Swarm Intelligence for Solving a Traveling Salesman Problem

Isabel Kuehner

German Aerospace Center (DLR) Oberpfaffenhofen
Institute of Communications and Navigation
Wessling, Germany
Baden-Wuerttemberg Cooperative State University
Mannheim, Germany
Email: isabel.kuehner@dlr.de

***Abstract*—Learning from the social behavior of animals, like bees or ants, opens the field for Swarm Intelligence (SI) algorithms. They can be applied to solve optimization problems, like the Traveling Salesman Problem (TSP). For SI algorithms, each member of the swarm benefits from the whole swarm and the whole swarm benefits from each individual member. The members communicate either directly or indirectly with each other in order to find an optimal solution. This paper presents an overview of three state-of-the-art SI algorithms, namely, the Ant Colony Optimization (ACO), the Particle Swarm Optimization (PSO), and the Bee Colony Optimization (BCO) for solving a TSP. All three algorithms have been implemented and tested. They have been evaluated with respect to the balance between exploration and exploitation.**

***Keywords*—Swarm Intelligence; Traveling Salesman Problem; Ant Colony Optimization; Particle Swarm Optimization; Bee Colony Optimization.**

I. INTRODUCTION

Many animal species work and live together in swarms. Insects find their optimal way to a food source by communicating with each other and working together. This observed behavior can be applied to optimization problems, e.g., the Traveling Salesman Problem (TSP). Algorithms have been developed, which simulate the swarm behavior of animals. Such algorithms are categorized as Swarm Intelligence (SI) algorithms.

All SI algorithms have in common that they have to create a balance between exploration and exploitation [1]. Exploration means finding new solutions for a problem. For the TSP, this is realized by creating new paths. Exploitation means the use of currently good solutions, i.e., the use of the best path known at the moment. If the swarm focuses on exploitation, it converges quickly towards a non-optimal solution. Therefore, both aspects, exploration and exploitation need to be balanced [1].

Insects in particular, such as ants, have a great influence on the development of SI algorithms. Their social interactions are a role model for an algorithm called Ant Colony Optimization (ACO) [2]. This algorithm is based on the food searching process of ants by leaving a pheromone trail on their path. Another widespread algorithm in the field of SI is the Particle Swarm Optimization (PSO). It is based on observations of bird flocks and the social interaction between each member of the flock [3]. The third algorithm presented throughout the paper, the Bee Colony Optimization (BCO) algorithm [4], has its origin in the foraging behavior of bees. Honeybees fan out searching for food and communicate their discoveries to the other bees after returning to the hive by means of dancing. All three algorithms have in common that each member of the swarm calculates a solution for the problem. This solution is then compared to the whole swarm or to the direct neighbors of the member. The comparison is either done directly or with indirect communication. The bees' waggle dance is an example for direct communication, whereas ants communicate indirectly by leaving pheromone trails.

The aim of this paper is to introduce the topic of SI, to present the three aforementioned techniques, and to evaluate if they are applicable to solve discrete optimization problems, e.g., the TSP. Furthermore, the importance of exploration and exploitation is highlighted and evaluated. The TSP describes a salesman who wants to visit a specific number of cities and tries to find the shortest way to connect those cities. He wants to visit every city only once. The city where he starts is, moreover, his destination. Throughout this paper, cities are called nodes and the connections between cities are referred to as edges. The edges have different lengths.

All three algorithms have been implemented and tested for the TSP. The experimental results only give an idea how the problem can be solved by the SI algorithms. Those examples are not optimized and better solutions may be possible. In contrast to [5], the experiments focus on the balance between exploration and exploitation. For the evaluation of performance and time efficiency, see [5].

The paper is structured as follows. The ACO algorithm is explained in Section II. Section III focuses on the PSO algorithm. In Section IV, the BCO is presented. The results obtained when applying each of the presented algorithms to a TSP are shown in Section V. The work is concluded in Section VI.

 ISBN: 978-1-61208-765-8

II. Ant Colony Optimization (ACO)

The ACO algorithm is based on the food searching process of ants. While moving, each ant leaves a pheromone trail on its path [6]. Ants communicate indirectly with the other members of the swarm, as information is mediated by the environment. This form of communications is known as stigmergy communication [7]. In the beginning, each individual ant chooses its way randomly. When there are, for instance, two possible ways to a potential food source, one shorter than the other, both paths have the same probability to be chosen. The ants selecting the short path reach the food source earlier than the others. The ants leave pheromones on their path to the food source. They take pieces of the food and bring them back to the nest. If there is much food to gather at the food source and it is good food, the ants leave a trail with more pheromones on their way back [6]. The ants, which chose the long path in the beginning reach the food source later. When returning to the nest, they prefer the path with more pheromones, which is the shorter path. The pheromones on the paths evaporate partly to avoid a convergence of the swarm towards local minima [8]. Nevertheless, the pheromone value on the shorter path is higher than on the longer one. As a result, all ants decide to take the short way in the end [6].

