Joao Pedro Schmitt

Ant Colony Optimization to solve Travelling Salesman Problem in Java

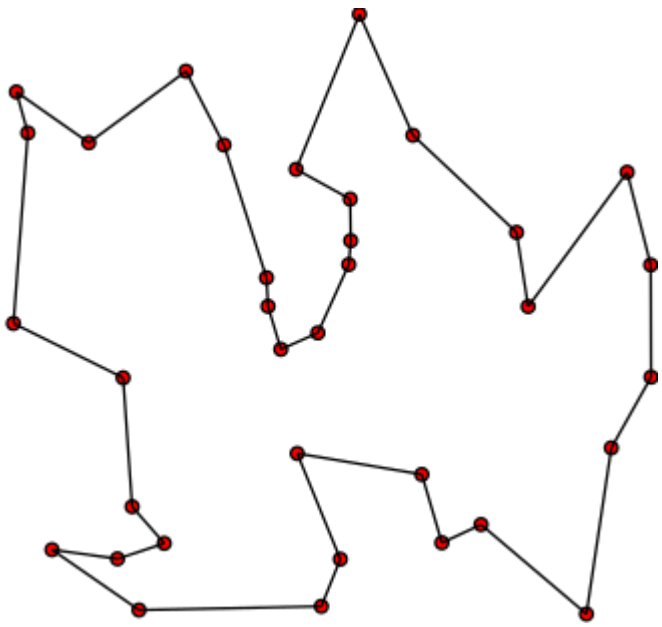
schmittjoaopedro . Data Mining, Java, Statistics 23 23+00:00 Setembro 23+00:00 20174 04+00:00 Novembro 04+00:00 2017 3 Minutes

Travelling Salesman Problem (TSP)

The travelling salesman problem (TSP) try the solve the following problem: “For a given list of cities (vertices) and the distances beteween each pair of cities (edges), what is the shortest route that visits each city only one time and return to the origin city?”. It is an NP-hard problem in combinatorial optimization, important in operations reasearch and theorical computer science.

In the theory of computational complexity the deterministic algorithms that execute all posibilities to try to find the best route needs a long (very long) time to find the solution. The main point is that, the complexity of the problem increase exponentially with the number of cities, so a metaheuristic that do not guarantee the best solution but can found an suboptimal solution is a good alternative to solve the TSP in a acceptable time.

The following image is an example of the TSP problem resolved:

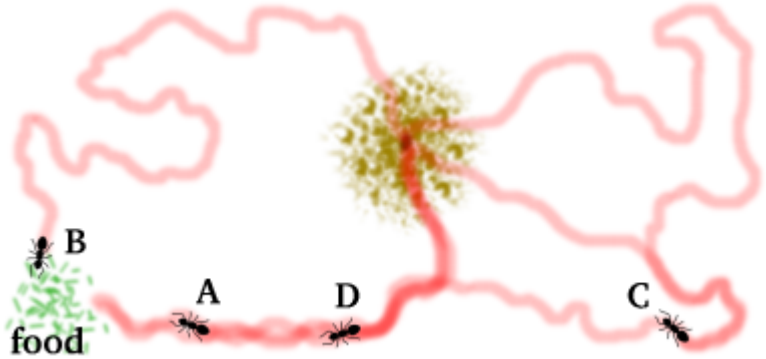


(https://camo.githubusercontent.com/64da9dccb05da2e8208d669eb75d4dc35e4e61ab/68747470733a2f2f75706c6f61642e77696b696d656469612e6f72672f77696b6970656469612f636f6d6d6f6e732f7468756d622f312f31312f474c504b5f736f6c7574696f6e5f6f665f615f74726176656c6c696e675f73616c65736d616e5f70726f626c656d2e7376672f33333070782d474c504b5f736f6c7574696f6e5f6f665f615f74726176656c6c696e675f73616c65736d616e5f70726f626c656d2e7376672e706e67)

(https://github.com/schmittjoaopedro/aco-tsp-java/blob/master/README.md#ant-colony-optimization-aco)Ant Colony Optimization (ACO)

Ant colony optimization is a meta-heuristic used to solve complex discret problems, normally used to search good solutions in graph systems. The principles of ACO are based on the natural behavior of ants, that in their daily life, one of the tasks that ants have to perform is search for food, in the region near of their nest. While ants are searching for food, their deposit on the ground a chemical substance called pheromone used to two main objectives: 1) a way to memorize how to back to the nest and 2) and lets a trail for that the other ants can calculate the next step decisions. The second objective is the most important part to the ants find good optmized path between the nest and the food source. The pheromone laid in the ground by the ants create a specie of network where de edges quality is relative to the amount of the pheromone deposited, this system is used probabilistically to ants takes decisions in the environment.

