



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 Artificial 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



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

SOLVING THE TRAVELLING SALESMAN PROBLEM
BY USING AN ARTIFICIAL ANT COLONY

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):

WAGNER
BARANDUN

First name(s):

RAPHAELA
GIANDRIN

With my signature I confirm that

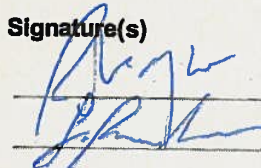
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

ZURICH, 13.05.14

Signature(s)



For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

1	Abstract	1
2	Individual contributions	1
2.1	Raphaëla Wagner	1
2.2	Giandrin Barandun	1
3	Introduction and Motivations	2
4	Description of the Model	3
4.1	Choosing a City	3
4.2	Moving Forward and Updating	4
4.3	Modifications	4
5	Implementation	5
5.1	Main program	5
5.2	Choose the Next City	6
6	Simulation Results and Discussion	6
6.1	Length of shortest tour	7
6.2	Influence of the Parameters α and β	11
6.3	Influence of the parameter q_0	12
7	Summary and Outlook	13
8	References	15

1 Abstract

On the following pages a model which solves the travelling salesman problem (TSP) by using an artificial ant colony (ASC) is reconstructed from a paper and explained to the reader. A few modifications to the model were made and the dependence of the model on different parameter was tested. The optimal solution to the TSP was found for all tested problems (30 and 51 city problems) after a reasonable number of rounds (500-2000) with this simple model.

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_agents_together.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).

Consulting the literature [1] one finds an interesting paper which uses an ant colony system (ACS) to solve a traveling sales man problem (TSP). Artificial ants, 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 3) from [1] and to calculate the shortest tour for different city environments. Those are obtained from the TSPLIB [2] and correspond to the data used in the reference paper [1]. 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.

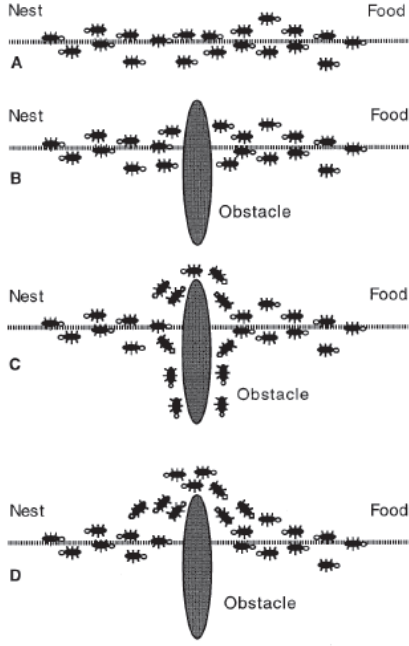


Figure 1: (A): Ants following a trail between food source and their nest. (B): An obstacle is placed to interrupt the track of the ants. (C): The column of ants splits into two groups each choosing a different way to circumvent the obstacle. (D): Due to the higher concentration of pheromone all ants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

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 ($q \leq q_0$) it looks for the city s which fulfils the following formula:

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

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 ($q > q_0$) the ant randomly chooses one of the remaining unvisited cities and accepts or rejects it according to the probability p_k :

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

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:

$$\tau(r, s) = (1 - \alpha) \cdot \tau(r, s) + \alpha \tau_0. \quad (3)$$

The newly introduced parameters α and τ_0 are explained in chapter 6.3. 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:

$$\tau(r, s) = (1 - \alpha) \cdot \tau(r, s) + \alpha \Delta \tau(r, s). \quad (4)$$

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.

4.3 Modifications

In the above model every ant started at city number 1 and waited for his predecessor to finish his tour completely before it started its own tour. In the paper [1] underlying this report the model was slightly different. All agents are placed randomly on any city in the network and move together. When the first agent chose his second city and moved there he waits until every ant has found its next city. So in every time

step all agents move for one city. The difference to the model before is that every ant feels the pheromone on the edges of all the other ants in the network. Whereas in the previous model the second ant only saw the trace the first ant left but did not feel any influence of the still following ants.

