Partitioning algorithms...

The Partitioning Problem

Definition: The partitioning problem is to assign \( n \) objects \( O = \{ o_1, ..., o_n \} \) to \( m \) blocks (also called partitions) \( P = \{ p_1, ..., p_m \} \), such that

- \( p_1 \cup p_2 \cup ... \cup p_m = O \)
- \( p_i \cap p_j = \{ \} \) \( \forall i, j \) and
- cost \( c(P) \) are minimized.

In system synthesis (simple model):
- objects = data flow graph nodes
- blocks = architecture graph nodes

Partitioning Methods - Overview

- Exact methods
  - enumeration of solutions
  - Integer Linear Programs (ILP)

- Heuristic methods
  - constructive methods
    - random mapping
    - hierarchical clustering
  - iterative methods (refinement methods)
    - greedy partitioners
    - Kernighan-Lin
    - simulated annealing
    - evolutionary algorithms (design space exploration)

Integer Programming Models

Ingredients:

- Cost function
- Constraints

Involving linear expressions of integer variables from a set \( X \)

Cost function:

\[
C = \sum_{x \in X} a_i x_i \quad \text{with} \quad a_i \in \mathbb{R}, x_i \in \mathbb{N} \quad (1)
\]

Constraints:

\[
\forall j \in J : \sum_{x \in X} b_{ij} x_i \geq c_j \quad \text{with} \quad b_{ij}, c_j \in \mathbb{R} \quad (2)
\]

Def.: The problem of minimizing (1) subject to the constraints (2) is called an integer programming (IP) problem.

If all \( x_i \) are constrained to be either 0 or 1, the IP problem said to be a 0/1 integer programming problem.

An Example...

minimize: \( C = 5x_1 + 6x_2 + 4x_3 \)
subject to:

\[
\begin{align*}
& x_1 + x_2 + x_3 \geq 2 \\
& x_1, x_2, x_3 \in \{0, 1\}
\end{align*}
\]

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
</tbody>
</table>

Remarks on Integer Programming

- Maximizing the cost function can be done by setting \( C = -C \)
- Integer programming is NP-complete.
- In practice, running times can increase exponentially with the size of the problem, but problems of some thousands of variables can still be solved with commercial solvers, depending on the size and structure of the problem.
- IP models can be a good starting point for modeling, even if in the end heuristics have to be used to solve them.
**Integer Linear Program for Partitioning**

- Binary variables $x_{ij}$
  - $x_{ij} = 1$: object $i$ in block $j$
  - $x_{ij} = 0$: object $i$ not in block $j$
- Cost $c_{ij}$, if object $i$ is in block $j$

Integer linear program:

$$\begin{align*}
x_{ij} \in \{0, 1\}, & \quad 1 \leq i \leq n, 1 \leq j \leq m \\
\sum_{j=1}^{m} x_{ij} = 1, & \quad 1 \leq i \leq n \\
\text{minimize} & \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij} x_{ij}, & \quad 1 \leq j \leq m, 1 \leq i \leq n \\
\text{Additional constraints} & \quad \text{example: maximum number of } n_j \text{ objects in block } j
\end{align*}$$

$$\sum_{i=1}^{n} x_{ij} \leq n_j, 1 \leq j \leq m$$

**Hierarchical Clustering (1)**

![Hierarchical Clustering Diagram (1)](image)

- Closeness between hierarchical objects: arithmetic mean

**Hierarchical Clustering (2)**

![Hierarchical Clustering Diagram (2)](image)

**Hierarchical Clustering (3)**

![Hierarchical Clustering Diagram (3)](image)

**Greedy Hw/Sw Partitioning**

- Bi-partitioning (simplest case): $P = \{P_{SW}, P_{HW}\}$

- Software oriented approach: $P = \{O, \{\}\}$
  - in sw all functions can be realized
  - performance might be too low => migrate objects to hw

- Hardware oriented approach: $P = \{\{\}, O\}$
  - in hw the performance is ok (assumes hw is always faster than sw)
  - cost might be too high => migrate objects to sw
Greedy Hw/Sw Partitioning

- Migration of objects into the other block (HW/SW), until there is no more improvement

- Generation of bi-partitions
  - regroup the object which gives the biggest gain in cost under constraints on the balance of partition sizes (e.g., minimum cut set and maximum cut set)

- Extension:
  - Kemighan-Lin

Iterative Methods - Simulated Annealing

- From Physics:
  - metal and gas take on a minimal-energy state during cooling down (under certain constraints).
    - at each temperature, the system reaches a thermodynamic equilibrium
    - the temperature is decreased sufficiently slowly
    - probability that a particle "jumps" to a higher-energy state:
      \[ P(i, e_{min}, T) = e^{-\frac{e_{min}}{kT}} \]

- Application to Combinatorial Optimization:
  - energy = cost of a solution (partition)
  - cost decreases with temperature, sometimes (with a certain probability) increases in cost are accepted.

Simulated Annealing

```c
// Cooling down
while (not frozen) {
    P = RandomMove(P);
    cost = cost + P;
    if (Accept(deltaCost, temp) > random[0, 1]) {
        P = P;
        cost = cost;
    }
    temp = DecreaseTemp(temp);
}
```

Evolutionary Algorithms

Principles of Evolution

- Selection
- Cross-over
- Mutation

A Generic Multiobjective EA

... an Example