Evolving Mean-Update Selection Methods for CMA-ES

Samuel Richter (snr359@mst.edu)
Daniel Tauritz (dtauritz@acm.org)

Natural Computation Laboratory
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65409

ECADA @ GECCO, July 2019

Introduction

- Covariance-Matrix Adaptation Evolution Strategies (CMA-ES) use a sample-update cycle to optimize functions

Introduction

- Covariance-Matrix Adaptation Evolution Strategies (CMA-ES) use a sample-update cycle to optimize functions
- CMA-ES keep several state variables, including search space mean, evolution path, and covariance matrix

Introduction

- Covariance-Matrix Adaptation Evolution Strategies (CMA-ES) use a sample-update cycle to optimize functions
- CMA-ES keep several state variables, including search space mean, evolution path, and covariance matrix
- Search space mean is updated every generation using sampled points

Overview

- We wish to tune CMA-ES to a particular problem class

Overview

- We wish to tune CMA-ES to a particular problem class
- We do this by tuning the mean-update method

Overview

- We wish to tune CMA-ES to a particular problem class
- We do this by tuning the mean-update method
- We evolve a new method of selecting the points used to update the mean

Objective

- Objective: use a Hyper-Heuristic to generate a new mean-update selection function for CMA-ES to tune it to a problem class

Objective

- Objective: use a Hyper-Heuristic to generate a new mean-update selection function for CMA-ES to tune it to a problem class
- Step 1: define a representation for selection functions to form a search space

Objective

- Objective: use a Hyper-Heuristic to generate a new mean-update selection function for CMA-ES to tune it to a problem class
- Step 1: define a representation for selection functions to form a search space
- Step 2: explore this space and determine the quality of the selection functions to find the best one

Selection Function Representation

- We use a two-part structure to represent a selection function

Selection Function Representation

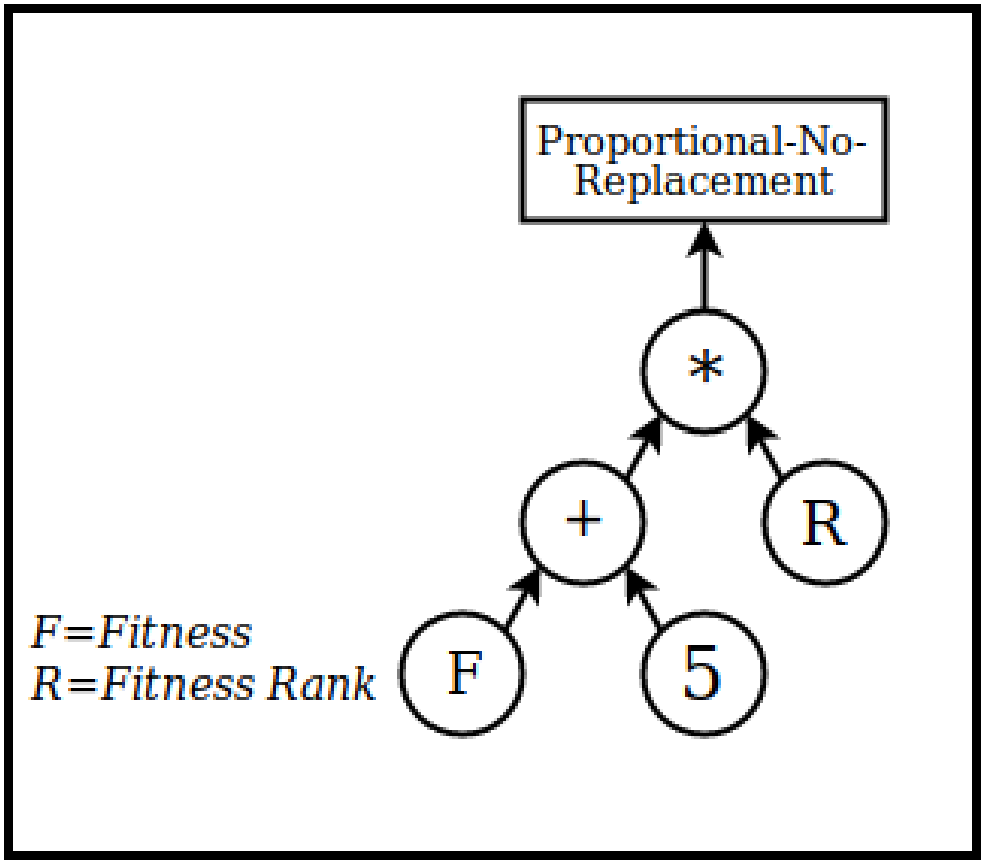
- We use a two-part structure to represent a selection function
- The first part is a GP-Tree encoding a real-valued function of each point's fitness, fitness ranking, and other metrics, returning a "desirability" score for each point

