Author: Olli Luukkonen

Date: 15.12.2021

**Documentation for ant colony optimization python code.**

**Description**

Purpose of this document is to show sources for methods and parameters used in aco.py algorithm. ACO-algorithm made by John Berroa [3] was used as a template and it was modified for truss optimization.

**Parameter descriptions: class AntColonyOptimizer (only a few parameters are in this list)**

|  |  |
| --- | --- |
| self.pheromone\_matrix | [1] |
| self.heuristic\_matrix | [1] |
| self.probability\_matrix | A picture containing text, gauge, clock, device  Description automatically generated[2] |
| self.penalty\_factor | Factor which increases penalty value. If self.penalty\_factor is too large, ACO converges more easily to local optimum. If value is too small, result might be infeasible. |

**Methods: class AntColonyOptimizer**

|  |  |
| --- | --- |
| get\_min\_f(self): | ***:return****: Minimum objective function value* |
| calculate\_member\_mass(self, member, cross\_section\_area): | Calculate truss member mass |
| initialize\_heuristic\_matrix(self): | Initial value for every matrix members are  1 / (weight of member) |
| update\_probabilities(self) | Calculate |
| initialize(self): | Initialize parameters and matrices |
| eval\_ant(self, ant, x): | Evaluates one ant:  Calculate objective function, penalty value and fitness  return: fitness |
| evaluate(self, ants): | Calculate fitness value for all ants  :param ants: List of ant objects  :return: best objective function, list of best design variables |
| local\_pheromone\_update(self, ant): | Update pheromone value: [1] |
| global\_pheromone\_update\_for\_one\_edge(self, fitness, rank, i, j): | [1, s. 743]  where values are replaced in following way:  λ = self.number\_of\_top\_ants  μ = rank  Lu = fitness value |
| global\_pheromone\_update(self, ants): | Choose best ants based on fitness value and update their pheromone value.  Number of best ants are self.number\_of\_top\_ants  Inspiration taken from equations:  and  [1] |

**Methods: class Ant**

|  |  |
| --- | --- |
| \_\_\_\_choose\_design\_variable(self, i, probability\_matrix, design\_variables): | Choose design variable (cross section area for truss) based on probability P    [2] |
| choose\_design\_variables(self, probability\_matrix, design\_variables): | Choose design variable based on probability P for all members. |
| get\_x(self): | Convert x\_index\_and\_value to np array of x  :return: design variables x |
| get\_fitness(self): | Return fval + penalty |

**References:**

[1] Camp, C. V. ja Bichon, B. J. Design of Space Trusses Using Ant Colony Optimiza-tion. eng.Journal of structural engineering (New York, N.Y.)130.5 (2004), s. 741–751.ISSN: 0733-9445.

[2] Kaveh, A. ja Talatahari, S. An improved ant colony optimization for the design ofplanar steel frames. eng.Engineering structures32.3 (2010), s. 864–873.ISSN:0141-0296.

[3] johnberroa, Ant Colony Optimization, Github, reference date 11.4.2021, [johnberroa/Ant-Colony-Optimization: A Python implementation of the Ant Colony Optimization algorithm for generating solutions to such problems as the Traveling Salesman Problem. (github.com)](https://github.com/johnberroa/Ant-Colony-Optimization)