Heterogenous Adaptive Ant Colony Optimization with 3-Opt Local Search for the Travelling Salesman Problem

Ahamed Fayeez Tuani[1,2*], Edward Keedwell[1] and Matthew Collett[3]

[1] *College of Engineering, Mathematics and Physical Sciences, Harrison Building, University of Exeter*

[2] *Centre for Telecommunication Research Innovation (CeTRI),Faculty of Electronics & Computer Engineering, Technical University of Malaysia Malacca*

[3] *Animal Behaviour Laboratory, College of Life and Environmental Sciences, University of Exeter*

Abstract

The majority of optimization algorithms require proper parameter tuning to achieve the best performance. However, it is well-known that parameters are problem-dependant as different problems or even different instances have different optimal parameter settings. Parameter tuning through the testing of parameter combinations is a computationally expensive procedure that is infeasible on large-scale real-world problems. One method to mitigate this is to introduce adaptivity into the algorithm to discover good parameter settings during the search. Therefore, this study introduces an adaptive approach to a heterogeneous ant colony population that evolves the alpha and beta controlling parameters for ant colony optimization (ACO) to locate near-optimal solutions. This is achievable by introducing a set of rules for parameter adaptation to occur in order for the parameter values to be close to the optimal values by exploring and exploiting both the parameter and fitness landscape during the search to reflect the dynamic nature of search. In addition, the 3-opt local search heuristic is integrated into the proposed approach to further improve fitness. An empirical analysis of the proposed algorithm tested on a range of Travelling Salesman Problem (TSP) instances shows that the approach has better algorithmic performance when compared against state-of-the-art algorithms from the literature.

Keywords: Ant Colony Optimization, Self-adaptive, Heterogeneity, Behavioral Traits, Coevolution Ant Colony Optimization

1. Introduction

Most metaheuristic optimization algorithms require parameters to be set before the run in order to solve combinatorial optimization problems. These parameters can significantly affect the performance of the algorithm and researchers typically spend significant periods of time to tune these parameters based on experience or to implement parameter settings suggested by the literature. It is well documented that different problems or even different instances of the same problem requires different parameter settings [1]. Whilst the tuning of parameters is possible for benchmark problems, on larger real-world problems it is often not possible to conduct a thorough exploration of the parameter settings due to the computational complexity involved in the calculation of the objective function. Furthermore, throughout an optimization, different stages of the search process may require different exploration and exploitation strategies and so the parameters that appear optimal at the beginning may not be optimal towards the end. As an example in ant colony optimization, whenever stagnation occurs (e.g. where all ants construct the same tour in the TSP) exploration of the search space is preferred at this stage in order to escape from the local optima, whereas over-exploration can prevent convergence at all and so damage the performance of the algorithm.

It is also widely accepted that the ability to tune or control the parameters of the metaheuristics in part plays an important role in achieving a fast, acceptable result and the results are very dependent on the parameter settings too. Parameter tuning is a process of finding good parameter values before the actual run of the algorithm by using multiple (shorter) algorithm runs whilst varying parameters and monitoring performance. Meanwhile, in parameter control, an algorithm starts a trial with initial parameters that are modified or adapted using several strategies as the trial progresses. However, parameter tuning is a non-trivial task that requires a deep understanding of the algorithm in use as well as the problem being solved. In addition, parameter tuning for each and every problem is almost impossible as it is a time-consuming and computationally expensive process. Whilst the tuning of parameters is possible for benchmark problems, on larger real-world problems it is often not possible to conduct a thorough exploration of the parameter settings due to the computational complexity involved in the calculation of the objective function. Equally important in an optimization algorithm

is that the algorithm must be able to maintain its exploratory nature even after converging to a set of solutions in order to improve the overall performance by being able to continuously explore and exploit the search space efficiently especially when applied to large problem instances. Having said that, over-exploration, where the algorithm continuously explores new search space without really perturbing the solutions found or converging to an optimal solution, may occur and this undesirable scenario causes a waste of valuable computational resources and function evaluations. Hence, enabling the algorithm to learn and solve the problems via parameter control method using a self-adaptation strategy can alleviate the costly parameter tuning procedure as well as create a robust algorithm.

Self-adaptive approaches have been shown to work well in other meta-heuristics, however, little research has been conducted in regards to the analysis of parameter adaptation methods in Ant Colony Optimization (ACO). This work describes a heterogeneous adaptive ant colony approach with which uses an evolutionary algorithm during the optimisation to adapt the ACO meta-parameters. Standard ACO makes use of a population of homogeneous ants that all share the same parameter settings. In this work, both homogeneous and heterogeneous ant colonies were deployed to explore and exploit the parameter space and search space simultaneously to locate near-optimal solutions. The homogeneous ants are greedy and have high preference towards shorter paths while the heterogeneous ants have individual preferences or what are known as 'behavioral traits' with differential preferences either towards pheromone intensity or the next-hop heuristic. The novel approach of this study, discussed in the following sections, explores the synergistic effects of the adaptive evolutionary process and heterogeneity to allow convergence towards colony-level parameter setting through this self-adaptive approach indirectly enabling the algorithm to locate better solutions. The approach does not require problem-specific insights while population diversity is preserved by implementing a Gaussian mutation to the selected ants to prevent the algorithm from stagnation. The study suggests that the proposed approach is able to locate better solutions by exploiting neighbouring regions as well as improving the convergence speed. Comparison against state-of-the-art algorithms also indicates effectiveness and robustness of the heterogeneous adaptive approach when tested on standard TSP benchmarks.

This remainder of the paper is organized as follows. Section 2 discusses ACO from biological point of view to conventional ACO algorithms. Section 3 is an overview of classification of parameter tuning and parameter

control approaches while Section 4 introduces Heterogeneous Adaptive ACO (HAACO) algorithm. In Section 5, the system is tested on a range of TSP instances and some concluding remarks are presented in Section 6.

2. Conventional Ant Colony Optimization (ACO)

There are several successful, conventional ACO algorithms from Ant System to Ant Colony System (ACS) each with significant contributions to the ACO field of study. These algorithms use a metaheuristic approach to find good solutions for an optimization problem based on a homogeneous set of agents, namely ants with the same alpha and beta values. However, as ACO is intrinsically a distributed method, it lends itself well to the concept of heterogeneity where ants possess different traits as they do in natural systems. [2].

Year	**Algorithm**	**Author**
1992	Ant System	Dorigo[3]
1996	Max Min Ant System	Stutzle and Hoos [4]
1997	Ant Colony System	Dorigo and Gambardella [5]
1997	Rank Based Ant System	Bullnheimer et al. [6]
2000	Best-Worst Ant System	Cordon et al. [7]

Table 1: Successful Conventional ACO

Table 1 shows a selection of the most successful conventional ACO algorithms. The concept of ACO was introduced in early nineties by [3] with the algorithm Ant System (AS), based on the natural behaviour of an ant colony during foraging for food from its nest. [3] showed that the natural pheromone-following behaviour could be exploited within a metaheuristic search algorithm that was capable of finding the shortest path. The algorithm required agents, virtual ants, to traverse a graph, exploring paths using virtual pheromone and the shortest hop distance to guide them. On the completion of a route, the ants lay pheromone according to the optimality of the route traversed and the entire graph is subjected to a process of evaporation, weakening the pheromone globally. As this study uses Max-Min Ant System (MMAS) as the base algorithm, AS will not be discussed in detail here thus readers are encouraged to refer to [3] for further detail.

A further improved variant of AS known as Max Min Ant System (MMAS) was introduced by Stutzle and Hoos [4]. The first contribution of MMAS is

to limit the pheromone trails to a maximum and minimum values. The introduction of max-min bounds helps to prevent early convergence of sub-optimal level and also improves exploration of the search space. MMAS also implements an occasional pheromone trail re-initialization method to improve exploration whenever stagnation behaviour occurs. Secondly, only the best ant, either iteration-best or global-best ant, in each iteration can deposit pheromone. Empirical studies have indicated that increasing the frequency of global-best deposition can lead to improved performance. MMAS is the best conventional ACO algorithm Stützle et al. [8] when compared against elitist AS and ACS and so this study uses MMAS as the base algorithm to implement the heterogeneous adaptive approach. In MMAS and ACO more generally, the likelihood of an ant selecting a path is calculated accord to Equation 1. Of particular interest here are the alpha and beta parameters that give the weighting to the two components that determine the path desirability, pheromone trail intensity and local heuristic. Alpha and beta parameters of Equation 1 are modified to implement the heterogeneous adaptive approach as described later.

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \epsilon N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \tag{1}$$

This probabilistic rule implements the ratio between choosing node j against sum of all unvisited nodes from node i. τ_{ij} and η_{ij} are pheromone trail intensity and heuristic information of edge i to j respectively where η_{ij} is $1/d_{ij}$ with d being the distance of i to j. α and β are the parameters that determine the importance of pheromone or heuristic. If node j has already been visited, then the probability of going to node j is 0.