Another small change which was tested is to increase the award for the shortest path which was found after one round. This means to add a constant amount of pheromone to each edge along the shortest path in the global update formula (equation 4).

$$\tau(r, s) = (1 - \alpha) \cdot \tau(r, s) + \alpha \Delta \tau(r, s) + \mathbf{0.1}. \quad (5)$$

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_agents_together.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. Furthermore 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_agents_together.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 a memory of the visited cities (*M_k*) and a trajectory vector which saves the sequence of how the cities where visited was implemented. In the first step every ant 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

The shortest tour length is calculated in different *rounds* for a 30 and 51 cities problem (called *oliver30* and *eil51* respectively) and averaged over 10 *trials*. Whereas one *round* means that every agent *once* walked through all cities in the network and for the next *round* the agents start with the update pheromone on their tours and one *trial* means that all agents have completed a certain amount of *rounds* and found a shortest path. For the next *trial* everything is set to the initial conditions and at the end the average of the shortest paths found in the different trials is taken. The error bars in the following plots always refer to the standard deviation from those repeated measurements.

The distances between the cities are determined using the given coordinates on a two dimensional euclidean plane. The distance $d_{i,j}$ between city i and j is then calculated by (*calc_Lnn.m*)

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \quad (6)$$

For *oliver30* the distances are rounded to the next integer values, whereas in *eil51* double distances are used. This leads to slightly different numbers for the shortest tour length.

6.1 Length of shortest tour

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 [3] is given by [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 24, 26, 27, 28, 29, 30, 2]. Using equation (6) the total tour length is calculated to $shortest\ path_{solution} = 420$. Comparing it to the sequence obtained from own simulations one observes a somewhat different sequence: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 24, 26, 27, 28, 29, 30] where the agents do not move from city 30 to 2 but to 1. However, the length of this shortest path also results in $shortest\ path_{simulation} = 420$. Obviously, there exist two solutions to this symmetric TSP using rounded distances.

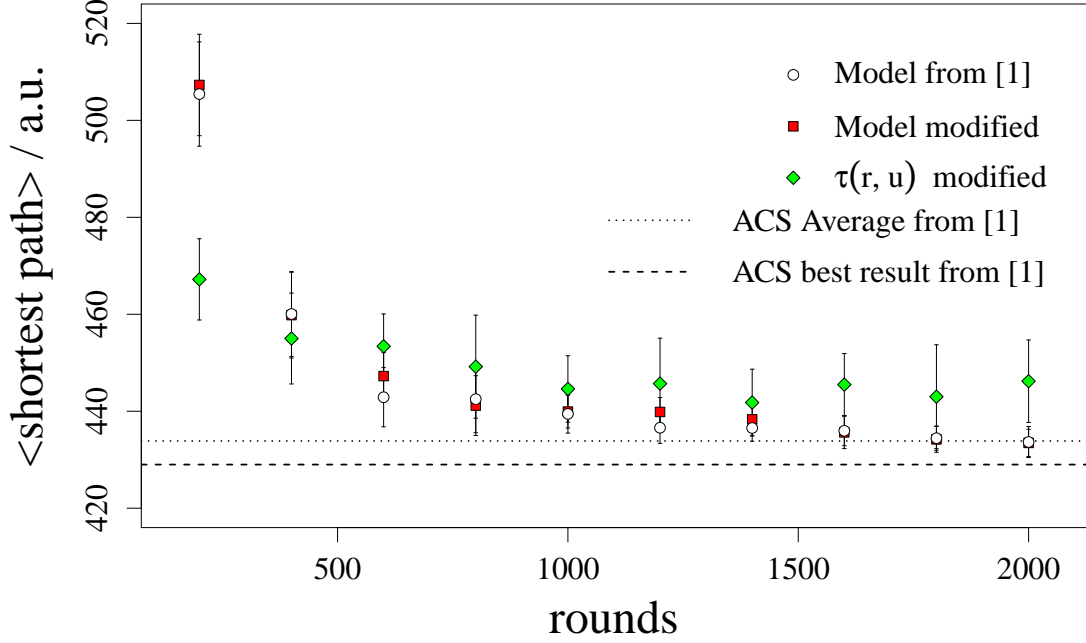


Figure 2: Shortest tour length averaged over 10 trials for different numbers of rounds and 10 agents with the corresponding standard deviation as error bars for problem *eil51*. Thereby two slightly different implementations of the model from literature [1] are done (white circle dots and red squares, see text). The green diamonds correspond to a model where the update formula $\tau(r, s)$ is changed (equation 5). The values approximate the optimal solution for ≈ 1200 rounds. The two different lines represent the best shortest tour and the average value measured according to [1]. The parameters are chosen as follows $\alpha = 0.1$, $\beta = 2$, $q_0 = 0.9$, $\tau_{\text{start}} = 0.1$.

