Ibermática digital
An avesa company

T5.2. Informe Calidad de los Datos

Ibermática – Quantumi3b

[15/07/2023]

XUNTA DE GALICIA
Unión Europea
Unha maneira de facer Europa.
Fondos Europeos
CESGA GALICIA SUPERCOMPUTING CENTER

Despregamento dunha infraestructura baseada en tecnoloxías cuánticas da información que permita impulsar a I+D+i en Galicia.

Apoiar a transición cara a una economía dixital.

Operación financiada pola Unión Europea, a través do FONDO EUROPEO DE DESENVOLVEMENTO REXIONAL (FEDER) como parte da resposta da Unión á pandemia da COVID-19

DATA	AUTOR	CAMBIOS	VERSIÓN
18/09/2023	Ibermática	Inicial	1

Tabla de contenidos

Lista de figuras

Lista de Tablas

No se encuentran elementos de tabla de ilustraciones.

Lista de acrónimos

Quantum-OPS	*Performs Common Linear Algebra Operations Used in Quantum Computing and Implements Quantum Algorithms*
A JSP CRO	*Job-Shop Scheduling Problem*

1. Metodología QCRISP-DM

Para asegurar que el proyecto se realiza de forma adecuada, y con una calidad objetiva, I3b aplica la metodología QRISP-DM a la implantación de los proyectos cuánticos. QRISP-DM es una metodología de desarrollo de sistemas y modelos cuánticos que permite el aseguramiento de la calidad técnica. Los datos deben ser extraídos, depurados y preparados para su uso e interpretación. Este método se divide en 6 fases: conocimiento del negocio, conocimiento de los datos, preparación de los datos, modelización, evaluación y desarrollo.

Figura 1: Esquema general de la metodología QCRISP-DM desarrollada por i3B.

Los enfoques de desarrollo del software cuántico deben ser, de manera similar al clásico, lo más iterativos, incrementales y ágiles posible para proponer la entrega de valor de los desarrollos. Por ello, se ha aplicado una metodología ágil "scrum" en el desarrollo del proyecto, y su despliegue en producción, mediante metodología "Quantum-OPS".

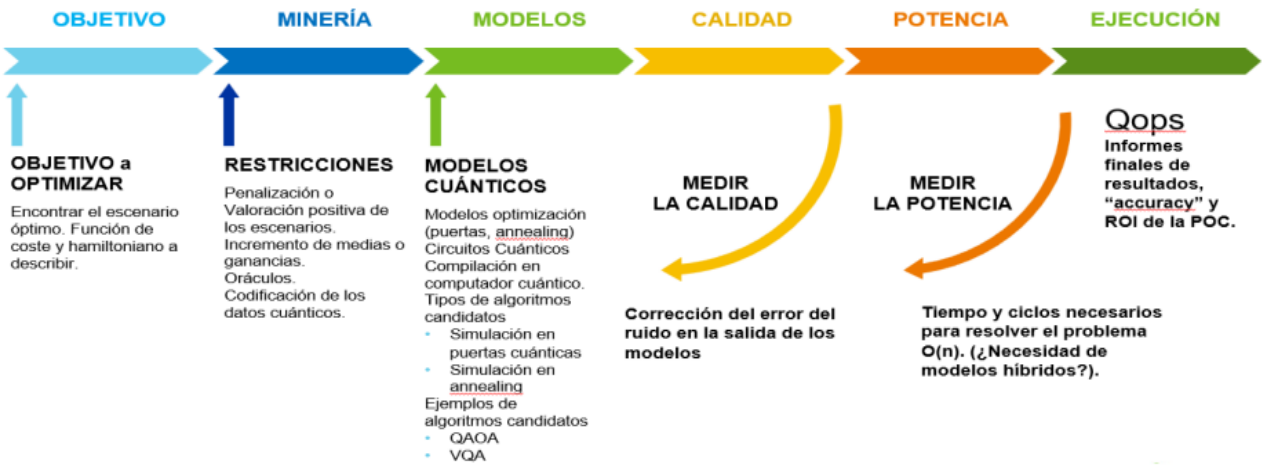

Figura 2: Diagrama de flujos de trabajo dentro de la metodología QCRISP-DM.

