Entrega 2: Optimización de Turnos y Análisis de Resultados

INF292 – Optimización

Campus San Joaquín – Grupo 2

Integrantes:

Fernanda López Saldías	202373638-4
Matías Romo Vargas	202373632-5
Simón Parra Saldías	202373560-4
Vicente Jiménez Sepúlveda	202373523-k

Octubre 2025

1. Modelo Matemático

Conjuntos, Parámetros y Variables

Conjuntos e índices:

T: Trabajadores, indexados por $i \in T$.

D: Días del horizonte de planificación, $D = \{1, 2, \ldots, H\}$, con $j \in D$.

S: Turnos disponibles por día, indexados por $t \in S$.

W_k: Días del fin de semana de la semana k, donde $W_k = \{\text{sábado}, \text{domingo}\}$, y $K = \left\lfloor \frac{H}{7} \right\rfloor$.

Parámetros:

$c_{i,j,t}$: Disposición de i para el turno t del día j, con valores entre 0 y 10.

$r_{j,t}$: Demanda requerida de trabajadores para el turno t del día j.

$a_{i,j,t}$: Disponibilidad binaria, definida como:

$$a_{i,j,t} = \begin{cases} 1 & \text{si } c_{i,j,t} > 0, \\ 0 & \text{si } c_{i,j,t} = 0. \end{cases}$$

Variables de decisión:

$x_{i,j,t}$: Binaria. 1 si el trabajador i es asignado al turno t del día j.

$y_{i,k}$: Binaria. 1 si el trabajador i realiza al menos un turno en el fin de semana k.

Función Objetivo

$$\max Z = \sum_{i \in T} \sum_{j \in D} \sum_{t \in S} c_{i,j,t} \cdot x_{i,j,t}$$

Interpretación: Se maximiza la disposición total del personal asignado, favoreciendo la asignación de turnos a quienes tienen mayor disposición declarada. Esto permite una planificación más eficiente y respetuosa con las preferencias individuales.

Restricciones

$$\text{(R1)} \quad \sum_{i \in T} x_{i,j,t} = r_{j,t} \quad \forall j \in D,\ t \in S$$

$$\text{(R2)} \quad x_{i,j,t} \leq a_{i,j,t} \quad \forall i, j, t$$

$$\text{(R3)} \quad \sum_{t \in S} x_{i,j,t} \leq 2 \quad \forall i, j$$

$$\text{(R4)} \quad x_{i,j,\text{noche}} + x_{i,j+1,\text{mañana}} \leq 1 \quad \forall i,\ j < H$$

$$\text{(R5)} \quad y_{i,k} + y_{i,k+1} + y_{i,k+2} \leq 2 \quad \forall i,\ k \leq K - 2$$

$$\text{(R5.1)} \quad x_{i,d,t} \leq y_{i,k} \quad \forall d \in W_k,\ t \in S$$

$$y_{i,k} \leq \sum_{d \in W_k} \sum_{t \in S} x_{i,d,t} \quad \forall i,\ k$$

$$\text{(R6)} \quad x_{i,j,t},\ y_{i,k} \in \{0, 1\} \quad \forall i, j, t,\ k$$

Explicación de las Restricciones

R1: Garantiza que cada turno sea cubierto exactamente por la cantidad requerida de trabajadores.

R2: Asegura que solo se asignen turnos a trabajadores disponibles.

R3: Limita la carga diaria de cada trabajador a un máximo de dos turnos.

R4: Impide asignaciones consecutivas de noche y mañana para respetar los descansos.

R5: Evita que un trabajador trabaje tres fines de semana seguidos, promoviendo el equilibrio.

R5.1: Define cuándo se activa la variable $y_{i,k}$, vinculándola con la actividad real en fines de semana.

R6: Define la naturaleza binaria de las variables de decisión, asegurando que las asignaciones sean discretas.

Notas explicativas

- **Eliminación de tope de carga:** Se descarta la restricción sobre el total de turnos por trabajador planteada en la entrega 1. La carga se regula mediante R3, R5 y la función objetivo.
- **R5 y R5.1:** R5 limita la frecuencia de trabajo en fines de semana; R5.1 define cuándo se activa esa condición. Juntas aseguran control lógico y preciso.
- **R4 en instancias pequeñas:** Si S solo incluye `día` y `noche`, R4 no aplica. Debe considerarse al validar instancias.

