U.P.B. Sci. Bull., Series C, Vol. 79, Iss. 2, 2017 ISSN 2286-3540

NORMALIZATION OF ACO ALGORITHM PARAMETERS

Alina E. NEGULESCU[1]

Due to the fact that Swarm Systems algorithms have been determined to be efficient in solving discrete optimization problems with proven applicability into practical world, the computer scientists are continuously and increasingly discovering swarm-inspired algorithms or improving existing ones. The scope of this paper is to present such an improvement brought to the Elitist Ant System (EAS) algorithm through the normalization of its parameters' values. It is important to mention that Normalized EAS (N-EAS) behaves exactly as EAS in terms of solution length and computational time since the algorithm behind is the same and the only difference is the normalization of the parameters. The advantage of N-EAS is that it spares computational time for otherwise running empirical test-runs for determining a good set of parameter values, like in the case of EAS.

Keywords: Swarm Intelligence (SI), Ant Colony Optimization (ACO), Elitist Ant System (EAS), Normalized EAS (N-EAS)

1. Introduction: The Natural Model

The "collective intelligence" of the swarm systems in nature have inspired computer scientists to design algorithms by modeling their behavior [4], [5]. As Brownlee remarked in [1], "collective intelligence emerges through the cooperation of *usually* large numbers of homogenous entities in the environment and examples include colonies of ants, bees, termites, flocks of birds or schools of fish". The amazing thing about these entities is the fact that they are able of find solutions to problems without requiring any centralized management, this way exhibiting self-organization, which is extremely helpful for finding food, relocating and protecting from predators [2].

As regards the ants, they communicate indirectly with each other using pheromones deposited in the environment [2], while the bees use the dance moves [3] and the fish and birds the proximities (for a direct communication). Fig. 1 presents the way by which the ants travel from their nest to a food source and back by using the shortest possible path. In the first image, Fig. 1.a, two groups of ants are preparing to leave the nest. Between the nest and the food source there are many possible routes they could choose from, however, only two seem relatively shorter. Assuming that the upper route is 2 times longer than the lower one, each group of ants will choose, with 50% probability, one of the two as they cannot

[1] PhD student, Computer Science Department, University POLITEHNICA of Bucharest, Romania, e-mail: alina.lascu@gmail.com

know from the beginning which one is shorter. Nevertheless, the group of ants which will choose the lower route (below the obstacle) will arrive at the food source approximately two times faster, as represented in Fig. 1.b.

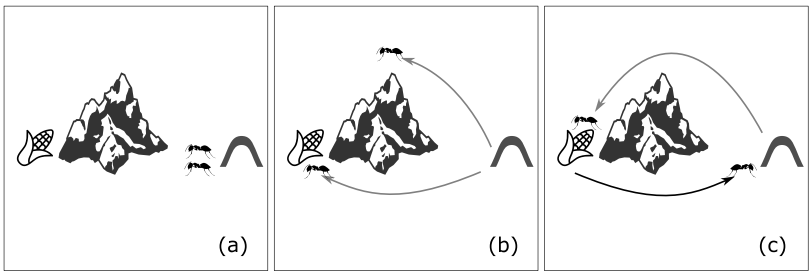

Fig. 1. Ants finding the shortest path to a food sources and then back to the nest

As the ants are moving, they deposit a pheromone trail in the environment. The pheromone, as described in [2] is a hormone produced by ants through which they establish a sort of indirect communication among them. Moreover, the ants returning back to the nest, reinforce the initial trail, which, as described in [3], will attract other ants in the proximity, determining them to follow this trail with a greater probability than the less intense ones, this way, laying more pheromone. This process, in its essence, represents the positive feedback loop system, because the higher the intensity of a pheromone trail, the higher the probability of an ant (or group of ants) choosing and reinforcing it, over and over again. This process stands at the basis of the *stigmergy* concept introduced by Grassé in [6], which describes an indirect form of communication mediated by the modifications in the environment.

