Entrega 2: Solver, 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

Noviembre 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 y Factibilidad

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

`Entrega_2_Grupo2_OPTI_SJ/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:

`Entrega_2_Grupo2_OPTI_SJ/data/`

El código fuente completo está disponible en el repositorio de GitHub [1]. Además, se incluye un script de chequeo que valida condiciones de factibilidad y genera resúmenes estadísticos por instancia.

Lógica de Generación

Se utilizó la semilla 42 para asegurar reproducibilidad. El generador elimina carpetas previas para evitar residuos y garantizar un entorno limpio en cada ejecución.

- Se generan aleatoriamente los días, trabajadores y turnos según el tipo de instancia:
  - *Small*: 5–7 días, 5–15 trabajadores, turnos `día` y `noche`.
  - *Medium*: 7–14 días, 15–45 trabajadores, turnos `mañana`, `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 distribución normal truncada en cero, ajustada por:
  - Disponibilidad efectiva (trabajadores con disposición positiva).
  - Capacidad máxima diaria (max_turnos_dia $\times$ trabajadores).
- Las disposiciones $c_{i,j,t} \sim \mathcal{U}\{0, 10\}$ se asignan por trabajador, día y turno.
- Cada instancia se guarda en dos formatos:
  - `.json`: incluye metadatos, demanda y disposiciones.
  - `.csv`: contiene las disposiciones individuales por fila.

Condiciones de Factibilidad en la Generación

El generador fue diseñado para producir instancias estructuralmente válidas, es decir, con datos completos, consistentes y compatibles con el modelo matemático. Esto implica que cada instancia contiene trabajadores, días, turnos, disposiciones y demandas bien definidas, sin errores lógicos ni violaciones formales.

Además, se incorporan controles que reflejan las restricciones del modelo desde la etapa de generación:

- **R1 (Cobertura):** La demanda por turno se ajusta según la cantidad de trabajadores disponibles con disposición positiva.
- **R3 (Carga diaria):** Se limita la suma de turnos por día, escalando la demanda si excede la capacidad total del equipo.
- **R5 (Fines de semana):** Se evita congestión en sábado y domingo, respetando la carga global del equipo.

Aunque no se incluye una restricción explícita sobre el total de turnos por trabajador (R6), la carga se regula indirectamente mediante R3, R5 y la función objetivo. Las instancias *small* tienden a ser factibles, mientras que en *medium* y *large* la factibilidad depende de la relación entre demanda y disposición.

Importante: Que una instancia sea estructuralmente válida no implica que sea resoluble. La factibilidad se evalúa posteriormente mediante el modelo LPE, que determina si existe una asignación que cumpla todas las restricciones. Este enfoque permite estudiar cómo influyen los parámetros generados aleatoriamente en la posibilidad de encontrar soluciones válidas, sin alterar artificialmente las instancias para forzar su resolución.

3. Metodología de Implementación del Solver

Esta sección describe la arquitectura del nuevo solver desarrollado usando `lpsolve55`, el flujo de resolución implementado y el mecanismo utilizado para procesar todas las instancias y almacenar sus resultados. Todo el desarrollo se encuentra documentado en `Entrega_2_Grupo2_OPTI_SJ/`, incluyendo los archivos:

`Ejecutor_Solver_lpsolve_2_Grupo2_OPTI_SJ.py` y `Solver_lpsolve_2_Grupo2_OPTI_SJ.py`.

Arquitectura del Sistema

El solver se organiza en dos componentes principales:

- **`Solver_lpsolve_2_Grupo2_OPTI_SJ.py`**: Implementa el modelo matemático directamente en lp_solve 5.5 y resuelve una única instancia.
- **`Ejecutor_Solver_lpsolve_2_Grupo2_OPTI_SJ.py`**: Recorre todas las carpetas de instancias, ejecuta el solver para cada archivo y genera los reportes correspondientes.

Esta separación permite mantener el código modular: un archivo se enfoca en la definición del problema y el otro en la ejecución masiva y la recolección de métricas.

