1 Problem description

The aim of this challenge is to improve the usage of a set of machines. A machine has several resources, such as RAM and CPU, and runs processes consuming these resources. Initially each process is assigned to a machine. In order to improve the machine usage, processes can be moved from one machine to another. Possible moves are subject to a set of hard constraints, such as resource capacity, and these moves are associated with various costs. A solution to this problem is a new process-machine assignment which satisfies all hard constraints and minimizes a given overall cost.

1.1 Decision variables

Let $\mathcal{M}$ be the set of machines, and $\mathcal{P}$ the set of processes. A solution is an assignment of each process $p \in \mathcal{P}$ to one and only one machine $m \in \mathcal{M}$; this assignment is noted by the mapping $M(p) = m$ in this document. The original assignment of process $p$ is denoted $M_0(p)$. Note the original assignment is feasible, i.e. all hard constraints are satisfied.

For instance, if $\mathcal{M} = \{m_1, m_2\}$ and $\mathcal{P} = \{p_1, p_2, p_3\}$, then $M(p_1) = m_1$, $M(p_2) = m_1$, $M(p_3) = m_2$ means processes $p_1$ and $p_2$ run on machine $m_1$ and process $p_3$ runs on machine $m_2$.

1.2 Hard constraints

1.2.1 Capacity constraints

Let $\mathcal{R}$ be the set of resources which is common to all the machines, $C(m, r)$ the capacity of resource $r \in \mathcal{R}$ for machine $m \in \mathcal{M}$ and $R(p, r)$ the requirement of resource $r \in \mathcal{R}$ for process $p \in \mathcal{P}$. Then, given an assignment $M$, the usage $U$ of a machine $m$ for a resource $r$ is defined as:

$$U(m, r) = \sum_{\substack{p \in \mathcal{P} \text{ such that} \\ M(p) = m}} R(p, r)$$
A process can run on a machine if and only if the machine has enough available capacity on every resource. More formally, a feasible assignment must satisfy the capacity constraints:

$$\forall m \in \mathcal{M}, r \in \mathcal{R}, \quad U(m, r) \leq C(m, r)$$

Consider for example machines $\mathcal{M} = \{m_1, m_2\}$, processes $\mathcal{P} = \{p_1, p_2, p_3\}$ and resources $\mathcal{R} = \{CPU, RAM\}$. Available CPU is $C(m_1, CPU) = 16$, $C(m_2, CPU) = 8$ and available RAM is $C(m_1, RAM) = 16$, $C(m_2, RAM) = 4$. CPU requirements are $R(p_1, CPU) = 6$, $R(p_2, CPU) = 1$ and $R(p_3, CPU) = 3$. RAM requirements are $R(p_1, RAM) = 13$, $R(p_2, RAM) = 3$ and $R(p_3, RAM) = 1$.

Assignments $M(p_1) = m_1$, $M(p_2) = m_1$, $M(p_3) = m_2$ and $M(p_1) = m_1$, $M(p_2) = m_2$, $M(p_3) = m_2$ satisfy capacity constraints.

However assignment $M(p_1) = m_2$, $M(p_2) = m_1$, $M(p_3) = m_2$ is not feasible as $m_2$ has not enough available CPU. In the same way, assignment $M(p_1) = m_1$, $M(p_2) = m_1$, $M(p_3) = m_1$ is not feasible as $m_1$ has not enough available RAM.