No less important is the negative feedback, which following the example provided in Fig. 1.c. represents the evaporation of the pheromone trail, due to natural conditions (i.e. wind). Because of evaporation, when the group of ants that have chosen the upper route reach the food source and must return to the nest, they will choose, with a higher probability, to return back using the lower route as it contains grater pheromone intensity. The evaporation of the pheromone trails from the environment is important because it allows the ants in the natural system to "forget" the long routes.

2. Review: ACO algorithms and their features

Inspired from the natural model of ants finding the shortest path to a food source and then back to the nest (Fig. 1), Marco Dorigo, who is considered the proponent of Ant Colony Optimization and one of the founders of the Swarm Intelligence research field, has proposed the Ant System [7] as the first Travelling Salesman Problem (TSP) dedicated algorithm.

TSP is an easy problem to understand but can be very hard to solve. A short description of the problem is that a travelling salesman must visit a list of cities where every two cities are directly connected, passing through all of them only once and returning to the starting one (resulting a Hamiltonian cycle in the associated problem's graph). The objective is that the Travelling Salesman will perform this task by using the shortest route. For a small number of cities (nodes in the graph) the problem is easy to solve, however, the complexity grows quickly as the number of possible combinations is increasing in a factorial manner.

The initial ACO algorithm and its variants are included in the meta-heuristics category [8]. Meta-heuristics is applied for TSP where an optimal solution is looked for in a discrete search-space where the candidate solutions increase in a factorial manner. Therefore, in general, the exhaustive search of the optimal solution is infeasible. As a result, for meta-heuristics, the scope is to find rather "good" solutions than to compute the best one.

For attaining algorithm improvement (following the design of the AS algorithm (1992) [8] – which stands at the foundation of ACO algorithms), other variants have been designed under the same umbrella. Since the scope of this paper is centered on the normalization of the parameters of the EAS algorithm, the other variants will only be briefly mentioned. The first improvement brought to the AS algorithm, as presented in [11], has been obtained by introducing (also in 1992) the Elitist AS (EAS) algorithm [12] which had the scope to weight the "best-so-far solution" higher, in order to increase the exploitation through bias. Consequently, in 1996, Stützle and Hoos had introduced the Max-Min Ant System (MMAS) algorithm [13] which followed the same design as the AS algorithm, however, as described in [14]: "(i) pheromone trails are only updated offline by the daemon (the arcs which were used by the best ant in the current iteration receive additional pheromone) and (ii) pheromone trail values are restricted to an interval $[\tau_{min}, \tau_{max}]$, while (iii) trails are initialized to their maximum value τ_{max}".

Soon after, the Rank-based AS (RAS) algorithm's improvement, proposed in [15] and referred in [11], consisted in the fact that "at each iteration the best-so far solution has the highest influence on the pheromone update, while a selection of the best solutions constructed at that current iteration influences the update depending on their ranks". Also, from the same category of successful ACO variants referred in [11], Ant Colony System (ACS) algorithm, introduced in [16], brought more improvements besides the pheromone update. The solution construction is particular, being called "pseudo-random proportional", while the best solution update rule is "only applied to the pheromone value τ_i, whose corresponding solution component c_i was added to the solution under construction". The overall benefit of that mechanism is that the ants are exploring the search space more, with every iteration. Following the natural model presented in Fig. 1 and adding the distance between nodes as heuristic information, the most important parameters in relation with the objective function ($l = tour_{length}$), for any ACO algorithm, are described below:

- *m* (the number of artificial entities);
- *n* (the number of nodes / cities);
- α (the influence of the pheromone intensity when choosing the next node to visit);
- β (the influence of the distance between the nodes when choosing the next node to visit);
- ρ (the pheromone evaporation speed);
- Q (the pheromone quantity deposited on the arches in the graph);
- ε (the multiplication value of the pheromone intensity for elitist ants) – applicable only for Elitist Ant System (EAS) algorithm, where the ants which find a better solution are allowed to deposit more pheromone.

