ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Solving the Travelling Salesman Problem by Using an Articficial Ant Colony

Raphaela Wagner & Giandrin Barandun

Zurich
May 2014

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Raphaela Wagner

Giandrin Barandun

Contents

1 Abstract

2 Individual contributions

2.1 Raphaela Wagner

With the aim of achieving a good model for solving the travelling salesman problem by the use of artificial ants Raphaela helped the group understanding the underlying paper and the included model. She contributed a great amount of explanations and ideas how to approach the whole project.
In a second step she took care of how to implement the raw data from the TSP-library into MATLAB and transform it to a usable form. Further more she coded the functions "eta.m", "global_pheromene_update.m", "test_funktionen.m" and "update.m" and helped improving and correcting the main program.
After the code was written she did a lot of testing with different problem sets and compared the solution of the program to known solutions.

When the group got stuck and did not see a way out of a specific problem she was the one to bring along a hot chocolate and cheer the group up again.

2.2 Giandrin Barandun

The paper which is thought to be reconstructed on the following pages was selected and suggested to the group by Giandrin and during the whole process he tried to have some influence on the project with his wide technical understanding of the problem. The codes for the functions "prob_dist.m", "calc_Lnn.m", "choose_city.m", "main_initialize_system.m", "coordinates.m" and "calc_dist.m" are his contributions as well as the collaboration on the main program. He searched the internet for known TS-problems and their solutions and put all data in a readable form. A lion's share for the program working at the end was his bug fixing in all the functions and programs and combining them to the running model.

3 Introduction and Motivations

Observing crawling ants how they manage to find a shortest path from a food source to their nest arises the question of how to model such a biological phenomena. It is known that the way ants organize their transporting system is based on a secreted chemical called pheromone. While ants move on a track they deposit a certain amount of pheromone. Since real ants prefer choosing lines of a high pheromone concentration, this messenger ensures that ants follow their members on a certain trail. To illustrate the effect of pheromone on an ant trail consider Figure 1. Real

ants follow a path between a food source and their nest (Fig. 1 A). Placing an obstacle on the trail forces the ants to find a way of restoring the interrupted track (Fig. 1 B). One expects half of the ants to turn right and half of them to turn left. In the beginning both ways around the obstacle are enriched with approximately the same amount of pheromone (Fig. 1 C). Since ants that have chosen the shortest path need less time to pass by the obstacle the number of ants per time is bigger compared to those who have chosen the longer path. Consequently the shorter path contains a higher concentration of deposited pheromone than the longer one. This follows from the assumption that all ants secrete the same amount of pheromone and move approximately at the same speed.
After a certain time more ants prefer the shorter path containing more pheromone until the longer one is completely neglected to circumvent the obstacle (Fig. 1 D).

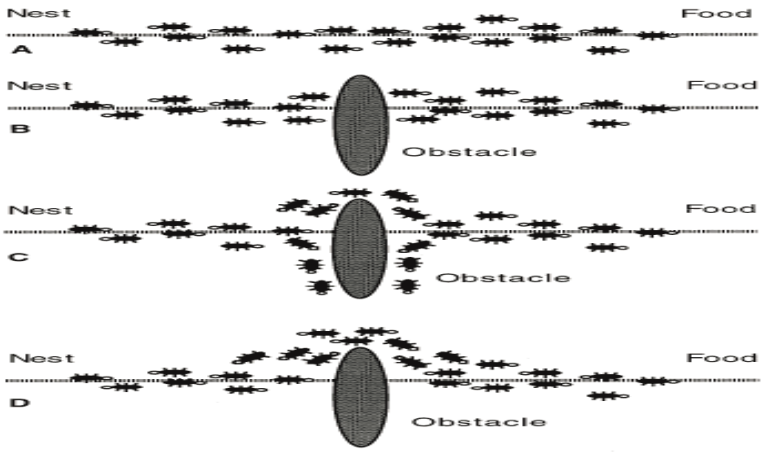

Figure 1: (A): Ants following a trail between food source and their nest. (B): An obstacle is placed to interrupt the track of the ants. (C): The column of ants splits into two groups each choosing a different way to circumvent the obstacle. (D): Due to the higher concentration of pheromone all ants have chosen the shortest path.