La metodología QRISP-DM persigue que en todo momento del proyecto se vayan evaluando una serie de métricas que, desde las primeras semanas del mismo, aseguren los objetivos funcionales, técnicos y de ROI pactados con CESGA.

Conocimiento del negocio

Los requisitos que debe cumplir el caso de uso elegido para este proyecto son dos:

- Ser de interés para los sectores estratégicos de la industria gallega, con datos lo más realistas y cercanos a una implementación de negocio posible, pero manteniéndose anónimos y publicables.
- Servir de plataforma para probar los métodos de separación de circuitos desarrollados por CESGA, presentando un desafío del volumen necesario para ello y permitiendo el uso de técnicas escalables.

Por ello, el problema de optimización elegido para el proyecto es el Job-Shop Scheduling Problem (JSP, también llamado JSSP). Se plantea la distribución de las tareas (tasks) de varios trabajos (jobs) en varios agentes (máquinas, empleados, vehículos...), teniendo como restricción el hecho de que cada uno de los agentes no están capacitados para realizar más que una colección determinada de todas las tareas existentes, y de que para comenzar cada tarea es necesario haber completado previamente todas las anteriores tareas de su mismo trabajo. En la imagen se muestra la solución de una instancia de JSP, con los segmentos representando distintas tareas. Segmentos del mismo color corresponden al mismo trabajo, y es posible identificar la tarea por su posición en el eje temporal en relación con el resto de las tareas de su mismo trabajo.

