Stochastic Search for Signal Processing Algorithm Optimization

Bryan Singer and Manuela Veloso
Computer Science Department
Carnegie Mellon University, Pittsburgh

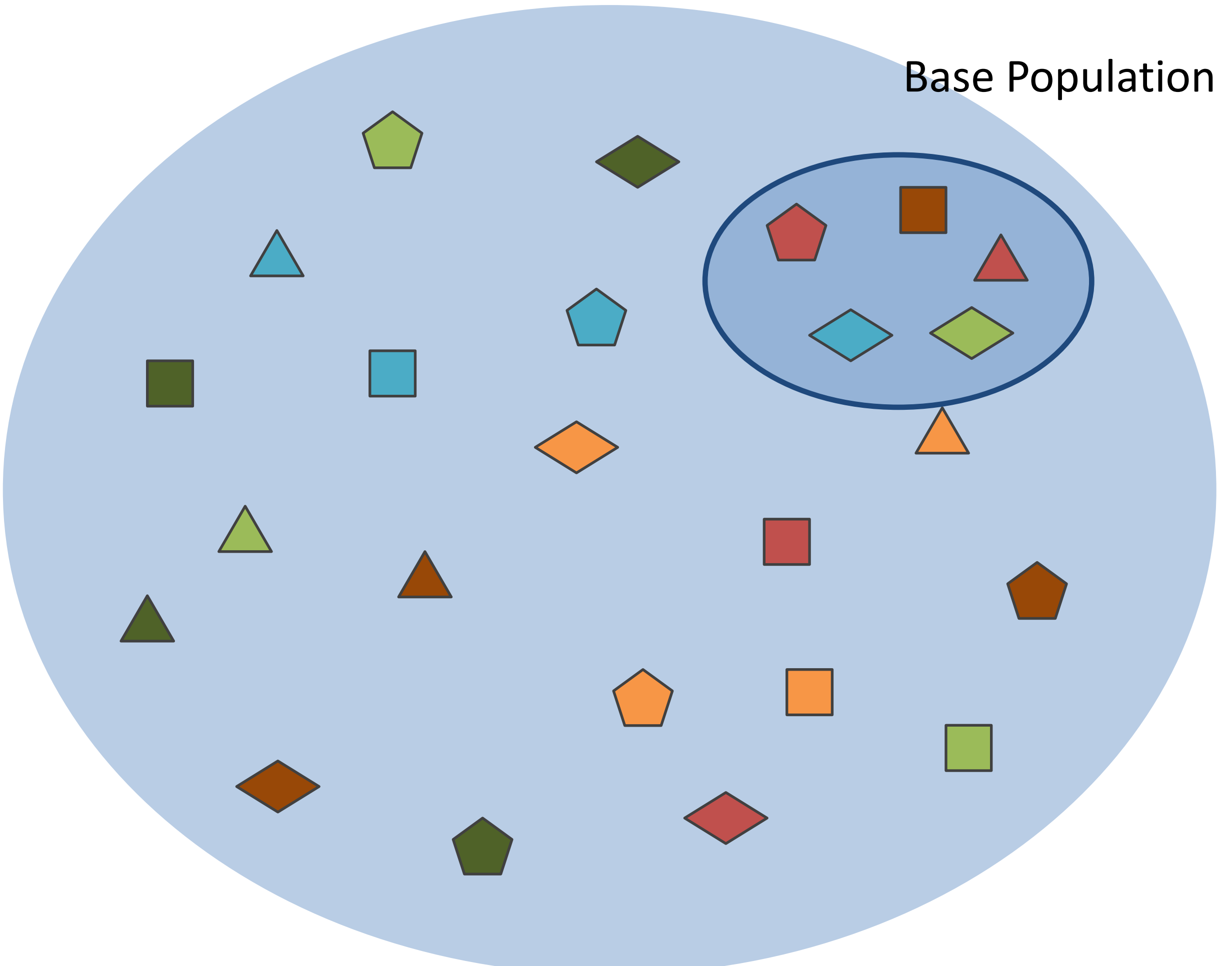
Liat Ben-Haim, November 16th 2011

Overview

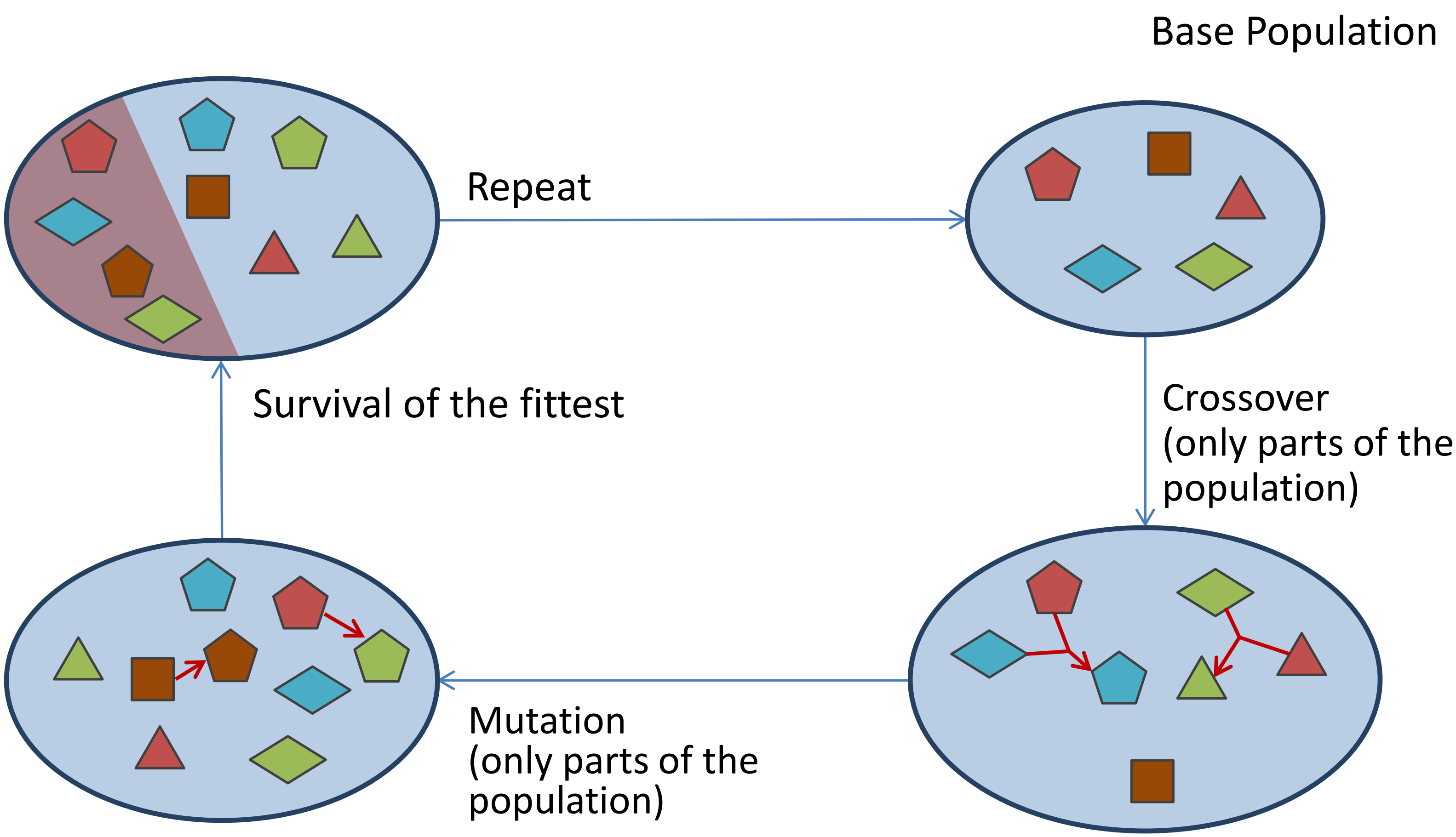
- **Genetic algorithm**
- **Signal processing**
  - Walsh-Hadamard Transform
- **STEER: Split Tree Evolution for Efficient Runtimes**
  - For Walsh-Hadamard Transform
  - Results Walsh-Hadamard Transform
  - For Arbitrary Transform
  - Results Arbitrary Transform
- **Conclusion: Strengths and Weaknesses**

