University of Reading

ACO for continuous function optimization: a performance analysis

Conference or Workshop Item

Accepted Version

Ojha, V. K. ORCID: https://orcid.org/0000-0002-9256-1192, Abraham, A. and Snášel, V. (2014) ACO for continuous function optimization: a performance analysis. In: 2014 14th International Conference on Intelligent Systems Design and Applications, 28-30 November 2914, Okinawa, Japan, pp. 145-150. Available at https://centaur.reading.ac.uk/82151/

It is advisable to refer to the publisher's version if you intend to cite from the work. See Guidance on citing.

Published version at: https://doi.org/10.1109/ISDA.2014.7066253

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.

www.reading.ac.uk/centaur

CentAUR

University of Reading

Central Archive at the University of Reading

Reading's research outputs online

ACO for Continuous Function Optimization: A Performance Analysis

Varun Kumar Ojha*, Ajith Abraham*,Václav Snášel*
*IT4Innovations, VŠB Technical University of Ostrava, Ostrava, Czech Republic
varun.kumar.ojha@vsb.cz, ajith.abraham@ieee.org, vaclav.snasel@vsb.cz

***Abstract*—Performance of the metaheurisitc algorithms are often subjected to their parameter tuning. An appropriate tuning of the parameters drastically improves the performance of a metaheuristic. The Ant Colony Optimization (ACO), a population based metaheuristic algorithm inspired by the foraging behaviour of the ants, is no different. Fundamentally, the ACO depends on the construction of new solution variable by variable basis using a Gaussian sampling with a selected variable from an archive of solutions. The selection of variable for sampling plays a vital role in the performance of the ACO. Present study offers a performance analysis of the ACO based on the tuning of its underling parameters such as selection strategy, distance measure metric and pheromone evaporation rate. Performance analysis suggests that the Roulette Wheel Selection strategy enhance the performance of the ACO due to its ability to provide non uniformity and wider coverage of selection in terms of selection of a solution for sampling. On the other hand, distance measure metric, 'Squared Euclidean' offers better performance than its competitor distance measure metrics. It is observed from the analysis that the ACO is sensitive towards the parameter evaporation rate. Therefore, a proper choice may improve its performance significantly. Finally, a comprehensive comparison between ACO and other metaheuristics suggested that the performance of the improvised ACO surpasses its counterparts.**

***Index Terms*—Metaheuristic; Ant colony optimization; Continuous optimization; Performance analysis;**

I. Introduction

Ant Colony Optimization (ACO) is a population based metaheuristic optimization algorithm inspired by the foraging behaviour of ants was initially proposed for solving the discreet optimization problems. Later, it was extended for the optimization of continuous optimization problems. In present study, we shall examine the strength and weakness of the classical ACO algorithm for function optimization. In present paper we shall follow the improvised version of the Ant Colony Optimization with an abbreviation ACO. The performance of the classical ACO is fundamentally dependent on the mechanism of the construction of new solutions variable by variable basis where an n dimensional continuous optimization problem has n variables. Therefore, the key to success of the ACO is in its construction of a new solutions. To construct a new solution, a variable from an available solutions archive is selected for a Gaussian sampling in order to obtained a new variable. Therefore, n newly obtained variable construct a new solution. The other influencing parameter to the performance of the ACC is distance measure metric, which is used for computing the average distance between i^{th} variable from a selected solution and the i^{th} variables of the other solutions in an available solutions set (solution archive). In the distance computation, the parameter pheromone evaporation rate plays an important role. In present study we shall analyse the impact of the mentioned parameters on the performance of the ACO. We shall also analyse and compare the performance of the improvised ACO with the other classical metaheuristic algorithms.

We shall explore the minute details of the ACO for the optimization of the continuous optimization problems in section II. A comprehensive performance analysis based on the underlying parameters of the ACO is provided in section III followed by a discussion and conclusion in sections IV.

