



## Taller 2

# PROGRAMACIÓN POR RESTRICCIONES

Robinson Duque, Ph.D

robinson.duque@correounivalle.edu.co

Octubre de 2025

**Importante:** Esta parte del proyecto de curso está pensada para un total de 18 horas de trabajo (el equivalente a una dedicación de 6 horas de trabajo durante 3 semanas).

## Introducción

En este proyecto aplicarás técnicas de Programación por Restricciones (CSP/COP) para extender el Job Shop Scheduling Problem (JSSP) clásico visto en clase. Tu objetivo será modelar y resolver dos variaciones del problema, incorporando nuevas restricciones y objetivos que reflejen un contexto industrial, logístico o de servicios más realista.

Cada equipo trabajará sobre dos versiones específicas del problema, con condiciones adicionales que requieren modificar el modelo base, adaptar las restricciones y evaluar su impacto en la solución.

## 1. Job Shop

Una fábrica tiene un conjunto de trabajos (jobs), cada uno compuesto por una secuencia de operaciones (tasks) que deben realizarse en máquinas específicas. Cada operación tiene:

- Una duración conocida
- Una máquina asignada
- Una precedencia (una operación no puede comenzar hasta que la anterior del mismo trabajo haya terminado)

Cada máquina puede ejecutar solo una operación a la vez, y el objetivo es planificar las operaciones para minimizar el tiempo total de finalización (makespan).

En clase se trabajó esta versión básica del Job Shop. A partir de ese modelo usted debe **escoger (2) dos variaciones** que incorporen nuevas restricciones, prioridades o contextos y presentar el informe correspondiente. En las siguientes secciones se presentan variaciones del modelo de las cuales podrá realizar su elección:

### 1.1. Job Shop con mantenimiento programado

**Contexto:** Cada máquina debe detenerse eventualmente para mantenimiento preventivo. Durante ese tiempo, no puede procesar operaciones.

#### Nuevas restricciones:

- Cada máquina  $m$  tiene intervalos  $[a_m, b_m]$  donde no está disponible.
- Ninguna operación puede ejecutarse en la máquina  $m$  durante esos intervalos.

**Objetivo:** Minimizar el makespan respetando los períodos de mantenimiento.

### 1.2. Job Shop con operarios limitados

**Contexto:** Algunas máquinas requieren un operario especializado, pero hay menos operarios que máquinas.

#### Nuevas restricciones:

- Un conjunto limitado de  $k$  operarios puede atender las máquinas.
- Dos operaciones simultáneas no pueden usar el mismo operario.
- Asignar cada operación a un operario disponible.

**Objetivo:** Minimizar el makespan y balancear la carga de trabajo de los operarios.

### 1.3. Job Shop con prioridades (pesos) y fechas límite

**Contexto:** Cada trabajo tiene una prioridad distinta  $w_i$ . Los trabajos deben completarse antes de cierta fecha  $due\_date_i$ .

#### Nuevas restricciones:

- Cada job  $i$  tiene una fecha límite  $due\_date_i$
- Penalizar las operaciones que terminan después de su fecha límite.

**Objetivo:** Minimizar la suma ponderada de tardanzas  $\sum w_i * \max(0, end_i - due\_date_i)$ .

## 2. Informe con Modelos y Conclusiones

### 2.1. Respecto a los modelos:

- Proponer un formato de entrada de datos **Datos.dzn** que permita configurar los parámetros de los modelos seleccionados.
- Generar 10 instancias para cada modelo con diferentes configuraciones.
- Proponga un modelo genérico para solucionar cada problema seleccionado (i.e., parámetros, variables, restricciones, función objetivo).
- Implemente cada modelo genérico en MiniZinc (ejemplo: **Job Shop-mantenimiento.mzn**).
- Incluya en el informe una tabla con pruebas realizadas sobre las 10 instancias creadas anteriormente y realice un análisis sobre los resultados obtenidos.

