### *La función de aptitud*

Para evaluar la aptitud de un individuo de la población, es necesario poner a prueba los parámetros/al individuo mediante la ejecución del algoritmo evolutivo de horarios. Tras ejecutarlo, se recuperan los valores de salida de “tiempo de ejecución”, “mejor solución”, “varianza final” y “total de errores” y se utilizan para calcular la aptitud de la solución.

Se utilizaron dos versiones de la función de aptitud para comparar los resultados del algoritmo bajo diferentes criterios, uno que prioriza la minimización del tiempo de ejecución y otro que minimiza el número de errores de la solución final del algoritmo evolutivo de horarios.

La primera función está definida de la siguiente manera:

,

*Función 1.*

: tiempo de ejecución

: número de errores

v: varianza final

Mientras que la segunda está definida de la siguiente forma:

,

*Función 2.*

: tiempo de ejecución

: número de errores

v: varianza final

El uso de una exponencial en ambas funciones permite “castigar” a las soluciones que tengan numerosos errores o grandes tiempos de ejecución. Inicialmente, el exponente no era modificado mediante ningún cociente, pero los valores que podía llegar a tomar la exponencial, en especial cuando se aplicaba sobre la variable tiempo, eran tan grandes que las variables en el programa de python eran incapaces de contenerlos. Por ello, se dividieron los valores de las variables para limitar su crecimiento pero manteniendo la penalización sobre valores que se alejaban del tiempo/número de errores deseados.

### *Operadores genéticos*

Al igual que en el primer algoritmo, se utilizó el promedio de la generación para separar aptos de no aptos.

La función de Cruzamiento y la función de Mutación tienen una estructura similar, pero difieren en la implementación de la función de aptitud y los operadores genéticos. Si bien comparten el mismo enfoque evolutivo, la manera en que evalúan a los individuos y aplican los operadores es diferente; así determinan a los individuos aptos y no aptos.

#### **Cruzamiento**

La función de Cruzamiento genera un nuevo individuo a partir de las características de sus padres. Para todos los parámetros que conforman a un individuo, se elige aleatoriamente, con misma probabilidad, si el nuevo individuo heredará ese parámetro de su “padre” o de su “madre”

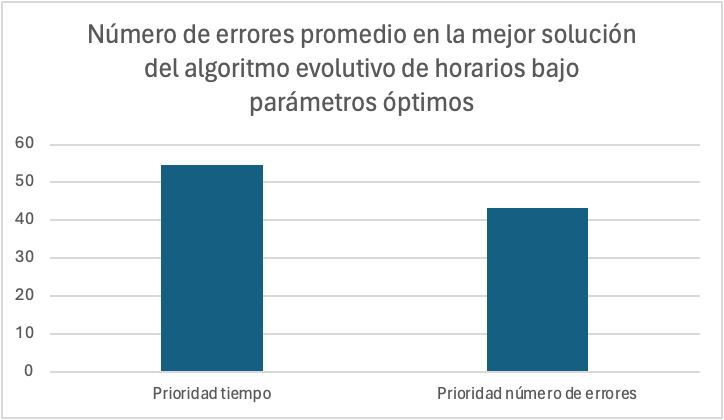
#### **Mutación**

Por su parte, la función de Mutación genera un nuevo individuo manteniendo algunas de las características del individuo original que modifica. Para todos los parámetros, se toma una decisión: se conserva el parámetro o se modifica. En el primer caso, el parámetro pasa inmediatamente al nuevo individuo, mientras que en el segundo, se genera aleatoriamente un nuevo valor. Estos nuevos valores pertenecen a intervalos que hacen de ellos congruentes con lo que pretenden representar. La probabilidad de mutar por ejemplo, solamente puede variar entre 0 y 1 mientras que el número de generaciones es un entero mayor a 1.

