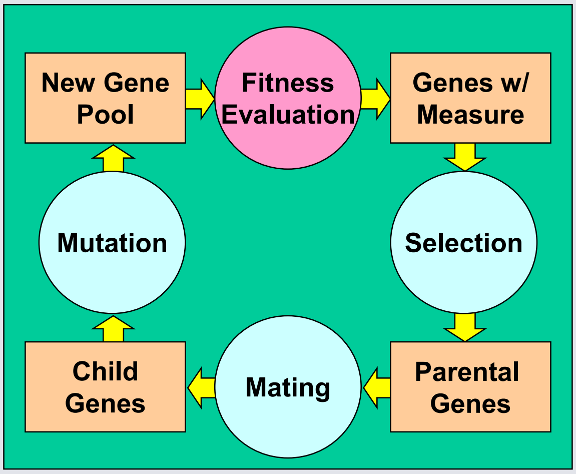
Evolutionary Programming using Ptolemy II

Greg Rohling
Georgia Tech Research Institute
Georgia Institute of Technology
Greg.Rohling@gtri.gatech.edu

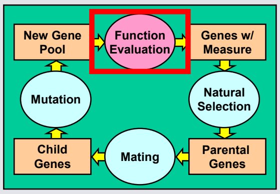
5th Biennial Ptolemy Miniconference
Berkeley, CA, May 9, 2003

Evolutionary Algorithms (EA)

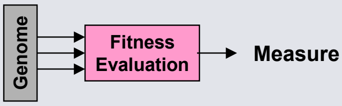
- Evolutionary Algorithms
  - Inspired by Holland 1975
  - Mimic the processes of plant and animal evolution
  - Find a maximum of a complex function.



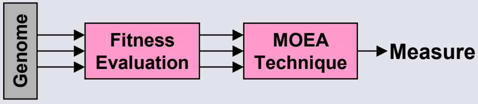
Multiple Objective Evolutionary Algorithms (MOEA)



- Any multi-input, single output function

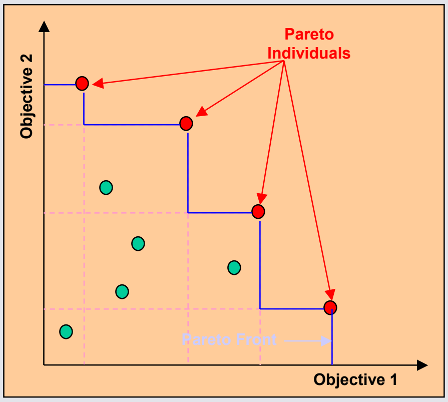


- Fun problems have multiple objectives
  - Pd, False Alarms
- Thus Multiple Objective Evolutionary Algorithms



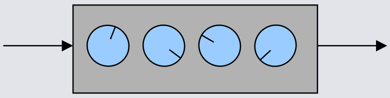
MOEA – Pareto Optimality

- Pareto Optimal or Non-dominated
  - Not out-performed in every dimension by any single individual
  - Pareto Front
- Dominated or inferior
  - Outperformed by some other individual in *EVERY* objective

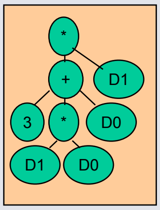


Evolutionary Programming (EP)

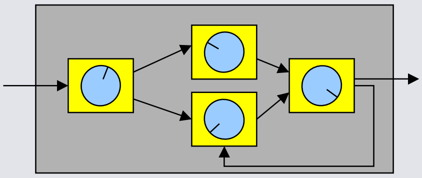
- Evolutionary Algorithms EA = Tuning parameters of an existing system



- EP = Deriving new systems by allowing tuning of parameters for components, AND allowing modification of system topology

