cnc Documentation

Release 1.0.0

Stefan Petrovich

Apr 30, 2019

CONTENTS:

CHAPTER

ONE

INSTALLATION

1.1 Requirements

This package depends on the following packages, which are automatically installed with the provided installer:

1. Python 3
2. numpy
3. bokeh
4. tqdm

1.2 How To Install

This package is distributed using *setuptools*, and can be installed using the *setup.py* file provided with the package. The package is installed into the currently active Python environment using the following command:

```
>>> python setup.py install
```

CHAPTER

TWO

USAGE

2.1 CNCOptimizer

The *CNCOptimizer* class is the only class needed in order to run the optimization. An instance of the class gets initialized with the path to the .code file that needs to be processed, and a flag that tells the optimizer whether to ignore recipes or not.

After instantiating an *CNCOptimizer* object, the *.optimize* method needs to be called, after which the *.save* method is used to save the result of the optimization to a .code file.

A visualization can be generated, if needed, using the *.visualize* method, which generates a *result.html* file, that can be viewed using any browser.

2.2 Full Example

A full example, which is also provided with this package, in the *run.py* file, is given below:

```
from cnc.optimization import CNCOptimizer

# Path to input file
path_file = './path_files/lea_stpl001_fused.code'

# Generate optimization object
opt = CNCOptimizer(path_file, recipe_grouping=False)

# Run the optimization
opt.optimize()

# Write the optimization to a file
opt.save('result')

# Generate visualization file
opt.visualize()
```

CHAPTER

THREE

REFERENCE

class cnc.optimization.**CNCOptimizer**(*file_path*, *recipe_grouping=True*)

Finds the shortest path for CNC cutting.

Parses the input file, and generates a number of GeneticAlgorithm objects, for different groups of lines, and performs the optimization in parallel. Makes visualizations of the cutting trajectory using bokeh.

Methods

generate_lines_from_file()	Parses the input file and generates Line objects for every line.
optimize()	Groups Line objects, and runs a thread per optimization group.
save(file_name)	Saves the results of the optimization to a file.
start_process(all_optimizations, node_name, ...)	Method that creates the optimization object, starts the optimization and saves the result.
visualize()	Visualizes the result of the optimization, using the Visualizer class.

generate_lines_from_file()

Parses the input file and generates Line objects for every line.

optimize()

Groups Line objects, and runs a thread per optimization group.

save(*file_name*)

Saves the results of the optimization to a file. Adds .code file extension if it's not specified.

Parameters

file_name [str] Filename of the file which will contain the optimized result.

start_process(*all_optimizations*, *node_name*, *node*, *pop_size*, *repro*, *crossover*, *mutation*, *num_generations*, *progress_bar_position*)

Method that creates the optimization object, starts the optimization and saves the result.

Parameters

all_optimizations [dict] Dictionary which will be used to store the optimization object that corresponds to every line group.

node_name [str] The name of the group which is being optimized.

node [list of Line] A list of Line objects, which represent the lines that belong to the group being optimized.

pop_size [int] Size of the population for every generation.

repro [float] Probability of reproduction.

crossover [float] Probability of crossover.

mutation [float] Probability of mutation.

num_generations [int] Number of iterations for which to run the algorithm.

progress_bar_position [int] Determines the row in which the progress bar will be displayed while optimizing.

visualize()
Visualizes the result of the optimization, using the Visualizer class.

class cnc.optimization.**GeneticAlgorithm**(*nodes*, *pop_size*, *repro*, *crossover*, *mutation*, *num_generations*, *progress_bar_position=0*)
Solves an instance of the travelling salesman problem, for the CNC machine.

Finds the shortest cutting tool travel path (SCTTP) using the genetic algorithm optimization method.

Methods

crossover()	Generates part of the population using crossover.
evaluate_generation()	Evaluates the path cost and fitness of the whole generation.
generate_distance_matrix()	Generates matrix of Euclidian distances between every two nodes.
mutation()	Mutates a set number of individuals in the population, by swapping two genes.
optimize()	Runs the optimization algorithm trying to find the shortest path.
reproduction()	Determines which individuals get to move to the next generation (which ones get cloned).

crossover()
Generates part of the population using crossover.