Equation 2 is used to deposit pheromone in every iteration where ρ represents the evaporation rate ($0 < \rho < 1$) while τ_{ij}^{best} is $1/L_{best}$ where L_{best} can be iteration-best tour length or global-best tour length which are considered during pheromone deposition. Readers are suggested to refer to [4] for a detailed explanation of MMAS.

$$\tau_{ij}(t) = \rho.\tau_{ij}(t-1) + \Delta\tau_{ij}^{best} \tag{2}$$

Other developed ACO methods include Elitist Ant System (EAS) [9], Ant-Q [10], Ant Colony System (ACS) [5], and ASRank [6] which each have strengths and weaknesses in comparison with the MMAS approach. Of closer interest to the proposed approach is Cordon et al. [7] that incorporates an

evolutionary algorithm method into ACO. The algorithm, as the name implies, allow the best ant to deposit pheromone while penalizing the worst ant by having additional pheromone evaporation on the worst path. Best Worst Ant System (BWAS) also implements pheromone trail mutation in order to enable diverse solutions. Lastly, as with MMAS, BWAS also implements pheromone trail re-initialization to τ_0 whenever the algorithm is stuck in local optima.

3. Previous work in Adaptive ACO

Eiben et al. [11] and Hinterding et al. [12] summarize and differentiate the parameter settings methods commonly used in Evolutionary Algorithms (EAs) into two main categories which are parameter tuning and parameter control. A self-adaptive algorithm, which is grouped under the parameter control methods, is defined as an algorithm where the parameters are encoded into of chromosomes and the selected parents undergo mutation and crossover operations to produce offspring. As this study focuses on the parameter control mechanism, previous work on parameter tuning will not be discussed here and readers can refer to [13] for further details.

A self-adaptation mechanism in ACO was introduced in [14] by applying crossover and mutation to evolve the parameters of α and β in relation to the transition rule based on the routing solutions found. The method was implemented on packet routing in communication network and improvements were noticed in system's performance over other methods. However, the efficacy of the approach is difficult to establish as it was not implemented on common benchmark problems. A similar adaptive approach was implemented in Botee and Bonabeau [15] using ACS as base algorithm and evolving as many as 11 parameters within a certain range of values compared to two parameters evolved in [14]. Although the authors indicated that improved performance in terms of final solutions as well as computational time were achieved from the proposed approach when tested on two small-scale TSP instances, one potential drawback, discussed in Pellegrini and Favaretto [16], is that the adaptation of so many parameters might hinder the performance of the algorithm. Moreover, the evolution of parameters in every iteration is unlikely to generate enough information about the performance of each ant when selecting the best and worst ant for replacement. Hence, this study allows the population to perform for several iterations before a decision is made based on their mean performance.

An adaptive method was proposed in [17] and [18] to adjust the pheromone deposition according to the fitness solution found to direct the algorithm towards better regions. According to the authors, this mechanism improves the convergence rate and prevents the algorithm from getting stuck in local optima. Pilat and White [19] proposed two parameter control methods that incorporate GA to ACS where the first method used GA to evolve a colony of ants where three parameters (β, ρ and $q0$) were encoded into the genome that represents an ant. The authors suggest that even though the first study did not improve on the base algorithm, it did give an insight into improvements that can be made as good solutions can be found within short period of time. Hence, the authors developed the second method that deploys a meta-level approach running GA alongside the ACS. The same three parameters as the first study were given a range of values and included in 4 randomly chosen ants in each iteration. Crossover, mutation and replacement occurs among the 4 ants only. The authors concluded that adapting the parameters is advantageous especially for limited function evaluations. In addition, the authors also indicated that each problem (each instance in the same problem) may require different optimal parameter settings in order to achieve optimal algorithmic performance. Gaertner [20] found that optimal parameter values are problem-dependant and thus introduced a hybrid ACO algorithm that is capable of automatically learning the optimal parameter values of a given TSP instance. The authors used a modified version of AS that incorporates the control parameter of ACS, $q0$ and an ant is initialized by a random parameter combination drawn from the given range. The author tested the proposed approach on a single 50 city TSP instance and the performance was slightly poorer in comparison to ACS with fine-tuned parameter settings. Another self-adaptive approach that used ACS as base algorithm was proposed by Yu et al. [21] who used the difference among tour lengths as an indicator for the α and ρ parameters to be adapted. Results indicate improved performance over ACS with fixed parameter settings. Yoshikawa [22] introduced a population of ants consisting of both normal and 'cranky' ants which prefer paths with lower pheromone values, the opposite of normal ants. The number of cranky ants in the population is controlled adaptively, so that when the algorithm is judged to be trapped in local optima, the number of cranky ants in the population is increased. The authors in [23] proposed an adaptive local search based ACO where the number of edges for swapping increases if local search discovers a better tour. The local search procedure is invoked for tours that are less than a minimum tour length set

by the authors and the edges for swapping is set at 2 before increasing linearly if better tour is found. One disadvantage of the proposed approach is that the algorithm requires a very high budget of function evaluations in order to achieve significantly improved performance. In 2010, Zhang and Lin [24] introduced an adaptive communication method for heterogeneous multiple ant colonies. After a fixed number of iterations, a sub-colony of ants with higher evolution co-efficient can choose the sub-colony to exchange information. Results indicate improved performance in the proposed approach when compared against the base algorithm. Interestingly, Pellegrini et al. [25] analyzes the performance of MMAS with both pre-determined (offline tuning) and parameter adaptation (online parameter control) schemes and suggested that the former method outperforms the latter when tested on TSP. However, it is well discussed in the literature that pre-determined parameter tuning method is time-consuming, computationally expensive and requires experience in selecting the parameter values as the mechanism involves trial and error method. Both Stutzle et al. [13] and Maur et al. [26] suggested that pre-scheduled parameter adaptation determined by a formula has a better performance compared to fixed, static parameter settings. [13] also suggest that a more detailed analysis is required to fully understand the contribution and advantage of adaptive methods as well as taking into account feedback from the current state of the algorithm. Jadon and Datta [27] proposed an ACO algorithm with an adaptive uniform mutation to escape from local optima. Comparison on several small-scale TSP indicates improved performance over other methods. Mavrovouniotis [28] proposed the adaptation of the pheromone evaporation rate as the search progresses to eliminate pheromone on poor path. The authors also suggested that the proposed adaptive scheme has better performance when compared to the fine-tuned algorithms. Li and Li [29] used mean information entropy in AS to adapt both α and β to control exploration and exploitation of the search space for construction time-cost optimization. The authors set a low α and high β to begin with and the α value increases and the β value decreases over time thus increasing the preference towards pheromone over heuristic information as the search progresses. However, it has been discussed earlier that an optimization algorithm may require different strategies at different stages of the search process therefore the modification of these values on a fixed schedule over time may not be an optimal approach. Ping et al. [30] introduced three sub-colony of ants consisting of ordinary ants that has a high probability in choosing path with high pheromone (high α), abnormal

ants that choose path with low pheromone (low α) and random ants that choose paths regardless of pheromone amount on the edges or path (zero α). Over time, the abnormal and random ants evolve and become ordinary ants adaptively over the search process, thus at the end, the whole population converges to ordinary ants. The authors implies that the heterogeneity introduced enabled the ants to locate better solutions while the adaptive approach allows the algorithm to escape from local optima. Most recently, Sun et al. [31] proposed a hybrid Particle Swarm Optimization (PSO)-ACO algorithm where PSO is used to find optimal α and β values which is then fed to the ACO algorithm to optimize task scheduling in cloud computing. Another study that implements a similar approach is proposed by Mahi et al. [32] who used PSO to optimize α and β parameters to improve the performance of ACO. The performance is further improved by a 3-opt local search. The authors modified the ACO algorithm to allow the ants to only use heuristic information in building their first tour and all ants were allowed to deposit pheromone based on the tour solution found. According to the authors, this approach allows the algorithm to start from a better solution compared to an ACO that starts with random tours. Therefore, this study also implements a similar approach by allowing greedy, homogeneous ants to explore the search space to locate better starting tours (explained later). A parallel ACO with 3-opt is proposed by Gülcü et al. [33] where the authors implemented several sub-colonies and created a master-slave communication paradigm for each sub-colony to share its best tour with other sub-colonies. The authors suggested that the parallel approach allows the algorithm to achieve good performance relatively quickly when tested on several TSP instances because of the distributed nature of the approach hence reducing the computational time. These two studies compare their approaches against a wide range of alternative approaches and are found to deliver state-of-the-art performance in the optimisation of the TSP using ACO and as such are used as the main comparators for the approach described here.

As a conclusion of the review, several variants of ants algorithm have been proposed to adapt various parameters to balance exploration and exploitation and to escaping from local optima. However, this often comes at a cost of additional function evaluations or additional computational processes that increases computational time. In contrast, we show that the heterogeneous adaptive approach described here is able to explore the parameter space to locate instance-optimal parameter settings that will improve the performance of the algorithm without incurring additional function evaluations.

Through comparisons with the state of the art algorithms shown above, we demonstrate that the algorithm also is capable of delivering state of the art performance via the introduction of Gaussian mutation mechanism.

