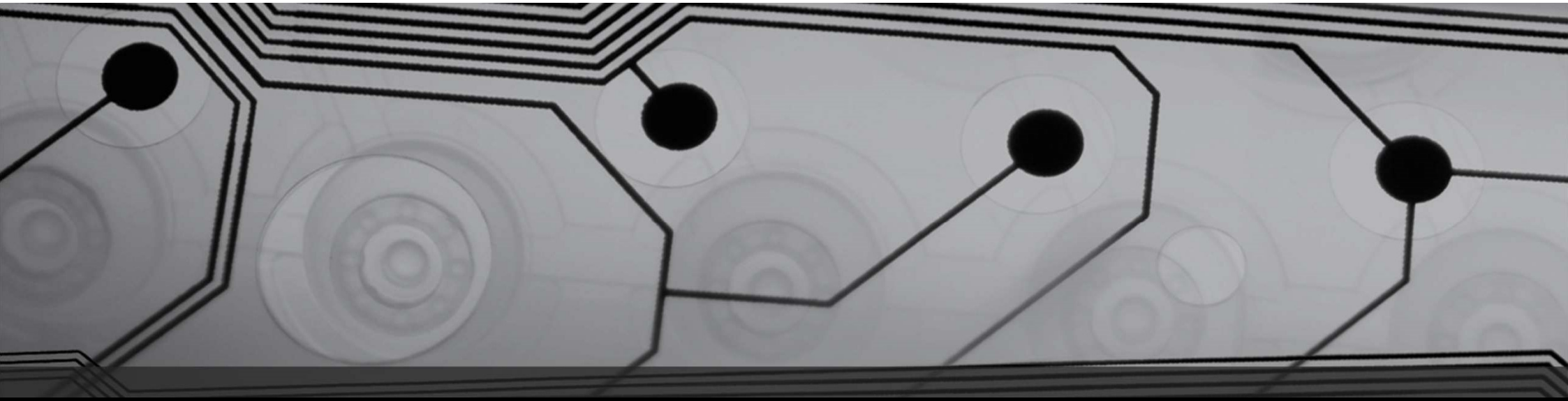
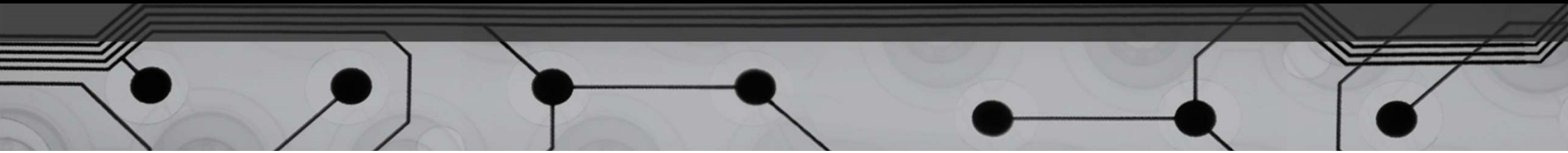


# Job Scheduling in MiniZinc

Knowledge Representation Project – Università della Calabria
Master Degree in Artificial Intelligence and Data Science
a.y. 2021/2022

UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA

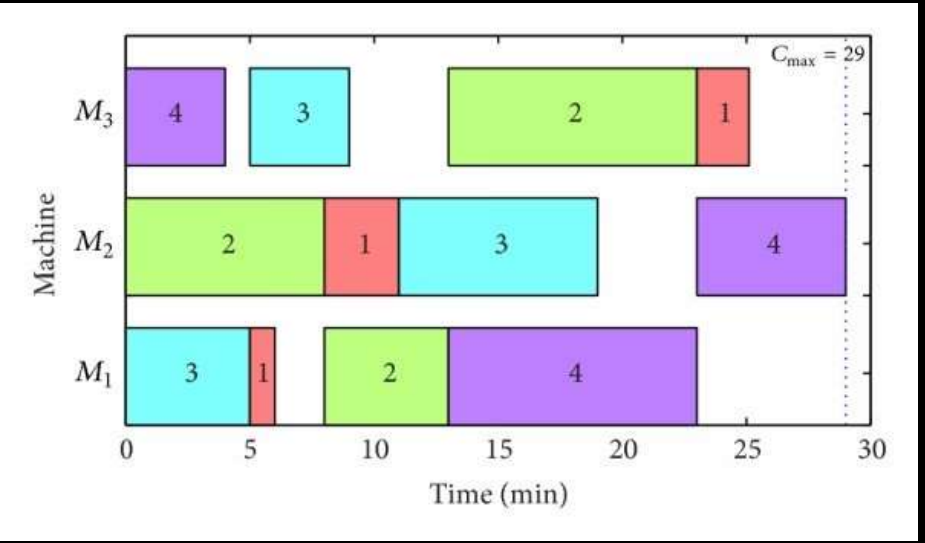


# Job Scheduling in MiniZinc
# Kr Project

- Job Scheduling Problems
- MiniZinc Models
- Java Execution Functions

# Job Scheduling Problems

- **Job Scheduling** (JS) or **Job Problem** (JP) is a popular optimization problem in computer science and operational research.
- This focus on assigning jobs to one or more resources or machines at particular times.

