# An Artificial Ecosystem Algorithm Applied to Static and Dynamic Travelling Salesman Problems

Manal T. Adham
Department of Computer Science
University College London
Gower Street, London, WClE 6BT, U.K.
Email: M.Adham@cs.ucl.ac.uk

Peter J. Bentley
Department of Computer Science
University College London
Gower Street, London, WClE 6BT, U.K.
Email: P.Bentley@cs.ucl.ac.uk

***Abstract*—An ecosystem inspired algorithm that aims to take advantage of highly distributed computer architectures is proposed. The motivation behind this work is to grasp the phenomenal properties of ecosystems and use them for large-scale real-world problems. Just as an ecosystem comprises many separate components that adapt together to form a single synergistic whole, the Artificial Ecosystem Algorithm (AEA) solves a problem by adapting subcomponents of a problem such that they fit together and form a single optimal solution. AEA uses populations of solution components that are solved individually such that they combine to form the candidate solution, unlike typical biology inspired algorithms like GA, PSO, BCO, and ACO that regard each individual in a population as a candidate solution. Like species in an ecosystem, the AEA may have species of components representing sub-parts of the solution that evolve together and cooperate with the other species. Three versions of this algorithm are illustrated: the basic AEA algorithm, and two AEA with Species. These algorithms are evaluated through a series of experiments on symmetric and dynamic Travelling Salesman Problems that show very promising results compared to existing approaches. Experiments also show very promising results for the Dynamic TSP making this method potentially useful for handling dynamic routing problems.**

## I. Introduction

In natural biology the evolution of species does not take place in a vacuum. Millions of species may evolve together, all sharing the same environment. Many take competitive roles: plants trying to out-grow each other to reach the sunlight, or predators trying to out-hunt each other. Many take symbiotic roles: insects pollenating plants, animals eating fruits and distributing seeds. Indeed many forms of symbiosis are so complete that some species may not survive without the other.

Those co-evolving species that share their environments form an ecosystem - a self-regulating complex of interactions that share the common goal of long term survival in the environment. Although their day-to-day activities may sometimes seem mutually destructive, when a holistic view is taken it can be seen that each ecosystem is a carefully evolved balance between all species. Any temporary imbalance is quickly corrected: too many of one species is soon reduced by predation from another. Too few of one species results in reduction of others (e.g. lack of food/pollinators).

So an ecosystem is a combination of systems which have adapted to solve - and help each other to solve - a different aspect of one overall problem (survival in their shared environment) [1]. From a computational point of view, the ecosystem is therefore of great interest. In this work we propose a novel Artificial Ecosystem Algorithm (AEA) inspired by these ideas. The AEA solves a problem by adapting subcomponents of a problem such that they fit together and form a single optimal solution, akin to the way an ecosystem comprises many separate components that adapt together to form a single synergistic whole. Like the different species in an ecosystem, the AEA may have species of components representing subparts of the solution that evolve together and cooperate with each other. In this way the AEA is designed to take advantage of highly distributed computer architectures and adapt to changing problems.

This paper is organised as follows: the next section surveys different biology inspired algorithms that have been applied to TSP. Section 3 describes the proposed AEA algorithms. The experiments section compares the proposed AEA algorithms and analyses their performance on TSP and DTSP. The final section discusses findings.

## II. Background

There is no shortage of bio-inspired algorithms, often population-based and designed to find solutions to large, noisy or ill-defined problems by searching in parallel. Most approaches represent an entire candidate solution in each individual of the population and any attempts to distribute the algorithm across many processors will divide populations, not individual solution evaluations. However there are some classes of problems that can become so prohibitively large that it is more appropriate to divide the problem itself into smaller pieces and have separate processors work on these smaller, more tractable pieces. A classic example of such a problem is the Travelling Salesman Problem. The Travelling Salesman Problem is a NP-Hard problem and as such is a fundamental combinatorial optimisation problem [2]. The dynamic variant DTSP, where the number of cities available may change over time, is arguably even more difficult [3]. There have been numerous attempts to solve TSP using biology inspired algorithms.

The Genetic Algorithm (GA) was inspired by the concepts of natural selection and survival of the fittest [4]. Due to

the nature of traditional crossover and mutation operators, building valid solution permutations is difficult. To counteract these issues several alternatives have been proposed. Davis [5] proposes Ordered Crossover (OX), which chooses a subsequence of a tour from one parent and then preserves the relative order from the other parent. Goldberg and Lingle [6] define Partially Mapped Crossover (PMX) operation; this uses two crossover points that define the interchanging mappings. Exchange Mutation [7] is a 2-opt heuristic that exchanges positions of cities. There are many variations of GAs that make use of multiple populations through co-evolution, or that assemble smaller components of solutions together (e.g. classifier systems) [4]. From this perspective, the Artificial Ecosystem Algorithm presented in the next section can be regarded as a new type of multi-species, distributed classifier system.

