Project 2 Heuristic Optimization: ACO for TSP

Rober Schäfer, Willi Müller, 27.05.2015

Introduction

In this work we present the measured runtimes of our Ant Colony Optimization implementation in Python 3.4, run with pypy3, on symmetric traveling salesperson (TSP) problem instances of 17 and 26 cities respectively (*gr17* and *fri26*).
We executed our program on a MacBook Air with Intel Core i5 with 4 logical cores (two cores with hyperthreading) clocked at up to 2.8GHz in TurboBoost and 8GB RAM, running OSX 10.9.5.
In the following we discuss the runtime of the algorithm while variating the problem size and the parameters rho, α, and β. For each parameter set we ran the algorithm 10 times and therefore present 10 groups of points as well as a curve representing the average of the 10 individual curves, computed by the mean of all values lying in 10 equally sized ranges of the horizontal axis (marked in axis on top). Each run is plotted in a different color leading to a palette of 10 colors.

Meaning of parameters in context of exploitation vs. exploration

According to the recommended paper by Stützle and Hoos [1], the parameter rho represents the persistence of the trail (therefore 1 – rho models the evaporation). Given the formula on the exercise slide, we translate the meaning of parameters alpha and beta as the balance between heuristic information and pheromone value.

All together, alpha, beta and rho give a statement about the ratio of exploitation vs. exploitation. Exploitation in this sense means how strongly an algorithm keeps up with the best solutions so far, exploration how often the algorithm gets off the beaten track.

Besides that, the parameter beta defines how much the algorithm should tend towards a local nearest neighbor heuristic.

Since the domain of the pheromone function “tau” lies between zero and one, a high value for alpha and therefore a high potency decreases the importance of the pheromones. The lower the value for alpha the stronger the algorithm relies on pheromones and thus it favours exploitation.

Because the rho denotes the persistence of the pheromone trail, the algorithm starts to “forget” former best solutions more quickly in case of a high rho value. We see a fast adoption of a better solution as exploitation.

Eventually, a high beta value results in a higher importance of the weighting function. It works in the opposite direction, if a node is far away and the weighting function yields a high value, the node is less likely to be chosen as the next hop. For this reason the beta as the exponent is negated. Having that said, a high beta leads to a local nearest neighbor heuristic.