Implementación de la Función Objetivo

El solver utiliza **Programación Lineal Entera**, modelada con `lp_solve 5.5`. La función objetivo maximiza la disposición total:

$$\text{máx } Z = \sum_{i \in T} \sum_{j \in D} \sum_{t \in S} c_{ijt} \cdot x_{ijt}$$

Listing 1: Implementación de la Función Objetivo en lpsolve55

```
# F.O.: Maximizar la suma de disposiciones
obj = [0.0] * nvars
for i in range(num_trabajadores):
    for j in range(num_dias):
        for t in range(num_turnos):
            idx = index_x(i, j, t) - 1
            obj[idx] = disposicion[i][j][t]

lpsolve('set_obj_fn', lp, obj)
```

Implementación de las Restricciones

El solver implementa las restricciones R1–R5.1 tal como se definieron en la especificación del proyecto:

- **R1: Cobertura exacta de demanda por turno.** Cada turno de cada día debe cubrir exactamente la demanda de trabajadores. Esto se implementa sumando todas las variables x_{ijt} correspondientes a un turno y agregando una restricción de igualdad con la demanda:

Listing 2: Restricción R1 en lpsolve55

```
# R1: Cobertura exacta de demanda
for j in range(num_dias):
    for t in range(num_turnos):
        row = fila_cero()
        for i in range(num_trabajadores):
            row[index_x(i,j,t)-1] = 1.0
        lpsolve('add_constraint', lp, row, '=', demanda[j][t])
```

- **R2: Prohibición de asignar turnos con disposición 0.** Si un trabajador no está disponible para un turno específico, se fija $x_{ijt} = 0$:

Listing 3: Restricción R2 en lpsolve55

```
# R2: Prohibicion de turnos no disponibles
for i in range(num_trabajadores):
    for j in range(num_dias):
        for t in range(num_turnos):
            if disposicion[i][j][t] == 0:
                row = fila_cero()
                row[index_x(i,j,t)-1] = 1.0
                lpsolve('add_constraint', lp, row, '=', 0)
```

- **R3: Máximo de 2 turnos por día por trabajador.** Un trabajador no puede estar asignado a más de dos turnos en un mismo día:

Listing 4: Restricción R3 en lpsolve55

```
# R3: Maximo 2 turnos por dia
for i in range(num_trabajadores):
    for j in range(num_dias):
        row = fila_cero()
        for t in range(num_turnos):
            row[index_x(i,j,t)-1] = 1.0
        lpsolve('add_constraint', lp, row, '<=', 2)
```

- **R4: Prohibición de turno noche seguido por turno mañana.** Se evita que un trabajador tenga turno noche seguido de turno mañana al día siguiente:

Listing 5: Restricción R4 en lpsolve55

```
# R4: No noche -> manana consecutiva
for i in range(num_trabajadores):
    for j in range(num_dias-1):
        row = fila_cero()
        row[index_x(i,j,t_noche)-1] = 1.0
        row[index_x(i,j+1,t_man)-1] = 1.0
        lpsolve('add_constraint', lp, row, '<=', 1)
```

- **R5: No trabajar 3 fines de semana consecutivos.** Esta restricción utiliza variables binarias auxiliares w_{ik} que indican si el trabajador i trabaja durante el fin de semana k. Se asegura que la suma de w de tres fines consecutivos no exceda 2:

Listing 6: Restricción R5 en lpsolve55

```
# R5: No trabajar 3 fines de semana consecutivos
for i in range(num_trabajadores):
    for k in range(num_fines-2):
        row = fila_cero()
        row[index_w(i,k)-1] = 1.0
        row[index_w(i,k+1)-1] = 1.0
        row[index_w(i,k+2)-1] = 1.0
        lpsolve('add_constraint', lp, row, '<=', 2)
```

- **R5.1: Enlace bidireccional entre x y w.** Cada w_{ik} se vincula con las variables x_{ijt} correspondientes a los días del fin de semana k:

Listing 7: Restricción R5.1 en lpsolve55

```
# R5.1: Enlace x <= w y w <= Sum x
for i in range(num_trabajadores):
    for k, dias_fin in enumerate(fines_de_semana):
        # x <= w
        for dia_idx in dias_fin:
            j = dia_idx-1
            for t in range(num_turnos):
                row = fila_cero()
                row[index_x(i,j,t)-1] = 1.0
                row[index_w(i,k)-1] = -1.0
                lpsolve('add_constraint', lp, row, '<=', 0)
        # w <= Sum x
        row = fila_cero()
        row[index_w(i,k)-1] = 1.0
        for dia_idx in dias_fin:
            j = dia_idx-1
            for t in range(num_turnos):
                row[index_x(i,j,t)-1] -= 1.0
        lpsolve('add_constraint', lp, row, '<=', 0)
```

