Autor/Autores: Sandra Atencio, Piero Tejada, Manuel Azpilcueta y Danilo Sanchez

Resumen

El trabajo abordó el problema de crear un cronograma semanal para 7 cursos, asignándoles horarios, aulas y profesores bajo restricciones obligatorias y opcionales. El objetivo fue generar un cronograma óptimo utilizando un algoritmo genético. La solución implicó el uso de operadores de cruce y mutación optimizados, logrando alcanzar el máximo valor de fitness (valor de 1,841) en solo 6 generaciones. El enfoque resultó efectivo y eficiente, cumpliendo con todas las restricciones del problema y demostrando que los algoritmos genéticos son adecuados para resolver problemas complejos de asignación de recursos.

1. Introducción

1.1 Descripción del problema:

El problema consiste en elaborar un **Cronograma Semanal para 7 cursos**, asignándole a cada uno un *horario, aula y profesor* que cumpla con restricciones obligatorias (i.e. no exceder la capacidad del aula o evitar conflictos de horarios entre profesores y aulas) y opcionales (i.e. tener en cuenta el horario preferido del profesor o rangos de horarios de no dictado por celebrarse eventos culturales).

1.2 Hipótesis del problema

Mediante el algoritmo genético se elaborará un **cronograma semanal para 7 cursos con asignaciones optimas de sus horarios, aulas y profesores**. Es decir, a través de la evolución de posibles soluciones producto de mutaciones y cruzamientos, el algoritmo detectará al mejor cronograma semanal tomando como base el valor de Fitness que suma (recompensa) decisiones correctas de asignación de horario, aula, profesor y de resta (penalización) decisiones incorrectas de asignación de horario, aula, profesor. Siendo el objetivo principal del algoritmo genético alcanzar el valor máximo de Fitness posible que ha sido calculado en 1,841 (= 7 cursos * 263 puntos máximos por asignaciones correctas).

2. Metodología

2.1 Descripción de la representación de estados/individuos elegida

Un individuo (alternativa de cronograma para los 7 cursos) se representa mediante un **cromosoma** conformado por 7 genes. Cada gen corresponde a un curso el cual contiene información sobre la asignación a ese curso de un horario, aula y profesor específico. En resumen, un cromosoma representa una posible solución al problema de asignación de horarios, aulas y profesores a cada uno de los 7 cursos.

La mejor solución se define a través de la función de Fitness, que evalúa qué tan bien cumple un individuo con las restricciones del problema que contiene penalizaciones (puntajes negativos) y recompensas (puntajes positivos).

2.2 Descripción del comportamiento entrada/salida del (los) enfoques.

El enfoque será el **Algoritmo Genético** como método para resolver el problema de construcción del cronograma semanal para 7 cursos.

2.2.1 Datos de Entrada:

- **Datos del problema (ver anexo 1):**

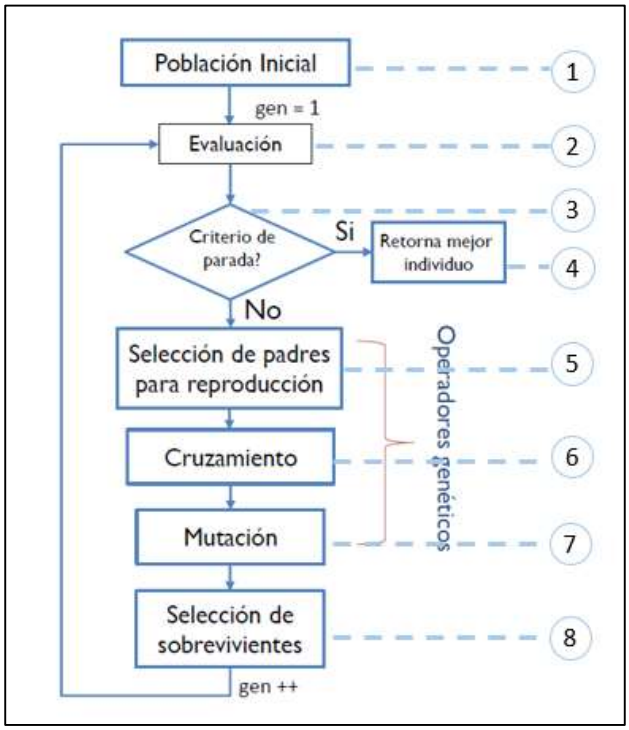
  - lista de 7 cursos, su descripción, alumnos inscritos, profesores que lo pueden dictar
  - lista de profesores y horarios de preferencia
  - lista de aulas y sus máximas capacidades
  - lista de horario de dictado de clases durante la semana
  - detalle de restricciones obligatorias y opcionales
- **Parámetros del algoritmo genético (ver 3.2.2):**
  - tamaño de la población inicial
  - número de generaciones
  - tasa de mutación
  - métodos de selección de padres y sobrevivientes
  - métodos de cruzamiento y de mutación

