Universidad San Francisco de Quito
Maestría en Inteligencia Artificial
Módulo: Inteligencia artificial
Taller 4

Edwin Montenegro, Alejandra Pinto, Gustavo Recalde, Galo Travez

16 de Marzo del 2024

Índice

1. USO DE APRENDIZAJE NO SUPERVISADO

El aprendizaje no supervisado es una rama fundamental en el campo de la inteligencia artificial y el análisis de datos. Se enfoca en identificar patrones y estructuras ocultas en conjuntos de datos sin la necesidad de etiquetas o supervisión externa. Esto lo hace invaluable en problemas de identificación de patrones, donde se busca comprender la estructura subyacente de los datos y descubrir relaciones inherentes entre las variables.

En el repositorio de talleres `https://github.com/Borreguin/WorkShop1-USFQ` se ha colocado un set de datos que contiene 4 variables:

Cuadro 1: Condiciones iniciales

Tagname	**Alias**	**Descripción**
V005_vent01_CO2	CO2 Ventilation NE	Cantidad de CO2 en ppm - Salida NE
V022_vent02_CO2	CO2 Ventilation SW	Cantidad de CO2 en ppm - Salida SO
V006_vent01_temp_out	Temp. Vent. NE Out	Temperatura en °C - NE
V023_vent02_temp_out	Temp. Vent. SW Out	Temperatura en °C - SO

Cuadro 2: Variables en el set de datos.

El set de datos contiene las mediciones de concentración de CO2 y de temperatura del sistema de ventilación de un edificio inteligente. Se considerará como patrón diario para este ejercicio, aquel que pertenece al día completo, es decir, desde 00:00 hasta 23:59 del día.

Hidden Markov Model es un modelo estadístico que se utiliza para modelar un proceso estocástico (oculto) a través de las observaciones de las muestras de salidas del proceso. Este modelo es muy útil para identificar secuencias, agruparlas consecuentemente con sus correspondientes estados ocultos.

1.1. Encontrar patrones uni-variable

Seleccionar 2 variables del cuadro anterior, y utilizando el Modelo Oculto de Markov, realizar agrupamiento de observaciones usando la librería hmmlearn. Para ello, a la variable elegida, desglosarla en observaciones de 24h cada una, es decir cada muestra contiene 24 valores del día, formando de esta manera una secuencia temporal de muestras.

En este ejercicio se desea resolver la pregunta: Dado un modelo HMM y una secuencia de observaciones, ¿cuál es la secuencia más probable de estados ocultos que generaron estas observaciones? Es decir, asignar una etiqueta de estado oculto a cada observación ingresada al modelo. El problema por resolver es conocido como "Decoding Problem"de HMM.
Sugerencia, para encontrar el número de componentes adecuado (estados ocultos), realizar el agrupamiento con $n_{comp} = n_{min}$ hasta $n_{comp} = n_{max}$, y mediante la función de verosimilitud encontrar el mejor modelo HMM que ajusta de mejor manera las observaciones, es decir, encontrar $n_{comp} = n_{best}$, donde n_{best} está en el rango: $[n_{min}, n_{max}]$.

1.2. Desarrollo

El código Python es una combinación de preparación de datos, visualización y análisis utilizando Modelos Ocultos de Markov y agrupamiento K-Means, aplicado a datos ambientales como los niveles de CO2 y temperatura. A continuación, se explica paso a paso el proceso realizado:

Preparación de Datos: Se leen los datos de un archivo CSV, indexados por una marca de tiempo, y se imprimen los tipos de datos. Esto está encapsulado en la función `prepare_data()`.

Visualización de Datos: Se trazan dos tipos de datos ambientales (niveles de CO2 y temperatura).Se utiliza la función `plot_data()`, que incluye etiquetas y una leyenda para claridad.

HMM para Análisis Univariado: Se crea un ejemplo simple de HMM con `n_components` representando el número de estados ocultos y `n_features` para las dimensiones de los datos. El HMM se entrena con datos sintéticos y se utiliza para decodificar una nueva secuencia de observaciones, clasificándolas potencialmente en diferentes estados ocultos. Los estados y otros parámetros del modelo se imprimen en la consola.

Visualización de Estados Decodificados a lo Largo del Tiempo: Se traza una secuencia decodificada de ejemplo contra el tiempo para visualizar los cambios en los estados ocultos.

Análisis de Datos con HMM: Para cada variable seleccionada, se crea y entrena un HMM separado. Se prueba el rango de $n_components para encontrar el mejor ajuste para los datos$.

Agrupamiento K-Means: Se generan datos de muestra, se ajusta un modelo K-Means y se visualizan los centros de los clústeres.

Visualización de Probabilidades de Estado: Después de entrenar el HMM, se visualiza la probabilidad de cada estado oculto para cada observación en un subtrazado para cada estado.