### 1.2.2 Conflict constraints

Processes are partitioned into services. Let $\mathcal{S}$ be a set of services. A service $s \in \mathcal{S}$ is a set of processes which must run on distinct machines. Note that all services are disjoint.

$$\forall s \in \mathcal{S}, (p_i, p_j) \in s^2, p_i \neq p_j \Rightarrow M(p_i) \neq M(p_j)$$

For instance $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$, $\mathcal{P} = \{p_1, p_2, p_3\}$, $\mathcal{S} = \{s^a, s^b\}$ with $s^a = \{p_1\}$ and $s^b = \{p_2, p_3\}$, and assignments $M_0(p_1) = m_1$, $M_0(p_2) = m_1$ and $M_0(p_3) = m_3$. Process $p_1$ can be reassigned to any machine, e.g. $M(p_1) = m_2$, $M(p_2) = m_1$ and $M(p_3) = m_3$. However $p_2$ cannot be reassigned to machine $m_3$, i.e. $M(p_1) = m_1$, $M(p_2) = m_3$ and $M(p_3) = m_3$, because process $p_3$ is a process of service $s^b$ too and is currently running on $m_3$.

### 1.2.3 Spread constraints

Let $\mathcal{L}$ be the set of locations, a location $l \in \mathcal{L}$ being a set of machines. Note that locations are disjoint sets. For each $s \in \mathcal{S}$ let $\text{spreadMin}(s) \in \mathbb{N}$ be the minimum number of distinct locations where at least one process of service $s$ should run. The constraints are defined by:

$$\forall s \in \mathcal{S}, \sum_{l \in \mathcal{L}} \min \left(1, \left| \{p \in s \mid M(p) \in l\} \right| \right) \geq \text{spreadMin}(s)$$

For instance $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$, $\mathcal{P} = \{p_1, p_2\}$, $\mathcal{S} = \{s\}$ with $s = \{p_1, p_2\}$, $\mathcal{L} = \{\{m_1, m_2\}, \{m_3, m_4\}\}$, $M_0(p_1) = m_1$, $M_0(p_2) = m_3$ and $\text{spreadMin}(s) = 2$. Process $p_1$ can be reassigned to $m_2$ or $m_4$. Process $p_2$ can be reassigned to $m_4$. But to satisfy the spread constraint, process $p_1$ cannot be
reassigned to \( m_3 \), and \( p_2 \) cannot be reassigned to \( m_1 \) or \( m_2 \) as in these cases only one location runs \( s \) processes.

### 1.2.4 Dependency constraints

Let \( \mathcal{N} \) be the set of neighborhoods, a neighborhood \( n \in \mathcal{N} \) being a set of machines. Note that neighborhoods are disjoint sets.

If service \( s^a \) depends on service \( s^b \), then each process of \( s^a \) should run in the neighborhood of a \( s^b \) process:

\[
\forall p^a \in s^a, \exists p^b \in s^b \text{ and } n \in \mathcal{N} \text{ such that } M(p^a) \in n \text{ and } M(p^b) \in n
\]

Note dependency constraints are not symmetric, i.e. service \( s^a \) depends on service \( s^b \) is not equivalent to service \( s^b \) depends on service \( s^a \).

Consider for instance \( \mathcal{M} = \{m_1, m_2, m_3, m_4\} \), \( \mathcal{P} = \{p_1, p_2, p_3\} \), \( \mathcal{S} = \{s^a, s^b\} \) with \( s^a = \{p_1\} \) and \( s^b = \{p_2, p_3\} \), initial assignments \( M_0(p_1) = m_1, M_0(p_2) = m_1 \) and \( M_0(p_3) = m_3 \), and neighborhood \( \mathcal{N} = \{\{m_1\}, \{m_2\}, \{m_3, m_4\}\} \). If \( s^a \) depends on \( s^b \), \( p_1 \) can be reassigned to \( m_3 \) or \( m_4 \) as \( p_3 \) is a process of service \( s^b \) and runs in the \( \{m_3, m_4\} \) neighborhood. However \( p_1 \) cannot be reassigned to \( m_2 \) as there is no \( s^b \) process running in the neighborhood of \( m_2 \). In the same way, process \( p_2 \) cannot be reassigned to any other machine as \( p_1 \) needs a \( s^b \) process in its neighborhood.

