Lecture 8

- Optimization problems
- Heuristic techniques
  - Simulated annealing
- Genetic algorithms
Mathematical Optimization

- Many design and synthesis tasks can be formulated as:
  
  \[
  \begin{align*}
  \text{Minimize} & \quad f(x) \\
  \text{Subject to} & \quad g_i(x) \geq b_i; \quad i = 1, 2, ..., m;
  \end{align*}
  \]
  
  where
  
  \( x \) is a vector of decision variables \((x \geq 0)\);
  \( f \) is the cost (objective) function;
  \( g_i \)'s are a set of constraints.

- If \( f \) and \( g_i \) are linear functions, we have a linear programming (LP) problem.
- LP can be solved by the simplex algorithm, which is an exact method.
  - It will always identify the optimal solution if it exists.

Combinational Optimization (CO)

- In many design problems, the decision variables are discrete.
  - The solution is a set, or a sequence, of integers or other discrete objects.

- Ex. A feasible solution for the graph coloring problem (with \( n \) nodes and \( m \) colors) is represented as
  
  \( x_i = j; \quad j \in \{1, 2, ..., m\}, \quad i = 1, 2, ..., n. \)

- Such problems are called combinatorial optimization problems.
ILP Formulation of Scheduling

- RC-scheduling is to decide on $x_{i,j}$ such that $x_{i,j}$ equals 1 if operation $i$ is scheduled at step $j$, and 0 otherwise ($i = 1, 2, ..., n; j = 1, 2, ..., m$).
- A set of constraints can be formulated:
  - C1: The start time of each operation should be unique: for $i$
    \[
    \sum_{j=1}^{m} x_{i,j} = 1
    \]
    The start time of an operation is consequently given as $(i = 1, 2, ..., n)$:
    \[
    t_i = \sum_{j=1}^{m} j \cdot x_{i,j}
    \]

Constraints for Scheduling

- C2: The precedence relation must be preserved:
  \[
  \sum_{j=1}^{m} j \cdot x_{p,j} \geq \sum_{j=1}^{m} j \cdot x_{q,j} + d_q, \quad p, q = 1, 2, ..., n : (v_q, v_p) \in E
  \]
- C3: The resource bounds must be met at every schedule step, i.e., the number of all operations at step $j$ of type $k$ must not larger than $a_k$:
  \[
  \sum_{i: t(v) = k} \sum_{p=j-d+1}^{i} x_{i,p} \leq a_k, \quad k = 1, 2, ..., n_{\text{res}} \quad j = 1, 2, ..., m
  \]
ILP Formulation

The resource-constrained scheduling problem can thus be stated as:

- To minimize $c^T t$ such that the three constrains $C1$, $C2$, and $C3$ are satisfied,
  - $t$ is the vector whose entries are the start times of the operations, i.e., $t = [t_1, t_2, ..., t_n]$.
- If $c = [0, 0, ..., 0, 1]^T$, this corresponds to minimizing the latency of the schedule, since $c^T t = t_n$ ($t_n$ is the start time of the sink operation).
- If $c = [1, 1, ..., 1]^T$, it corresponds to finding the earliest start times of all operations.

Features of CO Problems

- Most CO problems, e.g., system partitioning with constraints, for digital system designs are NP-complete.
- The time needed to solve an NP-complete problem grows exponentially with respect to the problem size $n$.
- For example, to enumerate all feasible solutions for a scheduling problem (all possible permutation), we have:
  - 20 operations in 1 hour (our assumption);
  - 21 operations in 20 hour;
  - 22 operations in 17.5 days;
  - ...
  - 25 operations in 6 centuries.
NP-Complete Problems

- Approaches for solving such problems exactly are usually based on *implicit* enumeration of the feasible solutions.
- They require, in the worst-case, an exponential number of steps.
- When the problem size, $n$, becomes large, as in most real-world applications, no exponential algorithms could be of practical use.
  - They are typically called intractable problems.

Lecture 8

- Optimization problems
  - Heuristic techniques
    - Simulated annealing
  - Genetic algorithms
Heuristics

- A heuristic seeks near-optimal solutions at a reasonable computational cost without being able to guarantee optimality or feasibility.

Motivations:
- Many exact algorithms involve a huge amount of computation effort.
- The decision variables have frequently complicated inter-dependencies.
- We have often nonlinear cost functions and constraints, even no mathematical functions.
- Ex. The cost function \( f \) can, for example, be defined by a computer program (e.g., for power estimation).