The pseudo-code for ACO applied to Travelling Salesman Problem (TSP) is the following:

```
function Ant_Colony_Optimization
{
      Initialize_Parameters;

      Initialize_Pheromone_Trails;

      while (stop_condition_not_met)
      {
            Construct_Solutions(m);
            Global_Update_Trails();
            Evaporate();
            Store_Best_Solution(m);
      }
}
```

Initialize_Parameters function has the role to set the algorithm parameters such as α, β, ρ and *m*, as well as loading the map (i.e. TSP benchmark). In order to “kick-start” the algorithm, a minimum value of pheromone (τ_0) is deposited equally on all the graphs' arches, this action being performed by the *Initialize_Pheromone_Trails* function (which is missing from the original AS algorithm).

The usual stop condition of the algorithm is represented by a maximum number of iterations or imposed time limit. While the condition is not met, *Construct_Solution* function will be applied to all *m* ants. The set of steps performed by this function includes the positioning of ants on the map, either randomly or predictably (ant 1 in node 1, ant 2 in node 2 and so on) and choosing the next node for finding the solution (shortest route) based on attractability. In this function, as initially implemented for Ant System (AS) algorithm and depicted in [6], the transition from node *i* to node *j* rule, is based on:

- d_{ij} – the distance (the cost) between i and j nodes;
- $\eta_{ij} = 1/d_{ij}$ – the heuristic information indicating the benefit of passing from the node i to the node j (it is equivalent with the visibility between the two nodes);
- $\tau_{ij}(t)$ – the pheromone intensity at the iteration t between the i and j nodes;
- $$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}}, if j \in N_i^k \\ 0, if j \notin N_i^k \end{cases} \quad (1)$$

The above formula, presented in [7], represents the probability p at the iteration t for an ant *k* in node i to choose node j as the next node in which it will travel, which is equal with the ratio between the intensity τ of the pheromone between the nodes i and j at the iteration t weighted with α, and multiplied by the visibility between the nodes i and j weighted with β […] and the sum of the products between the *intensity of the pheromones* between any two nodes through which the ant did not pass (weighted with α) and *the visibility* between any two nodes through which the ant did not pass (weighted with β). Also:

- N_i^k– is the set of nodes where the ant k is allowed to go (this set is composed by all the nodes which were not yet visited);
- α, β – determine the influence of the pheromone trail and of the heuristic information when choosing the next node.

This function is repeated for every ant (*m*). After all ants (artificial entities) have fulfilled their tour, the function *Global_Update_Trails* has the role to calculate the intensity of the pheromone on all visited arches as a sum of pheromone intensities. This function represents the positive feedback, essential for convergence to solution as in nature. The function for evaporation has likewise an important role, because otherwise, the algorithm would get trapped in local optima. Last but not least, all the tours performed by the ants are calculated and the best solution is stored.

With this overview in mind, the next section will be focusing on the EAS algorithm, which, for the purpose of this research undertaking, has been modified (i.e. through the normalization of the algorithm's parameters values) in order to allow avoiding the initial multiple test-runs needed for setting up the parameters values, which are dependent of the problem size and type.

3. Problem definition and proposed solution

A computational problem represents a task which can be suitable for being solved by a computer, which in essence, is equal to stating that an algorithm can solve the problem. Two main computational resources are considered when determining the complexity of a problem: time and storage. Owing to Moore's Law, the storage problem has become less stringent, as the memory capacity increased while its cost decreased, and, although not being considered of reduced importance, it is however, not impacting the solution as much as the time is.

Considering the applicability of swarm-inspired algorithms, such as ACO, for practical problems, the solution is imperative to be provided in real-time. Any computation that might be avoided due to the optimization of the algorithm's parameters have an immediate value for delivering an optimal solution within the time-constraint. A drawback of the classical approach is that the parameters of the algorithm are strongly dependent on the size and topology of the map (i.e. problem) [17]. As such, a few test-runs are needed before solving a problem in order to determine appropriate values for the main parameters α and β, thus impacting the solution delivery in terms of time. To address this drawback, researchers have modified the original algorithm to be able to adapt its parameters dynamically while solving the problem, as presented in [8], [9], [10], [11], [17] and [18].

