Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing

Said Aqil, Karam Allali

Luis Gustavo Suazo Rojas
Pablo Javier Gutierrez Aguirre
Nicolás Eduardo Netz Carrasco

Facultad de Ingeniería
Universidad de Concepción

Sistemas de Producción
19 de julio de 2022
Profesor: Eduardo Salazar H.

Tabla de Contenidos

Introduccion

- Presentar el problema del HFS
- Analizar la propuesta de solución
- Analizar los resultados
- Presentar discusión con 3 papers

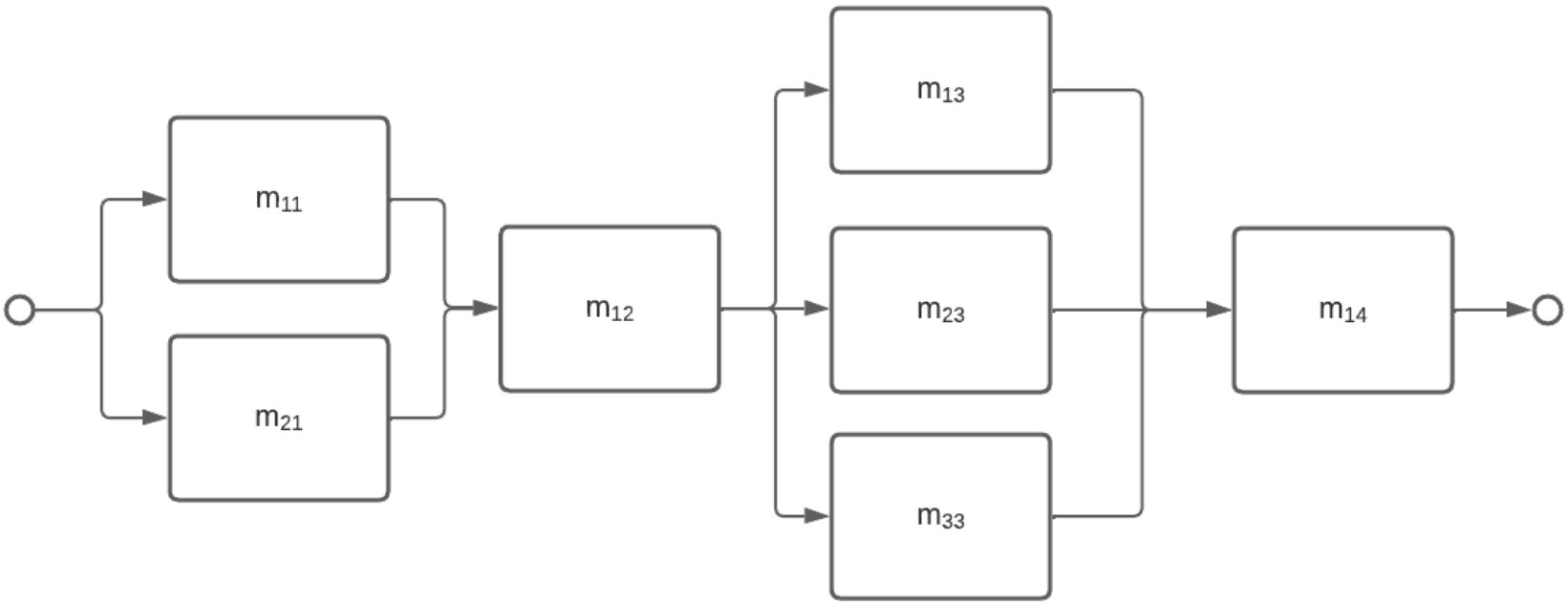
Tabla de Contenidos

Hybrid Flow Shop

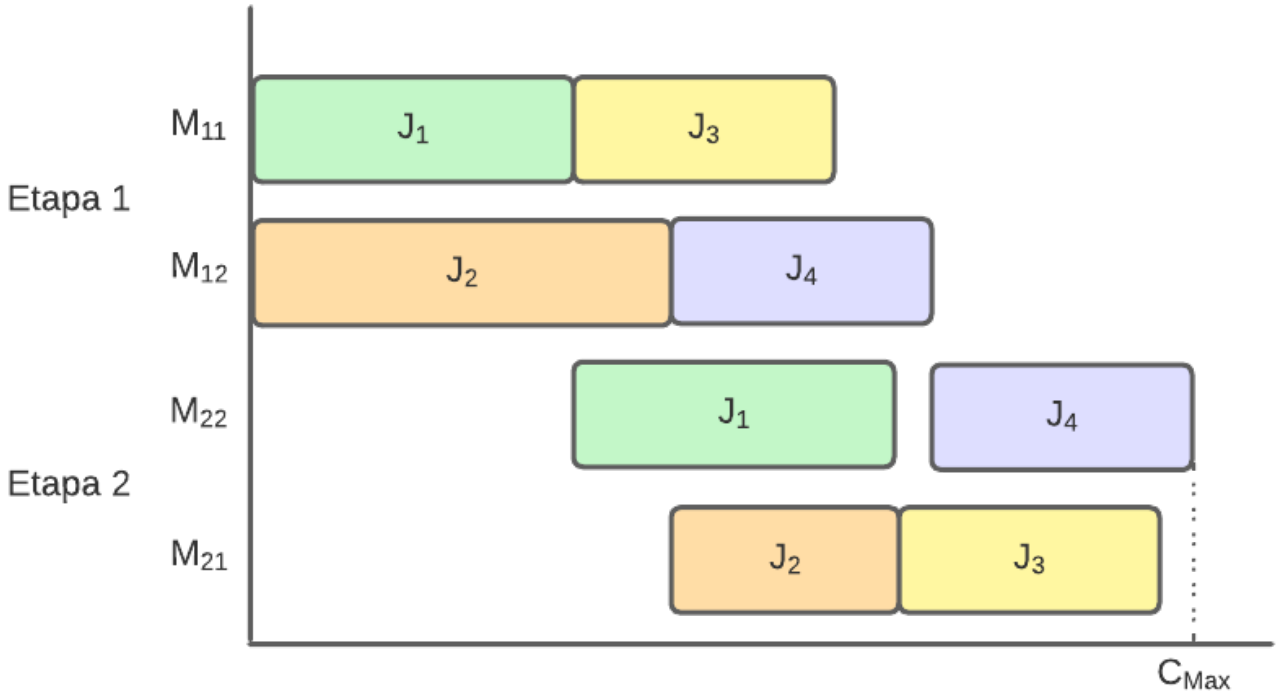
- El modelamiento del problema se puede realizar mediante Programación Matemática, sin embargo, el problema es clasificado como $NP - Hard$, por lo que se suele resolver con métodos aproximados
- El objetivo es obtener una calendarización para decidir qué trabajo asignar a qué maquina en qué momento, dados ciertos parametros
- En la literatura, las F.O utilizadas buscan por ejemplo minimizar el tardiness, el makespan, etc.

