4. Design Space Exploration of Embedded Systems

© Lothar Thiele
ETH Zurich, Switzerland
Contents of Lectures (Lothar Thiele)

1. Introduction to Embedded System Design
2. Software for Embedded Systems
3. Real-Time Scheduling
4. Design Space Exploration
5. Performance Analysis
Topics

► Multi-Objective Optimization
  ▪ Introduction
  ▪ Multi-Objective Optimization
  ▪ Algorithm Design
    • Design Choices
    • Representation
    • Fitness and Selection
    • Variation Operators
  ▪ Implementation

► Design Space Exploration
  ▪ System Design
  ▪ Problem Specification
  ▪ Example
Evolutionary Multiobjective Optimization Algorithms

What are Evolutionary Algorithms?

- randomized, **problem-independent** search heuristics
  → applicable to black-box optimization problems

How do they work?

- by iteratively improving a **population** of solutions by variation and selection
  → can find many different optimal solution in a single run
Black-Box Optimization

**Objective Function**

- Decision vector $x$
- Objective vector $f(x)$

(e.g. simulation model)

**Optimization Algorithm:**

- Only allowed to evaluate $f$
- (direct search)
The Knapsack Problem

Goal: choose subset that
- maximizes overall profit
- minimizes total weight

<table>
<thead>
<tr>
<th>weight</th>
<th>profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>750g</td>
<td>5</td>
</tr>
<tr>
<td>1500g</td>
<td>8</td>
</tr>
<tr>
<td>300g</td>
<td>7</td>
</tr>
<tr>
<td>1000g</td>
<td>3</td>
</tr>
</tbody>
</table>
The Solution Space
Observations: ① there is no single optimal solution, but ② some solutions are better than others.

Selecting a solution

Finding the good solutions
Decision Making: Selecting a Solution

Approaches:
- profit more important than cost (ranking)
- weight must not exceed 2400g (constraint)
Optimization Alternatives

- Use of *classical single objective optimization* methods
  - simulated annealing, tabu search
  - integer linear program
  - other constructive or iterative heuristic methods
- *Decision making* (weighting the different objectives) is done *before the optimization*.

- *Population based optimization methods*
  - evolutionary algorithms
  - genetic algorithms
- *Decision making* is done *after the optimization*. 
Optimization Alternatives

Scalarization
- Weighted sum

Population-based
- SPEA2

Parameter-oriented
- Scaling-dependent

Set-oriented
- Scaling-independent
Scalarization Approach

multiple objectives

\((y_1, y_2, \ldots, y_k)\)

transformation

parameters

\((w_1, w_2, \ldots, w_k)\)

single objective

\(y = w_1y_1 + \ldots + w_ky_k\)

example: weighting approach

maximization problem
A Generic Multiobjective EA

- **Population**
- **Archive**
- **Evaluate**
- **Sample**
- **Vary**
- **Update**
- **Truncate**

- **New Population**
- **New Archive**
An Evolutionary Algorithm in Action

max. $y_2$

min. $y_1$

hypothetical trade-off front
Design Space Exploration

- Specification
- Optimization
- Evaluation
- Implementation
Packet Processing in Networks

Embedded Internet Devices

Wearable Computing

Mobile Internet

Access

Core

method (a)(fsd)
for i=1 to
null
end for

call comm(a,dsf,*e);
end for

©UCB Rabaey
Network processor = high-performance, programmable device designed to efficiently execute communication workloads [Crowley et al.: 2003]
Optimization Scenario: Overview

**Given:**

1. specification of the task structure \((\text{task model})\) = for each flow the corresponding tasks to be executed
2. different usage scenarios \((\text{flow model})\) = sets of flows with different characteristics

**Sought:**

network processor implementation \((\text{resource model})\) = architecture + task mapping + scheduling

**Objectives:**

1. maximize performance
2. minimize cost

**Subject to:**

1. memory constraint
2. delay constraints
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - Representation
  - Fitness and Selection
  - Variation Operators
- Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
Dominance, Pareto Points

- A (design) point $J_k$ is *dominated* by $J_i$, if $J_i$ is
  - better or equal than $J_k$ in all criteria and
  - better in at least one criterion.

- A point is Pareto-optimal or a *Pareto-point*, if it is not dominated.