Consulting the literature **?** one finds an interesting paper which uses an ant colony system (ACS) to solve a traveling sales man problem (TSP). Artificial ants, also called agents are successively moving on a TSP graph between different cities. In the course of this they are following the constraint to visit each city once and return to their starting point. After all ants have completed their tour, the shortest one is rewarded by increasing the weight of the according tracks. This corresponds to a higher concentration of pheromone on the chosen tour.
The goal of this project is to implement the given model (see chapter 4 on page 6)

from **?** and to calculate the shortest tour for different city environments. Those are obtained from the TSPLIB (*http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/*) and correspond to the data used in the reference paper **?**. The aim is to figure out whether our code is able to produce the same length of the shortest tour or not. Next to that, the variation of parameters in the model is analysed. Precisely, these simulations try to answer questions like: How does the shortest tour length depend on the rewarding, i.e. on the amount of pheromone deposited? How fast is the decay in the shortest tour as a function of completed rounds? Moreover, the influence of the number of agents on the time needed to find the shortest tour is investigated.

4 Description of the Model

In this model a number of k ants is sent on a network of m cities with every ant starting at the same city. The next ant only starts when the previous ant has finished its tour which means there never are two ants in the network.

4.1 Choosing a City

Before moving on to the next city an ant has to decide where it wants to go. For this purpose it chooses randomly a number q between zero and one and if this number is smaller or equal than a certain parameter q_0 ($q \leqslant q_0$) looks for the city s which fulfils the following formula:

$$s = \arg\max\{[\tau(r,u)] \cdot [\eta(r,u)]^\beta\}, \qquad u \notin M_k. \tag{1}$$

The current city where the ant stays is denoted by r and only cities can be chosen which are not yet in the ants memory M_k which means the ant has not visited these cities. The matrix τ stores the information about the amount of pheromone on the edge between city r and city u and the function η gives the inverse of the distance between the two cities.
If the random number is bigger than q_0 ($q > q_0$) then the ant randomly chooses one of the remaining unvisited cities and accepts or rejects it according to the probability p_k:

$$p_k(r,s) = \frac{[\tau(r,s)] \cdot [\eta(r,s)]^\beta}{\sum_{u \notin M_k} [\tau(r,u)] \cdot [\eta(r,u)]^\beta} \tag{2}$$

This probability basically contains the same formula of τ and η as the one above but is now normalized with the sum over these relations of every unvisited city. One can clearly see that in the beginning the sum is big and the probabilities are small but favouring the edges with more pheromone and lower distance. At the end when

only one city is left the sum equals the term in the nominator and the probability becomes one for the last remaining city.

4.2 Moving Forward and Updating

Once the city has been chosen the ant moves along the edge and the pheromone on the path is updated according to:

$$\tau(r,s) = (1-\alpha)\cdot\tau(r,s) + \alpha\tau_0 \tag{3}$$

The newly introduced parameters α and τ_0 are explained in chapter 6. This update reduces the amount of pheromone on the chosen edge and helps avoiding very strong edges which would be taken by all the ants.
At the time the first ant has finished the tour the second one can start while the first ant still keeps in mind the trajectory of its tour which means the sequence of the city it has visited and the length of its tour but deletes its memory such that it is ready for a new tour. When all ants have completed one tour the shortest one is rewarded with pheromone according to the formula:

$$\tau(r,s) = (1-\alpha)\cdot\tau(r,s) + \alpha\Delta\tau(r,s) \tag{4}$$

This update is intended to give the edges along the shortest path a little head start in the following round. With this step the first round is complete and the second round can start.