A practical example is the distance measurement unit between two cities. If on a regular (physical) map, the distance between Berlin and Los Angeles is represented by several centimeters, in reality, the actual distance is 9,304 kilometers. The relation between the actual distance and the map distance is proportionally calculated. By normalizing these distances, that can vary from graph to graph, for example, between 10 and 10000 (on one graph) and between 0.01 and 50 (on another), the β parameter will be much less influenced. Regardless the graph, 10000 and 50 will always be maximum 1, while 10 and 0.01 will always normalize to 0 value. The rest of the distances in the graph will be calculated proportionally to the smallest and biggest value, specifically, they will take values in the $[0,1]$ domain.

Considering time saving, *by avoiding the initial multiple test-runs for setting up the parameters values*, the main improvements of the N-EAS algorithm, are as follows:

- All the distances will be transformed in [0, 1] domain; when a graph (i.e. a map) is loaded, the minimum and maximum distance between the nodes is determined, by using the formula (2), where a and b are any two nodes in the graph:

$$new_{distance[a][b]} = \frac{distance[a][b] - min_{distance}}{max_{distance} - min_{distance}} \quad (2)$$

- The length of an ant's k tour is also maintained normalized, as a sub-unitary value, by summing the length of the n arches in the tour and dividing the obtained length by n as presented in formula (3). Since all arches lengths are sub-unitary, summing n sub-unitary values and dividing their sum by n, the result will also be sub-unitary:

$$l_k = \frac{\sum_{arch=1}^{n} length(arch)}{n} \quad (3)$$

- α and β parameters are sub-unitary as well, and need to be true for:

$$\alpha + \beta = 1 \quad (4)$$

- ρ parameter (evaporation speed) also maintains the same sub-unitary trend;
- The pheromone intensity is initialized when the algorithm starts with an equal (sub-unitary) value on all arches, using formula (5). The m in the denominator is ensuring that even if all ants will use that arch in their paths, when updating the pheromone using the formula (6), the pheromone value will not exceed 1.

$$\tau_0 = \frac{1}{m} \quad (5)$$

- When the algorithm is running, the pheromone intensity is updated, through deposit and evaporation, this value is also sub-unitary;
- For deposit, the following formula is applied:

$$\tau_{ij} = \tau_{ij}^{(1-Q)*(1-l)} \quad (6)$$

- While for evaporation:

$$\tau_{ij} = \tau_{ij} * (1 - \rho) \quad (7)$$

Since the solutions obtained with the normalized algorithm represent a sub-unitary value, it is necessary to convert them back to the original units of measure. This step is required in order to be able to easily compare them towards other algorithms results. The formula (8) is applied for the inverse conversion:

$$sol_{lenght} = l_{best} * (max_{distance} - min_{distance}) + min_{distance} \quad (8)$$

After imposing all the parameters values within [0,1] domain, the next natural step was to test that the new algorithm has not been altered in terms of performance, by comparing the obtained results with the results obtained using the original algorithm. Once this step was performed successfully (N-EAS algorithm behaved exactly the same as EAS), the next one was to methodologically determine the optimal parameters values for a set of similar benchmarks. Consequently, a statistical study has been performed by modeling a rank I full factorial experiment FFE3^3 in order to determine the optimal values for the algorithm's parameters. Taking into consideration formula (4) where $\alpha + \beta = 1$, it resulted:

- 3^3 level combinations;
- $3^3 - 1$ independent combinations;
- the squared sums were determined for 3 main effects, every main effect having 2 independent combinations;
- between 2 factors there were C_2^3 interactions;
- an interaction between all 3 factors with 2^3 independent combinations.

The central point coordinates, means, standard deviations and the variation intervals for the 3 influential factors were calculated along with their correlations and regressions.

Variation intervals for all factors are equal to 1.0 and their central point is 0.5, low level and high level being equal to 0.0 and 1.0 (since they are normalized). The means for all factors is 0.5 and the standard deviations is 0.416. For the dependent variable (y = solution length) the mean is 0.276 and the standard deviations is 0.262.