### 1.2.5 Transient usage constraints

When a process \( p \) is moved from one machine \( m \) to another machine \( m' \) some resources are consumed twice; for example disk space is not available on machine \( m \) during a copy from machine \( m \) to \( m' \), and \( m' \) should obviously have enough available disk space for the copy. Let \( \mathcal{TR} \subseteq \mathcal{R} \) be the subset of resources which need transient usage, i.e. require capacity on both original assignment \( M_0(p) \) and current assignment \( M(p) \). Then the transient usage constraints are:

\[
\forall m \in \mathcal{M}, r \in \mathcal{TR}, \sum_{p \in \mathcal{P} \text{ such that } M_0(p) = m \vee M(p) = m} R(p, r) \leq C(m, r)
\]

Note there is no time dimension in this problem, i.e. all moves are assumed to be done at the exact same time. Then for resources in \( \mathcal{TR} \) this constraint subsumes the capacity constraint.

For instance \( \mathcal{M} = \{m_1, m_2, m_3\} \) and \( \mathcal{P} = \{p_1, p_2\} \), \( M_0(p_1) = m_1 \) and \( M_0(p_2) = m_2 \). \( \mathcal{R} = \{CPU, DISK\} \) and \( \mathcal{TR} = \{DISK\} \). \( C(m_1, CPU) = 3, C(m_2, CPU) = 3, C(m_3, CPU) = 3, C(m_1, DISK) = 10, C(m_2, DISK) = 10, C(m_3, DISK) = 7, R(p_1, CPU) = 1, R(p_2, CPU) = 1 \) and \( R(p_1, DISK) = 8, R(p_2, DISK) = 6 \).

Let’s suppose process \( p_2 \) was moved from \( m_2 \) to \( m_3 \), so \( M(p_2) = m_3 \) and \( M_0(p_2) = m_2 \). Process \( p_1 \) cannot be moved from \( m_1 \) to \( m_2 \) even if no process is currently running on machine \( m_2 \). This is due to the transient usage constraint which in some way still consumes 6 \( DISK \) on machine \( m_2 \).
1.3 Objectives

The aim is to improve the usage of a set of machines. To do so a total objective cost is built by combining a load cost, a balance cost and several move costs.

1.3.1 Load cost

Let \( SC(m, r) \) be the safety capacity of a resource \( r \in R \) on a machine \( m \in M \). The load cost is defined per resource and corresponds to the used capacity above the safety capacity; more formally:

\[
\text{loadCost}(r) = \sum_{m \in M} \max(0, U(m, r) - SC(m, r))
\]

For instance \( M = \{m_1, m_2\} \), \( P = \{p_1, p_2\} \) and \( R = \{r\} \). \( M_0(p_1) = m_1 \), \( M_0(p_2) = m_1 \), \( C(m_1, r) = 100 \), \( C(m_2, r) = 100 \), \( SC(m_1, r) = 10 \), \( SC(m_2, r) = 50 \), \( R(p_1, r) = 7 \) and \( R(p_2, r) = 12 \). Then \( \text{loadCost}(r) = \max(0, 12 + 7 - 10) = 9 \).

Moving process \( p_2 \) from machine \( m_1 \) to machine \( m_2 \), reduces the load cost from 9 to 0, i.e. \( \text{loadCost}(r) = \max(0, 7 - 10) + \max(0, 12 - 50) = 0 \).

1.3.2 Balance cost

As having available CPU resource without having available RAM resource is useless for future assignments, one objective of this problem is to balance available resources. The idea is to achieve a given target on the available ratio of two different resources. Let \( B \) be a set of triples defined in \( \mathbb{N} \times R^2 \). For a given triple \( b = \langle r_1, r_2, \text{target} \rangle \in B \), the balance cost is:

\[
\text{balanceCost}(b) = \sum_{m \in M} \max(0, \text{target} \cdot A(m, r_1) - A(m, r_2))
\]

with \( A(m, r) = C(m, r) - U(m, r) \)

