3.36pt

The Automated Design of Probabilistic Selection Methods for Evolutionary Algorithms

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 2018

Introduction

- For Evolutionary Algorithms (EA's), the method of parent/survival selection has a significant impact on performance

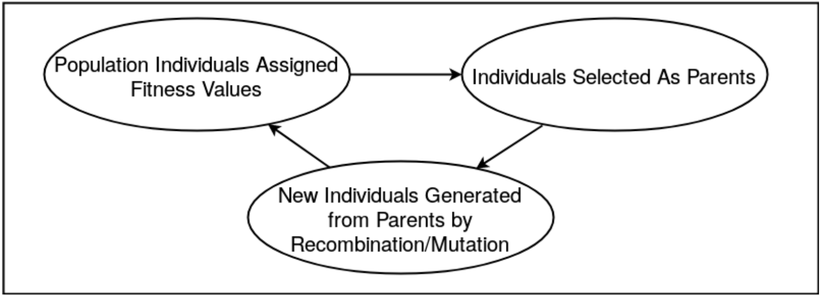
Introduction

- For Evolutionary Algorithms (EA's), the method of parent/survival selection has a significant impact on performance
- Manual Tuning of selection strategy can lead to improvement but is still limited to existing/conventional methods

Introduction

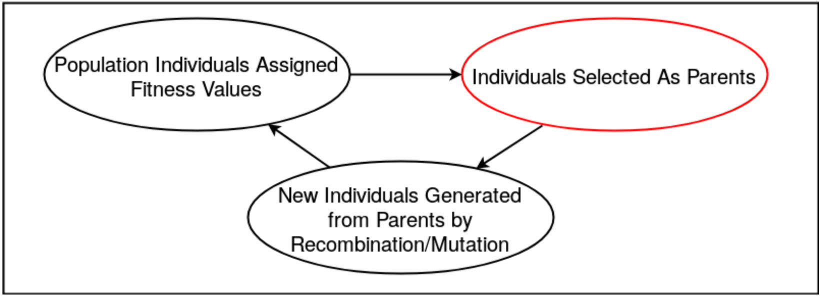
- For Evolutionary Algorithms (EA's), the method of parent/survival selection has a significant impact on performance
- Manual Tuning of selection strategy can lead to improvement but is still limited to existing/conventional methods
- Generation of new selection methods, specialized to particular problems, can improve performance further

Overview



Parent Selection and Survival Selection both strongly influence how genes survive in the population

Overview



We wish to develop a customized selection function specialized to a particular problem class. For this study, we chose to specialize parent selection, while keeping survival selection static.

Overview

- Objective: use a Hyper-Heuristic to generate a selection function for a particular EA operating on a particular problem class

Overview

- Objective: use a Hyper-Heuristic to generate a selection function for a particular EA operating on a particular problem class
- Step 1: define a representation for selection functions to form a search space

Overview

- Objective: use a Hyper-Heuristic to generate a selection function for a particular EA operating on a particular 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

Representation

- A straightforward way to represent selection functions would be to employ a Turing-complete algorithm space that takes a population as input, and returns an individual or pool of individuals

Representation

- A straightforward way to represent selection functions would be to employ a Turing-complete algorithm space that takes a population as input, and returns an individual or pool of individuals
- However, the space of Turing-complete algorithms is large and complex, making it difficult to search through

Representation

- We instead represent selection functions as mathematical functions, which determine the relative probability that any given individual is selected

Representation

- We instead represent selection functions as mathematical functions, which determine the relative probability that any given individual is selected
- This function uses an individual's fitness, fitness ranking, the population size, and the sum of population fitness as inputs, and uses typical mathematical operators (+, -, *, /) as well as a step function

Representation

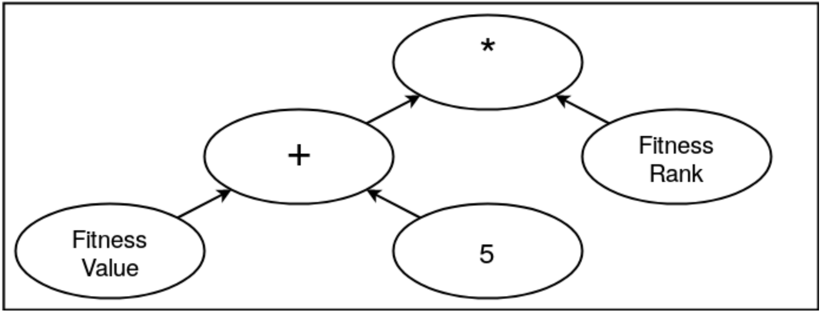
- We instead represent selection functions as mathematical functions, which determine the relative probability that any given individual is selected
- This function uses an individual's fitness, fitness ranking, the population size, and the sum of population fitness as inputs, and uses typical mathematical operators (+, -, *, /) as well as a step function
- The function is evaluated for each individual, and a weighted random selection is performed, using the function output as the weight for each individual