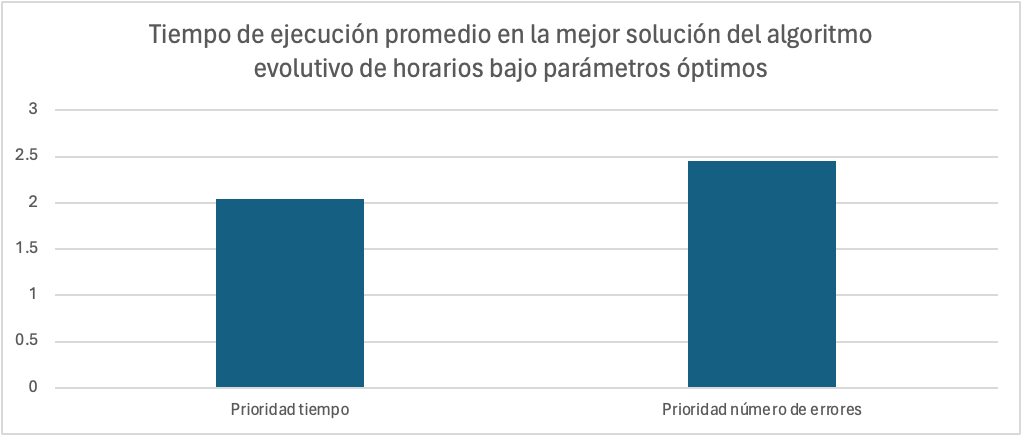
### *Resultados*

Una vez implementados ambos algoritmos, se llevaron a cabo múltiples ejecuciones utilizando diferentes valores de sus parámetros con el fin de comparar el rendimiento del algoritmo en diversas condiciones. Para cada uno de los 4 enfoques planteados, se realizaron 40 ejecuciones del algoritmo de algoritmos evolutivos.

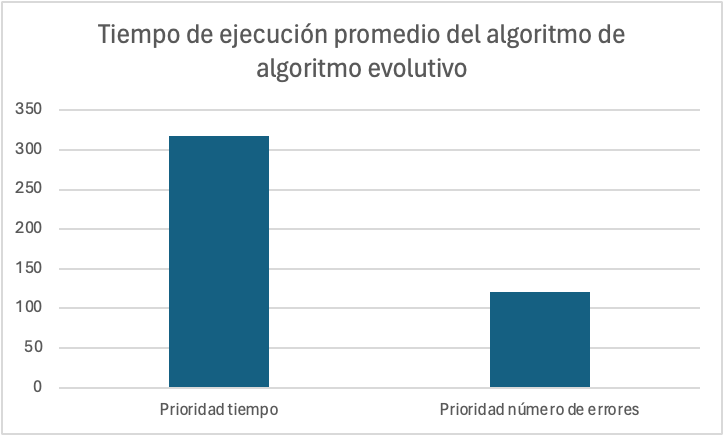
Tanto el primer como el segundo enfoque compartieron las mismas condiciones de parámetros. El tamaño de la población y el número de generaciones variaron entre 1 y 10, la probabilidad de mutación fue aleatoria y la varianza máxima estuvo entre 1 y 500. La diferencia principal entre ambos escenarios radicaba en la función de aptitud utilizada: el primero priorizaba el tiempo, mientras que el segundo daba prioridad al número de errores.