Not all such algorithms evolve solutions. There are several algorithms inspired by the collective behaviour of decentralized and self-organized organisms. For example, Ant Colony Optimization (ACO) is based on the foraging behaviour of ant colonies [8]. Artificial ants build solutions and exchange information through an indirect communication mechanism (stigmergy [9]). Ant System (AS) [10] was originally used to solve TSP and a successful variant is Max-Min Ant System [11]. A different approach is taken by the Particle Swarm Optimisation Algorithm (PSO), which models the social behaviour of bird flocking. PSO optimizes problems through exchanging information between individuals. Particles are first randomly generated, they then move towards an optimal solution by adjusting their velocities and positions through multiple iterations. PSO has also been hybridised with ACO to allow improved performance [12]. Recently, Feng and Liao [13] proposed a fuzzy adaptive learning algorithm designed to overcome the shortcomings of PSO and is applied to large scale TSP.

In another insect-inspired approach, the Bee Colony Optimization Algorithm (BCO) models the collective foraging behaviour of honey bees. One of the pioneering works related to BCO is Bee System [14], which is a hybrid of BCO with GA, here BCO enables a local search and GA operates on a global search level. Wong *et al.* [15] uses BCO with 2-opt heuristic to solve TSP. Another example is the Firefly Algorithm (FA) [16]. Fireflies produce short rhythmic flashes of light that has two fundamental functions: to attract mating partners (communication) and to attract potential prey. Jati *et al.* [16] proposes an Evolutionary Discrete FA (EDFA) focussing on using a simple form of FA to solve TSP. The Bacterial Foraging Optimisation Algorithm (BFOA) was introduced by Good and Sahin, it mimics the foraging behaviour of E.coli bacteria as they search for energy such that they can maximise nutrient consumption. When applied to TSP BFOA found optimal solutions up to the max of 14 cities, but as more cities were added performance was degraded [17]. Yet another algorithm modelled on birds, the Cuckoo Search Algorithm (CS) [18], mimics the parasitic behaviour of female cuckoo birds who lays eggs on a host birds nest. Each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. CS has been adapted to solve TSP by introducing a new category of cuckoo's [19].

Many bio-inspired algorithms are based on a single species rather than multiple interacting species; chromosome in GA, birds in PSO, ant in ACO and so on. An ecosystem [1] represents a community of biotic living organisms that are organized into populations of species, the interactions between these species, and the interactions with its abiotic non-living environment. As described by McCormack [20] ecosystems have powerful properties that need to be exploited. There have been several simulations of ecosystems in the field of Artificial Life including Daisyworld, which illustrates the Gaia Hypothesis [21]; Tierra, which is an ecosystem of computer programs that compete for CPU time and access to main memory [22]; Avida, which is an evolutionary biology software platform and has been used to investigate many aspects of competitive and cooperative coevolution [23]. There also have been some studies that take inspiration from ecological systems. Kirley [24] models the effects of ecological environmental dynamics on a population. Kirley [25] proposes an evolutionary algorithm that is inspired from spatial interactions and disturbances (natural disasters) within an ecosystem, looking at how populations evolve in response to these events. Another example is Invasive Weed Optimization (IWO) [26] which takes inspiration from colonization of weeds as weed is naturally very robust and adaptive, changing its behavior in accordance with changes in the environment. Nevertheless, despite these works, to date there has been very little work on general purpose optimisation algorithms inspired by the functioning of an Ecosystem.

## III. Artificial Ecosystem Algorithms

In this section we introduce the Artificial Ecosystem algorithm. We first describe the basic AEA algorithm and how it has been applied to the TSP. Then we describe two variations of the AEAS: the AEAS (SOM) and AEAS (K-Means), and apply them to TSP and DTSP. Finally we apply AEAS to the DTSP in order to assess its ability to learn and adapt to changing problems.

The dynamic TSP provides an improved theoretical model of unpredictable and changing real world situations. Its changing infrastructure means that it resembles applications such as routing in mobile ad-hoc networks. In the dynamic TSP cities can be in an enabled or disabled state, if cities are enabled they can form part of a tour and vice versa. At specific intervals during the running of the algorithm, a city is selected at random, if its state is enabled then it is switched to disabled and vice versa.

For all versions of the AEA, the following definitions are valid:

1) An ***Environment*** is the physical non-living environment that Individuals will be interacting with, normally formulated as an optimisation problem to be solved.

2) An ***Individual*** represents one or more segments of the solution; connecting these segments forms a solution segment or complete solution. Each individual encapsulates a fitness value $F_i$ representing its quality, a flag $P_i$ which denotes whether it formed part of the best solution so far, and, $G_i$, which is the generation at which the Individual was a part of the solution.
3) The ***Population*** represents a group of individuals (duplicate individuals in the same population is permitted).
4) ***Select*** refers to tournament based selection.
5) ***Potential Parents*** are generated by splitting the current solution into multiple chunks. These chunks are then used by the crossover phase to generate new individuals.
6) ***Turnover*** is the percentage of individuals who are removed and then replaced after each generation.
7) The ***Fitness Value ($F_i$)*** the fitness function is a weighted sum in Eqn1 that determines the fitness value. The fitness value is used in each iteration to evaluate the quality of the proposed solution. The weights were set after preliminary tuning experiments and they enable us to enforce the importance of different terms.