This simplified food searching process is simulated by the ACO algorithm. In the following, this algorithm is explained for an TSP application. For this application, the path is represented by a sequence of nodes, which are connected by edges. The symbols used in the equations are listed in Table I.

TABLE I. Symbols used in the formulas explained in Section II

symbol used	meaning
s	next node
r	current node
u	next possible node
k	ant
$J_{k(r)}$	all nodes that have not been visited yet by ant k
$\tau(r,u)$	pheromone value of an edge between r and u
$\eta(r,u)$	inverse of distance between r and u
β	parameter to manipulate the proportion between distance and pheromone value ($\beta > 0$)
q	random number between [0...1]
q_0	proportion between exploration and exploitation ($0 \leq q_0 \leq 1$)
S	random variable connected to the random-proportional rule
$p_k(r,s)$	probability to choose node s as next node
ρ	pheromone decay parameter for local update ($0 < \rho < 1$)
τ_0	initial pheromone value
α	pheromone decay parameter for global update ($0 < \alpha < 1$)

In the beginning, all edges have the same pheromone value, and each ant chooses its first tour randomly [9].
The ACO is divided into four steps:

1) all ants are planning their tour according to the pheromone value on the path,
2) ants leave pheromones on the path,
3) the pheromone value of the global-best path is updated,
4) the pheromone values on all edges partly evaporate.

Those steps are explained in the following and are visualized in Figure 1.

1) Path Planning: Each ant of the swarm plans its path according to the State Transition Rule

$$s = \begin{cases} \arg\max_{u \in J_{k(r)}} \{[\tau(r,u)] \cdot [\eta(r,u)]^\beta\}, \\ \quad \text{if } q \leq q_0 \text{ (exploitation)} \\ S, \quad \text{otherwise (biased exploration)} \end{cases}, \quad (1)$$

where r is the current node of the ant k, s is the next node, and q is calculated randomly [9]. Is q smaller than or equal to q_0, the ant chooses exploitation. Otherwise exploration is chosen. In the case of exploitation, the ant chooses the best path according to the value of pheromones on the edge $\tau(r,u)$ and the length of the distance between the current node r and a possible node u ($\eta(r,u)$). The balance between distance and pheromone value is regulated by β. For all $u \in J_{k(r)}$, i.e., all remaining nodes that have not been visited yet, $\tau(r,u) \cdot \eta(r,u)^\beta$ is calculated and the maximum is chosen [9].

If biased exploration is chosen, the next node is selected with the random-proportional rule

$$p_k(r,s) = \begin{cases} \frac{\tau(r,s)\cdot\eta(r,s)^\beta}{\sum_{u \in J_{k(r)}} \tau(r,u)\cdot\eta(r,u)^\beta} & \text{if } u \in J_{k(r)} \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

where S represents the result of this random-proportional rule [9]. Equation (2) calculates the probability for each node to be chosen based on the pheromone values on the edges and their length. Short edges with high pheromone values are preferred. For exploration, $\tau(r,s) \cdot \eta(r,s)^\beta$ is calculated as well [9]. This results in a weighted value, which includes the pheromone value on the path as well as the length of the path. For this reason, the exploration is referred to as biased exploration [2]. The term is divided by the sum of all $\tau(r,u) \cdot \eta(r,u)^\beta$, where u is a possible node that has not been visited yet [9].

After all ants have chosen their tour and have returned to the initial node, the pheromone values are updated.

2) Local Update Rule: While the ants take their tour, they leave pheromones on the path. In analogy to real ants, the values depend on the quality and quantity of the food they encountered. The more and the better the food, the more pheromones they leave [6]. The pheromone value of each edge, which is part of an ant's tour, is updated according to

$$\tau(r,s) = \rho \cdot \Delta\tau(r,s)\,, \quad (3)$$

with $0 \leq \rho \leq 1$ [9]. The value of $\Delta\tau(r,s)$ depends on the implementation [9]. One option is to set it to a const

$$\Delta\tau(r,s) = \tau_0\,. \quad (4)$$

There are other approaches, e.g., using Reinforcement Learning to determine $\Delta\tau(r,s)$ [9]. For the sake of simplicity, the initial pheromone value in (4) is used to update the pheromone values on the ant's path.