4. Heterogeneous Adaptive ACO Methodology

4.1. Heterogeneity

Heterogeneity in ACO was introduced in [34] and [35] by modifying Equation 1 to incorporate the behavioural traits of each ant thus producing Equation 3. This provides each ant with a different perspective of the search space by introducing α_k and β_k which represents the individual behavioural traits that determine the perceived importance of pheromone and heuristics respectively. We have shown in our previous study [35] that heterogeneous ants with parameters randomly drawn from a normal distribution of traits has better performance when compared against that from a uniform distribution, thus this approach is followed here.

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha_k}[\eta_{ij}]^{\beta_k}}{\sum_{l \epsilon N_i^k}[\tau_{il}]^{\alpha_k}[\eta_{il}]^{\beta_k}} \tag{3}$$

4.2. Heterogeneous Adaptive ACO

The heterogeneous nature of the population of ants allows the initial population to explore the most promising areas of the search space initially but there is no additional mechanism to modify these as the search progresses. Here we consider the initial population of parameter settings across ants as the initial population for an evolutionary algorithm which will adapt the parameters throughout the optimization. Figure 1 illustrates the main idea of the proposed approach where the parameters adapted in this study are the α and β values that controls the relative importance of pheromone and heuristics respectively. As shown in Figure 2, each ant has its own 'behavioral traits' represented by the α and β values. The ants will have fitness values associated to them based on the tours they built and both mean best and mean worst ants are selected based on their mean fitness value over w iterations. Meanwhile, the use of elitism ensures that the population retains the fittest individual in the population and where the mean worst ant over w iterations is replaced by the child of the mean best ant that had undergone mutation. Through the use of selection, elitism and mutation, the EA can generate new promising parameter settings during the ACO run as well as

maintaining diversity in the population. To achieve this, the algorithm requires a representation and this is achieved here through the use of a floating point representation. It is worth noting that other encodings are possible, in particular binary encodings although the floating-point encoding here is preferred due to the ability to test fine-grained changes to the range from the mutation operation.

Algorithm 1 represents the pseudo code of HAACO which describes the introduction of both greedy, homogeneous and Gaussian heterogeneous ants as well as the adaptive mechanism. Once an offspring is produced from the process of mutation, the offspring then replaces the mean best ant and is included in the population for exploration and exploitation in next iteration.

4.3. Homogeneous Initialisation

In keeping with the comparator approaches in [32] and [33], an initialization phase using greedy, homogeneous ants was deployed for several iterations to act as a guide for the Gaussian heterogeneous ants to explore and exploit the search space. The greedy, homogeneous ants consist of ants with a relative importance of 0 towards pheromone (α=0) and a very high preference towards the heuristic (next-hop distance) (β=10) that indicates high β during early stages of the search process is desirable. This will allow the greedy,homogeneous colony to locate good solutions for the heterogeneous ants to exploit very early on rather than starting with random tours as per conventional ACO algorithms. An experiment was conducted to determine the number of iterations required for the colony of homogeneous ants to locate good solutions. Two different variants of the proposed algorithm were created where HAACO-5 is the adaptive approach with greedy, homogeneous ants that were deployed for 5 iterations while HAACO-10 is the same algorithm except that the greedy, homogeneous ants were deployed for 10 iterations. It should be noted that this number of greedy tours approximates those seen in the other approaches, so as to confer no advantage to either algorithm.

Figure 3a and 3b illustrate the performance of the HAACO with two different deployment approach as stated above. The result indicates that allowing the greedy, homogeneous ants to explore the search space for first 5 iterations based on a greedy approach instead of random initial tours produces good initial solutions. The analysis also suggests that enabling the homogeneous ants to explore the search landscape for 5 iterations produced the best performance. From iteration 6 onwards, the algorithm introduces a

Algorithm 1 Pseudocode of HAACO.

```
Input: Distance Matrix of TSP;
Initialize parameters;
Initialize ants:
for i = 1 : number of ants do
    AlphaHet(i) = mean α, s.d α;
    BetaHet(i) = mean β, s.d β;
end for
AlphaHo=0; BetaHo=10;
Start Iteration:
for it = 1 : Max Iteration do
    if it < 6 then
        Alpha=AlphaHo;
        Beta=BetaHo;
    end if
    if it == 6 then
        Alpha=AlphaHet;
        Beta=BetaHet;
    end if
    for k = 1 : number of ants do
        Position each ant on starting node;
        while TourSize < n + 1 do
            Tour Construction;
            3-opt local search;
            Adaptation mechanism;
        end while
    end for
end for
Update Solution;
Update Pheromone;
Pheromone Evaporation;
Check if termination condition is met;
if True then
    Go to End;
else
    Go to 10;
end if
End
```

heterogeneous population of ants randomly drawn from a distribution that will exploit the good regions found by the greedy homogeneous ants to further locate better solutions. The population then evolves as the search progresses where the worst ant in every w iterations are replaced with the child of the best ant in that same w iterations (explained in following section).

4.4. Adaptive Interval

As the HAACO approach does not employ additional function evaluations, the assessment of the quality of α and β parameter settings is based on the mean performance of ants with those parameter settings and a frequency of sampling of this information must be specified. Furthermore, a parameter must be defined to determine how often the evolution of the parameters takes place. Both of these factors are considered in the *adaptive interval (AI)* which is the number of ACO iterations that are completed between evolutionary steps. Setting this value is important because it determines how many evolutionary steps are possible within a given ACO run of fixed length and because it determines the robustness of the sampling that underpins the objective function calculation. For example, low values of AI (e.g. 1) indicate that parameter adaptation occurs in each ACO iteration. This provides many EA iterations but each one will be based on only one tour from each ant, leading to potentially volatile changes to the best performing ants through time. A more moderate setting such as 5 will yield 1/5 of the evolutionary steps, but based on a more robust sample of 5 tours generated by the ants. This parameter was analyzed across several TSPs: eil101.tsp, ch150.tsp and d198.tsp respectively and 5 was selected as the best performing of these.

Figure 4, 5a and 5b illustrate the boxplots of the proposed approach with different adaptive interval, w. The figures indicate that w=5 produce the best performance when compared against other adaptive interval that we have analyzed in this section. The results show that fast adaptation is preferable over slow adaptation mechanism where information over 5 iterations are used to determine or select the best and worst ant for replacement.

4.5. Mutation

Clearly, the algorithm requires a mechanism to explore the parameter space and to generate new parameter values for evaluation. Meanwhile, the algorithm also has to preserve the information that it has already gathered thus drastic alteration is not preferable. This suggests that low amount of

mutation is desirable to prevent total loss of the fittest individuals over time especially when small population size is used. In addition, the mutation operator is also capable of preventing premature convergence to non-optimal solutions by enabling the algorithm to escape from local optima. Therefore, this study implements an approach where the mean best ant over w iterations will undergo mutation to cause a small,random change to the genotype before replacing the mean worst ant. The offspring will then re-join the heterogeneous population and will then explore the parameter space to locate the instance-optimal parameter setting. The crossover operator is not used in this study as suggested in [36] that genetic algorithm with random mutation alone results in better performance of the algorithm when compared against that of crossover and inversion operator.

4.6. Uniform vs Gaussian Mutation Width

The degree of random change that each application of the mutation operator can cause is related to the width of the Gaussian distribution used. An empirical study was conducted to compare the performance of both uniform and Gaussian mutation operator in order to be implemented in the final setup of HAACO. Initial experiments were conducted to determine suitable range for both uniform mutation, $M = U[-a, a]$ and standard deviation, σ for Gaussian mutation with mean 0, $M = G(0, \sigma)$. Table 3 and Table 4 show the results of the initial experiments conducted where the range of $[-0.05, 0.05]$ produces the best performance for uniform mutation while HAACO with a Gaussian mutation with standard deviation of 0.05 has an overall best performance. In addition to the results, the extent to which the uniform mutation and the standard deviation, σ of the Gaussian mutation operator was explored, it was expected that a large mutation width (large range of values for uniform mutation or high σ for Gaussian mutation) will cause the algorithm to engage in excessive exploration thus mimicking a random walk and too small a width (small range of values for uniform mutation or low σ for Gaussian mutation) will cause the algorithm to converge to a local optimum too quickly and result in very little exploration of the parameter space. Hence, a suitable mutation width is required in order to achieve a good performance. The Gaussian mutation operator which is a commonly used operator, is used in this study due to high probability of creating an offspring that is much closer to the genes of the parents especially when elitism selection method is used.

$$\alpha_{meanworst} = \alpha_{meanbest} + M_1[-a, a]/M_1(0, \sigma) \quad (4)$$

$$\beta_{meanworst} = \beta_{meanbest} + M_1[-a, a]/M_2(0, \sigma) \quad (5)$$

The mean best ant which has individual α and β values will undergo mutation using the equations above where M_1 and M_2 are two random values based on the mutation distribution used in the experiments. These values are then added to the α and β of the mean best ant thus creating a child of the mean best ant which replaces the mean worst ant. Table 5 compares the performance of HAACO with both uniform and Gaussian mutation with optimal settings as discussed above. It can be seen clearly that HAACO with Gaussian mutation has an overall best performance in terms of best, average and worst best cost in almost all TSP instances when compared against HAACO with uniform mutation. Therefore, Gaussian mutation with $mean = 0$ and standard deviation, $\sigma = 0.05$ will be applied in all experiments hereafter unless stated otherwise.