Proceso de Ejecución Masiva

El archivo `Ejecutor_Solver_lpsolve_2_Grupo2_OPTI_SJ.py` recorre automáticamente los directorios:

`data/small/`, `data/medium/`, `data/large/`

Para cada archivo JSON:

1. Se carga la instancia.
2. Se construye el modelo en `lp_solve 5.5`.
3. Se ejecuta el solver.
4. Se mide el tiempo de resolución.
5. Se clasifica el resultado como óptimo, factible o infactible.
6. Se guarda un archivo JSON en:

`Entrega_2_Grupo2_OPTI_SJ/resultados/<tamaño>/`

Estructura del Módulo de Métricas

El sistema acumula información global para producir un archivo resumen:

Listing 8: Estructura de Recolección de Métricas

```
resumen_total = {
    "fecha_ejecucion": datetime.now().isoformat(),
    "instancias_procesadas": 0,
    "instancias_optimas": 0,
    "instancias_infactibles": 0,
    "tiempo_total_segundos": 0,
    "resultados_por_tamano": {}
}
```

Ejemplo de Archivo de Resultados

Listing 9: Ejemplo de Resultado JSON


```json
{
  "id_instancia": 11,
  "tipo": "large",
  "dias": 23,
  "trabajadores": 47,
  "estado": "Optimal",
  "valor_objetivo": 9362.0,
  "tiempo_resolucion_segundos": 0.179936,
  "factible": true,
  "asignaciones": [
    {
      "trabajador": 1,
      "dia": 1,
      "dia_nombre": "domingo_1",
      "turno": "noche",
      "disposicion": 8
    }
  ],
  "fecha_resolucion": "2025-11-09T02:00:57.060042"
}
```

Notas Finales

- El sistema está diseñado para ser escalable y completamente automatizado.
- Todos los archivos están documentados en `Entrega_2_Grupo2_OPTI_SJ/solver/`.
- Se elimina explícitamente la restricción R6 (tope global de carga).
- La ejecución de instancias grandes puede demorar más debido a la naturaleza entera del problema.
- Para la utilización del solver es necesario descargar Anaconda, ya que es quien soporta la librería del solver pedido.

4. Resultados y Análisis

Resumen general de la ejecución

El modelo fue resuelto usando la herramienta **LPSolve** a través de un script que procesó las quince instancias generadas (*small*, *medium* y *large*).

El archivo `ejecutar_solver_batch.py` ejecutó el solver para cada instancia y registró los resultados en formato JSON.

En total, se obtuvieron 13 soluciones óptimas y 2 instancias infactibles. Los tiempos de ejecución fueron muy bajos (del orden de milisegundos), confirmando la eficiencia del modelo en instancias pequeñas y medianas.

Tamaño	**Total**	**Óptimas**	**Infactibles**	**Tiempo promedio (s)**
Small	5	5	0	0.000613
Medium	5	5	0	0.009846
Large	5	3	2	0.208286
Total	15	13	2	—

Cuadro 1: Resumen de resultados globales por tipo de instancia.

De la tabla anterior se puede observar que las instancias *small* y *medium* fueron completamente factibles y alcanzaron el óptimo en menos de 0.05 segundos. En cambio, dos de las instancias *large* resultaron infactibles, lo cual se analizará más adelante.

(a) Análisis de la función objetivo

La función objetivo maximiza la disposición total del personal asignado. A medida que el tamaño de la instancia crece (más trabajadores y días), el valor objetivo aumenta de manera proporcional al número de asignaciones posibles, manteniendo así, la coherencia del modelo.

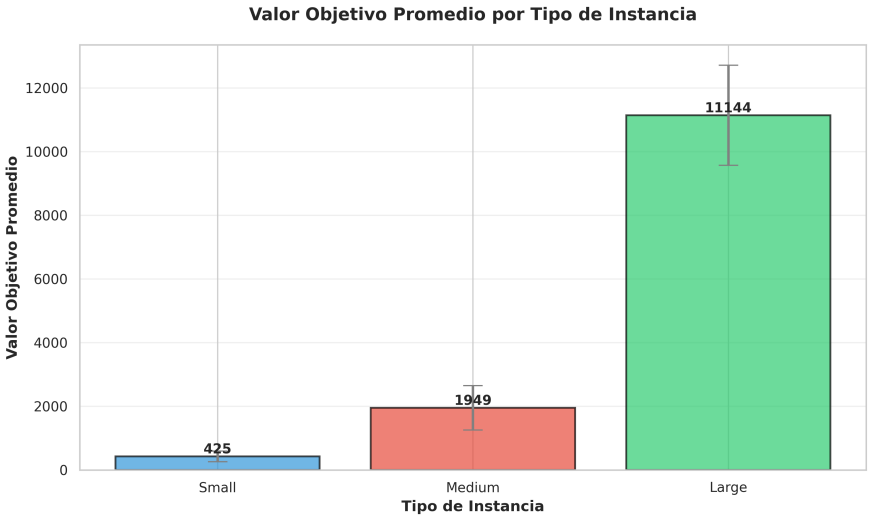

Figura 1: Promedio del valor objetivo por tipo de instancia.