 ISBN: 978-1-61208-765-8

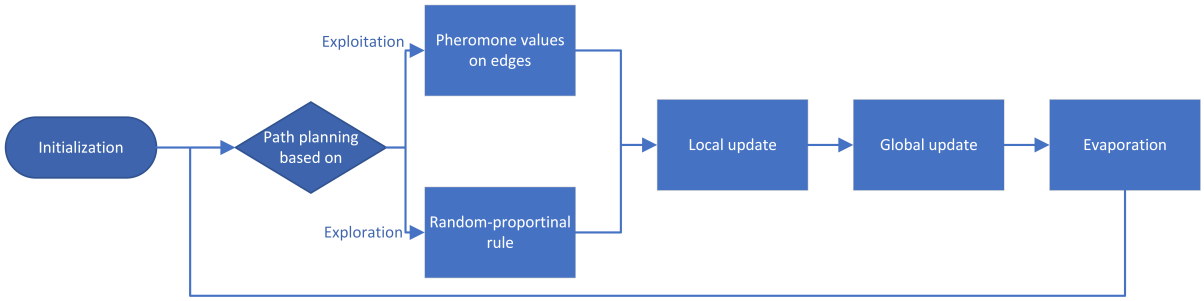

Figure 1. Steps of the ACO algorithm

3) Global Update Rule: After each ant of the swarm has completed its path and updated the pheromone values on the edges it has visited, the global update is performed. Extra pheromones are spread on the globally best tour, i.e., the shortest tour found so far. The shortest tour is identified and the pheromone values of each edge belonging to the globally best path is updated by

$$\tau(r,s) = \alpha \cdot \Delta\tau(r,s)\,, \tag{5}$$

where α is a predefined parameter between 0 and 1 [9].

4) Evaporation: To avoid rapid convergence, parts of the pheromone values evaporate in each iteration. Moreover, this offers the possibility to explore new areas [6]. For each edge, which is updated, the updating rules are modified, so parts of the pheromones evaporate [9]. The global update rule is modified and results in

$$\tau(r,s) = (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s)\,. \tag{6}$$

An evaporation factor is added to the local update as well,

$$\tau(r,s) = (1-\rho) \cdot \tau(r,s) + \rho \cdot \Delta\tau(r,s)\,. \tag{7}$$

The previously explained steps are iterated for a defined number of iterations.

Besides the TSP application, the ACO has been used in swarm robotics, e.g., for Unmanned Aerial Vehicles (UAVs) [10], or path planning on mobile robots in [11] and [12]. Additionally, the ACO has been applied to load balancing for peer-to-peer networks [13] or fuzzy logic controller [14]. Some variants published over the past few years are summarized in Table II.

TABLE II. ACO VARIANTS

Variant	Summary	Ref.
Inverted ACO (IACO)	inverts effect of pheromones	[13]
ACO Variants Subset Evaluation (AVSE)	finds best solution by comparing solutions of different ACO variants	[14]
Improved ACO (ICMPACO)	divides problem into sub-problems and introduces roles for the ants	[15]
Voronoi-based ACO (V-ACO) (V-ACO)	ACO combined with Voronoi partition with tournament selection method	[16]
Adaptive Continuous Ant Colony Optimization ($AACO_{\mathbb{R}}$)	adaption of evaporation rate based on relative diversity	[17]
Improved Continuous Ant Colony Optimization ($IACO\mathbb{R}$)	success-based random-walk selection (Brownian motion and Lévy flight)	[18]

III. PARTICLE SWARM OPTIMIZATION (PSO)

In contrast to the ACO, which is based on the social behavior of ants, the PSO has its origin in the observation of bird flocks. Imagine a bird flock or a fish school that is moving. Although, there can be hundreds of individuals, the movement of the whole swarm seems as they are one. To achieve this behavior, the individual elements of the swarm interact with their direct neighbors to reach a collective movement. To imitate the aforementioned behavior, the PSO algorithm was developed. The symbols used in equations throughout this section are explained in Table III.

TABLE III. SYMBOLS USED IN THE FORMULAS EXPLAINED IN SECTION III

symbol used	meaning
$\mathbf{v}_i$	velocity of particle i
c_1, c_2	acceleration coefficients
$\mathbf{R}_1$, $\mathbf{R}_2$	vector containing random values between [0...1]
$\mathbf{p}_i$	personal best position of particle i
$\mathbf{x}_i$	position of particle i
$\mathbf{p}_g$	best position of particles in neighborhood

To reach a swarm-like behavior, each individual of the swarm, in the field of PSO called particle, determines the best position with the best fitness. Therefore, it takes its own experience and its neighbors best positions into account. Fitness specifies how good a solution is. The PSO was first mentioned in [3] in the field of simulating bird flocks. For the algorithm presented by the authors, the particles change their position by modifying their velocity in each iteration. The PSO is based on an Adaptive Culture Model mentioned in [19]. It consists of three principles:

- evaluate: the ability to determine if something is good or bad.
- compare: the ability to compare own results with neighbors.
- imitate: the ability to imitate the behavior of superior neighbors .

