
SAGA*optimizeDocumentation*

Release 1.0.1.post2

Huan Jin, Hunter N.B. Moseley

Mar 06, 2019

Contents:

1	SAGA_optimize.py	1
2	User Guide	5
2.1	Description	5
2.2	Installation	5
2.3	Basic usage	6
3	The SAGA_optimize Tutorial	7
4	Indices and tables	9
	Python Module Index	11

This module provides the *SAGA* class to find the optimal solutions to a set of parameters based on a given energy function with the simulated annealing and genetic algorithm, and the *ElementDescription* class describing the parameter. The *Guess* class stores a set of *ElementDescription* instances to a given energy function and the *Population* class collects a group of *Guess* instances.

```
class SAGA_optimize.ElementDescription (low=0, high=0, name="", value=None, mutate=None)
```

ElementDescription class describes the optimized parameter to a given energy function.

```
__init__ (low=0, high=0, name="", value=None, mutate=None)
    ElementDescription initializer.
```

Parameters

- **low** (*double*) – minimum value for this element; OPTIONAL if immutable value specified.
- **high** (*double*) – maximum value for this element; OPTIONAL if immutable value specified.
- **name** (*str*) – OPTIONAL - the name of the element.
- **value** (*double*) – OPTIONAL - immutable value for this element.
- **mutate** (*str*) – the method that mutates the element; DEFAULT - mutatePopulation-RangedFloat.

```
class SAGA_optimize.Guess (elementDescriptions, elements, energy=0)
```

Guess class collects all the optimized parameters (*ElementDescription* instances) to a given energy function.

```
__init__ (elementDescriptions, elements, energy=0)
    Guess initializer.
```

Parameters

- **elementDescriptions** (*list*) – a list of *ElementDescription* instances.

- **elements** (*list*) – a list of values for the corresponding *ElementDescription* instances.
- **energy** (*double*) – the energy of the Guess to the given function.

clone()

Clones everything but the energy.

Returns the Guess instance.

Return type *Guess*

class SAGA_optimize.**Population** (*size, elementDescriptions, energyCalculation, direction=-1, initialPopulation=None*)

Population class collects a set of Guess instances.

__init__ (*size, elementDescriptions, energyCalculation, direction=-1, initialPopulation=None*)

Parameters

- **size** (*int*) – the number of *Guess* instances in the population.
- **elementDescriptions** (*list*) – a list of *ElementDescription* instances in the *Guess*.
- **energyCalculation** – the given energy function.
- **direction** (*int*) – (1 or -1) for determining lowest energy.
- **initialPopulation** – an initial *Population* instance.

class SAGA_optimize.**SAGA** (*stepNumber, startTemperature, temperatureStepSize, alpha, populationSize, energyCalculation, direction=-1, elementDescriptions=None, startPopulation=None, initialPopulation=None, crossoverRate=0.1, crossover=None, acceptedCriteria=None, mutationRate=1, annealMutationRate=1, maxEnergy=None, crossoverProbabilities=None, validGuess=None, bestOperation=None, bestResultsFile=None, allResultsFile=None*)

Implements a stimulated annealing / genetic algorithm optimization strategy.

__init__ (*stepNumber, startTemperature, temperatureStepSize, alpha, populationSize, energyCalculation, direction=-1, elementDescriptions=None, startPopulation=None, initialPopulation=None, crossoverRate=0.1, crossover=None, acceptedCriteria=None, mutationRate=1, annealMutationRate=1, maxEnergy=None, crossoverProbabilities=None, validGuess=None, bestOperation=None, bestResultsFile=None, allResultsFile=None*)

Parameters

- **stepNumber** (*int*) – number of simple steps to perform.
- **startTemperature** (*double*) – starting temperature.
- **temperatureStepSize** (*int*) – number of simple steps in a temperature step.
- **alpha** (*double*) – power of annealing rate; 1 is linear.
- **populationSize** (*int*) – size of the population of Guesses.
- **energyCalculation** – function to calculate the energy.
- **direction** (*int*) – optimization direction; 1 is maximizing; -1 is minimizing; DEFAULT is -1.
- **elementDescriptions** (*list*) – OPTIONAL - list of *ElementDescription* instances.

- **startPopulation** (*Population*) – OPTIONAL - *Population* instance to use as the starting population.
- **initialPopulation** (*Population*) – OPTIONAL - *Population* instance to initialize with.
- **crossoverRate** (*double*) – fractional rate of crossover versus mutation; DEFAULT is 0.1.
- **mutationRate** (*int*) – number of mutations to perform in creating a new Guess; DEFAULT is 1.
- **annealMutationRate** – whether to anneal mutationRate with temperature; DEFAULT is 1.
- **maxEnergy** (*double*) – OPTIONAL - override of maxEnergy for SA calculation.
- **validGuess** – function that tests if a Guess object is valid. DEFAULT is None.
- **bestOperation** – function to perform on best Guess object; DEFAULT is None.

addElementDescriptions (**elementDescriptions*)

Add elementDescriptions.

Parameters **elementDescriptions** (*ElementDescription*) – the *ElementDescription* instance.

optimize ()

Performs the optimization.

Returns *Population*.