En la Figura anterior se puede apreciar un crecimiento proporcional al tamaño de la instancia del valor objetivo promedio al pasar de *small* a *large*, esto es coherente con el aumento de la demanda y la disponibilidad total de turnos. Lo que demuestra que el modelo logra mantener la consistencia de asignaciones al priorizar trabajadores con mayor disposición.

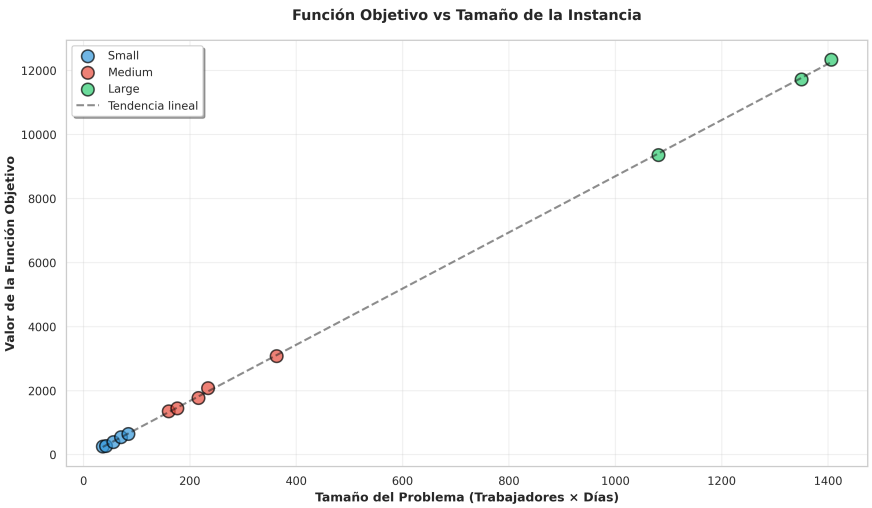

Figura 2: Comportamiento del valor objetivo en función del tamaño (número de variables).

En la Figura 2 se puede observar que la relación entre el valor objetivo y el número de variables muestra una tendencia ascendente con una leve dispersión. Esto refleja que el solver sigue encontrando soluciones óptimas, pero la complejidad combinatoria puede introducir ligeras variaciones en la calidad o el tiempo de resolución. En resumen, el modelo mantiene su desempeño y escalabilidad dentro del rango planificado.

(b) Análisis de infactibilidad

Que una solución sea infactible (o que el modelo es infeasible) significa que no existe ninguna combinación de valores que cumpla todas las restricciones del modelo al mismo tiempo.

En total, dos instancias de tipo *large* fueron declaradas infactibles. Esto puede deberse principalmente a la interacción entre las restricciones de cobertura (R1), carga diaria (R3) y descanso (R4). En ciertas combinaciones aleatorias de disponibilidad y demanda, puede ocurrir que la cantidad de trabajadores dispuestos para cubrir un turno crítico sea insuficiente o que las reglas de descanso hagan imposible una cobertura completa.

- En particular, R4 (*noche* $\rightarrow$ *mañana*) puede reducir la disponibilidad efectiva cuando los turnos nocturnos son muy numerosos.
- R3 (*máx. 2 turnos por día*) restringe aún más la flexibilidad, limitando las posibles reasignaciones.

Es importante destacar que esto no representa un error del modelo, sino una propiedad esperable en escenarios donde la demanda supera la capacidad disponible. De hecho, la existencia de instancias infactibles es útil para validar la robustez del generador y del modelo de optimización.

(c) Visualización tipo calendario para instancias pequeñas

En las instancias pequeñas, el modelo permite visualizar fácilmente la asignación diaria por trabajador. La siguiente figura muestra un ejemplo de calendario generado por el script `Generar_calendarios_2_Grupo2_OPTI_SJ.py`, donde cada celda indica los turnos asignados a cada empleado durante la semana.