Genetic Algorithm

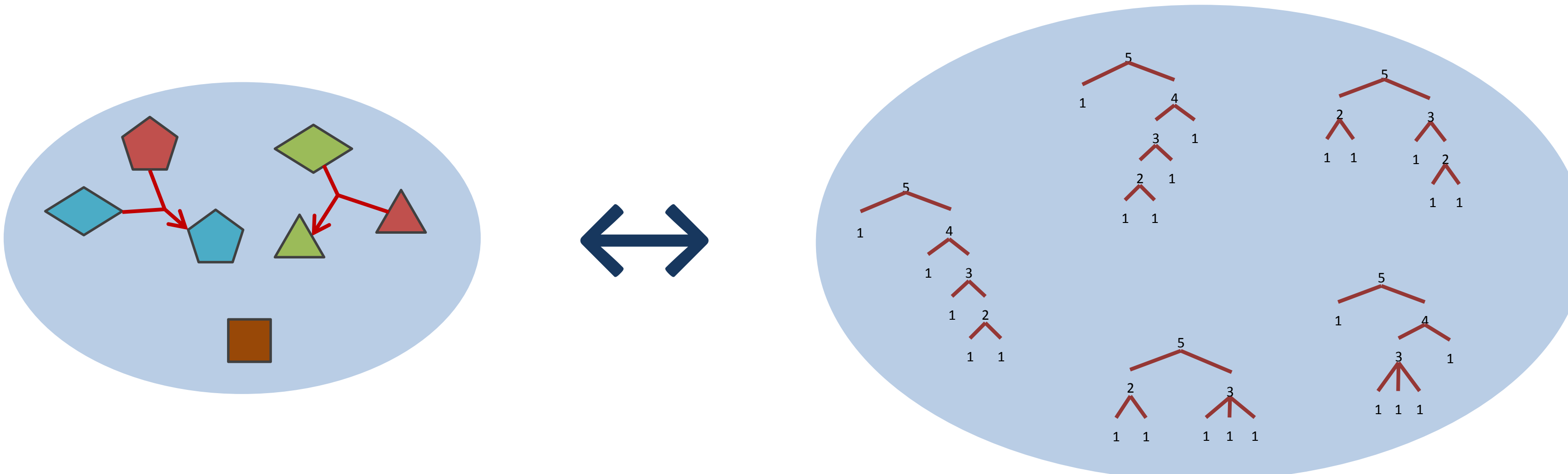
Complete
Search space
(very large)



Genetic Algorithm



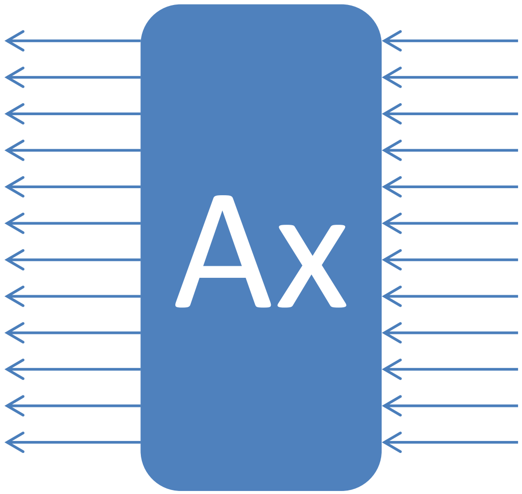
General ↔ STEER



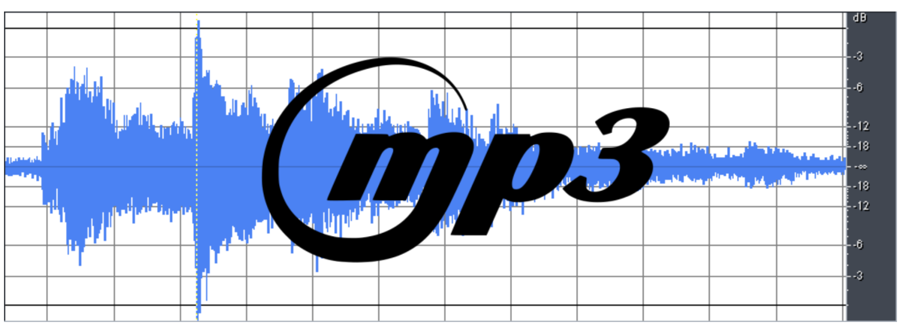
- **Population consists of algorithms**
  - Algorithms modeled as trees
- **Fitness is measured in runtime on given device**

- **Genetic algorithm**
- **Signal processing**
  - Walsh-Hadamard Transform
- **STEER: Split Tree Evolution for Efficient Runtimes**
  - For Walsh-Hadamard Transform
  - Results Walsh-Hadamard Transform
  - For Arbitrary Transform
  - Results Arbitrary Transform
- **Conclusion: Strengths and Weaknesses**

Signal Processing: y = Ax



Pictures (JPEG)



Audio (MP3)

http://commons.wikimedia.org/wiki/File:ALC_orig.png
http://de.wikipedia.org/w/index.php?title=Datei:Mp3.svg&filetimestamp=20091118142210
http://de.wikipedia.org/w/index.php?title=Datei:Phalaenopsis_JPEG.png&filetimestamp=20110430130839

Walsh-Hadamard Transform (WHT)

$$y = WHT(2^n) \cdot x \qquad WHT(2^n) = \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}_{\text{n factors}}$$

n = 1:

$$y = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot x$$

n = 2:

$$y = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot x = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{array}\right] \cdot x$$

n = 3:

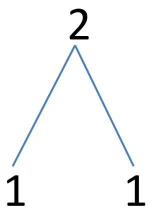
$$y = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot x = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \cdot x$$

Fast Walsh-Hadamard Transform

n = 2:

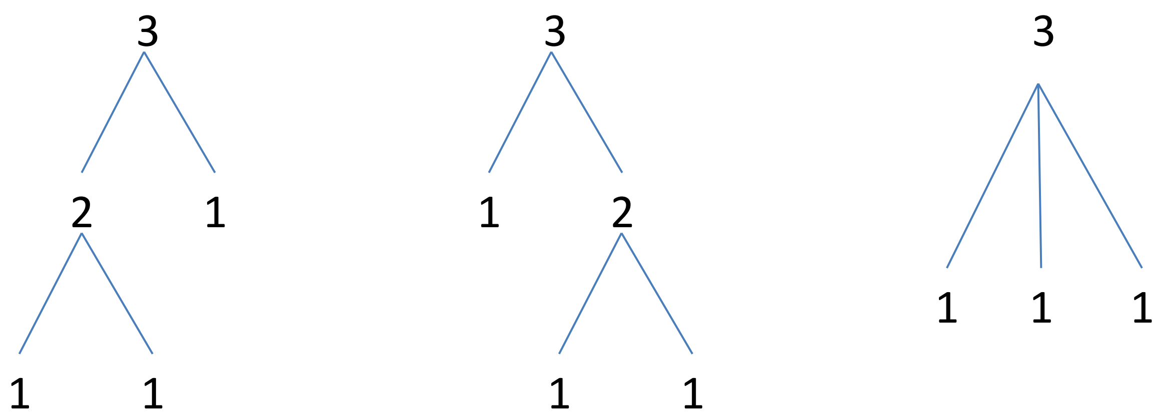
$$y = \overbrace{WHT(2^2) \cdot x}^{\text{Opcount: 12}} = [I_{2^1} \otimes WHT(2^1)] \cdot [WHT(2^1) \otimes I_{2^1}] \cdot x =$$

$$= \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \cdot x}_{\text{Opcount: 4}}}_{\text{Opcount: 4+4 = 8}}$$



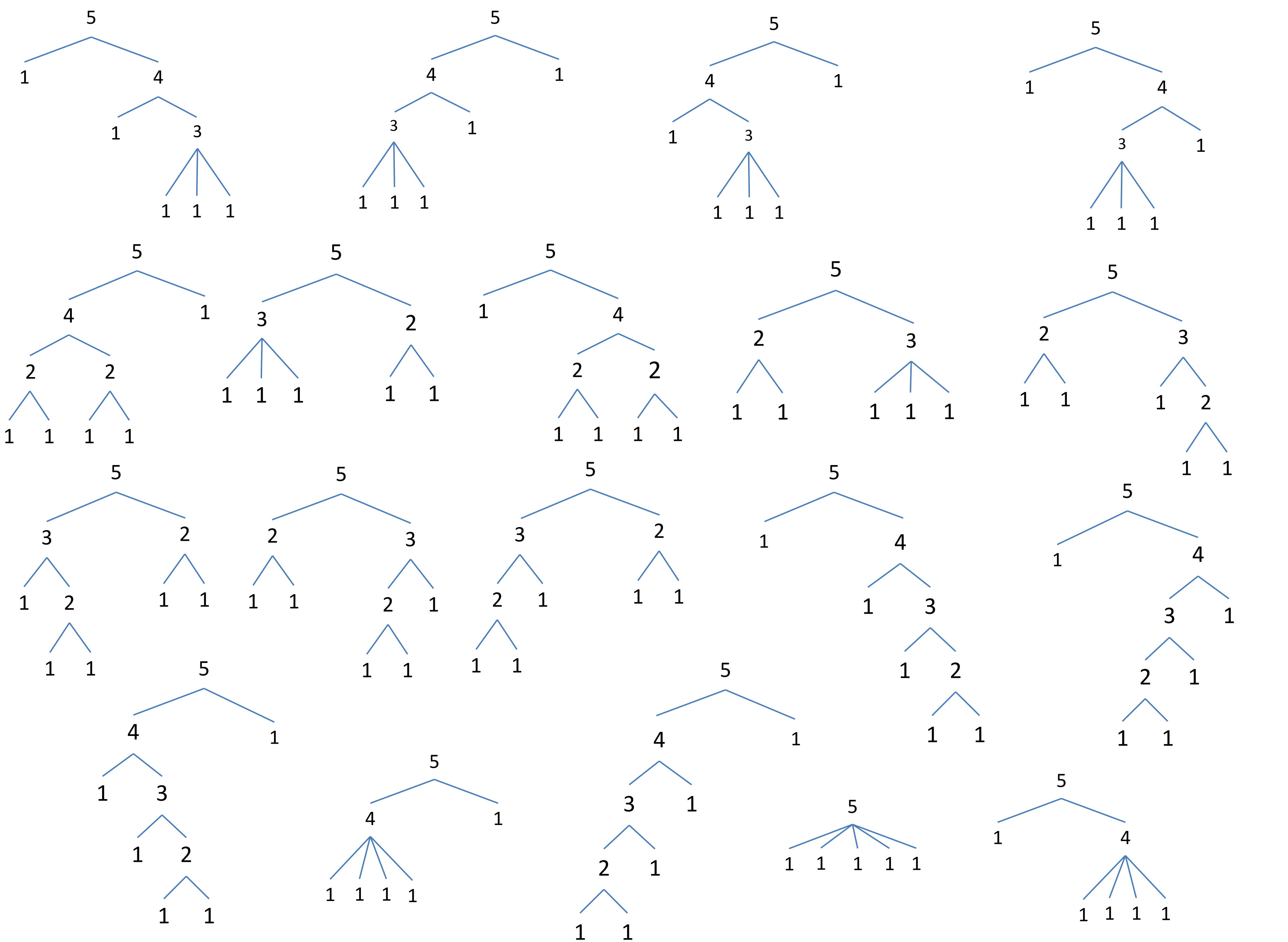
Fast Walsh-Hadamard Transform

n = 3:



Opcount: 24 24 24

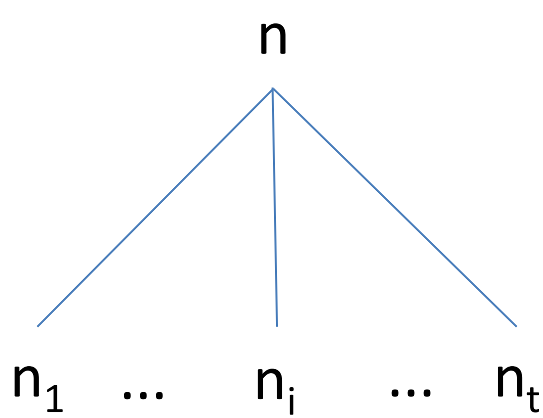
Opcount WHT(2^3): 56



General Break Down Rule

$$WHT(2^n) = \prod_{i=1}^{t} (I_{2^{n_1+\cdots+n_{i-1}}} \otimes WHT(2^{n_i}) \otimes I_{2^{n_{i+1}+\cdots+n_t}})$$