 ISBN: 978-1-61208-765-8

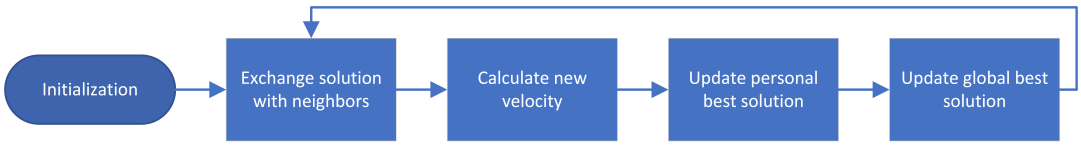

Figure 2. Steps of the PSO algorithm

The steps of the PSO algorithm follow the principles of the Adaptive Culture Model (Figure 2).

In each iteration, all particles of the swarm evaluate their own position. They have a "memory" to store all positions and are able to compare the current position to those stored in the past. The individual particle wants to return to a position that used to be better than the current position [3]. Each particle exchanges its position and their corresponding fitness with its neighbors. The neighborhood can either be the whole swarm, or is limited to a predefined number of nearest members of the swarm [19]. The particle's velocity is then updated according to its own results and its neighbors best positions. The velocity update is calculated by

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 \cdot R_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + c_2 \cdot R_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t)) \,, \quad (8)$$

where $\otimes$ indicates a point-wise vector multiplication. The velocity update can be divided into three parts [6]:

- momentum part
- cognitive part
- social part

The momentum part $\mathbf{v}_i(t)$ specifies the velocity of the particle i of the last iteration. Consequently, the particle stays on track. The cognitive part $c_1 \cdot R_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t))$ represents the particle's memory, where $\mathbf{p}_i$ refers to the best position of particle i, $\mathbf{x}_i(t)$ is the current position of i, c_1 is a acceleration coefficient, and $\mathbf{R}_1$ a vector containing random numbers between 0 and 1. The particle tends to go back to better positions it visited in the past. The social part is added by $c_2 \cdot \mathbf{R}_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))$. Here, c_2 is a acceleration coefficient, $\mathbf{R}_2$ refers to vector with a random numbers in the interval [0..1], and $\mathbf{p}_g$ is the best position of a particle in the neighborhood. The social part integrates the neighbors best positions and determines how much the particle is influenced by its neighbors' [6].

Equation (8) is designed for continuous problems, which needs to be adapted to discrete problems, e.g., the TSP. Therefore, the PSO uses permutations. Swap Sequences (SSs) are introduced, which replace velocities. One SS consists of multiple Swap Operators (SOs). A SO contains the information on which nodes are swapped [20]. As an example, a path (a, c, d, e, b) is defined that has been chosen by a particle and an SS $((1, 3), (5, 2))$. The swaps of all SOs (e.g., $(1, 3)$) are performed from left to right. The SO defines the indexes of the elements in the list which are swapped. So, after processing the first SO, the path changes to (d, c, a, e, b). Then, the second swap is performed and the path results in (d, b, a, e, c) [20].

For the TSP, the SS of each particle is updated every iteration according to (8). The difference between the current path of a particle and its personal best path is calculated by an SS. The algorithm searches for all swaps that are needed to transform the current path into the personal best path. The same procedure is followed for the current path and the global best solution. The terms $c_1 \cdot \mathbf{R}_1$ and $c_2 \cdot \mathbf{R}_2$, respectively, can be replaced by c_1 and c_2. Both variables are random values between 0 and 1 which determine the probability and the tendency to take the personal or global best solution [20].

After updating the velocity, i.e., the SS, the new path $\mathbf{x}_i(t+1)$ is calculated by

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \,, \quad (9)$$

where $\mathbf{x}_i(t)$ is the path and $\mathbf{v}_i(t+1)$ the velocity [21]. When SSs are utilized, the old path is permuted to compute the new path. Having the new path, the local update is conducted. If the distance of the new path is shorter than the distance of the personal best solution, the personal best path is updated. Furthermore, the new path is compared to the best path on a global level and the global best path is updated if $\mathbf{x}_i(t+1)$ is a better solution. The steps of the PSO are iterated multiple times, so the swarm is able to converge towards a collective solution.

