Tabu Search – Algorithm

1. select initial solution
2. select neighborhood set (based on current solution)
3. remove tabu solutions from set
4. set is empty
   - y: increase neighborhood
   - n: evaluate quality and select best solution from set
5. update tabu list
6. termination criteria satisfied
   - y: go back to 2
   - n: go back to 4

The brain of the algorithm is the tabu list that stores and maintains information about the history of the search.

In the most simple case a number of previous solutions are stored in the tabu list.

More advanced techniques maintain attributes of the solutions rather than the solutions itself.
Tabu Search – Organisation of the History

The history is maintained by the tabu list
Attributes of solutions are a very flexible mean to control the search

Example of attributes of a HW/SW partitioning problem with 8 tasks assigned to 1 of 4 different HW entities:
(A1) change of the value of a task assignment variable
(A2) move to HW
(A3) move to SW
(A4) combined change of some attributes
(A5) improvement of the quality of two subsequent solutions over or below a threshold value

Aspiration criteria: Under certain conditions tabus may be ignored, e.g. if
• a tabu solution is the best solution found so far
• all solutions in a neighborhood are tabu
• a tabu solution is better than the solution that triggered the respective tabu conditions

Intensification checks whether good solutions share some common properties
Diversification searches for solutions that do not share common properties
Update of history information may be recency-based or frequency-based (i.e. depending on the frequency that the attribute has been activated)
Tabu Search – Discussion

- easy to implement (at least the neighborhood search as such)
- non-trival tuning of parameters
- tuning is crucial to avoid cyclic search
- advantage of use of knowledge, i.e. feedback from the search (evaluation of solutions) to control the search (e.g. for the controlled removal of bottlenecks)
Heuristic Search Methods – Classification

Search strategy

- search area
  - global search (potentially all solutions considered)
  - local search (direct neighbors only – stepwise optimization)

- selection strategy
  - deterministic selection, i.e. according to some deterministic rules
  - random selection from the set of possible solutions
  - probabilistic selection, i.e. based on some probabilistic function

- history dependence, i.e. the degree to which the selection of the new candidate solution depends on the history of the search
  - no dependence
  - one-step dependence
  - multi-step dependence

Acceptance criteria

- deterministic acceptance, i.e. based on some deterministic function
- probabilistic acceptance, i.e. influenced by some random factor

Termination criteria

- static, i.e. independent of the actual solutions visited during the search
- dynamic, i.e. dependent on the search history
## Heuristic Search Methods – Classification

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Search strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Search area</td>
</tr>
<tr>
<td></td>
<td>local</td>
</tr>
<tr>
<td>hill-climbing</td>
<td>x</td>
</tr>
<tr>
<td>tabu search</td>
<td>x</td>
</tr>
<tr>
<td>simulated annealing</td>
<td>x</td>
</tr>
<tr>
<td>genetic algorithms</td>
<td>x</td>
</tr>
<tr>
<td>random search</td>
<td>x</td>
</tr>
</tbody>
</table>
### Heuristic Search Methods – Classification

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Search area</th>
<th>Selection strategy</th>
<th>History dependence</th>
<th>Acceptance criterion</th>
<th>Termination criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>local</td>
<td>global</td>
<td>det.</td>
<td>prob.</td>
<td>random</td>
</tr>
<tr>
<td>hill-climbing</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tabu search</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>simulated annealing</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>genetic algorithms</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random search</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Single Pass Approaches – Framework

- **derive guidelines** for solution construction
- **select subproblem**
- **decide subproblem** based on guidelines
- possibly recompute or adapt guidelines
- final solution constructed

The guidelines are crucial and represent the intelligence of the algorithm.
List Scheduling

List scheduling:

- subsequent selection of a task to be scheduled on some processor (or HW entity)
- operation is similar to a dynamic task scheduler of an operating system

assign priorities to the tasks according to some strategy

select executable task with highest priority

assign task to a processor according to some strategy

schedule complete?

priorisation strategy

assignment strategy
List Scheduling – Example (1)

Problem:
- 2 processors
- 6 tasks with precedence constraints
- find schedule with minimal execution time