II. Continuous Ant Colony Optimization (ACO)

The foraging behaviour of the ants inspired the formation of a computational optimization technique, popularly known as Ant Colony Optimization. Deneubourg *et al.* [1] illustrated that while searching for food, the ants, initially randomly explore the area around their nest (colony). The ants secrete a chemical substance known as pheromone on the ground while searching for the food. The secreted pheromone becomes the means of communication between the ants. The quantity of pheromone secretion may depend on the quantity of the food source found by the ants. On successful search, ants returns to their nest with food sample. The pheromone trail left by the returning ants guides the other ants to reach to the food source. Deneubourg *et al.* in their popular double bridge experiment have demonstrated that the ants always prefer to use the shortest path among the paths available between a nest and a food source. M. Dorigo and his team in early 90's [2], [3] proposed Ant Colony Optimization (ACO) algorithm inspired by the foraging behaviour of the ants. Initially, ACO was limited to discrete optimization problems only [2], [4], [5]. Later, it was extended to continuous optimization problems [6]. Blum and Socha [7], [8] proposed the continuous version of ACO for the training of neural network (NN). Continuous ACO is a population based metaheuristic algorithm which iteratively constructs solution. A complete sketch of the ACO is outlined in Figure 2. Basically, the ACO has three phases namely, Pheromone representation, Ant based solution construction and Pheromone update.

1) Pheromone Representation: Success of the ACO algorithm lies in its representation of artificial pheromone. The whole exercise of the ACO is devoted to maintain its artificial pheromone. The artificial pheromone represents a solution

S_1	s_1^1	s_1^2	$\dots$	s_1^i	$\dots$	s_1^n	$f(s_1)$	ω_1
S_2	s_2^1	s_2^2	$\dots$	s_2^i	$\dots$	s_2^n	$f(s_2)$	ω_2
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
S_j	s_j^1	s_j^2	$\dots$	s_j^i	$\dots$	s_j^n	$f(s_j)$	ω_j
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
S_k	s_k^1	s_k^2	$\dots$	s_k^i	$\dots$	s_k^n	$f(s_k)$	ω_k
	g^1	g^2		g^i		g^n		

Fig. 1. A typical Solution Archive/Pheromone Table. In a sorted solution archive for a minimization problem, the function value associated with the solutions are $f(s_1) \ll f(s_2) \ll \dots \ll f(s_k)$ therefore the weight associated with the solutions are $\omega_1 \gg \omega_2 \gg \dots \gg \omega_k$. The weight indicated the best solution should have highest weight. For the construction of new solution n Gaussian are sampled using a selected μ from the archive.

archive of a target problem. Socha and Dorigo [**?**], illustrated a typical representation of solution archive given in Figure 1. The solution archive shown in Figure 1 contains k number of solutions, each of which has n number of decision variables. In the case of a n dimensional benchmark optimization problem, variables in solution S_j indicates the variables of the optimization problem. Whereas, in the case of neural network (NN) training, a phenotype to genotype mapping is employed in order to represent NN as a vector of synaptic weights [9], [10]. Therefore, a solution S_j in the archive represent a vector of synaptic weights or a solution vector. A solution vector is initialized or created using random value chosen from a search space defined as $\mathcal{S} \in [min, max]$. I case of NN, min is defined as $x - \triangle x$ and max is defined as $x + \triangle x$, where x is set to 0. In the case of discrete version of ACO, a discrete probability mass function is used whereas, in case its continuous version, a continuous probability density function is derived from pheromone table. The probability density function is used for the construction of m new solutions. These m new solutions are appended to initial k solutions and then from $k + m$ solutions m worst solutions are removed. Thus, the size of solution archive is maintained to k.

2) Ant Based Solutions Construction: New solution is constructed variable by variable basis. The first step in the construction of new solution is to choose a solution from the set of solution archive based on its probability of selection. The probability of selection to the solutions in the archive is assigned using (1) or (2). The computation of the probability of selection a solution j given in (1) is based on the rank of the solution in the archive whereas, the probability of selection given in (2) is based on the fitness value of the solution. For the construction of the i^{th} ($i \in [1, n]$) variable of l^{th} (index into new solution set i.e. $l \in [1, m]$) solution, j^{th} ($j \in [1, k]$) solution from a solution archive is chosen based on its probability of selection as per (1) or (2). Several selection strategies may be adopted for the selection of j^{th} solution from a solution archive. The method of selection may be based on the probability of selection assigned based on the fitness function value or the weight assigned to the solution based on their rank. Hence, the probability of choosing j^{th} solution from the solution archive may be given as

$$p_j = \frac{\omega_j}{\sum_{r=1}^{k} \omega_r}, \tag{1}$$

or

$$p_j = \frac{f(s_j)}{\sum_{r=1}^{k} f(s_r)}, \tag{2}$$

where ω_j is a weight associated to the solution j computed as

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(rank(j)-1)^2}{2q^2k^2}}, \tag{3}$$

where q is a parameter of the algorithm. Since in (2) the smallest function value gets lowest probability, a further processing is required in order to assign highest probability to smallest function value. The mean of the Gaussian function is set to 1, so that the best solution acquire maximum weight. The $f(S_j)$ indicates the function value of the j^{th} solution. In case of the optimization problems, function value computation is straight forward whereas, in the case of the neural network training, the fitness of the solution is assigned using the Root Mean Square Error (RMSE) induced on NN for the given input training pattern (a given training dataset) [11]. The benchmark functions (including the RMSE computation of the benchmark dataset) are the minimization problems, therefore, the lower the value of a function higher the rank the solution in the solution archive. A detailed discussion on the selection methods is offered in section III-A.