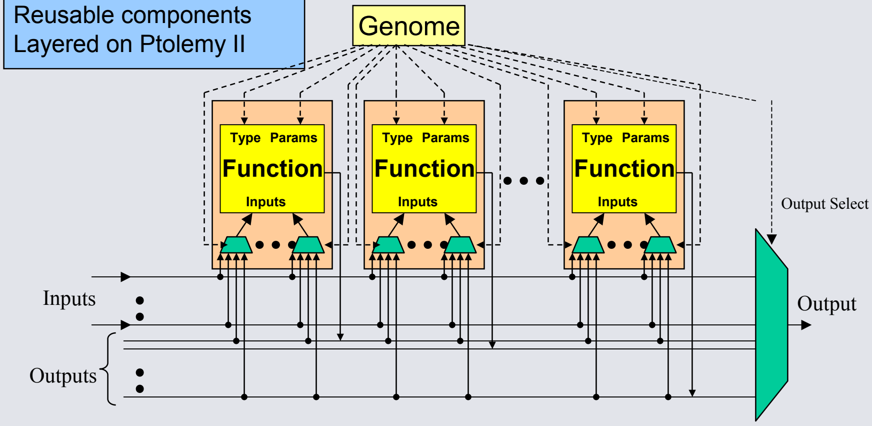


Koza 1996

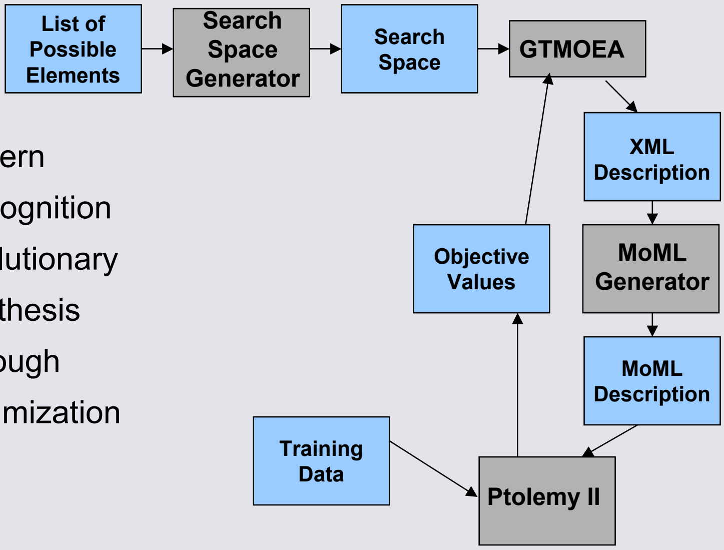


EP: Block Diagram Approach

Flexible topology
Feedback possible
Domain Specific Functions
Reusable components
Layered on Ptolemy II



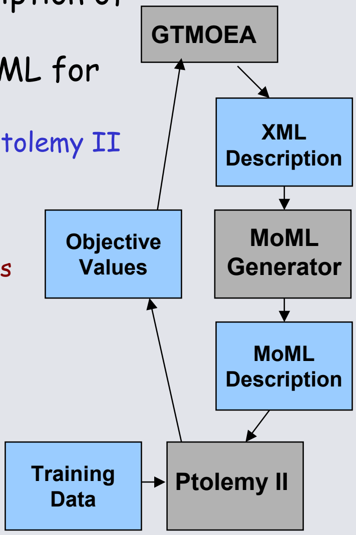
PRESTO



Pattern
Recognition
Evolutionary
Synthesis
Through
Optimization

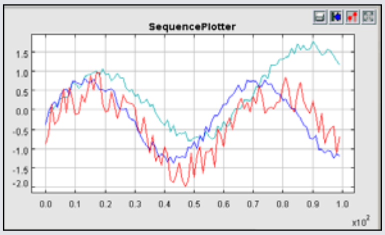
PRESTO: MoML Generator

- GTMOEA produces XML description of location in search space
- MoML Generator produces MoML for Ptolemy II evaluation.
  - Reads Meta data description of Ptolemy II elements
    - # of Ports
    - Data types for ports
    - Data type relations between ports
    - Support for feedback
    - # of input/output required
    - Attributes Specified
    - MoML description
  - Error Correction
    - Feedback
    - Data types
    - Unconnected ports



PRESTO Example

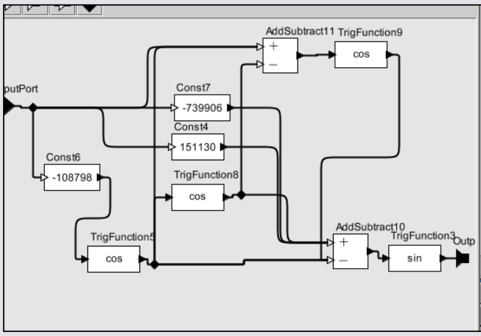
- Desire: Build box that takes input of a ramp function and best meets 3 objectives
- Objective Space
  - 3 sets of training data
    - Noisy sinusoids
  - Minimize least square error
- Search Space
  - AddSubtract
  - Scaler
  - Constant
  - Multiplier
  - Sin/Cos

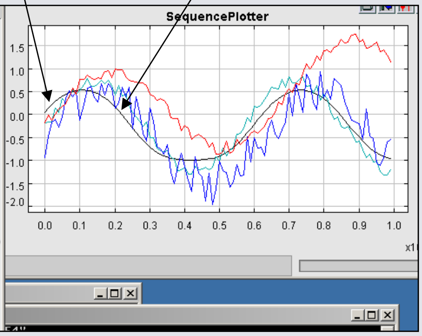


PRESTO Example

- Individual 6695, Good at Objective 2

Ptolemy II Description of System





PRESTO Real World Effort

- Creating “Sub-Algorithms” for AAR44 Missile Warning Receiver Operational Flight Program (OFP)
- Wrapped C++ (Really just C) OFP with JNI to allow calls from within Ptolemy II
- Using Discrete Event Domain to force Wrapped components to be called in sequential order
- Targeting 3 areas of the OFP

PRESTO Real World Effort

- Training/Evaluation Data
  - Sensor and Navigation data
  - 40 hrs of False Positive Data collected from flights
  - 10s of Live fire missile shots
  - 1000s of Simulated missile shots
- Objectives
  - Maximize
    - Probability of detection
    - Negative False Positive Count
    - Time To Intercept Minimum
    - Time To Intercept Average

PRESTO Advantages

- Tries many new ideas
- No preconceived notions
- Does not get discouraged with failure
- Works 24 hours a day 7 days a week
- Scalable
- Resulting Evolutionary Program can be graphically examined to understand algorithm evolved.

PRESTO Disadvantages

- MOEP - Requires evaluation of many (millions) of individuals
- Ptolemy II requires
  - 3 seconds to startup
  - Ptolemy Simulation running over 400 times slower than the C++-only implementation

Ptolemy II Suggestions

- Ptolemy II does a good job of error detection. How about adding default error correction?

Questions?