Visualización de Secuencias: Se trazan las secuencias originales de observaciones para visualizar los datos reales. También se trazan los estados ocultos determinados por el HMM para comparación.

1.3. Resultados

Figura 1:Niveles de CO2 a lo largo del tiempo: El gráfico muestra los niveles de concentración de CO2 de dos diferentes fuentes de ventilación trazados a lo largo del tiempo. Las fluctuaciones regulares sugieren un componente periódico en los cambios, posiblemente relacionado con ciclos de actividad humana diaria.

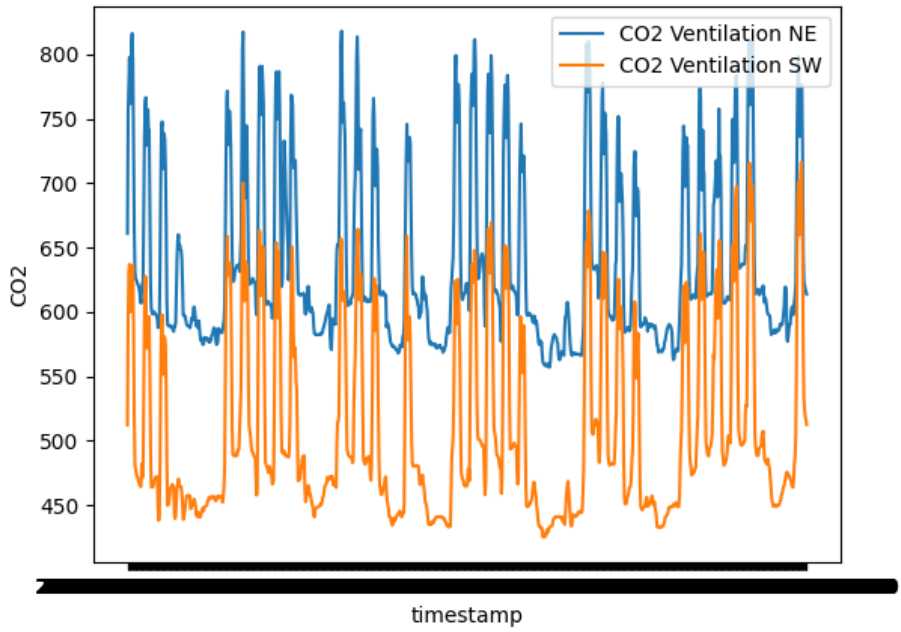

Figura 1: Niveles de CO2 a lo largo del tiempo

Figura 2:Temperatura a lo largo del tiempo: El gráfico muestra las lecturas de temperatura de las mismas dos fuentes de ventilación a lo largo del tiempo. Similar a los niveles de CO2, hay un patrón periódico, potencialmente correlacionando con ciclos diarios de temperatura.

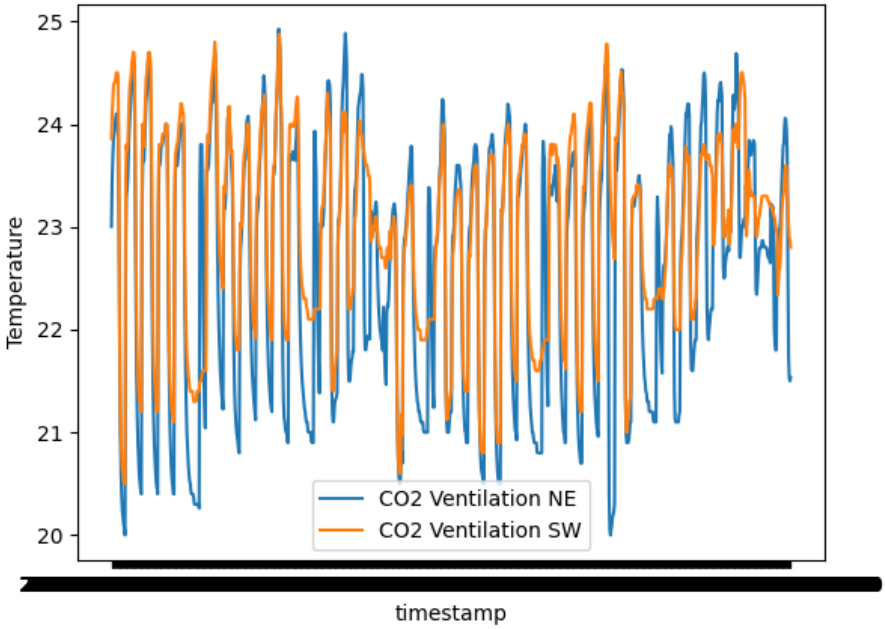

Figura 2: Temperatura a lo largo del tiempo

Figura 3:Secuencia de Estados Decodificados: El gráfico muestra la secuencia de estados decodificados a lo largo del tiempo, lo que puede representar diferentes regímenes o comportamientos en el sistema, como periodos de 'alta actividad' versus 'baja actividad'.