$n = n_1 + \cdots + n_t$ (n_j: positive integers)



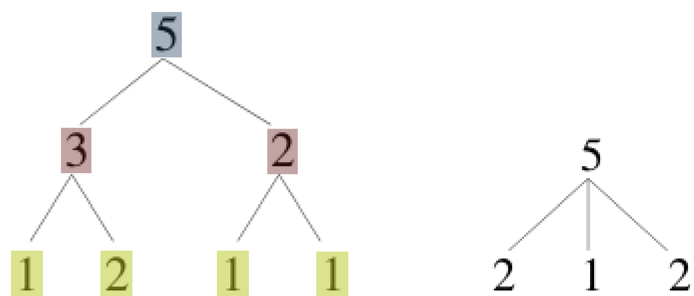
WHT Example

$$WHT(2^n) = \prod_{i=1}^{t} (I_{2^{n_1+\cdots+n_{i-1}}} \otimes WHT(2^{n_i}) \otimes I_{2^{n_{i+1}+\cdots+n_t}})$$

$$I_{2^0} \otimes A = A$$

$$A \otimes I_{2^0} = A$$

$$\begin{aligned} & WHT(2^5) \\ &= [WHT(2^3) \otimes I_{2^2}][I_{2^3} \otimes WHT(2^2)] \\ &= [\{(WHT(2^1) \otimes I_{2^2})(I_{2^1} \otimes WHT(2^2))\} \otimes I_{2^2}] \\ &\quad [I_{2^3} \otimes \{(WHT(2^1) \otimes I_{2^1})(I_{2^1} \otimes WHT(2^1))\}] \end{aligned}$$



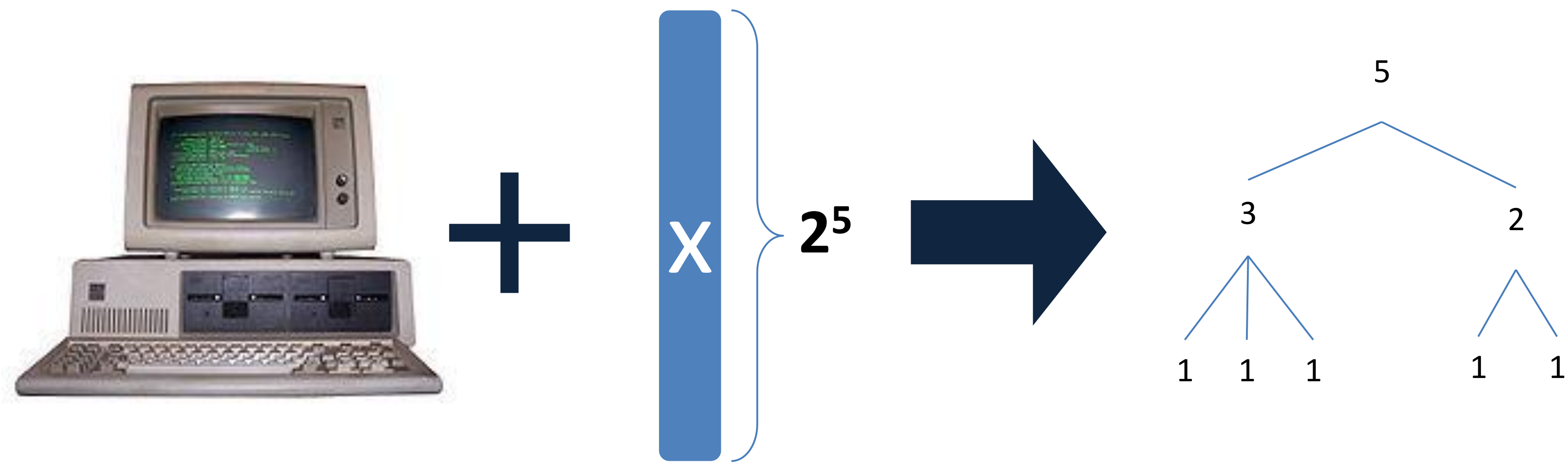
(a) (b)

Stochastic Search for Signal Processing Algorithm Optimization

- Genetic algorithm
- Signal processing
  - Walsh-Hadamard Transform
- **STEER: Split Tree Evolution for Efficient Runtimes**
  - For Walsh-Hadamard Transform
  - Results Walsh-Hadamard Transform
  - For Arbitrary Transform
  - Results Arbitrary Transform
- Conclusion: Strengths and Weaknesses

Goal

Given input signal x of size 2^n and specific device, find fastest program for this signal size and device



http://de.wikipedia.org/w/index.php?title=Datei:IBM_PC_5150.jpg&filetimestamp=20060811115558

Search Techniques

- **Exhaustive Search**
  - Does not scale
- **Dynamic Programming**
  - Assumption: «combination of optimal solutions for subproblems leads to optimal solution»
  - K-Best DP
  - Search space restriction: Binary trees
  - Bad choices lead to inferior solution
- **Split Tree Evolution for Efficient Runtimes: STEER**

