Optimisation Method Project

Markus Reiter, Josef Gugglberger & Michael Kaltschmid

July 6, 2021

This document provides some documentation and analysis of the project that is part of the course *Optimization Methods*. We implemented the *Firefly* algorithm [1] as well as the *Artificial Bee Colony* [2] algorithm. In the following we will briefly describe both algorithms and show some performance comparisons to other optimisation methods.

1 Firefly Algorithm

The firefly algorithm is a nature-inspired, meta-heuristic optimisation algorithm. The algorithm, presented by Yang et al., is derived from the flashing behaviour of fireflies. We will not go into biological details of this phenomena, but instead will focus on the simplified set of rules that artificial fireflies follow, as presented in [1]. Firstly, any firefly is attracted by any other firefly. Secondly, the attraction depends on the brightness of the fireflies, and lastly, the attraction decreases with increasing distance. Each firefly represents one solution to the optimisation problem. We initialise the fireflies randomly over the search space. The brightness of a firefly is determined by the firefly's error (or its reciprocal, depending if we seek for maximisation or minimisation). At each iteration, each firefly moves a little step into the direction of each other firefly that has a higher brightness. The formula of the movement is given as:

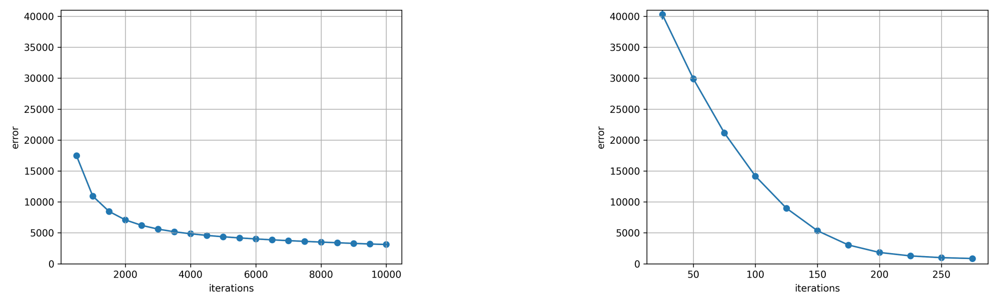
$$\boldsymbol{x_i} = \boldsymbol{x_i} + \beta_0 e^{-\gamma {r_{ij}}^2} + \alpha(rand - \frac{1}{2}) \tag{1}$$

where $\boldsymbol{x_i}$ is the current position, β_0 is the attractiveness at distance 0, γ is the absorption coefficient, r_{ij} is the distance between the two fireflies, α is a randomisation parameter, and *rand* a random number between 0 and 1.

1.1 Experiments

In this section, we compare our implementation of the firefly algorithm to the reference evolutionary algorithm on the DTLZ1 problem. For our tests we initialised 50 fireflies and optimise over 275 iterations. We used following parameters for all experiments: $\alpha = 0.001$, $\beta_0 = 1$, and $\gamma = 0.05$.

As table table 1 and figure figure 1 show, the firefly algorithm outperforms the reference evolutionary algorithm on the DTLZ1 problem, even with a smaller number of evaluations and a shorter run-time. We adapted the number of iterations we used in the firefly algorithm, such that the number of evaluations is roughly the same as in the reference algorithm, to allow for a fair comparison. We averaged the results of 10 subsequent runs and give the standard deviation in table 1. The number of evaluations done in the firefly algorithm is also averaged over 10 runs, because as opposed to the reference algorithm, the number of evaluations is not constant in each iteration.



(a) Error over iterations using the reference evolutionary algorithm.

(b) Error over iterations over using the firefly algorithm.

Figure 1: Comparison of firefly and evolutionary algorithm on the DTLZ1 problem.

Optimiser	**Error ± Deviation**	**Evaluations**	**Run-time [s]**
Firefly Algorithm	781.63 ± 42.28	227912	8.39
Reference Evolutionary Algorithm	3003.58 ± 87.35	250000	24.53

Table 1: Minimal errors after the given number of evaluations produced by optimisers on the DTLZ1 problem. Results are averaged over 10 subsequent runs.