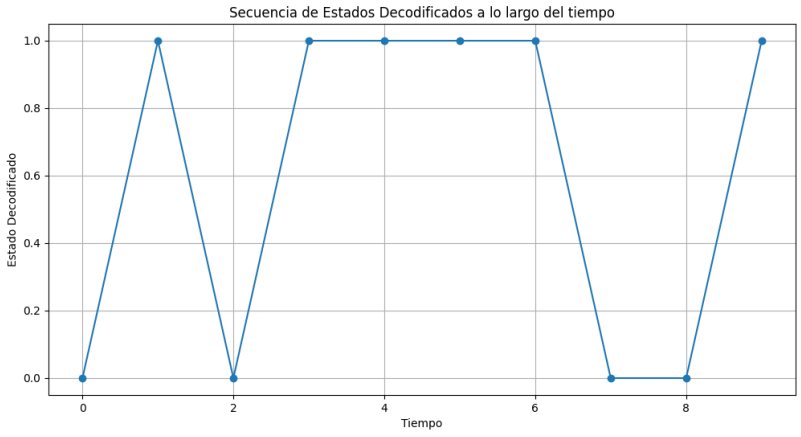

Figura 3: Secuencia de Estados Decodificados

Figura 4:Valores Diarios de CO2 Superpuestos: La figura muestra la variabilidad de los niveles de CO2 a lo largo de diferentes días, superponiendo patrones diarios para resaltar discrepancias y similitudes.

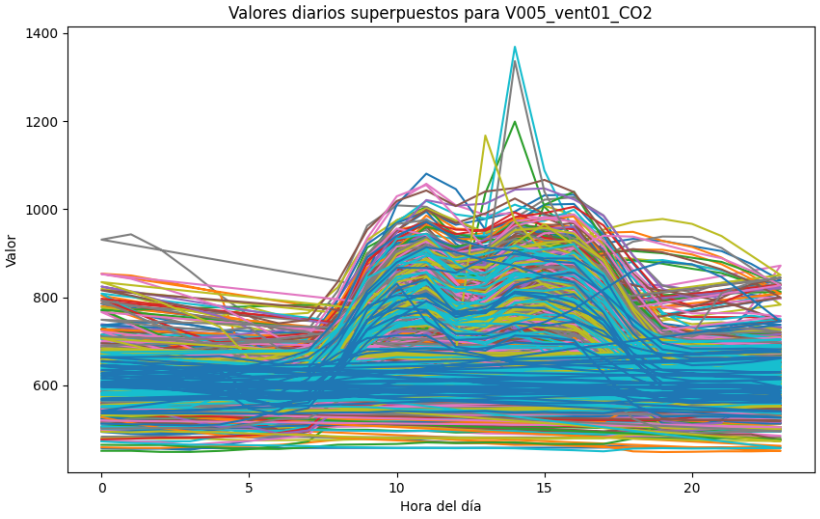

Figura 4: Valores Diarios de CO2 Superpuestos

Figura 5:Valores Diarios de Temperatura Superpuestos: La figura muestra patrones diarios de temperatura, indicando más variabilidad en comparación con los niveles de CO2, lo que sugiere que la temperatura puede estar influenciada por factores externos adicionales.

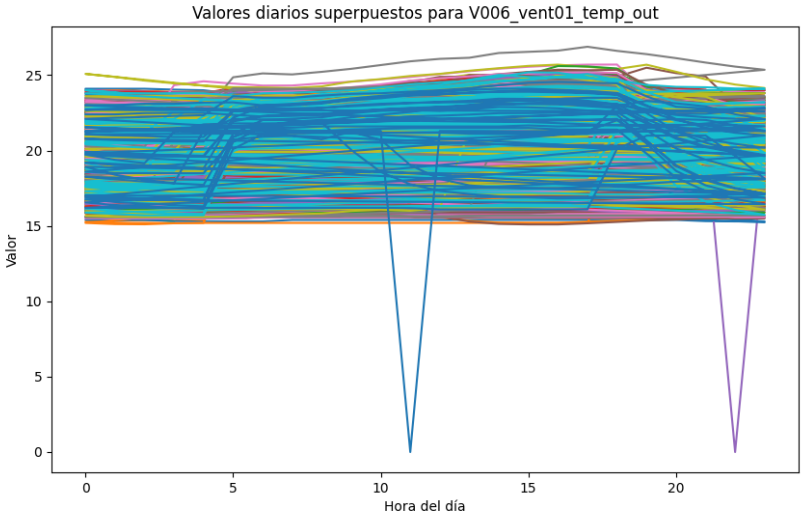

Figura 5: Valores Diarios de Temperatura Superpuestos

Figura 6:Resultado de Agrupamiento K-Means: El diagrama de dispersión demuestra la agrupación de puntos de datos en cuatro clústeres, con los centros de los clústeres destacados, sugiriendo agrupaciones naturales dentro de los datos.