$$F_i = w_1 F_{vi} + w_2 F_{ci} + w_3 F_{ti} \tag{1}$$

Where, $F_i$ = Fitness value for individual $i$. $F_{vi}$ = Normalised city count. $F_{ci}$ = Normalised cost for individual $i$. $F_{ti}$ = Normalised tour cost. $w_1$ = Weight for $F_{vi}$. $w_2$ = Weight for $F_{vi}$. $w_3$ = Weight for $F_{ti}$.

$$F_{vi} = \frac{c_t}{C} \tag{2}$$

Where, $c_t$ = Cities visited by the tour. $C$ = Total number of cities.

$$F_{ci} = \frac{i_{ci} - C_m}{C_x - C_m} \tag{3}$$

Where, $i_{ci}$ = Cost of individual $i$. $C_x$ = Maximum individual cost. $C_m$ = Minimum individual cost.

$$F_{ti} = \frac{t_c - T_m}{T_x - T_m} \tag{4}$$

Where, $t_c$ = Cost for individual $i$. $T_x$ = Maximum tour cost. $T_m$ = Minimum individual cost.

### A. Basic AEA

Algorithm 1 provides the basic Artificial Ecosystem Algorithm. A population of random individuals is created, where an individual is a fragment of an overall solution. From that population a new solution is assembled using tournament selection of individuals. *Potential Parents* is a pool of individuals created using individuals who have been selected to be part of a solution segment. The REMOVEANDREPLACEINDIVIDUALS proceduce uses tournament based selection to remove unfit individuals, then replaces them with individuals created using crossover of parents obtained from the pool of *Potential Parents*. As duplicate individuals are permitted, a procedure BALANCEFITNESS has been devised to ensure that duplicates have the same fitness.

**Algorithm 1** Basic AEA

```
Initialise Environment E
Initialise Population P
Set iteration counter to 1
loop
    Pick a random individual i1
    loop
        Select T_n compatible individuals
        Find individual i2 with highest F_i
        Add i2 to overall solution and update P_i and G_i
    until Overall solution is complete
    Update Potential Parents
    Update F_i for all individuals in the solution
    REMOVEANDREPLACEINDIVIDUALS
    BALANCEFITNESS
    Evaluate overall solution
    Increase Iteration counter
until Stopping criteria is met

procedure REMOVEANDREPLACEINDIVIDUALS
    R = Turnover * P
    loop
        Select T_m individuals based on their fitness
        Remove Individual with lowest F_i
    until R individuals have been removed
    loop
        Use crossover to create Individuals
    until R individuals have been added
end procedure

procedure BALANCEFITNESS
    F_u = 0
    loop
        Pick a random pair of individuals i1 and i2
        Fitness of i1 is F_i1; fitness of i2 is F_i2
        if F_i1 = 0 and F_i2 > 0 then
            F_i1 = F_i2
        else if F_i1 > 0 and F_i2 = 0 then
            F_i2 = F_i1
        end if
        F_u++
    until F_u Individual pairs have been picked
end procedure
```

*1) AEA applied to TSP:* In the Travelling Salesman Problem, a salesman is given a map of $n$ cities and he must travel to each city once and return to the start point with minimal travel distance. Given an undirected graph the outcome is a Hamiltonian cycle with minimal cost. For a symmetric TSP, the number of routes is $(n-1)!/2$. For a large number of cities the search space is tremendous and trying to brute force through all the different permutations of routes is computa-

tionally very expensive.

Algorithm 2 illustrates how the basic AEA can be applied to the TSP. Here the *Environment* holds all the cities. An *Individual* represents a subpath (the movement between two cities). Multiple individuals connected together form a *Tour* which is essentially the complete solution to the TSP, it is a path that goes through all the cities once and then returns to the start point. The *Fitness Value* $F_i$ is updated according to Eqn1.

**Algorithm 2** Basic AEA Applied to TSP

```
Initialise Environment E
Initialise Population P
Set iteration counter to 1
loop
    Pick a random individual i1
    Get the last City visited in i1 and
    Find all available Individuals
    loop
        Select T_n compatible individuals
        Find individual i2 with highest F_i
        Add i2 to overall solution and update P_i and G_i
    until Overall solution is complete
    Update F_i for all individuals in the solution
    REMOVEANDREPLACEINDIVIDUALS
    BALANCEFITNESS
    Evaluate overall solution
    Increase iteration counter
until Stopping criteria is met
```

### B. Artificial Ecosystem Algorithm with Species

The basic AEA demonstrates how a whole solution is built from smaller fragments in a single evolving population. However, an ecosystem is an interactive system that operates as a whole. It holds many populations, each of which represents a group of individuals of a particular species. The Artificial Ecosystem Algorithm with Species (AEAS) is based on this concept. In the AEAS, different species of individuals focus on different segments of the overall problem. The individuals in each species are chosen to form a valid segment of the solution as in the AEA, and they evolve in each species as in the AEA. However, overall solutions are now formed from the combination of solution segments in each species. To initialise the AEAS it is necessary to partition the overall problem into segments, where a species will address each segment. For a problem such as the TSP, this can be achieved using clustering algorithms to find groups of neighbouring cities. It is important to note that the Basic AEA is applied on a per species basis, each species is a separate unit and therefore solution segments for each species can be formed in parallel and on different processors.