5. Experimental Investigation of HAACO

5.1. Experimental Setup

An Intel Core i7 CPU-based computer with Windows 7 equipped with 4GB RAM was used to conduct the experiments and analysis. Matlab version R2015a was used to implement the base algorithm Max Min Ant system (MMAS) [4]. The results of the developed base algorithm were shown to closely replicate the performance to that of the original authors which can be referred to [34]. The travelling Salesman Problem (TSP) [37] was used as a benchmark problem using instances taken from TSPLIB. Each TSP instance contains a number of cities with associated x and y coordinates and the cost of travelling from one city to another can be calculated using the distance theorem. Therefore, the final tour cost produced by the proposed algorithm is the total distance travelled by the ants making a closed loop.

5.2. Experimental Results and Discussion

In this section, the results and analysis of the HAACO approach are presented. In particular, the ability for the approach explore and exploit the parameter space to locate instance-optimal parameter settings whilst generating competitive TSP tours is described. The proposed approach is compared against two state-of-the-art hybrid ACO algorithms for TSP in

[33], a parallel ACO algorithm with 3-opt and [32], a hybrid of PSO-ACO-3opt, as well as two variants of base algorithm, MMAS. MMAS1 is a variant of MMAS with 3 opt and greedy,homogeneous ants for first 5 iterations while MMAS2 represents a standard MMAS implemention also augmented with 3 opt. All parameters were set according to [33] and [32] except specific parameters such as mean α and mean β of heterogeneous ants which were set as 1 and 5 respectively. All experiments were stopped at 1000 iterations and iteration-best fitness solutions were improved by 3-opt local search procedure. All results quoted are from 20 independent trials unless stated otherwise.

Table 6 to 11 represent the performance comparison of HAACO against [33], [32] and MMAS variants in terms of best cost, average best cost and worst best cost respectively. A ranking mechanism was used to rank the performance of each algorithm where a ranking of 1 to 5 where 1 is the best algorithm in each TSP instance with 5 being the poorest. Referring to Table 6, HAACO has the best performance as it is capable of locating the optimum or overall best cost in 7/10 instances while it has the second-best cost in the other two TSP instances. Table 9 supports this claim by indicating the average ranking of each algorithm and HAACO has an average ranking of 1.4 followed by [33] in second with an average of 1.7. Table 7 shows the average best cost, corresponding to Table 10 which illustrates the results via the ranking system. Both HAACO and [33] have similar performance where both have lowest average best cost in 4/10 instances with an average ranking of 1.9. Lastly, Table 8 and Table 11 indicate that HAACO has a better performance in terms of worst best cost with an average ranking of 1.8 compared to 1.9 by [33]. Importantly, the Friedman statistical test based on the results in Table 5 with a 90% confidence interval indicates a p-value of 0.00159 thus suggesting significant difference in terms of the performance of the proposed algorithm compared to the other approaches. The improved performance in HAACO can be attributed to the capability of the algorithm to search the parameter space and quickly converge towards optimal parameter settings as well as exploiting the neighbouring regions via the Gaussian mutation. The greedy, homogeneous sub-colony was able to locate sub-optimal solutions very early in the search process and this acts as a guide for the Gaussian heterogeneous ants to exploit to locate better solutions. When considering a limited budget of function evaluations i.e 1000 iterations, this mechanism provides better starting solutions compared to a random start approach as used in most of ACO algorithms. Figure 6a and 6b show the comparison of the best cost with and without the use of

the greedy homogeneous ant colony. Figure 6a shows that the greedy ants enabled the algorithm to start with better, shorter tours very early on while Figure 6b illustrates that the algorithm starts off with random tours due to lack of pheromone information during the early stages.

Figure 7, 8 and 9 illustrate the parameter convergence using the convex hull and histogram to show the convergence of parameter values during the optimization. The convex hull forms the perimeter of most outer point in a Euclidean space while the histograms represent the number of ants within a particular range. From these figures it can be seen that the initial distribution of heterogeneous ants introduced from iteration 6 onward is followed by exploration and exploitation of the parameter space by the heterogeneous ants to locate instance-optimal parameter settings. It can also be noticed that in general, ants with high beta and low alpha values perform best. This depicts the exploration phase of the algorithm where the algorithm explores the fitness landscape to locate good to sub-optimal solutions. In addition to that, there is less information on the pheromone landscape for the ants to exploit hence the colony adapts to ants with high beta and low alpha values. The strategy changes as the search progresses when there is accumulation of pheromone on the edges hence ants become less reliant on the heuristics and utilizes the information on the pheromone landscape. Therefore, ants with higher alpha perform better in later stages hence the colony adapt to this. This shows that the proposed approach is capable of adapting its strategy between exploration and exploitation and at the same time exploring the parameter space to locate instance-optimal parameter settings. Figure 10a, 11a and 12a indicate that the algorithm quickly converges to good solutions before exploiting these parameter settings to achieve better solutions. The change from exploration phase to exploitation can be noticed in Figure 10b, 11b and 12b respectively where the algorithm has a high standard deviation early on before it converges to sub-optimal solutions and further perturbation is necessary to locate better solutions. The standard deviations also indicate that the proposed approach did not enter stagnation behaviour in all three TSP instances. However, the standard deviation of HAACO is almost 0 when tested on lin105.tsp as indicated in Figure 12b. The reason could be that the algorithm converges to the optimal solution that was found very early on during the search process. Lastly, the fast convergence claim is supported by the average lambda branching factor of HAACO in all three TSP instances where the number of branches being explored starts to decrease from iteration 200 onwards as can be seen in Figure 10c, 11c and 12c respectively.

This also indicates the nature of the algorithm which is capable of escaping from local optima as observed in Figure 10c and 11c where the algorithm still explores new edges even after a period of convergence and this can be due to the Gaussian mutation operator that leads the algorithm to new neighbouring regions. In addition, the proposed approach was tested on several large TSP instances in order to gauge the performance and the ability of the approach in exploring and exploiting a large search area. Table 12 and 13 both show the comparison results of HAACO against other approaches. The results are indicative that the proposed approach has a similar performance as compared to the other two approaches.

The advantage of the algorithm is in its ability to converge to good parameter settings quickly and exploit those values to obtain optimal settings. In addition, the small mutation width that was used in this study allows the ants to move to neighbouring regions rather than execute a large jump to non-optimal search space. Finally, it is important to note that this algorithm does not incur any additional function evaluation calls to implement the adaptive approach.

6. Conclusion

This paper introduces and performs experimentation with a new approach, HAACO which is tested on several TSP instances. Empirical studies have shown that adapting the parameters of an algorithm have an advantage over fine-tuned algorithms in their ability to react to the changing features of a search landscape as it approaches the optimum. The proposed approach is more feasible than the time consuming task of fine tuning the parameters that usually requires prior knowledge of the algorithm as well as the problem being solved. The proposed approach allows the algorithm to be able to adapt or interchange between different strategies i.e exploration and exploitation throughout the search process. Comparison against two state-of-the-art algorithms also suggest that the proposed approach is able to search the parameter space to locate the instance-optimal settings that would then allow the ants to explore the search space to find better fitness solutions. It is also noticeable that this approach suits one with a low or small budget of function evaluations as it is capable of converging to good or sub-optimal parameters quickly before exploitation of those areas to locate optimal settings. Future work requires the investigation of the approach on larger TSP instances. In conclusion, this study has explored the possibilities of enabling

the algorithm to self-adapt its α and β parameters thus allowing the algorithm to explore both parameter space and search space simultaneously to locate better solutions.

Acknowledgement

We would like to thank Faculty of Electronics and Computer Engineering (FKEKK), Technical University of Malaysia Malacca (UTeM) and the Ministry of Higher Education (MoHE) Malaysia for the financial support under the SLAB/SLAI program.

References

[1] A. Aleti, An Adaptive Approach to Controlling Parameters of Evolutionary Algorithms, Ph.D. thesis, Swinburne University of Technology, 2012.

[2] T. J. Czaczkes, C. Grüter, S. M. Jones, F. L. W. Ratnieks, Synergy between social and private information increases foraging efficiency in ants., Biology letters 7 (2011) 521–4.

[3] Dorigo, Marco., Optimization, learning and natural algorithms, PhD Thesis, Politecnico di Milano (1992).

[4] T. Stutzle, H. Hoos, MAX MIN Ant System and Local Search for the Traveling Salesman Problem, Ieee International Conference on Evolutionary Computation (Icec'97) (1997) 309–314.

[5] M. Dorigo, L. M. Gambardella, Ant colonies for the travelling salesman problem., Bio Systems 43 (1997) 73–81.

[6] B. Bullnheimer, R. F. Hartl, C. Strauu, A New Rank Based Version of the Ant System -A Computational Study, Central European Journal for Operations Research and Economics 7 (1) (1999) 25–38.

[7] O. Cordon, O. Cordon, I. F. de Viana, F. Herrera, L. Moreno, A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System, in: M. Dorigo, M. Middendorf, T. Sttzle (Eds.), ANTS 2002, IRIDIA, Universite @Libre de Bruxelles, Belgium, 2000, pp. 22–29.