5 Implementation

5.1 Main program

In the previous section the theoretical understanding of the model was tried to be imparted to the reader. Following in this section is an overview of how the model was implemented in MATLAB.
To start with a main program (*main_initialize_system.m*) was written which reads the data of a specific TSP and puts it into an upper triangle matrix form which contains the distances between the cities as elements. Further more all parameters like number of rounds and number of contributing ants (agents) can be adjusted in this main program. At the end it activates the function *main_main.m* with the purpose to find the shortest route.
In the main function there are two for-loops, one for the number of rounds and one for the number of ants (agents). For every agent we have a memory of the visited cities (*M_k*)and a trajectory vector which saves the sequence of how the cities where

visited. In the first step every ants chooses the next city with help of the function *choose_city.m* which is described below. The city is added to the memory of the ant and the trajectory vector and the pheromone on the edge is reduced according to the formula on page 75 in the paper. As long as there are unvisited cities this procedure is repeated for the agents and at the end the way to the start city is updated (pheromone, trajectory) and the memory reset. Then the second agent starts his tour.
After every agent has finished his route the shortest path is detected and the edges along this route are rewarded with pheromone with help of the trajectory vector and the function *global_pheromone_update.m*. At the end of this step the trajectory vector is reset for all ants and the second round can be initialized. After every round the shortest route of the current round is compared to the overall shortest route found until now and then kept or rejected depending on the result of the comparison.
When all rounds have been calculated the function gives the shortest path found in any of the rounds.

5.2 Choose the Next City

To choose which city the ant will go next two different methods are implemented according to the paper and one of them is selected based on a certain probability.
The first way to decide which city to go next is to optimize a function which depends on the amount of pheromone on the edge between the current city and the chosen one and its length. This method chooses the edge with the highest amount of pheromone and the shortest length whereas there are parameters to weight these two variables relatively to each other.
On the other hand a probability was assigned to every unvisited city again depending on the amount of pheromone and the length of the edge between the cities. Then a city was randomly chosen and accepted or rejected with its assigned probability.
At the end the function *choose_city.m* gives the number of the next city to visit back to the main function.

6 Simulation Results and Discussion

6.1 Length of shortest tour

In a first attempt the dependency of the shortest tour length on different parameters is analysed. Thereby the number of completed rounds each agent finishes is varied. Moreover the relative weighting of deposited pheromone and closeness is analysed by changing the parameter β and the value of the update rate α is optimized.

The shortest tour length is calculated for a 30 and 51 cities problem (called *oliver30* and *eil51*) and averaged over 10 trials. The distances between the cities are determined using the given coordinates on a two dimensional euclidean plane. The distance $d_{\mathrm{i,j}}$ between city i and j is then calculated by

$$d = sqrt((x_i - x_j)^2 + (y_i - y_j)^2). \tag{5}$$

For the thirty cities problem *oliver30* the distances are rounded to the next integers. For *eil51* double distances are used.

The trajectory vector of the shortest tour is defined as the sequence of city numbers which yields the optimal tour length. For the city environment *oliver30* the solution found in **?** is given by [13456789101112131415161718192021222325242627282930 2]. Using equation (5) the total tour length is calculated to shortest $\mathrm{path}_{\mathrm{solution}} = 420$. Comparing it to the sequence obtained from own simulations one observes a slightly different sequence: [12345678910111213141516171819202122325242627282930] where the agents don't move from city 30 to 2 but to 1. However, the length of the shortest path also results in shortest $\mathrm{path}_{\mathrm{simulation}} = 420$. Obviously, there exist two solution to this symmetric TSP using rounded distances.

In *eil51* the trajectory vector found using the model from **?** blabalbal. The length is balbalba shortest $\mathrm{path}_{\mathrm{solution}} = 426$.