Hybrid Flow Shop

- Serie de etapas de producción, cada una de las cuales tiene varias máquinas que funcionan en paralelo, donde al menos una etapa contiene más de una máquina para diferenciarse del Flow Shop tradicional.



Ejemplo de Solución



Otras condiciones

En la literatura se distinguen otras condiciones para definir el problema. En este caso de estudio se incluyen:

- Sequence Dependent Setup Time: Los tiempos de set up de las máquinas dependen sólo de los trabajos
- Máquinas paralelas: Los procesos de máquinas de una misma etapa no influyen los unos con los otros.

Modelo a estudiar

Minimizar la Tardanza Total (TT) de un Flow Shop Híbrido (HFS) con Tiempos de Setup Dependientes de la Secuencia (SDST) y máquinas paralelas no relacionadas.

SDST/HFS

Parámetros y definiciones

- n trabajos, $\mathbb{J} = \{J_1, J_2, ..., J_n\}$
- K etapas, $\mathbb{K} = \{1, 2, ..., K\}$
- m_k máquinas, $\mathbb{M}^{(k)} = \{M_{1k}, M_{2k}, ..., M_{m_k k}\}$
- Tiempo de proceso según máquina, trabajo y etapa, p_{ijk}
- Tiempo de preparación dependiente de la secuencia s_{hjk}
- Tiempo de entrega, d_j
- Tiempo de completación C_{ijk}
- Secuencia de trabajo inicial, $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$
- La secuencia de trabajos se asigna según SPT.
- Para los procesos entre etapas se utiliza la regla FIFO.
- Notación del problema $\alpha|\beta|\gamma$: $HFS_K((RM)_k)_{k=1}^K|d_j, SDST|\sum_{j=1}^n T_j$
- El primer índice corresponde a la naturaleza del problema, el segundo las restricciones y el último el parámetro a optimizar.

SDST/HFS: Supuestos

Supuestos

- Todos los trabajos y máquinas están disponibles al inicio.
- Cada trabajo puede ser procesado por una máquina en cada etapa.
- Cada máquina puede procesar solo un trabajo a la vez.
- No hay interrupciones.
- Capacidad ilimitada de almacenamiento entre etapas.

SDST/HFS: Ecuaciones que describen el modelo

- Tiempo de completación de un trabajo inicial π_j en una maquina i en la etapa k:

$$C_{i\pi_j k} = max\{s_{\pi_j \pi_j k}, C_{\pi_j (k-1)}\} + p_{i\pi_j k} \quad (1)$$

- Tiempo de completación de un trabajo π_j luego de procesar el trabajo π_θ en una maquina i en la etapa k:

$$C_{i\pi_j k} = max\{C_{i\pi_\theta k} + s_{\pi_\theta \pi_j k}, C_{\pi_j (k-1)}\} + p_{i\pi_j k} \quad (2)$$

- Tardanza del trabajo π_j según tiempo de termino en la estación K

$$T_{\pi_j} = max\{0, C_{\pi_j K} - d_{\pi_j}\} \quad (3)$$

- Función objetivo, tardanza total

$$TT = \sum_{j=1}^{n} T_{\pi_j} \quad (4)$$

Con lo anterior, se busca minimizar la tardanza total encontrando una permutación óptima de π^*.

SDST/HFS: Ejemplo

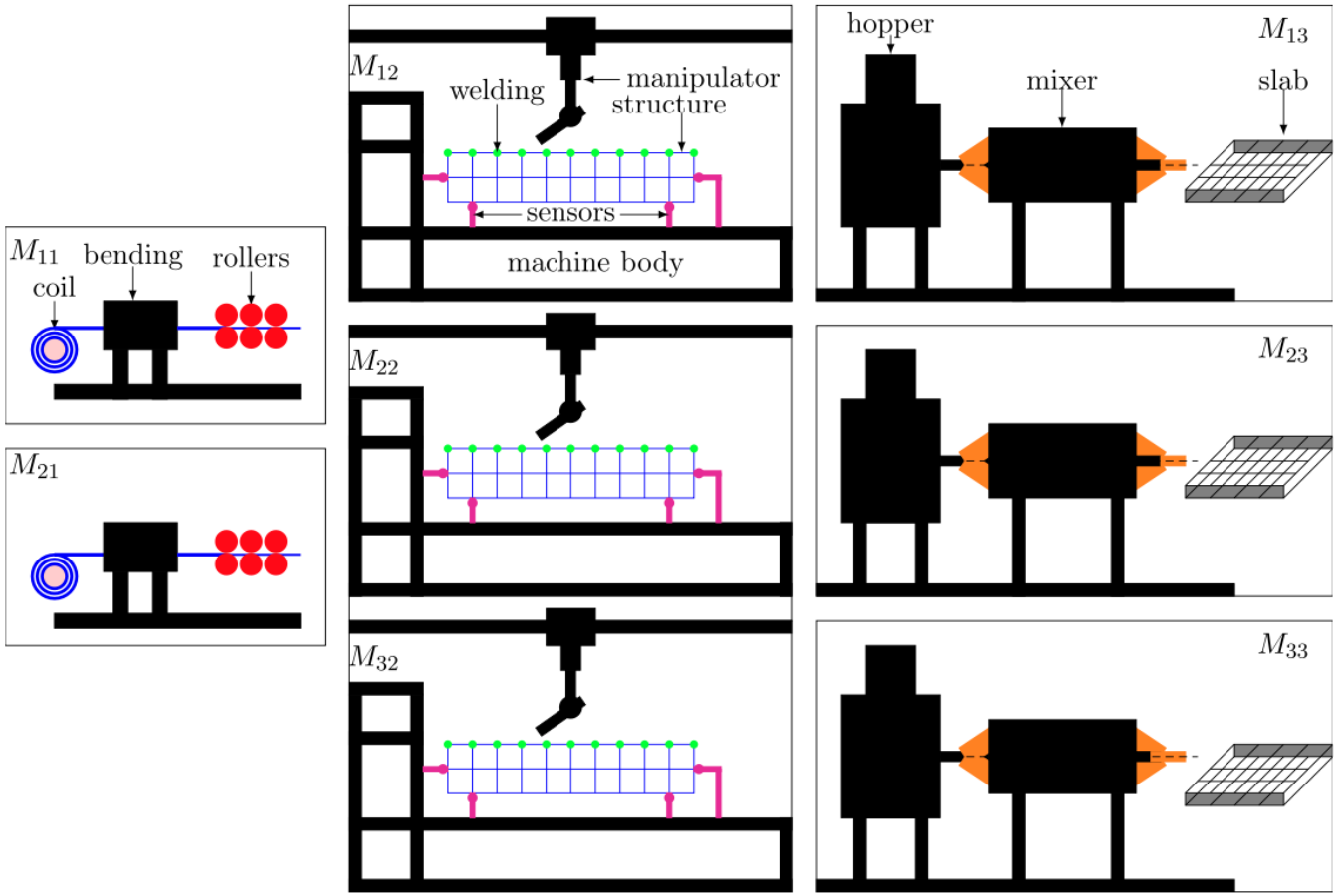
- Para un ejemplo de seis trabajos que se procesan en tres etapas.
- Primera etapa compuesta por dos máquinas, segunda y tercera etapa por tres máquinas.
- Maquinas paralelas no relacionadas en todas las etapas.
- Tiempos de entrega $d_j = \{23, 15, 27, 29, 19, 25\}$.

