

# Algoritmos Evolutivos - 2025

## Asignación de Salones para Exámenes

Martín Ochoa  
Facultad de Ingeniería, UDELAR  
Montevideo, Uruguay  
Email: martin.ochoa@fing.edu.uy

Mateo Vargas  
Facultad de Ingeniería, UDELAR  
Montevideo, Uruguay  
Email: mateo.vargas@fing.edu.uy

**Resumen**—En este informe se presenta una implementación de un algoritmo evolutivo que busca optimizar la asignación de salones a exámenes, considerando la separación temporal entre materias del mismo semestre, minimizando los recursos humanos necesarios para cada examen.

### I. DESCRIPCIÓN DEL PROBLEMA

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 horarios disponibles, maximice la distancia temporal entre materias de un mismo semestre (basándonos en la curricula sugerida de cada carrera), buscando además un uso eficiente de los recursos humanos.

### II. JUSTIFICACIÓN DE USAR AE

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.

Tomando en consideración lo anterior, el uso de un algoritmo evolutivo se presenta como una alternativa apropiada para abordar este problema. Estos algoritmos destacan por su capacidad de explorar de manera eficiente amplios espacios de búsqueda, identificando soluciones de buena calidad en tiempos de ejecución razonables. Además, permiten tratar problemas con múltiples objetivos de forma natural, mediante enfoques como el frente de Pareto, que posibilitan obtener soluciones de compromiso entre diferentes criterios sin requerir la formulación de una única función objetivo.

### III. ESTRATEGIA DE RESOLUCIÓN

#### A. Representación

Dado un conjunto de exámenes  $E = \{e_1, e_2, \dots, e_n\}$  y un conjunto de salones  $S = \{s_1, s_2, \dots, s_m\}$ , 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  $e_i$ .

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<sub>i</sub>**: 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.

#### B. 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.

## C. 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.

## D. 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 multiobjetivo.
- **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.

## E. Técnicas avanzadas,

Dado que se trata de un problema multiobjetivo y se empleará un modelo basado en el frente de Pareto para su resolución, resulta indispensable incorporar un mecanismo que mantenga la diversidad en la población, permitiendo así un muestreo adecuado del frente de Pareto. En particular, dado que el algoritmo utilizado es NSGA-II, se empleará el mecanismo de crowding distance provisto por este, el cual asegura una distribución equilibrada de las soluciones a lo largo del frente.

## IV. PROPUESTA DE EVALUACIÓN EXPERIMENTAL

### A. Generación de instancias

Se considerarán las instancias correspondientes a los períodos de exámenes de las carreras de Ingeniería, tomando en cuenta los tres períodos de exámenes del año 2024:

- **Febrero (02)**
- **Julio (07)**
- **Diciembre (12)**

Además, se definirá una **cuarta instancia promedio**, obtenida a partir del promedio del número de inscriptos en cada uno de los tres períodos anteriores. Esta instancia servirá como un caso general representativo.

Cada instancia incluirá la siguiente información:

- **Conjunto de exámenes:** materia, cantidad de inscriptos y duración del examen.
- **Relaciones entre materias:** vínculos entre asignaturas del mismo semestre, según la currícula sugerida.
- **Salones disponibles:** listado de aulas y sus respectivas capacidades.
- **Ventana temporal:** duración total del período de exámenes (máximo 25 días).

### B. 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, siguiendo una heurística simple.

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

### C. Calidad de soluciones

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.

#### D. Eficiencia computacional

La eficiencia computacional se evaluará mediante el tiempo de ejecución promedio (en segundos) obtenido a partir de múltiples ejecuciones independientes, complementado con su correspondiente desviación estándar para medir la variabilidad del desempeño.

### V. DECISIONES DE IMPLEMENTACIÓN

Para la implementación del algoritmo se usó Java 21 junto al framework de JMetal versión 6.6. El problema se modeló como un `IntegerProblem`, con operador de cruzamiento `TwoPointCrossover`, operador de mutación `IntegerPolynomialMutation` (con distribución polinomial de grado 5) y como operador de selección el `BinaryTournamentSelection` con el comparador `RankingAndCrowdingDistanceComparator`.

