Experiments on the Automatic Evolution of Protocols using Genetic Programming

Lidia Yamamoto and Christian Tschudin
Computer Networks Group, Computer Science Department
University of Basel, Switzerland
http://cn.cs.unibas.ch/

2nd IFIP International Workshop on
Autonomic Communication (WAC 2005)
Athens, Greece, October 3-5, 2005

Outline

- AC Self-management
- Adaptation and evolution
- Evolving autonomic network software
- Framework for protocol evolution
- Experiments
- Conclusions, next steps

Context

- BIONETS: Biologically-inspired Networks and Services
  - Situated and Autonomic Communication FET Initiative
  - 4-year project starting January 2006
  - Central question: How to make protocols and services evolve automatically during usage: runtime evolution
    - 1st step: Parameter evolution: analogy: genetic algorithms
    - 2nd step: Code evolution: analogy: genetic programming
- This talk:
  - Code evolution experiments using genetic programming

Introduction

- Autonomic Communication (AC)
  - Network elements conspire to do what we want
  - Self-*, including self-management
- **Fully self-managing networks** networks must take complete care of themselves, including **software maintenance**
  - Detect and correct software failures
  - Optimize software for specific context
  - Constantly keep on target and improve itself, without direct human intervention: autonomic!
- Thus: full AC requires **automated software evolution**
  - Otherwise humans must intervene to modify software

Adaptation and Evolution

Short-term adaptation

Long-term adaptation: Evolution

Change in behavior

Change in functionality

Evolving Autonomic Network Software

- Automated **evolution**
  - Functional scaling: beyond self-adaptive software
- Automatically generate new, better replacement code
  - ⇨ **Self-modifying code**
- During operation: **runtime evolution**
  - Resilience and survivability
    - Potentially hostile operational environment
    - Heterogeneous networks and users, competing interests
    - Code errors, malicious code, non-trusted parties, …
  - Fully cooperative learning scheme not realistic
- A possible path: **distributed on-line genetic programming**

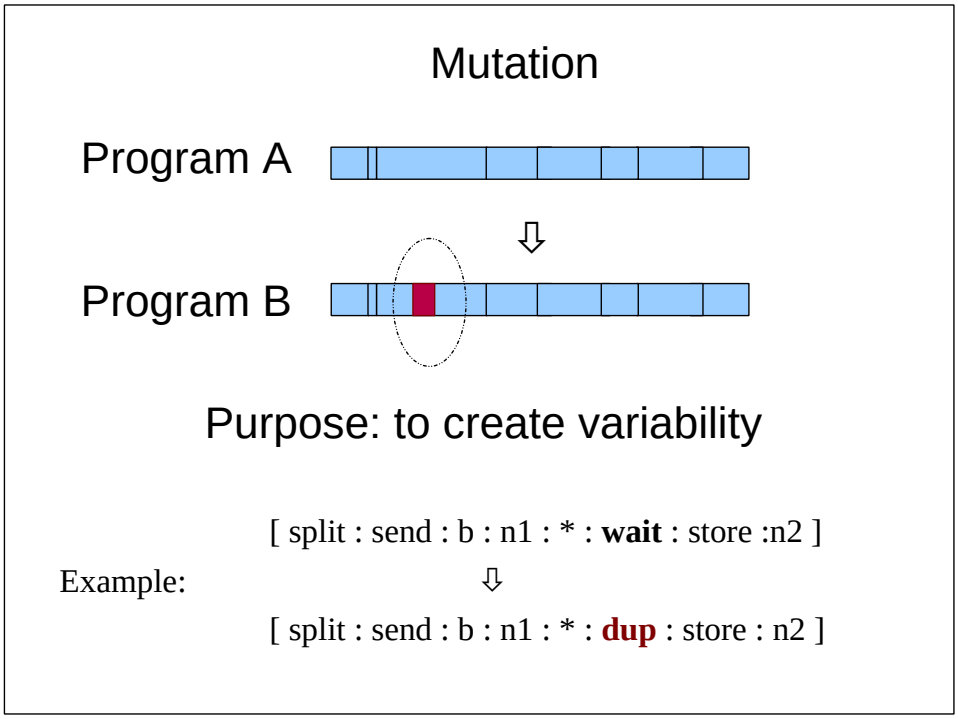
Evolving Autonomic Network Software

- Genetic Programming
  - Machine learning method for synthesizing programs
  - Agnostic program transformations: crossover, mutation
  - Natural selection: survival of the fittest
  - In general off-line: optimal solution as output
- **Distributed on-line genetic programming**
  - Program transformations rely on code mobility
  - Non-disruptive execution of synthesized programs
  - Competitive/hostile environment
    - Natural selection pressure ⇨ Survivability
    - Redundancy ⇨ Resilience