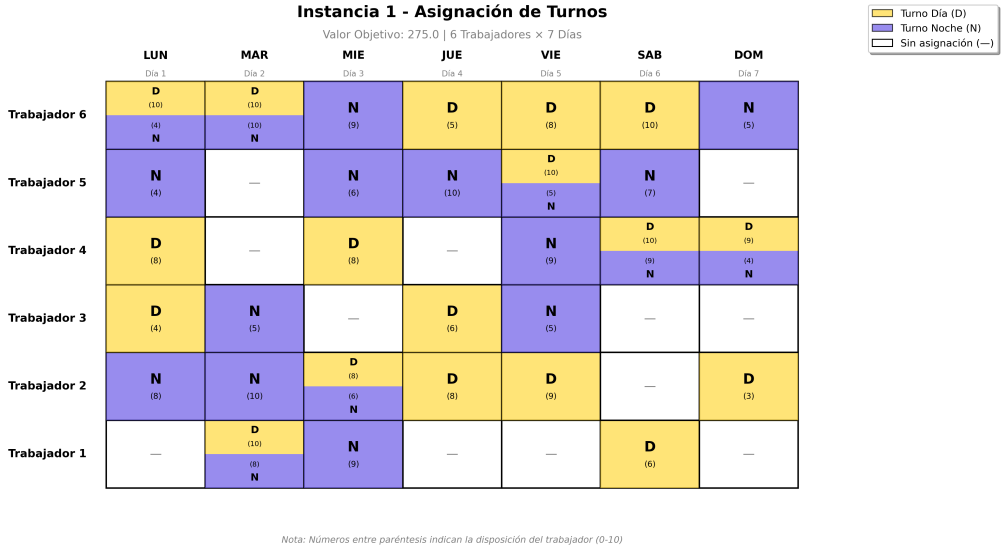

Figura 3: Ejemplo de calendario de asignaciones para una instancia *small*.

En este calendario, se puede observar que las asignaciones respetan las restricciones R3 y R4: ningún trabajador realiza más de dos turnos diarios ni tiene secuencias noche–mañana(teniendo en cuenta que no aplica para instancias small). El modelo logra tener una distribución equilibrada de la carga laboral entre los trabajadores disponibles, maximizando la disposición total.

Análisis de tiempos de resolución

Finalmente, se evaluó el tiempo de resolución promedio y su relación con el tamaño de las instancias y la cantidad de variables involucradas.

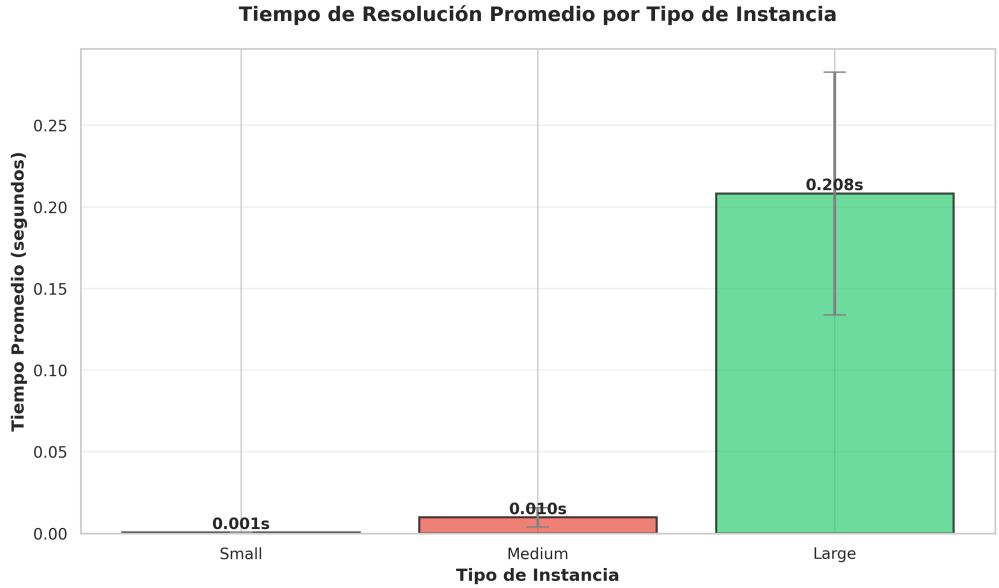

Figura 4: Tiempo promedio de resolución por tipo de instancia.