In Figure 2 and 3 the dependency of the shortest path length on the number of completed tours is shown. As expected the average length decreases asymptotically towards the value of the optimal path. For both city environments the program achieved to reach the averaged shortest path known from literature **?** and **?**. For thirty cities and 400 rounds, <shortest path> is in fact equal the best result found, whereas for 51 cities a lot more rounds would have been needed to achieve this. Anyhow, the best result obtained in a single run is $<$ shortest path $>= 429.48 \pm 1.86$.
In Figure 2 three models have been implemented and tested. The first (white circle dots) correspond exactly to the one described in **?** where all agents are moving at the same time step. The red squares represent the result obtained by a slightly modified model. The same mathematics has been used but the agents are exploring a new trail one by one. As can be seen, this change has little effect on the value of the shortest tour length and its error.
Adapting the pheromone update equations (**??**) and (4) on page 6 by adding a constant of 0.1, results in a steeper fall of the average shortest path. At 2000 rounds the final value does not yet reach the averaged shortest path length from **?**. One concludes that the added constant was chosen too high such that the global and local update only had a neglecting influence on $\tau(r, s)$.

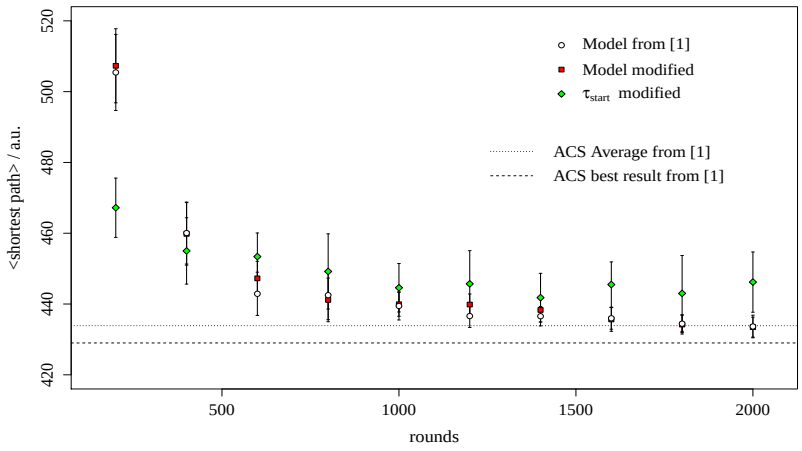

Figure 2: Shortest tour length averaged over 10 runs for different numbers of rounds. The value approximates the optimal solution for $\approx$ 1200 rounds.

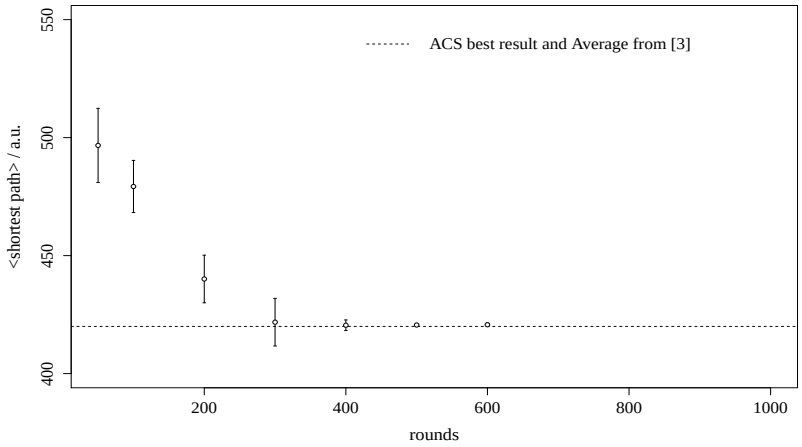

Figure 3: Shortest tour length averaged over 10 runs for different numbers of rounds. The value approximates the optimal solution for $\approx$ 400 rounds.

Figure 4 illustrates that the averaged shortest path length is optimized for α between 0.1 and 0.3. This is the region where the length of the tour and also the error of the shortest path is minimized. The reason why the shortest path is highest close zero and one, can be understood considering the pheromone update formulas (**??**) and (4) on page 6. For $\alpha = 0$ the local and global pheromone concentration

stays constant, no update is made. Hence the cities are only favored by closeness and not by the pheromone concentration anymore. On the other hand, if α is close to one the amount of pheromone after the local update has changed a lot. Randomly chosen trails which deviate from the shortest path will be weighted too much. Therefore the optimal update rate α is expected to be closer to zero.