The PSO has a wide range of applications, e.g., tuning of a Proportional-Integral-Derivative (PID) controller [22], or cloud computing [23]. Some variants, which have been published during the last years, are presented in Table IV.

TABLE IV. PSO VARIANTS

Variant	Summary	Reference
Adaptive Learning PSO (ALPSO)	employ Adaptive Learning strategy	[24]
Repository and Mutation based PSO (RMPSO)	introduces repositories for global best and personal best solutions	[25]
Decomposition Cooperative PSO (DCPSO); Merging Cooperative PSO (MCPSO)	assigns sub-swarms to sub-problems (divide-and-conquer)	[26]

IV. Bee Colony Optimization (BCO)

Another algorithm mentioned in the field of SI is the BCO. This algorithm is based on foraging behavior like the ACO [27]. In contrast to ants, honeybees communicate directly with the other members of the swarm. They transmit information without any physical interaction [7]. After the bees have been searching for food, they return to their hive. They dance in order to communicate the location of a food source. With the

 ISBN: 978-1-61208-765-8

honeybee's dance, they try to convince the other bees to choose the food source they are advertising [27]. Out of this behavior, an algorithm was designed to solve optimization problems like the TSP. This section focuses on the presentation of the BCO, following the steps of the algorithm visualized in Figure 3.

Each iteration of the algorithm is divided into multiple stages. During each stage, the bees build a partial solution and the following steps are conducted:

1) forward pass,
2) backward pass.

For the TSP application, the number of stages depends on the number of nodes m added to the partial solution during each iteration.

1) Forward Pass: During the forward pass, all bees go out of the hive and each bee builds its own partial solution. For the TSP application, a part of the path is created by adding m nodes [4]. In this implementation, the partial solutions are calculated randomly. After building their partial solution, the bees return to the hive. Then, the backward pass is performed.

2) Backward Pass: For the backward pass, the bees have two options [27]. They can either

- abandon their partial solution (exploitation) or
- dance and advertise their solution to the others (exploration).

If a bee decides to abandon its solution, it exploits the solution of another bee. The shorter the distance of the other bee's partial solution, the more likely the bee chooses this partial solution. After it has made a decision for a partial path, this part is added to the bee's own path [4]. For the TSP, every bee is allowed to visit every city only once. Consequently, it is crucial to check whether the chosen partial solution includes cities that have already been visited. If this is the case, for the implementation presented in this paper, the honeybee returns to its own partial solution.

After conducting forward and backward pass for all stages, a full path has been created. As a next step, the global best path is updated [27]. Therefore, the lengths of the paths of all bees are compared. The global best path is taken into account for the following backward passes and partial solutions of the global best path can also be chosen by the bees. After the global best path has been updated, one iteration is finished. To reach a convergence of the swarm, the steps of the BCO algorithms are iterated for a predefined number of iterations.

Variants of the BCO, e.g., have been employed for a swarm of autonomous drones [28] or path planning [29]. Especially, variants of the Artificial Bee Colony (ABC), which was first introduced in [30], have been published during the past few years. Some of those variants are summarized in Table V. In contrast to the BCO algorithm, the ABC divides the member of the swarm into different groups which perform different tasks.

All three algorithms have been implemented for the TSP and the following section presents the results obtained in experiments for all three algorithms.

TABLE V. BCO VARIANTS

Variant	Summary	Reference
Arrhenius Arrhenius aABC (aABC) (aABC)	position update equation combined with Arrhenius equation	[31]
Lévy Flight ABC (LFABC)	search strategy inspired by Lévy flight search	[32]
ABC with Reinforcement Learning (R-ABC)	reinforcement vector for solution update	[33]
Lbest Gbest ABC (LGABC)	take local best and global best solutions into account	[34]

V. EXPERIMENTAL RESULTS OF THE SI ALGORITHMS FOR THE TSP

The experiments conducted for the three algorithms presented above have a similar setup. Ten nodes are placed randomly on a grid. Those nodes represent the cities for all three algorithms. The distances between the nodes are different. Each algorithm is supposed to find a path which connects all nodes while traveling a minimum distance. The number of iterations has been limited to 200 for each algorithm. Each algorithm has been tested 100 times.

The algorithms have been implemented in Python. Goal of this paper was to test, if it is possible to solve the TSP with each of the algorithms. Therefore, the algorithms have not been implemented with respect to time efficiency and performance. In contrast to [5], the experiments focus on the importance of exploration and exploitation for each algorithm. For the evaluation of performance, see [5].

Table VI summarizes the parameter configuration used for the experiments.