[8] T. Stützle, H. H. Hoos, H. Hoos, Improvements on the Ant-System: Introducing the MAX-MIN Ant System Improving the Ant System: A Detailed Report on the MAX-MIN Ant System (1996).

[9] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 26 (1996) 29–41.

[10] L. M. Gambardella, M. Dorigo, Ant-Q: A Reinforcement Learning approach to the traveling salesman problem, Machine Learning Proceedings (1995) 252–260.

[11] A. E. Eiben, Z. Michalewicz, M. Schoenauer, J. E. Smith, Parameter control in evolutionary algorithms, Studies in Computational Intelligence 54 (2007) 19–46.

[12] R. Hinterding, Z. Michalewicz, A. E. Eiben, Adaptation in Evolutionary Computation: A Survey, Technical Report, ????

[13] T. Stutzle, M. Lopez Ibanez, P. Pellegrini, M. Maur, M. Montes De Oca, M. Birattari, M. Dorigo, T. Stützle, M. López-Ibáñez, Parameter Adaptation in Ant Colony Optimization, in: Autonomous Search, 2010, pp. 191–215.

[14] T. White, B. Pagurek, F. Oppacher, Connection Management using Adaptive Mobile Agents, in: H.R. Arabnia (Ed.), Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '98), pp. 802–809.

[15] H. M. Botee, E. Bonabeau, Evolving Ant Colony Optimization, Advances in Complex Systems 01 (2003) 149–159.

[16] P. Pellegrini, D. Favaretto, Quantifying the exploration performed by metaheuristics, Journal of Experimental & Theoretical Artificial Intelligence 24 (2012) 247–266.

[17] L. Yan-Jun, W. Tie-jun, An adaptive ant colony system algorithm for continuous-space optimization problems, Journal of Zhejiang University Vol.4 (2003) 40–46.

[18] H. N. Wang, S. Q. Sun, B. Liu, A Maturity-Based Adaptive Ant Colony Optimization Algorithm, Applied Mechanics and Materials 50-51 (2011) 353–357.

[19] M. L. Pilat, T. White, Using Genetic Algorithms to Optimize ACS-TSP, in: M. Dorigo, G. Di Caro, , M. Sampels (Eds.), Ant Algorithms. ANTS 2002, Springer, Berlin, Heidelberg, 2002, pp. 282–287.

[20] D. Gaertner, Natural Algorithms for Optimisation Problems Final Year Project Report, Masters thesis, Imperial College London, 2004.

[21] W.-j. Yu, X.-m. Hu, J. Zhang, R.-Z. Huang, Self-adaptive ant colony system for the traveling salesman problem, in: 2009 IEEE International Conference on Systems, Man and Cybernetics, IEEE, 2009, pp. 1399–1404.

[22] M. Yoshikawa, Adaptive Ant Colony Optimization with Cranky Ants, in: X. Huang, , S.-I. Ao, , O. Castillo (Eds.), Intelligent Automation and Computer Engineering, Springer Netherlands, 2010, pp. 41–52.

[23] M. R. HASSAN, M. M. ISLAM, K. MURASE, A New Local Search Based Ant Colony Optimization Algorithm for Solving Combinatorial Optimization Problems, IEICE Transactions on Information and Systems E93-D (2010) 1127–1136.

[24] P. Zhang, J. Lin, An adaptive heterogeneous multiple ant colonies system, in: Proceedings - 2010 International Conference of Information Science and Management Engineering, ISME 2010.

[25] P. Pellegrini, T. Stützle, M. Birattari, Off-line vs. On-line Tuning: A Study on MAX-MIN Ant System for the TSP, Springer, Berlin, Heidelberg, 2010, pp. 239–250.

[26] M. Maur, M. López-Ibáñez, T. Stützle, Pre-scheduled and adaptive parameter variation in MAX-MIN ant system, in: 2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010.

[27] R. S. Jadon, U. Datta, Modified ant colony optimization algorithm with uniform mutation using self-adaptive approach for travelling salesman

problem, in: 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), IEEE, 2013, pp. 1–4.

[28] M. Mavrovouniotis, Ant Colony Optimization in Stationary and Dynamic Environments, Ph.D. thesis, 2013.

[29] H. Li, P. Li, Self-adaptive ant colony optimization for construction time-cost optimization, Kybernetes 42 (2013) 1181–1194.

[30] G. Ping, X. Chunbo, C. Yi, L. Jing, L. Yanqing, Adaptive ant colony optimization algorithm, in: 2014 International Conference on Mechatronics and Control (ICMC), IEEE, 2014, pp. 95–98.

[31] W. Sun, Z. Ji, J. Sun, N. Zhang, Y. Hu, SAACO: A Self Adaptive Ant Colony Optimization in Cloud Computing, in: 2015 IEEE Fifth International Conference on Big Data and Cloud Computing, IEEE, 2015, pp. 148–153.

[32] M. Mahi, Ö. K. Baykan, H. Kodaz, A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem, Applied Soft Computing 30 (2015) 484–490.

[33] . Gülcü, M. Mahi, Ö. K. Baykan, H. Kodaz, A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem, Soft Computing 22 (2018) 1669–1685.

[34] A. Fayeez, E. Keedwell, M. Collett, H-ACO: A Heterogeneous Ant Colony Optimisation Approach with Application to the Travelling Salesman Problem, in: M. E. Lutton, E., Legrand, P., Parrend, P., Monmarché, N., Schoenauer (Ed.), EA2017, Springer International Publishing, Paris, France, 2017, pp. 149 – 162.

[35] A. T. I. Fayeez, E. Keedwell, M. Collett, Investigating Behavioural Diversity via Gaussian Heterogeneous Ant Colony Optimization for Combinatorial Optimization Problems, in: Proceedings of the 2nd International Conference on Advances in Artificial Intelligence - ICAAI 2018, ACM Press, New York, New York, USA, 2018, pp. 46–50.

[36] D. B. Fogel, J. W. Atmar, Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems, Technical Report, 1990.

[37] Reinelt, Gerhard, The traveling salesman : computational solutions for TSP applications, Springer-Verlag, 1994.

Table 2: Performance comparison of HAACO with several AI tested on several TSP instances. Results are of 20 trials, each trial = 1 000 iterations and those in bold indicates best of each category.

TSP	Opt	Best Cost			Average			Worst Cost		
		w=5	w=10	w=25	w=5	w=10	w=25	w=5	w=10	w=25
eil51	426	**426**	**426**	**426**	428.0	**427.5**	430.0	**430**	431	432
berlin52	7542	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**
st70	675	**675**	**675**	677	678.8	**678.2**	683.2	**683**	685	691
eil76	538	**538**	**538**	**538**	**541.5**	541.8	548.6	**545**	546	555
rat99	1211	**1211**	**1211**	**1211**	**1215.0**	1215.1	1217.9	**1224**	**1224**	1232
kroA100	21282	**21282**	**21282**	**21282**	21449.8	**21355.9**	21517.2	21680	**21457**	21792
eil101	629	**631**	632	**631**	**638.6**	639.5	646.4	**646**	648	657
lin105	14379	14426	14414	**14401**	14675.8	**14628.7**	14636.0	14922	**14863**	14910
ch150	6528	**6559**	6565	6577	**6588.2**	6591	6626.1	**6621**	6622	6680
kroA200	29368	29796	**29677**	29704	**30227.6**	30354.1	30320.4	30815	30916	**30810**

Table 3: Comparison of average best cost of HAACO with different uniform mutation range tested on several TSP instances where result in bold indicates best in each category. Results are of 20 trials, each trial = 1000 iterations.

Lower Bound	Upper Bound	Best			Average			Worst		
		eil51	kroA100	kroA200	eil51	kroA100	kroA200	eil51	kroA100	kroA200
-0.05	0.05	**426**	21416	**29499**	428.0	**21490.2**	**30023.4**	431	**21609**	**30664**
-0.075	0.075	**426**	**21355**	29601	428.7	21617.1	30058	434	21860	30833
-0.1	0.1	**426**	21416	29540	**427.7**	21531.4	30059.4	**430**	21877	30857
-0.2	0.2	**426**	21443	29524	428.7	21518	30107.9	433	21719	30967
-0.3	0.3	**426**	21379	29847	429.3	21587.5	30138.2	434	21768	30727
-0.4	0.4	**426**	21443	29603	430.4	21573.7	30150.2	434	21883	30991
-0.5	0.5	428	21393	29672	431.9	21604	30440.2	440	21899	31140

Table 4: Comparison of average best cost of HAACO with different Gaussian mutation width tested on TSP instances where result in bold indicates best in each category. Results are of 20 trials, each trial = 1000 iterations.

TSP	Opt	Heterogeneous Adaptive ACO (HAACO)							
		0.5	0.4	0.3	0.2	0.1	0.75	0.05	0.025
st70	675	690.6	689.4	684.0	681.7	680.4	682.8	**676.5**	679.4
ch150	6528	6649.8	6638.6	6608.6	6598.2	6596.2	6586.6	**6578.8**	6584.6
kroA200	29368	30708.1	30397.7	30525.4	30333.7	30084.0	29964.9	**29633.2**	29720.4

Table 5: Performance comparison of HAACO with uniform and Gaussian respectively on several TSP instances. Results are of 20 trials, each trial = 1 000 iterations and those in bold indicates best of each category.