*Gráfica 1. Tiempo vs, errores en errores.*



*Gráfica 2. Tiempo vs. errores en rendimiento horarios.*

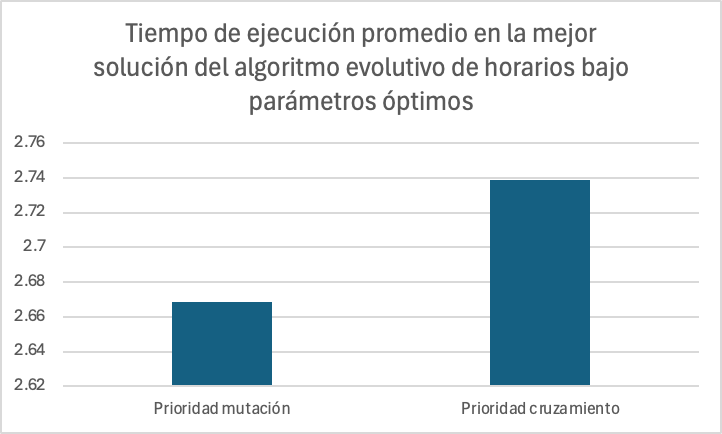


*Gráfica 3. Tiempo vs. errores en rendimiento general.*

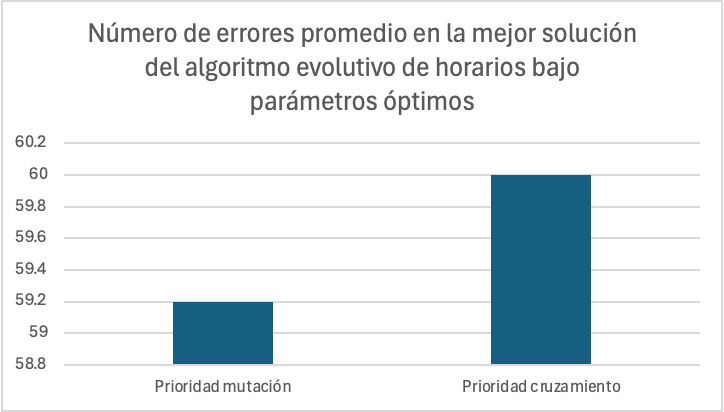
Los resultados promedio reflejaron la tendencia del elemento cuya función de aptitud se priorizaba. Al dar prioridad al tiempo, los mejores parámetros produjeron tiempos de ejecución más cortos para el algoritmo evolutivo de horarios, mientras que al priorizar el número de errores, se obtuvo una mayor calidad en las soluciones. Sin embargo, la disparidad fue más evidente en el caso del número de errores, con una diferencia de 11.2 errores.

En cuanto al tiempo, las diferencias fueron menos significativas, con una mejora de tan solo 0.4 segundos al cambiar la prioridad. Aunque las discrepancias en rendimiento y calidad no fueron considerablemente grandes, sí se observó una diferencia notable en el rendimiento del algoritmo evolutivo en sí. El tiempo promedio de ejecución cuando se priorizó el tiempo (en la ejecución del algoritmo evolutivo de horarios) fue casi tres veces mayor que cuando se priorizó el número de errores. Modificar la función de evaluación no proporcionó un beneficio considerable en términos de calidad de la solución, pero sí resultó en una disminución considerable del rendimiento.

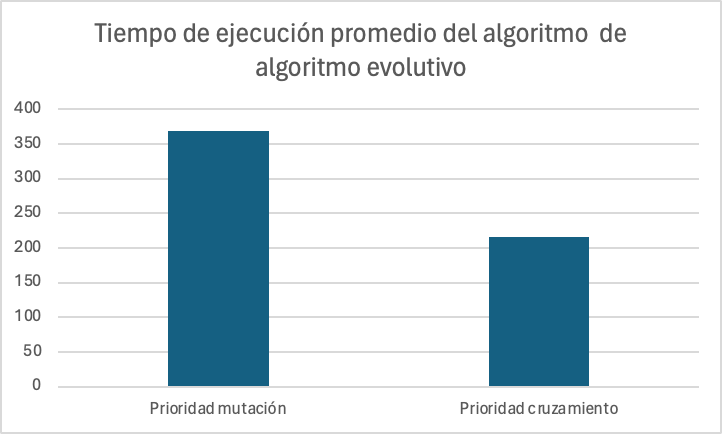
El tercer y cuarto enfoque compartieron función de aptitud y parámetros, excepto por la probabilidad de que se utilizaran los operadores evolutivos. En uno de ellos, la probabilidad de mutación fue del 75%, mientras que en el otro, la probabilidad de cruzamiento fue del 75%.