2.2.2 Datos de Salida (en pantalla)

- **Mejores Fitnesses**: muestra el valor total (suma de recompensa resta de penalización) de cada individuo. Finalizada cada generación, se muestra el Mejor Fitness (mayor) de entre todos los fitness asociados a todos los individuos de la generación relacionada.
- **Mejor solución:** después de ejecutarse las generaciones definidas, el algoritmo devuelve el mejor cromosoma encontrado (**mejor cronograma**) que representa la asignación óptima de horarios, aulas, y profesores a los 7 cursos
- **Valor de fitness de mejor solución:** El algoritmo también proporciona el valor de Fitness asociado con la mejor solución.
- **Grafica de Fitness:** se grafica el valor de fitness optimo en cada generación definida.

2.3 Descripción de operadores/funciones desarrolladas para solucionar el problema

A continuación, se muestra el flujo seguido por nuestro código con precisiones particulares en base al cas (i.e operadores)



Fuente : Material de Clase

1. **Población Inicial:**
   Inicia con una población aleatoria de 50 individuos
2. **Evaluación:**
   Evalúa los Fitness de los individuos y obtiene el Fitness máximo
3. **Criterio de parada:**
   Se ha definido como criterio de parada "Número máximo de generaciones", en caso que el Fitness llegue a 1841 deberá pasar 2

generaciones más para que el sistema pare.

4. **Retorno mejor individuo**
   En caso de encontrar el mejor Fitness= 1841 devuelve valores y finaliza sistema

5. **Selección de padres para reproducción:**
   De forma aleatoria se ha definido la selección de padres por el método "ruleta"

6. **Cruzamiento:**
   Se experimentó Onepoint y Uniform, y en este caso se obtuvo un mejor Fitness con un rápido resultado con "Onepoint" (Ver anexo)

7. **Mutación:**
   De la misma forma que en el punto 6 se obtuvo como resultado el tipo de mutación= 'MultiFlip' con una tasa de mutación= 5% (ver anexo)

8. **Selección de sobrevivientes:**
   Se elige a los sobrevivientes mediante el método de Ranking, en este caso a los 50 primeros fitness máximos.

2.4 Funciones Claves

Entre el grupo de funciones que conforman el algoritmo genético, resaltan 2 por su impacto en el cumplimiento del objetivo.

2.4.1 Función Fitness

def get_fitness (individuo, verbose=False)

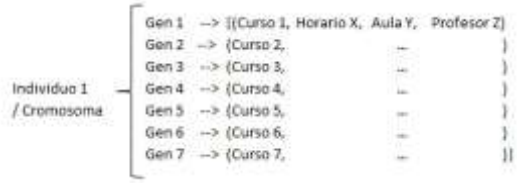
La función de Fitness evalúa la calidad de cada solución en un algoritmo genético, asignando un valor (Fitness) que refleja su desempeño respecto al problema. Esta función guía la selección de soluciones para la reproducción, buscando maximizar la eficiencia y optimización a lo largo de generaciones. El Fitness se calcula considerando las siguientes penalizaciones y recompensas:

1. Restricción de Aula en el tiempo: penalización = -101 y recompensa = 51
2. Restricción de Profesor en el tiempo: penalización = -102 y recompensa = 52
3. Restricción de Capacidad de aula vs. alumnos inscritos: penalización = -103 y recompensa = 53
4. Restricción de Duplicación de dictado de curso en la semana: penalización = -10000 y recompensa = 107

2.4.2 Función Generar Individuo

def generar_individuo_variedad (cursos, horarios, aulas, profesores)

Esta función está hecha para generar un **individuo**, donde cada individuo representa un posible cronograma de clases. De esta manera, el objetivo de esta función es asignar un horario, un aula y un profesor a cada curso de manera que se maximice la diversidad en estas asignaciones, evitando incurrir en las restricciones definidas en la función Fitness. En general, la configuración tiene la siguiente estructura:



Fuente: Elaboración Propia en base al caso

Donde:

X = código de horario seleccionado para el curso C00x. Ver anexo 1
Y= código de aula seleccionada para el curso C00x. Ver anexo 1
Z| = código de profesor seleccionado para el curso C00x. Ver anexo 1

3.Experimentación y Resultados

3.1 Setup experimental:

3.1.1 Descripción de datos usados

Como se detalló en punto 2.2.1, los datos usados fueron proporcionado como parte de la descripción del problema. Como parte del desarrollo del algoritmo, se definieron datos adicionales relevantes: tamaño de población, numero de generaciones y pmut.

3.1.2 Describir las métricas de evaluación

Las métricas de evaluación usadas son:

• **Fitness**: La métrica principal utilizada para la evaluación es el Fitness, tal como se detalla en el punto 2.4.1. Esta métrica permite identificar la mejor solución al problema planteado.

• **Valor de Generación de Convergencia**: La relevancia de este valor radica en que buscamos una solución óptima dentro de un número de iteraciones razonable. Es crucial evitar un número de iteraciones demasiado bajo, que podría comprometer la exploración y explotación del espacio de soluciones, así como un número excesivo de iteraciones, que incrementaría innecesariamente los recursos computacionales requeridos para el proceso, afectando la eficiencia del algoritmo.

3.1.3 Descripción de los experimentos hechos (qué componentes/ parámetros se probaron, que valores, qué estrategia de validación).

3.1.3.1 La Experimentación definida en 2 etapas

El objetivo de esta experimentación es encontrar la mejor solución (cronograma de clases) utilizando un algoritmo genético. Con base en los resultados que se presentarán en el punto 3.2, hemos concluido que el enfoque adoptado de 2 etapas, permite abordar y resolver eficientemente el problema planteado en el punto 1.1. Entendemos como una solución eficiente aquella que logra el óptimo de Fitness con un número razonable de generaciones, según lo expuesto en el punto 3.1.2. Por otro lado, es importante destacar el impacto de los parámetros en los resultados, especialmente en lo que respecta al número adecuado de individuos y generaciones. Además, resaltamos la importancia del uso de operadores combinados de cruzamiento y mutación en el modelo, lo que demuestra su peso en la obtención de soluciones óptimas.

A continuación, se describen las dos etapas.

Etapa 1 del Experimento

Busca determinar **los parámetros óptimos del algoritmo que son** el número de individuos en la población, el número de generaciones, la probabilidad de mutación (pmut), así como el tipo de operador o combinaciones de operadores a usar. Una vez definidos estos parámetros, se pudo proceder a ejecutar el algoritmo genético – usando los datos de partida tal cual se ha descrito en el punto 2.2.1 - con la intención de hallar la solución óptima. Iniciamos con el establecimiento de un escenario base en el

cual se llevaron a cabo cuatro simulaciones con tamaños de población de 2, 6, 10 y 50 individuos. En todas las simulaciones, se fijó el número de generaciones constante en 100 y la probabilidad de mutación constante en 0.50. Con estos parámetros, se exploraron las siguientes combinaciones de operadores (escenarios):