TSP	Opt	Best Cost		Average		Worst Cost	
		Uni	Gau	Uni	Gau	Uni	Gau
eil51	426	**426**	**426**	428	**427.5**	431	**430**
berlin52	7542	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**
st70	675	**675**	**675**	679.9	**676.5**	684	**678**
eil76	538	**538**	**538**	542.6	**542.0**	547	**545**
rat99	1211	1212	**1211**	1214.7	**1214.1**	1220	**1218**
kroA100	21282	21330	**21282**	21554.4	**21364.2**	21828	**21445**
eil101	629	631	**630**	638.5	**632.5**	645	**635**
lin105	14379	**14379**	**14379**	14453.6	**14411.8**	14525	**14483**
ch150	6528	**6554**	6566	6581.7	**6578.8**	6599	**6595**
kroA200	29368	29604	**29483**	30150.4	**29633.2**	30973	**29755**

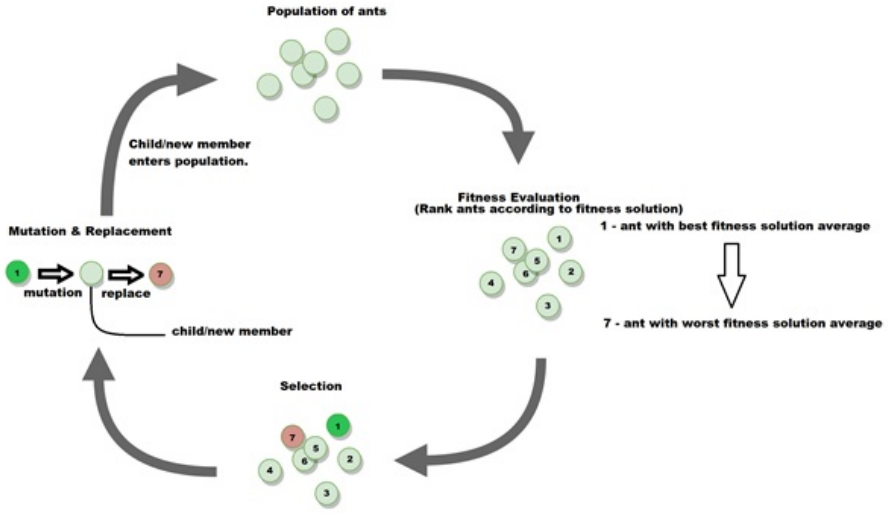


Figure 1: The iterative procedure of the proposed approach.

Table 6: Best cost comparison of HAACO against other approaches tested on several TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]	MMAS1	MMAS2
eil51	426	**426**	**426**	**426**	427	**426**
berlin52	7542	**7542**	**7542**	**7542**	**7542**	**7542**
st70	675	**675**	676	676	**675**	682
eil76	538	**538**	**538**	**538**	**538**	**538**
rat99	1211	**1211**	1213	1224	1212	1212
kroA100	21282	**21282**	**21282**	21301	21315	21379
eil101	629	630	**629**	631	631	631
lin105	14379	**14379**	**14379**	**14379**	**14379**	**14379**
ch150	6528	6566	6570	**6538**	6554	6566
kroA200	29368	29483	29533	**29468**	29485	29488

Table 7: Comparison of average best cost of HAACO against other approaches tested on several TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]	MMAS1	MMAS2
eil51	426	427.5	**426.35**	426.45	429.4	428.5
berlin52	7542	**7542**	**7542**	7543.2	**7542**	**7542**
st70	675	**676.5**	677.85	678.2	683.8	685.2
eil76	538	542.0	539.85	**538.3**	542.8	543.5
rat99	1211	**1214.1**	1217.1	1227.4	1216.9	1219.4
kroA100	21282	21364.2	**21326.8**	21445.1	21528.3	2121513.7
eil101	629	632.5	**630.55**	632.7	640.4	640.9
lin105	14379	14411.8	14393.0	**14379.15**	14429.2	14433.0
ch150	6528	6578.8	6601.4	**6563.95**	6603.9	6581.0
kroA200	29368	**29633.2**	29644.5	29646.05	29799.4	29760.3

Table 8: Comparison of worst best cost of HAACO against other approaches tested on several TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]	MMAS1	MMAS2
eil51	426	430	**427**	428	433	432
berlin52	7542	**7542**	**7542**	7548	**7542**	**7542**
st70	675	**678**	679	681	691	692
eil76	538	545	542	**539**	547	551
rat99	1211	**1218**	1225	1230	1226	1229
kroA100	21282	21445	**21382**	21554	21917	21810
eil101	629	**635**	639	638	651	650
lin105	14379	14483	14422	**14381**	14542	14594
ch150	6528	**6595**	6627	6622	6675	6617
kroA200	29368	29755	**29721**	29957	30307	30033

Table 9: Ranking comparison of HAACO against other approaches based on the best cost when tested on several TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]	MMAS1	MMAS2
eil51	426	1	1	1	2	1
berlin52	7542	1	1	1	1	1
st70	675	1	2	2	1	3
eil76	538	1	1	1	1	1
rat99	1211	1	3	4	2	2
kroA100	21282	1	1	2	3	4
eil101	629	2	1	3	3	3
lin105	14379	1	1	1	1	1
ch150	6528	3	4	1	2	3
kroA200	29368	2	5	1	3	4
Average Ranking		**1.4**	2	1.7	1.9	2.3

Table 10: Ranking comparison of HAACO against other approaches based on the average best cost when tested on several TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]	MMAS1	MMAS2
eil51	426	3	1	2	5	4
berlin52	7542	1	1	2	1	1
st70	675	1	2	3	4	5
eil76	538	3	2	1	4	5
rat99	1211	1	3	5	2	4
kroA100	21282	2	1	3	5	4
eil101	629	2	1	2	3	4
lin105	14379	3	2	1	4	5
ch150	6528	2	4	1	5	3
kroA200	29368	1	2	3	5	4
Average Ranking		**1.9**	**1.9**	2.3	3.8	3.9

Table 11: Ranking comparison of HAACO against other approaches based on the worst best cost when tested on several TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]	MMAS1	MMAS2
eil51	426	3	1	2	4	5
berlin52	7542	1	1	2	1	1
st70	675	1	2	3	4	5
eil76	538	3	2	1	4	5
rat99	1211	1	2	5	3	4
kroA100	21282	2	1	3	5	4
eil101	629	1	3	2	5	4
lin105	14379	3	2	1	4	5
ch150	6528	1	4	3	5	2
kroA200	29368	2	1	3	5	4
Average Ranking		**1.8**	1.9	2.5	4.0	3.9

Table 12: Comparison of best cost of HAACO against other approaches tested on several large TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]
rd400	15281	15603	**15578**	15594
fl417	11861	11960	11972	**11947**
pr439	107217	108730	**108482**	108530
pcb442	50778	**51780**	51962	52131

Table 13: Comparison of average best cost of HAACO against other approaches tested on several large TSP instances. Result in bold indicates best of each category.

TSP	Opt	HAACO	[33]	[32]
rd400	15281	15644.2	**15613.9**	15691.30
fl417	11861	**11979.5**	11987.4	11980.4
pr439	107217	108950.6	**108702**	108965.4
pcb442	50778	**52179.8**	52202.4	52368.1

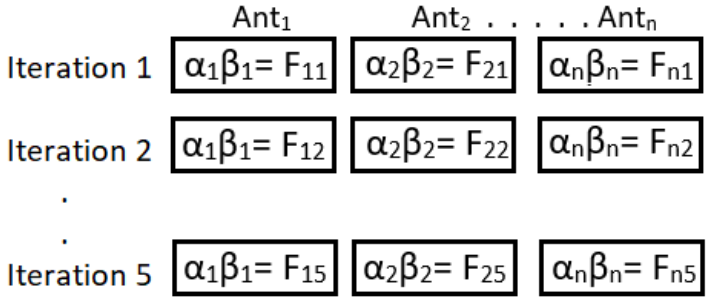
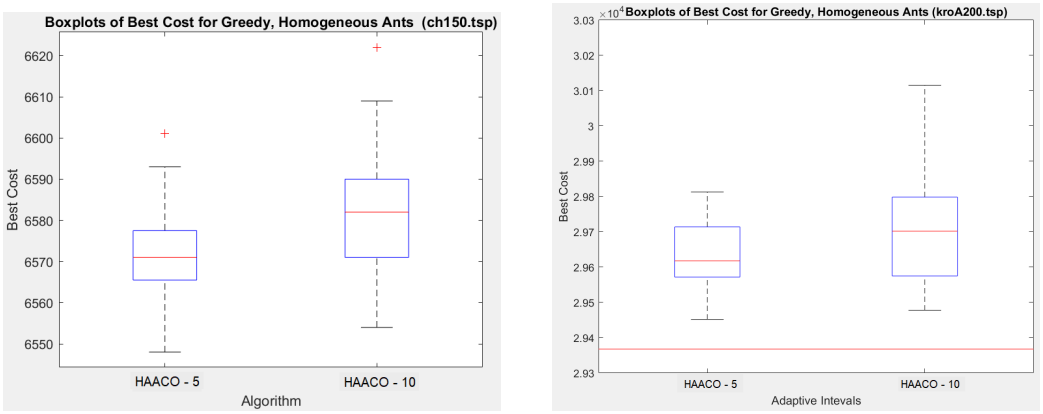


Figure 2: Selection of mean best ant and mean worst ant using elitism method.