TABLE VI. PARAMETERS USED FOR EXPERIMENTS

ACO		PSO		BCO	
parameter	**value**	**parameter**	**value**	**parameter**	**value**
iterations	200	iterations	200	iterations	200
population	100	population	50	population	100
β	0.7	c_1	0.6	m	3
q_0	0.8	c_2	0.4		
ρ	0.7				
τ_0	10.0				
α	0.9				

For each algorithm, the average length of the path that has been chosen by the swarm is evaluated. As the experiments were repeated 100 times for each algorithm, the path length is averaged over all repetitions. Figure 4 includes the experimental results of all three algorithms. The ACO is visualized in blue, the BCO in red, the PSO in green and the optimal solution in black.

For the ACO algorithm, the swarm consists of 100 ants. In Figure 5, the ants only focus on exploitation. They always choose the path with the highest pheromone value. This leads to a fast convergence of the swarm, but it converges to a local minimum. The swarm agrees on a non-optimal path.

 ISBN: 978-1-61208-765-8

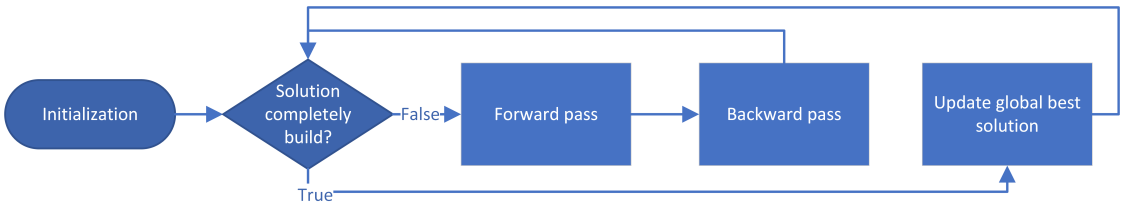

Figure 3. Steps of the BCO algorithm

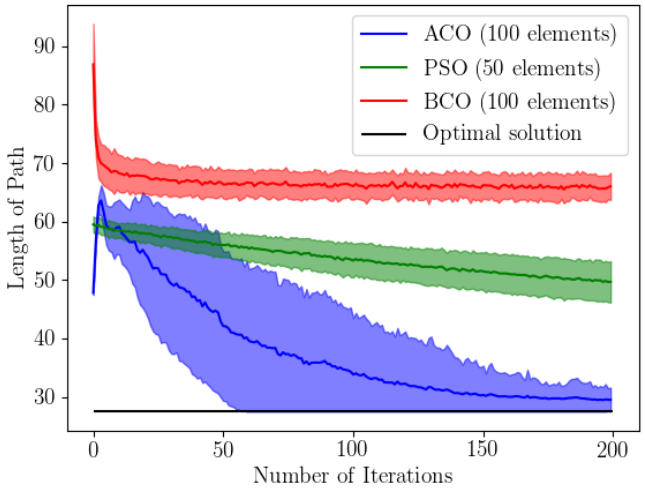

Figure 4. Average length of path by solving the TSP with the ACO (blue), the BCO (red) and the PSO (green)

Figure 4 (blue) shows that better results are obtained if a balance between exploration and exploitation is created. It takes more time until the swarm converges to a solution. However, the swarm's solution is better than the one found by focusing only on exploitation. This experiment shows that the balance between exploration and exploitation is important for SI algorithms.

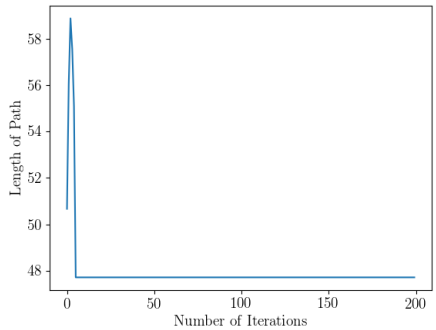

Figure 5. Proportion between exploration and exploitation is $q_0 = 1.0$. The ants choose exploitation with a probability of 100%

The results of the PSO algorithm in Figure 4 (green) are obtained with a swarm of 50 particles. The path length decreases with the number of iterations. The number of total iterations was limited, so the algorithm may have performed better if more iterations would have been allowed. The average path length is decreasing constantly due to an effective balance between exploration and exploitation. The experiments have shown that a large number of particles in the swarm leads to worse results when solving a TSP with PSO.

It is the other way around for the BCO. The more honeybees form the swarm, the better are the experimental results. In Figure 4, the swarm consists of 100 bees and is visualized in red. The advantage of this algorithm is the number of parameters. The user only has to define the size of the partial solutions. The disadvantage, as seen in the graph, is the convergence of the swarm. In contrast to the other algorithms, the length of the paths found decreases rapidly, but in comparison to the other algorithms, towards a non-optimal solution. This means exploitation predominates over exploration.