*1) Artificial Ecosystem Algorithm with Species applied to TSP:* The AEA with species (AEAS) splits the problem into small segments of cities, an optimal tour is constructed for each segment, the subtours are then connected to build a complete tour, see algorithm 3. The final solutions formed from these multiple segments is valid but not necessarily optimal. This is because there exist internal constraints that stop the algorithm from building invalid permutations of solutions. It is possible to use many different algorithms to perform the problem decomposition. Here we describe the use of two alternatives for TSP: the Self Organising Map (SOM) and K-Means clustering. However, the AEAS may use any suitable methods for problem decomposition. It is not always evident which clustering algorithm will be superior for a given problem. This is outlined by the Impossibility Theorem [27], which states that no clustering algorithm can satisfy all data clustering axioms.

*a) AEAS (SOM):* Self Organising Map (SOM) is an unsupervised learning algorithm that allows the mapping from high dimensional space to low dimensional space, whilst preserving the topological structure. In this study we map to a 2-dimensional space. SOM is an artificial neural network that works in two phases; training and mapping. In the training phase, competitive learning is used by neurons to learn from a sample, thereby allowing a topological ordering of the map. In the mapping phase, input vectors are classified. There are three important variables that must be adjusted to suit the problem size: the height and width of the 2-dimensional map and the neighborhood radius that determines how much the neighborhood is influenced by [28].

*b) AEAS (K-means):* The number of clusters K is first chosen and the centroids are initialised. Choosing the value of K is one of the most difficult problems in clustering data [29], we used trial and error preliminary experiments to set the value for K. The cities are then assigned to clusters based on their spatial proximity to the respective centroids. Next the centroids are recalculated and continuously updated based on the spatial positions of all the cities in the cluster until there is no movement.

**Algorithm 3** AEA with Species Applied to TSP/ Dynamic TSP

```
Initialise Environment E
loop
    SOM/K-means to decompose TSP to clusters of cities
    for each Cluster C_n do
        Create a population P_n of individuals
    end for
    for each Species do
        Apply Basic AEA
    end for
    Connect all the segments to form a complete solution
    Evaluate overall solution
until Stopping criteria is met
```

*2) Artificial Ecosystem Algorithm with Species applied to the Dynamic TSP:* AEAS is applied to the DTSP with one difference: The AEAS applied to the TSP will terminate when there is no improvement in the current global optimum for

a set number of iteration. In contrast, the AEAS applied to the DTSP terminates when the algorithm has reached the maximum iteration count. This is is because the DTSP is a constantly changing problem with no fixed optimum solution.

## IV. Experiments

Three experiments are performed to investigate AEA and AEAS's ability to solve different instances of TSP and DTSP. All the Artificial Ecosystem Algorithms were developed in Java. The following TSP data was used:

1) Artificial Circle TSP data: Equidistant 2D points lying on a circle were constructed with an origin (450,350). Six circle datasets were created where the number of cities (c) and radius (r) were c:6 r:450, c:12 r:465, c:18 r:468, c:24 r:469, c:30 r470, c:54 r:470.
2) Real TSP data from TSPLIB [30] namely: Eil51, which is a 51 city problem and Eil75, which is a 75 city problem.

In order to ensure the reliability of these experiments we performed 30 independent runs. We then measured performance by calculating the deviation from the optimal tour cost using Eqn5. In addition to this we measure minimum, maximum and average values for tour cost as well as the number of evaluations used to find the optimal tour cost.

$$D = 100\frac{\bar{c} - c^*}{c^*} \tag{5}$$

Where, $D$ = Deviation from optimal. $\bar{c}$ = Mean tour cost. $c^*$ = Optimal tour cost.

The following settings hold for all experiments. The values of K clusters, map width, map height and neighborhood were set by running trial and error experiments and the best combination of values found were then used in the formal experimental process.

| | |
|---|---|
| Turnover | 20% |
| MaxGen | 100 000 |
| $T_n$ | 10% |
| $T_m$ | 10% |
| $w_1$ | 0.8 |
| $w_2$ | 0.1 |
| $w_3$ | 0.1 |

TABLE I: Experimental Setup

### A. Experiment 1

This experiment establishes the effects of population size on the different Artificial Ecosystem Algorithms and also demonstrates the effectiveness of the AEAS compared to the basic AEA.