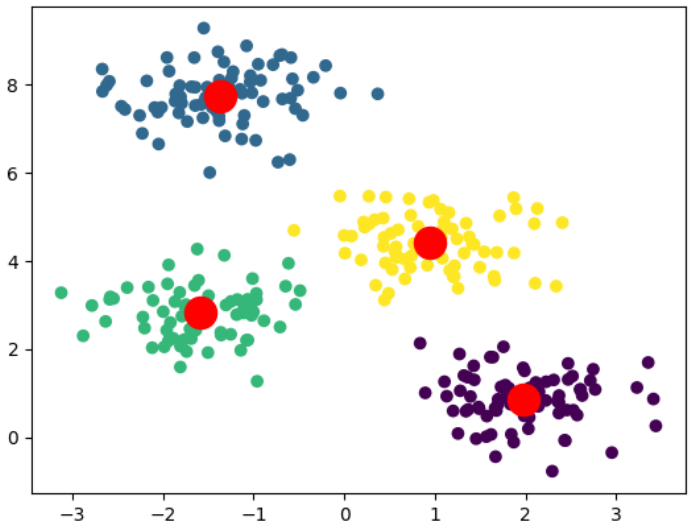

Figura 6: Resultado de Agrupamiento K-Means

Figura 7:Probabilidades de Estados Ocultos: El conjunto de gráficos muestra las probabilidades de cada estado oculto para las observaciones, indicando la confianza del modelo en su asignación de estado a lo largo del tiempo.

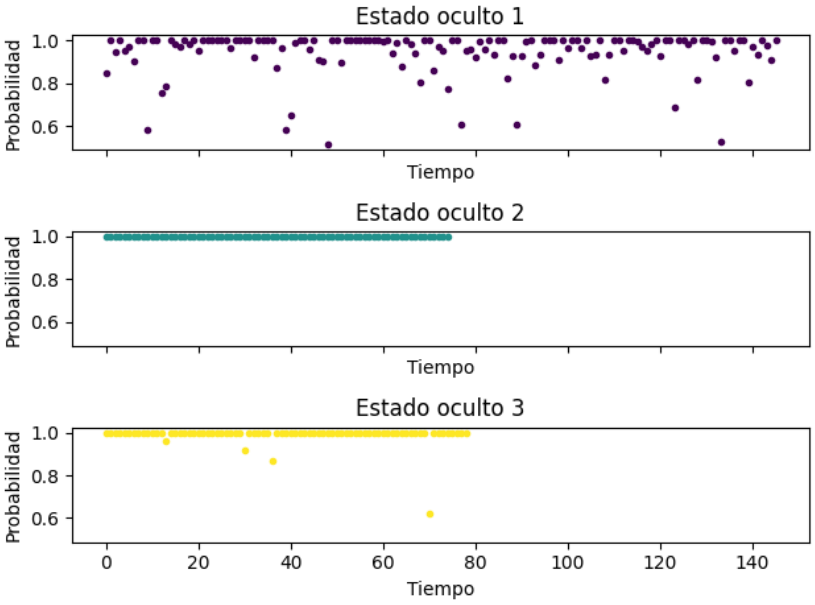

Figura 7: Probabilidades de Estados Ocultos

Figura 8:Secuencia de Observación para la Variable CO2: Muestra las observaciones reales para una de las variables a lo largo de un día, proporcionando una visión del dato bruto que el modelo está interpretando.

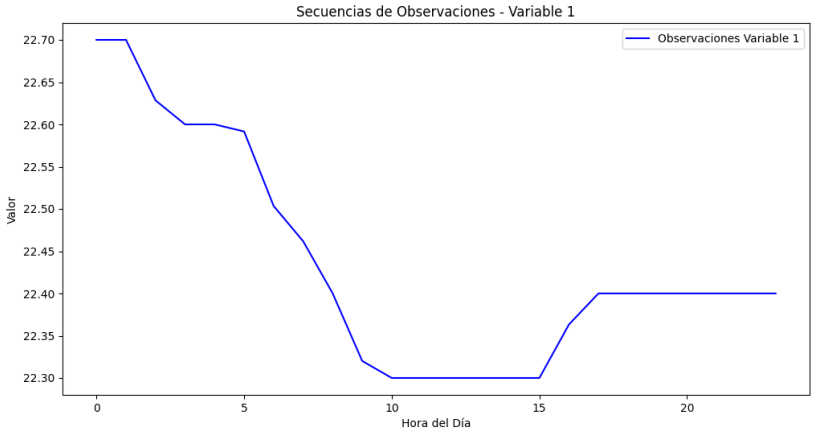

Figura 8: Secuencia de Observación para la Variable CO2

Figura 9:Estados Ocultos para la Variable CO2 Muestra los estados ocultos asignados para cada observación, que pueden interpretarse como diferentes condiciones o comportamientos detectados por el modelo.