Para el algoritmo se creó una clase propia llamada `NSGAII_WithTelemetry`, basada en la clase de JMetal `NSGA-II`. Se introdujeron modificaciones como la integración de un *fitness tracker*, una clase que se creó llamada `EvolutionTracker` que se encarga de llevar y al finalizar guardar en un CSV la evolución del *fitness* del algoritmo, incluyendo métricas como el número de asignaciones, la separación entre exámenes relacionados y la cantidad de soluciones factibles por generación. Adicionalmente, se integró un operador de reparación `SolutionRepairOperator` que se ejecuta después de los operadores genéticos para mantener la factibilidad de las soluciones.

Para el modelado del problema se definió una clase llamada `ClassroomAssignmentProblem` extendiendo la clase `AbstractIntegerProblem`. El constructor recibe un objeto `ProblemInstance` como parámetro que representa la instancia del problema, es decir los exámenes con sus inscriptos y duraciones, los salones con sus capacidades, y las relaciones de conflicto entre materias. En base a esto define la cantidad de variables que serán evaluadas (número de exámenes multiplicado por el tamaño de cada slot, que es 6), define qué posición de los genes van a representar a qué examen, día y salones, y establece los rangos válidos para cada variable: el índice de examen puede ser de 0 a  $n$  (donde  $n$  representa vacío), el día de 0 a 24, y cada salón de 0 a  $m$  (donde  $m$  representa sin salón adicional).

Con el fin de optimizar lo más posible la función de *fitness* se establecieron una serie de estructuras para bajar el tiempo computacional de acceso a ciertos datos. Por ejemplo, se crearon listas pre-ordenadas de salones por capacidad (tanto ascendente como descendente), de forma que no hay que ordenarlos en cada iteración. A su vez se tiene un array

que contiene el mínimo de salones necesarios por materia, calculado mediante una estrategia greedy, y otro array con los bloques necesarios por materia. Estas estructuras permiten accesos rápidos durante la decodificación y evaluación de soluciones.

Dado que se utiliza un evaluador secuencial `SequentialSolutionListEvaluator` y considerando que la función de decodificación y evaluación requiere acceso a estructuras de datos compartidas (como la matriz de ocupación durante la decodificación), se decidió ejecutar el algoritmo en un solo núcleo, evitando el overhead de sincronización entre hilos y garantizando la consistencia de los datos.

### VI. CONFIGURACIÓN DE PARÁMETROS

En esta sección se detallan las pruebas realizadas para buscar la mejor configuración de parámetros del algoritmo evolutivo. Los parámetros que se van a probar junto con sus valores posibles son los siguientes:

TABLE I  
CONFIGURACIÓN DE PARÁMETROS DEL ALGORITMO EVOLUTIVO

Parámetro	Valores	Total
Tamaño de población	50, 100, 200	3
Probabilidad de cruzamiento	0.6, 0.7, 0.8	3
Probabilidad de mutación	0.1, 0.01, 0.001	3
<b>Combinaciones totales</b>		<b>27</b>

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

#### A. Metodología estadística

Para evaluar estadísticamente los resultados obtenidos, se siguió el siguiente procedimiento:

1) *Test de normalidad*: Primero se realizó un test de normalidad de Shapiro-Wilk sobre los valores de hipervolumen obtenidos para cada configuración paramétrica. Este test evalúa si los datos siguen una distribución normal, lo cual es un requisito para aplicar tests paramétricos como el ANOVA.

Los resultados del test de Shapiro-Wilk mostraron que algunos casos no siguen una distribución normal ( $p$ -valor  $< 0.05$ ), por lo que se procedió a utilizar tests no paramétricos para el análisis estadístico.

2) *Test de Kruskal-Wallis*: Dado que algunos grupos no cumplen con el supuesto de normalidad, se aplicó el test no paramétrico de Kruskal-Wallis para comparar las diferencias entre grupos. Este test es el equivalente no paramétrico del ANOVA de una vía y permite comparar si hay diferencias significativas entre tres o más grupos independientes sin requerir normalidad ni homocedasticidad.