For instance \( M = \{m_1, m_2\} \), \( P = \{p_1, p_2, p_3\} \) and \( M_0(p_1) = m_1 \), \( M_0(p_2) = m_1 \), \( M_0(p_3) = m_2 \). \( R = \{\text{CPU, RAM}\} \), available CPU is \( C(m_1, \text{CPU}) = 16 \), \( C(m_2, \text{CPU}) = 8 \) and available RAM is \( C(m_1, \text{RAM}) = 16 \), \( C(m_2, \text{RAM}) = 12 \). CPU requirements are \( R(p_1, \text{CPU}) = 2 \), \( R(p_2, \text{CPU}) = 8 \) and \( R(p_3, \text{CPU}) = 5 \). RAM requirements are \( R(p_1, \text{RAM}) = 8 \), \( R(p_2, \text{RAM}) = 1 \) and \( R(p_3, \text{RAM}) = 1 \). Machines should be balanced such that for one unit of available CPU, 2 units of RAM are available, i.e. \( B = \{(\text{CPU, RAM, 2})\} \).

Then the balance cost is:

\[
\text{balanceCost}(\langle \text{CPU, RAM, 2} \rangle) = \max(0, 2 \cdot 6 - 7) + \max(0, 2 \cdot 3 - 11)
\]
\[
= 5 + 0
\]
\[
= 5.
\]

If process \( p_1 \) is moved from \( m_1 \) to \( m_2 \), then the cost is:
balanceCost((CPU, RAM, 2)) = \max(0, 2 \cdot 8 - 15) + \max(0, 2 \cdot 1 - 3) = 1 + 0 = 1.

1.3.3 Process move cost

Some processes are painful to move; to model this soft constraint a process move cost is defined. Let \( PMC(p) \) be the cost of moving the process \( p \) from its original machine \( M_0(p) \).

\[
\text{processMoveCost} = \sum_{\substack{p \in P \text{ such that } M(p) \neq M_0(p)}} PMC(p)
\]

For instance \( M = \{m_1, m_2\} \) and \( P = \{p_1, p_2\}, M_0(p_1) = m_1, M_0(p_2) = m_1, PMC(p_1) = 1, PMC(p_2) = 10 \) and process \( p_2 \) is moved from machine \( m_1 \) to machine \( m_2 \), \( M(p_2) = m_2 \). Then \( \text{processMoveCost} = 10 \).

1.3.4 Service move cost

To balance moves among services, a service move cost is defined as the maximum number of moved processes over services. More formally:

\[
\text{serviceMoveCost} = \max_{s \in S} \left( |\{p \in s \mid M(p) \neq M_0(p)\}| \right)
\]

Consider for instance \( M = \{m_1, m_2, m_3\}, P = \{p_1, p_2, p_3, p_4\}, S = \{\{p_1, p_2\}, \{p_3, p_4\}\}, M_0(p_1) = m_1, M_0(p_2) = m_2, M_0(p_3) = m_1, M_0(p_4) = m_2, M(p_1) = m_2, M(p_2) = m_1, M(p_3) = m_1 \) and \( M(p_4) = m_3 \). Then \( \text{serviceMoveCost} = \max(2, 1) = 2 \).

1.3.5 Machine move cost

Let \( MMC(m_{\text{source}}, m_{\text{destination}}) \) be the cost of moving any process \( p \) from machine \( m_{\text{source}} \) to machine \( m_{\text{destination}} \). Obviously for any machine \( m \in M \), \( MMC(m, m) = 0 \). The machine move cost is then the sum of all moves weighted by relevant \( MMC \):

\[
\text{machineMoveCost} = \sum_{p \in P} MMC(M_0(p), M(p))
\]