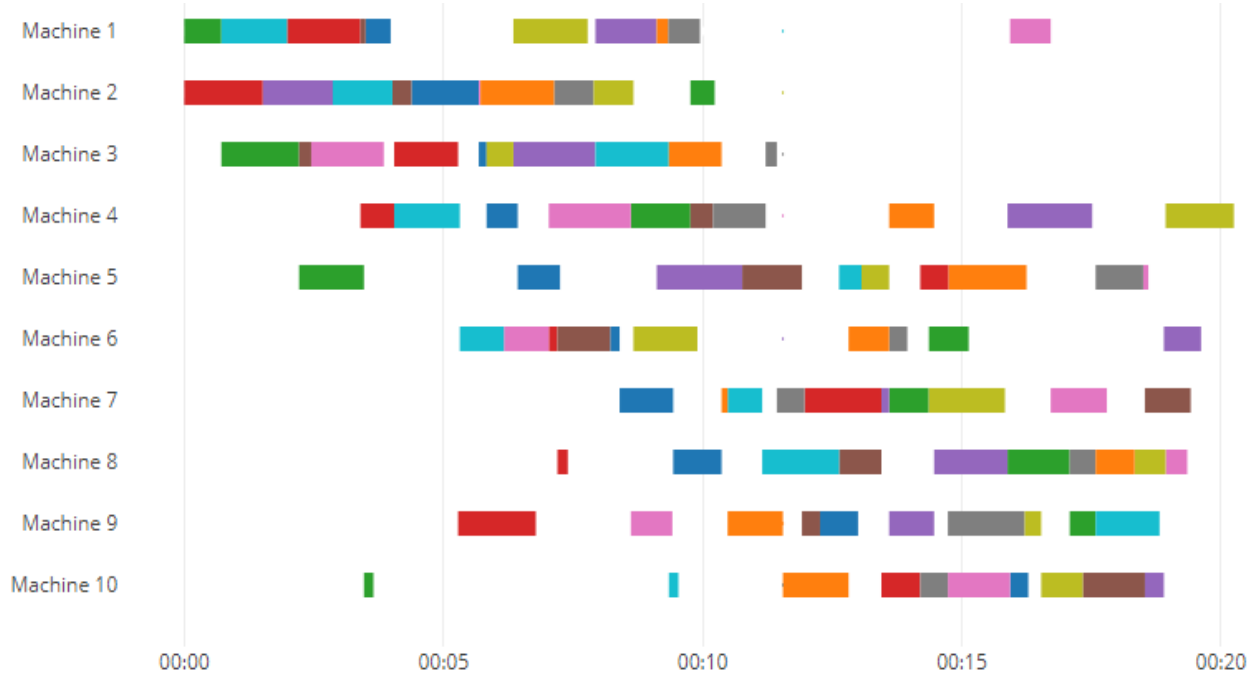

Figura 3: Visualización de la resolución de una instancia del JSP. Cada barra de color representa una task, perteneciendo las task del mismo color a un mismo job. Para cada job, es necesario terminar las task {1, 2, 3, ..., i-1} antes de iniciar la task {i}. En esta resolución, por ejemplo, la task 1 del job verde se ejecuta en la máquina 1, y al finalizar permite que la máquina 3 comience la task 2. Las task 3 y 4 son también tomadas (en secuencia) inmediatamente después por las máquinas 5 y 10, pero la task 5 no es iniciada por la máquina 4 hasta haber completado las task roja, cian, azul y púrpura.

Conocimiento de los datos

Para los primeros dos ejemplos, de tamaños máquinas-por-tareas 5x5 y 10x10, se utilizarán datos sintéticos, que no requieren de un estudio de calidad. Para el caso final se utilizarán datos de un proyecto llevado a cabo por I3b, por lo que el estudio de calidad y el pivotado de los datos ya ha sido realizado con anterioridad, permitiéndonos centrar el proyecto presente en la implementación de soluciones mediante QAOA y separación de circuitos.

En cuanto a los casos, el origen de los datos es el siguiente:

- Caso de 5 trabajos, 5 máquinas: Origen del Dato: https://github.com/dwave-examples/job-shop-scheduling-cqm
- Caso de 10 máquinas, 10 trabajos: https://wurmen.github.io/Genetic-Algorithm-for-Job-Shop-Scheduling-and-NSGA-II/implementation%20with%20python/GA-jobshop/Example1.html
- Caso de uso de cliente de Ibermática: Olanet_Secuenciador. Empresa cliente de Ibermática. Más de 100 máquinas, trabajos realizados durante dos años de procesos reales.

Preparación de los datos

Los datos utilizados en la tercera fase, por ser parte de un proyecto externo, serán anonimizados antes de ser utilizados en este proyecto. Sin embargo, de manera crucial, se seguirá partiendo de datos reales con una clara implementación de mercado.

a) ***Caso de 5 trabajos, 5 máquinas***.

En este caso, se realiza la carga de los datos y el análisis de la calidad de los mismos.

La entrada de los datos tiene el formato siguiente:

```
#Num of jobs: 5
#Num of machines: 5
              task 0          task 1          task 2          task 3          task 4
  job id    machine    dur  machine    dur  machine    dur  machine    dur  machine    dur
--------  ---------  -----  ---------  -----  ---------  -----  ---------  -----  ---------  -----
       0          1      3          3      4          4      4          0      5          2      1
       1          3      4          2      0          1      0          0      1          4      0
       2          1      5          4      5          2      4          0      0          3      3
       3          4      1          3      4          0      2          2      0          1      2
       4          1      3          3      3          0      0          2      0          4      1
```

Figura 4: Input del JSP para 5 máquinas y 5 jobs de hasta 5 tasks cada uno. En el ejemplo, la task 3 del job 0 debe ser ejecutado en la máquina 0, y tardaría 5 unidades de tiempo en completarse. Aquellos tasks "fantasma" de duración 0 indican que su job está compuesto de menos de 5 tasks reales.

Granularidad: Cada una de las tareas a asignar en los distintos trabajos/máquinas. Es decir, la granularidad es la secuencia ***Máquina/Trabajo/Tarea***.

- ***Cadencia***: La secuencia de cada Granularidad (cada trabajo), en las distintas máquinas, en cada ciclo (job id). En definitiva, el ***tiempo de inicio*** (t) en el que cada tarea se realiza, por trabajo y máquina.