(a) (b)

Figure 3: Boxplots representing best cost of HAACO-5 and HAACO-10 respectively when tested on ch150.tsp and kroA200.tsp.

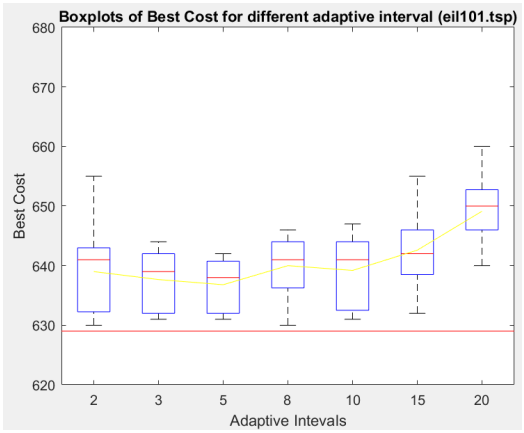
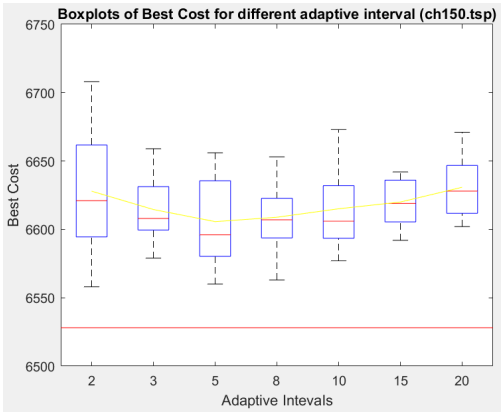
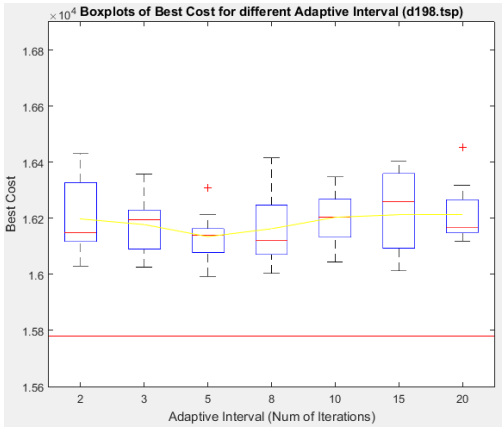


Figure 4: Boxplots representing best cost of HAACO with different adaptive interval tested on eil101.tsp. Yellow line indicates the average best cost while red line indicates the optimum for eil101.tsp

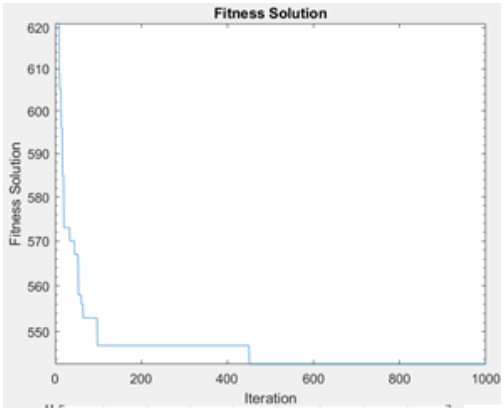


(a) ch150.tsp

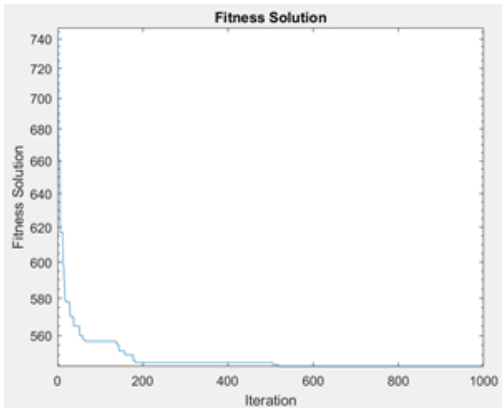


(b) d198.tsp

Figure 5: Boxplots representing best cost of HAACO with different adaptive interval tested on ch150 and d198.tsp. Yellow line indicates the average best cost while red line indicates the optimum for the TSP instances respectively



(a) Tour of HAACO



(b) Tour of het

Figure 6: Comparison of tours made by HAACO and heterogeneous ants when tested on eil76.tsp (Opt:538)

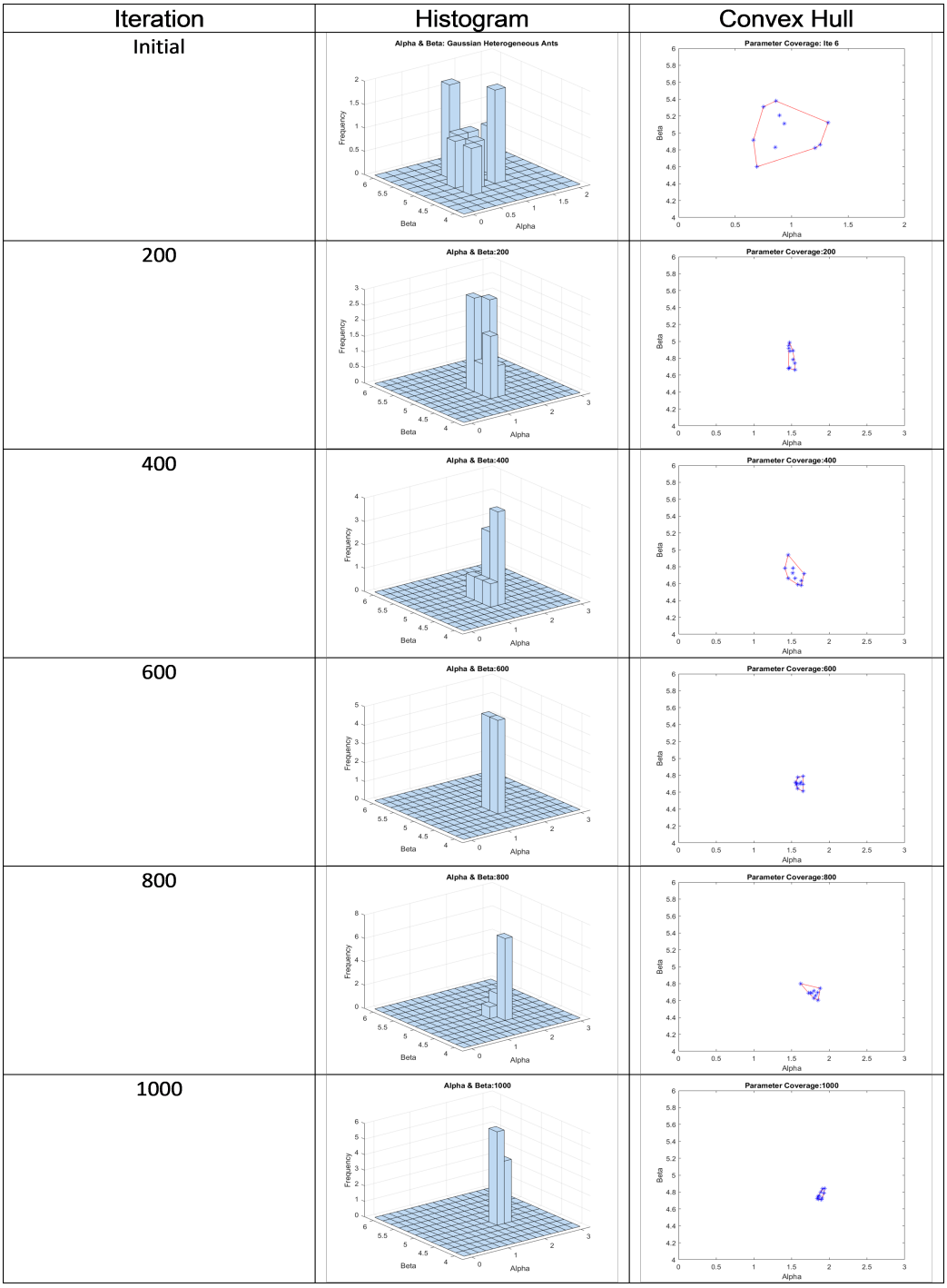


Figure 7: Parameter Convergence of HAACO tested on st70.tsp.