Favoring heuristic information over pheromone values

Rho=0.1

fri26

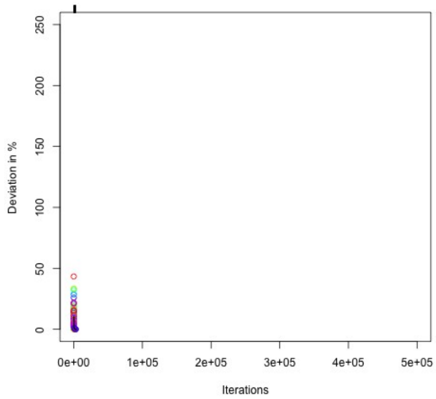

Figure 1: alpha=0.1, beta=8

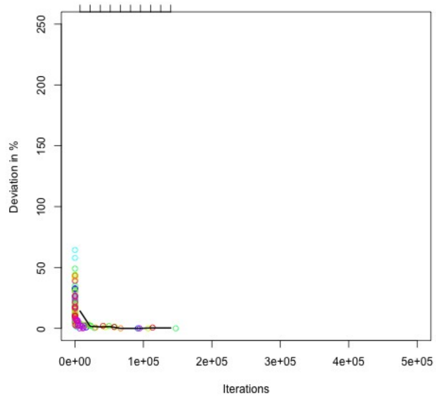

Figure 2: alpha=1, beta=8

gr17

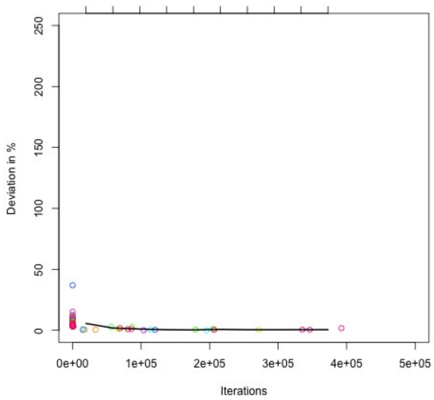

Figure 4: alpha=0.1, beta=8

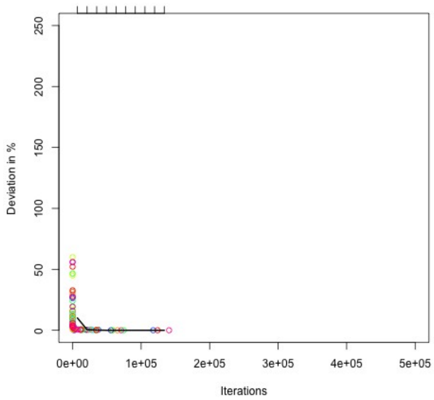

Figure 3: alpha=1, beta=4

Rho=0.5

fri26

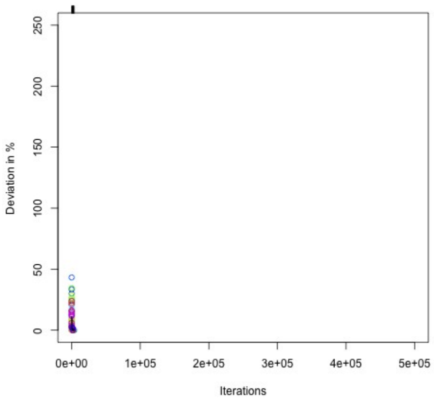

Figure 5: alpha=0.1, beta=8

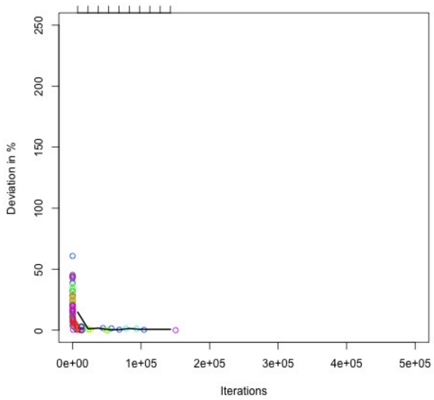

Figure 6: alpha=1, beta=4

gr17

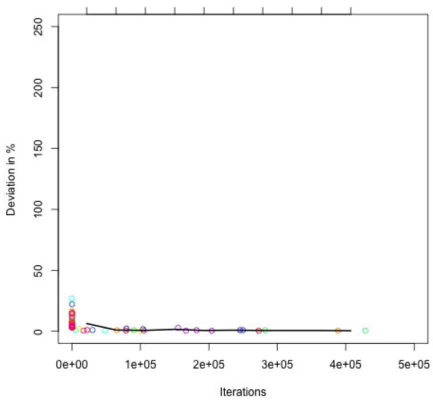

Figure 8: alpha=0.1, beta=8

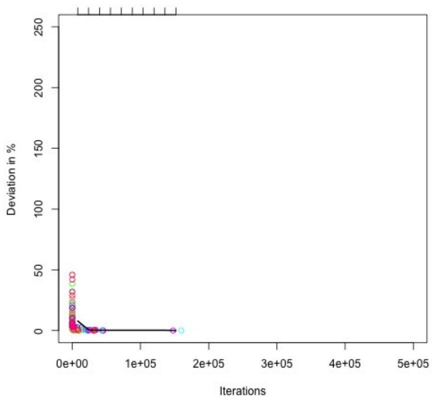

Figure 7: alpha=1, beta=4

Rho=0.8

fri26

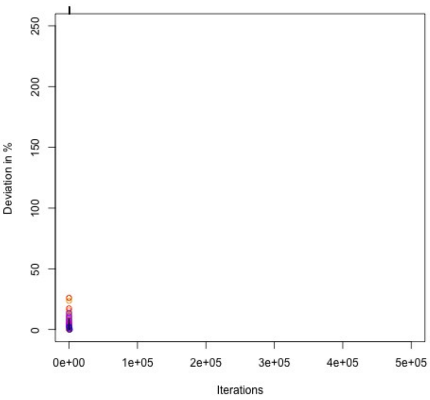

Figure 9: alpha=0.1, beta=8

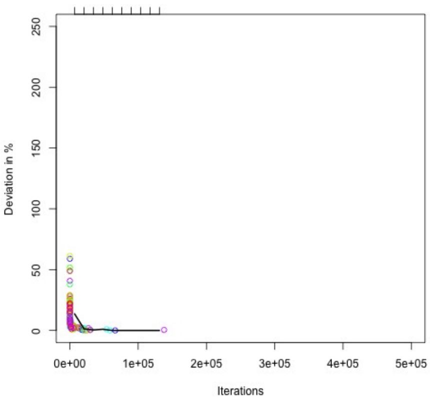

Figure 10: alpha=1, beta=4

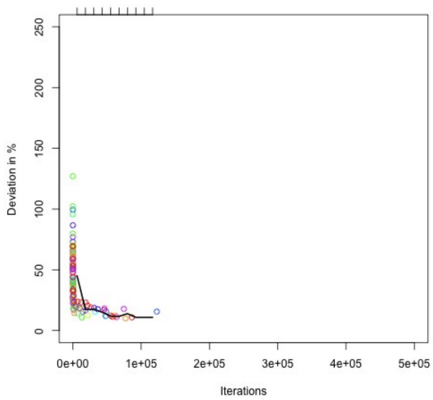

Figure 11: alpha=1, beta=2

gr17

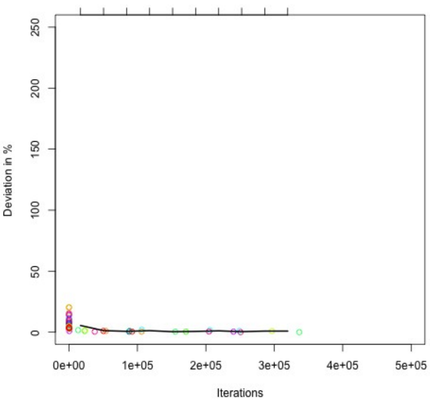

Figure 12: alpha=0.1, beta=8