El test de Kruskal-Wallis evalúa las siguientes hipótesis:

- $H_0$ : Las medianas de todos los grupos son iguales.
- $H_1$ : Al menos un grupo tiene una mediana diferente.

Los resultados del test de Kruskal-Wallis mostraron un estadístico  $H = 772.89$  con un p-valor de  $3.52 \times 10^{-146}$ , indicando que existen diferencias estadísticamente significativas entre las diferentes configuraciones de parámetros evaluadas (p-valor  $< 0.05$ ).

Para determinar la configuración paramétrica que mejor se comporta, pero sin ser demasiado optimista tomando únicamente la mejor solución individual, se calculó la media del hipervolumen de las 27 combinaciones y se seleccionó la configuración con la mejor media. Esto resultó en la siguiente configuración:

- Tamaño de población: 100
- Probabilidad de cruzamiento: 0.8
- Probabilidad de mutación: 0.001

Esta configuración será utilizada de aquí en adelante para todas las evaluaciones posteriores.

## VII. COMPARACIÓN ENTRE EL ALGORITMO GREEDY Y NSGA-II

En esta sección se comparan los resultados obtenidos mediante un algoritmo Greedy determinista (Best-Fit) y el algoritmo evolutivo multiobjetivo NSGA-II, aplicados al problema de asignación de salones para exámenes. La comparación se realiza considerando tanto la calidad de las soluciones obtenidas como el costo computacional asociado a cada enfoque.

### A. Instancia promedio

1) *Algoritmo Greedy*: El algoritmo Greedy implementado sigue una estrategia determinista de asignación Best-Fit, priorizando la minimización del número total de asignaciones de salones y garantizando la factibilidad de la solución.

Para la instancia analizada, el algoritmo Greedy obtuvo los siguientes resultados:

- Asignaciones totales: 294
- Separación promedio entre exámenes: 8.44 días
- Déficit de capacidad: 0
- Materias sin asignar: 0
- Solución factible: Sí
- Tiempo de ejecución: 23 ms

Estos resultados evidencian que el enfoque Greedy es altamente eficiente desde el punto de vista computacional y logra soluciones factibles con un uso mínimo de recursos. Sin embargo, al tratarse de un enfoque mono-objetivo implícito, no optimiza explícitamente la separación temporal entre exámenes, lo cual impacta negativamente en la calidad del cronograma resultante desde el punto de vista académico.

2) *Algoritmo NSGA-II*: Para el algoritmo NSGA-II se seleccionó la configuración paramétrica con mejor desempeño promedio en términos de hipervolumen, identificada como:

- Tamaño de población: 100
- Probabilidad de cruzamiento: 0.8
- Probabilidad de mutación: 0.001

El algoritmo fue ejecutado durante 101 generaciones, con un total de 100 000 evaluaciones. El tiempo total de ejecución fue de aproximadamente 215 segundos.

Desde el punto de vista evolutivo, el algoritmo presenta un comportamiento estable y consistente. En la Figura 1 se observa la evolución de ambos objetivos a lo largo de las generaciones. Mientras que el objetivo de minimizar asignaciones se mantiene estable en torno al valor inicial (294), el objetivo de maximizar la separación promedio muestra una mejora progresiva y sostenida, pasando de 8.96 días a 14.36 días, lo que representa una mejora del 60.3%.

Asimismo, el porcentaje de soluciones factibles alcanza el 100% en las primeras generaciones y se mantiene constante hasta el final de la ejecución, lo que indica que el algoritmo logra respetar de forma efectiva las restricciones duras del problema.

