The project you will be working on is the ROADEF/EURO challenge of 2012. It is a competition jointly organized by the French Operational Research and Decision Aid society (ROADEF) and the European Operational Research society (EURO). This competition aims at developing a solution method for an industrial optimization problem proposed by an industrial partner. In 2012, the problem was proposed by Google. This problem was a large-scale machine reassignment problem.

Download the archive available at the URL sent by email and extract the files and folders into a directory called *Numerics-project*.

1 Problem description

The objective of this challenge is to improve the utilization of a set of machines. The problem is formally defined in the document titled *Problem-definition.pdf*. Here is an overview. A machine has several resources, for example RAM or CPU, and runs processes that consume these resources. Initially, each process is assigned to a machine. In order to improve machine utilization, processes can be moved to another machine. Hard constraints (resource capacity, conflict, spread, dependency, transient usage) limit the possibilities of moving processes. A solution to the problem is an assignment of processes to machines satisfying all hard constraints. The cost (i.e., the objective function value) of a solution is the sum of load, balance and move related costs. The problem consists in finding a feasible solution with minimum cost.

When describing the constraints and the objective function in the document *Problem-definition.pdf*, it assumes that a solution is given (i.e., an assignment of the processes to the machines). The notation $M(p)$ refers to the machine to which p is assigned ($M(p) \in \mathcal{M}$). Below, you have the notations used for the data of the problem.

- $\mathcal{P}$: set of processes
- $\mathcal{M}$: set of machines
- $M_0(p)$: machine on which process $p \in \mathcal{P}$ is initially assigned to $[M_0(p) \in \mathcal{M}]$
- $\mathcal{R}$: set of resources (e.g., CPU,RAM,DISK)
- $R(p, r)$: usage of resource $r \in \mathcal{R}$ by process $p \in \mathcal{P}$ $[R(p, r) \geqslant 0]$
- $C(m, r)$: capacity of resource $r \in \mathcal{R}$ for machine $m \in \mathcal{M}$ $[C(m, r) \geqslant 0]$
- $SC(m, r)$: safety capacity of resource $r \in \mathcal{R}$ for machine $m \in \mathcal{M}$ $[SC(m, r) \geqslant 0]$
- $\mathcal{S} \subseteq 2^{\mathcal{P}}$: set of services (a service is a subset of processes)
- $Spreadmin(s)$: minimum spread of service s (the minimum number of distinct locations where at least one process of service s should run)
- $Dep(s)$: set of services on which s depends $[Dep(s) \subset \mathcal{S}]$
- $\mathcal{L} \subseteq 2^{\mathcal{M}}$: set of locations (a location is a subset of machines)
- $\mathcal{N} \subseteq 2^{\mathcal{M}}$: set of neighborhoods (a neighborhood is a subset of machines)
- $\mathcal{B} \subset \mathcal{R} \times \mathcal{R} \times \mathbb{N}$: set of triples (or 3-tuple) $[b = (r_1, r_2, target) \in \mathcal{B}]$
- $weight_{loadCost}(r)$: coefficient for the load cost associated with resource $r \in \mathcal{R}$ $[weight_{loadCost}(r) \geqslant 0]$

- $weight_{balanceCost}(b)$: coefficient for the balance cost associated with triple $b \in \mathcal{B}$ $[weight_{balanceCost}(b) \geqslant 0]$
- $weight_{processMoveCost}$ $[weight_{processMoveCost} \geqslant 0]$
- $PMC(p)$: cost of moving process p on a machine different than $M_0(p)$ $[PMC(p) \geqslant 0]$
- $weight_{serviceMoveCost}$: coefficient for the service move cost $[weight_{serviceMoveCost} \geqslant 0]$
- $weight_{machineMoveCost}$: coefficient for the machine move cost $[weight_{serviceMoveCost} \geqslant 0]$
- $MMC(m, m')$: cost of moving a process from machine $m \in \mathcal{M}$ to machine $m' \in \mathcal{M}$ $[MMC(m, m') \geqslant 0]$

