UNIVERSIDAD TORCUATO DI TELLA

Tecnología Digital V: Diseño de algoritmos
Licenciatura en Tecnología Digital
Segundo Semestre, 2025

TP2 - Heurísticas para el Problema de Programación de Taller de Flujo (FSSP)

El contexto

El **Problema de Programación de Taller de Flujo** (Flow Shop Scheduling Problem, FSSP) es un problema de optimización combinatoria clásico con gran relevancia en la ingeniería de producción y la investigación operativa. Consiste en determinar el orden en que deben procesarse un conjunto de trabajos en una serie de máquinas, de modo que todos los trabajos sigan el mismo orden de máquinas y se optimice un criterio, típicamente el **makespan** (tiempo total necesario para completar todos los trabajos).
El FSSP modela procesos productivos donde todos los trabajos atraviesan las mismas etapas (por ejemplo, ensamblaje, pintura, inspección), como ocurre en la industria automotriz, electrónica o de manufactura en serie. Encontrar una buena secuencia de trabajos permite reducir tiempos de producción, minimizar inventarios intermedios y aumentar la eficiencia del sistema productivo.
El FSSP es un problema $\mathcal{NP}$-hard, lo que motiva el uso de heurísticas y metaheurísticas para obtener soluciones de alta calidad en tiempos computacionales razonables. Entre ellas, la **heurística de Nawaz–Enscore–Ham (NEH)** se ha consolidado como una de las más efectivas y es considerada un punto de partida clásico para el diseño de nuevas técnicas.

El modelo

Sea un conjunto de n **trabajos** $J = \{J_1, \ldots, J_n\}$ y un conjunto de m **máquinas** $M = \{M_1, \ldots, M_m\}$. Cada trabajo J_i debe procesarse en las m máquinas en el mismo orden $M_1 \to M_2 \to \cdots \to M_m$. El tiempo de procesamiento de J_i en la máquina M_k es $p_{ik} \in \mathbb{R}_{>0}$.
Dada una secuencia $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ que indica el orden de los trabajos, se pueden calcular los tiempos de inicio y finalización de cada operación de forma recursiva. El objetivo es minimizar el **makespan** $C_{\max}$, definido como el tiempo de finalización de la última operación del último trabajo.

$$C_{\max} = \max_i C_{\pi_i, m}$$

Los datos

Se utilizarán las 31 instancias de benchmark de **ORLib**, disponibles en este archivo. Cada instancia contiene en la primera línea el número n de trabajos y m de máquinas. Luego, siguen n líneas, una para cada trabajo i, consistiendo en una secuencia de números $0,\ t_{i0}, 1,\ t_{i1},\ \ldots,\ m,\ t_{im}$, correspondientes a los tiempos de procesamiento del trabajo i en cada máquina.

El trabajo

El trabajo práctico consiste en implementar distintas heurísticas para el FSSP, combinarlas y evaluarlas utilizando las instancias de benchmark. Los grupos pueden ser de hasta 3 personas.

El trabajo puede implementarse en Python o C++, según decisión del grupo, justificando la elección. Se pide:

1. **Heurísticas constructivas.** Implementar (al menos) dos heurísticas constructivas distintas para el FSSP, donde una de ellas deberá ser la **Heurística de Nawaz–Enscore–Ham (NEH)**. Analizar la complejidad de los algoritmos y justificar su implementación.

2. **Operadores de búsqueda local.** Implementar (al menos) dos operadores de búsqueda local, tales como intercambio (*swap*) e inserción (*insertion*), analizando su complejidad y comportamiento.

3. **Métodos combinados.** Proponer un método que combine una heurística constructiva con los operadores de búsqueda local.

4. **Experimentación y discusión.** Realizar una experimentación exhaustiva con las instancias de benchmark provistas, comparando los métodos propuestos. Los métodos a comparar deben incluir, como mínimo:

   - Cada heurística constructiva de manera independiente.
   - Heurística constructiva + operador de búsqueda local (por ejemplo, NEH + swap o NEH + inserción).
   - Heurística constructiva (alguna) + combinación de operadores de búsqueda local.

   La experimentación debe considerar tanto la calidad de las soluciones como los tiempos de ejecución. Las decisiones de diseño deben estar justificadas por los resultados experimentales.

5. **Diseño e implementación.** Aplicando conocimientos de materias previas, la entrega debe incluir un diseño de clases adecuado. Incluir casos de test para al menos 2 clases. Se permite el uso de herramientas de AI para asistencia en diseño o testing (no para implementación directa), siempre que se documente claramente qué partes fueron generadas o influenciadas por AI, adjuntando los enlaces a las conversaciones pertinentes.

6. **Informe, presentación de resultados y entrega del código.** El informe debe describir los algoritmos, operadores, decisiones de diseño, implementación, testing, resultados experimentales e instrucciones de compilación y ejecución.

La realización de los puntos anteriores comprenden el 85% de la nota final. Para obtener el 15% restante, se pide diseñar, implementar y evaluar al menos una estrategia de randomización y/o metaheurística.

Modalidad de entrega

Se pide presentar el modelo y la experimentación en un informe de máximo 12 páginas que contenga:

- introducción al problema,
- descripción de los algoritmos implementados, según corresponda,
- consideraciones generales respecto a la implementación, incluyendo dificultades encontradas,

- resumen de resultados obtenidos en la experimentación,
- conclusiones, posibles mejoras y observaciones adicionales que consideren pertinentes.

Junto con el informe debe entregarse el código con la implementación del modelo. El mismo debe ser entendible, incluyendo comentarios que faciliten su corrección y ejecución.

Fechas de entrega

Formato Electrónico: **Jueves 27 de noviembre de 2025, 23:59 hs**, enviando el trabajo (informe + código) vía el campus virtual.
Importante: El horario es estricto. Las entregas recibidas después de la hora indicada serán considerados re-entrega.