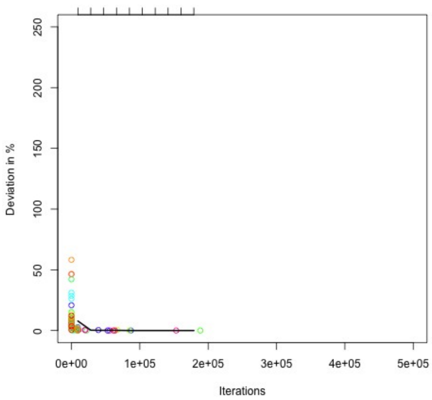

Figure 13: alpha=1, beta=4

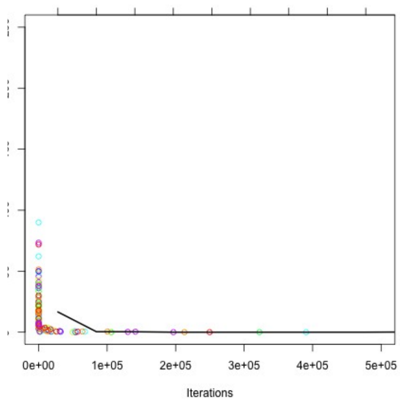

Figure 14: alpha=1, beta=2

Equally weighting pheromones and heuristic information

fri26

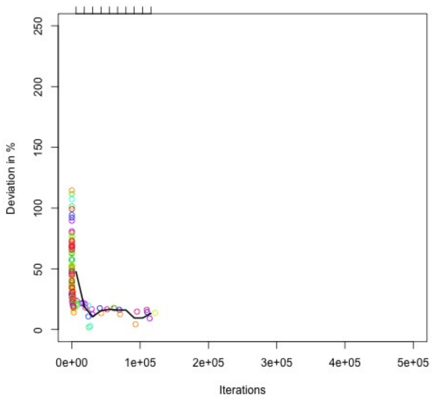

Figure 15: alpha=2, beta=2, rho=0.1

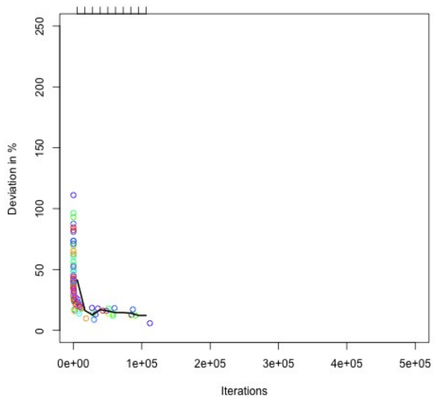

Figure 16: alpha=2, beta=2, rho=0.5

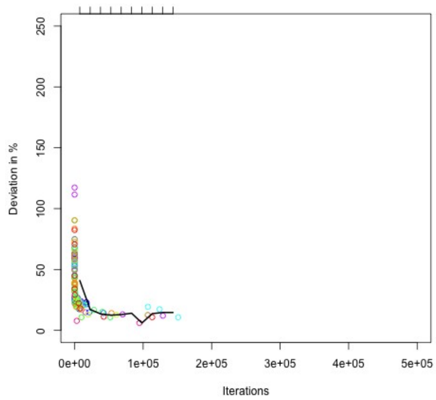

Figure 17: alpha=2, beta=2, rho=0.8

gr17

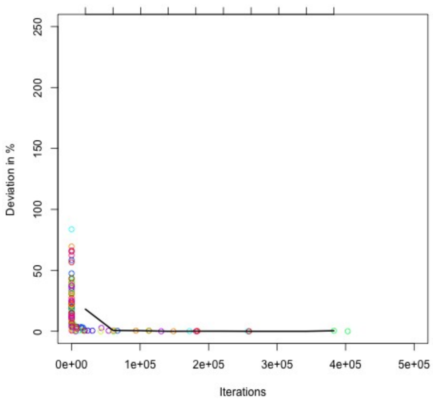

Figure 18: alpha=2, beta=2, rho=0.1

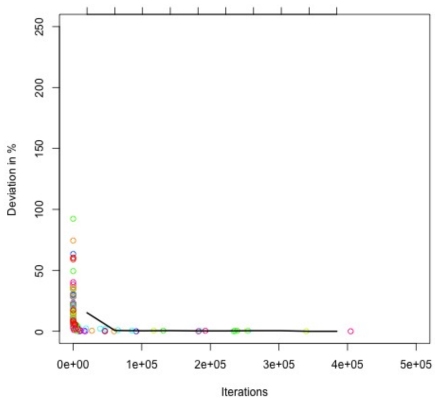

Figure 19: alpha=2, beta=2, rho=0.5

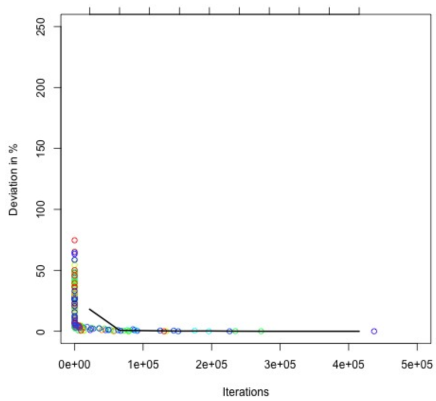

Figure 20: alpha=2, beta=2, rho=0.8

Favoring pheromone values over heuristic information

fri26

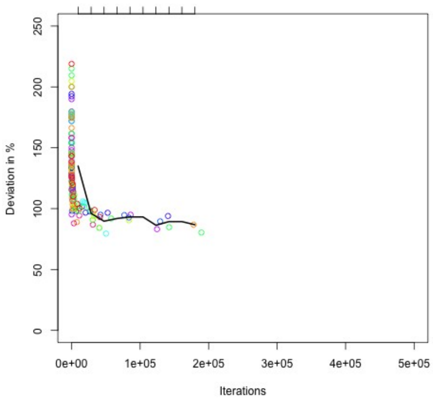

Figure 23: alpha=3, beta=0, rho=0.1

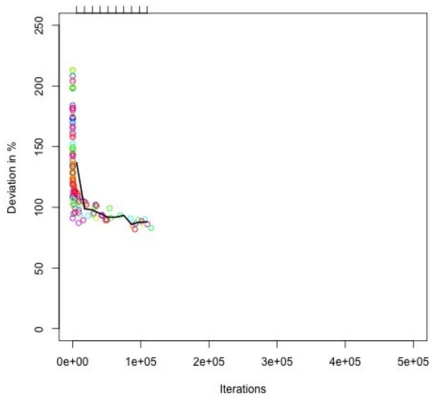

Figure 21: alpha=3, beta=0, rho=0.5

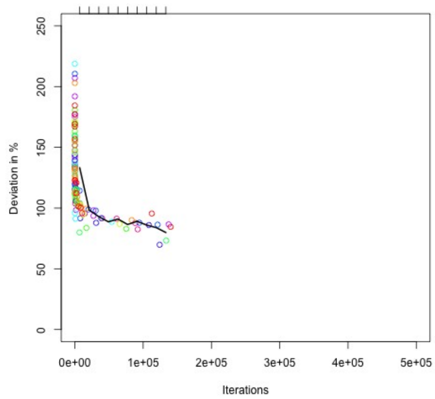

