Standard Optimization Techniques

Peter Marwedel
TU Dortmund, Informatik 12
Germany

These slides use Microsoft clip arts. Microsoft copyright restrictions apply.
Structure of this course

Numbers denote sequence of chapters

2: Specification & modeling
3: ES-hardware
4: system software (RTOS, middleware, …)

Design repository

6: Application mapping
7: Optimization
5: Evaluation & validation (energy, cost, performance, …)

8: Test

[“Appendix”: Standard Optimization Techniques]
Integer linear programming models

Ingredients:
- Cost function
- Constraints

Involving linear expressions of integer variables from a set $X$

Cost function

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

Constraints:
$$\forall j \in J : \sum_{x_i \in X} b_{i,j} x_i \geq c_j \text{ with } b_{i,j}, c_j \in \mathbb{R} \quad (2)$$

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

If all $x_i$ are constrained to be either 0 or 1, the ILP problem is said to be a 0/1 integer linear programming problem.
Example

\[ C = 5x_1 + 6x_2 + 4x_3 \]
\[ x_1 + x_2 + x_3 \geq 2 \]
\[ x_1, x_2, x_3 \in \{0,1\} \]

<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>

Optimal
Remarks on integer programming

- Maximizing the cost function: just set \( C' = -C \)
- Integer programming is NP-complete.
- Running times depend exponentially on problem size, but problems of >1000 vars solvable with good solver (depending on the size and structure of the problem)
- The case of \( x_i \in \mathbb{R} \) is called linear programming (LP). Polynomial complexity, but most algorithms are exponential, in practice still faster than for ILP problems.
- The case of some \( x_i \in \mathbb{R} \) and some \( x_i \in \mathbb{N} \) is called mixed integer-linear programming.
- ILP/LP models good starting point for modeling, even if heuristics are used in the end.
- Solvers: lp_solve (public), CPLEX (commercial), …
Evolutionary Algorithms (1)

- **Evolutionary Algorithms** are based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a given problem.

- The population is arbitrarily initialized, and it evolves towards better and better regions of the search space by means of randomized processes of
  - **selection** *(which is deterministic in some algorithms)*,
  - **mutation*, and
  - **recombination** *(which is completely omitted in some algorithmic realizations)*.

[Bäck, Schwefel, 1993]
Evolutionary Algorithms (2)

- The environment (given aim of the search) delivers a quality information (fitness value) of the search points, and the selection process favours those individuals of higher fitness to reproduce more often than worse individuals.
- The recombination mechanism allows the mixing of parental information while passing it to their descendants, and mutation introduces innovation into the population.

[Bäck, Schwefel, 1993]
Evolutionary Algorithms

Principles of Evolution

1. Selection

2. Mutation

3. Cross-over

© Thiele
An Evolutionary Algorithm in Action

max. $y_2$

min. $y_1$

hypothetical trade-off front
Issues in Multi-Objective Optimization

- How to maintain a diverse Pareto set approximation?
  ② density estimation

- How to prevent nondominated solutions from being lost?
  ③ environmental selection

- How to guide the population towards the Pareto set?
  ① fitness assignment
A Generic Multiobjective EA

population -> evaluate sample vary -> new population

archive -> update truncate -> new archive
### Example: SPEA2 Algorithm

<table>
<thead>
<tr>
<th>Step 1:</th>
<th>Generate initial population $P_0$ and empty archive (external set) $A_0$. Set $t = 0$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2:</td>
<td>Calculate fitness values of individuals in $P_t$ and $A_t$.</td>
</tr>
<tr>
<td>Step 3:</td>
<td>$A_{t+1} = \text{nondominated individuals in } P_t \text{ and } A_t$. If size of $A_{t+1} &gt; N$ then reduce $A_{t+1}$, else if size of $A_{t+1} &lt; N$ then fill $A_{t+1}$ with dominated individuals in $P_t$ and $A_t$.</td>
</tr>
<tr>
<td>Step 4:</td>
<td>If $t &gt; T$ then output the nondominated set of $A_{t+1}$. Stop.</td>
</tr>
<tr>
<td>Step 5:</td>
<td>Fill mating pool by binary tournament selection.</td>
</tr>
<tr>
<td>Step 6:</td>
<td>Apply recombination and mutation operators to the mating pool and set $P_{t+1}$ to the resulting population. Set $t = t + 1$ and go to Step 2.</td>
</tr>
</tbody>
</table>
Summary

Integer (linear) programming

- Integer programming is NP-complete
- Linear programming is faster
- Good starting point even if solutions are generated with different techniques

Simulated annealing

- Modeled after cooling of liquids
- Overcomes local minima

Evolutionary algorithms

- Maintain set of solutions
- Include selection, mutation and recombination