En el gráfico se puede observar que el tiempo crece de forma proporcional al pasar de instancias pequeñas a grandes. También se puede ver, que incluso en el caso *large*, el tiempo promedio se mantiene bajo (0.2 s), lo que confirma la eficiencia del modelo implementado.

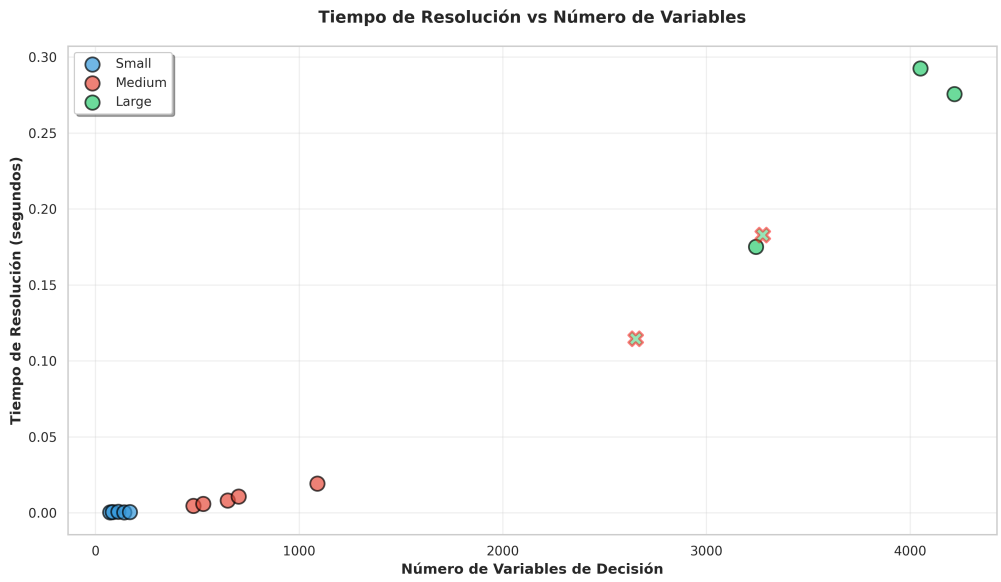

Figura 5: Tiempos de resolución en función del número de variables.

En la figura anterior se puede observar que, aunque el tiempo de resolución tiende a aumentar con el número de variables, este crecimiento no es abrupto ni exponencial, lo que indica que el modelo escala de forma controlada y que el solver maneja de forma eficiente los casos con mayor tamaño. En general, el comportamiento temporal es coherente con la naturaleza del modelo, donde el aumento de combinaciones posibles no afecta de manera drástica la capacidad de resolución gracias a la estructura binaria y a las restricciones acotadas del problema.

Al comparar los resultados de tiempo entre los tres tamaños, se evidencia que la diferencia entre *small* y *medium* es mínima, mientras que el salto hacia *large* representa el aumento esperado en complejidad, pero sin llegar a comprometer la factibilidad ni el rendimiento. Esto demuestra que el modelo es estable y eficiente, incluso cuando la cantidad de trabajadores, días o turnos se multiplica varias veces.

Conclusión general

En conclusión, el modelo propuesto demuestra ser sólido, eficiente y coherente con los objetivos planteados. A lo largo de las distintas instancias, se ha podido confirmar que:

- El modelo mantiene la escalabilidad, incrementando el valor de la función objetivo de forma proporcional al tamaño de la instancia.
- Los tiempos de resolución son reducidos incluso en casos de gran tamaño, evidenciando una implementación eficiente en **LPSolve**.
- Las instancias infactibles se explican por condiciones realistas de sobrecarga de demanda, lo cual logra validar la robustez del modelo.
- Las soluciones factibles respetan todas las restricciones definidas, reflejando una distribución equilibrada de la carga laboral.

Además, el análisis conjunto de los resultados y los tiempos de resolución permite concluir que el modelo no solo optimiza correctamente la asignación de personal, sino que también lo hace con un costo computacional bastante bajo. Esto sugiere que la formulación matemática podría escalar a entornos reales más amplios sin comprometer la eficiencia del proceso.

En resumen, el sistema de generación, modelamiento y resolución cumple con los criterios de calidad, desempeño y factibilidad requeridos, demostrando así, un comportamiento consistente, interpretable y aplicable en contextos reales de planificación de turnos.

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#.