Consider for instance \( M = \{m_1, m_2\}, P = \{p_1, p_2, p_3, p_4\}, M_0(p_1) = m_1, M_0(p_2) = m_1, M_0(p_3) = m_2, M_0(p_4) = m_2, M(p_1) = m_2, M(p_2) = m_1, M(p_3) = m_1 \) and \( M(p_4) = m_1 \). MMC matrix is: \( MMC(m_1, m_1) = 0, MMC(m_1, m_2) = 10, MMC(m_2, m_1) = 7 \) and \( MMC(m_2, m_2) = 0 \). Then \( \text{machineMoveCost} = 10 + 0 + 7 + 7 = 24 \).
1.3.6 Total objective cost

The total objective cost is a weighted sum of all previous costs. It is the cost to minimize.

\[
\text{totalCost} = \sum_{r \in R} \text{weight}_{\text{loadCost}}(r) \cdot \text{loadCost}(r) \\
+ \sum_{b \in B} \text{weight}_{\text{balanceCost}}(b) \cdot \text{balanceCost}(b) \\
+ \text{weight}_{\text{processMoveCost}} \cdot \text{processMoveCost} \\
+ \text{weight}_{\text{serviceMoveCost}} \cdot \text{serviceMoveCost} \\
+ \text{weight}_{\text{machineMoveCost}} \cdot \text{machineMoveCost}
\]

2 I/O file formats

To ease the file format description, consider the following example with four machines, three processes, two resources and two services.

\[
\begin{align*}
\mathcal{M} &= \{m_1, m_2, m_3, m_4\} \\
\mathcal{P} &= \{p_1, p_2, p_3\} \\
\mathcal{R} &= \{r_1, r_2\} \\
\mathcal{S} &= \{s^a, s^b\} \text{ with } s^a = \{p_1, p_2\}, s^b = \{p_3\} \\
\mathcal{N} &= \{n_1, n_2\} \text{ with } n_1 = \{m_1, m_2\}, n_2 = \{m_3, m_4\} \\
\mathcal{L} &= \{l_1, l_2, l_3\} \text{ with } l_1 = \{m_1, m_2\}, l_2 = \{m_3\}, l_3 = \{m_4\} \\
\mathcal{TR} &= \{r_1\} \\
\mathcal{B} &= \{(r_1, r_2, 20)\}
\end{align*}
\]

\[
\begin{align*}
C(m_1, r_1) &= 30, C(m_1, r_2) = 400, SC(m_1, r_1) = 16, SC(m_1, r_2) = 80 \\
C(m_2, r_1) &= 10, C(m_2, r_2) = 240, SC(m_2, r_1) = 8, SC(m_2, r_2) = 160 \\
C(m_3, r_1) &= 15, C(m_3, r_2) = 100, SC(m_3, r_1) = 12, SC(m_3, r_2) = 80 \\
C(m_4, r_1) &= 10, C(m_4, r_2) = 100, SC(m_4, r_1) = 8, SC(m_4, r_2) = 80 \\
R(p_1, r_1) &= 12, R(p_1, r_2) = 10 \\
R(p_2, r_1) &= 10, R(p_2, r_2) = 20 \\
R(p_3, r_1) &= 6, R(p_3, r_2) = 200
\end{align*}
\]

\[s^b \text{ depends on } s^a\]

\[
\begin{align*}
\text{spreadMin}(s^a) &= 2, \text{spreadMin}(s^b) = 1 \\
PMC(p_1) &= 1000, PMC(p_2) = 100, PMC(p_3) = 1
\end{align*}
\]

<table>
<thead>
<tr>
<th>MMC(m_i, m_j)</th>
<th>m_1</th>
<th>m_2</th>
<th>m_3</th>
<th>m_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>m_2</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>m_3</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>m_4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
weight\textsubscript{loadCost}(r_1) = 100, weight\textsubscript{loadCost}(r_2) = 10
weight\textsubscript{balanceCost}((r_1, r_2, 20)) = 10
weight\textsubscript{processMoveCost} = 1
weight\textsubscript{serviceMoveCost} = 10
weight\textsubscript{machineMoveCost} = 100

And the original solution is: \( M_0(p_1) = m_1, M_0(p_2) = m_4, M_0(p_3) = m_1 \)
A new solution could be: \( M(p_1) = m_1, M(p_2) = m_3, M(p_3) = m_2 \)