In Figure 2 and 3 the dependency of the shortest path length on the number of completed rounds is shown. As expected the average length decreases asymptotically towards the value of the optimal path. For *oliver30* $\langle \text{shortest path} \rangle$ is in fact equal to the best result found using 400 rounds, whereas in *eil51* a lot more rounds would have been needed to achieve this. Anyhow, the best result obtained for *eil51* in a single run is $\text{shortest path}_{\text{simulation}} = 429.48$. Comparing the ACS average path length and best yielded result to values from literature ([1] and [3]) one concludes that the implementation of the model was a full success (table 1). For both problems the optimal tour length is found and the average value in *eil51* is similar to the one from literature¹.

¹Unfortunately there are no confidential intervals given in [1].

	ACS average [1] and [3]	ACS best result [1] and [3]	ACS average simulation	ACS best result simulation
<i>eil51</i>	433.87	429	433.7 ± 3.2	429.48
<i>oli30</i>	(N/A)	420	420.5 ± 0.5	420

Table 1: Simulation results of two city environments compared to values from literature ([1] and [3]). Own simulations: *eil51* the agents completed 2000 rounds, for *oli30* 400. Literature: *eil51* the agents completed 500 rounds, for *oli30* there is no average shortest path length determined.

In Figure 2 three models have been implemented and tested. The first (white circle dots) correspond exactly to the one described in [1] where all agents are moving at the same time step. The red squares represent the result obtained by a slightly modified implementation. The same mathematics has been used but the agents are exploring a new trail one by one (see chapter 4). As can be seen, this change has little effect on the value of the shortest tour length and its error.

Adapting the pheromone update equation (4) on page 4 by adding a constant of 0.1 (as described in chapter 4) does not result in better values for the shortest path. On the contrary it seems this model will not easily find the shortest path at a reasonable number of rounds. One concludes that this type of rewarding the shortest path is quite likely to end in a local minimum where it cannot get out due to the local shortest path being rewarded more and more after every round.

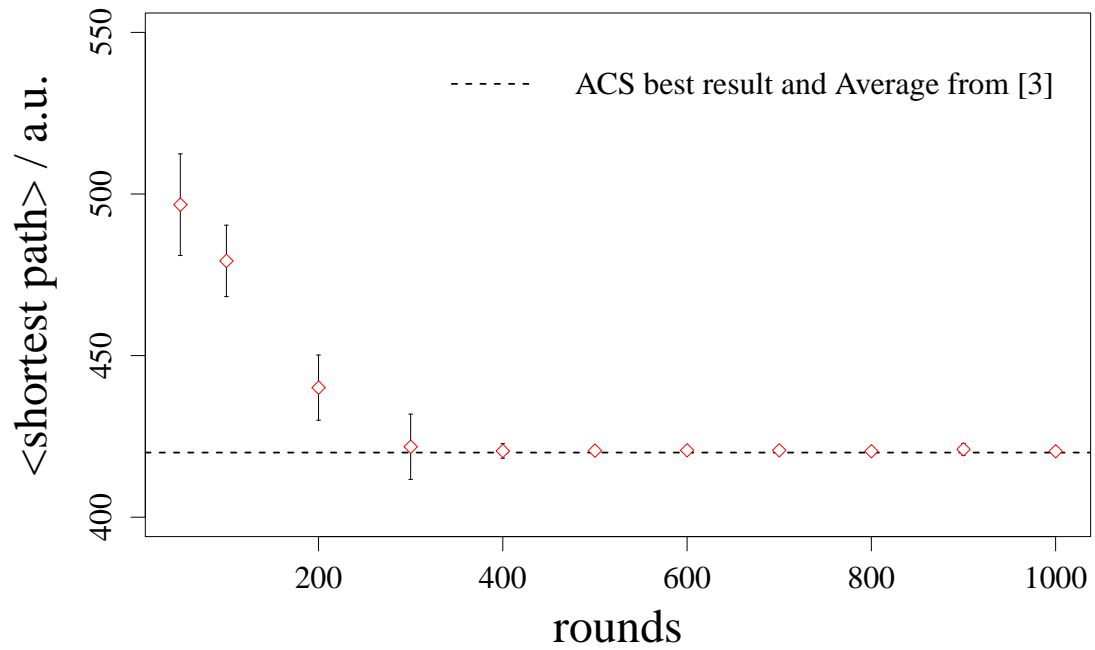


Figure 3: Shortest tour length averaged over 10 trials for different numbers of rounds and 10 agents with the corresponding standard deviation as error bars for problem *olvier30*. The value approximates the optimal solution at ≈ 400 rounds. For this simulation the model and its implementation are done in the same way as in [1].