2.1 Description

The *SAGA_optimize* package provides a simple Python interface for solving boundary-value inverse problem based on the simulated annealing and genetic algorithm.

2.2 Installation

SAGA_optimize runs under Python 3.6+ and is available through python3-pip. Install via pip or clone the git repo and install the following dependencies and you are ready to go!

2.2.1 Install on Linux

Pip installation (method 1)

```
pip3 install SAGA-optimize
```

GitHub Package installation (method 2)

Make sure you have `git` installed:

```
cd ~/
git clone https://github.com/MoseleyBioinformaticsLab/SAGA_optimize.git
```

Dependencies

SAGA_optimize requires the following Python libraries:

- **JSONPickle** for saving Python objects in a JSON serializable form and outputting to a file.

2.3 Basic usage

The *SAGA_{optimize}* package is used to find the optimal solutions to a set of parameters based on a given energy function calculated using the set of parameters.

Note: Read *The SAGA_{optimize} Tutorial* to learn more and see code examples on using the *SAGA_{optimize}* as a library.

The SAGA_optimize Tutorial

The *SAGA_optimize* package is a novel type of combined simulated annealing and genetic algorithm used to find the optimal solutions to a set of parameters based on a given energy function calculated using the set of parameters.

To perform *SAGA_optimize*, we need define an energy function first.

```
>>> def energyCalculation(elements):      # example of energy function.
>>>     energy = 0
>>>     for index in range(len(elements)):
>>>         energy += abs(index+1-elements[index])
>>>     return energy
```

Then we need to construct the parameters (*ElementDescription* instances) for optimization. *ElementDescription* instance initialization may require the range of the parameter, and its name. Default value and mutation function can also be provided. Please refer to the API documentation for detailed information.

We can construct *ElementDescription* instances first, and then pass them into SAGA.

```
>>> import SAGA_optimize
>>> element1 = SAGA_optimize.ElementDescription(low=0, high=10, name=
↪ 'element1')      # ElementDescription instance creation.
>>> element2 = SAGA_optimize.ElementDescription(low=0, high=10, name=
↪ 'element2')
>>> element3 = SAGA_optimize.ElementDescription(low=0, high=10, name=
↪ 'element3')
>>> element4 = SAGA_optimize.ElementDescription(low=0, high=10, name=
↪ 'element4')
>>> element5 = SAGA_optimize.ElementDescription(low=0, high=10, name=
↪ 'element5')
>>> elements = [element1, element2, element3, element4, element5]
>>> saga = SAGA_optimize.SAGA(stepNumber=100000, temperatureStepSize=100,
↪ startTemperature=0.5, elementDescriptions=elements,
                                alpha=1, direction=-1,
↪ energyCalculation=energyCalculation, crossoverRate=0.5, mutationRate=3,
                                annealMutationRate=1, populationSize=20)
```

Or we can create SAGA instance first, and then add the ElementDescription instances.

```
>>> import SAGA_optimize
>>> saga = SAGA_optimize.SAGA(stepNumber=100000, temperatureStepSize=100,
↪ startTemperature=0.5, alpha=1, direction=-1,
                                energyCalculation=energyCalculation,
↪ crossoverRate=0.5, mutationRate=3, annealMutationRate=1,
                                populationSize=20)                # SAGA_
↪ instance creation.
>>> saga.addElementDescriptions(SAGA_optimize.ElementDescription(low=0,
↪ high=10, name='element1'), SAGA_optimize.ElementDescription(low=0, high=10,
↪ name='element2'),
                                SAGA_optimize.ElementDescription(low=0,
↪ high=10, name='element3'), SAGA_optimize.ElementDescription(low=0, high=10,
↪ name='element4'),
                                SAGA_optimize.ElementDescription(low=0,
↪ high=10, name='element5'))                # Add optimized parameters.
```

Next, we can conduct optimization.

```
>>> optimized_population = saga.optimize()                # the population_
↪ returned after the optimization.
>>> bestIndex = optimized_population.bestIndex            # To get the best_
↪ index of the Population.
>>> bestGuess = optimized_population.bestGuess            # To get the best_
↪ Guess instance of the Population.
>>> print(bestGuess)
```

```
Guess: Energy = 0.010800440413622603 Parameters: Element 1 = 0.
↪ 9986605131302921 Element 2 = 2.0049781612156004 Element 3 = 3.
↪ 0036003043186144 Element 4 = 3.999532176465393 Element 5 = 5.
↪ 000414664475093
```

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

S

SAGA_optimize, 1

Symbols

`__init__()` (SAGA_optimize.ElementDescription method), 1
`__init__()` (SAGA_optimize.Guess method), 1
`__init__()` (SAGA_optimize.Population method), 2
`__init__()` (SAGA_optimize.SAGA method), 2

A

`addElementDescriptions()` (SAGA_optimize.SAGA method), 3

C

`clone()` (SAGA_optimize.Guess method), 2

E

ElementDescription (class in SAGA_optimize), 1

G

Guess (class in SAGA_optimize), 1

O

`optimize()` (SAGA_optimize.SAGA method), 3

P

Population (class in SAGA_optimize), 2

S

SAGA (class in SAGA_optimize), 2
SAGA_optimize (module), 1