Optimización mediante Algoritmos Evolutivos: Asignación de Salones para Exámenes

Ingeniería en Computación – Facultad de Ingeniería (UdelaR)

1. Introducción

El presente trabajo aborda el problema de la **asignación de salones para exámenes** dentro de un periodo máximo de 25 días. El objetivo es generar un cronograma que respete las restricciones de aforo y duración y evite solapamientos entre materias de un mismo semestre (basándonos en la curricula sugerida de cada carrera), buscando además un uso eficiente de los recursos humanos evitando asignar salones de más.

Este tipo de problema pertenece a la categoría de *problemas de optimización combinatoria con restricciones*, caracterizado por un espacio de búsqueda de gran tamaño y múltiples objetivos en conflicto. Por ello, se propone resolverlo mediante un **algoritmo evolutivo multiobjetivo** utilizando la biblioteca `jMetal`.

2. Formulación del problema

Dado un conjunto de exámenes $E = \{e_1, e_2, \ldots, e_n\}$ y un conjunto de salones $S = \{s_1, s_2, \ldots, s_m\}$, se busca asignar a cada examen:

- Un día $d_i \in [0, 24]$ (máximo 25 días),
- Una franja horaria
- Una asignación de salones $C_i \subseteq S$ tal que el aforo total de C_i cubra la cantidad de inscriptos en e_i.

Se pretende que la planificación resultante cumpla las restricciones y optimice las siguientes metas:

1. Maximizar la separación entre exámenes de un mismo semestre de una misma carrera universitaria.
2. Minimizar la cantidad total de salones asignados a cada examen.

3. Codificación de los individuos

El espacio de búsqueda se representa mediante un vector basado en *slots*. Cada slot corresponde a una posible asignación de examen y contiene la siguiente información:

$$\text{slot} = [\text{examen\_index}, \text{dia}, \text{salon}_1, \text{salon}_2, \text{salon}_3, \text{salon}_4]$$

Donde:

- **examen_index**: Índice del examen (0 a $n-1$, donde n es el número de materias). El valor n representa un slot vacío.
- **dia**: Índice del día preferido (0 a 24, correspondiente al período máximo de 25 días).
- **salon**$_i$: Índice del salón i (0 a $m-1$, donde m es el número de salones). El valor m representa "sin salón adicional".

El vector completo tiene tamaño $n \times 6$, donde n es el número de exámenes (un slot por cada examen posible) y 6 corresponde al tamaño de cada slot (1 examen + 1 día + 4 salones).

La **decodificación** se realiza procesando cada slot en orden:

1. Si el examen es vacío, se ignora el slot.
2. Se obtienen los salones válidos (excluyendo el valor "sin salón").
3. Se busca el horario más temprano en el día preferido donde **todos** los salones estén libres simultáneamente.
4. Si no hay disponibilidad en el día preferido, se intenta en días cercanos.
5. Se asigna el examen a ese horario en todos los salones especificados.

Esta estrategia garantiza que todos los salones de un mismo examen compartan el mismo día y horario, cumpliendo la restricción de simultaneidad.

4. Funciones objetivo

Se definen dos funciones objetivo principales:

1) Minimizar cantidad de salones utilizados

$$f_1 = \sum_{k=1}^{n} C_k$$

donde C_k es la cantidad de salones asignados al examen e_k. Este objetivo busca minimizar el número total de asignaciones materia-salón, promoviendo un uso eficiente de los recursos.

2) Maximizar separación entre exámenes del mismo semestre

$$f_2 = -\frac{1}{|S|} \sum_{(i,j) \in S} |d_i - d_j|$$

donde S es el conjunto de pares de exámenes pertenecientes al mismo semestre y d_i, d_j son los días asignados a cada examen. El signo negativo se debe a que `jMetal` minimiza, por lo que se minimiza $-f_2$ para maximizar la separación promedio.