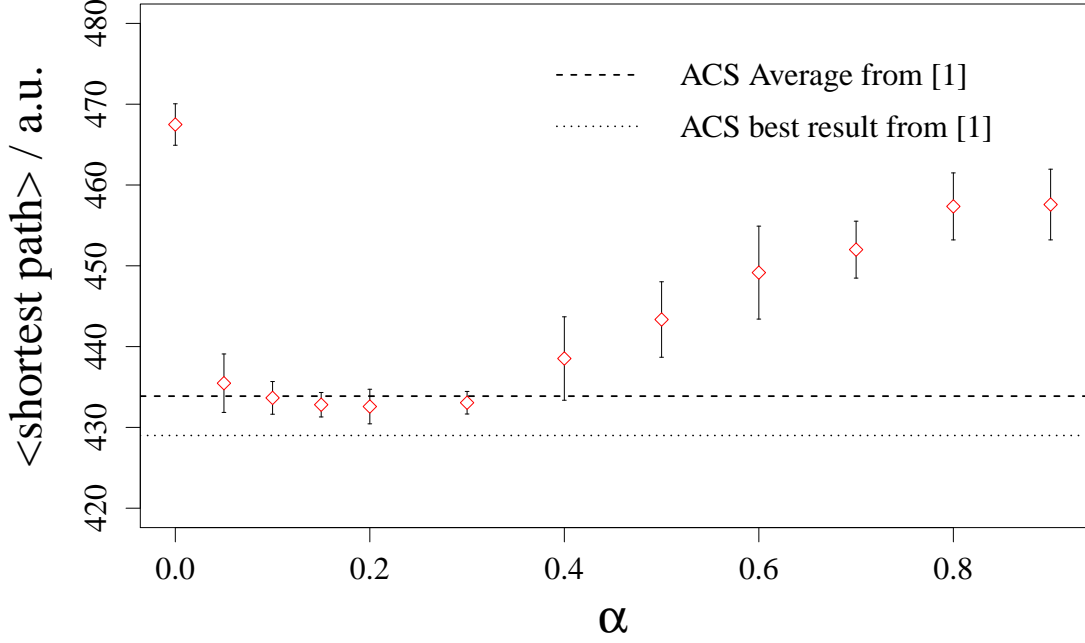


Figure 4: The update rate parameter α changes the averaged shortest tour length. The simulation is done for a 51-city problem using ten agents which all completed 2000 rounds. Other parameters are set to $\beta = 2$, $q_0 = 0.9$ and $\tau_{start} = 0.1$. The values are averaged over ten trials. By setting $\alpha = 0.1$ the length of the averaged shortest path is comparable to the value found in literature [1] and the change in pheromone update is held low.

6.2 Influence of the Parameters α and β

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 (3) and (4) on page 4. For $\alpha = 0$ the local and global pheromone concentration stays constant, no update is made. Hence the cities are only favoured by closeness and not by the pheromone concentration any more. 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.

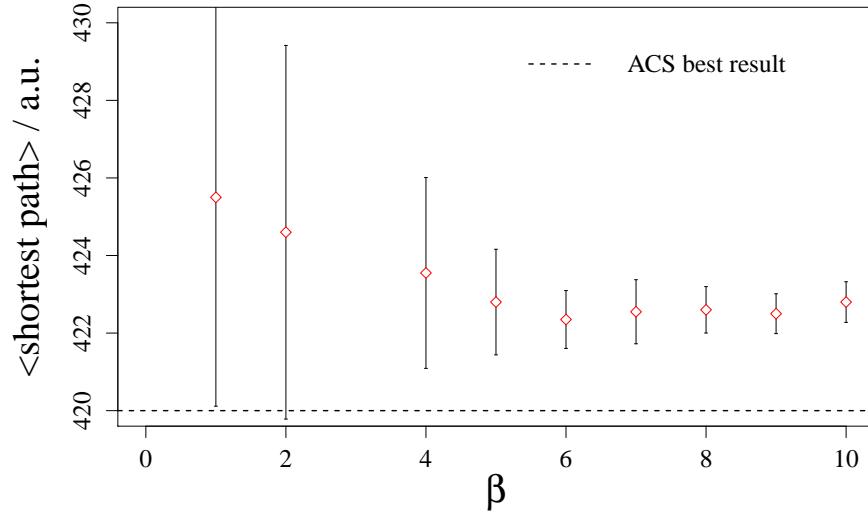


Figure 5: Variation of the parameter β for the oliver30 using ten ants going 400 rounds averaged over 20 trials. The parameters are chosen as $\alpha = 0.1$, $q_0 = 0.9$ and $\tau_{\text{start}} = 0.1$.