Representation

- We instead represent selection functions as mathematical functions, which determine the relative probability that any given individual is selected
- This function uses an individual's fitness, fitness ranking, the population size, and the sum of population fitness as inputs, and uses typical mathematical operators (+, -, *, /) as well as a step function
- The function is evaluated for each individual, and a weighted random selection is performed, using the function output as the weight for each individual
- An additional boolean variable controls whether selection is performed with replacement

Representation

A selection function is represented by a mathematical function (encoded in a parse tree), and the boolean variable.



Representation - Psuedocode

Algorithm 1 Probabilistic Selection Function

```
procedure ProbabilisticSelection(P)
    for i ← 1, n do
        W(i) ← (P(i).Fitness + 5) · P(i).FitnessRank
    end for
    selected ← WeightedSelection(P, W)
    remove selected from P
    return selected
end procedure
procedure WeightedSelection(P, W)
    w_min ← minimum(W)
    s ← 0
    for all w ∈ W do
        if w_min < 0 then
            s ← s + (w − w_min)
        else
            s ← s + w
        end if
    end for
    r ← random(0, s)
    i ← 1
    while r > W(i) do
        r ← r − W(i)
        i ← i + 1
    end while
    return P(i)
end procedure
```

Representation - Psuedocode

Algorithm 1 Probabilistic Selection Function

procedure ProbabilisticSelection(P)
    for $i \leftarrow 1, n$ **do**
        $W(i) \leftarrow (P(i).Fitness + 5) \cdot P(i).FitnessRank$
    end for
    $selected \leftarrow$ WeightedSelection(P, W)
    remove $selected$ **from** P
    return $selected$
end procedure

Representation - Psuedocode

```
procedure WeightedSelection(P, W)
    w_min ← minimum(W)
    s ← 0
    for all w ∈ W do
        if w_min < 0 then
            s ← s + (w − w_min)
        else
            s ← s + w
        end if
    end for
    r ← random(0, s)
    i ← 1
    while r > W(i) do
        r ← r − W(i)
        i ← i + 1
    end while
    return P(i)
end procedure
```

Representation - Example

Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

Population Member	1	2	3	4
Fitness	300	250	200	350

Representation - Example

Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

Population Member	1	2	3	4
Fitness	300	250	200	350
Fitness Rank (least to greatest)	3	2	1	4

Representation - Example

Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

Population Member	1	2	3	4
Fitness	300	250	200	350
Fitness Rank (least to greatest)	3	2	1	4
$(Fitness + 5) * FitnessRank$	915	510	205	1420

Representation - Example

Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

Population Member	1	2	3	4
Fitness	300	250	200	350
Fitness Rank (least to greatest)	3	2	1	4
$(Fitness + 5) * FitnessRank$	915	510	205	1420
P(selection)	0.3	0.167	0.067	0.466

Representation - Example

Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

Population Member	1	2	3	4
Fitness	300	250	200	350
Fitness Rank (least to greatest)	3	2	1	4
$(Fitness + 5) * FitnessRank$	915	510	205	1420
P(selection)	0.3	0.167	0.067	0.466
$Fitness + 100 * PopulationSize$	700	650	600	750
P(selection)	0.259	0.241	0.222	0.278

Representation - Example

Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

Population Member	1	2	3	4
Fitness	300	250	200	350
Fitness Rank (least to greatest)	3	2	1	4
$(Fitness + 5) * FitnessRank$	915	510	205	1420
P(selection)	0.3	0.167	0.067	0.466
$Fitness + 100 * PopulationSize$	700	650	600	750
P(selection)	0.259	0.241	0.222	0.278
$step(Fitness, 300) * Fitness$	300	0	0	350
P(selection)	0.461	0.0	0.0	0.549