The experiments have shown that the TSP can be solved by all of the three algorithms. The balance between exploration and exploitation is important. Each algorithm has its advantages and disadvantages. To make a decision which algorithm is used, the number of elements in the swarm plays a role. Moreover, the number of parameters can be an advantage and a disadvantage. On the one hand, tuning takes a lot of time, on the other hand parameters make it possible to apply the algorithm for specific problems. For the TSP application presented in the paper, the ACO algorithm performs best. It includes a weighting of the solutions created by all ants for all iterations. The PSO and the BCO, in contrast, only remember the best solution found. According to the no-free-lunch theorem, it is not possible to favor one of the algorithms over the others for all problems [35]. It is necessary to evaluate each technique for the given application and make the decision which technique to use, based on the problem.

VI. Conclusion and Future Work

All three state-of-the-art SI algorithms are capable to solve a TSP. As PSO was designed for continuous problems, it takes more effort to implement it for discrete problems and the initial algorithm needs to be modified. The other algorithms can be implemented in a straight-forward way for the TSP. For the TSP, ACO performs best, but it depends on the problem, if the algorithms are suitable. Furthermore, exploration and exploitation need to be balanced.

 ISBN: 978-1-61208-765-8

This paper serves as an introduction to SI algorithms. Future work will focus on other applications for SI algorithms. SI algorithms will be implemented and evaluated for board games, in particular for Halma (Chinese Checkers). In this game, each player has 10 or 15 game characters, depending on the number of players. The game characters represent the swarm. For each move, only one game character is allowed to move and the size of the swarm is relatively small. As a result, a combination of the ACO and a modified BCO algorithm is thinkable. The local path of each game character is planned by applying the ACO algorithm. When it is a player's turn, only one game character is allowed to move. The decision which character is chosen is made by the BCO algorithm.

The presented algorithms can also be applied to robotic swarms. For this application, the algorithms can be combined to assign different routes to drones for a disaster management mission.

ACKNOWLEDGEMENT

This paper has been written during a cooperative study program at the Baden-Wuerttemberg Cooperative State University Mannheim and the German Aerospace Center (DLR) Oberpfaffenhofen at the Institute of Communications and Navigation.

REFERENCES

[1] M. Beekman, G. A. Sword, and S. J. Simpson, "Biological foundations of swarm intelligence," in *Swarm Intelligence: Introduction and Applications*, C. Blum and D. Merkle, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–41. [Online]. Available: https://doi.org/10.1007/978-3-540-74089-6_1

[2] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial life*, vol. 5, no. 2, pp. 137–172, 1999.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 11 1995, pp. 1942–1948.

[4] D. Teodorović, "Bee colony optimization (bco)," in *Innovations in Swarm Intelligence*, C. P. Lim, L. C. Jain, and S. Dehuri, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 39–60. [Online]. Available: https://doi.org/10.1007/978-3-642-04225-6_3

[5] J. Odili, M. N. M. Kahar, A. Noraziah, and S. F. Kamarulzaman, "A comparative evaluation of swarm intelligence techniques for solving combinatorial optimization problems," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417705969, 2017.

[6] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm Intelligence: Introduction and Applications*, C. Blum and D. Merkle, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 43–85. [Online]. Available: https://doi.org/10.1007/978-3-540-74089-6_2

[7] V. Trianni, *Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots*. Springer, 2008, vol. 108.

[8] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge : MIT Press, 2004.

[9] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[10] Z. Ziyang, Z. Ping, X. Yixuan, and J. Yuxuan, "Distributed intelligent self-organized mission planning of multi-uav for dynamic targets cooperative search-attack," *Chinese Journal of Aeronautics*, vol. 32, no. 12, pp. 2706–2716, 2019.

[11] Y. Liu, J. Ma, S. Zang, and Y. Min, "Dynamic path planning of mobile robot based on improved ant colony optimization algorithm," in *Proceedings of the 2019 8th International Conference on Networks, Communication and Computing*, 2019, pp. 248–252.

[12] R. Uriol and A. Moran, "Mobile robot path planning in complex environments using ant colony optimization algorithm," in *2017 3rd international conference on control, automation and robotics (ICCAR)*. IEEE, 2017, pp. 15–21.

[13] S. Asghari and N. J. Navimipour, "Resource discovery in the peer to peer networks using an inverted ant colony optimization algorithm," *Peer-to-Peer Networking and Applications*, vol. 12, no. 1, pp. 129–142, 2019.