- ***Objetivo de Minería***: Función de Evaluación. En este caso, es el tiempo global (la suma de los tiempos) de todas las secuencias con respecto a su Granularidad y su Cadencia.

- ***Objetivo de Modelado***: Modelo Prescriptivo (Optimización), en dónde se busca minimizar la duración global de las secuencias, con ciertas restricciones. Estas restricciones son las siguientes:
  - Las tareas deben ejecutarse secuencialmente.
  - El campo "Dur" se refiere a la duración del procesamiento de una tarea.
  - Cuando la duración es 0 para una tarea, la tarea no se ejecutará.
  - El código impone una relación uno a uno entre tareas y máquinas, por lo tanto, la cantidad de tareas por trabajo siempre es igual a la cantidad de máquinas en el problema.

El problema entra dentro de la categoría: "Job Shop Scheduling Problem"

El flujo es el siguiente.

1) Ante una entrada del tipo

```
#Num of jobs: 5
#Num of machines: 5
```

	task 0		task 1		task 2		task 3		task 4	
job id	machine	dur	machine	dur	machine	dur	machine	dur	machine	dur
0	1	3	3	4	4	4	0	5	2	1
1	3	4	2	0	1	0	0	1	4	0
2	1	5	4	5	2	4	0	0	3	3
3	4	1	3	4	0	2	2	0	1	2
4	1	3	3	3	0	0	2	0	4	1

Figura 5: Entrada de tipos de datos.

Se modeliza los datos de entrada, pivotándolo en base a la Granularidad y Cadencia, según la siguiente forma:

```
#Number of jobs: 5
#Number of machines: 5
#Completion time: 22.0
```

	machine 0			machine 1			machine 2			machine 3			machi
job id	task	start	dur	task	start	dur	task	start	dur	task	start	dur	task	sta
0	3	16	5	0	5	3	4	21	1	1	8	4	2	
1	3	6	1	2	5	0	1	4	0	0	0	4	4	
2	3	17	0	0	0	5	2	13	4	4	17	3	1	
3	2	9	2	4	19	2	3	14	0	1	4	4	0	
4	2	18	0	0	9	3	3	21	0	1	14	3	4	

Figura 6: Output del JSP para el input de la figura 4.

Con lo que queda la siguiente configuración:

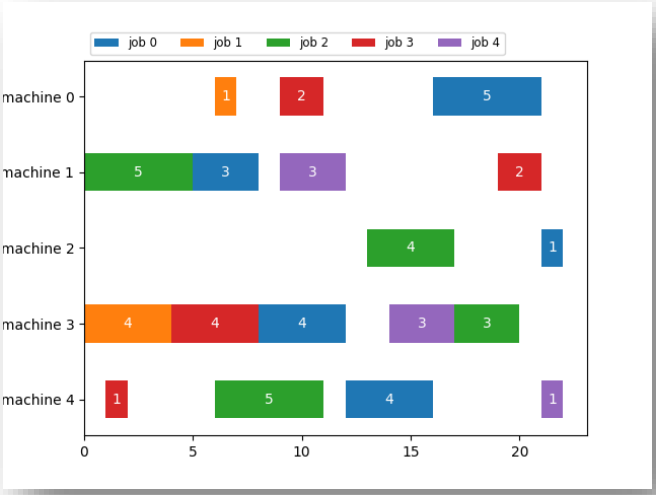

Figura 7: Visualización del output del JSP de la figura 5 obtenido para el input de la figura 4. Los números indican la duración de cada task.

Esta configuración es claramente ineficiente, con lo que se busca una configuración óptima.

Modelado

El problema de optimización elegido, el Job Scheduling Problem (JSP), se ha resuelto mediante QAOA y se han probado con dicho escenario las técnicas de fraccionamiento de circuitos desarrolladas por CESGA. Como paso previo, se han realizado dos aproximaciones "Toy Model" con datos sintéticos.