$$[p_{ij1}]_{2\times 6} = \begin{bmatrix} 6 & 2 & 4 & 8 & 5 & 5 \\ 5 & 4 & 6 & 6 & 7 & 4 \end{bmatrix}; \quad [p_{ij2}]_{3\times 6} = \begin{bmatrix} 5 & 6 & 7 & 8 & 6 & 4 \\ 6 & 8 & 4 & 6 & 10 & 8 \\ 8 & 7 & 5 & 5 & 12 & 3 \end{bmatrix}$$

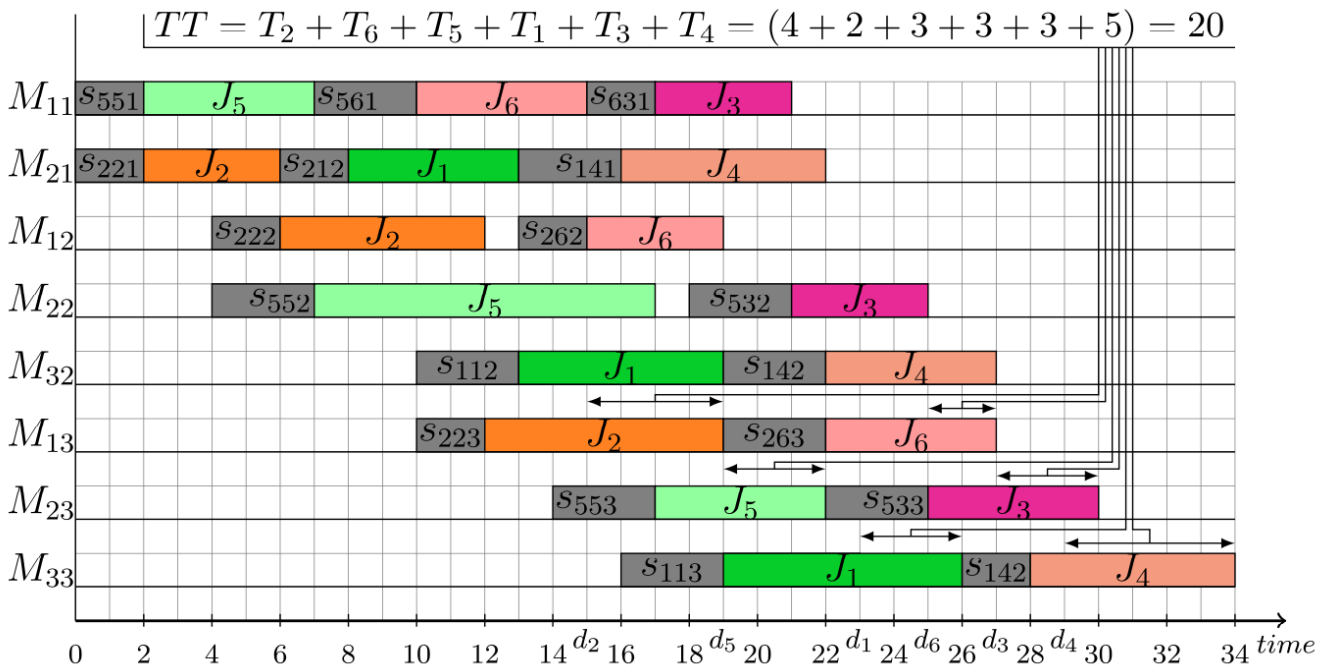
$$[p_{ij3}]_{3\times 6} = \begin{bmatrix} 8 & 7 & 7 & 6 & 4 & 5 \\ 4 & 9 & 5 & 4 & 5 & 8 \\ 7 & 8 & 5 & 6 & 6 & 7 \end{bmatrix}; \quad [s_{hj1}]_{6\times 6} = \begin{bmatrix} 3 & 3 & 2 & 3 & 3 & 1 \\ 2 & 2 & 2 & 1 & 2 & 2 \\ 4 & 2 & 4 & 2 & 4 & 3 \\ 2 & 3 & 3 & 2 & 2 & 2 \\ 4 & 2 & 1 & 2 & 5 & 3 \\ 2 & 2 & 2 & 4 & 2 & 1 \end{bmatrix}$$

$$[s_{hj2}]_{6\times 6} = \begin{bmatrix} 3 & 3 & 3 & 3 & 2 & 1 \\ 3 & 2 & 2 & 1 & 2 & 2 \\ 1 & 3 & 4 & 2 & 1 & 3 \\ 2 & 3 & 3 & 2 & 3 & 2 \\ 2 & 2 & 3 & 2 & 3 & 3 \\ 2 & 2 & 3 & 4 & 2 & 1 \end{bmatrix}; \quad [s_{hj3}]_{6\times 6} = \begin{bmatrix} 3 & 1 & 3 & 2 & 2 & 4 \\ 2 & 2 & 3 & 2 & 3 & 3 \\ 4 & 2 & 3 & 1 & 1 & 4 \\ 2 & 3 & 2 & 2 & 2 & 3 \\ 2 & 1 & 3 & 3 & 3 & 3 \\ 2 & 4 & 2 & 1 & 4 & 2 \end{bmatrix}$$