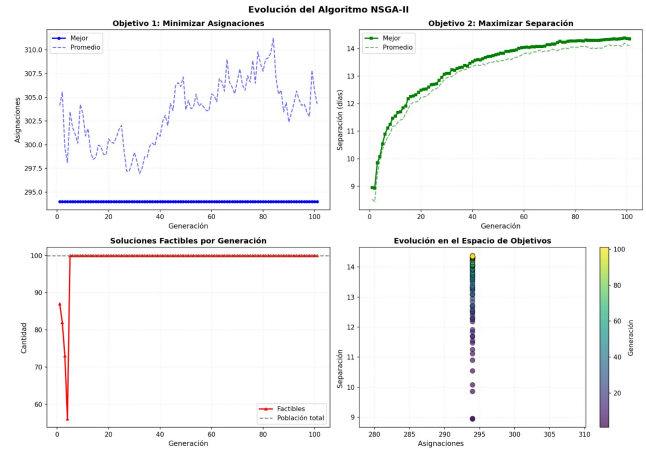


Fig. 1. Evolución del algoritmo NSGA-II: objetivos, factibilidad y espacio de búsqueda.

3) *Análisis del Frente de Pareto*: El algoritmo NSGA-II produce un conjunto de soluciones no dominadas que representan distintos compromisos entre los objetivos en conflicto. En la Figura 2 se muestra el frente de Pareto aproximado obtenido para la última generación, compuesto por 10 soluciones factibles.

El frente evidencia claramente el trade-off entre ambos objetivos: a medida que se incrementa la separación promedio entre exámenes, aumenta también el número de asignaciones de salones requeridas. Este comportamiento no puede ser capturado por el enfoque Greedy, el cual retorna una única solución puntual.

Como solución representativa del frente se seleccionó un punto de compromiso con los siguientes valores:

- Asignaciones totales: 302
- Separación promedio: 14.09 días

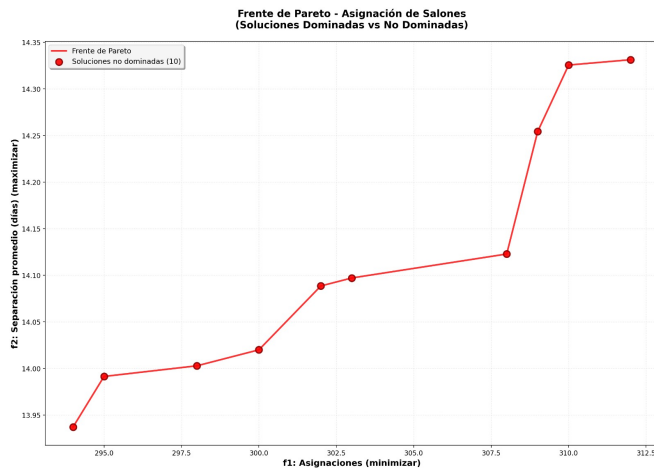


Fig. 2. Frente de Pareto aproximado obtenido por NSGA-II.

- Déficit de capacidad: 0
- Materias sin asignar: 0
- Solución factible: Sí

4) *Comparación Global:* La Tabla II resume la comparación directa entre el algoritmo Greedy y la solución de compromiso obtenida mediante NSGA-II.

TABLE II  
COMPARACIÓN ENTRE GREEDY Y NSGA-II

Métrica	Greedy	NSGA-II
Asignaciones reales	294	302
Exceso de salones	0	8
Separación promedio (días)	8.44	14.09
Tiempo de ejecución (ms)	23	215898
Solución factible	Sí	Sí

Del análisis se desprende que el algoritmo NSGA-II logra una mejora sustancial en la separación temporal entre exámenes (incremento del 66.9%), a costa de un aumento moderado en el número de asignaciones (2.7%) y un costo computacional significativamente mayor. Esto refleja claramente la diferencia entre un enfoque determinista orientado a eficiencia y un enfoque evolutivo multiobjetivo orientado a la calidad global de la solución.

En síntesis, mientras que el algoritmo Greedy resulta adecuado cuando se prioriza la rapidez de ejecución y el uso mínimo de recursos, el algoritmo NSGA-II permite explorar compromisos más equilibrados entre objetivos en conflicto, generando cronogramas de mayor calidad académica, especialmente en contextos donde la separación entre evaluaciones es un factor relevante.