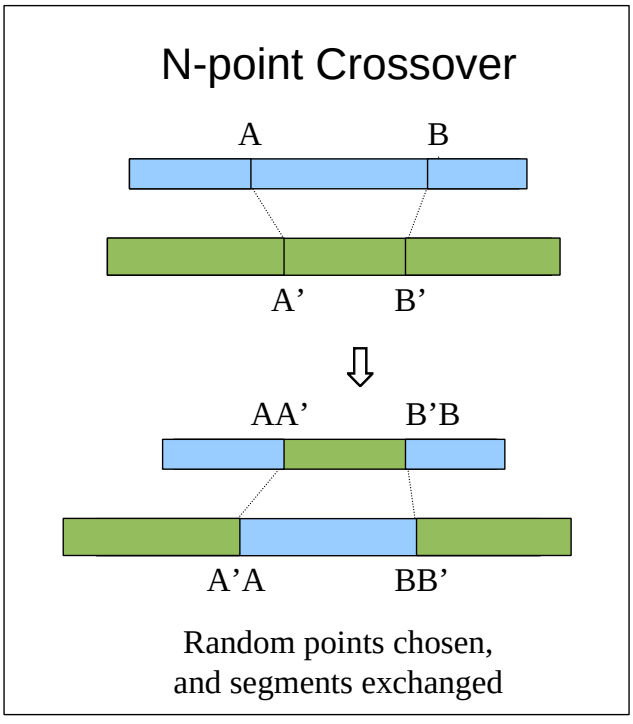
Framework for Protocol Evolution

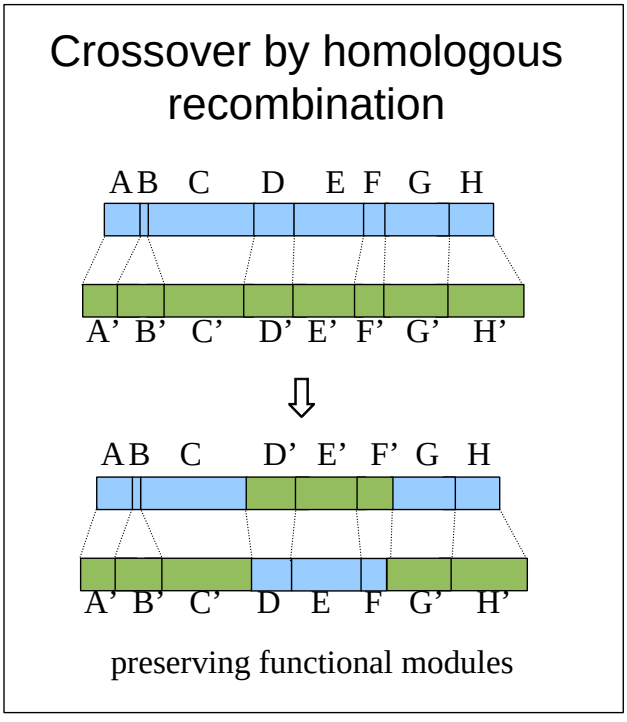
- Start with working protocol implementations
- Evolution via genetic programming
- **Fraglets** chemical programming model
  - Easy GP: any code fragment (fraglet) is a valid program
  - Parallelism ⇨ redundancy ⇨ resilience
    - Resilience to code loss: initial results in WAC 2004
- Still off-line (simulations), but assumptions for on-line
  - Small population
  - Limited number of generations
  - Resilience: minimize service disruptions due to malfunctioning code