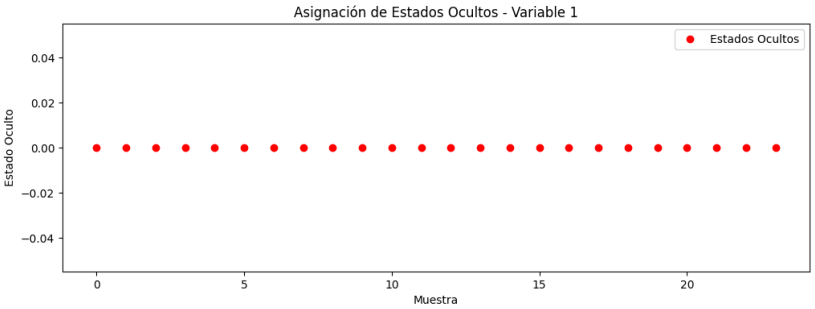

Figura 9: Estados Ocultos para la Variable CO2

1.4. Conclusiones

- La periodicidad de los datos de CO2 y temperatura muestra fuertes patrones diarios, probablemente impulsados por los ciclos naturales y la actividad humana.
- Los modelos ocultos de Markov pueden decodificar secuencias observables en estados latentes, revelando la dinámica subyacente del sistema.
- La variabilidad en los patrones diarios superpuestos tanto para el CO2 como para la temperatura indica que los factores externos pueden influir en estas lecturas más allá de los ciclos predecibles.
- La agrupación K-Means puede identificar con éxito distintas agrupaciones dentro de datos multi-dimensionales, proporcionando una visualización de la segmentación de datos.
- Los HMM no solo identifican estados, sino que también brindan una comprensión probabilística de las asignaciones de estados, lo que agrega profundidad al análisis.
- La visualización es clave para comprender el comportamiento de conjuntos de datos complejos y el rendimiento de los modelos analíticos.

En comparación al taller 3 problema 1 (de agrupamiento), ¿se observa una diferencia en la manera que k-means agrupa en comparación al agrupamiento realizado por HMM? ¿Ha logrado HMM un mejor agrupamiento que k-means?

- HMM se utiliza para identificar estados ocultos basados en secuencias de datos observados, mientras que K-means se aplica para agrupar los datos en grupos basados en sus características.
- K-means Clustering: Divide los datos en K clústeres distintos en función de la media de los puntos de datos de cada clúster. Este método no está supervisado y requiere especificar el número de clústeres de antemano. Funciona bien con grandes conjuntos de datos y es eficaz, pero asume que los clústeres son esféricos y de igual tamaño, lo que no siempre es el caso.
- Modelo oculto de Markov (HMM): Este es un modelo estadístico que asume que un proceso subyacente es un proceso de Markov con estados ocultos. HMM es especialmente conocido por su aplicación en el reconocimiento de patrones temporales, como el habla, la escritura a mano, el reconocimiento de gestos, el etiquetado de partes de la oración, el seguimiento de partituras musicales, las descargas parciales y la bioinformática.
- Para determinar si HMM ha logrado un mejor agrupamiento que K-means, sería necesario evaluar los resultados en función de los objetivos específicos del análisis. Si el componente temporal y las transiciones de estado son importantes para la agrupación en clústeres, HMM podría proporcionar un mejor contexto y, por lo tanto, considerarse mejor. Sin embargo, si el objetivo es simplemente dividir el espacio en función de la similitud y el componente temporal no es tan crítico, K-means podría ser suficiente.

2. ALGORITMOS GENÉTICOS

El taller contiene el código para la ejecución de un Algoritmo Genético, el objetivo de este algoritmo es llegar a generar la frase objetivo: "GA Workshop! USFQ" que tiene 17 letras. El proceso de generación lo realizará a partir de una población de 100 frases aleatorias (individuos) de tamaño de 17 letras. Las poblaciones generadas deberán interactuar cada una hasta llegar a producir la frase objetivo. Una vez alcanzado un individuo que sea igual a la frase objetivo, el algoritmo genético se detendrá.

2.1. Posibles resultados de ejecución:

1. El algoritmo llega a producir el individuo deseado en el número de interacciones que se le ha definido, en este caso, el algoritmo ha convergido a una solución óptima.
2. El algoritmo no logra producir el individuo deseado en el número de interacciones que se le ha definido, en este caso ocurren dos situaciones: a) el algoritmo puede que esté cerca del objetivo y le falta más iteraciones para llegar a una solución óptima, b) el algoritmo nunca puede converger y en lugar de acercarse al objetivo se aleja.