1. Uniform
2. One Point
3. One Point & Flip
4. One Point & MultiFlip
5. Uniform & Flip
6. Uniform & MultiFlip

Este primer experimento tuvo como fin el de identificar **el mejor set de parámetros que permitiría un equilibrio entre exploración y explotación del espacio de soluciones para alcanzar la convergencia y lograr el valor máximo de fitness de 1,841**, utilizando el número de generaciones como una métrica clave ya que se buscar que un número de iteraciones razonable como lo expuesto en el punto 3.1.2. Los detalles de las 6 simulaciones se encuentran en los anexos 2 al 7. Los resultados de esta etapa se detallan en 3.2.1

Etapa 2 del Experimento

Logrado el éxito en la etapa 1, pasamos a realizar la etapa 2, que consiste en el uso de los parámetros definidos en el experimento 1 en el algoritmo genético para hallar la mejor solución posible a través del Fitness y el resultado de generaciones requerida para alcanzar dicho valor. En esta fase, se incluyó un criterio de parada (ver 2.3.1 etapa #3) que permite finalizar el proceso cuando se alcanza la convergencia, optimizando así la eficiencia del algoritmo y reduciendo el tiempo de ejecución. Los resultados de esta etapa se detallan en 3.2.2

3.2 Resultados y Discusión:

3.2.1 Resultados de la etapa 1 del experimento

La primera etapa dio como resultado que los mejores parámetros para correr la etapa 2 son:

- **Cruzamiento:** One Point
- **Mutación:** MultiFlip
- **Población:** 50
- **Generaciones:** 100
- **Pmut:** 5%

A continuación, se sustenta dicho resultado: Cada escenario de los 6 definidos en la etapa 1, fue evaluado en términos de su capacidad para alcanzar la convergencia y lograr el valor máximo de fitness de 1,841 (Ver anexos 2 al 7). El criterio para seleccionar la mejor combinación de **operadores de cruzamiento y mutación** se basó en identificar aquella (combinación) que lograra el objetivo de Fitness con un **número razonable de generaciones (ver punto 3.1.2).** Luego de finalizadas las simulaciones anteriores, se concluye que los resultados más eficientes, entendidos como aquellos que logran el objetivo (máximo fitness) y mantienen la diversidad, son aquellos en los cuales se combinan el cruzamiento con mutación (en los modelos combinados de cruzamiento y mutación, se logra el fitness máximo en 15 de las 16 simulaciones según anexos 4 al 7). Así, consideramos que el mejor modelo es el de **One Point & MultiFlip**, ya que logra el fitness máximo de 1,841 en las 4 simulaciones, y lo hace con el menor

número razonable de generaciones posibles, particularmente con poblaciones de 10 y 50 individuos (8 y 9 generaciones, respectivamente), lo que muestra un equilibrio adecuado entre eficiencia y exploración. Entre ambas poblaciones, optamos por un la de 50 individuos, ya que apoya la variedad de soluciones en el punto de convergencia. Por lo tanto, **One Point & MultiFlip** es el modelo más eficiente para este problema La siguiente tabla muestra la comparación de las distintas simulaciones en el logro del fitness objetivo (en rojo valores de Fitness = 1841):

Poblacion / Tipo de Cruz. - Mut.	2 ind.	6 ind.	10 ind.	50 ind.
1. Uniform	1377	1529	1685	1841
2. One Point	1225	1537	1841	1841
3. One Point & Flip	1841	1841	1841	1841
4. One Point & MultiFlip	1841	1841	1841	1841
5. Uniform & Flip	1841	1841	1841	1841
6. Uniform & MultiFlip	1689	1841	1841	1841