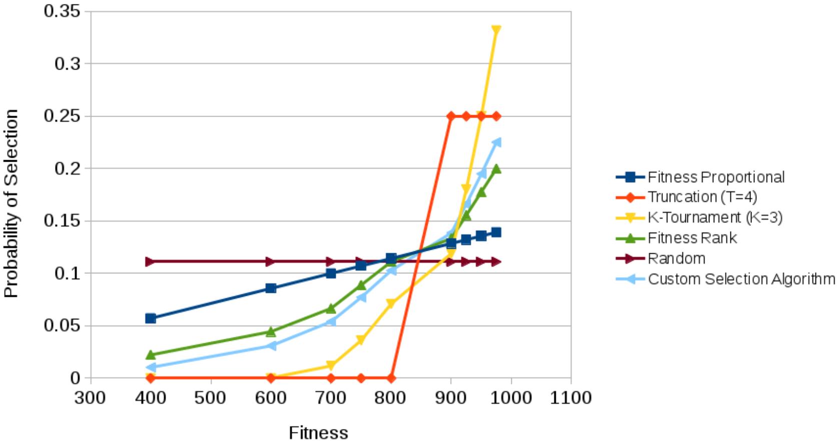
Representation - Example

Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

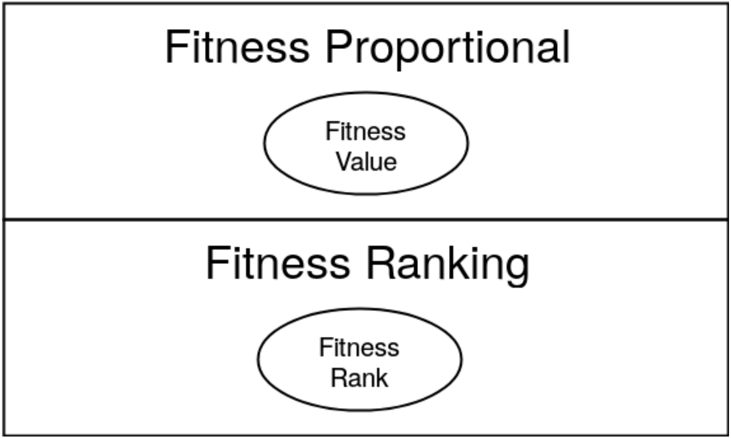
Population Member	1	2	3	4
Fitness	300	250	200	350
Fitness Rank (least to greatest)	3	2	1	4
$(Fitness + 5) * FitnessRank$	915	510	205	1420
P(selection)	0.3	0.167	0.067	0.466
$Fitness + 100 * PopulationSize$	700	650	600	750
P(selection)	0.259	0.241	0.222	0.278
$step(Fitness, 300) * Fitness$	300	0	0	350
P(selection)	0.461	0.0	0.0	0.549
P(fitness-proportional)	0.273	0.227	0.182	0.318
P(fitness-ranking)	0.3	0.2	0.1	0.4

Representation

Different selection functions, including conventional and our evolved selection functions, offer different probability distributions over the population

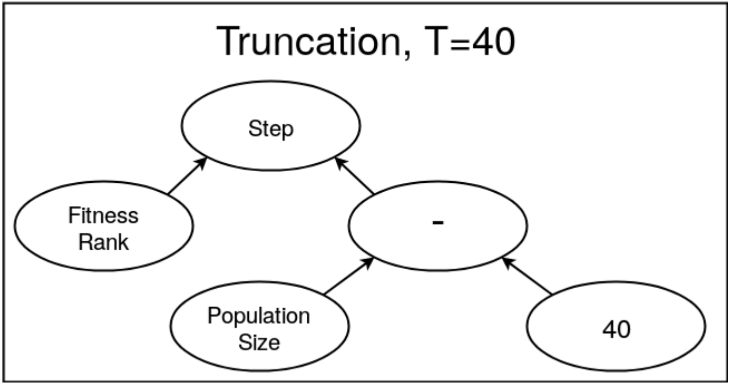


Many conventional selection functions can be represented in this format



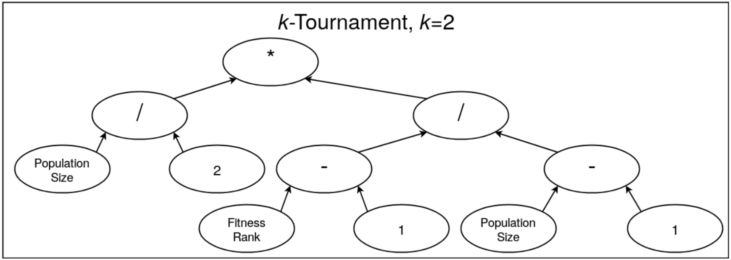
Representation

Many conventional selection functions can be represented in this format



Representation

Many conventional selection functions can be represented in this format



Representation

- This format allows us a more manageable search space of selection functions, while also being robust enough to represent both conventional selection functions and a wide variety of new selection functions

