Contents

[MAIN FOLDER: gpolnel (from lab classes – last version of 22nd May) 2](#_Toc167666616)

[Folder: **algorithms** 2](#_Toc167666617)

[File: genetic\_algorithm.py 2](#_Toc167666618)

[File: population\_based.py 3](#_Toc167666619)

[File: random\_search.py 4](#_Toc167666620)

[File: search\_algorithm.py 5](#_Toc167666621)

[Folder: **operators** 6](#_Toc167666622)

[File: initializers.py 6](#_Toc167666623)

[File: selectors.py 7](#_Toc167666624)

[File: variators.py 8](#_Toc167666625)

[Folder: **problems** 9](#_Toc167666626)

[File: inductive\_programming.py 9](#_Toc167666627)

[File: neuroevolution.py 11](#_Toc167666628)

[File: problem.py 12](#_Toc167666629)

# MAIN FOLDER: gpolnel (from lab classes – last version of 22nd May)

## Folder: algorithms

### File: genetic\_algorithm.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| GeneticAlgorithm (PopulationBased) | **\_\_init\_\_**  (self, pi, initializer, selector, mutator, crossover, p\_m=0.2, p\_c=0.8, pop\_size=100, elitism=True, reproduction=False, seed=0, device="cpu") |  |
| **PopulationBased.\_\_init\_\_**  (self, pi, initializer, mutator, pop\_size, seed, device) | **PopulationBased** imported from gpolnel.algorithms.population\_based |
| **\_set\_pop**  (self, pop\_repr, tree=True) | If True, creates a pop tree |
| **solve**  (self, n\_iter=20, tol=None, n\_iter\_tol=5, start\_at=None, test\_elite=False, verbose=0, log=0, log\_path='./log/', log\_xp='GPOL') | Implements pseudo-code |
| GSGP  (GeneticAlgorithm) |  | Re-implements GA for GSGP |
| **\_\_init\_\_**  (self, pi, initializer, selector, mutator, crossover, p\_m=0.95, p\_c=0.05, pop\_size=100, elitism=True, reproduction=False, path\_init\_pop=None, path\_rts=None, seed=0, device='cpu') |  |
| **GeneticAlgorithm.\_\_init\_\_**  (self, pi, initializer, selector, mutator, crossover, p\_m, p\_c, pop\_size, elitism, reproduction, seed, device) | **GeneticAlgorithm** from this file |
| **\_set\_best\_sol**  (self) | Encapsulates the set method of the best\_sol attribute of PopulationBased algorithm. |
| **\_initialize**  (self, start\_at=None) |  |
| **solve**  (self, n\_iter=20, tol=None, n\_iter\_tol=5, start\_at=None, test\_elite=False, verbose=0, log=0, log\_path='./log/gsgp.log', log\_xp='GSGP-GPOLNEL') | solve procedure of a GSGP |
| **\_verbose\_reporter**  (self, it, timing, pop, verbose=0) |  |
| **\_create\_log\_event**  (self, it, timing, pop, log, log\_xp='GPOLNEL') | Implements a standardized log-event  Creates a log-event for the underlying best-so-far solution. |
| **write\_history**  (self, path) |  |

### File: population\_based.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| PopulationBased (RandomSearch) | **\_\_init\_\_**  (self, pi, initializer, mutator, pop\_size=100, seed=0, device="cpu") |  |
| **RandomSearch.\_\_init\_\_**  (self, pi, initializer, seed, device) | **RandomSearch** imported from gpolnel.algorithms.random\_search |
| **\_initialize**  (self, start\_at=None, tree=False) | Initializes at given point in *S* |
| **\_set\_pop**  (self, pop\_repr) | Encapsulates the set method of the population attribute of PopulationBased algorithm. |
| **\_set\_best\_sol**  (self) | Encapsulates the set method of the best\_sol attribute of PopulationBased algorithm. |
| **\_create\_log\_event**  (self, it, timing, pop, log, log\_xp='GPOLNEL') | Implements a standardized log-event  Creates a log-event for the underlying best-so-far solution. |
| **\_verbose\_reporter**  (self, it, timing, pop, verbose=0) | Reports the progress of the solve on the console. |
| **solve**  (self, n\_iter=20, tol=None, n\_iter\_tol=5, start\_at=None, test\_elite=False, verbose=0, log=0) |  |
| **elite\_replacement**  (self, offs\_pop, min\_=None, best\_parent=None) | If the best\_parent is better than the best individual in offs\_pop, replaces the worst individual in offs\_pop by the best\_parent. |
| *@staticmethod*  **\_get\_phen\_div**  (pop) | Returns the phenotypic diversity of a population. |