2.2. Casos de estudio:

- **Caso de estudio 1:** DEFAULT (cuenta número de coincidencias para su función de evaluación)
- **Caso de estudio 2:** DISTANCE (encuentra la diferencia entre el individuo objetivo y el mejor individuo de cada generación)

2.3. Estructura del proyecto:

- AlgoritmosGeneticos/GA.py: Programa Principal
- AlgoritmosGeneticos/constants.py: Constantes del proyecto
- AlgoritmosGeneticos/generalSteps.py: Pasos Generales del Algoritmo Genético
- AlgoritmosGeneticos/operation.py: Operaciones dentro de los Pasos Generales
- AlgoritmosGeneticos/util.py: Funciones de ayuda

Importante: Durante las operaciones del AG, no es permitido "tomar", "copiar", "extraer" información del individuo objetivo, lo que se puede realizar es "leer", "comparar" información del objetivo a fin de comparar el mejor candidato.

2.4. Ejecución de los casos:

Para ejecutar los casos, en el archivo `AlgoritmosGeneticos/GA.py` ejecutar la función `main`.

2.5. Ejercicio Propuesto

1. **Ejecute los dos casos de estudio y explique los resultados de ejecución de cada caso de estudio.**

   Resultados caso 1: Objetivo alcanzado: Generación 982: GA Workshop! USFQ - Aptitud: 17

   Resultados Caso 2: Objetivo no alcanzado en las iteraciones establecidas

   En el primer caso de estudio, la configuración por defecto utiliza una medida de aptitud basada en la coincidencia directa de caracteres entre el individuo y el objetivo. La "aptitud" aquí es el número de caracteres correctos en las posiciones correctas. La convergencia del algoritmo nos indica que el algoritmo es capaz de explorar eficazmente el espacio de soluciones y acumular mejoras incrementales que finalmente llevan a la solución exacta. La aptitud final de 17 indica una coincidencia perfecta con el objetivo, donde cada carácter en la posición correcta suma a la aptitud total.

   En el segundo caso, se cambia la configuración para utilizar la distancia como la medida de aptitud, donde una aptitud más baja es mejor. Además, se selecciona el mejor individuo basado en la menor distancia (aptitud), y se generan nuevas poblaciones con un enfoque que también considera la distancia.

   La falta de convergencia y la aptitud negativa en la última generación sugieren que el método de evaluación y selección utilizado en este caso no es el adecuado para guiar el algoritmo hacia la solución de manera eficiente. La aptitud negativa, en particular, indica que el cálculo de la distancia produce valores que no se alinean bien con una mejora progresiva hacia el objetivo.

2. **¿Cuál sería una posible explicación para que el caso 2 no finalice como lo hace el caso 1? Revisar el archivo `util.py` función `distance`.**

   La distancia como medida de aptitud es más difícil de optimizar, porque los cambios aleatorios tienen igual probabilidad de aumentar o disminuir la distancia. Es posible que las mutaciones y cruces no siempre produzcan cambios que mejoren la aptitud, haciendo más difícil encontrar el camino hacia el objetivo. Al centrarse en minimizar la distancia, el algoritmo puede favorecer cambios que, aunque acerquen en términos de distancia global, no necesariamente produzcan una mejor aproximación visual o estructural al objetivo.

3. **Realice una correcta implementación para obtener la distancia/diferencia correcta entre dos individuos en el archivo `util.py` función `distance`.**

   La corrección de la implementación sería modificar el código de distance de la siguiente manera:

```python
def distance(list1: List[int], list2: List[int]):
acc = 0
for e1, e2 in zip(list1, list2):
    acc += abs(e1 - e2)  # Uso de valor absoluto para la diferencia de cada par
diff_len = abs(len(list1) - len(list2))  # Diferencia absoluta de longitud
acc += diff_len  # Considerar un costo por la diferencia de longitud
if len(list1) == 0 and len(list2) == 0:
    return None
return acc
```

Listing 1: Código Experimento 1

4. **¿Sin alterar el parámetro de mutación `mutation_rate`, se puede implementar algo para mejorar la convergencia y que esta sea más rápida? Implemente cualquier mejora que permita una rápida convergencia. Pista: ¿Tal vez elegir de manera diferente los padres? ¿Realizar otro tipo de mutación o cruce?**