Once the j^{th} solution is picked up, in the second step, it is required to perform a Gaussian sampling. A Gaussian distribution is given as

$$g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)}{2\sigma^2}}, \tag{4}$$

where μ is S_j^i and σ is the average distance between the i^{th} variable of the selected solution S_j and the i^{th} variable of the all other solutions in the archive. Various distance metrics adopted for computing the distance are comprehensively discussed in section III-B. The Gaussian sampling parameter σ may be expressed as

$$\sigma = \xi D_i, \tag{5}$$

where the constant $\xi > 0$, is a parameter of the algorithm, known as pheromone evaporation rate (learning rate) and D_i is the computed average distance between the selected solution and all the other solutions in the archive. For an example Manhattan distance D_2 may be given as

$$D_2 = \sum_{r=1}^{k} \frac{\mid S_r^i - S_j^i \mid}{k-1}. \tag{6}$$

procedure ACO($k, n, \Delta x, f(.), m, \xi, \epsilon$) ▷ $k \rightarrow$ Archive size, $n \rightarrow$ dimension, $f(.) \rightarrow$ objective function, $m \rightarrow$ # of new solution , $\xi \rightarrow$ evaporation rate and $\epsilon \rightarrow$ stopping criteria.
  for $i = 1$ to k **do**
    for $j = 1$ to n **do**
      $S_{ij} := rand(min, max)$
    end for
    $f_i = function(S_i)$ ▷ Compute function value.
  end for
  $S := Sorting(S)$; ▷ Sorting in ascending order of f.
  repeat
    for $l = 1$ to m **do**
      for $i = 1$ to n **do**
        Choose a solution S_{ji} according to probability of selection where $j \in [1, k]$.
        $\mu_i = S_{ji}$ and σ_i as per (5)
        $S'_{li} = \mathcal{N}(\mu_i, \sigma_i)$
      end for
      $f_l = function(S'_l)$
    end for ▷ m new solution constructed
    $S'' = S + S'$; ▷ $|S| = k$, $|S'| = m$ and $|S''| = k + m$, appending m new solution to k solutions.
    $S'' = Sorting(S'')$
    $S := S'' - S_m$; ▷ Truncate m poor solutions.
  until Stopping criterion is satisfied
  return f_0 ▷ Optimum function value.
end procedure

Fig. 2. Continuous Ant Colony Optimization (ACO)

3) Pheromone Update: In the final phase of the ACO, the m number of newly constructed solutions are appended to initial k solutions. The fitness of $k+m$ solutions are ordered in acceding sense. In the subsequent step, m number of worst solutions are chopped out from $k+m$ solution. Thus, the size of solution archive is being maintained to k. The complete discussion about the ACO is summed up in the algorithm given in Figure 2.

III. Performance Evaluation

The ACO algorithm mentioned in Figure 2, is implemented using Java programming language. The performance of ACO algorithm is observed against the tuning of various underlying parameters. The parameters taken into consideration for performance analysis are, (i) Selection strategies used for the selection of solution for Gaussian sampling (ii) Distance metric and (iii) pheromone evaporation rate. We shall test the algorithm over the benchmark function mentioned in the Table I. The expression of the benchmark functions are as follows

$$-a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right)+a+\exp(1), \tag{7}$$

$$\sum_{i=1}^{d}x_i^2, \tag{8}$$

$$\sum_{i=1}^{d}ix_i^2, \tag{9}$$

$$\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2], \tag{10}$$

$$10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)], \tag{11}$$

$$\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1, \tag{12}$$

$$\sum_{i=1}^{d}x_i^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^2, \tag{13}$$

$$(x_1-1)^2\sum_{i=1}^{d}i(2x_i^2-x_i-1)^2, \tag{14}$$

and

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}e_i^2} \tag{15}$$

where in (15) $e_i = (\hat{y}_i - y_i)$ is the difference between the target value y_i and predicted value $\hat{y}_i$ of a training dataset.