The correlation between all the independent variables is 0.0 and the correlations between each independent variable and the dependent variable are: $x_1 \leftrightarrow y = -0.499$, $x_2 \leftrightarrow y = 0.382$ and $x_3 \leftrightarrow y = 0.481$.

The computed regressions for the three independent factors are:

- x_1: $R = 0.498$, $R^2 = 0.248$, $\beta = -0.498$, std. err. of $\beta = 0.173$, $B = -0.313$, std. err. of $B = 0.109$, $p - value = 0.008$;
- x_2: $R = 0.382$, $R^2 = 0.146$, $\beta = 0.382$, std. err. of $\beta = 0.184$, $B = 0.240$, std. err. of $B = 0.116$, $p - value = 0.049$;
- x_3: $R = 0.481$, $R^2 = 0.231$, $\beta = 0.481$, std. err. of $\beta = 0.175$, $B = 0.302$, std. err. of $B = 0.110$, $p - value = 0.011$.

Interpreting the computed regressions presented above, the following conclusions were drawn:

- the $p - values$ are indicating that the three null hypotheses for the factors are rejected, since they are below the chosen value of 0.05, meaning that all the factors are influencing the solution length;

- the x_1 factor, which represents the β parameter of the algorithm, has negative values for the standardized (β) and the unstandardized (B) coefficient, indicating that there is an inverse relation between it and the solution length.
- the factors x_2 and x_3, representing the ρ and ε parameters have positive values for the standardized (β) and the unstandardized (B) coefficient, indicating that there is direct relation between them and the solution length.

The above conclusions are also supported by the graphical representation of the interactions between every two independent factors and the solution length (Figs. 2 - 4).

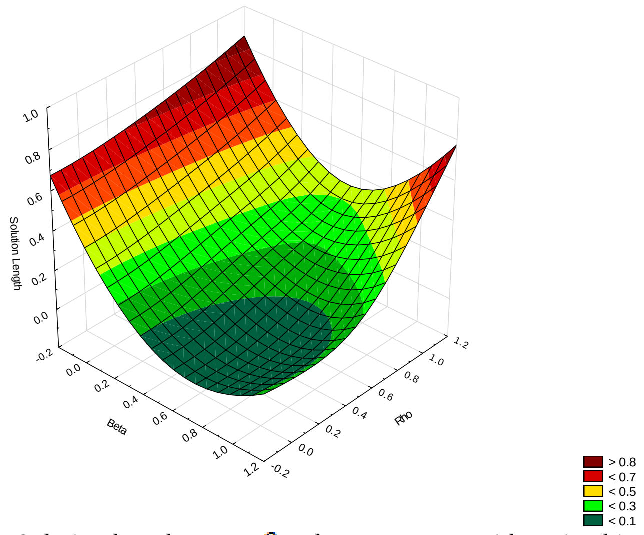

Fig. 2. Solution length versus β and ρ parameters with optimal intervals: $\beta \in [0.4, 1.0]$, $\rho \in [0.0, 0.6]$.

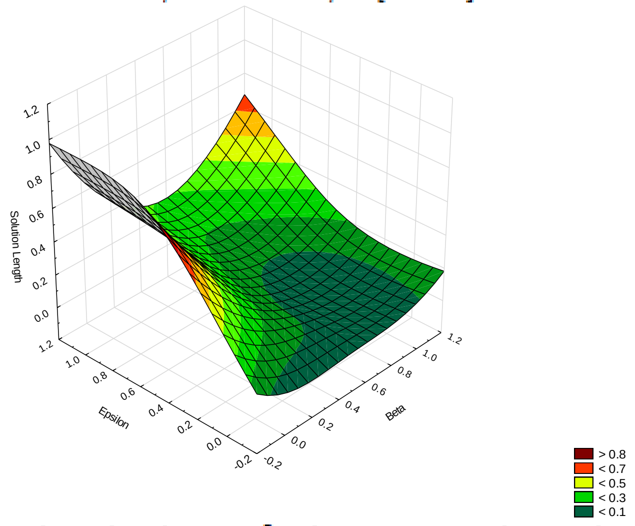

Fig. 3. Solution length versus β and ε parameters with optimal intervals: $\beta \in [0.3, 1.0]$, $\varepsilon \in [0.0, 0.6]$