*1) Setup:* The same experiment was performed for all three versions of the AEA. Tables II, IV and III provide the parameter values varied in this experiment.

| Cities | 6 | 12 | 18 | 24 | 30 | 54 |
|---|---|---|---|---|---|---|
| Population | 500 | 1000 | 2000 | 4000 | 6000 | 8000 |

TABLE II: Baseline AEA Experimental Setup

| Cities | 6 | 12 | 18 | 24 | 30 | 54 |
|---|---|---|---|---|---|---|
| Population | 300 | 500 | 700 | 1000 | 2000 | |
| Map width | 2 | 3 | 4 | 6 | 8 | 10 |
| Map hieght | 2 | 3 | 4 | 6 | 8 | 10 |
| Neighborhood | 2 | 4 | 6 | | | |

TABLE III: SOM AEA Experimental Setup

| Cities | 6 | 12 | 18 | 24 | 30 | 54 |
|---|---|---|---|---|---|---|
| Population: | 300 | 500 | 700 | 1000 | 2000 | |
| K Cluster: | 2 | 4 | 6 | 10 | | |

TABLE IV: K-means AEA Experimental Setup

*2) Results:* Fig. 1a, 1b and 1c represent how much the solution deviated from the optimal using different number of populations for a given number of cities, where zero (no deviation) represents the optimum. Please note the scale on Fig. 1c is different as the deviation from optimal was very small.

The Basic AEA achieved optimal solutions for smaller TSP problem sizes (cities= 6 and 12) and for lower population sizes, but it was unable to cope with larger problems. Varying the population size for the Basic AEA provided unexpected results - in general smaller population sizes provided solutions closer to the optimal. This may be because of the way tournament selection is used to assemble complete solutions - larger population sizes may make it harder to find good quality individuals to assemble into good overall solutions.

Both versions of AEAS provided considerably improved results. AEAS (K-means) achieved excellent solutions for all TSP problems, and again showed that smaller population sizes seem to be more effective. AEAS (SOM) achieved very good results, although not quite as good as AEAS (K-means). Interestingly AEAS (SOM) appeared to show best results for specific and different population sizes for each TSP size; the exact cause of this behaviour remains to be investigated.

### B. Experiment 2

This experiment compares the different algorithms in terms of the number of evaluations used to find a solution and the deviation from the best solution so far.

*1) Setup:* In this experiment we fix the population size to 500 and compare all three AEA algorithms by varying the problem size. Table V describes the parameters varied in this experiment.

| Cities | 6 | 12 | 18 | 24 | 30 | 54 |
|---|---|---|---|---|---|---|
| K Cluster | 4 | 6 | 8 | 10 | | |
| Map width | 2 | 3 | 4 | 6 | 8 | 10 |
| Map hieght | 2 | 3 | 4 | 6 | 8 | 10 |
| Neighborhood | 2 | 4 | 6 | | | |

TABLE V: Baseline, SOM and K-means AEA Experimental Setup

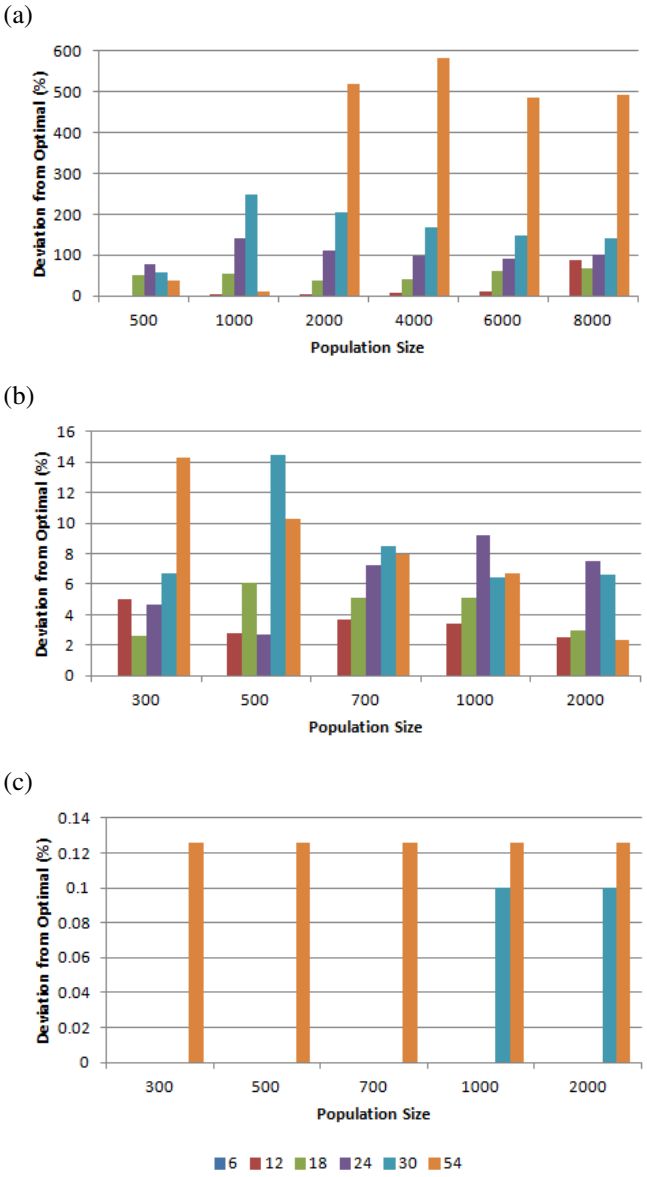


Fig. 1: Deviation from optimal for different TSP sizes against population sizes (a) Basic AEA. (b) AEAS (SOM). (c) AEAS (K-means). It should be noted that the scale for the Deviation from Optimal axis for the three figures is radically different, clearly indicating the poor relative performance of the basic AEA compared to AEAS.

