ACO-Pants 0.5.2

Latest version

```
pip install ACO-Pants
```

Released: Sep 9, 2014

A Python3 implementation of the ACO Meta-Heuristic

Navigation

Project description

Release history

Download files

Project description

A Python3 implementation of the Ant Colony Optimization Meta-Heuristic

Overview

Pants provides you with the ability to quickly determine how to visit a collection of interconnected nodes such that the work done is minimized. Nodes can be any arbitrary collection of data while the edges represent the amount of “work” required to travel between two nodes. Thus, **Pants** is a tool for solving traveling salesman problems.

The world is built from a list of nodes and a function responsible for returning the length of the edge between any two given nodes. The length function need not return actual length. Instead, “length” refers to that the amount of “work” involved in moving from the first node to the second node - whatever that “work” may be. For a silly, random example, it could even be the number of dishes one must wash before moving to the next station at a least dish-washing dish washer competition.

*These details have **not** been verified by PyPI*

Project links

Homepage

Meta

- **License:** LICENSE.txt
- **Author:** Robert Grant

Solutions are found through an iterative process. In each iteration, several ants are allowed to find a solution that "visits" every node of the world. The amount of pheromone on each edge is updated according to the length of the solutions in which it was used. The ant that traveled the least distance is considered to be the local best solution. If the local solution has a shorter distance than the best from any previous iteration, it then becomes the global best solution. The elite ant(s) then deposit their pheromone along the path of the global best solution to strengthen it further, and the process repeats.

You can read more about Ant Colony Optimization on Wikipedia.

Installation

Installation via `pip`

```
$ pip3 install ACO-Pants
```

Usage

Using **Pants** is simple. The example here uses Euclidean distance between 2D nodes with `(x, y)` coordinates, but there are no real requirements for node data of any sort.

1. Import **Pants** (along with any other packages you'll need).

```
import pants
import math
import random
```

2. Create your data points; these become the nodes. Here we create some random 2D points. The only requirement for a node is that it is distinguishable from all of the other nodes.

```
nodes = []
for _ in range(20):
  x = random.uniform(-10, 10)
```

```
  y = random.uniform(-10, 10)
  nodes.append((x, y))
```

3. Define your length function. This function must accept two nodes and return the amount of "work" between them. In this case, Euclidean distance works well.

```
def euclidean(a, b):
    return math.sqrt(pow(a[1] - b[1], 2) + pow(a[0]
```

4. Create the `World` from the nodes and the length function.

```
world = pants.World(nodes, euclidean)
```

5. Create the `Solver`.

```
solver = pants.Solver()
```

6. Solve the `World` with the `Solver`. Two methods are provided for finding solutions: `solve()` and `solutions()`. The former returns the best solution found, whereas the latter returns each solution found if it is the best thus far.

```
solution = solver.solve(world)
# or
solutions = solver.solutions(world)
```

7. Inspect the solution(s).

```
print(solution.distance)
print(solution.tour)    # Nodes visited in order
print(solution.path)    # Edges taken in order
# or
best = float("inf")
for solution in solutions:
  assert solution.distance < best
  best = solution.distance
```

Run the Demo

Included is a 33 “city” demo script that can be run from the command line.

```
user@host:~$ pants-demo -h
usage: pants-demo [-h] [-V] [-a A] [-b B] [-l L] [-p P]
                  [-c N] [-d D]

Script th;at demos the ACO-Pants package.

optional arguments:
  -h, --help          show this help message and exit
  -V, --version       show program's version number and
  -a A, --alpha A     relative importance placed on phe
  -b B, --beta B      relative importance placed on dis
  -l L, --limit L     number of iterations to perform; d
  -p P, --rho P       ratio of evaporated pheromone (0 <
  -e E, --elite E     ratio of elite ant's pheromone; de
  -q Q, --Q Q         total pheromone capacity of each a
  -t T, --t0 T        initial amount of pheromone on eve
                      default=0.01
  -c N, --count N     number of ants used in each iterat
  -d D, --dataset D   specify a particular set of demo d

For best results:
  * 0.5 <= A <= 1
  * 1.0 <= B <= 5
  * A < B
  * L >= 2000
  * N > 1

For more information, please visit https://github.com/r
user@host:~$ pants-demo
Solver settings:
limit=100
rho=0.8, Q=1
alpha=1, beta=3
elite=0.5

Time Elapsed              Distance
-------------------------------------------------
           0:00:00.017490 0.7981182992833705
           0:00:00.034784 0.738147755518648
           0:00:00.069041 0.694362159048816
           0:00:00.276027 0.6818083968312925
           0:00:00.379039 0.6669398280432167
           0:00:00.465924 0.6463548571712562
```

```
           0:00:00.585685 0.6416519698864324
           0:00:01.563389 0.6349308484274142
--------------------------------------------------
Best solution:
         0 = (34.02115, -84.267249)
         9 = (34.048194, -84.262126)
         6 = (34.044915, -84.255772)
        22 = (34.061518, -84.243566)
        23 = (34.062461, -84.240155)
        18 = (34.060461, -84.237402)
        17 = (34.060164, -84.242514)
        12 = (34.04951, -84.226327)
        11 = (34.048679, -84.224917)
         8 = (34.046006, -84.225258)
         7 = (34.045483, -84.221723)
        13 = (34.051529, -84.218865)
        14 = (34.055487, -84.217882)
        16 = (34.059412, -84.216757)
        25 = (34.066471, -84.217717)
        24 = (34.064489, -84.22506)
        20 = (34.063814, -84.225499)
        10 = (34.048312, -84.208885)
        15 = (34.056326, -84.20058)
         5 = (34.024302, -84.16382)
        32 = (34.118162, -84.163304)
        31 = (34.116852, -84.163971)
        30 = (34.109645, -84.177031)
        29 = (34.10584, -84.21667)
        28 = (34.071628, -84.265784)
        27 = (34.068647, -84.283569)
        26 = (34.068455, -84.283782)
        19 = (34.061281, -84.334798)
        21 = (34.061468, -84.33483)
         2 = (34.022585, -84.36215)
         3 = (34.022718, -84.361903)
         4 = (34.023101, -84.36298)
         1 = (34.021342, -84.363437)
Solution length: 0.6349308484274142
Found at 0:00:01.563389 out of 0:00:01.698616 seconds.
user@host:~$
```

Known Bugs

None of which I am currently aware. Please let me know if you find otherwise.

Troubleshooting

Credits

- Robert Grant rhgrant10@gmail.com

License

GPL

Switch to desktop version