SDST/HFS: Ejemplo



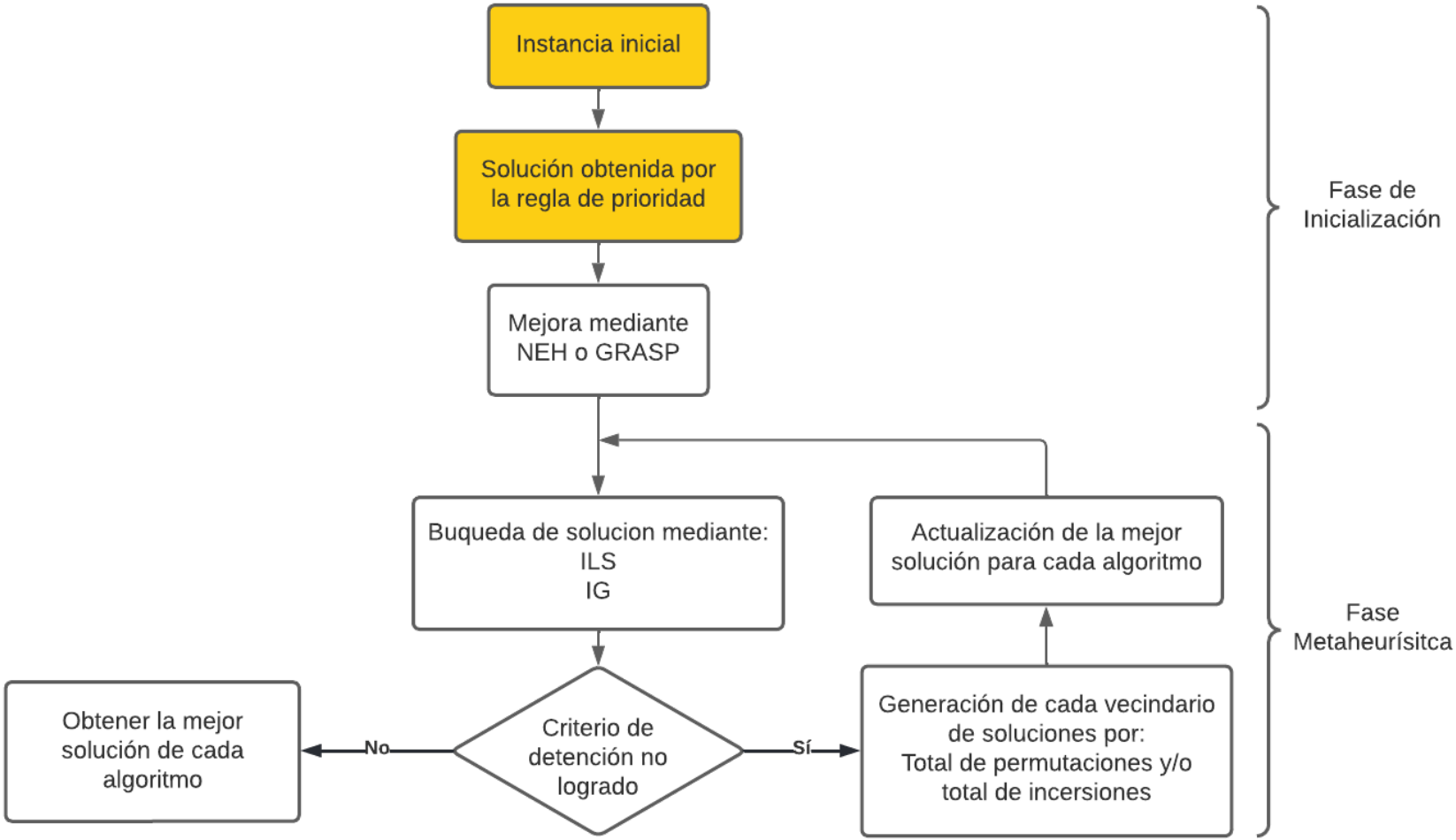
SDST/HFS: Solución gráfica



- Se tiene que la solución óptima es $\pi = \{5, 2, 1, 6, 4, 3\}$.
- Tardanza total de 20 unidades de tiempo.

Tabla de Contenidos

Modelos y Métodos de Solución



Modelos y Métodos de Solución: Representación

- La representación del problema se hace mediante una permutación de n trabajos, llamada π.

Ejemplo $n = 7$

$$\pi = [7, 5, 3, 1, 4, 2, 6]$$

Solución Inicial

- Se generan 4 reglas para obtener la solución incial, que permiten, dados los parámetros generar un ranking.
- Se seleccionan las mejores permutaciones π, una para cada regla.

Modelos y Métodos de Solución: Inicialización

$$p_{jk}^{-} = \min_{1 \leq i \leq m_k} (p_{ijk}) \tag{5}$$

$$p_{jk}^{+} = \max_{1 \leq i \leq m_k} (p_{ijk}) \tag{6}$$

$$\bar{p}_{jk} = \frac{\sum_{i=1}^{m_k} p_{ijk}}{m_k} \tag{7}$$

$$s_{jk}^{-} = \min_{1 \leq h \leq n, (h \neq j)} (s_{hjk}) \tag{8}$$

$$s_{jk}^{+} = \max_{1 \leq h \leq n, (h \neq j)} (s_{hjk}) \tag{9}$$

$$\lambda \in [0, 1] \tag{10}$$

Modelos y Métodos de Solución: Inicialización

Regla de prioridad $\delta_j^{(1)}$: (mínimo, mínimo)

$$\delta_j^{(1)} = \lambda \times \left| \sum_{k=1}^{K} p_{jk}^{-} - d_j \right| + (1-\lambda) \times \left| \sum_{k=1}^{K} s_{jk}^{-} - d_j \right| \tag{11}$$

Regla de prioridad $\delta_j^{(2)}$: (máximo, máximo)

$$\delta_j^{(2)} = \lambda \times \left| \sum_{k=1}^{K} p_{jk}^{+} - d_j \right| + (1-\lambda) \times \left| \sum_{k=1}^{K} s_{jk}^{+} - d_j \right| \tag{12}$$

Modelos y Métodos de Solución: Inicialización

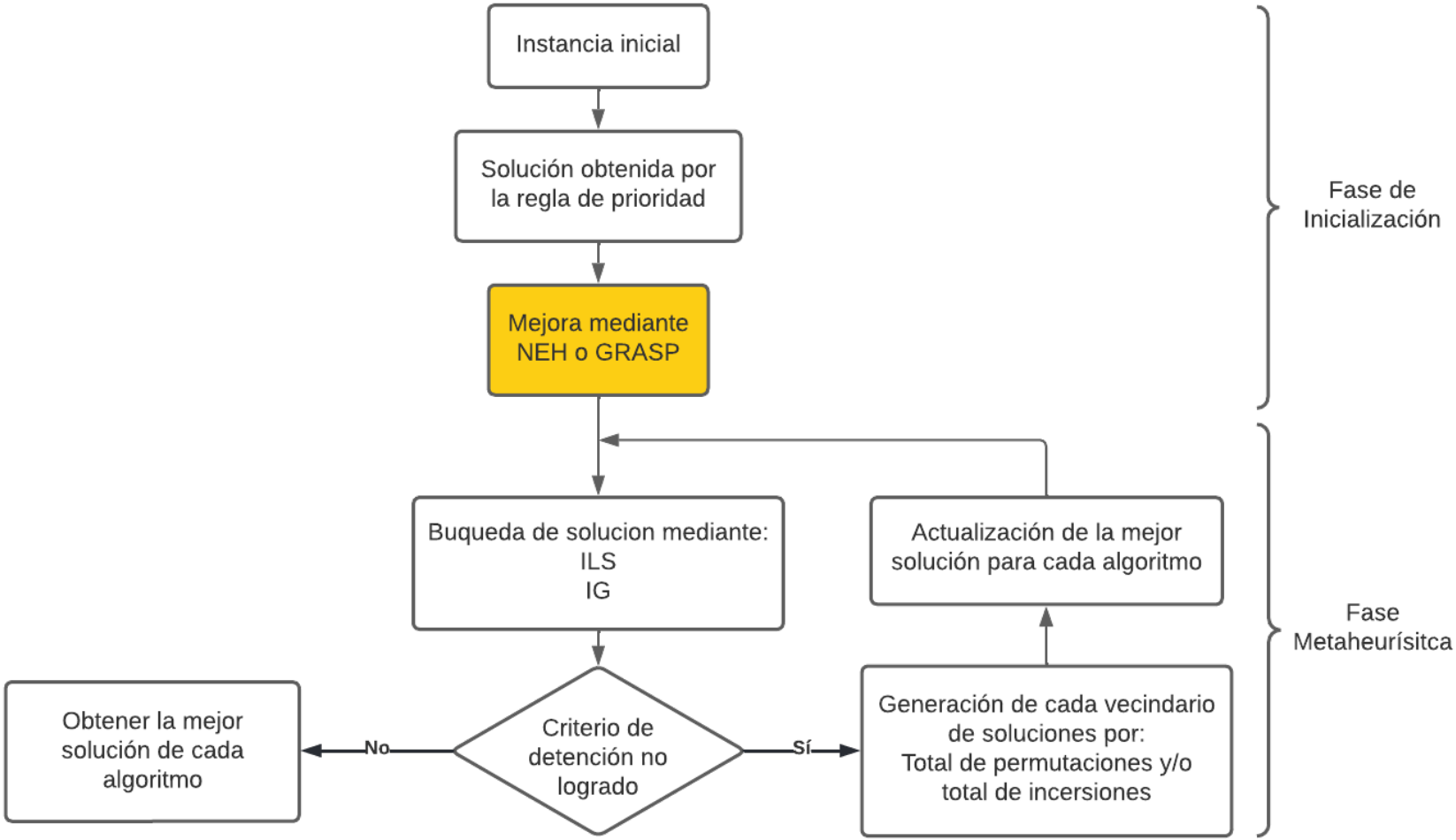
Regla de prioridad $\delta_j^{(3)}$: (promedio, mínimo)