Takes two individuals at a time, based on fitness and combines them, using the Order 1 Crossover method.

evaluate_generation()
Evaluates the path cost and fitness of the whole generation.

Path cost is calculated as the Euclidian distance between the second poin in a node and the first point in the next node. Fitness is calculated as the maxmimum possible path cost, minus the actual path cost.

generate_distance_matrix()
Generates matrix of Euclidian distances between every two nodes.

mutation()
Mutates a set number of individuals in the population, by swapping two genes.

optimize()
Runs the optimization algorithm trying to find the shortest path.

reproduction()
Determines which individuals get to move to the next generation (which ones get cloned).

class cnc.optimization.**Line**(*line_type*, *starting_point*, *endpoint*, *recipe*)
Line which represents where the CNC head will perform cutting.

Methods

get_endpoint()	Returns the endpoint of a line.
get_line_type()	Returns the type of line.
get_recipe()	Returns the recipe of a line.
get_starting_point()	Returns the starting point of a line.
get_thikness()	Returns thikness of EDGEDEL_LINE line type.
set_thikness(thikness)	Set the line thikness, which is only specified for EDGEDEL_LINE line types.

get_endpoint()
Returns the endpoint of a line.

Returns

endpoint [np.array] Numpy array of two coordinates, X2 and Y2, representing the endpoint of cutting.

get_line_type()
Returns the type of line.

Returns

line_type [str] Name of line type.

get_recipe()
Returns the recipe of a line.

Returns

recipe [str] Recipe number.

get_starting_point()
Returns the starting point of a line.

Returns

starting_point [np.array] Numpy array of two coordinates, X1 and Y1, representing the starting point of cutting.

get_thikness()
Returns thikness of EDGEDEL_LINE line type.

Returns

thikness [str] Number representing the thikness of the line.

set_thikness(*thikness*)
Set the line thikness, which is only specified for EDGEDEL_LINE line types.

Parameters

thikness [str] Number representing the thinkess of the line. Not used for calculations, only when writing to new .code file.

class cnc.visualization.**Visualizer**(*result*, *initial*)
Visualizes the result of the optimization for the CNC shortest cutting tool travel path, using bokeh.

Methods

generate_tool_path(data, step_size)	Generates the whole trajectory of the cutting tool, with a step of *step_size*.
populate_plot(plot, data)	Adds data to the plot, like starting and endpoints of cutting, lines representing the cutting path and lines representnig the non-cutting path.
split_line(start, end, increment)	Function that generates a number of points between two points.
visualize()	Generates a plot using bokeh, which displays the initial trajectory and the optimized trajectory of the cutting tool.

generate_tool_path(*data*, *step_size*)

Generates the whole trajectory of the cutting tool, with a step of *step_size*.

Parameters

data [list of Line] List of line objects, from which the trajectory needs to be generated.

step_size [int] The Euclidian distance between two steps of the trajectory.

Returns

out [touple of np.arrays] Touple where the 1st and 2nd element represents the X and Y coordinates of the whole cutting tool trajectory, respectively.

populate_plot(*plot*, *data*)

Adds data to the plot, like starting and endpoints of cutting, lines representing the cutting path and lines representnig the non-cutting path.

Parameters

plot [bokeh figure object] Figure object on which the content of the *data* object will be displayed.

data [list of Line] List of line objects, which have to be drawn on the figure.

Returns

plot [bokeh figure object] Figure populated with data from the *data* object.

split_line(*start*, *end*, *increment*)

Function that generates a number of points between two points.

Generates two vectors, which represent the X and Y coordinates between points *start* and *end*.

Parameters

start [np.array] Numpy array containing X and Y of the starting point.

end [np.array] Numpy array containing X and Y of the endpoint.

increment [int] Euclidian distance between two successive entries in the *start* and *end* points.

Returns

out [touple of np.arrays] Touple where the 1st element is the X coordinates and the 2nd element is the Y coordinates of the line.

visualize()

Generates a plot using bokeh, which displays the initial trajectory and the optimized trajectory of the cutting tool.

PYTHON MODULE INDEX

C

INDEX