Trial and error experiments were used to set the values for K clusters and SOM height, width and neighborhood. Different combinations of the above values were used for different city sizes.

*2) Results:* Fig. 2a and 2b summarise performance of the basic AEA, AEAS(SOM) and AEAS(K-means) algorithms in terms of the evaluations and the deviation from optimal. The figures show that the baseline AEA does not scale well with an increased TSP problem size, the number of evaluations required to form a solution drastically increased and solution obtained massively deviated from the optimal tour cost.

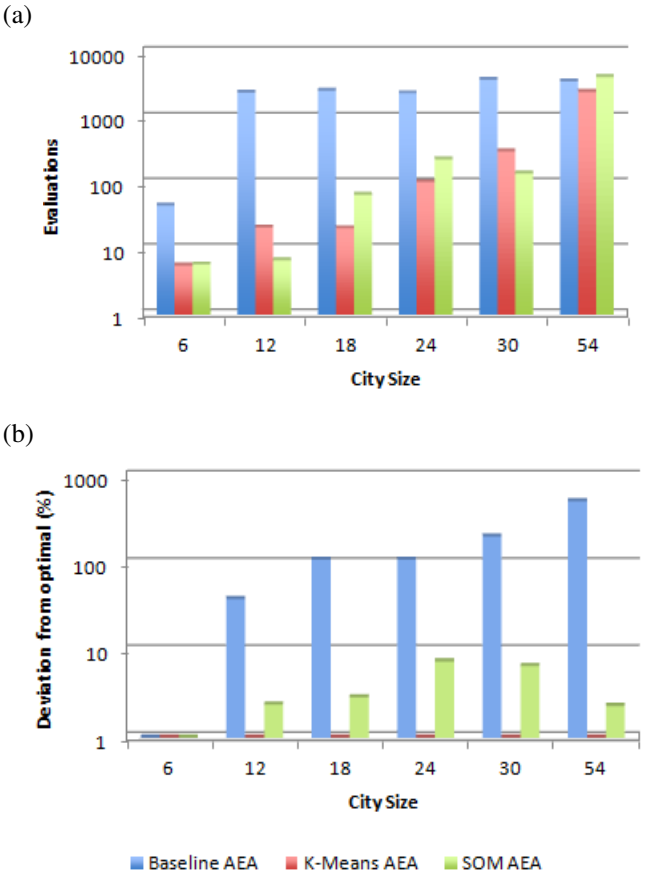


Fig. 2: (a) Comparison of number evaluations needed by AEA methods for different TSP sizes - using a logarithmic scale. (b) Comparison of deviation from optimal of AEA methods for different TSP sizes.

The results show that AEAS scales very well with increased problem complexity.The AEAS (K-Means) generally outperforms the SOM in terms of deviation from optimum, however the AEAS (SOM) generally used fewer evaluations. The difference is interesting and may point to the use of AEA (K-Means) as the approach with the most potential for the future.

### C. Comparison with other approaches

It is important to examine the capabilities of this new method in the context of results reported in the literature for other bio-inspired algorithms applied to TSP. Table 6 shows the results of AEAS (SOM) and AEAS (K-means) for different TSPs compared to GA, BCO and IWD.

For the larger TSP problems tested here, AEAS performs extremely well - as good as or better than other bio-inspired methods reported in the literature. For 101 cities AEAS also finds near optimal solutions, clearly demonstrating the potential of this approach.

### D. Experiment 3

Ecosystems have the ability to adapt to changing conditions over time [24]. We therefore perform a final experiment to

TABLE VI: Comparative Analysis, results were taken from [31] [32] and represent the best results over different runs

| File | Cities | Opt | AEAS(SOM) | AEAS(K-means) | GA | ACO | IWD |
|---|---|---|---|---|---|---|---|
| Eil51 | 51 | 426 | 452 | 445.5 | 445.8 | 427 | 471 |
| Eil76 | 76 | 538 | 569 | **555** | | 676 | 559 |
| Eil101 | 101 | 629 | **631.6** | 670.3 | | | |

| Cities | 24 | 30 | 54 |
|---|---|---|---|
| K Cluster | 5 | | |
| Population | 200 | | |
| DP Interval | 10 | 50 | 100 |

TABLE VII: Dynamic TSP Experimental Setup

analyse AEAS's ability to find solutions to dynamic problems. To achieve this we apply the AEAS (K-means) to the DTSP.

*1) Setup:* Table VII describes the parameters used in this experiment. The DTSP gains or loses a city randomly every DP interval, this determines how frequently the problem changes. The termination criteria is a fixed maximum number of iterations. We focus on looking at AEAS's behaviour when applied to the DTSP with 30 cities, as preliminary experiments indicate the algorithm performs in a similar manner for DTSP with fewer or more cities and correspondingly fewer or more clusters.