2 Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm is a nature-inspired, meta-heuristic optimisation algorithm. It was introduced by Karaboga [2] and is motivated by the behaviour of honey bees.

The model defines three components: employed bees, unemployed bees, and food sources. Employed bees search for new food sources by looking in the neighbourhood of other food sources they have memorised. They show the gathered information to the unemployed onlooker bees which get recruited by a probability dependent on the food source quality, called fitness. Employed bees that are not able to improve the quality of their food source multiple times (i.e. hit the abandonment limit) will become unemployed scout bees search for a new food source randomly.

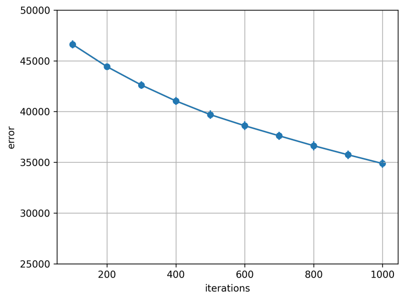
Since we are seeking to solve the Travelling Salesman Problem (TSP), each food source is defined as a route of cities. Each city is defined by a x and a y coordinate. In the TSP we aim to get the optimal order of cities, which leads to a minimal travelling distance. We introduce problem specific knowledge to the algorithm by defining its neighbourhood search. Based on the idea from Li et al. [3], we obtain the neighbour of a route by swapping the order of some cities. Each city will get swapped with a certain probability, and it will get swapped with a randomly chosen city. In this case, the fitness of a food source is the distance of a route of cities.

2.1 Experiments

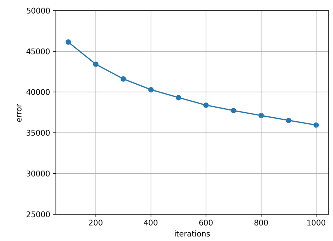
In this section, we compare our implementation of the Artificial Bee Colony algorithm to the reference evolutionary algorithm on the TSP problem.

In table 2 and figure 2 we see the ABC algorithm performed slightly worse than the reference evolutionary algorithm. As the standard deviations in table 2 show, the results of the ABC algorithm where more stable when compared to the EA.

For our tests, we initialised 40 bees and optimise over 1000 iterations. We set the randomisation factor $\alpha = 0.005$ and the abandonment limit to 200. The two approaches are very comparable in terms of their run-time, although the bee colony algorithm performs a higher number of evaluations over the same amount of iterations.



(a) Error over iterations using the reference evolutionary algorithm.



(b) Error over iterations using the bee colony algorithm.

Figure 2: Comparison of ABC and evolutionary algorithm on the TSP problem.

Optimiser	**Error ± Deviation**	**Evaluations**	**Run-time [s]**
Bee Colony Algorithm	35939.18 ± 144.21	80040	2.93
Reference Evolutionary Algorithm	35012.41 ± 482.07	25000	2.90

Table 2: Minimal errors after the given number of evaluations produced by optimisers on the TSP problem. Results are averaged over 10 subsequent runs.

3 Conclusion

In this project we successfully implemented two nature-inspired optimisation algorithms, namely the Firefly algorithm and the Artificial Bee Colony algorithm. We evaluated the Firefly algorithm on the DTLZ1 problem, while the ABC algorithm was used to solve the Travelling Salesman problem. Both algorithms were compared to a given reference algorithm and we were able to achieve similar performance on both problems. In the case of the Firefly algorithm on the DTLZ1 problem, our approach even outperformed the Evolutionary algorithm. The Artificial Bee Colony algorithm achieved comparable results on the much harder Travelling Salesman problem.

References

[1] Xin-She Yang and Adam Slowik. "Firefly algorithm". In: *Swarm Intelligence Algorithms*. CRC Press, 2020, pp. 163–174.

[2] scholarpedia. 2010. URL: `http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm` (visited on 07/04/2021).

[3] Li Li et al. "A discrete artificial bee colony algorithm for TSP problem". In: *International Conference on Intelligent Computing*. Springer. 2011, pp. 566–573.