A. Selection Method

The selection of solution is critical to the performance of the ACO provided in Figure 2. We shall analyse, how the selection strategies influence the performance the ACO. Several selection strategies may be adopted for the selection of solutions which have probability of selection assigned as

TABLE I
THE BENCHMARK OPTIMIZATION PROBLEMS/FUNCTIONS CONSIDERS FOR THE EXPERIMENT IN ORDER TO EVALUATE THE PERFORMANCE OF ACO ALGORITHMS. THE EXPRESSIONS AND THE RANGE OF THE SEARCH SPACE ARE MENTIONED.

Function		Expression	Dim.	Range	$f(x^*)$
F1	Ackley	as per (7)	d	-15,30	0.0
F2	Sphere	as per (8)	d	-50,100	0.0
F3	Sum Square	as per (9)	d	-10,10	0.0
F4	Dixon & Price	as per (14)	d	-10,10	0.0
F5	Rosenbrook	as per (10)	d	-5,10	0.0
F6	Rastring	as per (11)	d	-5.12,5.12	0.0
F7	Griewank	as per (12)	d	-600,600	0.0
F8	Zakarov	as per (13)	d	-10,10	0.0
F9	abolone(RMSE)	as per (15)	90	-1.5,1.5	0.0
F10	baseball(RMSE)		170		0.0

per (1) (named as Weight) or (2) (named as FitVal). Now, we have Roulette Wheel Selection (RWS), Stochastic Universal Sampling (SUS) and Bernoulli Heterogeneous Selection (BHS) selection strategies available at hand. Therefore, each of these three selection strategy may be used to select an individual which has probability of selection assigned using either of the probability assignment method mentioned in (1) and (2). Therefore, we have six different strategies available at hand. The six selection strategies are namely, RWS(FitVal), RWS(Weight), SUS(FitVal), SUS(Weight), BHS(FitVal) and BHS(Weight).

In Roulette Wheel Selection, the individuals occupy the segment of wheel. The slice (segment) of the wheel occupied by the individuals are proportional to the fitness of the individuals. A random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained. Each time an individual required to be selected, a random number $rand(0, 1)$ is generated and tested against the roulette wheel. The test verifies that the random number R fall in the span of which segment of roulette wheel. The individual corresponding to the segment to which the random number R belongs to is selected.

Unlike roulette wheel selection, the stochastic universal sampling uses a single random value to sample all of the solutions by choosing them at evenly spaced intervals. Initially, it is required to fix the number of candidates to be selected. Let, k be the number of solution need to be selected. For the selection of first candidate, the random number R in the case of SUS is generated in $[0, 1/k]$. For other candidates, random number $R_i := R_{i-1} + 1/k$ is obtained, where i indicates individual to be selected.

The Bernoulli Heterogeneous Selection (BHS) depend on the Bernoulli distribution [12] may be described as follows. For k independent variables representing the function value of solutions, where k indicates the number of solutions. Therefore, to select an individual, the BHS may act as follows. The Bernoulli distribution [12] is a discrete distribution having two possible outcomes labelled by $sel = 0$ and $sel = 1$ in which $sel = 1$ ("success") occurs with probability pr and $sel = 0$ ("failure") occurs with probability $qr = 1-pr$, where $0 < pr < 1$. Therefore, it has probability density function

$$Pr(sel) = \begin{cases} 1 - pr & \text{for } sel = 0 \\ pr & \text{for } sel = 1. \end{cases} \tag{16}$$

Accordingly, the outline of Bernoulli Heterogeneous Selection algorithm proposed in Figure 3.

The parameters setting for the performance evaluation of ACO based on selection strategy is as follows, the solution archive $k = 10$, $n = 30$, $m = 10$, $\xi = 0.5$, $\epsilon = 1000$ iterations and distance metric chosen is D2 (Manhattan). The experimental results of the various selection strategy is provided in Table II where the values are the mean of the functions F1 to F10 listed in Table I where the each function F_i has and its value computed over an average of 20 trials. In

1: **procedure** BHS(P) ▷ $P \rightarrow$ vector containing probability of selection of the individuals in a population k
2: **for** j = 1 to k **do**
3: $R := rand(0, 1)$ ▷ random value
4: **if** (select(R, p_j)) **then**
return Solution j is selected
5: **end if**
6: **end for**
7: **end procedure**
8: **procedure** SELECT(R, p_j)
9: **if** ($R < p_j$) **then**
return true
10: **else**
return false
11: **end if**
12: **end procedure**

Fig. 3. Bernoulli Heterogeneous Selection (BHS)