$$\delta_j^{(3)} = \lambda \times \left| \sum_{k=1}^{K} \bar{p}_{jk} - d_j \right| + (1-\lambda) \times \left| \sum_{k=1}^{K} s_{jk}^{-} - d_j \right| \quad (13)$$

Regla de prioridad $\delta_j^{(4)}$: (promedio, máximo)

$$\delta_j^{(4)} = \lambda \times \left| \sum_{k=1}^{K} \bar{p}_{jk} - d_j \right| + (1-\lambda) \times \left| \sum_{k=1}^{K} s_{jk}^{+} - d_j \right| \quad (14)$$

Modelos y Métodos de Solución



Modelos y Métodos de Solución: NEH

Algoritmo de Nawaz-Enscore-Ham

Algorithm 1: Algoritmo NEH

Input : π given by $\delta_j^{(u)}$

1 Ordenar de forma descendente $\delta_j^{(u)}$ para obtener una secuencia $\pi_0 = (\pi_1, \pi_2, \ldots, \pi_n)$

2 Construir ambas subsecuencias $\pi^{(1)} = (\pi_1, \pi_2), \pi^{(2)} = (\pi_2, \pi_1)$

3 $\pi_{NEH} \leftarrow best(\pi^{(1)}, \pi^{(2)})$

4 **for** $j \leftarrow 3$ **to** n **do**

5 Probar el trabajo π_j insertándolo en diferentes posiciones de la soluciones construida actual y guardar en $\pi^{(b)}$ la que tenga menor tardanza total

6 $\pi_{NEH} \leftarrow \pi^{(b)}$

7 **end**

Output: π_{NEH}

- Construye una secuencia agregando trabajos sucesivamente y evaluando múltiples inserciones dentro de la secuencia parcialmente construida

Takeaway

- Es usado en la inicialización
- Genera una mejora sobre las 4 permutaciones iniciales $\pi_i \qquad i = 1, 2, 3, 4$
- Elige la que tiene menor Tardiness.

Modelos y Métodos de Solución: GRASP

Greedy Randomized Adaptive Search Procedure

Algorithm 2: Algoritmo GRASP

Input : π given by $\delta_j^{(u)}$

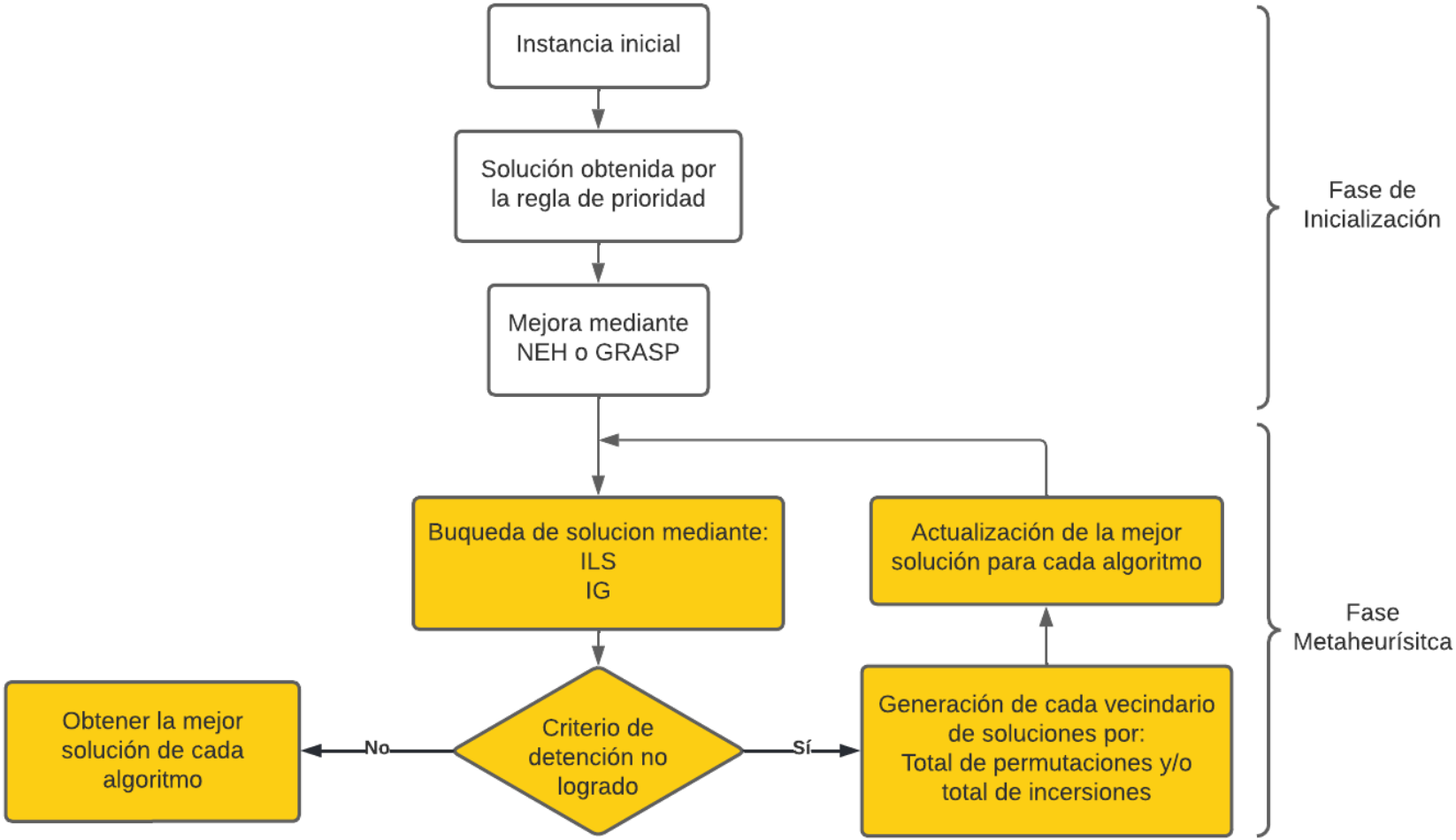
1 Ordenar de forma descendente $\delta_j^{(u)}$ para obtener una secuencia $\pi_0 = (\pi_1, \pi_2, \ldots, \pi_n)$
2 $\Gamma \leftarrow \pi_0$
3 $\pi \leftarrow \emptyset$
4 Evaluar la tardanza total para cada trabajo $TT(\pi_j), j = 1, \ldots, n$
5 **while** $\Gamma \neq \emptyset$ **do**
6 $\quad TT_{min} \leftarrow min\{TT(\pi_j), \pi_j \in \Gamma\}$
7 $\quad TT_{max} \leftarrow max\{TT(\pi_j), \pi_j \in \Gamma\}$
8 $\quad$ Construir lista RCL:
9 $\quad$ RCL$\leftarrow \{\pi_j \in \Gamma, TT_{\pi_j} \in [TT_{min}, TT_{min} + \rho \times (TT_{max} - TT_{min})]\}$
10 $\quad$ Seleccionar un elemento π_r aleatorio de RCL y escoger la posición que de el menor valor de tardanza total en la solución construida $\pi \leftarrow \pi \cup \pi_r$ y actualizar Γ
11 **end**
Output: π