Genetic Operators

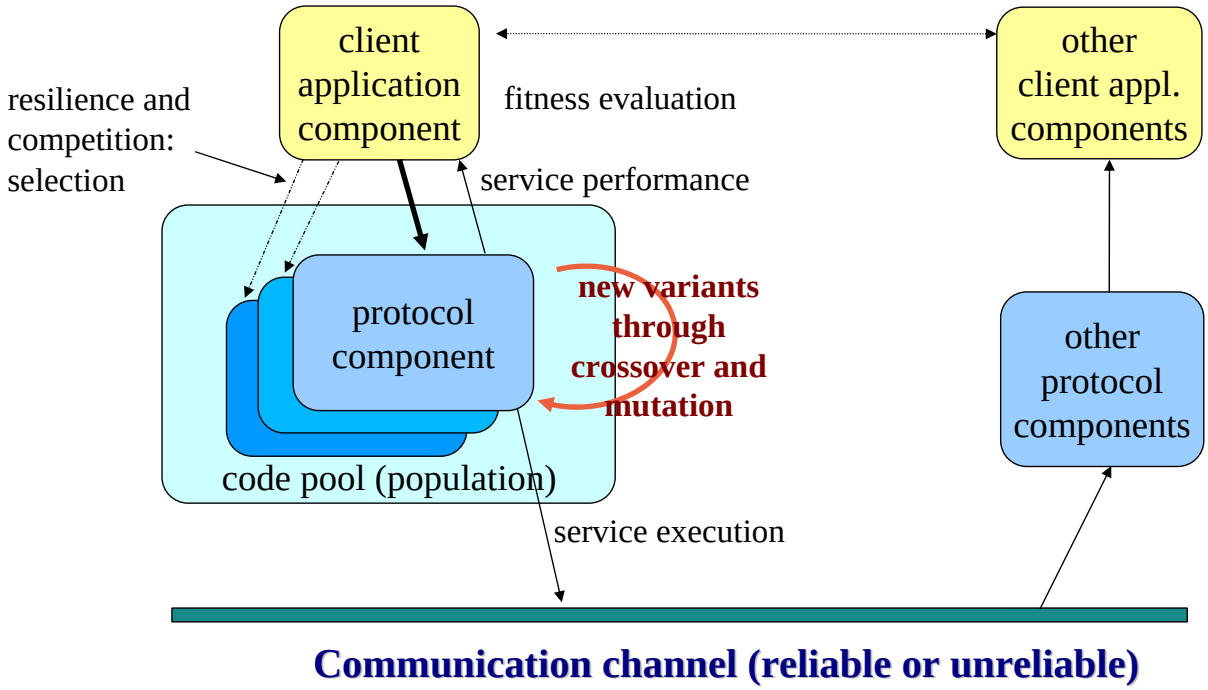


Genetic Operators





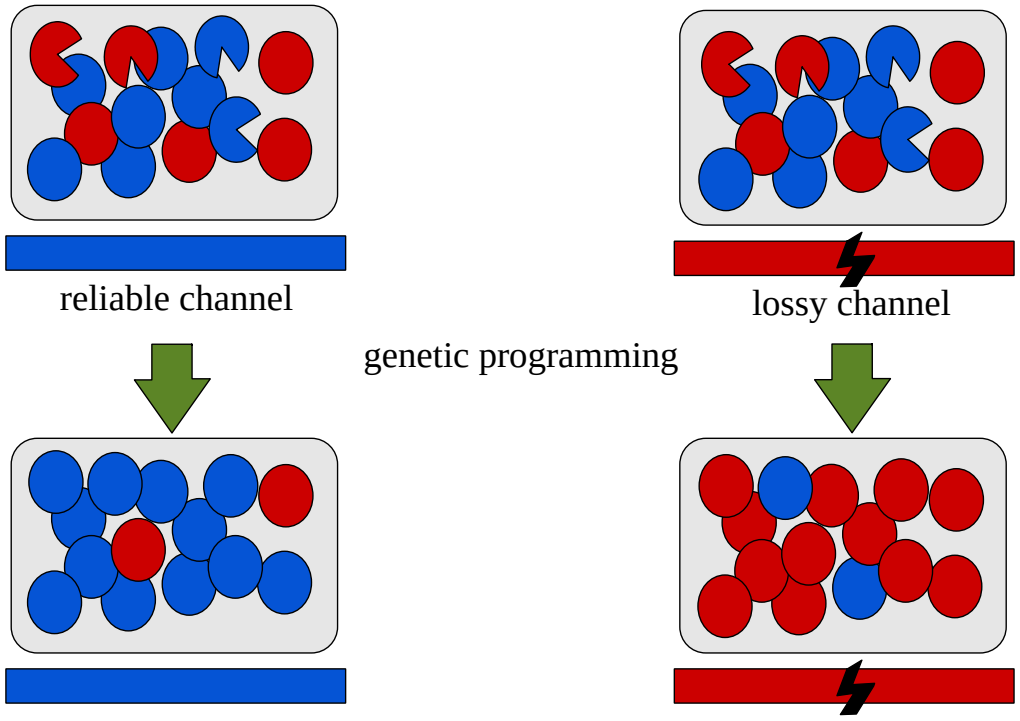
Framework for Protocol Evolution



Experiments

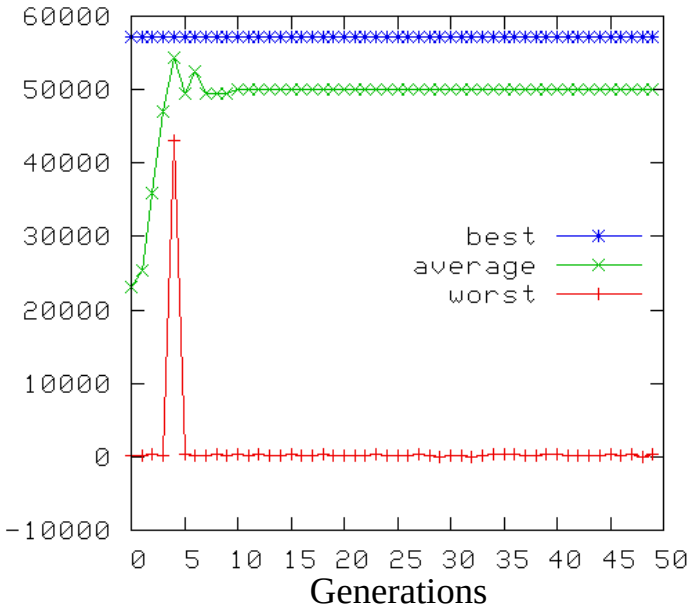
- Reliable transmission protocols
  - Initial population: Multiple variants (alternative implementations): more/less efficient
- Two types of channel
  - Reliable transmission channel (no packet loss)
  - Unreliable (lossy) channel
- Application: requires 100% reliable transmission
  - Underlying protocol must retransmit lost packets if channel is unreliable
  - Fitness evaluation = measure of reliability = score
- Adaptation: selection and dissemination of suitable code