Representation

- This format allows us a more manageable search space of selection functions, while also being robust enough to represent both conventional selection functions and a wide variety of new selection functions
- In addition, we can guarantee that all functions in this space are valid selection functions

Representation

- This format allows us a more manageable search space of selection functions, while also being robust enough to represent both conventional selection functions and a wide variety of new selection functions
- In addition, we can guarantee that all functions in this space are valid selection functions
- We cannot guarantee that all possible selection functions can be represented in this format, but this is an acceptable tradeoff for the more easily-searchable space of selection functions

Meta-EA

- We use Koza-style GP to evolve the trees representing the selection strategies

Meta-EA

- We use Koza-style GP to evolve the trees representing the selection strategies
- Each strategy is evaluated by running a sub-level EA utilizing that strategy on a fixed set of benchmark instances from the NK-Landscape problem class, with N=40 and K=8

Meta-EA

- We use Koza-style GP to evolve the trees representing the selection strategies
- Each strategy is evaluated by running a sub-level EA utilizing that strategy on a fixed set of benchmark instances from the NK-Landscape problem class, with N=40 and K=8
- After the meta-EA is run, the selection strategies are tested for generalization on a separate set of "testing" instances from the same problem class

Meta-EA Parameters

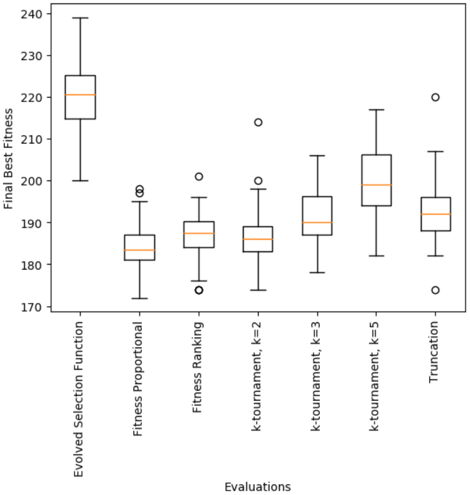
Parameter	Value
Population Size	50
Offspring Size	50
Evaluation Count	2500
Max GP-Tree Initialization Depth	3
Parent Selection	k-tournament, k=10
Survival Selection	Random
Mutation	Subtree Regeneration
Crossover	Subtree Crossover
Parsimony Pressure Coefficient	1
Mutation Rate	0.1
Training Instances	20
Runs per Training Instance	3
Testing Instances	50
Runs per Testing Instance	100
Range for Constant Terminals	[-100, 100]
Range for Random Terminals	[-100, 100]

Base EA Parameters

Parameter	Value
Population Size	100
Offspring Size	20
Genome Length (N)	40
Locus Length (K)	8
Locus Minimum Value	0
Locus Maximum Value	8
Parent Selection	(evolved in Meta-EA)
Survival Selection	Random
Termination Criteria	Convergence
Generations to Convergence	25
Mutation	Random Bit Flip
Mutation Rate	0.05
Crossover	Uniform Crossover

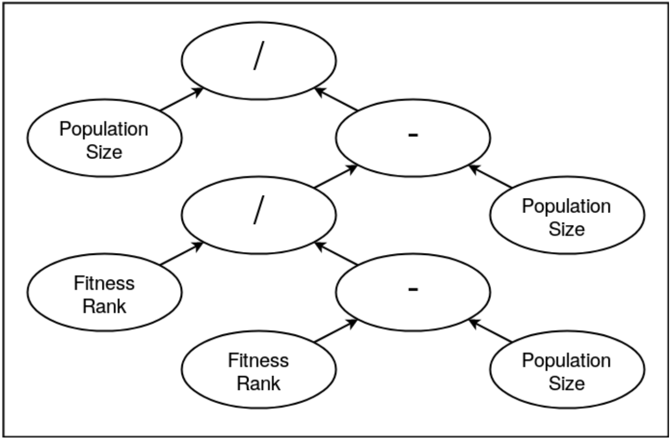
Results - Comparison with Conventional Selection Methods

The final best selection strategy evolved by the Meta-EA outperformed all conventional selection strategies tested on 94% of the 50 testing instances.