*2) Results:* Fig. 3a, 3b and 3c show that for a 30 city problem, the algorithm takes on average around 60 evaluations to find an optimal solution. However, when the number of cities changes, subsequent learning is extremely rapid - an average about 4 iterations. The lower the DTSP interval is, the less chance the algorithm has to recover and therefore the performance is degraded.

We define the recovery time to be the amount of time (number of evaluations) it takes for the algorithm to return to an optimal state. Fig. 4 shows the number of changes to the problem. The x axis shows the number of changes to the problem so far in the algorithm, the y axis shows the average number of evaluations it takes to find a solution. This graph shows very clearly that once the algorithm learns the problem it takes very little time to recover after subsequent changes of the problem space. These are very promising results as they indicate that the algorithm has a natural ability to retain and reuse previous good solutions and fragments of good solutions, unlike other population-based algorithms which converge on current solutions and cannot retain earlier solutions.

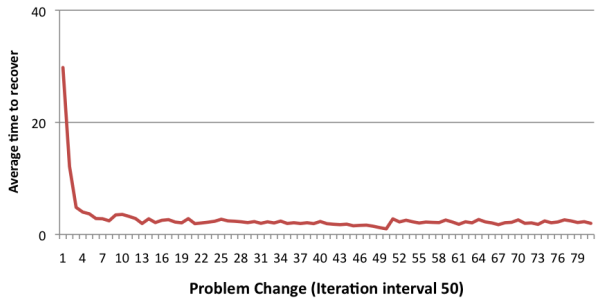


Fig. 4: Average time to recover against DTSP problem changes.

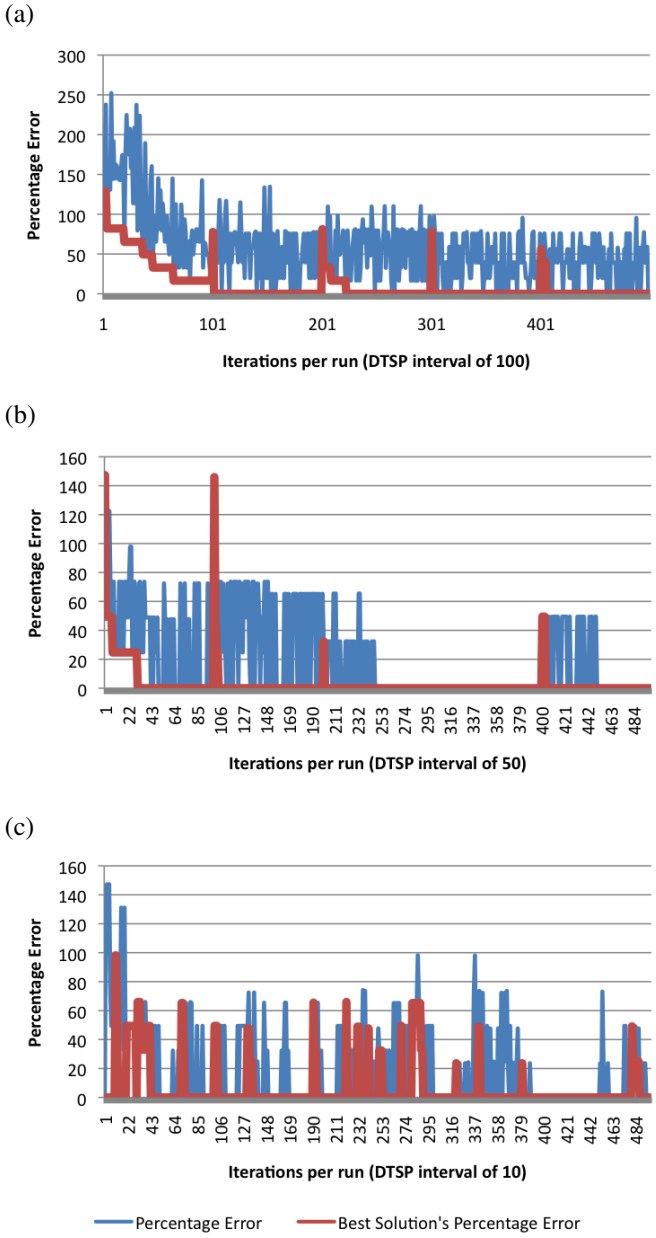


Fig. 3: (a) Solutions percentage error for a DTSP interval of 100. (b) Solutions percentage error for a DTSP interval of 50. (c) Solutions percentage error for a DTSP interval of 10.