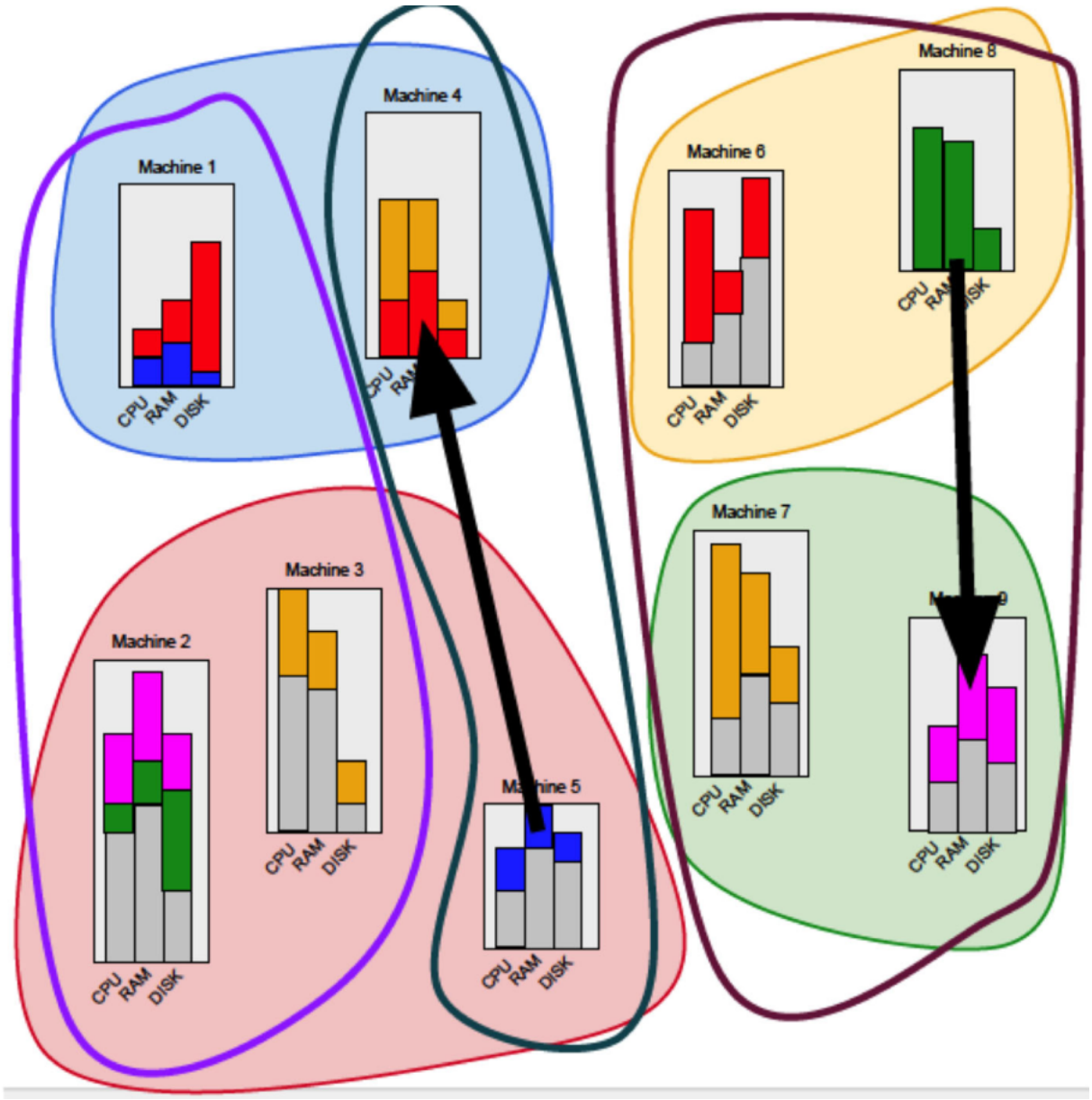

Figure 1: An example of a problem instance with 18 processes and 9 machines

Processes are assigned to machines and occupy a certain amount of resources (CPU, RAM and disk). Each process is a colored rectangle and the color identifies the process service. The height of each process on each machine represents the proportion of the capacity of each resource that it requires. Machines are partitioned into four locations (circles with a colored background) and neighborhoods (circles with a transparent background). There are also two dependencies represented by the arrows (the service colored in blue depends on the service colored in red / the service colored in green depends on the service colored in purple)

2 Work to do

For this project, you must develop a solution method for the machine reassignment problem. It can be an algorithmic solution or can make use of mathematical programming solvers. You must implement this method using the Python programming language. A code base is provided. It reads the data files (see `parser.py`) and fills in data structures. There are two main data structures (classes) called `Data` that stores all the data of the problem and `Solution` that stores a solution to the problem. Both these classes are in file `problem.py`. It also contains a solution checker (see `checker.py`). The file `run.py` contains the main function.

You are invited to complete the code base with the work requested. One should access your solution method through the following method you should implement:

```
# Build a solution to the machine assignment problem for the given Data object
# data :  object of class Data containing the problem data (see problem.py)
# maxTime :  maximum time limit (in seconds) -> you should try to take it into account
into your method
# verbose :  if it is set to False, there should not be anything print into the console
# Returns an object of class Solution (see problem.py)

def solve(data:  problem.Data, maxTime:  int, verbose:  bool) -> problem.Solution:
```

You are free to implement your code in the Jupyter notebook file *Project.ipynb* or to do your implementation in a Python file (that can be dependent of other Python files if needed).

The package `numpy` should be installed in the Python environment you use. If you want to solve a mixed integer linear programming formulation, you should work with an environment where the package `mip` is installed.

3 Evaluation

You will work by teams of 3 or 4. You will present your solution method and your results during a first presentation on Friday afternoon. After the end of the week, you can continue working on the project until December, when there will be a final defense. You will submit your code, your results and a short document explaining your solution method before the defense (the deadline is not determined yet).

According to the submitted results, teams will be ranked according to the following scheme. The solution where no process is reassigned (i.e, $M(p) = M_0(p)$) is feasible. We call this solution *the reference solution*. The score of a team $\mathtt{t} \in \mathtt{T}$ for a given instance $\mathtt{i} \in \mathtt{I}$ is given by the difference between the cost of the team solution $c_{\mathtt{t},\mathtt{i}}$ and the best cost among the teams divided by the cost $c^0_{\mathtt{i}}$ of the reference solution.

$$\text{score}_{\mathtt{t},\mathtt{i}} = \frac{c_{\mathtt{t},\mathtt{i}} - \min_{\mathtt{t}' \in \mathtt{T}} c_{\mathtt{t}',\mathtt{i}}}{c^0_{\mathtt{i}}} \qquad \mathtt{i} \in \mathtt{I}, \mathtt{t} \in \mathtt{T}$$

The score of a team is the sum of its scores on all instances.

$$\text{score}_{\mathtt{t}} = \sum_{\mathtt{i} \in \mathtt{I}} \text{score}_{\mathtt{t},\mathtt{i}} \qquad \mathtt{t} \in \mathtt{T}$$

If the solution of a team `t` for an instance `i` is infeasible or if an errors has occurred within the solution method, then we set $c_{\mathtt{t},\mathtt{i}} = 2 \times c^0_{\mathtt{i}}$. The set `I` is composed of the instances of set A for the first review of the project on Friday.

Apart from the team ranking, you will be evaluated on the clarity and correctness of the presentation of you work in the short document and during the defense.