Design Space Exploration

SS 2012

Jun.-Prof. Dr. Christian Plessl

Custom Computing
University of Paderborn
• motivation for design space exploration

• design space exploration using evolutionary algorithms

• case studies
Motivation

- challenge: many real-world synthesis problems cannot be solved optimally
  - large search space (combinatorial explosion)
  - metrics to be optimized can be estimated only crudely without implementation (e.g. power consumption, performance, chip area, ...)
  - no closed form solution available (e.g. because of non-linear constraints or metrics)
  - no general definition what “optimal” means in the case of multiple, contradicting optimization goals

power consumption?

communication bus usage?

execution time?

code size?
Design Space Exploration

• objective
  – instead of finding one optimal solution …
  – … determine many alternative solutions and their characteristics

• use cases
  – present designer with choices to select from
  – understand design trade-offs
  – create starting points for refined designs

iterative stochastic search process

initialize candidates
  e.g. randomly generated

generate new solution candidates
  chose allocation and binding

evaluate candidates
  compute schedule and other metrics

update set of solutions
  select which solution(s) to pursue further
Evolutionary Algorithms – Basic Principles

1. selection

2. crossover

3. mutation
Evolutionary Algorithms (EA)

minimize $g(x) = x^2$
Evolutionary Algorithms (EA)

- evolutionary algorithms are randomized search heuristics
  - problem-independent (meta heuristics)
  - population based
  - use variation (crossover, mutation) and selection

- application domains
  - when optimization problem is complex and 'diffuse'
    - examples: system synthesis, path planning in robotics
  - multiobjective optimization
    - several conflicting criteria, eg. performance vs. cost vs. power consumption
    - EAs find Pareto fronts (set of Pareto points)
Dominance, Pareto Points

• **definition:** a (design) point \( J_i \) is dominated by point \( J_k \), if \( J_k \) is equal or better than \( J_i \) in all criteria and better in at least one criterion.

\[
J_i < J_k
\]

• **definition:** a (design) point is **Pareto-optimal** or a **Pareto point**, if it is not dominated by any other point.
Multiobjective Optimization (1)

decision space

\((x_1, x_2, \ldots, x_n)\)

objective space

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

minimize \(f\)

difficulties:

1. large search space
2. multiple optima

Pareto optimal = not dominated

dominated
Multiobjective Optimization (2)

• classic single-objective methods
  – e.g. simulated annealing, ILP, hierarchical clustering, ...
  – decision making before optimization
    ▪ weighted cost function
    ▪ multi-stage optimization
      – e.g. hierarchical clustering with different closeness functions
  – decision making after optimization
    ▪ multiple optimization runs with varying weights

• population-based methods
  – evolutionary algorithms
  – decision making after optimization
    ▪ the goal is to explore the design space
Weighted Cost Function

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

transformation

parameters

single objective

\(y\)

example: weighting approach

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

\(y = w_1 y_1 + \ldots + w_k y_k\)

maximization problem
EAs for Multiobjective Optimization

EA operations
1. selection
2. recombination
3. mutation
Fitness Assessment by Pareto Ranking

- fitness function:

\[ F' (J) = \sum_{i=1..N, J \neq J_i} \begin{cases} 
1 : & J_i \prec J \\
0 : & \text{else} 
\end{cases} \]

execution time

\[ \begin{align*}
F'(1) &= 3 \\
F'(2) &= 1 \\
F'(3) &= 1 \\
F'(4) &= 2 \\
F'(5) &= 1 \\
F'(6) &= 0
\end{align*} \]
Fitness Assessment by Non-dominated Sorting

1. start with set of all solutions J
2. determine set of non-dominated solutions (A₀) in J, assign fitness value φ₀ to these solutions
3. remove A₀ from J (J₁ = J \ A₀), determine set of non-dominated solutions (A₁) in A₀, assign fitness value φ₁ to these solutions
4. repeat until Jᵢ is empty
Example: Strength Pareto EA (1)

1. save non-dominated solutions (elitism)
2. reduce non-dominated set by means of clustering
3. assign fitness values (Pareto-based)
4. perform binary tournament selection
5. recombination
   Mutation
Example: Strength Pareto EA (2)

clustering: reduce non-dominated set but do not destroy characteristics
Example: Strength Pareto EA (3)

- “lighter better than darker” → guidance towards Pareto-optimal set
- “few better than many” → maintenance of diversity

fitness assignment scheme:
1. non-dominated solutions:
   \[ \text{fitness} = \# \text{dominated solutions} \]
2. dominated solutions:
   \[ \text{fitness} = \# \text{non-Pareto solutions} + \sum \text{fitness of dominators} \]

Example: Strength Pareto EA (3)
Design Space Exploration with EAs

“chromosome” = encoded allocation + binding

1 selection
2 recombination
3 mutation

Individual

allocation
binding

decode allocation
decode binding

scheduling

fitness evaluation

fitness

user constraints

design point (implementation)
Challenges

• encoding of (allocation+binding)
  – simple encoding
    ▪ e.g. one bit per resource, one variable per binding
    ▪ easy to implement
    ▪ many infeasible partitionings
  – encoding + repair
    ▪ e.g. 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
  – find a good starting point (seed) to accelerate convergence

• mutation and recombination
  – operators must be defined that the whole search space can be reached
behavioral specification of a video codec for video compression
Case Study - Video Coder (2)

problem graph of the video coder
Case Study - Video Coder (3)

- frame memory
- dual ported frame memory
- block matching module
- input module
- subtract/add module
- DCT/IDCT module
- Huffman encoder

h261 architecture template
EA Design Space Exploration Tool
Case Study - Solution 1

INM

OUTM

FM

RISC2

SBS
Case Study - Code Synthesis (1)

- synchronous data flow graph (SDF) formalism
  - programming model for streaming signal processing applications
  - allows expressing coarse-grained parallelism
  - suitable as specification for systems with parallel execution

- application is defined as a network of nodes and arcs
  - nodes (actors)
    - can execute arbitrary functions
    - consume and produce tokens
    - number of consumed/produced tokens is shown as a label
  - arcs (communication channels)
    - actor communicate via FIFO buffers

![Diagram of an actor] example for an actor, reads 3 tokens on input arc and produces 4 tokens on output
Case Study - Code Synthesis (2)

synchronous data flow graph

CD 44.1kHz
sample rate converter

DAT 48 kHz

software implementation

decisions

1 schedule
ABABABCCABABA...

2 code generation model

inlining

CALL (A)
CALL (B)
CALL (C)

subroutines

CODE (A)
CODE (B)
CODE (C)

looping

FOR 1 TO 2
CODE (A)
CALL (B)
CODE (C)
CODE (A)
Case Study - Code Synthesis (3)

program memory

data memory

looping

execution time

subroutines

may increase

saves

save

increase

trade-offs
trade-off surface for TI TMS320C40 digital signal processor
Case Study - Code Synthesis (5)

TI TMS320C40 DSP

Motorola DSP56k

ADSP 2106x DSP
Changes

- 2012-06-15 (v1.1.0)
  - updated for SS2012
- 2011-06-14 (v1.0.0)
  - initial version