FUENTE: Anexos 2 – 7

Para reforzar la conclusión anterior, realizamos 6 simulaciones adicionales únicamente con los 2 tipos de mutación (Flip y MultiFlip) para el set de # de Individuos = 50, # de Generaciones = 100, Tipo de Cruce = One Point y 3 tipos de Pmut: 5%, 10%, y 100%. El resultado final (ver anexos 8 y 9) concluye que las tasas de mutación moderadas (5% o 10%) son suficientes para lograr un buen rendimiento del algoritmo, y que tanto el operador Flip como el MultiFlip son válidos y no muestran diferencias significativas. Dado un escenario de "empate" y basándonos en la primera conclusión, optamos por el operador **MultiFlip.** Asimismo, seleccionamos, por eficiencia operativa el Pmut de 5%, ya que realizará la mitad de mutaciones que Pmut 10%) y obtendrá resultados similares a este.

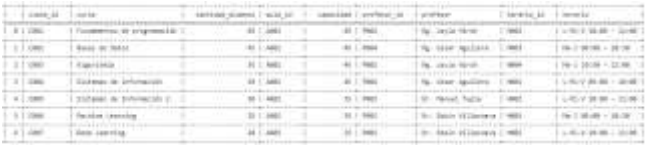
3.2.2 Resultados de la etapa 2 del experimento

Los parámetros definidos son:

- **Cruzamiento:** One Point
- **Mutación:** MultiFlip
- **Población:** 50
- **Generaciones:** 100
- **Pmut:** 5%

Así, se procede a correr el algoritmo genético en búsqueda de la mejor solución – Cronograma para los 7 cursos - que se muestra a continuación:

CRONOGRAMA OPTIMO DE DICTADO DE CLASES



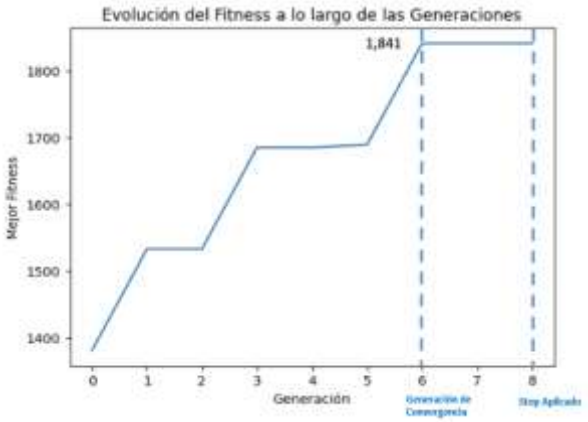
FUENTE: Código Python de Solución de Problema (Anexo 11)

Se comprueba lo requerido por el programa en el sentido de NO incumplir las restricciones obligatorias:

- Se dictan los 7 cursos durante la semana sin duplicidad (Restricción de duplicado
- NO se excede la capacidad de clases (Restricción de Capacidad)
- Un profesor NO está dictando al mismo tiempo (Restricción de Profesor en el tiempo)
- Las clases NO están ocupadas al mismo tiempo en dictado de clases (Restricción de Aula en el tiempo

En cuanto a la eficiencia del código, podemos observar que la solución alcanzó el valor máximo esperado de 1,841. Además, como se mencionó anteriormente, la

efectividad del código también se mide por la capacidad de alcanzar el máximo fitness en un número razonable de generaciones, que en este caso fue de 6. Después de la generación 6, y tras un seguimiento posterior en las siguientes 2 generaciones, se activó la función de parada (Stop) para evitar que el algoritmo continúe ejecutando lo cual consumiría recursos innecesarios, dado que ya se había alcanzado el objetivo de identificar al individuo con el máximo fitness (1,841) en 2 iteraciones anteriores..