Figure 22: alpha=3, beta=0, rho=0.8

gr17

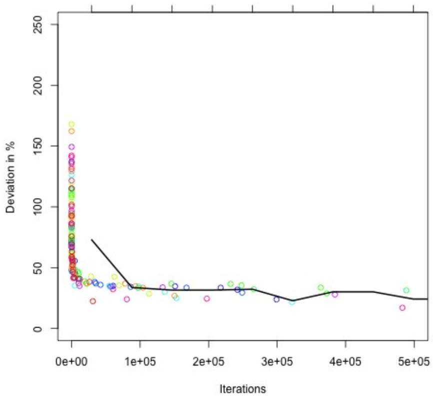

Figure 24: alpha=3, beta=0, rho=0.1

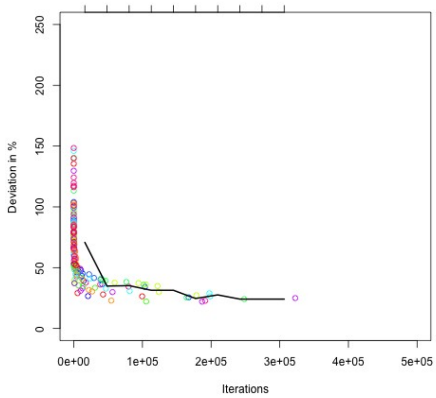

Figure 25: alpha=3, beta=0, rho=0.5

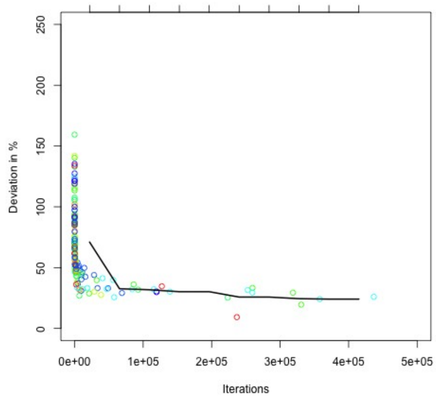

Figure 26: alpha=3, beta=0, rho=0.8

Discussion of Effect of Parameters and Problem Size

Strongly weighting heuristic information leads to fastest convergence
It can be observed that the algorithm gets better with a high value for beta compared to alpha. In the Figures 1-14 can be seen that beta=8 and alpha=0.1 converges more quickly than beta=2 and alpha=1). This holds true independently of the value of rho as can be seen by comparing Figures with the same alpha and beta for the same problem but with varying rho as can be seen e.g. in the Figures 1, 5, and 8 or the Figures 2, 6, and 10 respectively. Although a greater rho slows the convergence process down, the general trend of higher ratio meaning faster convergence is not changed.
However, the problem size has an interesting effect: In Figure 11 it is visible, that for the larger problem *fri26* the algorithm is no longer able to find a nearly optimal solution. There was no run out of 10, which found a solution at most 15% more costly than the optimum.

Equally weighting heuristic information and pheromones leads to slower convergence for larger rho
In Figure 15-20, especially Figure 15-17 as they show the harder problem *fri26*, is is visible that the temporarily best-known solution of the algorithms converges slower to the optimum as rho increases. This might be due to the fact that poor routes which have been chosen are more likely to be chosen again as the pheromones evaporate less quickly for greater rho. These poor routes are also more likely to be chosen in comparison to other scenarios, because the heuristic information of the edge weight beta is considered not important enough in this parameter setting.