5. Restricciones

Las restricciones que definen la validez de una solución son:

- **Aforo:** el conjunto de salones asignado debe cubrir los inscriptos. Se penaliza el déficit de capacidad.
- **Disponibilidad:** un salón no puede usarse en dos exámenes a la vez. Esto se garantiza durante la decodificación.
- **Límite temporal:** el período total no puede superar los 25 días.
- **Asignación simultánea:** Si un examen requiere más de un salón, todos deben compartir el mismo día y horario. Esto se garantiza por la estrategia de decodificación.
- **Asignación completa:** todas las materias deben estar asignadas. Se penaliza la cantidad de materias no asignadas.

Las restricciones se manejan mediante:

- **Decodificación inteligente** que busca horarios disponibles automáticamente.
- **Restricciones en jMetal** que penalizan soluciones infactibles (déficit de capacidad y materias no asignadas).
- Un **operador de reparación** que reasigna exámenes conflictivos cuando es posible.

6. Operadores evolutivos

Los operadores seleccionados son:

- **Inicialización:** aleatoria con combinaciones válidas de salones.
- **Selección:** Para la selección de los individuos de la población usaremos BinaryTournamentSelection, usado típicamente en metodologías como NSGA-II que ha demostrado dar buenos resultados para problemas multi-objetivo.
- **Cruce:** Para la parte de cruzamiento usaremos el método de cruzamiento de dos puntos. individuos.
- **Mutación:** `IntegerPolynomialMutation`, alternando salón y/o posición del examen en el salón.

7. Evaluación

En cada evaluación:

1. Se decodifica la solución.
2. Se verifica el cumplimiento de las restricciones.
3. Se calculan las funciones objetivo.

El algoritmo evolutivo utilizado es `NSGA-II`, apropiado para problemas multiobjetivo.

7.0.1. Propuesta de evaluación experimental

7.0.2. Generación de instancias

Se considerarán las instancias correspondientes a los períodos de exámenes de las carreras de Ingeniería, para los tres períodos de exámenes del 2024:

- Febrero (02) - Julio (07) - Diciembre (12) A su vez, se tendrá una cuarta instancia: - Promedio de todas las intancias: Se hará un promedio entre los inscriptos en cada período, obteniendo una cuarta instancia.

Cada instancia incluirá:

- Conjunto de exámenes (materia + cantidad de inscriptos + duración de examen).
- Relaciones entre materias de un mismo semestre acorde a la currícula sugerida.
- Salones disponibles y su capacidad.
- Duración y ventana temporal del período de exámenes (máximo 25 días).

7.0.3. Comparación con otras técnicas

Para evaluar la calidad de las soluciones obtenidas mediante el algoritmo evolutivo, se implementará un algoritmo greedy que asignará exámenes secuencialmente en función de su tamaño y restricciones buscando el mejor fit, siguiendo una heurística simple.

El greedy servirá como baseline o referencia comparativa frente a las soluciones multiobjetivo obtenidas con el AE.

7.0.4. Criterios de evaluación

Se evaluará la calidad y eficiencia de las soluciones mediante las siguientes métricas:

- Separación promedio entre exámenes relacionados (calidad académica). - Número total de salones utilizados (eficiencia de recursos). - Tiempo de cómputo requerido por cada método y cantidad de generaciones necesarias. - Valor de fitness (promedio, media, mejor y desviación) y cantidad de veces que se obtuvo.

8. Desarrollo

9. Evaluación de resultados

En esta sección detallaremos los resultados obtenidos de las ejecuciones del algoritmo bajo diversas configuraciones paramétricas e instancias del problema mencionadas anteriormente.

9.1. Configuración de parámetros

- Tamaño de población: 50, 100, 200
- Probabilidad de cruzamiento: 0.6, 0.7, 0.8
- Probabilidad de mutación: 0.1, 0.01, 0.001

En base a estas configuraciones, se generan 27 combinaciones para realizar con cada instancia del problema.

9.1.1. Comparación de resultados

10. Conclusiones

El modelo planteado permite generar planificaciones válidas de exámenes considerando múltiples criterios. La codificación entera facilita la implementación en `jMetal`.

Como posibles extensiones:

- Considerar restricciones de disponibilidad de docentes.
- Incluir búsqueda local híbrida sobre soluciones no dominadas.