Priorisation strategy **HLFET**
- length of the longest (critical) path to the sink node (node 6)

Assignment strategy
- first fit

Resulting schedule:

**Legend:**
- *green:* estimated times
- *red:* levels (priorities)
List Scheduling – Example (2)

Problem (unchanged):
- 2 processors
- 6 tasks with precedence constraints
- find schedule with minimal execution time

Priorisation strategy SCFET
(smallest co-level first with estimated times)
- length of the longest (critical) path to the source node (node 1)

Assignment strategy
- first fit

Resulting schedule:

Legend:
green: estimated times
blue: co-levels (priorities)
Clustering - Basics

Partitioning of a set of nodes in a given number of subsets

1. assign each node to a different cluster
2. compute the "distance" between any pair of clusters
3. select the pair of clusters with the highest affinity
4. merge the clusters
5. termination criteria holds

Application:
- processor assignment (load balancing – minimize interprocess communication)
- scheduling (minimize critical path)
- HW/SW partitioning

Clustering may be employed as part of the optimization process, i.e. combined with other techniques
Clustering

**probabilistic**

Each node belongs with certain probabilities to different clusters

**deterministic**

A node belongs to exactly one cluster or not

**hierarchical**

Starts with a distance matrix of each pair of nodes

Exact method: always the same result

Termination after all nodes belong to one cluster (bottom-up)

**partitioning**

Starts with given number of K clusters (independent from nodes)

Results depend on the chosen initial set of clusters

Termination after a given number of iterations (top-down)
Hierarchical Clustering

Stepwise reduction of the number of clusters

1. Determine the distance between each pair of nodes
2. Select the smallest distance
3. Replace the selected pair in distance matrix by a cluster representative
4. Recompute distance matrix

Algorithm is kind of subsequent merger of nearest neighbors (nodes/clusters)
Hierarchical Clustering

Dendrogram

Algorithm is kind of subsequent merger of nearest neighbors (nodes/clusters)
Partitioning Clustering (k-means)

Choose positions of k initial cluster representative

assign each node to the nearest cluster representative

Recompute positions of the cluster representative
   Based on the positions of the nodes in each cluster

Number of iterations reached

\(\text{\textbf{y}}\)

\(\text{\textbf{n}}\)
Clustering – Application to Load Balancing

assign each node to a different cluster

compute the sum of the communication cost between any pair of clusters

select the pair of clusters with the highest communication cost that does not violate the capacity constraints

merge the clusters

Optimization goal:
- minimize inter-process (inter-cluster) communication
- limit maximum load per processor (cluster) to 20

reduction of comm. cost without violation of constraints possible

y

n
Clustering – Application to Load Balancing (2 processors)
Clustering – Hierarchical Algorithms

Single linkage

Complete Linkage

Centroid-based

Algorithms implement different methods to compute the distance between two clusters.
Distance between groups is estimated as the smallest distance between entities.

Example:

\[
d_{(2,4)5} = \min[d_{25}, d_{45}] = d_{45} = 4.1
\]
### Clustering – Single Linkage

#### Table 1: Clustering Results

<table>
<thead>
<tr>
<th>Cluster #</th>
<th>P1</th>
<th>C24</th>
<th>P3</th>
<th>C57</th>
<th>P6</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>7.1</td>
<td>5</td>
<td>6.1</td>
<td>9.2</td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>0</td>
<td>2.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>7.1</td>
<td>3</td>
<td>5.4</td>
<td>6</td>
</tr>
<tr>
<td>P5</td>
<td>4.1</td>
<td>2.2</td>
<td>5.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>5.1</td>
<td>0</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>0</td>
<td>6</td>
<td></td>
<td>1.4</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Diagram 1: Clustering Visualization

- **Clusters:** P1, P2, P3, P4, P5, P6, P7
- **Cluster Connections:**
  - P1 connected to P2 and P4
  - P2 connected to P3
  - P3 connected to P4
  - P4 connected to P5
  - P5 connected to P6
  - P6 connected to P7

---

**Note:**
- The table values represent distances between points.
- The diagram visualizes the clustering based on these distances.
Clustering – Group Average

Distance between groups is defined as the average distance between all pairs of entities.

