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 \( c_{m,i}^{p_1} \) and \( c_{m,i}^{p_2} \) on processor \( p_1 \) and \( p_2 \), respectively. Furthermore, each task \( v_i \) comes with an energy requirement \( c_{e,i}^{p_1} \) and \( c_{e,i}^{p_2} \) on processor \( p_1 \) and \( p_2 \), respectively.

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

Solution: First, an auxiliary variable needs to be introduced. This will be a binary variable which will bind a task \( i \) to processor \( j \) when \( X_{i,j} = 1 \). We then formulate the total energy by taking into account the binding, in the following way:

\[
\min \sum_{i=1}^{8} \sum_{j=1}^{2} c_{e,i,j} \cdot X_{i,j}
\]

such that:

\[
\sum_{j=1}^{2} X_{i,j} = 1, i = 1 \ldots 8
\]

\[
X_{i,j} \in \{0, 1\}
\]

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

Solution: To do this, we only need to add one further constraint to the ILP. The auxiliary variable can be used to count the number of tasks bound to each processor:

\[
\sum_{i=1}^{8} X_{i,1} = \sum_{i=1}^{8} X_{i,2}
\]

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.
Solution: It should be noted that we have to minimize a quantity that combines both memory and energy. Here, we will use variables $c_{i,j}$ and $c_{i,j}^m$ as unitless, and we’ll use a simple weighted sum to combine them:

$$\min \left\{ \sum_{i=1}^{8} \sum_{j=1}^{2} c_{i,j}^e \cdot X_{i,j} + 2 \cdot \sum_{i=1}^{8} \sum_{j=1}^{2} c_{i,j}^m \cdot X_{i,j} \right\}$$

such that

$$\sum_{j=1}^{2} X_{i,j} = 1, \ i = 1..8 \quad (5)$$

Another approach is to give the energy and memory requirements variables units. Here, one would simply need to project them to a scalar space using carefully chosen coefficients. This would also have to be done if one wished to convert the optimal unitless weighted sum back to energy and memory units.

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 $K$.

Solution:

Just like the previous question, we use the unitless variable $c_{i,j}^m$ and we restrict its sum of tasks bound to $p_1$ to be less that the also unitless quantity $K$.

$$\sum_{i=1}^{8} X_{i,1} \cdot c_{i,1}^m < K \quad (7)$$

3.1.e)

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

Solution:

For this question, it is important to formulate the problem using linear function. If we use a function which gives the maximum memory usage, this is no longer linear, and an ILP cannot solve this. Therefore, we need to introduce another auxiliary variable $L$. We can then add restrictions to this variable such that it is greater than or equal to the memory used by each processor. This formulation ensures that the memory of each processor is evaluated independently and that their maximum is minimized.

$$\min L$$

such that

$$L \geq \sum_{i=1}^{8} c_{i,1}^m \cdot X_{i,1} \quad (8)$$

$$L \geq \sum_{i=1}^{8} c_{i,2}^m \cdot X_{i,2} \quad (9)$$

Note that trying to minimize the difference between the memory requirements is not an equivalent formulation. For starters, adding a subtraction can introduce negative values. Second, the difference is a relative value, while minimizing the maximum memory requirement concerns absolute values.
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, T1 ... T4, can be executed on different components of the target architecture, as defined by Table 1.

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.

Solution:

Binding T1 to the MIPS and T2 to the FPGA/ASIC is not feasible, since MIPS and FPGA are not directly connected in the architecture. Other pairs of bindings for which the same restriction applies are: T1 MIPS and T3 FPGA/ASIC, T2 FPGA/ASIC and T4 MIPS, T3 FPGA/ASIC and T4 MIPS. Combining the functional and connectedness restrictions, we can conclude that FPGA and ASIC can never be utilized for these settings.

A solution to resolve this problem is for instance to merge Bus 1 and Bus 2 to a common shared bus.

Table 1: Possible bindings for tasks T1 ... T4.

<table>
<thead>
<tr>
<th>Component</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPS</td>
<td>√</td>
<td>—</td>
<td>—</td>
<td>√</td>
</tr>
<tr>
<td>DSP</td>
<td>—</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>FPGA</td>
<td>—</td>
<td>√</td>
<td>√</td>
<td>—</td>
</tr>
<tr>
<td>ASIC</td>
<td>—</td>
<td>—</td>
<td>√</td>
<td>—</td>
</tr>
</tbody>
</table>
3.3 Clustering

Generate partitions for the graph in Fig. 3 by hierarchical clustering.

The edge weights in Fig. 3 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.
Figure 4: Hierarchical Clustering with "average closeness"

Figure 5: Hierarchical Clustering with "minimum closeness"