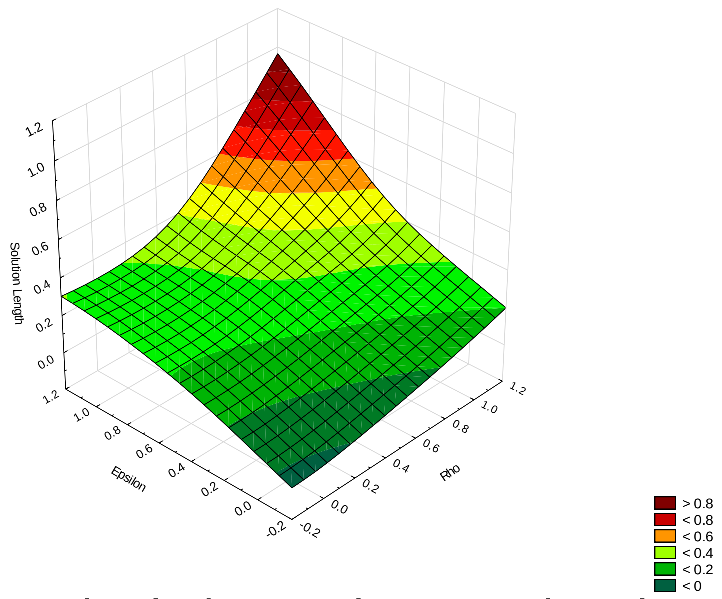

Fig. 4. Solution length versus ρ and ε parameters with optimal intervals: $\rho \in [0.0, 1.0]$, $\varepsilon \in [0.0, 0.6]$

After determining, for each FFE graph, the optimal domain values for the factors, by using a simple intersection of these domains can further allow identifying the optimal algorithm's parameters values. Taking into the account the relation $\alpha + \beta = 1$ and the fact that $\beta \in [0.4, 1.0]$ (resulting from the intersection of the domains) the optimal values are $\alpha = 0.3$ and $\beta = 0.7$. Since both ρ and ε domains are in the $[0.0, 0.6]$ domain, the optimal values of these parameters were determined to be $\rho = 0.5$ and $\varepsilon = 0.5$.

4. Conclusions

It is a well-known fact that ACO algorithm's performance of meta-heuristics is very dependent on the parameters values. This is the reason for which the computer scientists, that perform researches in this field, dedicate a lot of effort into determining appropriate values for them. The determination of suitable values for these parameters can be performed either *a priori*, or *a posteriori* the algorithm's employment [19].

The disadvantage of the *a priori* approach is that is effortful, time-consuming and usually done by trial and error without guaranteeing optimal values. However, these values can be optimally *a priori* obtained by *methodologically* testing all possible combinations. An alternative to the *a priori* determination is the *a posteriori* one, which consists in altering the algorithm's parameter values while the algorithm is running for solving a problem. By allowing automatic fine tuning of its parameters in relation with the problem's instance, the algorithm becomes robust, even if it was initially designed for a different context.

The main advantage of the approach presented herein (i.e. normalization of the algorithm’s parameters values), which falls into the *a priori* category, consists in the fact that the initial test-runs for setting up the algorithm’s parameters (determining its values) can be extrapolated for other similar maps. Other advantages are that all the parameters are normalized to the [0,1] domain and that $\alpha + \beta = 1$, resulting in a much clearer ratio between the attractiveness of the heuristic information (the distance) and the pheromone intensity, when the artificial entities choose the next node in the graph. The normalization performed on EAS did not alter the algorithm and its performance, therefore a comparison between N-EAS and other ACO algorithms was considered unnecessary since there are other studies comparing EAS with other algorithms. It should be highlighted as a conclusion the fact that the foremost objective of this initiative was achieved, meaning, spearing computational time used by pre-tests in order to determine a good set of algorithm's parameters values.