2. Generador de Instancias

El generador fue implementado en **Python** y se encuentra en:

`generador/src/Generador_1_Grupo2_OPTI_SJ.py`

Este script genera **cinco instancias por cada tamaño** (*small*, *medium*, *large*), siguiendo los rangos establecidos en el enunciado. Las instancias se almacenan en carpetas separadas dentro de:

`generador/instancias/`

El código fuente completo está disponible en el repositorio de GitHub [1].

Además, el repositorio incluye un script de chequeo que genera resúmenes estadísticos por instancia, útiles para el análisis posterior. Este script complementa el generador y se encuentra en el mismo repositorio citado.

Lógica de Generación y Formato de Salida

- Se definen aleatoriamente los días y trabajadores según el tipo de instancia:
  - *Small*: 5–7 días, 5–15 trabajadores, turnos `dia` y `noche`.
  - *Medium*: 7–14 días, 15–45 trabajadores, turnos `manana`, `tarde`, `noche`.
  - *Large*: 14–28 días, 45–90 trabajadores, mismos turnos que *medium*.
- Para cada día y turno, se genera una demanda $r_{j,t}$ con una distribución normal truncada en cero, con media proporcional al número de trabajadores y turnos, y desviación estándar del 20 %. Luego:
  1. Se limita la demanda por la disponibilidad real del turno (trabajadores con disposición positiva).
  2. Si la suma diaria excede la capacidad máxima (max_turnos_dia $\times$ trabajadores), se escala proporcionalmente.
- Para cada trabajador, día y turno, se genera una disposición $c_{i,j,t} \sim \mathcal{U}\{0, 10\}$.
- Cada instancia se guarda en dos formatos:
  - `.json`: contiene `id_instancia`, `tipo`, `dias`, `trabajadores`, `demanda_dias` y `disposicion`.
  - `.csv`: archivo plano con las disposiciones individuales, con columnas:

    `trabajador, dia, turno, disposicion`

    Este formato permite inspeccionar rápidamente las preferencias de cada trabajador, pero no incluye la demanda ni otros metadatos.

Descripción del Código

El script realiza dos tareas principales:

1. **Inicialización:** Elimina las carpetas existentes en `../instancias/` para evitar duplicados y crea nuevas subcarpetas para cada tipo de instancia.
2. **Generación:** Para cada tipo de instancia, se generan 5 archivos con datos aleatorios. El script permite fijar una semilla manualmente para reproducir instancias específicas, o usar una semilla aleatoria por defecto (`random.seed()`) para obtener variabilidad entre ejecuciones. Para esta entrega, se utilizó la semilla `42`, al igual que en ejecuciones anteriores, con el fin de mantener consistencia en las pruebas. El proceso incluye:
   - Asignación de días y trabajadores dentro de los rangos definidos.
   - Generación de nombres de días con formato `lunes_1`, `martes_1`, etc.
   - Cálculo de demanda por turno usando una distribución normal truncada en cero.
   - Ajuste de demanda por disponibilidad y escalado por capacidad diaria.
   - Generación de disposiciones por trabajador, día y turno.
   - Escritura de archivos `.json` y `.csv`.

Fragmento del Código

Listing 1: Generador de instancias con control de capacidad

```
def generar_instancias(max_turnos_dia=2, dispo_umbral=0):
    random.seed()  # Semilla aleatoria para variabilidad
    semana_dias = ["lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "
        domingo"]

    tipos = ["small", "medium", "large"]
    dias_rangos = [(5, 7), (7, 14), (14, 28)]
    trabajadores_rangos = [(5, 15), (15, 45), (45, 90)]
    turnos_rangos = [["dia", "noche"], ["manana", "tarde", "noche"], ["manana", "tarde"
        , "noche"]]

    id_instancia = 1
    for tipo, dias_rng, trab_rng, turnos in zip(tipos, dias_rangos, trabajadores_rangos
        , turnos_rangos):
        for i in range(5):  # 5 instancias por tipo
            dias = random.randint(*dias_rng)
            trabajadores = random.randint(*trab_rng)
            cantidad_turnos = len(turnos)
            # Generacion de disposicion y demanda omitida por brevedad
```

CHAKA
MATI Y FER

Referencias

[1] Matías Romo Vargas et al. *proyecto-inf292-Generador de Instancias para Asignación de Turnos Hospitalarios*. Accedido en octubre de 2025. 2025. URL: https://github.com/josephors/proyecto-inf292#.