Adaptation Experiment

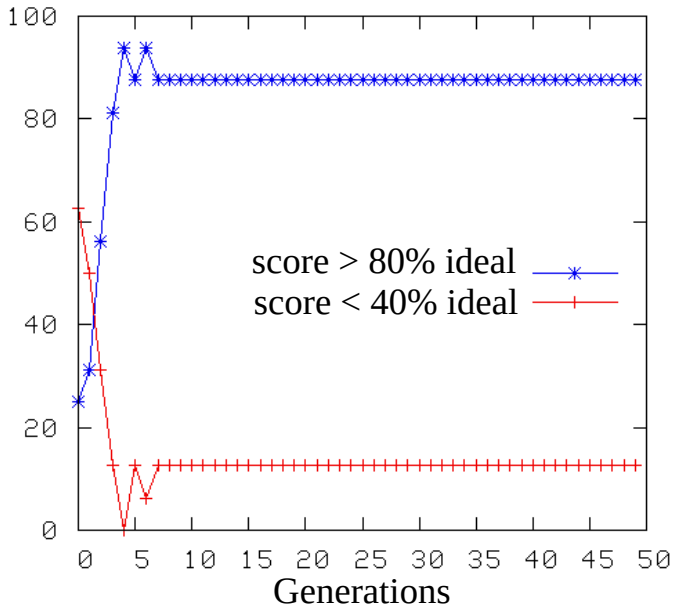


Adaptation Experiment: No Packet Loss

Scores

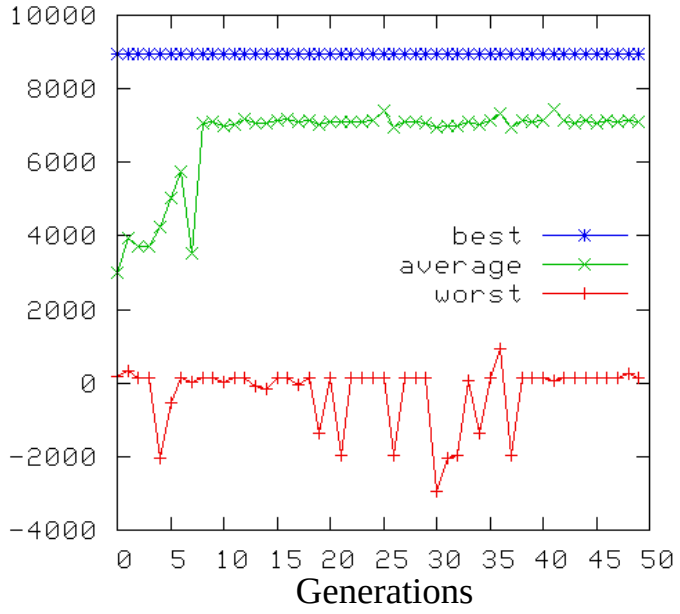


% Individuals

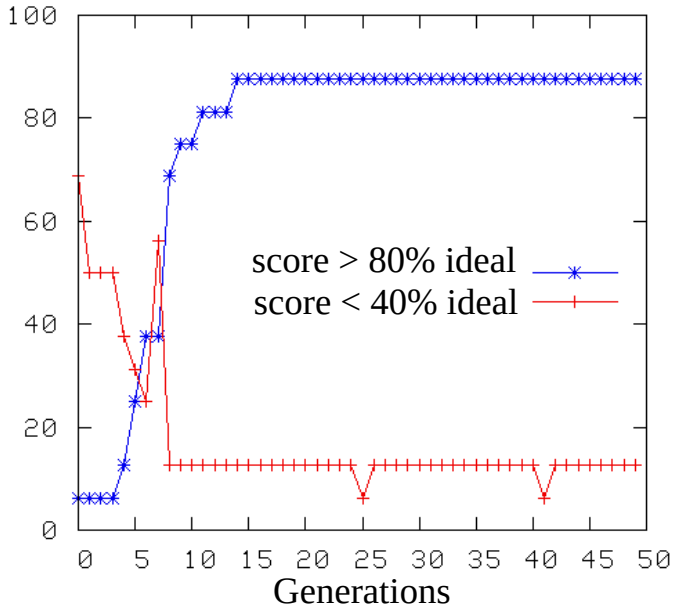


Adaptation Experiment: With Packet Loss

Scores



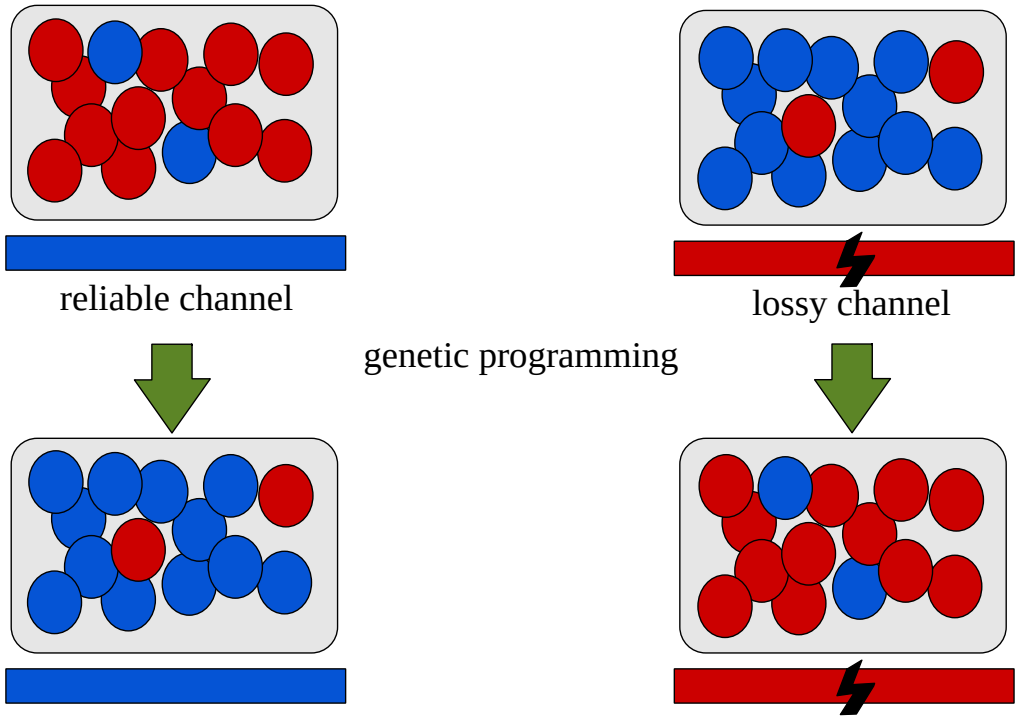
% Individuals



Adaptation Experiment

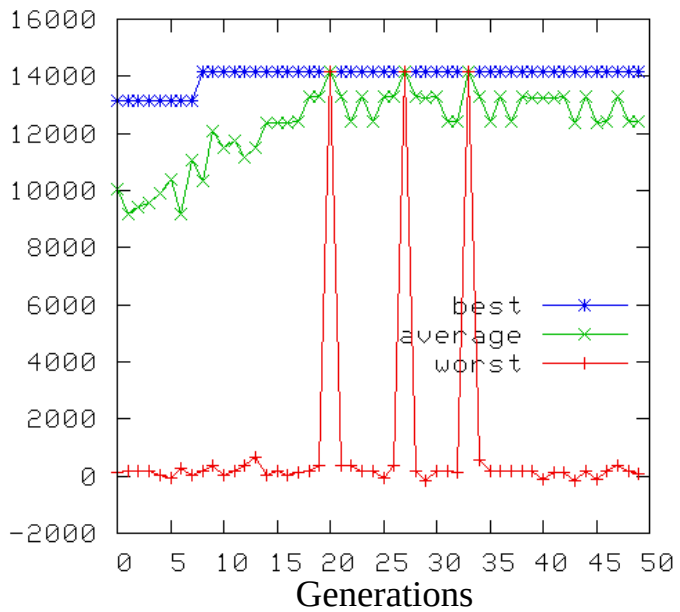