## V. Conclusion

Evolution and the behaviour of interacting members of populations have long been an inspiration for designers of bio-inspired algorithms. However, the combined behaviour of many species as they evolve, adapt and combine to form a

single ecosystem, had not been used as inspiration previously. This work presented the first ecosystem-inspired algorithm designed to take advantage of highly distributed computer architectures and tackle large-scale problems. Just as an ecosystem comprises of many separate components that adapt together to form a single synergistic whole, the Artificial Ecosystem Algorithm (AEA) solves a problem by adapting subcomponents of a problem such that they fit together and form a single optimal solution. This work also introduced the AEAS - an AEA with species of components representing subparts of the solution that evolve together and cooperate with the other species. AEAS data decomposition is not limited to SOM and K-means, different levels of clustering can be used to expose different dimensions of the data. For example, density Based Clustering can be used to avoid the formation of highly packed clusters. It is also possible to look at the topology of the search space and according to its properties select an appropriate decomposition method. The AEA and two versions of the AEAS were evaluated by application to the symmetric Travelling Salesman Problems of varying size. The experiments showed that smaller population sizes were more effective, and that the use of species in AEAS to solve segments of the solution enabled the algorithm to find better solutions compared to AEA. Indeed comparisons of AEAS with the performance of other more established bio-inspired methods provided very encouraging results. Finally, experiments where AEAS (K-Means) was applied to the Dynamic TSP showed promising results as this method was able to adapt to changing problems very effectively and also retain good solutions and fragments of solutions to dynamically changing problems. This is significant as it makes this algorithm a potential candidate for real world problems such as routing in ad hoc or dynamically changing networks.

## REFERENCES


[1] W. Gurney and R. M. Nisbet, *Ecological dynamics*. Oxford University Press, Oxford, 1998.

[2] G. Reinelt, *The traveling salesman: computational solutions for TSP applications*. Springer-Verlag, 1994.

[3] C. Li, M. Yang, and L. Kang, "A new approach to solving dynamic traveling salesman problems," in *Simulated Evolution and Learning*. Springer, 2006, pp. 236–243.

[4] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[5] L. Davis, "Applying adaptive algorithms to epistatic domains." in *IJCAI*, vol. 85, 1985, pp. 162–164.

[6] D. E. Goldberg and R. Lingle Jr, "Alleleslociand the traveling salesman problem," in *Proceedings of the 1st international conference on genetic algorithms*. L. Erlbaum Associates Inc., 1985, pp. 154–159.

[7] W. Banzhaf, *Biological Cybernetics*, vol. 64, no. 1, pp. 7–14, 1990.

[8] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 36–39.

[9] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 851–871, 2000.

[10] 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.

[11] T. Stutzle and H. Hoos, "Max-min ant system and local search for the traveling salesman problem," in *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE, 1997, pp. 309–314.

[12] P. Shelokar, P. Siarry, V. K. Jayaraman, and B. D. Kulkarni, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization," *Applied mathematics and computation*, vol. 188, no. 1, pp. 129–142, 2007.

[13] H.-M. Feng and K.-L. Liao, "Hybrid evolutionary fuzzy learning scheme in the applications of traveling salesman problems," *Information Sciences*, vol. 270, pp. 204–225, 2014.

[14] T. Sato and M. Hagiwara, "Bee system: finding solution by a concentrated search," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 4. IEEE, 1997, pp. 3954–3959.

[15] L.-P. Wong, M. Y. H. Low, and C. S. Chong, "Bee colony optimization with local search for traveling salesman problem," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 03, pp. 305–334, 2010.

[16] G. K. Jati *et al.*, "Evolutionary discrete firefly algorithm for travelling salesman problem," in *Adaptive and Intelligent Systems*. Springer, 2011, pp. 393–403.

[17] B. Good and F. Sahin, "Bacterial foraging approach to classic traveling salesman problem," in *2006 International Conference on Computational Science and Education*, 2006.

[18] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. IEEE, 2009, pp. 210–214.

[19] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Improved and discrete cuckoo search for solving the travelling salesman problem," in *Cuckoo Search and Firefly Algorithm*. Springer, 2014, pp. 63–84.

[20] J. McCormack, "Artificial ecosystems for creative discovery," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 301–307.

[21] A. J. Watson and J. E. Lovelock, "Biological homeostasis of the global environment: the parable of daisyworld," *Tellus B*, vol. 35, no. 4, pp. 284–289, 1983.

[22] T. S. Ray, "Evolution, ecology and optimization of digital organisms," *Santa Fe*, 1992.

[23] C. Adami and C. T. Brown, "Evolutionary learning in the 2d artificial life system avida," in *Artificial life IV*, vol. 1194. Cambridge, MA: MIT Press, 1994, pp. 377–381.

[24] J. R. Dyer and P. J. Bentley, "Plantworld: Population dynamics in contrasting environments." in *GECCO Late Breaking Papers*, 2002, pp. 122–129.

[25] M. Kirley, "A cellular genetic algorithm with disturbances: Optimisation using dynamic spatial interactions," *Journal of Heuristics*, vol. 8, no. 3, pp. 321–342, 2002.

[26] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, no. 4, pp. 355–366, 2006.

[27] J. Kleinberg, "An impossibility theorem for clustering," *Advances in neural information processing systems*, pp. 463–470, 2003.

[28] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[29] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[30] G. Reinelt, *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.

[31] G. Vahdati, M. Yaghoubi, M. Poostchi, and S. Naghibi, "A new approach to solve traveling salesman problem using genetic algorithm based on heuristic crossover and mutation operator," in *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of*. IEEE, 2009, pp. 112–116.

[32] D. Gupta, "Solving tsp using various meta-heuristic algorithms," *International Journal of Recent Contributions from Engineering, Science & IT (iJES)*, vol. 1, no. 2, pp. pp–26, 2013.