Heuristic Approaches to CO

<table>
<thead>
<tr>
<th>Problem specific</th>
<th>Generic methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructive</td>
<td></td>
</tr>
<tr>
<td>• List scheduling</td>
<td>• Divide and conquer</td>
</tr>
<tr>
<td>• Greedy allocation/binding</td>
<td>• Branch and bound</td>
</tr>
<tr>
<td>Transformational</td>
<td></td>
</tr>
<tr>
<td>Iterative improvement</td>
<td>• Neighborhood search</td>
</tr>
<tr>
<td>• Kernighan-Lin algorithm for graph partitioning</td>
<td>• Simulated annealing</td>
</tr>
<tr>
<td></td>
<td>• Tabu search</td>
</tr>
<tr>
<td></td>
<td>• Genetic algorithms</td>
</tr>
</tbody>
</table>
Branch-and-Bound

- Traverse an implicit tree to find the best leaf (solution).

4-City Travelling Salesman Problem

```
0 1 2 3
0 0 3 6 41
1 0 40 5
2 0 4
3 0
```

Total cost of this solution = 88

Branch-and-Bound Ex

- Low-bound on the cost function.
- Search strategy

```
0 1 2 3
0 0 3 6 41
1 0 40 5
2 0 4
3 0
```

- \( L \geq 0 \)
- \( \{0\} \)
- \( L \geq 3 \)
- \( \{0,1\} \)
- \( L \geq 43 \)
- \( \{0,1,2\} \)
- \( L = 88 \)
- \( \{0,1,2,3\} \)
- \( L = 88 \)

- \( L \geq 6 \)
- \( \{0,2\} \)
- \( \{0,1,2\} \)
- \( L \geq 46 \)
- \( \{0,1,3\} \)
- \( L = 92 \)
- \( \{0,2,1,3\} \)
- \( L = 10 \)
- \( \{0,2,3,1\} \)
- \( L = 18 \)
- \( \{0,3\} \)
- \( L \geq 41 \)
- \( \{0,3,1\} \)
- \( L \geq 46 \)
- \( \{0,3,1,2\} \)
- \( L = 88 \)
- \( \{0,3,2\} \)
- \( L \geq 45 \)
- \( \{0,3,2,1\} \)
- \( L = 88 \)
Neighborhood Search Method

- **Step 1 (Initialization)**
  
  (A) Select a starting solution \( x_{\text{now}} \in X \).
  
  (B) \( x_{\text{best}} = x_{\text{now}}, \ best\_cost = c(x_{\text{best}}) \).

- **Step 2 (Choice and termination)**
  
  Choose a solution \( x_{\text{next}} \in N(x_{\text{now}}) \), in the neighborhood of \( x_{\text{now}} \).
  
  If no solution can be selected, or the terminating criteria apply, then the algorithm terminates.

- **Step 3 (Update)**
  
  Re-set \( x_{\text{now}} = x_{\text{next}} \).
  
  If \( c(x_{\text{now}}) < \ best\_cost \), perform Step 1(B).
  
  Goto Step 2.

The Descent Method

- **Step 1 (Initialization)**

- **Step 2 (Choice and termination)**
  
  Choose \( x_{\text{next}} \in N(x_{\text{now}}) \) such that \( c(x_{\text{next}}) < c(x_{\text{now}}) \), and terminate if no such \( x_{\text{next}} \) can be found.

- **Step 3 (Update)**

  The descent process can easily be stuck at a local optimum:
Dealing with Local Optimality

- Enlarge the neighborhood.
- Start with different initial solutions.
- To allow “uphill moves”:
  - Simulated annealing
  - Tabu search

Lecture 8

- Optimization problems
- Heuristic techniques
  - Simulated annealing
- Genetic algorithms
Annealing

Annealing is the slow cooling of metallic material after heating:
- A solid material is heated past its melting point.
- Cooling is then slowly done.
- When cooling stops, the material settles usually into a low energy state (corresponding to a stable structure).
- In this way, the properties of the materials are improved.

The annealing process can be viewed intuitively as:
- At high temperature, the atoms are randomly oriented due to their high energy caused by heat.
- When the temperature reduces, the atoms tend to line up with their neighbors, but different regions may have different directions.
- If cooling is done slowly, the final frozen state will have a near-minimal energy state.
Simulated Annealing

Annealing can be simulated using computer simulation:

- Generate a random perturbation of the atom orientations, and calculate the resulting energy change.
- If the energy has decreased, the system moves to this new state.
- If energy has increased, the new state is accepted according to the laws of thermodynamics:
  At temperature $t$, the probability of an increase in energy of magnitude $\Delta E$ is given by
  $$ p(\Delta E) = \frac{1}{e^{\frac{\Delta E}{k \cdot t}}} $$
  where $k$ is called the Boltzmann’s constant.