### File: random\_search.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| RandomSearch (SearchAlgorithm) | **\_\_init\_\_**  (self, pi, initializer, seed=0, device="cpu") |  |
| **SearchAlgorithm.\_\_init\_\_**  (self, pi, initializer, device) | **SearchAlgorithm** imported from gpolnel.algorithms.search\_algorithm |
| **\_initialize**  (self, start\_at=None) |  |
| **\_get\_random\_sol**  (self) | Generates random initial solution. |
| **\_create\_log\_event**  (self, it, timing, log) | standardized log-event |
| **\_verbose\_reporter**  (self, it, timing) | Prints status of the solve at a given iteration. Uses the best-so-far solution. |
| **solve**  (self, n\_iter=20, tol=None, n\_iter\_tol=5, start\_at=None, test\_elite=False, verbose=0, log=0) |  |

### File: search\_algorithm.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| SearchAlgorithm |  | Focused on Iterative Search Algorithms |
| **\_\_init\_\_**  (self, pi, initializer, device="cpu") |  |
| **\_initialize**  (self, start\_at=None) |  |
| **\_create\_log\_event**  (self) | standardized log-event - pass |
| **\_verbose\_reporter**  (self) | Prints status of the solve at a given iteration, w/ best-so-far solution. (pass) |
| **solve**  (self, n\_iter=20, tol=None, n\_iter\_tol=5, start\_at=None, test\_elite=False, verbose=0, log=0) |  |
| **\_check\_tol**  (self, last fit, tol, n\_iter\_bare) |  |
| **\_get\_best**  (self, cand\_a, cand\_b) | Compares two candidate solutions and returns the best – based on fitness. |
| **\_get\_worst**  (self, cand\_a, cand\_b) | Compares two candidate solutions and returns the worst – based on fitness. |

## Folder: operators

### File: initializers.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| Terminal |  | Tree terminal class represents terminal nodes of GP trees. |
| **\_\_init\_\_**  (self, constant\_set, p\_constants, n\_dims, device) |  |
| **\_initialize**  (self) | Terminal nodes can be constants or dataset features. |
| **erc**  (self) | Initializes Ephemeral Random Constant, returns a tensor generated constant. |
| **cte**  (self) | Initializes a random constant. |
| **dataset\_feature**  (self) | Initializes the dataset feature, returns an int index of dset feature. |
| ERC | **\_\_init\_\_**  (self, min, max) |  |
| **\_\_str\_\_**  (self) |  |
| Constant | **\_\_init\_\_**  (self, values) |  |
| **\_\_str\_\_**  (self) |  |
|  | **grow**  (space) | Implements Grow initialization algorithm for GP. |
| **prm\_grow**  (sspace) |  |
| **grow\_**  () |  |
| **full**  (sspace) |  |
| **prm\_full**  (sspace) |  |
| **full\_**  (sspace) |  |
| **full**  (sspace) |  |
| **rhh**  (sspace, n\_sols) | Ramped half-and-half init |
| **nn\_init\_individual**  (sspace) |  |
| **nn\_init**  (sspace, n\_sols) |  |

### File: selectors.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
|  | **prm\_tournament**  (pressure) | Implements tournament selection, returns index of selected solution, not representation – used for GA or GSGP. |
| **tournament**  (pop, min\_) | Returns index of most-fit from pool. |
| **roulette\_wheel**  (pop, min\_) | Returns index of solution after fitness proportionate |
| **rank\_selection**  (pop, min\_) | Return index of higher ranked solution. |
| **Rnd\_selection**  (pop, min\_) | Returns random index in [0, len(pop)[ range |

### File: variators.py

|  |  |  |
| --- | --- | --- |
| **Function** | Sub-functions | Origin / notes |
| *Inductive Programming* | | |
|  | **swap\_xo**  (p1, p2) | Swap xo = std GP xo – swaps parents’ two random selected subtrees. |
|  | **hoist\_mtn**  (repr\_) | Offpring = parent w/ rnd selected subtree replaced by another random subtree. |
| **prm\_point\_mtn**  (sspace, prob) |  |  |
| **point\_mtn**  (repr\_) |  |
| **prm\_subtree\_mtn**  (initializer) |  |  |
| **subtree\_mtn**  (repr\_) | Subtree mutation = std GP mutation |
| **prm\_gs\_xo**  (initializer, device) |  | Geometric Semantic Crossover |
| **gs\_xo**  (p1, p2) |  |
| **prm\_gs\_mtn**  (initializer, ms) |  | Geometric Semantic Mutation |
| **gs\_mtn**  (repr\_) |  |
| **prm\_efficient\_gs\_xo**  (X, initializer) |  | Efficient variant of GSC |
| **efficient\_gs\_xo**  (p1, p2) |  |
| **prm\_efficient\_gs\_mtn**  (X, initializer, ms) |  | Efficient variant of GSM |
| **efficient\_gs\_mtn**  (repr\_) |  |
| *Neuroevolution* | | |
|  | **nn\_xo**  (p1, p2) | # CODE HERE |
| **prm\_nn\_mtn**  (ms, sspace) |  |  |
|  | **nn\_mtn**  (repr\_) | # CODE HERE |