2.1 Instance input file format

In order to keep the file format as simple as possible, the instance input file is a list of integers and booleans. Values are space separated and should respect the following order:

<table>
<thead>
<tr>
<th>Number of resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each resource ( r_i ):</td>
</tr>
<tr>
<td>Boolean(is in TR)</td>
</tr>
<tr>
<td>weight\textsubscript{loadCost}(r_i)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each machine ( m_i ):</td>
</tr>
<tr>
<td>Neighborhood ( m_i ) belongs to</td>
</tr>
<tr>
<td>Location ( m_i ) belongs to</td>
</tr>
<tr>
<td>Capacities, ( C(m_i, r_1) C(m_i, r_2) C(m_i, r_3) \ldots )</td>
</tr>
<tr>
<td>Safety capacities, ( SC(m_i, r_1) SC(m_i, r_2) SC(m_i, r_3) \ldots )</td>
</tr>
<tr>
<td>MMC(( m_i, m_1 )) MMC(( m_i, m_2 )) MMC(( m_i, m_3 )) \ldots</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of services</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each service ( s^a ):</td>
</tr>
<tr>
<td>spread\textsubscript{Min}(s^a)</td>
</tr>
<tr>
<td>Number of services ( s^a ) depends on and the list of those services</td>
</tr>
<tr>
<td>e.g. 3 ( s^a ) ( s^d ) ( s^e )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each process ( p_i ):</td>
</tr>
<tr>
<td>Service ( p_i ) belongs to</td>
</tr>
<tr>
<td>Requirements, ( R(p_i, r_1) R(p_i, r_2) R(p_i, r_3) \ldots )</td>
</tr>
<tr>
<td>PMC(( p_i ))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of balance objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each balance objective ( b_i ):</td>
</tr>
<tr>
<td>balance triple, ( r_j r_k target )</td>
</tr>
<tr>
<td>weight\textsubscript{balanceCost}(b_i)</td>
</tr>
</tbody>
</table>

| weight\textsubscript{processMoveCost} |
| weight\textsubscript{serviceMoveCost} |
| weight\textsubscript{machineMoveCost} |
Next table illustrates the instance input format using previous instance as an example:

<table>
<thead>
<tr>
<th>2</th>
<th>Number of resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Resource #0 is transient</td>
</tr>
<tr>
<td>100</td>
<td>weight of resource #0</td>
</tr>
<tr>
<td>0</td>
<td>Resource #1 is not transient</td>
</tr>
<tr>
<td>10</td>
<td>weight of resource #1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>Number of machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Machine #0 is in neighborhood #0</td>
</tr>
<tr>
<td>0</td>
<td>Machine #0 is in location #0</td>
</tr>
<tr>
<td>30 400</td>
<td>Capacities of machine #0</td>
</tr>
<tr>
<td>16 80</td>
<td>Safety capacities of machine #0</td>
</tr>
<tr>
<td>0 1 4 5</td>
<td>Moving cost from machine #0 to machines #0, #1, #2 and #3</td>
</tr>
<tr>
<td>0</td>
<td>Machine #1 is in neighborhood #0</td>
</tr>
<tr>
<td>0</td>
<td>Machine #1 is in location #0</td>
</tr>
<tr>
<td>10 240</td>
<td>Capacities of machine #1</td>
</tr>
<tr>
<td>8 160</td>
<td>Safety capacities of machine #1</td>
</tr>
<tr>
<td>1 0 3 4</td>
<td>Moving cost from machine #1 to machines #0, #1, #2 and #3</td>
</tr>
<tr>
<td>1</td>
<td>Machine #2 is in neighborhood #1</td>
</tr>
<tr>
<td>1</td>
<td>Machine #2 is in location #1</td>
</tr>
<tr>
<td>15 100</td>
<td>Capacities of machine #2</td>
</tr>
<tr>
<td>12 80</td>
<td>Safety capacities of machine #2</td>
</tr>
<tr>
<td>4 3 0 2</td>
<td>Moving cost from machine #2 to machines #0, #1, #2 and #3</td>
</tr>
<tr>
<td>1</td>
<td>Machine #3 is in neighborhood #1</td>
</tr>
<tr>
<td>2</td>
<td>Machine #3 is in location #2</td>
</tr>
<tr>
<td>10 100</td>
<td>Capacities of machine #3</td>
</tr>
<tr>
<td>8 80</td>
<td>Safety capacities of machine #3</td>
</tr>
<tr>
<td>5 4 2 0</td>
<td>Moving cost from machine #3 to machines #0, #1, #2 and #3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>Number of services</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>spreadMin of service #0</td>
</tr>
<tr>
<td>0</td>
<td>Service #0 doesn’t depend on other services</td>
</tr>
<tr>
<td>1</td>
<td>spreadMin of service #1</td>
</tr>
<tr>
<td>1</td>
<td>Service #1 depends on one service</td>
</tr>
<tr>
<td>0</td>
<td>Service #1 depends on service #0</td>
</tr>
</tbody>
</table>
3 | Number of processes
0 | Process #0 is a process of service #0
12 10 | Requirements of process #0
1000 | Process Move Cost of process #0

