Método Constructivo Greedy Determinista para Minimizar el Tiempo Máximo $\max(W_j)$ en Sistemas Put-to-Light (PTL)

1. Contexto y Conceptos Fundamentales

Un sistema Put-to-Light (PTL) se utiliza para clasificar pedidos de manera ágil y ordenada en un entorno logístico. Cada pedido agrupa uno o varios productos y debe asignarse a una posición (llamada "salida") dentro de alguna zona del PTL.

1.1. Zonas y Posiciones

- El sistema PTL se compone de varias zonas: $\{Z_1, Z_2, \ldots, Z_j\}$.
- Cada zona Z_j posee un número limitado de posiciones que pueden ocuparse. Denotamos ese límite por n_{s_j}.
- Las posiciones (o salidas) se representan como $\{S_1, S_2, \ldots, S_k\}$. Cada posición S_k pertenece a exactamente una zona Z_j. Para indicar esto, se define un parámetro binario $s_{j,k}$ tal que:
  - $s_{j,k} = 1$ si la posición k está en la zona j.
  - $s_{j,k} = 0$ en caso contrario.

1.2. Pedidos

Se tiene un conjunto de pedidos $\{P_1, P_2, \ldots, P_i\}$. Cada pedido:

- Representa un conjunto de productos, o un encargo de un cliente.
- Necesita una única posición (salida) para ser clasificado.
- No se puede dividir un pedido: todo el pedido P_i va a una sola posición S_k.

1.3. Tiempo de Asignación tiempo(i, k)

Asignar el pedido P_i a la posición S_k necesita cierto tiempo adicional, el cual puede considerarse como la sumatoria de:

1. Tiempo de manipulación (escaneo de productos por ejemplo).
2. Distancia interna en la zona (traslado de ida y vuelta para depositar los productos).

Ese tiempo se denota tiempo(i, k). Lo calculamos según la ecuación (7) del archivo "Problem Definition.pdf", e ignoramos la asignación a trabajadores específicos pero integramos los SKUs y las distancias:

$$\text{tiempo}(i,k) \;=\; \sum_{m \in \text{SKUs de i}} \Big[t_{r_{i,m}} + 2\big(\tfrac{d_{j,k}}{v}\big) \Big]$$

donde:

- $t_{r_{i,m}}$ es el tiempo de manipular la referencia m en el pedido i.
- $d_{j,k}$ es la distancia en la zona j hasta la posición k.
- v es la velocidad de desplazamiento.
- El "2" es porque se considera el trayecto de ida y vuelta para depositar los productos.

2. Objetivo: Minimizar $\max(W_j)$

Para balancear el sistema, lo que nos interesa es que la zona más cargada no quede excesivamente saturada. Por ello, definimos:

- W_j: tiempo total (o carga total) que acumula la zona j. Una vez se asignan todos los pedidos, la zona j suma los tiempos de aquellos pedidos que cayeron en sus posiciones.
- La función objetivo es minimizar $\max(W_j)$, es decir, hacer que la zona más sobrecargada quede lo menos saturada posible.

Restricciones

- Un pedido no se parte: cada P_i va a una sola posición S_k.
- Una posición no se repite: si P_i usa la posición S_k, ningún otro pedido puede tomar S_k.
- En cada zona Z_j, no se puede ocupar más de n_{s_j} posiciones con diferentes pedidos.

3. Descripción del Método Constructivo (Greedy Determinista)

El método constructivo se denomina "greedy" o "codicioso" porque, en cada decisión (asignar un pedido), busca la mejor opción local para cumplir con el objetivo de mantener bajo $\max(W_j)$. Una vez hecha la asignación, no se desasigna ni se reconsidera.

3.1. Paso 1: Ordenar los Pedidos

Antes de la asignación, ordenamos los pedidos de mayor a menor tiempo promedio. Por ejemplo:

1. Para cada pedido P_i, calculamos su tiempo promedio $\left(\frac{1}{|\text{salidas}|}\sum_k \text{tiempo}(i,k)\right)$.
2. Ordenamos de mayor a menor. Así colocamos primero los pedidos más costosos (en términos de tiempo) para controlar su efecto en $\max(W_j)$.

3.2. Paso 2: Asignar cada Pedido

Tomando los pedidos en ese orden:

- Para el pedido que estamos asignando, se consideran todas las posiciones (salidas) k que estén libres (no usadas) y cuya zona Z_j aún no haya llegado al límite (n_{s_j}).
- Simulamos asignar P_i a cada salida k. Esto implica sumar $\text{tiempo}(i,k)$ en la zona correspondiente j.
- Calculamos cómo cambiaría $\max(W_j)$ con esa asignación temporal.
- Se elige la posición k^* que menos incremente $\max(W_j)$.
- Se fija esa asignación. Esto significa:
  - Marcar esa salida k^* como ocupada,
  - Sumar $\text{tiempo}(i,k^*)$ de forma permanente a W_j.
  - Aumentar en 1 el conteo de posiciones ocupadas en la zona j.