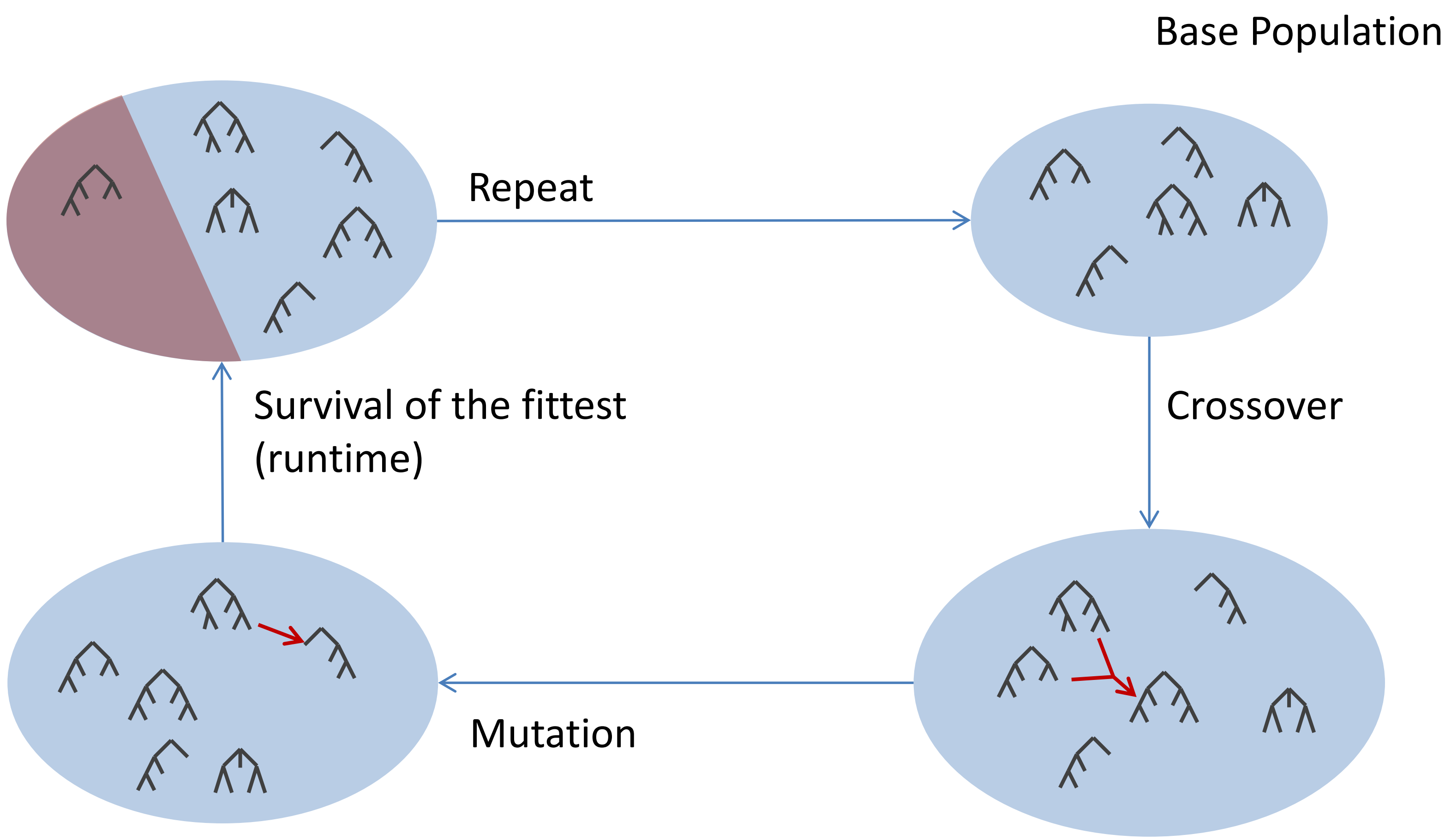
STEER

- **Split Tree Evolution for Efficient Runtimes**
  - Genetic Algorithm
  - Part of the SPIRAL research group
    - «Can we teach computers to write fast libraries?»
  - Adapted to the system used by the research group

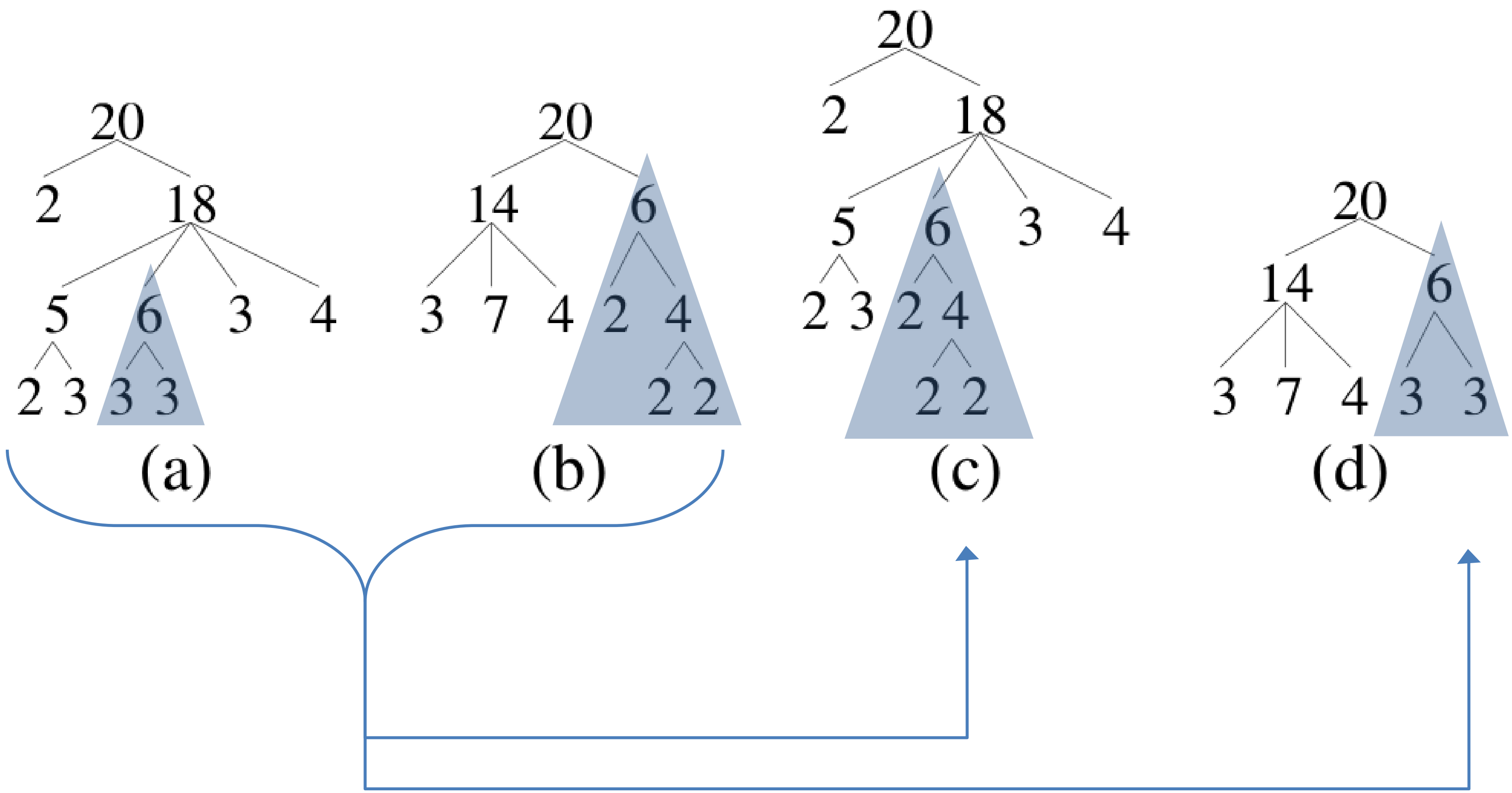
SPIRAL
Software/Hardware Generation for DSP Algorithms