- Construye una solución factible cuya vecindad se analiza hasta encontrar un mínimo local

Takeaway

- Es usado en la inicialización
- Genera una mejora sobre las 4 permutaciones iniciales $\pi_i \qquad i = 1, 2, 3, 4$
- Elige la que tiene menor Tardiness.

Modelos y Métodos de Solución



Vecindarios

Dada una solución $\pi = (7, 6, \mathbf{3}, \mathbf{5}, \mathbf{4}, \mathbf{2}, 1)$ con $p = 3$ y $q = 6$ se definen los siguientes vecindarios.

Vecindario $\mathcal{N}_1$

Se seleccionan dos valores, p y q con $p < q$ y se hace una inserción con desplazamiento a la izquierda.

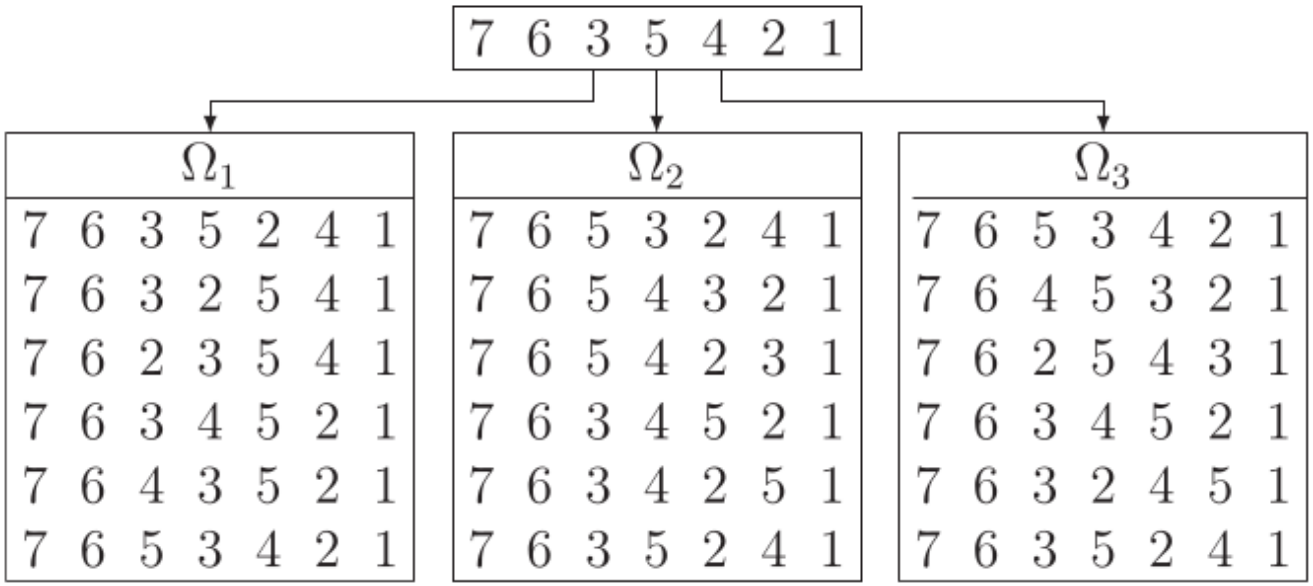
Vecindario $\mathcal{N}_2$

Se seleccionan dos valores, p y q con $p < q$ y se hace una inserción con desplazamiento a la derecha.

Vecindario $\mathcal{N}_3$

Se seleccionan dos valores, p y q con $p < q$ y se hace una permutación con intercambio de posición sin repetición.

Vecindarios



Iterative Local Search

- Algoritmo de simple implementación y gran rendimiento en problemas de scheduling.
- Está basado en dos fases, la primera es explorar el vecindario tras una perturbación y la segunda fase consiste en aceptar o rechazar la nueva solución de acuerdo a un criterio.
- En este caso, los autores usan un parámetro de temperatura, como en el Simulated Annealing:

$$T_e = \mu_0 \frac{\sum_{k=1}^{K} \sum_{j=1}^{n} \min_{1 \leq i \leq m_k} (p_{ijk} + s_{jjk})}{10Kn}$$

Algorithm 3: Algoritmo ILS

Input : π obtenido por NEH o GRASP

$\pi^* \leftarrow \pi$
$TT^* \leftarrow TT(\pi)$
while *Criterio de detención no cumplido* **do**
  Generar la mejor secuencia π' del vecindario $\Omega_k(\pi)$ de π y evaluar $TT\pi'$
  if $TT(\pi') \leq TT(\pi)$ **then**
    $\pi \leftarrow \pi'$
    if $TT(\pi') \leq TT(\pi^*)$ **then**
      $\pi^* \leftarrow \pi'$
  else
    if $rand() \leq e^{-\frac{TT(\pi')-TT(\pi)}{T_e}}$ **then**
      $\pi \leftarrow \pi'$
end

Output: π^*

Iterative Greedy

- Método basado en la aplicación iterativa de procedimientos de construcción de soluciones.
- Se genera una solución mediante métodos constructivos.
- Dicha solución se deconstruye para generar una nueva solución parcial.
- Mediante procesos de aceptación se evalua si dicho proceso debe volver a repetirse, o aceptar la mejor solución obtenida.