[14] O. Castillo, E. Lizárraga, J. Soria, P. Melin, and F. Valdez, "New approach using ant colony optimization with ant set partition for fuzzy control design applied to the ball and beam system," *Information Sciences*, vol. 294, pp. 203–215, 2015.

[15] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE access*, vol. 7, pp. 20 281–20 292, 2019.

[16] C. Xiong, D. Chen, D. Lu, Z. Zeng, and L. Lian, "Path planning of multiple autonomous marine vehicles for adaptive sampling using voronoi-based ant colony optimization," *Robotics and Autonomous Systems*, vol. 115, pp. 90 – 103, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889018304469

[17] M. Omran and R. Polakova, "A memetic and adaptive continuous ant colony optimization algorithm," in *International Conference on Theory and Application of Soft Computing, Computing with Words and Perceptions*. Springer, 2019, pp. 158–166.

[18] M. G. Omran and S. Al-Sharhan, "Improved continuous ant colony optimization algorithms for real-world engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 818–829, 2019.

[19] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*, ser. The Morgan Kaufmann series in evolutionary computation. San Francisco [u.a.] Morgan Kaufmann, 2009.

[20] K.-P. Wang, L. Huang, C.-G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, vol. 3, 11 2003, pp. 1583–1585.

[21] I. Khan, S. Pal, and M. K. Maiti, "A modified particle swarm optimization algorithm for solving traveling salesman problem with imprecise cost matrix," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2018, pp. 1–8.

[22] S. Momani, R. El-Khazali, and I. M. Batiha, "Tuning pid and piλdλ controllers using particle swarm optimization algorithm via el-khazali's approach," vol. 2172, no. 1, p. 050003, 2019.

[23] A. Naseri and N. J. Navimipour, "A new agent-based method for qos-aware cloud service composition using particle swarm optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 1851–1864, 2019.

[24] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, and X.-L. Shen, "A hybrid particle swarm optimization algorithm using adaptive learning strategy," *Information Sciences*, vol. 436-437, pp. 162 – 177, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025518300380

[25] B. Jana, S. Mitra, and S. Acharyya, "Repository and mutation based particle swarm optimization (rmpso): A new pso variant applied to reconstruction of gene regulatory network," *Applied Soft Computing*, vol. 74, pp. 330–355, 2019.

[26] J. Douglas, A. Engelbrecht, and B. Ombuki-Berman, "Merging and decomposition variants of cooperative particle swarm optimization: New algorithms for large scale optimization problems," in *Proceedings of the 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, 2018, pp. 70–77.

[27] K. Diwold, M. Beekman, and M. Middendorf, "Honeybee optimisation – an overview and a new bee inspired optimisation scheme," in *Handbook of Swarm Intelligence: Concepts, Principles and Applications*, B. K. Panigrahi, Y. Shi, and M.-H. Lim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 295–327. [Online]. Available: https://doi.org/10.1007/978-3-642-17390-5_13

[28] A. Viseras, T. Wiedemann, C. Manß, V. Karolj, D. Shutin, and J. M. Gomez, "Beehive inspired information gathering with a swarm of autonomous drones," *Sensors*, October 2019. [Online]. Available: https://elib.dlr.de/129660/

[29] Z. Li, Z. Zhang, H. Liu, and L. Yang, "A new path planning method based on concave polygon convex decomposition and artificial bee colony algorithm," *International Journal of Advanced Robotic Systems*, 2020. [Online]. Available: https://doi.org/10.1177/1729881419894787

 ISBN: 978-1-61208-765-8

[30] D. Karaboga, "An idea based on honey bee swarm for numerical optimization, technical report - tr06," *Technical Report, Erciyes University*, 01 2005.

[31] S. Kumar, A. Nayyar, and R. Kumari, "Arrhenius artificial bee colony algorithm," in *International conference on innovative computing and communications*. Springer, 2019, pp. 187–195.

[32] H. Sharma, J. C. Bansal, K. Arya, and X.-S. Yang, "Lévy flight artificial bee colony algorithm," *International Journal of Systems Science*, vol. 47, no. 11, pp. 2652–2670, 2016.

[33] S. Fairee, S. Prom-On, and B. Sirinaovakul, "Reinforcement learning for solution updating in artificial bee colony," *PloS one*, vol. 13, no. 7, 2018.

[34] H. Sharma, S. Sharma, and S. Kumar, "Lbest gbest artificial bee colony algorithm," in *2016 International conference on advances in computing, communications and informatics (ICACCI)*. IEEE, 2016, pp. 893–898.

[35] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

 ISBN: 978-1-61208-765-8