0 | Process #1 is a process of service #0
10 20 | Requirements of process #1
100 | Process Move Cost of process #1

1 | Process #2 is a process of service #1
6 200 | Requirements of process #2
1 | Process Move Cost of process #2

1 | Number of balance costs
0 1 20 | Triple \(\text{resource } #0, \text{resource } #1, \text{target } 20\) for balance cost #0
10 | weight for balance cost #0

1 | Weight of Process Move Cost
10 | Weight of Service Move Cost
100 | Weight of Machine Move Cost

### 2.2 Solution input/output file format

The same file format is used to define the original solution (input) and the optimized solution (output). This file format is simply a list of assignment for all processes. As the number of processes is defined in the instance input file, machine indices are enough to define a solution.

Then the input file for the previous example is:

```
0 3 0
```

The total cost of this original solution is \((400+1300)+2500+0+0+0 = 4200\).

Moving process \(p_2\) from machine \(m_4\) to \(m_3\) reduces the balance cost from 2500 to 1700 and the load cost of resource \(r_1\) from 400 to 200. The new total cost is: \((200+1300)+1700+100+10+200 = 3510\). Then the optimal solution is achieved by moving process \(p_2\) from machine \(m_1\) to \(m_2\) with a total cost of \(400+1600+101+10+300 = 2411\).

The corresponding output file is:

```
0 2 1
```
2.3 Variable ranges for this challenge

The aim of this challenge is to concentrate on the optimization problem, therefore set sizes are limited to:

- Number of machines: 5,000
- Number of resources: 20
- Number of processes: 50,000
- Number of services: 5,000
- Number of neighborhoods: 1,000
- Number of dependencies: 5,000
- Number of locations: 1,000
- Number of balance costs: 10

All other integers are indices or 32-bits unsigned integers.

As usual in the ROADEF/EURO Challenge, three data sets will be provided:

- Data set A: number of processes is limited to 1,000. This small data set is public and is used during the qualification phase;
- Data set B: number of processes varies from 5,000 to 50,000. This medium / large data set is public and is used to evaluate proposed solvers;
- Data set X: number of processes varies from 5,000 to 50,000. This medium / large data set is private and is used to evaluate proposed solvers.

2.4 Solution checker

In order to check if a produced solution is valid or not, and to compute the total objective cost, the source code of a solution checker is available. The syntax is:

```
solution_checker instance_filename original_solution_filename new_solution_filename
```

This solution checker will be used during the challenge to evaluate produced solutions. Note the aim is to check the solution, not the instance; so the solution checker assumes the instance and the original solution are valid.