IG: Algoritmo

Algorithm 4: IG algorithm

input : π obtained by NEH or GRASP

$\pi^* \leftarrow \pi, TT^* \leftarrow TT(\pi)$

while *Stopping criterion is not satisfied* **do**

  $\Gamma \leftarrow \emptyset$, $\pi' \leftarrow \pi$

  for $h = 1$ *to* d **do**

    Extract a random job π_h *from the* π' *and add it to* Γ *subset*

  end

  for $h = 1$ *to* $|\Gamma|$ **do**

    Extract the job π'_h *from the subset* Γ *and test the job on the different positions in the* π' *and choose the best position giving the best(TT),* $\tilde{\pi}$ *is the best sequence*

  end

  Generate all neighboring solutions $\Omega_g(\tilde{\pi})$ *of the* N_g

  for $u = 1$ *to* $|\Omega(\tilde{\pi})|$ **do**

    $\pi'' \leftarrow$ *the best sequence obtained by the neighborhood* $\Omega_g(\tilde{\pi})$

  end

  if $TT(\pi'') \leqslant TT(\tilde{\pi})$ **then**

    $\tilde{\pi} \leftarrow \pi''$

  end

  if $TT(\tilde{\pi}) \leqslant TT(\pi)$ **then**

    $\pi \leftarrow \tilde{\pi}$

    if $TT(\tilde{\pi}) \leqslant TT(\pi^*)$ **then**

      $\pi^* \leftarrow \tilde{\pi}$

    end

  else

    if $rand() \leqslant e^{-\left(\frac{TT(\tilde{\pi}) - TT(\pi)}{T_e}\right)}$ **then**

      $\pi \leftarrow \tilde{\pi}$

    end

  end

end

output: π^*

Diagrama

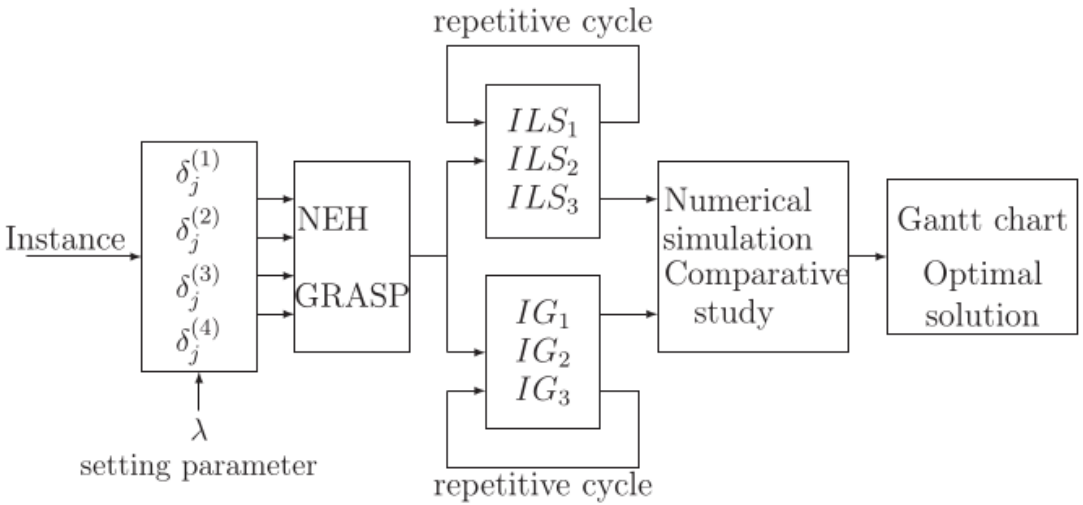

repetitive cycle
Instance
$\delta_j^{(1)}$
$\delta_j^{(2)}$
$\delta_j^{(3)}$
$\delta_j^{(4)}$
λ
setting parameter
NEH
GRASP
ILS_1
ILS_2
ILS_3
IG_1
IG_2
IG_3
repetitive cycle
Numerical simulation Comparative study
Gantt chart
Optimal solution

Tabla de Contenidos

Descripción de Instancias

- Se genera un set de 1080 instancias de distintos tamaños de problemas.
- Se clasifican en instancias relativamente pequeñas y relativamente grandes, según el número de trabajos.

Relativamente Pequeñas

- $n \in \{10, 20, ..., 90\}$
- $m_k \in [2, 6]$
- $K \in [2, 6]$
- $p_{ijk} \in [1, 49]$
- $s_{jhk} \in [1, 20]$

Relativamente Grandes

- $n \in \{100, 120, ..., 200\}$
- $m_k \in [6, 12]$
- $K \in [6, 10]$
- $p_{ijk} \in [50, 99]$
- $s_{jhk} \in [10, 30]$

Descripción Instancias

Además, los deadlines son definidos por:

$$d_j = \omega \sum_{k=1}^{K} \left(\max_{1 \leq i \leq m_k} (p_{ijk} + s_{jjk}) \right)$$

Donde ω es una variable arbitraria para el tamaño de la instancia que está entre 1 y 3 para las relativamente pequeñas, y entre 4 y 6 para las relativamente grandes.

Evaluar Resultados

Para comparar los resultados, se calcula el promedio relativo porcentual de la desviación ('Average Relative Percentaje Deviation')

$$ARPD = \frac{1}{R}\sum_{r=1}^{R}(\frac{TT^{r}_{algo} - TT_{best}}{TT_{best}})x100\,\%$$

- TT^{r}_{algo} representa el valor del tardiness total para el algoritmo usado
- TT_{best} corresponde al menor tardiness encontrado en el set de algoritmos.
- R es el número de algoritmos repetidos para cada problema.