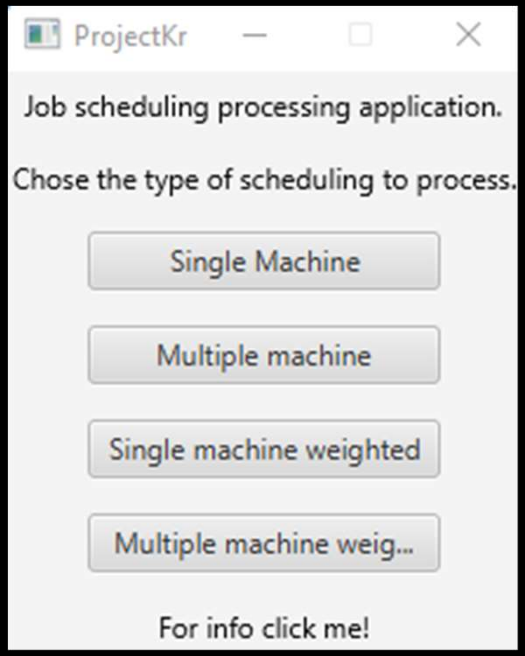



The schedule of 4 job and 3 machines.

# Job Scheduling Problems

- **There are several type of job scheduling problems.**
- Our application is focused on process **single** or **multiple** machine(s) problem with or without assigning **weights** to jobs.
- In addition, the objective function aims to minimize the sum of **completion times**.

- The home page allows users to select the type of job scheduling problem to process.



# MiniZinc Models

Our project requires four MiniZinc models (one per type of job scheduling problem)

- Single Machine Model
- Single Machine with weights Model
- Multiple Machines Model
- Multiple Machines with weights Model

- For example the *jobScheduling_MultipleMachines_weights.mzn* file is structured as shown in the next slide.

# MiniZinc Models

*jobScheduling_MultipleMachines_weights.mzn* file

```
int: njobs;
int: nmachines;
set of int:jobs=1..njobs;
set of int:machines=1..nmachines;
set of int: dom = 0..1;

array[jobs] of int: p;%input tempi di processamento
array[jobs] of int: w;%input dei pesi
array[machines] of int: wMachines;%input dei pesi sulle macchine

constraint assert(njobs > 0, "ValueError: njobs must be non-negative, you have to schedule at least 1 job");
constraint assert(nmachines > 1, "ValueError: machines must be more than one, otherwise select single machine option");
constraint assert(nmachines < njobs, "ValueError: machines must be less than jobs");
constraint assert(forall(i in jobs)(p[i]>0),"ValueError: processing times must be at least 1");
constraint assert(forall(i in jobs)(w[i]>0),"ValueError: weights must be at least one");
constraint assert(forall(i in machines)(wMachines[i]>0),"ValueError: weights on Machines must be at least 1, otherwise select multiple machine option");


array[machines,jobs] of var dom: assign;% tempi di completamento

var int: v;
constraint forall(i in machines)(assign[i,i]=1);
constraint forall(j in jobs)(sum(i in machines)(assign[i,j]) = 1);
constraint forall(i in machines)(v>=sum(j in jobs)(assign[i,j]*p[j]));

%vincolo sulle macchine che possono processare job con pesi <= wMachines[i]
constraint forall(i in machines, j in jobs)(assign[i,j]*w[j] <= wMachines[i]);

solve minimize v;

output[
  show_int(3, assign[i,j]) ++ if j==njobs then "\n" else " " endif | i in machines, j in jobs
];
output["total completion time minimized =\(v)\n"];
```

# Java Execution Functions

- To invoke the **MiniZinc Solver** in Java it is necessary to run a simple command line.

```
> Minizinc.exe --solver Gecode filename.dzn filename.mzn
```