Se han analizado los modelados matemáticos en distintos casos, mostrando a continuación, el análisis del caso de 5 trabajos, 5 máquinas. Como se ha visto en el apartado anterior, podemos configurar el problema de la asignación de trabajos a máquinas en el tiempo, según la siguiente nomenclatura:

- ***Parámetros de uso:***

n: número de trabajos.
m: número de máquinas.
t: número de tarea.
J: el conjunto de *trabajos* ({0,1,2,...,n}).
M: el conjunto de *máquinas* ({0,1,2,...,m}).
T: el conjunto de *tareas* ({0,1,2,...,m}), que tiene la misma dimensión que M.
M($\boldsymbol{j}$,t): es la máquina que procesa la tarea t del trabajo j.
T($\boldsymbol{j}$,i): es la tarea que procesa la máquina i para el trabajo j.
D($\boldsymbol{j}$,t): es la duración de procesamiento que la tarea t necesita para el trabajo j.
$\boldsymbol{V}$: máxima apertura posible.

- ***Variables:***

$\boldsymbol{w}$: una variable entera positiva que define el tiempo de finalización del JSP.
x($\boldsymbol{j}$, $\boldsymbol{i}$): variables enteras positivas utilizadas para modelar el inicio de cada trabajo j en la máquina i.
y($\boldsymbol{k}$,$\boldsymbol{i}$): variables binarias que definen si el trabajo k precede al trabajo j en la máquina i.

- ***Objetivo***:

Minimiza el indicador $\boldsymbol{w}$ del problema JSP dado.

- ***Restricciones:***

a) ***Restricción de precedencia***: asegura que todas las tareas de un trabajo se ejecuten en el orden dado.

$$x_{j,M_{j,t}} - x_{j,M_{j,t-1}} \geq D_{j,t-1} \qquad \forall j \in J \; \forall t \in T \mid t \geq 1 \qquad (1)$$

Esta restricción garantiza que una tarea para un trabajo determinado, j, en una máquina, $M(j,t)$, comience cuando finalice la tarea anterior. Como ejemplo, para las tareas consecutivas 3 y 4 del trabajo 3 que se ejecutan en la máquina 4 y 1, respectivamente, suponiendo que la tarea 3 tarde 12 horas en finalizar, agregamos esta restricción: $x_{3,4} \geq x_{3,3} + 12$.

```
#Num of jobs: 5
#Num of machines: 5
```

	task 0		task 1		task 2		task 3		task 4	
job id	machine	dur	machine	dur	machine	dur	machine	dur	machine	dur
0	1	3	3	4	4	4	0	5	2	1
1	3	4	2	0	1	0	0	1	4	0
2	1	5	4	5	2	4	0	0	3	3
3	4	1	3	4	0	2	2	0	1	2
4	1	3	3	3	0	0	2	0	4	1

Figura 4 (bis)

b) ***Restricción de no superposición:*** garantiza que varios trabajos no utilicen ninguna máquina al mismo tiempo.

$$x_{j,i} - x_{k,i} + \left(D_{k,T_{k,i}} - D_{j,T_{j,i}}\right) y_{j,k,i} + 2\, y_{j,k,i} \left(x_{k,i} - x_{j,i}\right) \geq D_{k,T_{k,i}} \quad \forall j \in J, \forall k \in J, j < k, \forall i \in M \tag{2}$$

c) ***Restricción del intervalo de creación:*** el intervalo de creación de un problema JSP se puede calcular obteniendo el ***tiempo máximo de finalización*** para la última tarea de todos los trabajos.

$$w - x_{j,M_{j,m-1}} \geq D_{j,m-1} \qquad \forall j \; in \; J \tag{3}$$

- if `y_j,k,i = 0` job `j` is processed after job `k` :

$$x_{j,i} - x_{k,i} \geq D_{k,T_{k,i}} \quad \forall j \in J, \forall k \in J, j < k, \forall i \in M$$

- if `y_j,k,i = 1` job `k` is processed after job `j` :

$$x_{k,i} - x_{j,i} \geq D_{j,T_{j,i}} \quad \forall j \in J, \forall k \in J, j < k, \forall i \in M$$

Figura 8: Forma de la del intervalo de creación en el hamiltoniano QUBO.