Para mejorar la convergencia y que esta sea más rápida, se podría considerar ajustes en cómo se seleccionan los padres, así como en los métodos de cruce y mutación Un ajuste podría ser implementar una selección de padres que favorezca a los individuos con mejor aptitud pero que también mantenga la diversidad genética para evitar la convergencia prematura. Una estrategia equilibrada es la Selección por torneo o Ranking, donde los individuos son ordenados por su aptitud y la probabilidad de ser seleccionados se asigna según su ranking. Esto asegura que los individuos más aptos tengan una mayor probabilidad de ser elegidos, pero los menos aptos también tienen una oportunidad, manteniendo la diversidad. Por otro lado también se puede probar una mejora en el Método de Cruzamiento, como alternativa se puede usarel cruzamiento de Dos Puntos que es una variante que puede generar descendencia con una mayor variabilidad genética en comparación con el cruzamiento de un solo punto. En este método, se eligen dos puntos de cruzamiento aleatorios en el genoma, intercambiando los segmentos intermedios entre los dos padres para crear la descendencia. Para imlementarlo, realizamos los siguientes cambios en el codigo:

una funcion de seleccion por ranking: *def rank_selection(population, aptitudes)* dentro del archivo 'operation.py'

```
    def rank_selection(population, aptitudes):
      ranked_population = [x for _, x in sorted(zip(aptitudes, population))]
      selection_probabilities = [((rank+1)/len(population)) for rank in range(len(population))]
      parents = random.choices(ranked_population, weights=selection_probabilities, k=2)
      return parents

```

Listing 2: Código Seleccion por rating

y la función de mejora en el cruzamiento:

```
    def two_point_crossover(parent1, parent2):
      if len(parent1) < 3:
      return parent1, parent2
      crossover_point1 = random.randint(1, len(parent1)//2)
      crossover_point2 = random.randint(crossover_point1+1, len(parent1)-1)
      child1 = parent1[:crossover_point1] + parent2[crossover_point1:crossover_point2] + parent1[crossover_point2:]
      child2 = parent2[:crossover_point1] + parent1[crossover_point1:crossover_point2] + parent2[crossover_point2:]
      return child1, child2

```

Listing 3: Código Cruce por dos puntos

Estas mejoras creímos que deberían aumentar la efectividad del algoritmo genético, permitiendo una convergencia más rápida, sin embrago durante la ejecucion, no se notaron mejoras. Las mejoras en la selección de padres y el método de cruzamiento pueden no haber sido significativamente diferentes en términos de generar diversidad genética o dirigir la población hacia el óptimo global. La selección por ranking y el cruzamiento de dos puntos probablemente no introducen suficiente variabilidad o no guían efectivamente la búsqueda, por lo que el impacto en la convergencia es mínimo.

5. **Cree un nuevo caso de estudio 3. Altere el parámetro de mutación `mutation_rate`, ¿ha beneficiado en algo la convergencia? Qué valores son los más adecuados para este parámetro. ¿Qué conclusión se puede obtener de este cambio?**

   Creamos el caso 3:

```
  def case_study_3(_objetive):
    population = generate_population(100, len(_objetive))
    n_iterations = 1000
    ga = GA(population, _objetive, mutation_rate=0.01, n_iterations)  # Comenzamos con 0.01 como tasa base

    # Establecemos las configuraciones optimizadas basadas en las recomendaciones
    ga.set_evaluation_type(AptitudeType.BY_DISTANCE)
    ga.set_best_individual_selection_type(BestIndividualSelectionType.MIN_DISTANCE)
    ga.set_new_generation_type(NewGenerationType.MIN_DISTANCE)
```

```
10
11      # Ejecutamos el GA con diferentes tasas de mutacion
12      for mutation_rate in [0.001, 0.01, 0.05, 0.1, 0.2]:
13        print(f"Probando con tasa de mutacion: {mutation_rate}")
14        ga.mutation_rate = mutation_rate
15        ga.run()
16
```

Listing 4: Código Caso 3

Resultados: Probando con tasa de mutación: 0.001 Objetivo no alcanzado en las iteraciones establecidas 1000 Probando con tasa de mutación: 0.01 Objetivo alcanzado: Generación 378: GA Workshop! USFQ - Aptitud: 0 Probando con tasa de mutación: 0.05 Objetivo alcanzado: Generación 95: GA Workshop! USFQ - Aptitud: 0 Probando con tasa de mutación: 0.1 Objetivo alcanzado: Generación 56: GA Workshop! USFQ - Aptitud: 0 Probando con tasa de mutación: 0.2 Objetivo no alcanzado en las iteraciones establecidas 1000

Tasas de Mutación Bajas (0.001): No proporcionan suficiente diversidad genética, lo que puede resultar en una incapacidad para alcanzar el objetivo dentro del límite de iteraciones.

Tasa Óptima de Mutación (0.01 - 0.1): Facilita una exploración y explotación eficientes del espacio de búsqueda, logrando el objetivo de manera efectiva. La tasa de mutación de 0.1 resultó en la convergencia más rápida hacia la solución, alcanzando el objetivo en 56 generaciones.

Tasas de Mutación Altas (0.2): Introducen demasiada variabilidad, probablemente destruyendo combinaciones genéticas beneficiosas y previniendo la convergencia dentro del número establecido de iteraciones.

podemos concluir que existe una relación entre la tasa de mutación y la eficacia del algoritmo genético, donde una tasa de mutación óptima promueve una convergencia rápida y eficiente, mientras que tasas demasiado bajas o altas pueden ser contraproducentes. La iteración y ajuste de parámetros son muy importante para encontrar el equilibrio adecuado para cada problema específico.