# Java Execution Functions

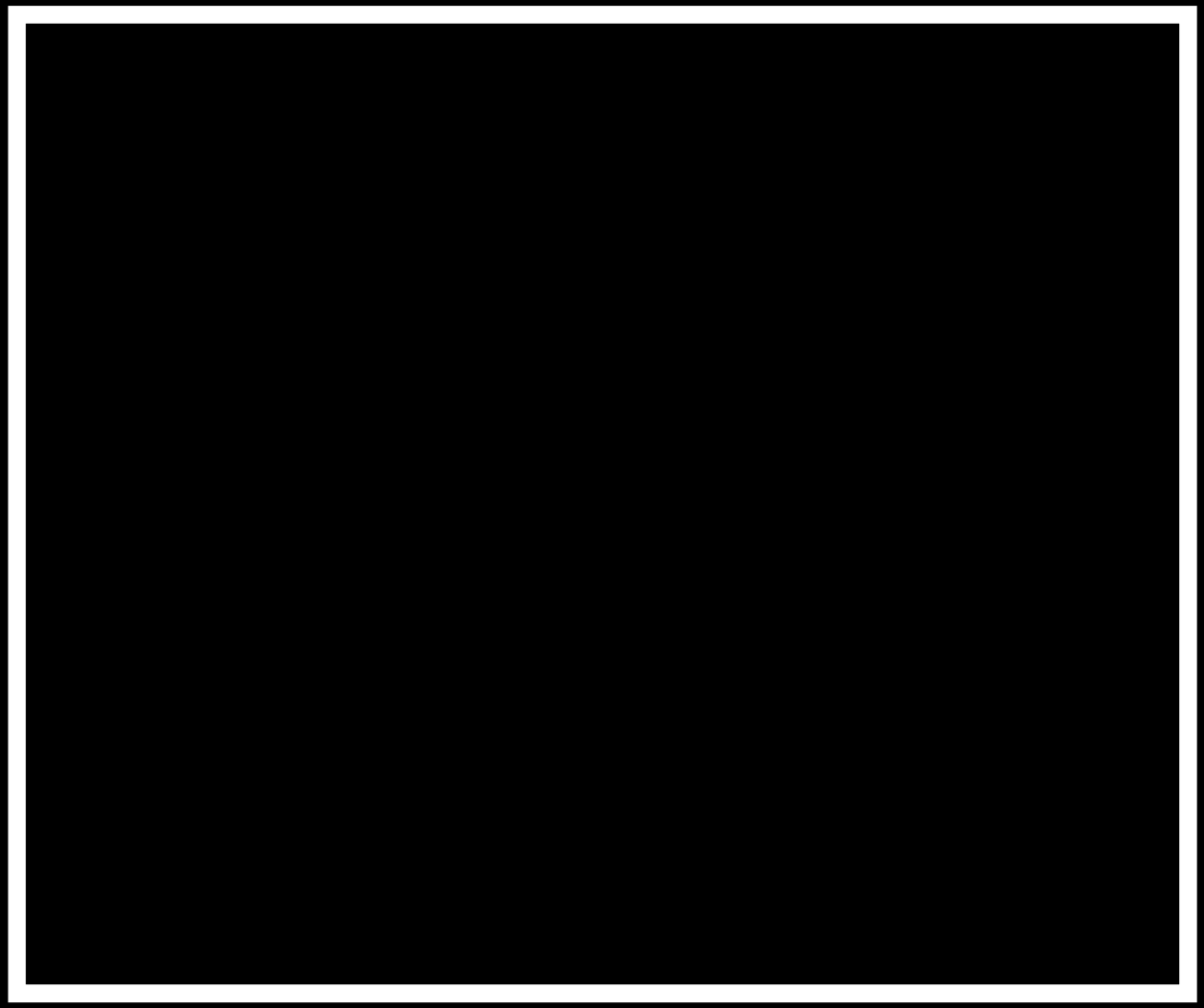
- In our code the function *runSolver(String s)* executes the right model in base of the matching of the type passed as parameter.

```
public  String runSolver(String s){
    String cmd = "C:\\Program Files\\MiniZinc\\minizinc.exe --solver Gecode jobSchedulingInput.dzn
    Runtime run = Runtime.getRuntime();
    Process pr = null;
    String l = "";
    try {
        pr = run.exec(cmd);
        try {
            pr.waitFor();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        BufferedReader buf = new BufferedReader(new InputStreamReader(pr.getInputStream()));
        String line = "";
        while (true) {
                if (!((line=buf.readLine())!=null)) break;
            System.out.println(line);
            l+="\n"+line;
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println(l);
return l;
}
```

# Java Execution Functions

- This function is called when all the text field are filled and the user click on the button “Execute”.
- In the right part of the window it is shown the output of the MiniZinc Solver.



# Demo

To download the app:
GitHub

# THANKS

Project and Presentation by Boca Paolo, Canonaco Martina, Fazio Francesco, Moscato Giuseppe and Passarelli Chiara.