*Gráfica 4. Mutación vs. cruzamiento en rendimiento horarios.*



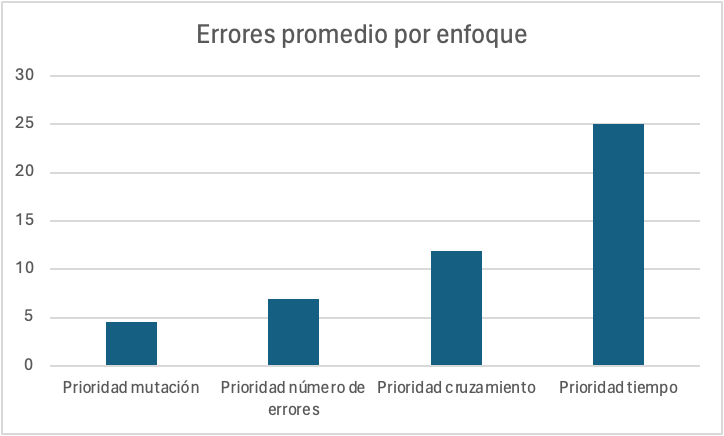
*Gráfica 5. Mutación vs. cruzamiento en errores.*



*Gráfica 6. Mutación vs. cruzamiento en rendimiento general.*

Los resultados fueron muy similares para ambos enfoques en cuanto al rendimiento y calidad de la solución proporcionada por el algoritmo evolutivo de horarios con los parámetros que cada uno encontraba. Sin embargo, el rendimiento del algoritmo evolutivo en sí mostró diferencias entre ambos enfoques. El enfoque que priorizaba la mutación tuvo un tiempo de ejecución promedio mayor que el enfoque que priorizaba el cruzamiento. Una vez más, observamos que los dos enfoques no muestran diferencias significativas en términos de rendimiento y calidad de la solución, pero sí en cuanto al rendimiento del algoritmo evolutivo en sí.

Además de las comparaciones en el rendimiento del algoritmo evolutivo, se pusieron a prueba algunas de las mejores soluciones encontradas por cada enfoque.



*Gráfica 7. Errores promedio por enfoque.*

El enfoque que prioriza la mutación mostró ser el más efectivo, pues consiguió un promedio de 4.55 errores, mientras que priorizar el tiempo demostró ser el peor de todos los enfoques.

### *Conclusiones*

Este proyecto nos ha brindado una comprensión más profunda sobre la importancia de los algoritmos evolutivos en la resolución de problemas complejos, como la programación de horarios de exámenes en instituciones educativas. Estos algoritmos ofrecen una solución a desafíos que tradicionalmente han sido difíciles de abordar en la optimización de la gestión académica, lo que representa un avance significativo en este campo.

Además, utilizar un algoritmo evolutivo para maximizar el rendimiento y la calidad de las soluciones de otro algoritmo evolutivo resultó ser una estrategia efectiva en nuestro estudio. Sin embargo, es importante reconocer que el rendimiento del algoritmo de algoritmos evolutivos puede verse limitado por el tiempo de ejecución, especialmente al trabajar con grandes poblaciones y generaciones. Para su implementación práctica a gran escala, se requeriría una optimización adicional.

A pesar de estas limitaciones, hemos logrado identificar la utilidad de los algoritmos evolutivos en la optimización de la gestión académica y del rendimiento de otros algoritmos evolutivos. También reconocemos la necesidad de mejorar su eficiencia para adaptarlos a contextos donde el tiempo de procesamiento es un factor crítico. Este proyecto representa un paso importante hacia la aplicación más efectiva de algoritmos evolutivos en la optimización de procesos académicos.

### Referencias bibliográficas

Araujo, L., & Cervigón, C. (2009). Algoritmos Evolutivos Un enfoque práctico. México: Alfaomega.

Russell, S., & Norvig, P. (2004). Inteligencia Artificial. Un enfoque moderno (2da ed.). España: Pearson Educación, S.A.