TABLE II
AN EXPERIMENTAL RESULT FOR THE PERFORMANCE EVALUATION OF SELECTION STRATEGIES. THE RESULTS INDICATED THE SUPERIORITY OF RWS METHOD OVER OTHER SELECTION STRATEGIES

Selection probability assignment	Selection Method			
	RWS	SUS	BHS	Rank 1
function fitness value based	28.202	101.252	34.931	41.725
weight computed based on rank	35.105	95.216	36.503	

other words, the each function values are the average over 20 distinct trials/instances and the final value given in Table I is the average of each function. Examining Table II, it may be observed that the RWS selection strategy outperform the other selection strategies. The RWS selection with the probability of selection computed based on function value yields best result among the mentioned selection strategies. The performance of BHS selection strategy is competitive to RWS selection. The result in table indicate the perform SUS is worst among the all mentioned selection methods. From Table II, it is interesting to note that performance over probability of selection based on function value performs better than its counterpart with an exceptional being SUS case.

To investigate the differences in the performance of the selection strategies indicated above, three hundred selections by each of the selection methods is plotted graphically in Figure 5. As per the algorithm outlined in Figure 2, to construct a single solution, variable by variable, we need to select a solution from the solution archive of size k individuals n number of times, where n is the number of variables in a solution. Hence, to construct m number of solutions each of which having n variables $m \times n$ selection is required. Figure 5 represents the construction of 10 new solution from a solution archive of size 10 and each solution in the archive is having 30 variables representing the dimension of a function

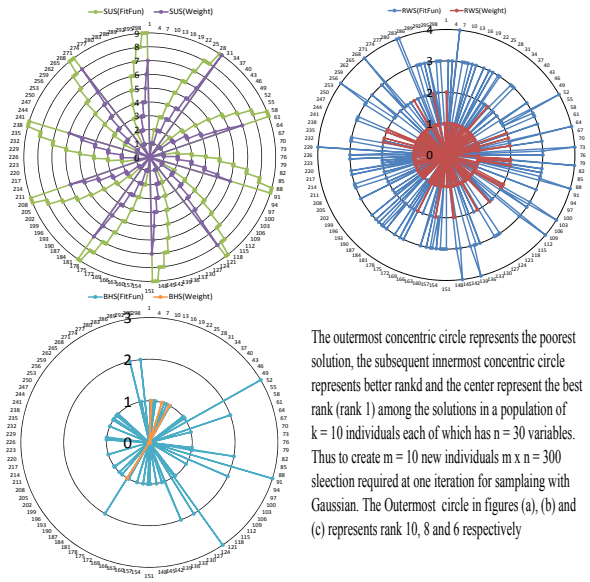

Fig. 4. Mapping of the individual selection according to their rank. The mapping indicate the span of rank coverage in selection and the distribution of selection. (a) top left - SUS(green - FitVal, purple - weight), (a) top right - RWS(indigo - FitVal, red - weight) and (a) bottom left - BHS(blue - FitVal, orange - weight)

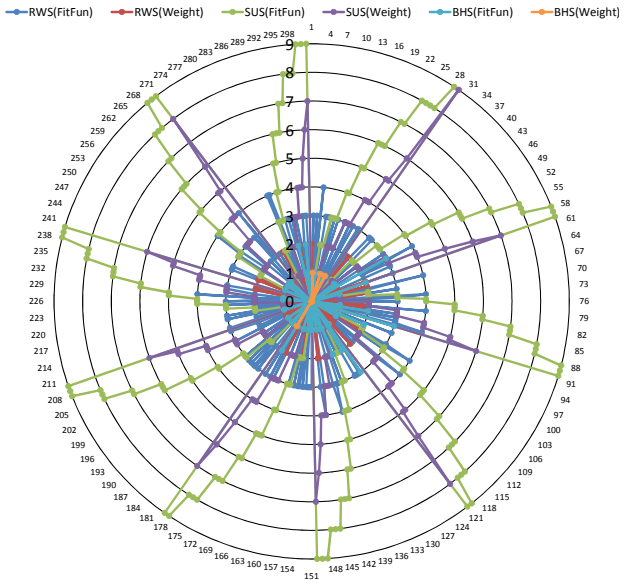

Fig. 5. A combined representation of the selection strategies. The center represent an individual with rank 1 and outermost circle represents an individual with rank 10.