http://www.spiral.net

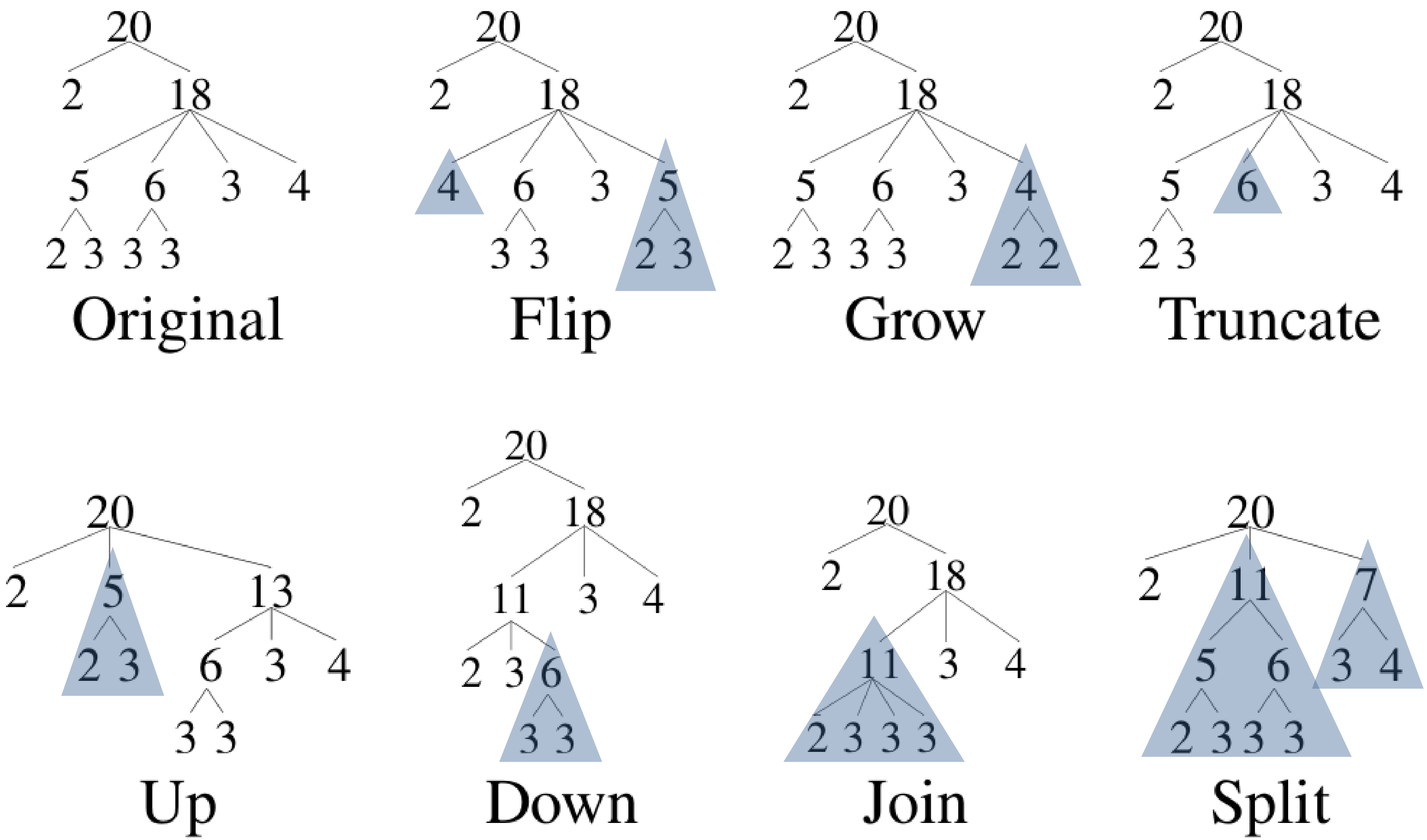
STEER: Genetic Algorithm



WHT: Crossover

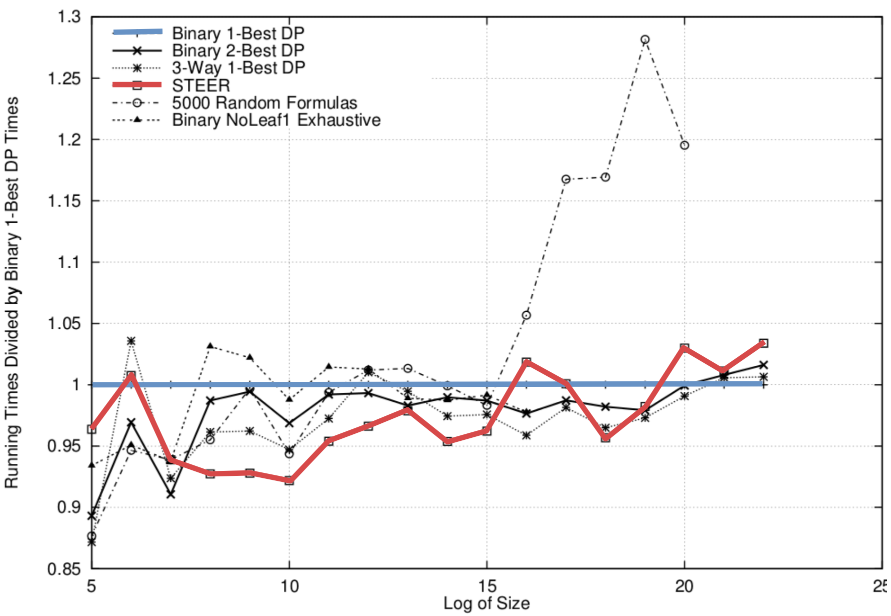


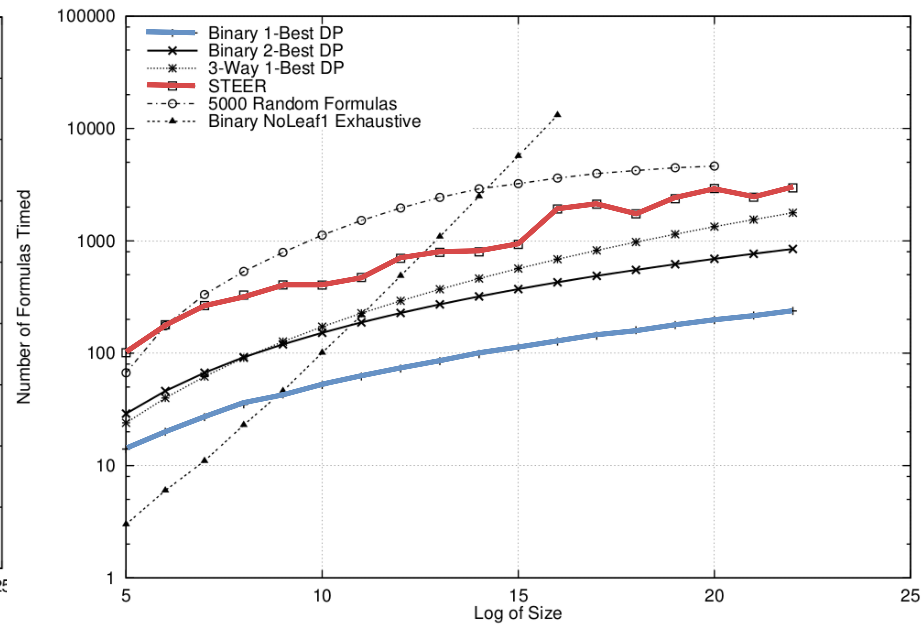
WHT: Mutation



Results: WHT

- **Pentium III 450 MHz (32-Bit Architecture)**
- **Linux 2.2.5-15**
- **WHT Package from Johnson and Püschel**
  - «In search of the optimal Walsh-Hadamard transform», 2000
  - Leaves of sizes 2^1 to 2^8
  - Unrolled straight-line code