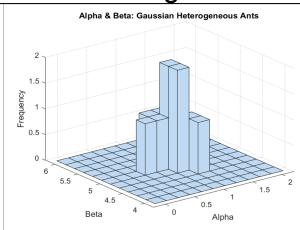
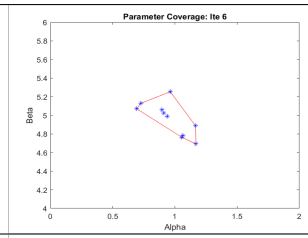
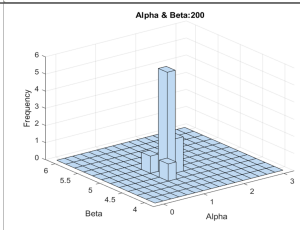
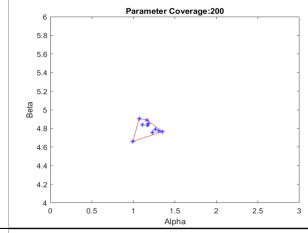
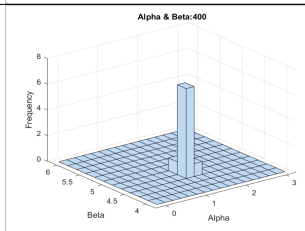
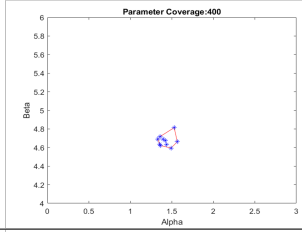
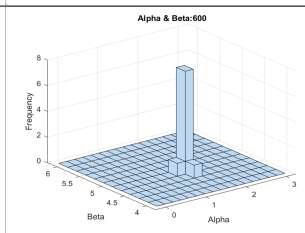
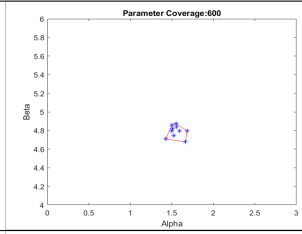
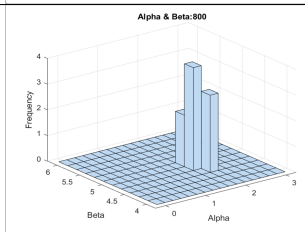
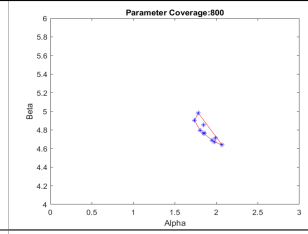
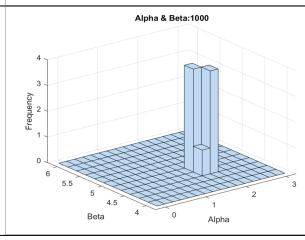
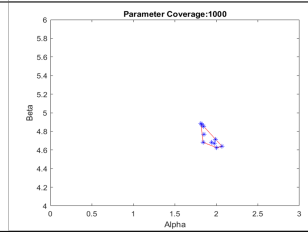
Iteration	Histogram	Convex Hull
Initial		
200		
400		
600		
800		
1000		

Figure 8: Parameter Convergence of HAACO tested on lin105.tsp.

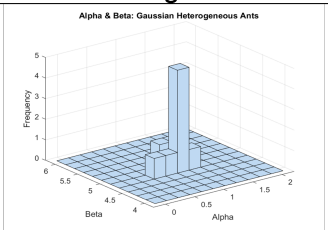
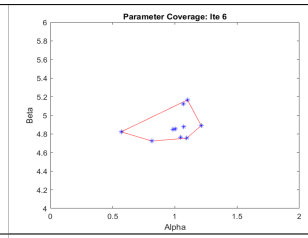
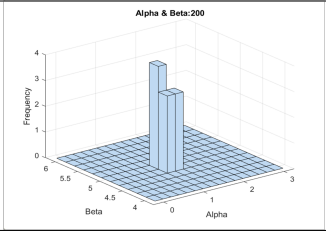
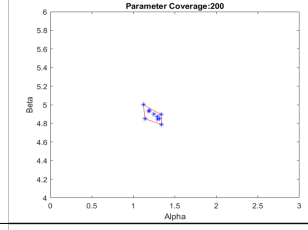
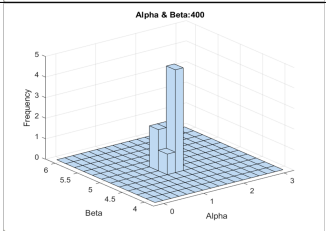
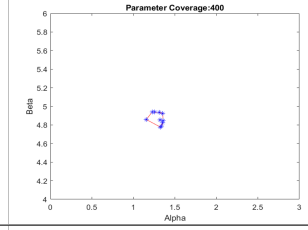
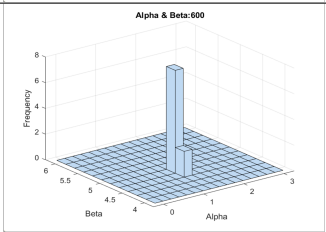
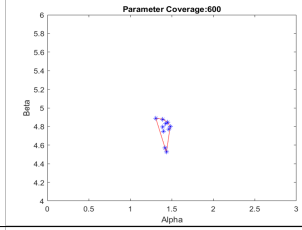
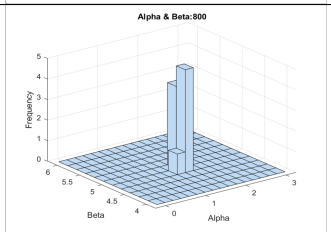
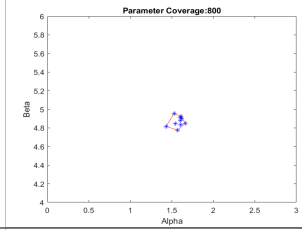
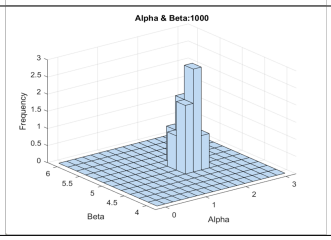
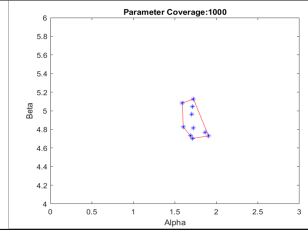
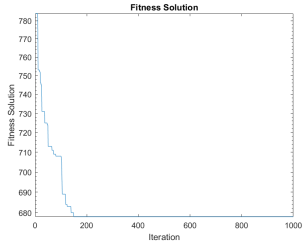
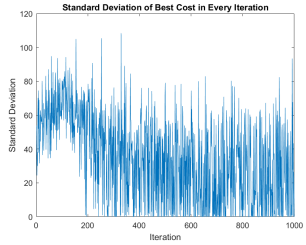
Iteration	Histogram	Convex Hull
Initial		
200		
400		
600		
800		
1000		

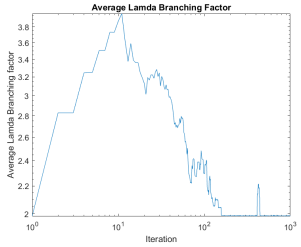
Figure 9: Parameter Convergence of HAACO tested on ch150.tsp.



(a) Convergence of HAACO: st70.tsp

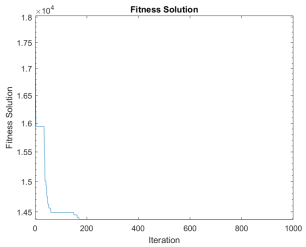


(b) Standard Deviation: st70.tsp

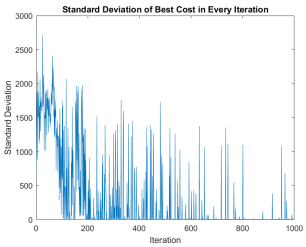


(c) Ave lamda branching factor: st70.tsp

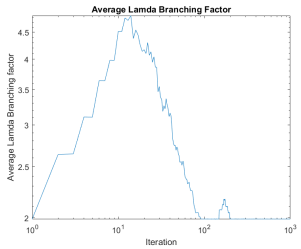
Figure 10: Analysis of HAACO tested on st70.tsp



(a) Convergence of HAACO: lin105.tsp

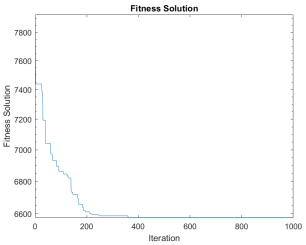


(b) Standard Deviation: lin105.tsp

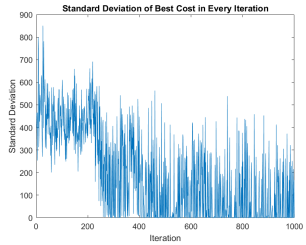


(c) Ave lamda branching factor:lin105.tsp

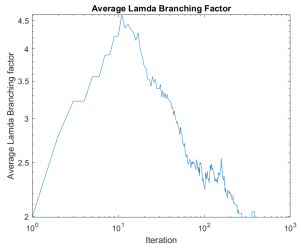
Figure 11: Analysis of HAACO tested on lin105.tsp



(a) Convergence of HAACO: ch150.tsp



(b) Standard Deviation: ch150.tsp



(c) Ave lamda branching factor:ch150.tsp

Figure 12: Analysis of HAACO tested on ch150.tsp