2. Algoritmo y pasos:

Pseudo-código:

a) Leer los datos y crear una estructura pivotada en fase a la Granularidad y Cadencia Definida.
   - Esta estructura es un objeto de tipo de datos, incluía información general sobre la instancia, ***w*** (que es la suma de todas las duraciones) y tres diccionarios:
     - task_duration: que es la duración de una determinada tarea de un trabajo.
     - task_machine: que es la máquina en la que se realiza una determinada tarea de un trabajo.
     - maquina_tarea: para cada relación trabajo-tarea, cual es la máquina que realiza esta tarea.

b) Inicializar la clase con los datos origen.

c) Inicializar el modelo de Optimización (QUBO/QAOA/Tensor Networks/Clásica)

d) Definir las variables que componen el problema.
   - El objetivo de minería (w), una sola variable que representa el tiempo del proceso total. Es la función objetivo.
   - Un diccionario con las variables enteras x(***j***,i), cada una de las cuales representa el tiempo de uso de la máquina i para el trabajo j.
   - Un diccionario con las variables binarias y(***j***, ***k***, ***i***), que es 1 si el trabajo j precede al trabajo k en la máquina i.

e) Introducir las restricciones de precedencia.
   - Estas restricciones se introducen una por una mediante un método definido en la clase. Por lo tanto, introduce un bucle doble, por lo que para cada trabajo y tarea introduce una restricción. Para hacer la restricción se usa las variables x(j,i), que representan el tiempo de uso de la máquina i para el trabajo j. Por lo tanto, la diferencia en el tiempo de inicio de la tarea j y la tarea $j-1$ debería ser, al menos, mayor que la duración de la tarea j-1.

f) Introducir las restricciones de precedencia cuadrática.
   - En este caso, tenemos un ciclo triple, para representar todas las relaciones entre ***pares*** de trabajos y máquinas. Por lo tanto, para cada trabajo j, trabajo k y máquina i, el código introduce una restricción usando un método definido en la clase.
   - Además, para cada tarea $i-j$ y la máquina i, el sistema verifica cuál es la tarea de cada trabajo que se realiza en esta máquina. Entonces, si ambas tareas j y k tienen una duración positiva en la máquina i, significa que necesitan usar esa máquina.

g) Evaluar la función de coste (función de evaluación sobre variable w):
   - En este caso, tenemos un solo bucle para cada trabajo. Como criterio, el intervalo de tiempo debe ser mayor que el tiempo de inicio de la última tarea de un trabajo más la duración de esta tarea. Se define el objetivo, que es la variable "w". La variable para optimizar es solo un número entero que debe minimizarse, pero teniendo en cuenta la última restricción, por lo que aseguramos que siempre será el tiempo de finalización de la última tarea del trabajo más largo.