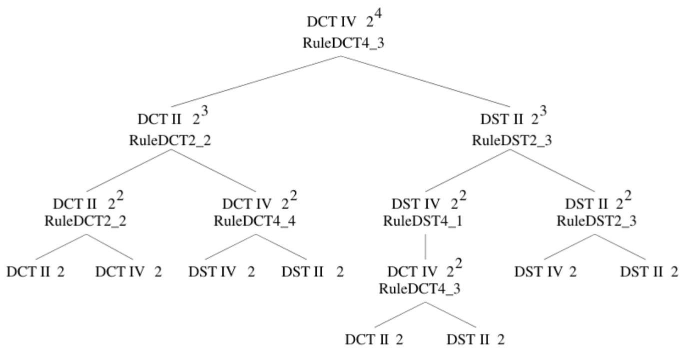


Stochastic Search for Signal Processing Algorithm Optimization

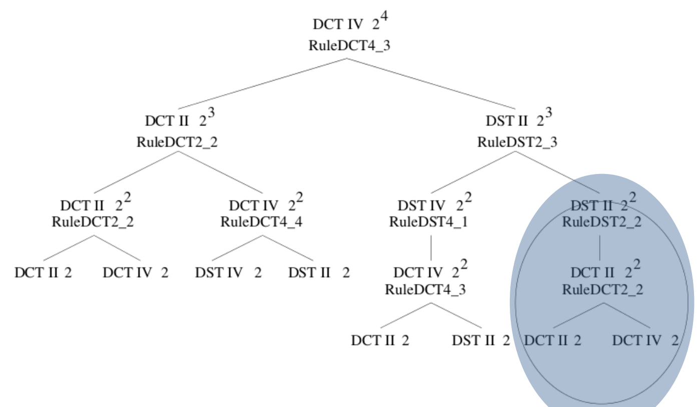
Arbitrary Signal Transform

- **Random tree generation:**
  - Randomly choose an applicable break down rule
  - Apply to node, to generate a random set of children
  - Recursively apply to each child
- **Crossover**
  - Equivalent nodes: same size and transform
- **New mutations**