The communication between ants is made indirect by stigmergy with the environment, if the path has large concentration of pheromone, this is probably due to its shorter length that allowed ants to travel faster, resulting in a large number of travels through the path therefore with much more ants depositing pheromone on it. Furthermore, over time the evaporation on the ground reduce the intensity of path pheromone, the evaporation make ants forgot path with low quality and increase the capacity of the ants to explore new paths. The image below gives an example of the ant system:



(https://camo.githubusercontent.com/4d6348e74666cbc87528dd39609efac4d3c3928e/687474703a2f2f6d7574652d6e65742e736f75726365666f7267652e6e65742f696d616765732f616e74732f616e74696967672616d332e706e67)

(https://github.com/schmittjoaopedro/aco-tsp-java/blob/master/README.md#canonical-ant-colony-optimization-aco-algorithm)Canonical Ant Colony Optimization (ACO) algorithm

Meanwhile, a good number of improvements were inserted in the ants system (AS) with extensions as elitism, ranking, bounding, etc. Nevertheless, the most important development is the description of the Ant Colony Optimization Metaheuristic by Dorigo and Di Caro. The canonical main part of the algorithm is given below:

Algorithm 1 – Pseudo-code for Ant Colony Optimization

```
 Initialize parameters
Initialize pheromone trails
Create ants
while Stopping criteria is not reached do
    Let all ants construct their solution
    Update pheromone trails
    Allow Daemon Actions
end while
```

The detail of the ACO is the Daemon Actions that can perform problem specific operations or centralized operations, which use global knowledge of the solutions. Examples of operations that can be executed in Daemon Actions are: control the feasibility of each solution, give extra pheromone quantity to the best solutions, use some local search routine to improve the solutions, etc.

ACO algorithm have some building blocks that need to be understanded, and in some cases modeled to specific problems to obtain best results. The following building blocks determine the success of the algorithm:

- Method chosen to construct the solutions
- Heuristic information
- Pheromone updating rule
- Transition rule and probability function
- Parameters values
- Termination condition

(https://github.com/schmittjoaopedro/aco-tsp-java/blob/master/README.md#how-to-use)How to use

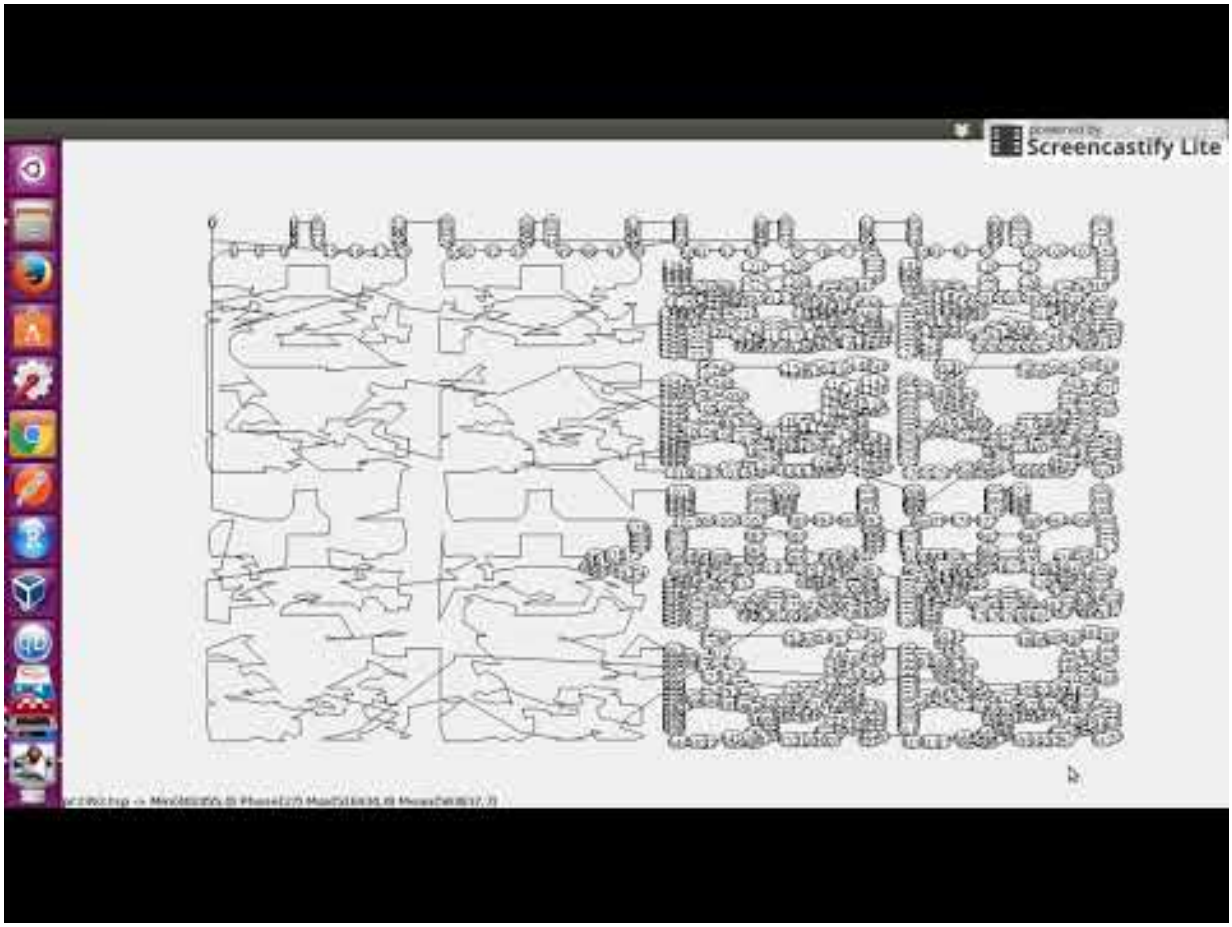
The project was constructed with maven, so to see the code working just open the project in your favorite Java IDE and run the main class (Program.java). Note that the main classe is pointing to the path of the tsp folder inside the repository, so take care for that the path is configured correctly. The source code is at GitHub (https://github.com/schmittjoaopedro/aco-tsp-java).