R E F E R E N C E S

[1] *J. Brownlee*, Clever Algorithms: Nature-Inspired Programming Recipes, Brisbane: Creative Commons Australia, 2011.

[2] *D. Gordon*, “Control without hierarchy,” Nature, vol. 4468, p. 143, 2007.

[3] *T. D. e. a. Seeley*, "Group decision making in honey bee swarms," American Scientist, vol. 94, p. 220–229, 2006.

[4] *A. Moallem and S. Ludwig*, “Using artificial life techniques for distributed grid job scheduling,” in Proceedings of the 2009 ACM symposium on Applied Computing, 1091-1097, 2009.

[5] *D. De Rango and A. Socievole*, "Meta-heuristics techniques and Swarm Intelligence," Mobile Ad-hoc Networks: Applications, pp. http://www.intechopen.com/books/mobile-ad-hoc-networks-applications/meta-heuristics-techniques-and-swarm-intelligence-in-mobile-ad-hoc-networks, 2011.

[6] *P. Grassé*, "The automatic regulations of collective behavior of social insect and "stigmergy"," Journal of Psychology and Normative Pathology, vol. 57, pp. 1-10, 1960.

[7] *M. Dorigo, V. Maniezzo and A. Colorni*, "The Ant System. Optimization by a colony of coorperating agents," IEEE Trans. on Systems, Man and Cybernetics, Part B, vol. 26, no. 1, pp. 29-41, 1996.

[8] *M. Dorigo*, “Optimisation, Learning and Natural Algorithms,” Politecnio di Milano, Milano, 1992.

[9] *E. Bonabeau, M. Dorigo and G. Theraulaz*, Swarm Intelligence. From Natural to Artificial Systems, New York: Oxford University Press, 1999, p. 9.

[10] *W. Gutjahr*, "Ant colony optimization: recent developments in theoretical analysis," Theory of Randomized Search Heuristics, pp. 225-254, 2011.

[11] *C. Blum*, "Ant colony optimization: Introduction and recent trends," Physics of Life Reviews, vol. 2, p. 353–373, 2005.

[12] *M. Dorigo*, "Phd Thesis: Optimization, learning and natural algorithms," Dipartimento di Elettronica, Politecnico di Milano, Milano, 1992.

[13] *T. Stützle and H. Hoos*, "The MAX –MIN ant system and local search for the Travelling Salesman Problem," Proceedings of IEEE International Conference on Evolutionary Computation and Evolutionary Programming, pp. 209-314, 1997.

[14] *M. Dorigo, G. di Caro and L. Gambardella*, "Ant algorithms for discrete optimization," Artificial Life, vol. 5, no. 2, pp. 137-172 , 1999.

[15] *B. Bullnheimer, R. Hartl and C. Strauss*, "A new rank-based version of the Ant System: A computational study," Central European J Operations Res Econom, vol. 7, no. 1, pp. 25-38, 1999.

[16] *M. Dorigo and L. Gambardella*, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans Evolutionary Computing, vol. 1, no. 1, pp. 53-66, 1997.

[17] *T. Stützle, M. Lopez-Ibanez, P. Pellegrini, R. Maur, M. Montes de Oca, M. Birtarri and M. Dorigo*, "Parameter Adaptation in Ant Colony Optimization," IRIDIA - Technical Report Series, vol. 002, pp. 1-24, 2010.

[18] *S. Negulescu, I. Dzitac and A. Lascu*, "Synthetic Genes for Artificial Ants. Diversity in Ant Colony Optimization Algorithms," International Journal of Computers, Communication & Control, vol. 5, no. 2, pp. 216-223, 2010.

[19] *F. Hutter, H. H. Hoos and K. Leyton-Brown*, "Identifying Key Algorithm Parameters and Instance Features using Forward Selection," in Proceedings of the 7th International Conference on Learning and Intelligent Optimization, Berlin, Heidelberg, 2013.