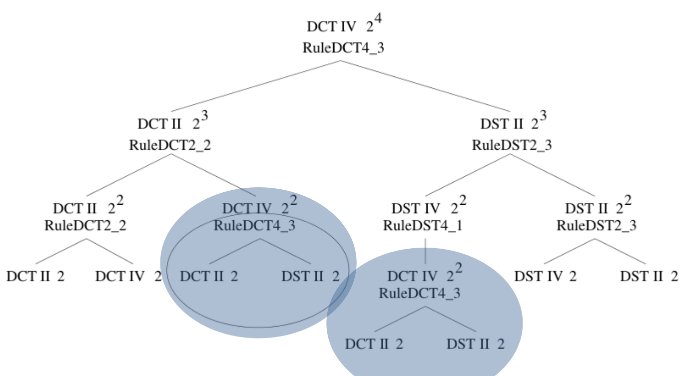
Arbitrary Transform: Mutation



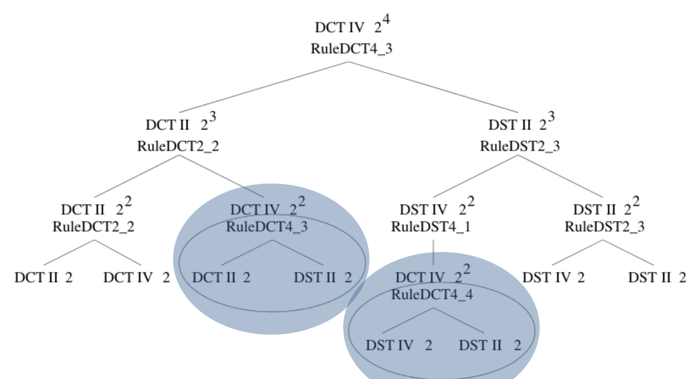
Original



Regrow

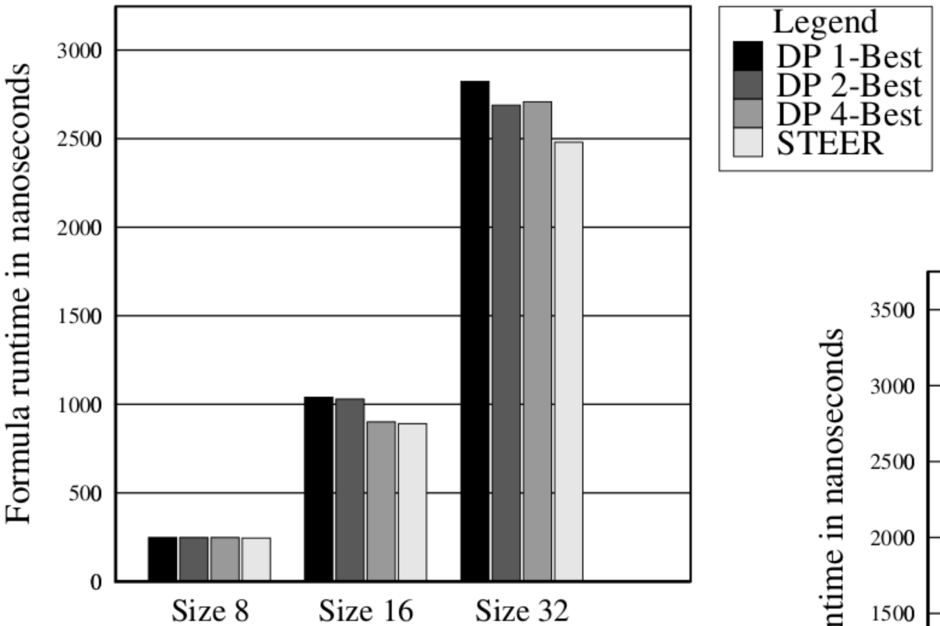


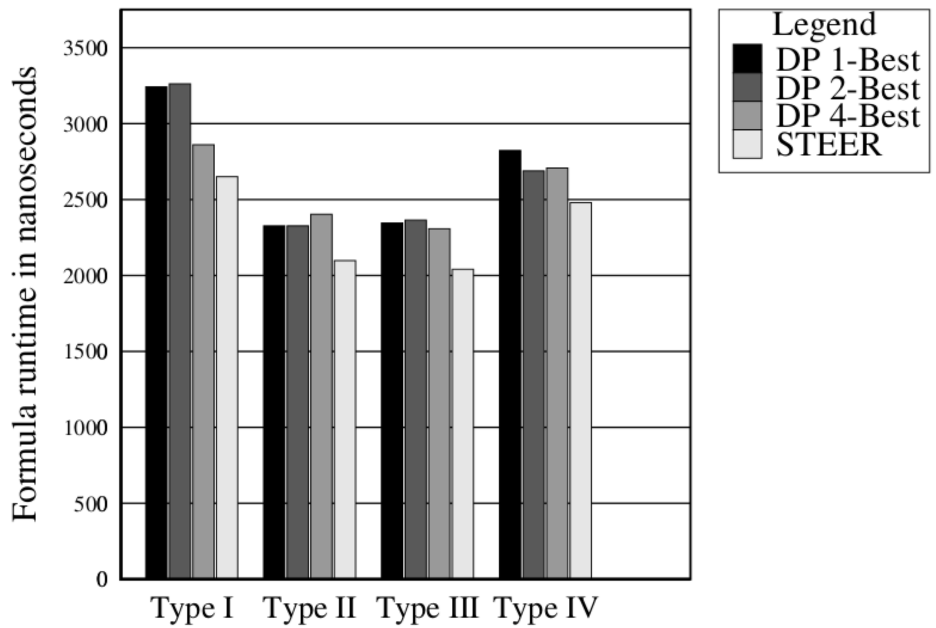
Copy

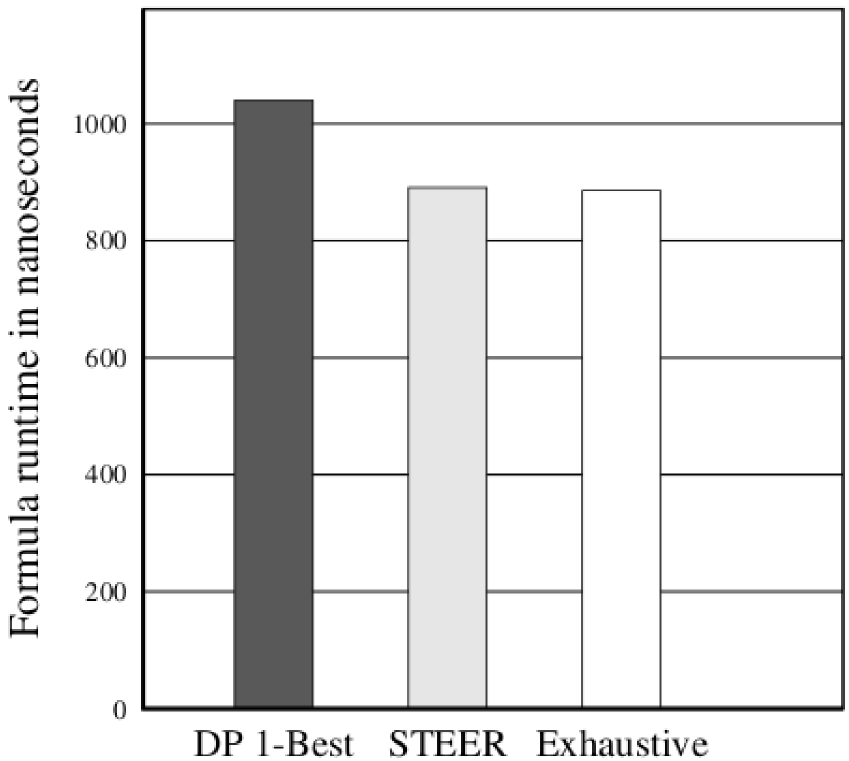


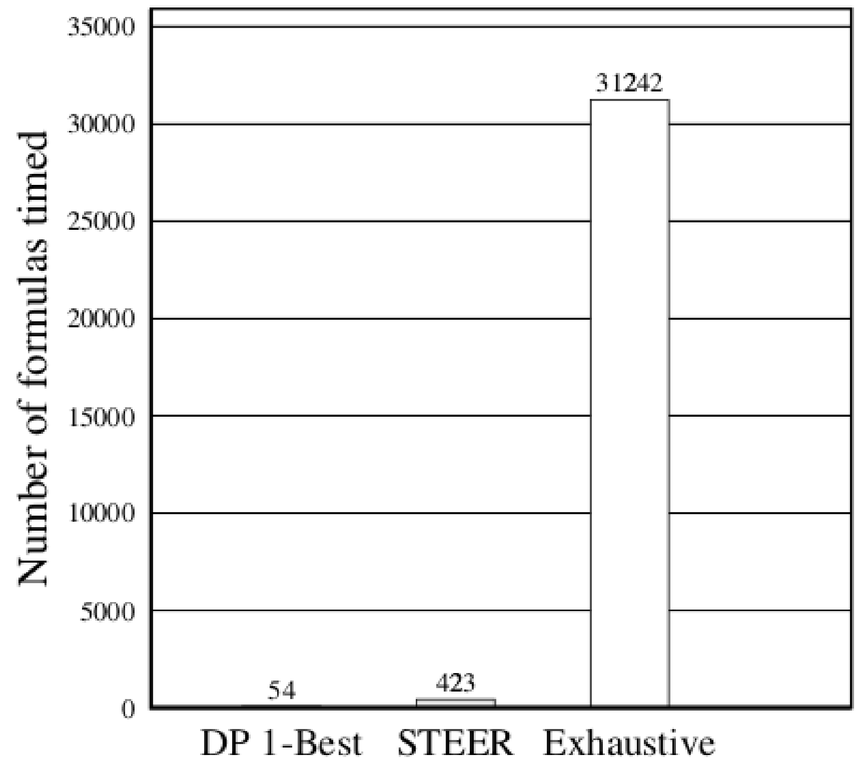
Swap

Results: Arbitrary Transform






Formula runtime in nanoseconds
0
200
400
600
800
1000
DP 1-Best
STEER
Exhaustive


Number of formulas timed
0
5000
10000
15000
20000
25000
30000
35000
54
423
31242
DP 1-Best
STEER
Exhaustive

Strengths & Weaknesses

- STEER can be used for arbitrary transforms using SPIRAL
- Finds good if not necessarily optimal solutions
- Found solutions are generally better than DP
- Times significantly less formulas than exhaustive search

- Missing parameters of the evolutionary algorithm
- No guarantee for a «good» solution
- Times more formulas than DP
- No mention of how long STEER usually runs
- How much better than Ax?

Questions?