University of St.Gallen

Institute of Computer Science

Web-based Autonomous Systems, FS2023
Danai Vachtsevanou, Andrei Ciortea
{firstname}.{lastname}@unisg.ch

Exercise 7: Stigmergic Interaction for Smart Colony Optimization

Deadline: **April 25, 2023; 23:59 CET**

In this exercise, you will gain hands-on experience in programming self-organizing systems in which the environment *mediates the interaction* among autonomous agents through a pheromone infrastructure. You will implement a Multi-Agent System whose agents and environment behave according to an Ant Colony Optimization (ACO) algorithm toward solving an instance of the Travelling Salesman Problem (TSP).

Specifically, you will implement the ant-cycle variant of the Ant System algorithm discussed in the lecture[1], and you will apply the algorithm to a symmetric TSP[2] instance, specified here[3]: Given a set of *48 nodes* and their coordinates, find a roundtrip of minimal total length visiting each node exactly once. The goal is to reach a solution (route) whose length is as close as possible to $L_{min} = 10628$ – the approximate length of the optimal solution (optimal route)[4] for the TSP instance. To interpret the specifications of the TSP instance and solution, please consult the documentation of TSPLIB[5]. Additional specifications of symmetric TSP instances and solutions can be found in TSPLIB[6].

For implementing this task, you can use the provided template, available here[7]. You are free to modify the template as you like (including changing the project structure, the class dependencies, the provided method signatures and bodies, etc.).

① Task 1 (3.5 points): Implement the environment to apply ACO to a TSP instance

Your first task is to implement the environment so that it represents the *topology* of the given TSP instance and enables the *update of pheromone trails phase* of the Ant System algorithm.

Task 1.1: Initialize the environment of the TSP instance Your Ant System should be applied to the environment topology described in the given TSP instance specification, available here[3]. The topology is static and is given as a complete undirected graph consisting of n=48 vertices (i.e., $\frac{n*(n-1)}{2}$ edges). Each vertex represents a city in the USA, and each edge represents a path between two cities.

Implement a class `Environment` that represents the *environment topology* based on the TSP instance specification. Note that the 3rd-party Python library tsplib95[8] enables parsing files of the TSPLIB library (e.g. TSP problem and solution files).

Additionally, on that class, implement and call the method `initialize_pheromone_map()`, to enable the *initialization of pheromone trails* in the environment based on the Ant System algorithm.

[1]Dorigo, M., & Stützle, T. (2004). Ant colony optimization. Chapter 3.3.1 Ant System. The MIT Press.
[2]Dorigo, M., & Stützle, T. (2004). Ant colony optimization. Chapter 3.1 The Traveling Salesman Problem. The MIT Press.
[3]TSP att48 specification: https://github.com/HSG-WAS-SS23/exercise-7/blob/main/att48-specs/att48.tsp
[4]att48 optimal solution: https://github.com/HSG-WAS-SS23/exercise-7/blob/main/att48-specs/att48.opt.tour
[5]TSPLIB documentation: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf
[6]Example TSP problems and solutions: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
[7]Exercise template: https://github.com/HSG-WAS-SS23/exercise-7
[8]tsplib95 3rd-party Python library: https://pypi.org/project/tsplib95/

Task 1.2: Enable the environment to handle pheromone trails Except for representing the topology of the TSP and enabling the initialization of pheromone trails, the environment should also enable the *addition and evaporation of pheromone trails.*

Implement the method `update_pheromone_map()` of the class `Environment` so that pheromone trails are updated in the environment after each cycle (i.e. after the ants have constructed their tours during the current iteration). The pheromones should be updated based on the evaporation rate $\boldsymbol{\rho}$, as well as based on the ants' movements during the current iteration, according to the Ant System algorithm (see lecture slides).

② Task 2 (4.5 points): Implement the ant colony to apply ACO to a TSP instance

Your second task is to implement the ants of an ant colony that enables the *tour construction phase* of the Ant System algorithm (in ant-cycle mode) and solves the given TSP instance.

Task 2.1: Enable ants to select which locations to visit Artificial ants are situated in the environment, and have the ability to estimate the distance of paths in the environment, as well as select paths towards visiting all locations in the environment.

Implement the method `get_distance()` of the class `Ant`, so that an ant can compute the distance of a path, i.e. the distance between two cities in the environment. The distance should be computed based on the pseudo-euclidean distance algorithm[9], required by the given TSP.

Implement the method `select_path()` of the class `Ant`, so that an ant selects its next path. Path selection should be based on the random proportional rule of the Ant System algorithm (including the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) which was introduced in the lecture.

Task 2.2: Enable ants to visit all locations in the environment Now that the environment infrastructure is available, artificial ants could visit all the possible locations of the environment.

Implement the method `run()` of the class `Ant`, so that an ant selects and visits locations until it has visited all possible locations of its environment. An ant should be responsible for keeping track of the locations it visited (during an iteration), as well as of the distance traveled (during an iteration).

Task 2.3: Solve the TSP instance The ant colony will work towards generating an optimized solution to the TSP instance, based on the Ant System algorithm.

Implement the method `solve()` of the class `AntColony`, so that you produce a solution (`solution`) and the distance traversed for this solution (`shortest_distance`).

③ Task 3 (2 points): Working with Ant Colony Optimization

Your third task is to reflect on different aspects of the Ant Colony Optimization algorithm based on the contents of the lecture as well as your own results from Tasks 1 and 2, where applicable. Reply to the questions that follow in a short report (max. 2 pages of text, the more concise the better).

1.) How do the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ impact the performance of your algorithm (comparing the produced solution to the optimal solution)? Describe and interpret the behavior of your ant colony for different parameter values, while considering that the ant population, the number of iterations, and the evaporation rate remain fixed.

[9]See Section 2.5 Pseudo-Euclidean distance in http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf

2.) How does the evaporation rate $\boldsymbol{\rho}$ affect the performance of your algorithm (comparing the produced solution to the optimal solution)? Describe and interpret the behavior of your ant colony for different evaporation rates, while considering that the ant population, the number of iterations, and the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ remain fixed.

3.) How would you modify your implementation in order to apply the ACO algorithm to a *dynamic* traveling salesman problem (DTSP), i.e a TSP in which cities can be added or removed at run time?

④ **Hand-in Instructions** By the deadline, hand in via Canvas a **zipfile** that contains:

1.) a PDF file that includes the link to the GitHub fork containing **code for Tasks 1 and 2**, and your **report for Task 3**;

2.) any additional instructions for how to run your code (if non-obvious).