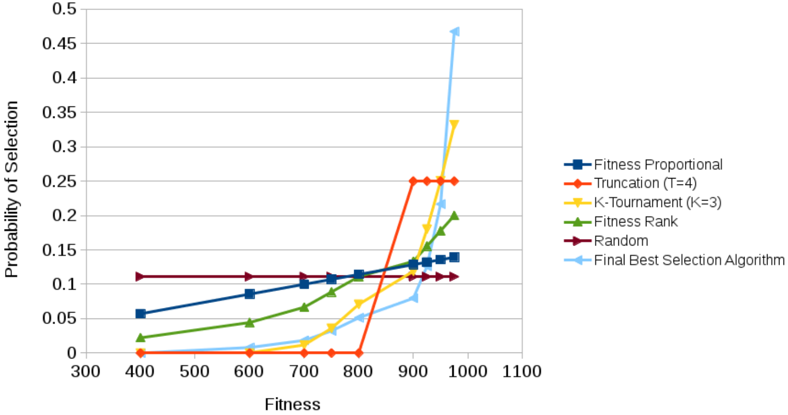
Results - Final Best Evolved Selection Strategy

$P(selection) \propto PopulationSize/((FitnessRank/(FitnessRank - PopulationSize)) - PopulationSize)$, selecting **with** replacement



The Final Best Evolved Selection Strategy

Results - Final Best Evolved Selection Strategy



$P(selection) \propto PopulationSize/((FitnessRank/(FitnessRank - PopulationSize)) - PopulationSize)$, selecting **with** replacement

Limitations

- The Meta-EA can require large amounts of computing resources, especially if the sub-level EA is computationally expensive

Limitations

- The Meta-EA can require large amounts of computing resources, especially if the sub-level EA is computationally expensive
- The evolved selection methods are static, and do not adapt/self-adapt as the evolution progresses

Limitations

- The Meta-EA can require large amounts of computing resources, especially if the sub-level EA is computationally expensive
- The evolved selection methods are static, and do not adapt/self-adapt as the evolution progresses
- The sub-level EA uses static parameters for population size, number of offspring generated, mutation rate, and survival selection

Future Work

- Investigate tuning of survival selection, as well as parent selection

Future Work

- Investigate tuning of survival selection, as well as parent selection
- Tune selection strategies for other problems, including real-world problems

Future Work

- Investigate tuning of survival selection, as well as parent selection
- Tune selection strategies for other problems, including real-world problems
- Evolve dynamic/self-adjusting selection strategies

Future Work

- Investigate tuning of survival selection, as well as parent selection
- Tune selection strategies for other problems, including real-world problems
- Evolve dynamic/self-adjusting selection strategies
- Evolve selection strategies for an EA using dynamic/adaptive parameters, rather than static/fixed parameters

Future Work

- Investigate tuning of survival selection, as well as parent selection
- Tune selection strategies for other problems, including real-world problems
- Evolve dynamic/self-adjusting selection strategies
- Evolve selection strategies for an EA using dynamic/adaptive parameters, rather than static/fixed parameters
- Investigate other weighted selection strategies apart from a weighted random selection.

Future Work

- Investigate tuning of survival selection, as well as parent selection
- Tune selection strategies for other problems, including real-world problems
- Evolve dynamic/self-adjusting selection strategies
- Evolve selection strategies for an EA using dynamic/adaptive parameters, rather than static/fixed parameters
- Investigate other weighted selection strategies apart from a weighted random selection.
- Investigate systems to encourage more diversity in parents selected, beyond a binary "with/without replacement" setting

“Take Home Message”

- An EA running on a problem class can gain a performance benefit with a selection function specialized to that problem class (with all other parameters unchanged)

"Take Home Message"

- An EA running on a problem class can gain a performance benefit with a selection function specialized to that problem class (with all other parameters unchanged)
- A Meta-EA Hyper-heuristic can be used to search through a space of selection functions to find one that significantly outperforms common selection functions for a particular problem class