- The domination relation imposes a partial order on all design points
  - We are faced with a set of optimal solutions.
  - Divergence of solutions vs. convergence.
Multi-objective Optimization

Definition 1 (Dominance relation)
Let \( f, g \in \mathbb{R}^m \). Then \( f \) is said to dominate \( g \), denoted as \( f \succ g \), iff

1. \( \forall i \in \{1, \ldots, m\} : f_i \geq g_i \)
2. \( \exists j \in \{1, \ldots, m\} : f_j > g_j \)

Definition 2 (Pareto set)
Let \( F \subseteq \mathbb{R}^m \) be a set of vectors. Then the Pareto set \( F^* \subseteq F \) is defined as follows: \( F^* \) contains all vectors \( g \in F \) which are not dominated by any vector \( f \in F \), i.e.

\[
F^* := \{ g \in F | \forall f \in F : f \succ g \} \tag{1}
\]
Multi-objective Optimization

Maximize \((y_1, y_2, \ldots, y_k) = f(x_1, x_2, \ldots, x_n)\)

Pareto set = set of all Pareto-optimal solutions

Pareto optimal = not dominated

-dominated

-worse

-better

incomparable

incomparable
Randomized (Black Box) Search Algorithms

Idea: find good solutions without investigating all solutions

Assumptions: better solutions can be found in the neighborhood of good solutions
information available only by function evaluations

$t = 1$: (randomly) choose a solution $x_1$ to start with

$t \geq t+1$: (randomly) choose a solution $x_{t+1}$ using solutions $X_1, \ldots, X_t$
Types of Randomized Search Algorithms

- **EA** (evolutionary algorithm) ≥ 1
- **TS** (tabu search) 1
- **SA** (simulated annealing) 1

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Memory</th>
<th>Selection</th>
<th>Variation</th>
<th>Mating Selection</th>
<th>Environmental Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EA</strong></td>
<td>≥ 1</td>
<td>both</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TS</strong></td>
<td>1</td>
<td>no mating</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SA</strong></td>
<td>1</td>
<td>no mating</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Limitations of Randomized Search Algorithms

The No-Free-Lunch Theorem

All search algorithms provide in average the same performance on all possible functions with finite search and objective spaces.

[Wolpert, McReady: 1997]

Remarks:

- Not all functions equally likely and realistic
- We cannot expect to design the algorithm beating all others
- Ongoing research: which algorithm suited for which class of problem?
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
  - **Algorithm Design**
    - *Design Choices*
    - Representation
    - Fitness and Selection
    - Variation Operators
  - Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
Design Choices

representation fitness assignment mating selection

parameters

environmental selection variation operators
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
Comparison of Three Implementations

2-objective knapsack problem

- SPEA2
- VEGA
- extended VEGA

Trade-off between distance and diversity?
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - **Representation**
  - Fitness
  - Variation Operators
- Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
Representation

search space  decoder  solution space  objectives  objective space

solutions encoded by vectors, matrices, trees, lists, ...

Issues:

• completeness (each solution has an encoding)
• uniformity (all solutions are represented equally)
• redundancy (cardinality of search space vs. solution space)
• feasibility (each encoding maps to a feasible solution)
**Example: Binary Vector Encoding**

**Given:** graph

**Goal:** find minimum subset of nodes such that each edge is connected to at least one node of this subset (minimum vertex cover)

<table>
<thead>
<tr>
<th>nodes selected?</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>…</td>
</tr>
</tbody>
</table>
Example: Integer Vector Encoding

**Given:** graph, k colors

**Goal:** assign each node one of the k colors such that the number of connected nodes with the same color is minimized (graph coloring problem)

<table>
<thead>
<tr>
<th>nodes</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>colors</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>...</td>
</tr>
</tbody>
</table>
Example: Real Vector Encoding

\[ G2(\vec{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2 \prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \]

parameters \[ x_1 \quad x_2 \quad x_3 \quad x_4 \quad \ldots \quad x_n \]

values \[
\begin{array}{cccccc}
0.33 & 0.53 & 1.03 & 3.25 & \ldots & 9.83
\end{array}
\]

[Michalewicz, Fogel: How to Solve it. Springer 2000]
Tree Example: Parking a Truck

Goal: find function $c$ with $u = c(x, y, d, t)$
Search Space for the Truck Problem

Operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLUS(a,b)</td>
<td>returns a+b</td>
</tr>
<tr>
<td>MINUS(a,b)</td>
<td>returns a−b</td>
</tr>
<tr>
<td>MUL(a,b)</td>
<td>returns a*b</td>
</tr>
<tr>
<td>DIV(a,b)</td>
<td>return a/b, if b &lt;&gt; 0, else 1</td>
</tr>
<tr>
<td>ATG(a,b)</td>
<td>returns atan2(a,b), if a&lt;&gt; 0, else 0</td>
</tr>
<tr>
<td>IFLTZ(a,b,c)</td>
<td>returns b, if a&lt;0, else returns c</td>
</tr>
</tbody>
</table>

Arguments:

- X: position x
- Y: position y
- DIFF: cab angle d
- TANG: trailer angle t

Search space: set of symbolic expression using the above operators and arguments
Example Solution: Tree Representation

\[ u = (x - d) \times (y + t) \]

The tree representation encodes the function (symbolic expression): \( u = (x - d) \times (y + t) \)
A Solution Found by an EA

truck simulation

encoded tree
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - Representation
  - Fitness and Selection
  - Variation Operators
- Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
**Fitness Assignment**

**Fitness** = scalar value representing quality of an individual (usually). Used for mating and environmental selection

\[ F_i = f( m(i) ) \]

**Difficulties:**
- multiple objectives have to be considered (Pareto set is sought)
- multiple optima need to be approximated (how to consider diversity?)
- constraints are involved which have to be met
Example Pareto Ranking

- **Fitness function:**
  \[
  F(J) = \sum_{i=1,\ldots,N, J \neq J_i} \left\{ \begin{array}{ll}
  1 & : J_i < J \\
  0 & : \text{else}
  \end{array} \right.
  \]

execution time

\[
\begin{align*}
F(1) &= 3 \\
F(2) &= 1 \\
F(3) &= 1 \\
F(4) &= 2 \\
F(5) &= 1 \\
F(6) &= 0
\end{align*}
\]
Constraint Handling

Constraint = \( g(x_1, x_2, \ldots, x_n) \geq 0 \)

\( g \geq 0 \) \quad \text{feasible}

\( g < 0 \) \quad \text{infeasible}

**Approaches:**

- construct initialization and variation such that infeasible solutions are not generated (resp. not inserted)
- representation is such that decoding always yields a feasible solution
- calculate constraint violation (\(- g(x_1, x_2, \ldots, x_n)\)) and incorporate it into fitness, e.g., \( F_i = f(m(i)) - g(x_1, x_2, \ldots, x_n) \) (fitness to be maximized)
- code constraint as a new objective
Two conflicting goals:

- **exploitation** (converge fast)
- **exploration** (avoid getting stuck)

Two types of selection:

- **mating selection** = select for variation
- **environmental selection** = select for survival
Example: Tournament Selection

= integrated sampling rate assignment and sampling

1. uniformly choose $T$ individuals at random independently of fitness
2. compare fitness and copy best individual in mating pool

$T = \text{tournament size} \ (\text{binary tournament selection means } T=2)$
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - Representation
  - Fitness and Selection
  - Variation Operators
- Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
Example: Vector Mutation

Bit vectors:

1 0 1 1 1 0

each bit is flipped with probability 1/6

1 0 0 1 1 0

Permutations:

1 2 3 4 5 6

swap

1 2 3 4 5 6

rearrange

1 0 1 1 1 0

1 4 3 2 5 6

1 3 4 2 5 6
Example: Vector Recombination

Bit vectors:

\[
\begin{array}{cccccc}
1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccc}
1 & 1 & 0 & 0 & 0 & 1 \\
\end{array}
\]

Permutations:

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
6 & 2 & 3 & 4 & 1 & 5 \\
\end{array}
\]

parents

child

\[
\begin{array}{cccccc}
1 & 2 & 3 & 6 & 4 & 5 \\
\end{array}
\]
Example: Recombination of Trees

MULT

PLUS

MINUS

DIFF

Y

TANG

X

Y

exchange
### 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>
**Idea (Step 2):** calculate dominance rank weighted by dominance count

**Note:** lower raw fitness = better quality
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - Representation
  - Fitness and Selection
  - Variation Operators
- **Implementation**
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
What Is Needed…

A framework that
- Provides ready-to-use modules (algorithms / applications)
- Is simple to use
- Is independent of programming language and OS
- Comes with minimum overhead

Idea: separate problem-dependent from problem-independent part
PISA: Implementation

application independent:
- mating / environmental selection
- individuals are described by IDs and objective vectors

handshake protocol:
- state / action
- individual IDs
- objective vectors
- parameters

application dependent:
- variation operators
- stores and manages individuals

selector process

shared file system

text files

variator process
Additional information about several topics:
- PISALib for easy implementation of the interface in C available (see How to write a module?)
- New modules: LOTZ2, Knapsack, FEMO, NSGA2