Probability of Accepting Higher-Energy States

$$ p(\Delta E) = \frac{1}{e^{\frac{\Delta E}{k \cdot t}}} $$
Simulated Annealing for CO

- The SA algorithm could be applied to combinatorial optimization:

<table>
<thead>
<tr>
<th>Therodynamic simulation</th>
<th>Combinatorial optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>System states</td>
<td>Feasible solutions</td>
</tr>
<tr>
<td>Energy</td>
<td>Cost</td>
</tr>
<tr>
<td>Change of state</td>
<td>Moving to a neighboring solution</td>
</tr>
<tr>
<td>Temperature</td>
<td>“Control parameter”</td>
</tr>
<tr>
<td>Frozen state</td>
<td>Final solution</td>
</tr>
</tbody>
</table>

Simulated Annealing for CO

- SA is a modified descent method for neighborhood search.
- It allows uphill moves in a controlled way.

- The frequency of accepting uphill moves is governed by a probability function which changes as the algorithm progresses.
The SA Algorithm

Select an initial solution \( x_{\text{now}} \in X \);
Select an initial temperature \( t > 0 \);
Select a temperature reduction function \( \alpha \);

Repeat

Repeat

Randomly select \( x_{\text{next}} \in N(x_{\text{now}}) \);
\[ \delta = \text{cost}(x_{\text{next}}) - \text{cost}(x_{\text{now}}) \];
If \( \delta < 0 \) then \( x_{\text{now}} = x_{\text{next}} \)
else generate randomly \( p \) in the range \((0, 1)\) uniformly;
If \( p < \exp(-\delta/t) \) then \( x_{\text{now}} = x_{\text{next}} \);

Until iteration_count = nrep;
Set \( t = \alpha(t) \);

Until stopping condition = true.
Return \( x_{\text{now}} \) as the approximation to the optimal solution.

A HW/SW Partitioning Example

![Graph showing the cost function value over iterations. The optimum is indicated at iteration 1000.](image)
The Cooling Schedule I

- Initial temperature (IT):
  - IT must be "hot" enough to allow an almost free exchange of neighborhood solutions, if the final solution is to be independent of the starting one.
  - Simulating the heating process:
    - A system can be first heated rapidly until the proportion of accepted moves to rejected moves reaches a given value; e.g., when 80% of moves leading to higher costs will be accepted.
    - Cooling will then start.

The Cooling Schedule II

- Temperature reduction scheme:
  - A large number of iterations at few temperatures or a small number of iterations at many temperatures.
  - Typically \( \alpha(t) = a \times t \), where \( a < 1 \);
    - \( a \) should be large, usually between 0.8 and 0.99.
  - For better results, the reduction rate should be slower in middle temperature ranges.

- Stopping conditions:
  - Zero temperature - the theoretical requirement.
  - A number of iterations or temperatures has passed without any acceptance of moves.
  - A given total number of iterations have been completed (or a fixed amount of execution time).
Problem-Specific Decisions

- The neighborhood structure should be defined such that:
  - All solutions should be reachable from each other.
  - Easy to generate randomly a neighboring feasible solution.
  - Penalty for infeasible solutions, if the solution space is strongly constrained.
  - The cost difference between $s$ and $s_0$ should be able to be efficiently calculated.
  - The size of the neighborhood should be kept reasonably small.

- Many decision parameters must be fine-tuned based on experimentation on typical data.

Lecture 8

- Optimization problems
- Heuristic techniques
  - Simulated annealing
- Genetic algorithms
Genetic Algorithms

Evolution of a population:

- current generation
- next generation

Evolution:
- Selection and Alternation
- Mutation
- Crossover

The Simple Genetic Algorithm

begin
  
t = 0;
  Initialize P(t);
  while termination criterion not reached do
    begin
      Evaluate P(t);
      Select P(t + 1) from P(t);
      Alter P(t + 1);
      t = t + 1;
    end
  end
end

Evaluation: Based on the cost function.
Selection: Proportionate selection is usually used.
**The Roulette Wheel Selection Scheme**

### Genetic Operator I

- Crossover of chromosomes, controlled by a crossover rate:

  - 01101000101010101111
  - 11001110100010001011
  - 11001000101010101111
  - 01101110100010001011
  - 01101000101010101111
  - 11001110100010001011

...
Genetic Operator II

- Mutation, controlled by a mutation rate:

  0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 1 1 1
  \[\rightarrow\]
  0 1 1 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1

- Survival, a fixed number of best individuals will copy themselves.

Control Parameters

- Population size:
  - typical: 50 – 200
  - trade-off: small v. large population size
- Crossover rate: 0.6 - 0.9
- Mutation rate: 0.001 - 0.01
- Stopping conditions:
  - Numbers of generations
  - Progress
  - Diversity of population
- Selection mechanisms:
  - Experiments
  - Analysis
  - Adapting parameter settings
Summary

- Design and synthesis tasks are often optimization problems.
- Due to the complexity of the optimization problem, heuristic algorithms are widely used.
- Many general heuristics are based on neighborhood search principles.
- SA is applicable to almost any combinatorial optimization problem, and very simple to implement.
- Genetic algorithms simulate the evolution process of nature and are widely applicable.