6. **Cree un nuevo caso de estudio 4. Altere el tamaño de la población, ¿es beneficioso o no aumentar la población?**

```
1     def case_study_4(_objetive):
2       mutation_rate = 0.01  # Mantenemos una tasa de mutacion constante para esta
      prueba
3       n_iterations = 1000
4       population_sizes = [50, 100, 200, 500, 1000]  # Diferentes tamanos de
      poblacion para probar
5
6       for size in population_sizes:
7         print(f"Probando con tamano de poblacion: {size}")
8         population = generate_population(size, len(_objetive))
9         ga = GA(population, _objetive, mutation_rate, n_iterations)
10
11
```

Listing 5: Código Caso 5

Resultados: Conclusion: La variación con diferentes tamaños de población demuestran que aumentar el tamaño de la población puede mejorar significativamente la rapidez con la que un algoritmo genético converge hacia una solución óptima, gracias a una mayor diversidad genética que facilita tanto la exploración como la explotación del espacio de búsqueda. Sin embargo, es importante encontrar un equilibrio adecuado que considere tanto la efectividad del algoritmo como las limitaciones computacionales.

7. **De todo lo aprendido, cree el caso de estudio definitivo (caso de estudio 5) el cual tiene lo mejor de los ítems 4, 5, 6.**

   Combinando todo lo aprendido, generando mejoraas en al seleccion de padres, cambiando el método de cruzamiento y realizando varaiciones en los paramteros de mutacion y población, escogiendo los mas optimos de las pruebas, generamos el caso de estudio 5 con el siguiente codigo y el resultado, como era de esperarse fue mejor:

```
    def case_study_5(_objetive):
      population = generate_population(1000, len(_objetive))
      mutation_rate = 0.1 #mejor tasa de mutacion probada
      n_iterations = 1000 # mejor tamano de poblacion probado
      ga = GA(population, _objetive, mutation_rate, n_iterations)
      ga.set_evaluation_type(AptitudeType.BY_DISTANCE)
      ga.set_best_individual_selection_type(BestIndividualSelectionType.
MIN_DISTANCE)
      ga.set_new_generation_type(NewGenerationType.MIN_DISTANCE)
      # Aplicando nuevos tipos de seleccion de padres y de cruzamiento
      ga.set_parent_selection_type(ParentSelectionType.RANK)
      ga.set_crossover_type(CrossoverType.TWO_POINT)
      ga.run()


```

Listing 6: Código Caso 4

Resultados: Objetivo alcanzado: Generación 21: GA Workshop! USFQ - Aptitud: 0

2.6. Conclusiones

- Este caso nos muestra la importanciade una adecuada optimización de parámetros y la selección de estrategias avanzadas en el rendimiento de los algoritmos genéticos, logrando una notable eficiencia al alcanzar la solución óptima en solo 21 generaciones. REsumiendo los puntos clave empleados podemos concluir:
- Optimización de Parámetros: La combinación óptima de un gran tamaño de población (1000) y una tasa de mutación elevada (0.1) geeró una exploración efectiva del espacio de búsqueda y una rápida convergencia hacia la solución.
- Estrategias de Selección y Cruzamiento: La implementación de la selección de padres por ranking y el cruzamiento de dos puntos demostró ser efectiva con estos parámetros, facilitando una explotación eficiente de las soluciones y manteniendo la diversidad genética necesaria para evitar óptimos locales.
- Importancia del Ajuste Fino: Este estudio resalta el valor del ajuste fino y la experimentación con diferentes configuraciones del algoritmo genético, adaptándolas específicamente al problema en cuestión para optimizar el rendimiento.
- Enfoque Iterativo: El resultado demuestra la necesidad de un proceso iterativo de experimentación para identificar la mejor configuración de parámetros y estrategias, demostrando que la optimización de algoritmos genéticos es altamente dependiente del contexto del problema.
- La conclusión es que la optimización cuidadosa de los parámetros, junto con la elección de estrategias avanzadas para la selección de padres y el cruzamiento, puede mejorar significativamente la eficacia de los algoritmos genéticos, permitiendo alcanzar soluciones óptimas de manera más rápida y eficiente.

Referencias

[1] F. Berzal, "Algoritmos Genéticos," in *G2 Genetic Algorithms*, [Online]. Available: file:///path/to/G2%20Genetic%20Algorithms.pdf. Accedido el 18 de Marzo de 2024.

[2] C. A. Coello Coello, "Introducción a la computación evolutiva," in *INtroduccion a la computacion evolutiva_apuntes_Coello, Carlos*, [Online]. Available: file:///path/to/Introduccion%20a%20la%20computacion%20evolutiva_apuntes_Coello,%20Carlos.pdf. Accedido el 18 de Marzo de 2024.