F1. Therefore, $10 \times 30 = 300$ selection are made at one iteration. Figure 5 illustrate the mapping of selection made in a single iteration of the algorithm ACO used for optimization of function F1. In figure 5, ten concentric circles represent the solutions in the solution archive. The center of the circle (marked 0) indicate the solution with rank 1 (highest) while the subsequent outer concentric circle indicates the representation of increasing rank of solution. Therefore, the center indicated the best solution whereas the outermost concentric circle indicate the worst solution. Hence from the center to the outermost concentric circle each circle in increasing diameter represents 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 6^{th}, 7^{th}, 8^{th}, 9^{th} and 10^{th} ranked solution respectively.

The six selection strategies namely, RWS(FitVal), RWS(Weight), SUS(FitVal), SUS(Weight), BHS(FitVal) and BHS(Weight) are represented by indigo, red, green, purple, blue and orange coloured lines. Examining Figures 4(a) and 5, SUS(FitVal) and SUS(Weight) selection strategy selects the solutions uniformly in each of the new individual construction. The distribution of the selection (picking a solution) is uniform (from best to worst) throughout 300 selection. Therefore, the selection are repeat at a step of 30 (dimension). It may also be observed that the coverage of selection is distributed from the best to worst selection. The results provided in Table II indicates poor performance of SUS selection strategy. Therefore, an uniform selection with wider coverage of ranks happens to be poor selection strategy. In case of RWS(FitVal) and RWS(Weights) selection mapping provided in Figures 4(b) and 5 indicated a span of selection from rank 1 (center marked 0) to rank 5 (outermost circle marked 4) in case of function value and rank 1 to rank 3 in case of Weight. However, its selection is mostly distributed within the range of rank 1 to 4 in case of FitVal and rank 1 to 2 in case of Weight. It is worth mentioning that unlike SUS case, in RWS the selections is not uniform throughout 300 selections for construction of each of the 10 new solutions. The non uniform selection with coverage of selection to an adequate range of best to worst solution helps RWS selection strategy to achieve better performance over its competitor selection strategy. Similar to RWS, BHS selection also offer non uniform selection of individuals but on the contrary to the RWS its coverage of rank is mostly concentrated to fittest individual in the archive. From Figures 4(c) and 5, it may be observed that the BHS selection is non-uniform but its selection spans upto rank 3 among the 10 individuals whereas the RWS spans upto rank 5. From Figures 4(a), 4(b), 4(c) and 5, it may observed that probability of selection computed based on the weights indicated in purple (in Figure 4(a)), red (in Figure 4(b)) and orange (in Figure 4(c)) behaves similar to the probability of selection computed based on function value but, it tend to prefer selection towards best ranks. However, the results provide in Table II indicated the the preference of better rank in case of SUS offers better result than preferring each individuals in archive.

B. Distance Measure Metric

After the selection of a solution from an archive, another crucial operation in ACO algorithm is sampling of the selected solution. To sample the selected solution, parameter μ and σ need to be computed. As discussed in section II, the parameter μ is the i^{th} variable of the j^{th} solution selected for sampling and the parameter σ is computed as per expression (5) where distance D_i ($i \in [1, 10]$) is a distance metric listed in Table III. The distance computed between the selected solution (point)

TABLE III
AN EXPERIMENTAL RESULT OVER THE DISTANCE MEASURE METRIC INDICATED THAT SUPERIORITY OF SQUARED EUCLIDEAN DISTANCE OVER ALL OTHER DISTANCE METRICS

#	Distance Measure Metric		Mean Fun. Value
	Expression	Metric Name	
D1	$(\sum \lvert x_i - y_i \rvert^r)^{1/r}$	Minkowsky (r = 0.5)	28.792
D2		Manhattan (r = 1)	33.203
D3		Euclidean (r = 2)	44.578
D4		Minkowsky (r = 3)	45.211
D5		Minkowsky (r = 4)	51.909
D6		Minkowsky (r = 5)	53.702
D7	$\sum (x_i - y_i)^2$	Squared Euclidean	14.308
D8	$\max \lvert x_i - y_i \rvert$	Chebychev	93.642
D9	$\frac{\sum \lvert x_i - y_i \rvert}{\sum x_i + y_i}$	Bray Curtis	98.983
D10	$\sum \frac{\lvert x_i - y_i \rvert}{\lvert x_i \rvert + \lvert y_i \rvert}$	Canberra	103.742

with other solution (points) in the solution archive is critical to the performance of ACO algorithm. To compute distance between two points, usually the Euclidean distance metric is used. In general for computing distance between points (x_1, x_2, ..., x_n) and (y_1, y_2, ..., y_n) Minkowski distance of order r is used. The Euclidean distance is a spacial case of Minkowski distance metric, where r = 2. An experimental result over all the distance metric mentioned in the Table III. The experimental setup is as follows: $k = 10$, $n = 2$, $m = 10$, $\xi = 0.5$ and $\epsilon = 1000$ iterations. It may be noted that for the experimentation purpose the RWS selection with probability selection based on function value is used. Examining table III, it is found that the Squared Euclidean (D6) is performed better than all the other distance metric. However, it may also be noted that the performance of ACO decreases over the increasing order of 'r' of Minkowski metric.