3.3. Paso 3: Continuar hasta Agotar Pedidos

Se repite el proceso para el siguiente pedido hasta haber cubierto todos. Al final, la zona j tiene un valor W_j que es la suma de los tiempos de clasificación de todos los pedidos que terminaron en las posiciones pertenecientes a Z_j.

4. Pseudocódigo del Método Greedy Determinista

ALGORITMO `build_greedy_minmax()`**:**

1. **LEER los datos del problema:**
   - `pedidos`: lista de pedidos (p.ej. [P1, P2, ...]).
   - `salidas`: lista de posiciones (p.ej. [S1, S2, ...]).
   - `zonas`: lista de zonas (p.ej. [Z1, Z2, ...]).
   - `s`$[(j, k)]$: 1 si la salida k pertenece a la zona j, 0 si no.
   - `n_sal`$[j]$: número máximo de posiciones ocupables en la zona j.
   - `tiempo`$[(i, k)]$: tiempo de asignar el pedido i a la salida k, de acuerdo con la ecuación (7) (ignorando parte de trabajadores).
2. **INICIALIZAR:**
   - $X = \{\}$ (diccionario de asignación, donde $X[i] = k$)
   - `used_positions` $= \emptyset$ (conjunto de salidas ya ocupadas)
   - `zone_usage`$[j] = 0$ (para cada zona j, cuántos pedidos se han asignado)
   - $W[j] = 0.0$ (para cada zona j, la suma de tiempos o "carga total")
3. **DEFINIR función** `tiempo_promedio_pedido`(i)**:**
   - `total` $= 0$
   - **para** cada salida k en `salidas`:

     $$\texttt{total} \leftarrow \texttt{total} + \texttt{tiempo}[(i, k)]$$

   - **retornar**: `total` / `(número_de_salidas)`
4. **ORDENAR la lista de pedidos:**
   - `pedidos_ordenados` = ORDENA(`pedidos`) *según* `tiempo_promedio_pedido`(i) en orden DESCENDENTE.
   - (Los de mayor tiempo promedio primero).
5. **PARA cada pedido** i **en** `pedidos_ordenados`**:**

   (5.1) `best_k` = NULO, `best_valor` $= +\infty$

   (5.2) **PARA cada salida** k **en** `salidas`**:**
   - Si k **no** está en `used_positions`:
     - `zona_k` = ZONA a la que pertenece k (donde `s`$[($`zona_k`$, k)] = 1$)

– **si** $\texttt{zone\_usage}[\texttt{zona\_k}] < \texttt{n\_sal}[\texttt{zona\_k}]$:

- *Probar asignación temporal* $i \to k$:

$$\texttt{old\_w} \leftarrow W[\texttt{zona\_k}]$$

$$W[\texttt{zona\_k}] \leftarrow W[\texttt{zona\_k}] + \texttt{tiempo}[(i,k)]$$

$$\texttt{new\_max} = \max\big(W[j] \mid j \in \texttt{zonas}\big)$$

- **Si** $\texttt{new\_max} < \texttt{best\_valor}$:

$$\texttt{best\_valor} = \texttt{new\_max}, \quad \texttt{best\_k} = k$$

- *Revertir* asignación:

$$W[\texttt{zona\_k}] \leftarrow \texttt{old\_w}$$

(5.3) **SI** $\texttt{best\_k} \neq$ NULO:

- $X[i] = \texttt{best\_k}$
- $\texttt{used\_positions}$.agregar($\texttt{best\_k}$)
- $\texttt{zona\_best}$ = ZONA a la que pertenece $\texttt{best\_k}$
- $W[\texttt{zona\_best}] \leftarrow W[\texttt{zona\_best}] + \texttt{tiempo}[(i, \texttt{best\_k})]$
- $\texttt{zone\_usage}[\texttt{zona\_best}] \leftarrow \texttt{zone\_usage}[\texttt{zona\_best}] + 1$

(5.4) **EN CASO CONTRARIO**:

- No se encontró salida factible para ese pedido (se deja sin asignar)

6. **AL final**, calcular:

$$\texttt{max\_time} = \max\big(W[j] \mid j \in \texttt{zonas}\big).$$

7. **RETORNAR** (X, W, $\texttt{max\_time}$).

5. Limitaciones

- **No garantiza la solución óptima**: Las decisiones tempranas pueden condenar la asignación final a un resultado subóptimo; no hay "reacomodo" posterior.
- **Puede ser muy sensible** al **orden de pedidos**: Asignar grandes pedidos primero a una zona puede saturarla, y a veces un orden distinto generaría un mejor resultado.
- **No reoptimiza**: Una vez asignado un pedido, no se retira aunque pudiera mejorar globalmente.