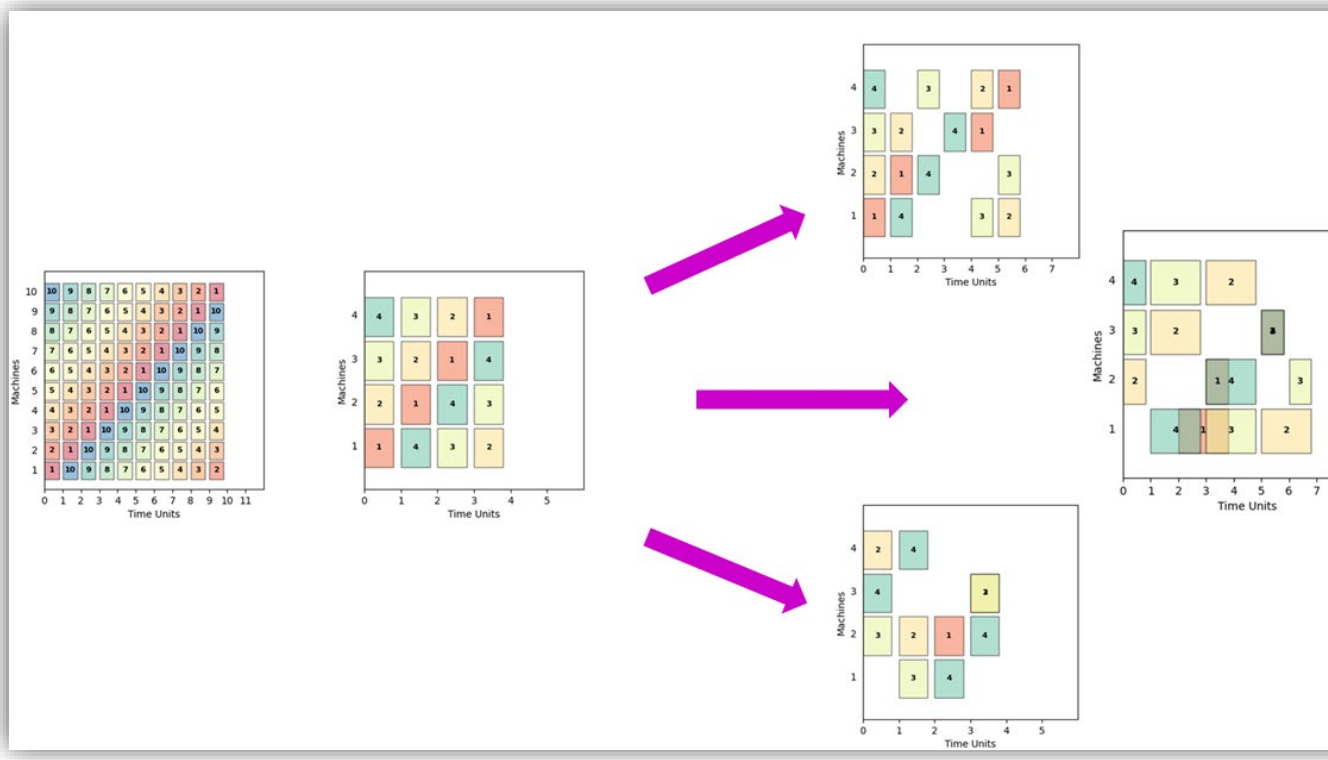

Figura 8: Diagrama indicativo de distintos errores posibles en los resultados de un JSP. En el centro, a la izquierda, un caso de 10 máquinas, pero de resolución trivial (como se puede observar en lo sistemático de la ordenación de las tasks), correctamente resuelto. En el centro, un caso de 4 máquinas y 4 jobs, cada uno de 1 unidad de tiempo, también resuelto correctamente. Las tres flechas señalan a tres resoluciones, cada una con un error posible.

h) La flecha superior señala a una resolución en el que se han respetado las restricciones, pero se ha obtenido un tiempo no-optimo: se puede observar que de no existir las dos unidades de tiempo en las que cada máquina está inactiva, se llegaría a la solución óptima de la imagen central. La flecha central señala a la resolución de un caso en el que cada task tiene una duración de 1 o de 2 unidades de tiempo, pero en la que no se ha respetado la restricción "b" (Restricción de no superposición). Por último, la flecha inferior indica una resolución errónea en la que no tosas las task han sido completadas.

Evaluación

Los resultados obtenidos mediante los procesos y técnicas testados en el proyecto serán comparados para su validación con las soluciones obtenidas mediante otros métodos, ya sean clásicos, cuánticos o híbridos. El método de comparación se elegirá en cada caso según del volumen y la complejidad del sistema a resolver. En el caso final en el que se utilizarán datos reales, la comparación se realizará contra los resultados ya obtenidos por i3B para dicho set de datos.

Despliegue

Una vez concluido el proyecto se han recogido los resultados y las técnicas desarrolladas, tanto para su publicación en revistas científicas como para su futura implementación en próximos proyectos, planteándose también los próximos pasos a seguir en el desarrollo de dichas técnicas.