### 2.2. Respecto al informe:

El grupo deberá entregar un informe en formato pdf, que contenga, al menos, los siguientes aspectos, por cada uno de los ejercicios desarrollados:

- El modelo: una descripción del modelo (parámetros, variables, restricciones, anotaciones de búsqueda) y una justificación de su adecuación al problema planteado.
- Detalles importantes de implementación: lo más relevante de la implementación, sin incluir código (por ejemplo restricciones redundantes y rompimiento de simetrías en caso de encontrar alguna).
- La búsqueda: La descripción de las diferentes estrategias de búsqueda exploradas.
- Pruebas: descripción de las pruebas realizadas a su implementación.
- Análisis: del modelo, de las pruebas, de las estrategias de búsqueda.
- Conclusiones: Esta es una de las partes más interesantes del trabajo (pero no por ello la que más vale). En ella se espera que usted analice los resultados obtenidos y justifique claramente sus afirmaciones.

## 3. Evaluación

La evaluación de cada taller se hará de acuerdo al número de ejercicios desarrollados, a la claridad y calidad de los informes y al funcionamiento de las implementaciones. En específico se calificará según los siguientes criterios:

1. **Definición de variables de los modelos (10):** Se espera que las variables de los modelos estén precisamente definidas (su significado), y sus dominios estén precisamente definidos. Nunca se confunden variables con parámetros. Además de estar precisamente definidos, los dominios de las variables son apropiados para los problemas que modelan y no contienen valores que desde el inicio se sabe que no pueden tomar.
2. **Definición de las restricciones de los modelos (15):** Se espera que al definir las restricciones de los problemas, se tengan en cuenta todas las restricciones mencionadas en el problema, modelando todos los aspectos relevantes. Las restricciones se definen independientemente de la implementación, en un lenguaje formal preciso y claro, y se justifica su corrección o adecuación, cuando esta no es evidente. Las restricciones, en general, no están condicionadas por los valores que puedan tomar las variables; este tipo de restricciones sólo se dejan en caso de no haber otra opción.
3. **Claridad para imponer restricciones que rompen simetrías (10):** Se espera que cuando el problema lo permita se definan restricciones que rompan simetrías, se expliquen claramente qué simetrías rompen y cómo recuperar las soluciones que se pierden al imponer esas restricciones. Si no hay lugar a restricciones que rompan simetrías, se espera al menos un párrafo argumentando por qué no, o que diga que no lograron definir ninguna.
4. **Claridad para imponer restricciones redundantes (10)** Se espera que cuando el problema lo permita se definan restricciones redundantes, se explique claramente por qué son redundantes y se justifique que con ellas el espacio de búsqueda se poda. Si no hay lugar a restricciones redundantes, se espera al menos un párrafo argumentando por qué no o que diga que no lograron definir ninguna.
5. **Claridad en la escogencia de las mejores estrategias de búsqueda (15):** Se espera que no sólo se hayan explorado los modelos con la estrategia de búsqueda por defecto, sino que se haya reflexionado alrededor del tema de la exploración, y producto de esa reflexión se hayan definido diversas anotaciones (al menos dos, pero pueden ser más) que den lugar a diversas estrategias de búsqueda y que se analicen y comparen dichos comportamientos aprovechando la estadística que provee MiniZinc sobre las búsquedas (nodos explorados, nodos fallidos, tiempo de respuesta, ...).
6. **Calidad de las pruebas (5):** Se espera que cada modelo se haya probado con un suficiente número de pruebas para poder analizar a profundidad el modelo. Por ello este criterio lo que evalúa es si esa batería de pruebas es suficiente para verificar la corrección del modelo y para explorar las diferentes estrategias de búsqueda y compararlas. También se espera que los resultados de las pruebas se sistematicen de alguna manera (tablas, gráficas, . . .) y se presenten en el informe.

7. **Coherencia entre los modelos, las implementaciones y los análisis presentados en el informe (25):** Se espera que todos los modelos descritos hayan sido implementados coherentemente en MiniZinc y que los análisis correspondan con las pruebas de esos modelos en MiniZinc.
8. **Redacción del informe (10):** Se espera que el informe sea completo, claro, preciso, fácil de leer, bien presentado, contenga los análisis más importantes y las conclusiones.

## 4. Instrucciones finales

Se debe entregar una carpeta (en formato zip) de nombre *PPR-CodEstudiante1-CodEstudiante2* que contenga los siguientes archivos:

1. Un archivo **readme.txt** que enumere y describa el contenido de la carpeta. Debe describir cómo correr los modelos.
2. Un informe (en formato PDF) que indique los aspectos más importantes de su modelo y de su implementación.
3. Todos los archivos fuentes (\*.mzn y \*.dzn) con la implementación de los modelos y los datos de prueba.