For the parameter β a similar analysis has been done for a 30-city environment and ten agents. 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.

6.3 Influence of the parameter q_0

In Figure 6 one can observe the influence of q_0 on the shortest path found after 800 rounds averaged over 10 trials for the problem *oliver30*. This parameter is a measure of how random the next city is chosen (see chapter 4). If the model acts with a small q_0 the following cities are mostly chosen randomly and do not depend strongly on the amount of pheromone on the edge or the length of the edge. As one can see a q_0 below 0.6 leads to a path which is a few units longer than the known solution and the error gets bigger as well. Whereas q_0 's in the range of 0.8 to 0.95 result in a average path which lies on the known solution and the error is vanishingly small. The problem with a $q_0 = 1$ is the fact that the ants can get stuck in a strong local minimum. The 'local' shortest path gets strongly rewarded and the edges get stronger and stronger

and the ants never see a reason to change their path. Therefore one needs a certain probability that one 'crazy' ant tries some new path which can lead to a new shortest path found.

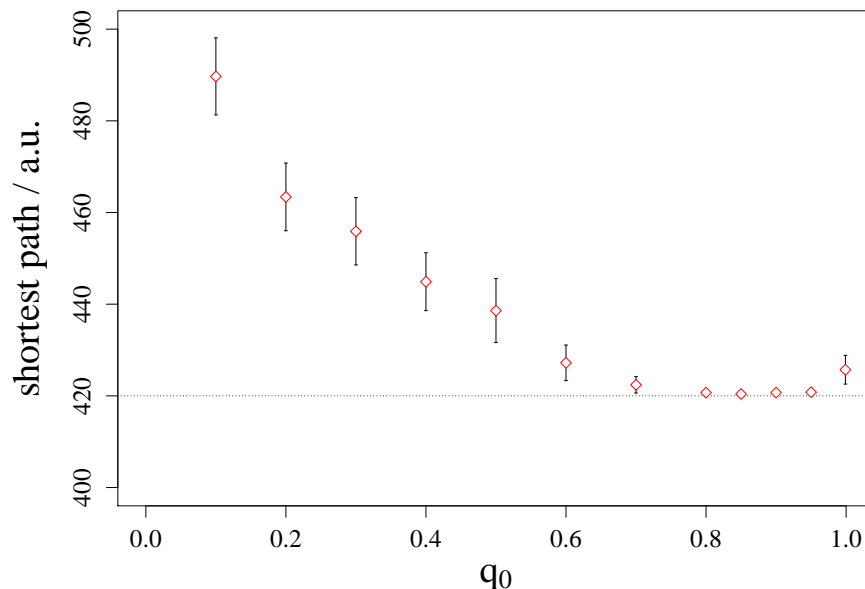


Figure 6: The shortest path the simulation found after 800 rounds for the problem *oliver30* is plotted against the probability q_0 (see text for impact of q_0). The optimal path is shown as a dotted line along with error bars of the data. If no error bars are visible then the errors are smaller then the extent of the points.

In the appendix one can find two more graphs which investigate the behaviour of the model when changing the number of agents or the initial amount of pheromone on the edges. Both were either due to a high standard deviation useless or were not flabbergasting at all.

7 Summary and Outlook

The first aim of this project, namely the reproduction of the results from [1] was achieved. For two different city problems the shortest path averaged over ten or twenty trials and the best shortest tour ever reached correspond to the solutions known from [1], [2], [3]. Using two slightly different implementations of the same

program do not show a significant difference in the obtained results.

The second goal set was the dependency of the averaged shortest path on various parameters such as α , β , q_0 and τ_{start} . On one hand the analysis motivated chosen parameter values in [1] and on the other hand it manifested the independence of the shortest path (for example see 7 in the appendix on page 15). Moreover the obtained results revealed interesting characteristics of the model and helped for deeper understanding.

An investigation on the time the program needs to find the shortest path for various number of agents did not show a significant correlation. Therefore the impact of the agent number on the computation time would be an interesting topic to focus in a next project. Besides, for further investigations on TSP containing more nodes (cities) one should definitely optimize the MATLAB code for faster calculation.