C. Evaporation Rate

The parameter evaporation rate (ξ) in ACO algorithm is treated as learning rate. The performance evaluation of ACO based on the evaporation rate with the following parameter combination $k = 10$, $n = 20$, $m = 10$, and $\epsilon = 1000$ iterations is illustrated in Figure 6. It may be noted that for the experimentation purpose the RWS selection with probability selection based on function value is used and the distance metric D6 is chosen for the computation of σ. In Figure 6, the values along the vertical axis represents the mean functions F1 to F10 (where each function value is averaged over twenty trails) listed in Table I while the values along the horizontal axis represent evaporation rate ξ. The performance of ACO is evaluated by regulating the evaporation rate between 0.1 and 0.1. Investigating Figure 6, it may be observed that a valley shaped curve is formed. Initially, for $\xi = 0.3$, a high mean of function value is noted. While changing the value of ξ, a substantial improvement is being observed in the performance of the ACO. Hence, the performance of ACO is highly sensitive to the parameter ξ. It may be observed from Figure 6, that the increasing the value of ξ enhance the performance of ACO. However, the performance of ACO slightly declined on further increasing the value of ξ onward 0.5. A sudden high drop in performance if observed at the evaporation rate high sensitivity towards evaporation rate can be observed from the Figure 6.

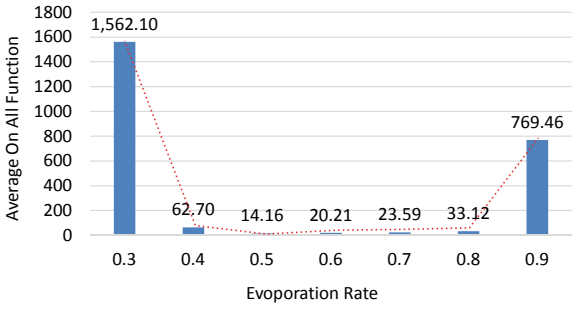

Fig. 6. An experimental results over different value of evaporation rate ξ. The results indicate a high sensitivity towards the parameter ξ.

D. Comparison with Other metaheuristics

An experiment conducted for the comparison between the improvised ACO and other classical metaheuristic algorithm such as Particle Swarm Optimization (PSO) and Differential Evaluation.

The parameter setting adopted for ACO is as follows: the population $k = 10$, $n = 20$, and $\epsilon = 1000$ iterations, evaporation rate ξ is set to 0.5, selection method for classical ACO is BHS(Weight) and distance metric is D2 (Manhattan) whereas for ACO* (improvised parameters) selection method is RWS(FitVal) and the distance metric is D6 (Squared Euclidean) is considered.

The PSO [13] is a population based metaheuristic algorithm inspired by foraging behaviour of swarm. A swarm is basically a population of several particles. The mechanism of PSO depends on the velocity and position update of a swarm. The velocity in PSO is updated in order to update the position of the particles in a swarm. Therefore, the whole population moves towards an optimal solution. The influencing parameters are cognitive influence C_1, social influence C_2 are set to 2.0 and $InertiaWeightHigh$ is set to 1.0 and $InertiaWeightLow$ is set to 0.0. The other parameter population size set to 10 and ϵ set to 1000 iterations.

The DE [14], [15] inspired by natural evolutionary process is a popular metaheuristic algorithm for the optimization of continuous functions. The parameter of DE such as $weightfactor$ is set to 0.7, the $corssoverfactor$ is set to 0.9 are the major performance controlling parameter. In present study the DE version $DE/rand-to-best/1/bin$ [16] is used. The other parameter population size set to 10 and ϵ set to 1000 iterations. Examining Table IV, it may be concluded that at the present mentioned experimntal/ parameter setup, the improvised version of ACO outperform the classical metaheuristics. However, from the present paper and study and the availability of no free lunch theorem [17], it is clearly evident that the mentioned meta-heuristcs are subjected to parameter tuning. Hence a claim of superiority of the present improvised ACO is subject to its comparisons parameter tuning of the other mentioned metaheuristics.