FUENTE: Código Python de Solución de Problema (Anexo 10)

4. Conclusión

Tomando como los resultados obtenidos y la hipótesis planteada, se puede concluir lo siguiente sobre los enfoques desarrollados y el problema abordado:

1. **Eficiencia del Enfoque**: El algoritmo genético cumplió exitosamente con el objetivo de hallar el mejor cronograma vía maximización del fitness, alcanzando el valor óptimo de 1,841 puntos en 6 generaciones. Esto valida la hipótesis de que el uso de un algoritmo genético es efectivo para resolver este problema complejo de asignación de recursos bajo restricciones específicas y apuntado al logro de eficiencia operativa (número razonable de generaciones).

2. **Optimización de Parámetros**: La combinación de los operadores "One Point" para el cruzamiento y "MultiFlip" para la mutación, junto con una población de 50 individuos y una tasa de mutación del 5%, resultó ser la más eficiente. Este enfoque optimizado permitió lograr el objetivo con un equilibrio adecuado entre exploración y explotación del espacio de soluciones.

3. **Cumplimiento de Restricciones**: El enfoque desarrollado no solo alcanzó el valor máximo de fitness, sino que también cumplió estrictamente con todas las restricciones obligatorias del problema (ver punto2.4.1), confirmando la robustez del algoritmo para manejar múltiples restricciones simultáneamente.

4. **Eficiencia en el Uso de Recursos**: La implementación de una función de parada fue crucial para evitar el consumo innecesario de recursos al detener el proceso una vez alcanzado y repetido el fitness máximo en dos generaciones consecutivas, lo que subraya la eficiencia del enfoque desarrollado.

En resumen, el enfoque de solución de este problema basado en algoritmos genéticos demostró ser una **solución efectiva y eficiente** para el problema de asignación de horarios, aulas y profesores a cada uno de los 7 cursos, cumpliendo con la hipótesis inicial y ofreciendo un método robusto para abordar problemas similares en otros contextos.

5. Sugerencias de trabajos futuros

1. Dado que la infraestructura necesaria para largas corridas (alto número de individuo y generaciones) y la respectiva visualización del output (truncamiento de reporte en pantalla sobre > 5,000 líneas que llevo a crear una rutina de envío de output a .txt pero este camino tuvo eventos de crash continuos), superó las capacidades técnicas en manos del equipo de trabajo, se recomienda una mejora en el acceso a infraestructura más potente (i.e AWS).

2. Por otro lado, consideramos que el enfoque basado en algoritmos genéticos, empleado en la resolución del problema planteado, tiene un amplio potencial de aplicación en otros ámbitos de la vida. A continuación, 4 ejemplos de problemas que podrían abordarse con este enfoque incluyen:

- **Optimización de rutas de transporte para servicios de delivery de alimentos frescos**, maximizando la eficiencia y minimizando los tiempos de entrega.
- **Planificación de la utilización de salas de operación en hospitales** para cirugías no urgentes, optimizando la asignación de recursos (médicos) y reduciendo tiempos de espera del paciente con efecto en reducción de costos de hospitalización por menor tiempo de estadía.
- **Gestión de la asignación de aviones, mangas y horarios en aeropuertos**, asegurando un uso eficiente de los recursos (aviones) y mejorando la coordinación.
- **Distribución y asignación de espacios en eventos, como la organización de salas de cine**, optimizando la ocupación de espacios y mejorando la experiencia del usuario.

6. Link del repositorio del trabajo

Las credenciales para poder tener acceso vía Github son:

Link: https://github.com/grupocronograma/Cronograma/projects?query=is%3Aopen

Username: grupocronograma

Clave: Cronograma12345

7. Declaración de contribución de cada integrante

Tem	Sandra A,	Piero T.	Manuel A.	Danilo S.
Codificación	3	3	2	3
Experimentación , simulación	3	3	3	3
Aporte a la estrategia de la solución	4	2	4	4
Estructura del informe final y presentación	4	3	4	4

Donde (1) es Participación básica y (5), es Alta participación

8. Referencias

Los materiales del Curso tales como presentaciones y videos, han sido utilizados para guiar el desarrollo del problema.

Anexos en Documento pdf : Informe Final – Anexos – CRONOGRAMA DE CLASES - 310824