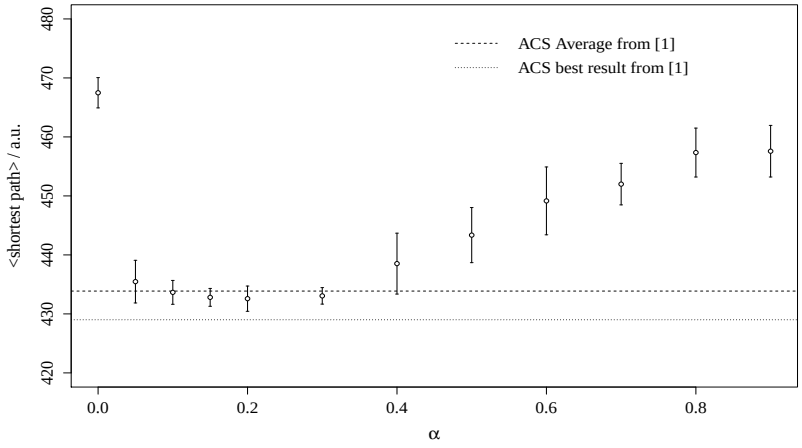

Figure 4: The update rate parameter α changes the averaged shortest tour length. The simulation is done for a fifty city problem using ten agents which all completed 2000 rounds. Other parameters are set to $beta = 2$, $q_0 = 0.9$ and $tau_{??} = 0.1$. The values are averaged over ten runs. By setting $\alpha = 0.1$ the length of the averaged shortest path is comparable to the value found in literature **?** and the change in pheromone update is held low.

For the parameter β a similar analysis has been done for a thirty city environment and ten agents. The motivation for setting $\beta = 2$ in **?** is barely understood from figure 5. The decrease of the error for higher values of β is explained by the fact that the criteria of closeness between the cities is weighted stronger than the pheromone concentration (see equation (2) and (1)). Thus, the exploration of new paths is suppressed and the deviation from the averaged path length is small. In the region of $\beta \approx 2$ the bigger error bars indicate that the relative importance of pheromone and closeness is chosen such that the ants are forced to try new trails. The probability of getting stuck in a local minimum is reduced.

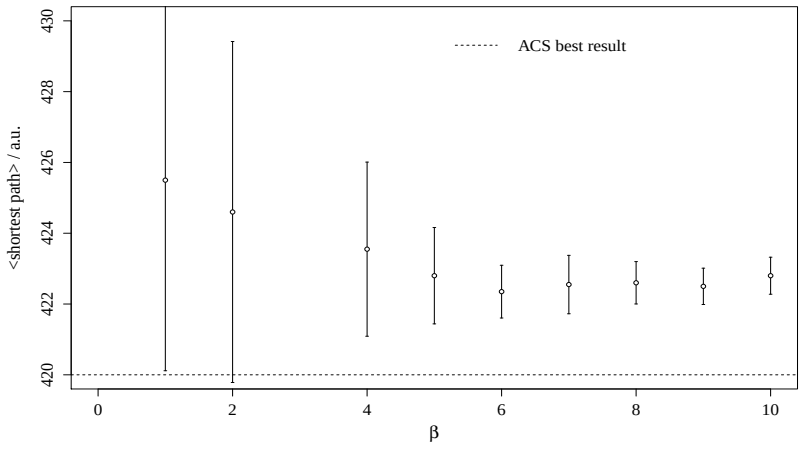

Figure 5: Variation of the parameter β for the oliver30 using ten ants going 400 rounds averaged over 20 runs. The parameters are chosen as $\alpha = 0.1$,$q_0 = 0.9$ and $tau_{??} = 0.1$.

6.2 Model adaptions

7 Summary and Outlook

8 References