- Successful individuals (ideal scores) remain and also spread in the population
- After a few generations
  - More than 80% of individuals in the population have a score close to the ideal
  - Less than 20% of individuals have poor score
- But genetic variability is severely reduced, population becomes very uniform

Re-adaptation Experiment

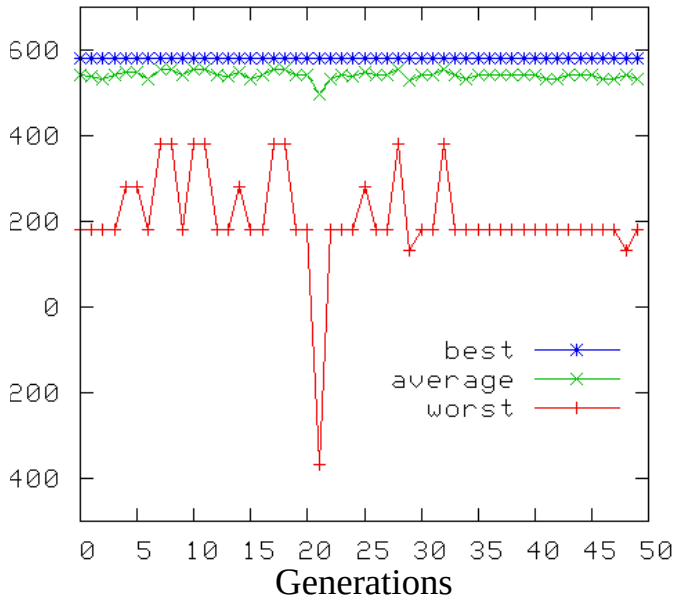


Re-adaptation Experiment (1)