A little description of the classes of the algorithm are given below:

- Ant: Model an ant, has the current state of the ants (tour information) and has the logic to process the state transition.
- Environment: Model the environment, contains the information about the graph and the pheromone deposited by the ants.
- Parameters: Contains the global parameters used by the ACO
- Program: call the main building blocks of the ACO (Algorithm 1).
- Statistics: calculate the algorithm evolution statistics, ex: best tour found.
- TspReader: create a instance of the TSP problem.
- Visualizer: plot a graph with the vertices and edges showing the best tour found.

(https://github.com/schmittjoaopedro/aco-tsp-java/blob/master/README.md#conclusions)Conclusions

We can conclude that the meta-heuristics are very useful to solve very complex problems with a low cost and time consumption. And even using a very simple implementation of the ACO without any extension cited above, the results were very significant showing a good aproximation of the best solution cost cited in the TSP site.



(http://www.youtube.com/watch?feature=player_embedded&v=q11W1ZERSYk)

(https://github.com/schmittjoaopedro/aco-tsp-java/blob/master/README.md#references)References

TSP Library (https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/)

Travelling Salmesman Problem (https://en.wikipedia.org/wiki/Travelling_salesman_problem)

Ant Colony Optimization: a literature survey (http://wps.fep.up.pt/wps/wp474.pdf)

Book Ant Colony Optimization (https://mitpress.mit.edu/books/ant-colony-optimization)

Aco-metaheuristic site (http://www.aco-metaheuristic.org/)

Com as etiquetas :
ACO,
Ant Colony Optimization,
Data Mining,
ia,
java,
metaheuristic,
Programmation,
Software,

technology,
Travelling Salesman Problem,
TSP

Publicado por schmittjoaopedro

Graduado como bacharel em Sistemas de Informação pelo Centro Universitário Católica de Santa Catarina campus Jaraguá do Sul. Formado no Ensino Médio pelo Senai com Técnico em Redes de Computadores Articulado. Atualmente desenvolvedor JEE/Web em Sistemas de Engenharia na WEG. Pesquisador no período de faculdade em Informática pela Católica de Santa Catarina. Contato 47 - 99615 2305 E-mail: schmittjoaopedro@gmail.com Web page: https://joaoschmitt.wordpress.com/ Linkedin: https://www.linkedin.com/in/joao-pedro-schmitt-60847470/ Curriculum lattes: http://lattes.cnpq.br/9304236291664423 Twitter: @JooPedroSchmitt Ver todos os artigos de schmittjoaopedro

□