Optimization Problems (variant)

LOTZ - Demonstration Program (more...)
- Source in C
- Binaries: Solaris, Windows, Linux

LOTZ2 - Leading Ones Trailing Zeros (more...)
- Source in C
- Binaries: Solaris, Windows, Linux

Knapsack Problem (more...)
- Source in C
- Binaries: Solaris, Windows, Linux

EXPO - Network Processor Design Problem (more...)
- Binaries: (incl. JRE 1.4.1) Solaris, Windows, Linux
- Binaries: (pure JAVA, no JRE) All platforms

Optimization Algorithms (selector)

SEMO - Demonstration Program (more...)
- Source in C
- Binaries: Solaris, Windows, Linux

FEMO - Fair Evolutionary Multiobjective Optimizer (more...)
- Source in C
- Binaries: Solaris, Windows, Linux

SPEA2 - Strength Pareto Evolutionary Algorithm 2 (more...)
- Source in C
- Binaries: Solaris, Windows, Linux

NSGA2 - Nondominated Sorting Genetic Algorithm 2 (more...)
- Source in C
- Binaries: Solaris, Windows, Linux

Last updated on April 6, 2003 by BE.
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - Representation
  - Fitness and Selection
  - Variation Operators
- Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
Design Space Exploration

Application → Mapping → Architecture

Mapping → Estimation
Embedded System Design

multiobjective optimization

evaluation

allocation bindings

performance / cost vector

architecture

construct architecture

map application

estimate performance

architecture

performance

architecture template

task graph

binding restrictions
Example: System Synthesis

Given:

algorithm mappings architectures

Goal:

schedule mapping architecture

Objectives: cost, latency, power consumption
Evolutionary Algorithms for DSE

EA

1. selection
2. recombination
3. mutation

“chromosome” = encoded allocation + binding

individual

allocation

bindin

decode allocation

decode binding

scheduling

fitness evaluation

fitness

user constraints

design point

(implementation)
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - Representation
  - Fitness and Selection
  - Variation Operators
- Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
- Example
Problem graph $G_P(V_P,E_P)$:

Interpretation:
- $V_P$ consists of functional nodes $V_P^f$ (task, procedure) and communication nodes $V_P^c$.
- $E_P$ represent data dependencies.
Architecture graph $G_A(V_A, E_A)$:

- $V_A$ consists of functional resources $V_A^f$ (RISC, ASIC) and bus resources $V_A^c$. These components are potentially allocatable.
- $E_A$ model directed communication.
**Definition:** A specification graph is a graph \( G_S=(V_S,E_S) \) consisting of a problem graph \( G_P \), an architecture graph \( G_A \), and edges \( E_M \). In particular, \( V_S=V_P \cup V_A \), \( E_S=E_P \cup E_A \cup E_M \).
Basic model - synthesis

Three main tasks of synthesis:

• **Allocation** \(\alpha\) is a subset of \(V_A\).

• **Binding** \(\beta\) is a subset of \(E_M\), i.e., a mapping of functional nodes of \(V_P\) onto resource nodes of \(V_A\).

• **Schedule** \(\tau\) is a function that assigns a number (start time) to each functional node.
**Definition**: Given a specification graph $G_S$, an **implementation** is a triple $(\alpha, \beta, \tau)$, where $\alpha$ is a feasible allocation, $\beta$ is a feasible binding, and $\tau$ is a schedule.
Example
Challenges

- Encoding of (allocation+binding)
  - simple encoding
    - eg. one bit per resource, one variable per binding
    - easy to implement
    - many infeasible partitionings
  - encoding + repair
    - eg. simple encoding and modify such that for each $v_p \in V_P$ there exists at least one $v_a \in V_A$ with $\beta(v_p) = v_a$
    - reduces number of infeasible partitionings
- Generation of the initial population, mutation
- Recombination
Topics

- Multi-Objective Optimization
  - Introduction
  - Multi-Objective Optimization
- Algorithm Design
  - Design Choices
  - Representation
  - Fitness and Selection
  - Variation Operators
- Implementation
- Design Space Exploration
  - System Design
  - Problem Specification
  - Example
behavioral specification of a video codec for video compression
problem graph of the video coder
Exploration Case Study - Solution 1
Exploration Case Study - Solution 2