Example:

\[ d_{(2,4)5} = \frac{1}{2} (d_{25} + d_{45}) = 4.8 \]

<table>
<thead>
<tr>
<th>Cluster #</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>7.2</td>
<td>5</td>
<td>7.1</td>
<td>6.1</td>
<td>9.2</td>
<td>7</td>
</tr>
<tr>
<td>P2</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>1.4</td>
<td>5.4</td>
<td>3</td>
<td>6.7</td>
</tr>
<tr>
<td>P3</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>2.2</td>
<td>2.8</td>
<td>4.3</td>
<td>4.3</td>
</tr>
<tr>
<td>P4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>4.1</td>
<td>2.2</td>
<td>5.4</td>
</tr>
<tr>
<td>P5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>5.1</td>
<td>1.4</td>
</tr>
<tr>
<td>P6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>P7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>
### Clustering - Group Average

#### Cluster # P1 P2 P3 P4 P5 P6 P7
- P1 0 7.2 5 7.1 6.1 9.2 7
- C24 - 0 2.6 4.5 2.6 - -
- P3 - - 0 3.6 4.3 - -
- C57 - - - 0 5.6 - -
- P6 - - - - 0 - -

#### Cluster # P1 C243 C57 P6
- P1 0 6.4 6.1 5.4 9.2 - -
- C243 - 0 4.8 2.5 2.6 - -
- C57 - - 0 5.1 - - -
- P6 - - - - 0 - -
Clustering – Centroid-based

Determine distances between centroids \((k, l)\)
Merge centroids with the least distance

\[
d(k, l) = \sqrt{(C_{x_k} - C_{x_l})^2 + (C_{y_k} - C_{y_l})^2}
\]
Clustering – Centroid-based

Cluster # | C1   | C24  | C3   | C57  | C6
---|---|---|---|---|---
C1 | 0 | 7.1 | 5 | 6.5 | 9.2
C24 | - | 0 | 2.5 | 5.4 | 2.5
C3 | - | - | 0 | 3.5 | 4.3
C57 | - | - | - | 0 | 5.5
C6 | - | - | - | - | 0
Differences between Clustering Algorithms

- Single Linkage
- Complete Linkage
- Centroid Linkage
- K-means
- Ward
### Clustering Variants

#### Clustering methods

<table>
<thead>
<tr>
<th>Partitioning methods</th>
<th>Hierarchical methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>Agglomeration</td>
</tr>
<tr>
<td>Fuzzy-c-means</td>
<td>(bottom up)</td>
</tr>
<tr>
<td>SOM</td>
<td>Division</td>
</tr>
<tr>
<td>Clique</td>
<td>(top down)</td>
</tr>
<tr>
<td>One Pass</td>
<td>Wards</td>
</tr>
<tr>
<td>Gustafson-Kessel algorithm</td>
<td>Tree Structural Vector</td>
</tr>
<tr>
<td></td>
<td>Quantification</td>
</tr>
<tr>
<td></td>
<td>Macnaughton-Smith</td>
</tr>
<tr>
<td></td>
<td>algorithm</td>
</tr>
</tbody>
</table>

#### Distance Metrics

- Euclidean
- Manhattan
- Minkowsky
- Mahalanobis
- Jaccard
- Camberra
- Chebychev
- Correlation
- Chi-square
- Kendalls’s Rank
- Correlation
Clustering – Discussion

- Results
  - Exact results (single linkage)
  - Not-exact results → often several iterations are necessary (K-means)

- Metrics
  - Strong impact to clustering results
  - Not each metric is suitable for each clustering algorithm
  - Decision for one- or multi-criteria metrics (separated or joint clustering)

- Selection of Algorithm
  - Depends strongly on the structure of the data set and the expected results
    - Some algorithms tend to separate outliers in own clusters → some large clusters and a lot of very small clusters (complete linkage)
    - Only few algorithms are able to detect also branched, curved or cyclic clusters (single linkage)
    - Some algorithms tend to return clusters with nearly equal size (K-means, Ward)