Favoring pheromones over heuristic information leads to poor performance
It is notable that the initial solutions are the worst of all parameter settings, especially for larger problems. For the problem *fri26*, the initial solutions lie up to 230% above the optimal solution and average around 140% above the optimum in the first 1400 iterations (see Figure 21-23). But even after 10 minutes no run succeeded to find a solution at most 60% above the optimum.
The same effects can be observed for the smaller problem *gr17* but on a different scale: the average of the best found solutions lie around 30% above the optimum with outliers ranging within 10% (see Figure 24-26).

Smaller problem gr17 almost always converges more slowly than fri26
Another observation is that the smaller problem *gr17* seems to converge more slowly than the larger one *fri26* as is indicated by comparing the Figures 1 and 4, 5, 8, as well as 9 and 12. This effect even continues for equal alpha and beta, both set to 2, and for a larger alpha than beta.

Smaller problem gr17 continue to improve slightly over longer time then fri26
However, it should be noted that the solutions for the smaller problem are usually more optimal after less iterations and continue to improve slightly over a longer timespan, whereas for the larger problems the algorithms is not able to improve found solutions after less iterations. This effect is nearly always visible when comparing the Figures in the first (*fri26)* and second lines (*gr17)* of a section. When halting the search after having found a solution, let's say, no worse than at most 10% above the optimum, this effect would only still be observable in Figures 21-26, where alpha is higher than beta and the solutions are poor even for the small problem *gr17*.

Parameter setting chosen by Stützle and Hoos [1] does not perform best
In the experiments run by Stützle and Hoos [1] were chosen alpha=1, beta=2, rho=0.8. This setting did not perform best in our experiments. In our measurements for the problems we chose, which are smaller than the ones discussed in [1], a larger beta turned out to lead to faster convergence and more optimal solutions. This becomes clear when comparing the more desirable curves for the harder problem *fri26* in

Figure 1, 5, and 9, where alpha=0.1 and beta=8 with Figure 11, which shows the same parameter set as in the publication by Stützle and Hoos.

Exploitation is fairly good but a simple nearest neighbor heuristic gives the quickest performance boost
It is no big surprise that a nearest neighbor heuristic would give a quick drop in execution time compared with a random search. This is reflected in our discussion results. Also, we get a another fair improvement if we don't stick too much to a former best solution wich is obviously worse than a current better solution. Our explanation is: If it was hard to make a decent path even better, it's unlikely that there are many better paths close to it.

Bonus Points

Swiss42 Problem (42 nodes):

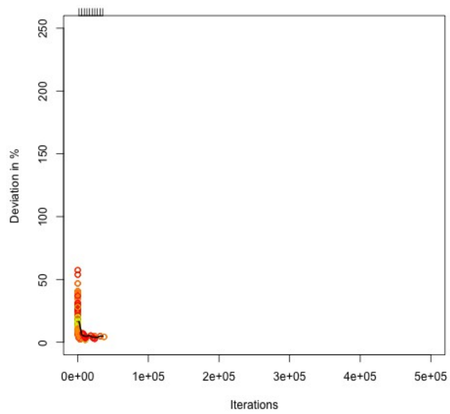

Figure 27: Swiss42 alpha=0.1 beta=8.0 rho=0.1

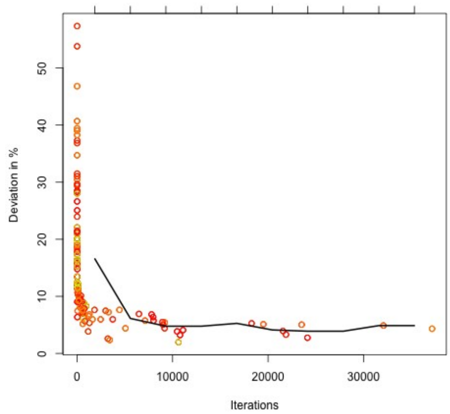

Figure 28: Swiss42(scaled) alpha=0.1 beta=8.0 rho=0.1

Two out of our four workers finished within 10 minutes.
Figure 8 shows the performance on this tsp instance along with figure 8 which has a more convenient axis scaling. We used the following parameters:
rho = 0.1, alpha = 0.1 and beta = 8.

References

[1] Thomas Stützle and Holger H. Hoos. 2000. MAX-MIN Ant system. *Future Gener. Comput. Syst.* 16, 9 (June 2000), 889-914.