Resultados: Resultados en Instancias

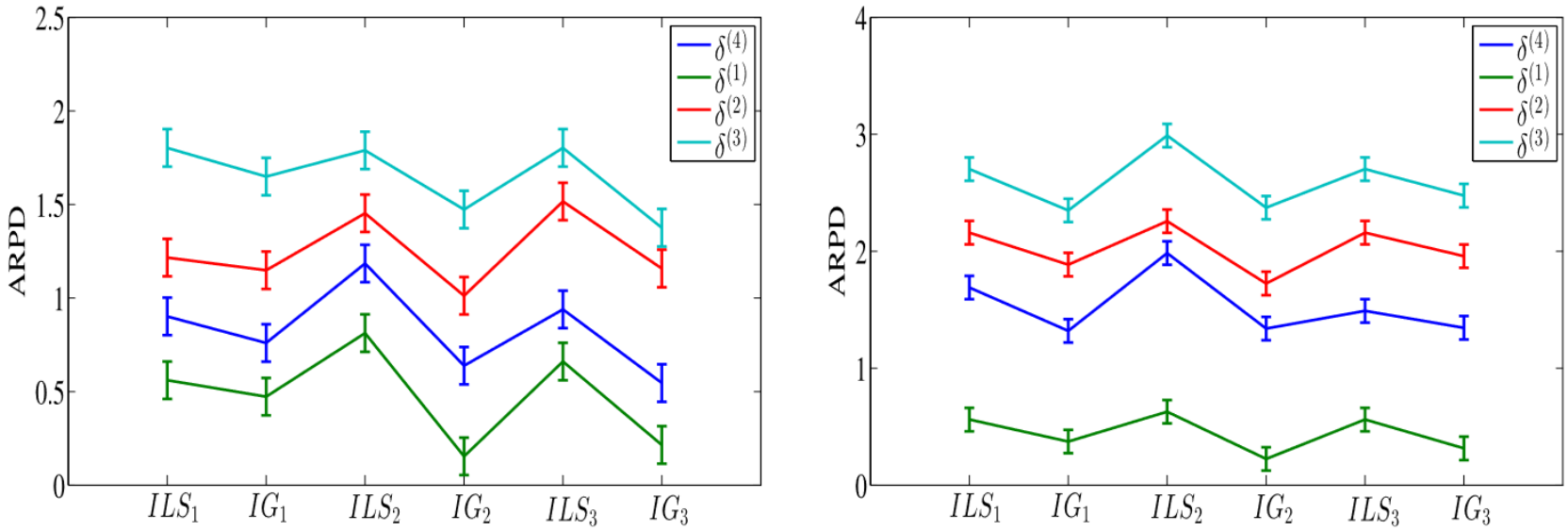
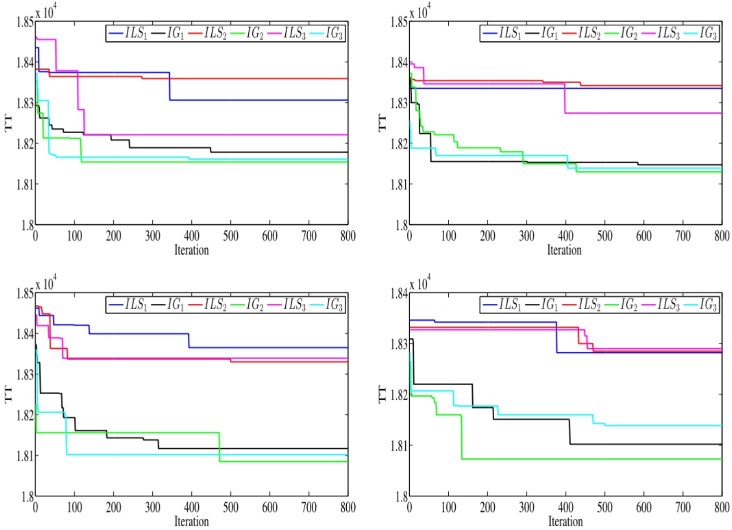


Figura 1: Izquierda: NEH, Derecha: GRASP

Resultados: GRASP



Resultados: NEH

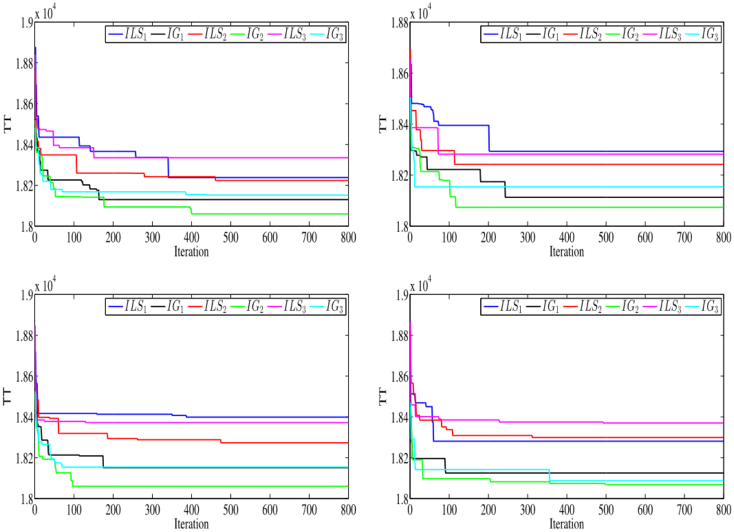


Tabla de Contenidos

Conclusión

- Se revisó el SDST/HFS propuesto.
- Se estudió el problema del HFS.
- La importancia de las metaheurísticas en problemas $NP - Hard$.
- Reglas de inicialización como NEH y GRASP.
- 6 metaheurísticas diferencias por vecindades y algoritmos principales.
- Probado en 1080 instancias pequeñas y grandes.
- Iterative Greedy con vecindad 2 entregó buenos resultados en terminos de calidad y convergencia.
- Limitaciones: 10 etapas.
- Trabajos futuros: Multiobjetivo, costo de producción, consumo de energía, máquinas deshabilitadas, etc.

Tabla de Contenidos

Scheduling hybrid flowshop with sequence-dependent setup times and due windows to minimize total weighted earliness and tardiness

- Propuesto por Ankit Khare y Sunil Agrawal en 2019.
- Hybrid flowshop con tiempos de setup dependientes de la secuencia
- Metaheurísticas basadas en solución única
  - IG
  - ILS
- Metaheurísticas basadas en poblaciones
  - Squirrel search algorithm (SSA)
  - Whale optimization algorithm (WOA)
  - Grey wolf optimization (GWO)
- Ventanas de entrega
- Mejoras mediante busquedas locales (VNS)

Multi-objective scheduling in hybrid flow shop: Evolutionary algorithms using multi-decoding framework

Takeaway

- Plantea un SDST HFS con maquinas no relacionadas.
- Multi-Objetivo: Minimiza el TT y Total Set Up Time
- Resuelve el problema con Algoritmos Evolutivos: Usan un MDF: Multi-decoding Framework

1. Validan la eficiencia con resultados numéricos
2. Enfoque novedoso al usar MDF con EA.
3. El MDF genera un espacio de búsqueda muy grande, extendiendo el tiempo de procesamiento

Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem

- Weishi Shao, Zhongshi Shao y Dechang Pi en 2020.
- HFS sin SDST.
- Busca Minimizar Makespan mediante la implementación de DNEH e IG
- Interesante enfoque en HFS Paralelos
- Análisis numérico de soluciones
- Gran enfoque en la generación de diversidad en soluciones

Referencias I

[AA20] Said Aqil y Karam Allali. «Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing». En: *Expert Systems with Applications* 162 (2020), pág. 113716. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.113716. URL: https://www.sciencedirect.com/science/article/pii/S0957417420305406.

[BF11] J. Behnamian y S.M.T. Fatemi Ghomi. «Hybrid flowshop scheduling with machine and resource-dependent processing times». En: *Applied Mathematical Modelling* 35.3 (2011), págs. 1107-1123. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2010.07.057. URL: https://www.sciencedirect.com/science/article/pii/S0307904X10003069.

Referencias II

[Gra+79] R.L. Graham y col. «Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey». En: *Discrete Optimization II*. Ed. por P.L. Hammer, E.L. Johnson y B.H. Korte. Vol. 5. Annals of Discrete Mathematics. Elsevier, 1979, págs. 287-326. DOI: https://doi.org/10.1016/S0167-5060(08)70356-X. URL: https://www.sciencedirect.com/science/article/pii/S016750600870356X.