## Folder: problems

### File: inductive\_programming.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| SML (Problem) |  | Supervised ML problem in scope of Inductive Programming. ‘IP-SML’ |
| **\_\_init\_\_**  (self, sspace, ffunction, dl\_train, dl\_test=None, min\_=None, n\_jobs=1) | dl = DataLoader |
| **\_evaluate\_sol**  (self, sol, test) | Encapsulates the evaluation of a single solution w/ the ffunction and dset of current SML instance |
| **\_evaluate\_sol\_data\_loader**  (self, sol, data\_loader) | ‘’… ‘’ given by data\_loader arg. |
| **\_evaluate\_sol\_ffunction\_data\_loader**  (self, ffunction, sol, data\_loader) |  |
| **\_evaluate\_sol\_ffunction**  (self, ffunction, sol, test) |  |
| **\_evaluate\_pop\_ffunction**  (self, ffunction, pop) |  |
| **\_is\_feasible\_sol**  (self, repr\_) |  |
| **\_is\_feasible\_pop**  (self, repr\_) |  |
| **evaluate\_sol**  (self, sol, train=True, test=False) |  |
| **evaluate\_sol\_data\_loader**  (self, sol, data\_loader) |  |
| **evaluate\_pop**  (self, pop) |  |
| **predict\_sol\_data\_loader**  (self, repr\_sol, data\_loader, device) |  |

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| SMLGS (Problem) |  | Implements inductive programming OP. |
| **\_\_init\_\_**  (self, sspace, ffunction, X, y, train\_indices, test\_indices, min\_=None) |  |
| **Problem.\_\_init\_\_**  (self, sspace, ffunction, X, y, train\_indices, test\_indices, min\_=None) | **Problem** imported from gpolnel.problems.problem |
| **\_evaluate\_pop\_ffunction**  (self, ffunction, pop, test=False) |  |
| **evaluate\_pop**  (self, pop, test=False) |  |
| **\_evaluate\_sol\_ffunction**  (self, ffunction, sol, test) |  |
| **evaluate\_sol**  (self, sol, test=False) |  |
| **\_create\_log\_event**  (self, it, timing, pop, log, log\_xp='GPOLNEL') |  |

### File: neuroevolution.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| SMLNN (SML) |  | Implements SML problem in scope of IP-OPs. |
| **\_\_init\_\_**  (self, sspace, ffunction, dl\_train, dl\_test=None, min\_=None, n\_jobs=1) |  |
| **Problem.\_\_init\_\_**  (self, sspace, ffunction, X, y, train\_indices, test\_indices, min\_=None) | **Problem** imported from gpolnel.problems.problem |
| **\_evaluate\_sol\_ffunction\_data\_loader**  (self, ffunction, sol, data\_loader) |  |
| **\_evaluate\_sol\_ffunction**  (self, ffunction, sol, test) |  |
| **\_evaluate\_pop\_ffunction**  (self, ffunction, pop) |  |
| **\_is\_feasible\_sol**  (self, repr\_) |  |
| **\_is\_feasible\_pop**  (self, repr\_) |  |
| **predict\_sol\_data\_loader**  (self, repr\_sol, data\_loader, activation, device) |  |

### File: problem.py

|  |  |  |
| --- | --- | --- |
| **Class definitions** | Functions | Origin / notes |
| Problem |  | Definition of OP. |
| **\_\_init\_\_**  (self, sspace, ffunction, min\_=True) |  |
| **\_is\_feasible\_sol**  (self, repr\_) | pass |
| **\_is\_feasible\_pop**  (self, repr\_) | pass |
| **evaluate\_sol**  (self, sol) | pass |
| **evaluate\_pop**  (self, pop) | pass |
| **set\_bad\_fit\_sol**  (self, sol, test=False, device=’cpu’) | Sets ‘very bad’ fitness if solution not valid. |
| **set\_bad\_fit\_pop**  (self, pop, device=’cpu’) |  |

Is the ‘’pass’’ like to define when it’s needed to use?

**A screenshot of a computer

Description automatically generated**