Selection Function Representation

- We use a two-part structure to represent a selection function
- The first part is a GP-Tree encoding a real-valued function of each point's fitness, fitness ranking, and other metrics, returning a "desirability" score for each point
- The second part is a final selection step that selects points based on their desirability scores

Representation

A selection function is represented by a mathematical function (encoded in a parse tree) and a selection method.



Meta-EA

- We use a meta-EA to search through the space of possible mean-update selection functions

Meta-EA

- We use a meta-EA to search through the space of possible mean-update selection functions
- Koza-style GP is used to evolve the trees, with an extra gene encoding the final selection step and any parameters to it

Meta-EA

- We use a meta-EA to search through the space of possible mean-update selection functions
- Koza-style GP is used to evolve the trees, with an extra gene encoding the final selection step and any parameters to it
- Each run of the meta-EA targets one of the 24 noiseless test functions in the Comparing Continuous Optimizers function set

Meta-EA

- We use a meta-EA to search through the space of possible mean-update selection functions
- Koza-style GP is used to evolve the trees, with an extra gene encoding the final selection step and any parameters to it
- Each run of the meta-EA targets one of the 24 noiseless test functions in the Comparing Continuous Optimizers function set
- Dimensions of 2, 3, 5, and 10 are used, each with their own meta-EA

Meta-EA

- Within the meta-EA, a mean-update selection function is rated by running CMA-ES with it

Meta-EA

- Within the meta-EA, a mean-update selection function is rated by running CMA-ES with it
- CMA-ES is run on a number of instances from the problem class. The solution rate is used as the fitness of the selection function

Meta-EA

- Within the meta-EA, a mean-update selection function is rated by running CMA-ES with it
- CMA-ES is run on a number of instances from the problem class. The solution rate is used as the fitness of the selection function
- After the meta-EA concludes, CMA-ES is run with the best selection function on new instances, to test for generalization

Meta-EA Parameters

Parameter	Value
Population Size	40
Offspring Size	40
Evaluation Count	4000
Max GP-Tree Initialization Depth	4
Parent Selection	k-tournament, k=4
Survival Selection	Truncation
Mutation	Subtree Regeneration
Crossover	Subtree Crossover
Parsimony Pressure Coefficient	0.0005
Mutation Rate	0.25
Range for Constant Terminals	[-100, 100]
Range for Random Terminals	[-100, 100]
Number of Runs (Training)	5
Number of Runs (Testing)	200

Results

- On problem classes 4, 6, 12, 17, 18, 19, 20, and 21, the tuned CMA-ES achieved a 20% greater solution rate than base CMA-ES for at least one dimensionality

Results

- On problem classes 4, 6, 12, 17, 18, 19, 20, and 21, the tuned CMA-ES achieved a 20% greater solution rate than base CMA-ES for at least one dimensionality
- For function class 6 and $D = 10$, success rate increased from 0% to 96%

Results

- On problem classes 4, 6, 12, 17, 18, 19, 20, and 21, the tuned CMA-ES achieved a 20% greater solution rate than base CMA-ES for at least one dimensionality
- For function class 6 and $D = 10$, success rate increased from 0% to 96%
- For function class 12, success rate increased from 44% to 100%

Results

- On problem classes 4, 6, 12, 17, 18, 19, 20, and 21, the tuned CMA-ES achieved a 20% greater solution rate than base CMA-ES for at least one dimensionality
- For function class 6 and $D = 10$, success rate increased from 0% to 96%
- For function class 12, success rate increased from 44% to 100%
- Very few cases where the tuned CMA-ES performs worse, and only 7.4% worse at worst

Future Work

- Tune selection strategies for other problems, including real-world problems

Future Work

- Tune selection strategies for other problems, including real-world problems
- Reduce *a priori* computation required for tuning

Future Work

- Tune selection strategies for other problems, including real-world problems
- Reduce *a priori* computation required for tuning
- Tune more CMA-ES variants

“Take Home Message”

Performance of CMA-ES can be improved on a particular problem class by using a meta-EA to tune the method by which sampled points are selected and used to update the search space mean