TABLE IV
AN EXPERIMENT CONDUCTED FOR COMPARISON BETWEEN CLASSICAL METAHEURISTICS ALGORITHMS. THE RESULTS INDICATE PERFORMANCE SUPERIORITY OF ACO ALGORITHM OVER OTHER ALGORITHM MENTIONED IN TABLE. **NOTE:** ACO INDICATES THE ORIGINAL VERSION OF ACO, ACO INDICATES IMPROVISED ACO, PSO - CLASSICAL PARTICLE SWARM OPTIMIZATION AND DE - DEFERENTIAL EVOLUTION.

Funtion	Test	ACO	ACO	PSO	DE
F1	$f(x^*)$	1.72	1.63	17.86	11.16
	var	0.05	0.01	3.80	13.92
F2	$f(x^*)$	0.69	0.02	7875.01	1610.96
	var	0.02	0.00	1.45E+07	3402.52
F3	$f(x^*)$	5.57	0.47	488.92	40.81
	var	0.73	4.17	2.00E+05	200.54
F4	$f(x^*)$	131.42	65.23	2.20E+05	2763.53
	var	4501.12	6160.92	9.35E+09	37334.53
F5	$f(x^*)$	127.56	32.24	81.27	22.27
	var	308.44	618.88	569.12	56.08
F6	$f(x^*)$	0.46	0.06	62.64	13.22
	var	0.27	0.01	712.26	39.88
F7	$f(x^*)$	4.93	12.72	458.32	44.71
	var	1.60	294.76	20947.18	149.05
F8	$f(x^*)$	11.68	1.05	36556.80	2162.60
	var	324.92	0.03	4.18E+09	13945.71

IV. DISCUSSION AND CONCLUSION

A comprehensive performance analysis of Ant Colony Optimization is offered in present study. The parameter such as selection strategy, distance measure metric and evaporation rate are put into meticulous tuning. The selection of a variable in construction of new solution. The assignment of the probability of selection to the individuals in the selection strategy influence the performance ACO. Analysing the results produce by the various selection strategy, it may be conclude that the selection strategy, RWS together with the probability of selection computed based on the function value offer better result than its counterparts. The advantages with RWS strategy is due to its ability to maintain non uniformity in selection and prefering not only the best solution in a population of individuals. Rather than sticking to Manhattan distance metric only, it is interesting to test several available distance measure metric for computing average distance between the selected solution and all the other solutions. It is observed from the experiments that the distance metric Squared Euclidean offer better performance among the mentioned distance metric in present study. It may also observed from the analysis that the ACO is highly sensitive towards its parameter, pheromone evaporation rate which control the magnitude of the average distance between the selection solution (variable) to all the other solution (individuals) in the population. A comparison between classical metaheuristic indicated the dominance of the ACO algorithm in present experimental setup. However, as evident from the present study and the no free lunch theorem, the metaheuristic algorithms are subjected to parameter tuning. Therefore, claim of superiority of one metaheuristic algorithm over other will always be under scanner.

ACKNOWLEDGMENT

This work was supported by the IPROCOM Marie Curie initial training network, funded through the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement No. 316555.

REFERENCES

[1] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of Insect Behavior*, vol. 3, pp. 159–169, 1990.

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

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

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. Vol 26, no. No. 1, pp. 29 – 41, 1996.

[5] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2, 1999, pp. –1477 Vol. 2.

[6] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research, ElSEVIER*, pp. 1155–1173, November 2006, dOI : 10.1016/j.ejor.2006.06.046.

[7] C. Blum and K. Socha, "Training feed-forward neural networks with ant colony optimization: An application to pattern classification," in *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*. IEEE, 2005, pp. 6–pp.

[8] K. Socha and C. Blum, "An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training," *Neural Computing and Applications*, vol. 16, no. 3, pp. 235–247, 2007.

[9] X. Yao, "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, vol. 8, no. 4, pp. 539–567, 1993. [Online]. Available: http://dx.doi.org/10.1002/int.4550080406

[10] A. Abraham, "Meta learning evolutionary artificial neural networks," *Neurocomputing*, vol. 56, no. 0, pp. 1 – 38, 2004.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994.

[12] E. W. Weisstein, "Bernoulli distribution," mathWorld–A Wolfram Web Resource http://mathworld.wolfram.com/BernoulliDistribution.html.

[13] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, 1995, pp. 39–43.

[14] R. Storn and K. Price, "Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces," 1995.

[15] ——, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 398–417, 2009.

[17] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67–82, 1997.