Scores lossy to non-lossy

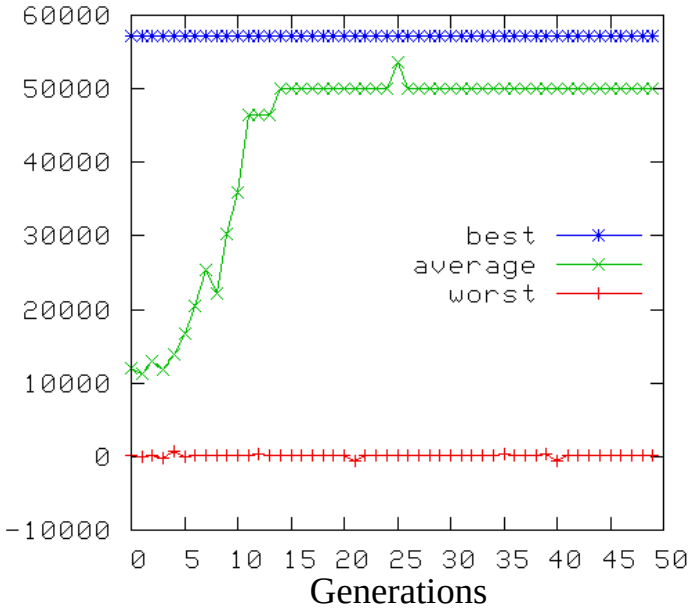


Scores non-lossy to lossy

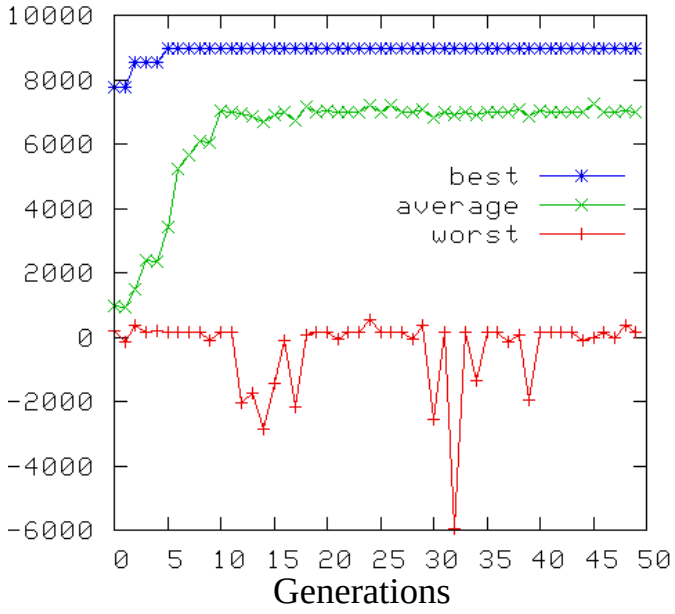


Re-adaptation Experiment (2)

Scores lossy to non-lossy



Scores non-lossy to lossy



Is On-Line Protocol Evolution Possible?

- Answers so far:
  - Can "survival of the fittest" strategy really make best protocols spread in the population?
    - Yes, and quickly, but then genetic variability is lost
  - Can they readapt?
    - Yes, provided that at least one already adapted individual is present in population

Lessons Learned

- Code modification via genetic operators:
  - Crossover
    - Homologous recombination is safe but limited
      - Unable to "create" really new code
    - Unbounded crossover (random points) leads to "intron growth" phenomenon
      - Rise of polluted code containing useless garbage
  - Mutation
    - Currently too slow and random
    - Low probability to produce viable individuals

Conclusions

- Experiments show conditions under which adaptation and re-adaptation are possible: first steps towards evolution
- Resilience at the population level achieved via selection of best and elimination of unsuitable code variants
- GP can do much better than random search, but size of search space still too vast (~ 10^200 for these simple experiments)

Next Steps

- Improve genetic operators
  - Compromise between code safety and variability
  - Hybrid operators using deterministic and formal methods
- Resilience at the individual level
  - Inspired by genome redundancy, metabolic pathways
- Decentralized fitness evaluation
  - Redundant protocol execution circuits
  - Trust and reputation
- Propagation of evolved protocols
  - User and node mobility, code mobility