8 References

- [1] *Ant Colonies for the Travelling Salesman Problem*; Marco Dorigo, Luca Maria Gambardella; Lugano, Switzerland; 24.10.1996
- [2] <https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>
- [3] <http://stevedower.id.au/blog/research/oliver-30/>

Appendix

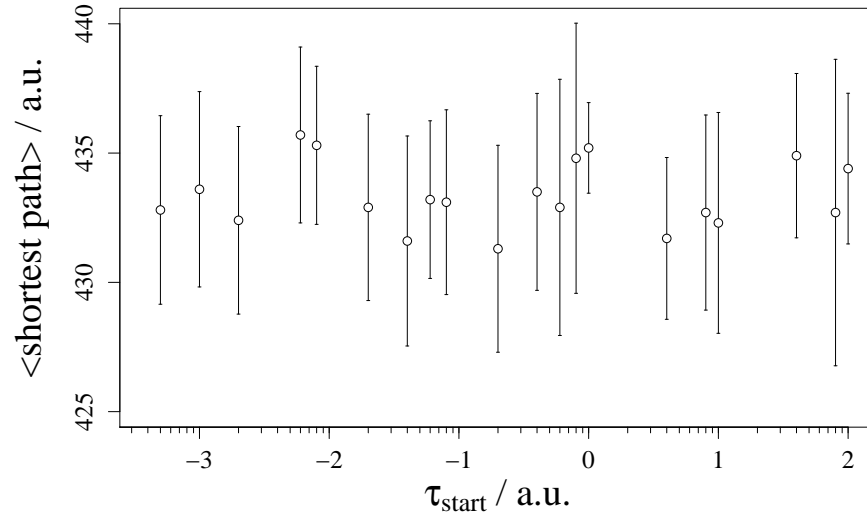


Figure 7: The shortest path the simulation found after 1200 rounds averaged over 10 trials for the problem *eil51* is plotted against the initial tau put in the pheromone matrix. The figure shows no influence of this τ_{start} on the shortest path the simulation found neither on the error.

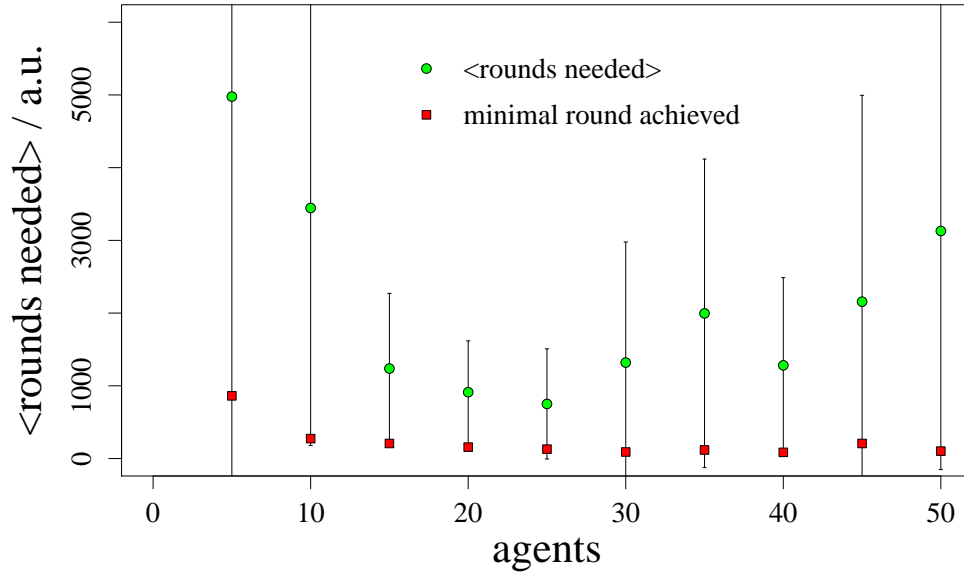


Figure 8: To determine after how many rounds the shortest path is definitely found the program is adapted such that the for-loop over the number of rounds is replaced by a while loop. When the optimal tour length, known from previous calculations, is achieved three times in series the while loop is left. This plot shows the number of rounds needed to fulfil this criteria for different ant colony sizes. The values are averaged over ten trials and show a large spread. The bigger the error bars the more possible it is for agents to get stuck in a local minimal which does not correspond to the optimal tour. But due to the big uncertainties no direct correlation between the average time needed to find the shortest path and the number of agents can be made. However, the smallest number of rounds needed to find the solution in ten trials decreases for more than five agents.