- Quality of clustering results
  - The mean variance of the elements in each cluster (affinity parameter) is often used
  - In general the homogeneity within clusters and the heterogeneity between clusters can be measured
  - However, the quality prediction can be only as good as the quality of the used metric!
Branch and Bound with Underestimates

Application of the $A^*$ algorithm to the scheduling problem

Example: scheduling on a 2-processor system (processors A and B)

Process graph

```
1  5  2
  5
2  8  3
  9
4  3
```

f(x) = g(x) + h(x)
g(x) exact value of partial schedule
h(x) underestimate for remainder (rem)

\[ h(x) = \min (\text{altern.rem.proc}, \text{rem.comm.} + \text{rem.proc.}) \]

x = start, then x = best of X, where X = growing set of known solutions (min of comm+proc.)

\[ f(1) = 5 + \min((9 + 3), (2+8+3)) = 5+12 = 17 \]

Scheduled to: A               B

Search is terminated when min \{f(x)\} is a terminal node (in the search tree)
Branch and Bound with Underestimates

Example: computation of $f(3)$

Diagram:

- Case 1: $A = \text{path } 1-2-4$
  - $g(3) = 5 + 8 = 13$
  - $h(3) = \min(3, (5+9+3))$
  - $f(3) = 16$

- Case 2: $A = \text{path } 1-3-4$
  - $g(3) = 5$
  - $h(3) = 5 + 9 + 3$
  - $f(3) = 22$

Math expressions:

$$f(x) = g(x) + h(x)$$

- $g(x)$: exact value of partial schedule
- $h(x)$: underestimate for remainder

Table:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Timeline diagrams for cases 1 and 2.
Berechnung

- Annahme: Kommunikation auf gleichem Processor läuft parallel zur CPU
  - $f(1) = 5 + 9 + 3 = 17$  $f(2)$ Gleiches Ergebnis, da egal, ob Prozess 1 auf A oder B läuft
  - $f(3) = 5 + 8 + \min(5 + 9 + 3, 6 + 3) = 22$
    Schedule: Prozesse 1 und 2 auf Prozessor A
  - $f(4) = 5 + 2 + 8 + \min(3; 9 + 1 + 3) = 18$
    Prozess 1 auf A, 2 auf B (daher +2 Kommunikation und proc 8+5)
  - $f(5) = 5 + 9 + \min(1 + 3; 2 + 8 + 3) = 18$
    Schedule: Prozesse 1 und 3 auf Prozessor A
- $f(6) = f(3)$
- $f(7)$ = weitere Fallunterscheidungen mit Rest
Branch and Bound with Underestimates

Application of the A* algorithm to the scheduling problem

Example: scheduling on a 2-processor system (processors A and B)

Process graph

Search Tree

Legend:
- **green**: processing times
- **blue**: communication times

\[ f(x) = g(x) + h(x) \]
- \( g(x) \) exact
- \( h(x) \) underestimate rest

Search is terminated when \( \min \{f(x)\} \) is a terminal node (in the search tree)
References

Motivation for the Course – Why is this important?

What are `Integrated HW/SW-Systems`? Any computer system consists of hardware and software! But: HW is often hidden and not considered important by SW developers.

Indicators that HW is important:

- capacity
- responsiveness and delay
- predictability
- reliability
- safety
- power consumption
- cost
- ...

Systems where HW/SW relation is obvious:

- embedded systems
- real-time systems
- reliable systems
- safety-critical systems

=> Knowledge of HW/SW interaction is required!
Content IHS 2

- Motivation and overview
- Development process and tasks
- System requirements
- Behavioral models overview
  - FSM, NDFSM, FSM composition
  - PN, DFG, CFG, CDFG
- Specification languages details
  - Statecharts
  - SDL
  - VHDL
- Optimization
- Performance evaluation
- High-level Synthesis

IHS 3 (for M.S. students)
Details on
- Validation,
- Testing
- Fault coverage
- Structural Test
- Functional Test
- Functional validation
- Performance/temporal validation