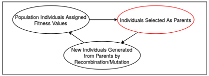
2018-07-09

Automated Design of Probabilistic Selection Methods for EA's

Overview

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.

Our choice to improve parent selection, rather than survival selection or both simultaneously, was mostly arbitrary for the sake of proof-of-concept

Automated Design of Probabilistic Selection Methods for EA's

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

1. We are generating selection functions by defining a search space containing new selection functions, then searching through that space

- 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

1. This is a straightforward representation, but not the only possible straightforward 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

1. The Turing-complete space has the additional challenge of ensuring that the algorithms generated are valid selection functions

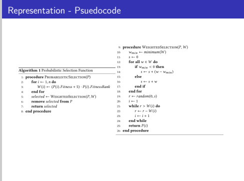
- We instead represent selection functions as mathematical functions, which determine the relative probability that any given individual is selected
- This function uses an individual's fitness, fitness ranking, the population size, and the sum of population fitness as inputs, and uses typical mathematical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

1. We chose these particular terminals and operators because all of the conventional selection functions we tested against (fitness-proportional, k-tournament, etc.) can be represented by them.
   The step function, which is formally the "Heaviside Step Function", in this case, is a binary function. When the left operand is greater than or equal to the right operand, the function returns 1; otherwise, it returns 0

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

1. When the boolean variable is True, individuals may be selected more than once within a single generation. When it is False, an individual can only be selected once per generation



Here we show the psuedocode of the generated selection function. It is composed of two parts: an assignment of function outputs to population members (line 3) and a weighted selection being performed using the function outputs as weights. "FitnessRank" refers to the individual's fitness ranking in the population, where 1 is the ranking of the lowest-fitness individual and n is the ranking of the highest-fitness individual, in a population of n individuals. This particular code is generated by the example given earlier (the exact portions of code affected are highlighted in the next slide).

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
```

Here we show the first part of the algorithm in more detail. The portion of code highlighted in blue is the portion changed by a different function tree. The line highlighted in red is present if the "replacement" bit is set to True, and removed if the bit is set to False.

Automated Design of Probabilistic Selection Methods for EA's

└Representation - Psuedocode

Representation - Psuedocode

```
9:  procedure WeightedSelection(P, W)
10:     w_min ← minimum(W)
11:     s ← 0
12:     for all w ∈ W do
13:         if w_min < 0 then
14:             s ← s + (w − w_min)
15:         else
16:             s ← s + w
17:         end if
18:     end for
19:     r ← random(0, s)
20:     i ← 1
21:     while r > W(i) do
22:         r ← r − W(i)
23:         i ← i + 1
24:     end while
25:     return P(i)
26: end procedure
```

Here we show the second part of the algorithm in more detail. It is identical to a standard roulette-wheel fitness proportional selection, except for the weights for each individual. Instead of using the fitness values directly, the output of the selection function is used. Note that a typo exists in our published paper, where line 22 is incorrectly "r = r + W(i).

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

Here we provide an example of the function output and P(selection) for a second function

2018-07-09

Automated Design of Probabilistic Selection Methods for EA's

└Representation - Example

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

Here we provide a third function as another example

Automated Design of Probabilistic Selection Methods for EA's

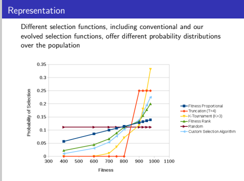
Representation - Example

Representation - Example

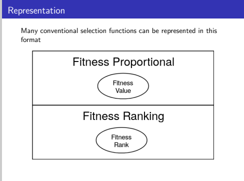
Selection Function: $P(selection) \propto (Fitness + 5) * FitnessRank$

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

Here we show fitness proportional and fitness rank selection for comparison as well



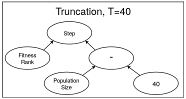
This line graph shows the probability of selected for each individual in a population of 9 individuals, with fitness values of 400, 600, 700, 750, 800, 900, 925, 950, and 975. The "Custom Selection Algorithm" is the same one used in previous examples, $P(selection) \propto (Fitness + 5) * FitnessRank$



Variations of Fitness-Value and Fitness-Rank selection are seen fairly often in the population. Variations of truncation and K-tournament are less common.

Representation

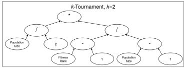
Many conventional selection functions can be represented in this format



Variations of Fitness-Value and Fitness-Rank selection are seen fairly often in the population. Variations of truncation and K-tournament are less common.

Representation

Many conventional selection functions can be represented in this format



Variations of Fitness-Value and Fitness-Rank selection are seen fairly often in the population. Variations of truncation and K-tournament are less common.

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

We chose N=40 and K=8 to generate problem classes that were difficult enough to warrant the use of an EA, but easy enough that a performance benefit could be gained by varying the selection function only.

- 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

We chose N=40 and K=8 to generate problem classes that were difficult enough to warrant the use of an EA, but easy enough that a performance benefit could be gained by varying the selection function only.

- 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

We chose N=40 and K=8 to generate problem classes that were difficult enough to warrant the use of an EA, but easy enough that a performance benefit could be gained by varying the selection function only.

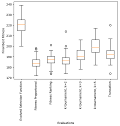
Base EA Parameters

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

At the time the paper was written, the parameters were human-chosen arbitrarily, but now, iRace is used to find good parameters to test against.

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.

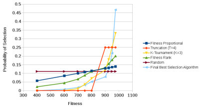


The boxes on these box-and-whisker extend from first to third quartile, and indicate the median. The top and bottom whiskers extend to the most extreme data points within $Q3 + 1.5 * IQR$ and $Q1 - 1.5IQR$, respectively, where IQR is the interquartile range. Data points indicated with circles are outside the whiskers and considered outliers

Automated Design of Probabilistic Selection Methods for EA's

Results - Final Best Evolved Selection Strategy

Results - Final Best Evolved Selection Strategy



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

Here we show the selection chances again, but with a line corresponding to the final evolved best function, and how it would select from the example population given (note that this example population was selected arbitrarily, not part of any experiment, and that the behavior shown may not correspond to what the evolution was trending toward).