3.1 Integer Linear Programming

An application is modeled as a set of tasks \( \{v_1 \ldots v_8\} \). We want to determine the binding of the tasks to two processors: \( p_1, p_2 \). Each task \( v_i \) comes with a memory requirement \( c_{m_i}^1 \) and \( c_{m_i}^2 \) on processor \( p_1 \) and \( p_2 \), respectively. Furthermore, each task \( v_i \) comes with an energy requirement \( c_{e_i}^1 \) and \( c_{e_i}^2 \) on processor \( p_1 \) and \( p_2 \), respectively.

3.1.a) Specify an Integer Linear Program (ILP) for computing an energy-minimal partitioning.

3.1.b) Extend the ILP from a) by enforcing that an equal number of tasks is assigned to each processor.

3.1.c) Specify an ILP for computing a partitioning that respects both energy and memory requirements, but weighs the total memory usage twice as much as the total energy usage. The memory usage is the sum of all task memory requirements, and the total energy usage is the sum of the individual task energies.

3.1.d) Extend the ILP from c) by enforcing that the total memory usage for processor \( p_1 \) does not exceed the (unitless) quantity \( K \).

3.1.e) Specify an ILP for computing a partitioning that minimizes the maximal memory usage for each of the two processors.

3.2 Specification Graph

Fig. 1a) shows a task graph, and Fig. 1b) the available architecture to implement it. The four tasks of the task graph, \( T_1 \ldots T_4 \), can be executed on different components of the target architecture, as defined by Table 1.

<table>
<thead>
<tr>
<th>Component</th>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( T_1 )</td>
</tr>
<tr>
<td>MIPS</td>
<td>( \checkmark )</td>
</tr>
<tr>
<td>DSP</td>
<td>—</td>
</tr>
<tr>
<td>FPGA</td>
<td>—</td>
</tr>
<tr>
<td>ASIC</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 1: Possible bindings for tasks \( T_1 \ldots T_4 \).

3.2.a) Architecture Graph

Construct the problem graph for the task graph in Fig. 1a) by inserting communication nodes. Construct the architecture graph for the target architecture in Fig. 1b).
3.2.b) Specification Graph
Construct the specification graph. Which constraints for the binding and the allocation can be seen? Modify the target architecture to get more implementation alternatives.

3.3 Clustering
Generate partitions for the graph in Fig. 2 by hierarchical clustering.
The edge weights in Fig. 2 denote the closeness between the objects. For computing the closeness values of the newly inserted edges to the hierarchical clusters use:

(a) the average value
(b) the minimum

of the closeness values before the clustering.