Detection of Pheromone Laying Event in Foraging Data of Harvester Ants Using Change Point Analysis Method

by

Safeeul Bashir Safee

B.Sc., Chittagong University of Engineering & Technology, 2011

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Computer Science

The University of New Mexico

Albuquerque, New Mexico

May, 2017

Dedication

To my parents and my elder brother for their support, encouragement and the Corvette they're giving me for graduation.

Acknowledgments

I would like to thank my thesis supervisor Professor Dr. Melanie E. Moses, for her enormous support for selecting the topic to study. Without her guidance, it would have been a tough journey for me in this field. I would also like to thank Dr. Tatiana P. Flanagan for her initial guidance to me for this topic. My heart full of gratitude to Professor Dr. Abdullah Mueen for his guidance towards my degree. I am thankful to all the members of Moses Biological Computation Lab for their suggestions towards my research.

Detection of Pheromone Laying Event in Foraging Data of Harvester Ants Using Change Point Analysis Method

by

Safeeul Bashir Safee

B.Sc., Chittagong University of Engineering & Technology, 2011

M.S., Computer Science, University of New Mexico, 2017

Abstract

Communication is an important factor in the foraging performance of social insects, like ants. During foraging, ants keep track of the food sources by using memory (site fidelity) or communicating it through pheromones. Previous field experiments showed that the rate of seed collection depends on the distribution of food in the environment. If food is spatially clustered, then ants recruit nest mates to collect from large clusters. However, we don't know when the recruitment occur. To explore this question, we analyzed foraging rates on different sizes of piles in the simulation. Using a power law distribution to arrange seeds in piles of different sizes, we observed that for significantly large piles of seeds, the ants take more time to discover a pile, but once discovered, seeds are collected at an increased rate from that pile. We also observed that ants may repeatedly lose track of found piles and then re-find them. Using change point analysis on seed intake time series, we were able to trace the discovery of piles by detecting changes in the foraging rate. We use simulations to determine how to correlate change points with recruitment events, and then use that relationship to infer when recruitment occurs in field data.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Social insects are basically species that lives in colonies and manifest three characteristics [1]. a. Group integration[2], b. Division of labor[3] and c. Overlapping of generations[21]. These creatures have survived many mass level extinction events that happened in the earth. Millions of years of genetic evolution to survive in the hostile environment helped them to adapt to the environment and master the strategies of survival.

Their strategies to avoid congestion and optimize their movements to move or forage in most efficient ways without any central authorities has attracted so many researchers and scientists over the centuries[15]. As a result, we got a new branch of science which is called myrmecology. In modern computer science, machine learning[6], complex interactive networks[10], parallel computing[4] and many other topics have been inspired by the studying and modeling of ants.

Harvester ants are social insects. They forage from the environment. Most of their foraging activities are during the morning or in the evening sessions. And their foraging activity is at the top during the summer[12, 20]. In this study, we will mostly talk about *Progonomyrmex* species which are group foragers[19]. Foraging

activities of harvester ants including *Progonomyrmex sp.* depends on many factors like environment temperature, light, and availability of seeds[20].

Social insects like ants use pheromone to communicate with each other to perform their daily activities which also includes foraging[13]. The previous study has shown that they follow three strategies to forage from the environment[7]. Ants use memory to remember the location of the food source. They communicate with other ants using pheromone and they perform the random walk in the spatial dimension in search of food.

Foraging of ants from a particular food source depends mostly on the how food is distributed in the environment[18]. To analyze the foraging strategies of ants field experiments has been conducted on three species of Pogonomyrmex desert harvester ants. Foods were distributed among the nest in different distributions to observe the effect of food density on foraging. It was demonstrated that ants take some time to discover the large pile of seeds, but they start recruiting from the food source once they discover it[7].

Based on this behavior an agent-based model CPFA[11] is developed by Moses Biological Computation Lab. CPFA is implemented on various platforms. The purpose agents of CPFA is to collect resources from the spatial environment by the strategies mentioned above.

As mentioned previously, they use three different strategies to forage[5, 7], it was not clear what strategies they use to recruit from a clumped food source[17]. Our initial observation showed that when ants discover a pile they lay pheromone trail for other ants to follow. When other ants start following the pheromone trail, their foraging rate goes up for that pile. We have used change point detection algorithm to detect that change in foraging rate. Our goal was to determine what strategies ants mostly use to forage from clumped food source. It was difficult to determine from

the field data whether the detected change points indicates the pheromone laying event. To do so we have used simulations.

We have used CPFA to simulate the field experiments. The environment of CPFA has been tuned to mimic the field experiments for three different species *P. Rugosus*, *P. Maricopa* and *P. Desertorum*. We have implemented different change point detection algorithms on the simulated data[8, 16, 14]. The change point detection algorithms were tuned to detect change points when the pheromone is laid. For each of the species the change point detection algorithm has different sets of parameters. As from the simulation, we know exactly when the pheromone was laid and site fidelity was used, we verified our change point detection algorithms by using the simulation data. Based on the results of the simulation we have selected best change point detection method. And applied the result to the field data.

We observe ants use pheromone more when the food sources are clumped. And they discover the pile more frequently if the food source is large. They don't use the pheromone to recruit from the food source that is scattered in the environment.

As we mentioned Foraging of ants depends on many factors including temperature, availability of food, distance from food to the nest and so many[20, 9]. So in some of the field experiments, ants did not collect enough seeds. Which is why we were unable to detect any change points in some of the experiments for each of the species. Since the CPFA does not depend on these conditions, we have detected change points in each of the simulations.

Chapter 2

Background Study

Ants are social insects. These small tiny creatures have survived major extinction level events in the world. They have been evolved millions of years to survive in the worlds. Their strategies to find resources for their survival are fascinating. We ran several experiments on desert harvester ants at the field to observe how they forage. We have selected three different species of harvester ants to observe their foraging strategies. Those ants are *P. Rugosus*, *P. Desertorum* and *P. Maricopa*.

Our goal was to figure out how they find resources from an environment and what is the effect of the distribution of information on their foraging. So we ran experiments on ants. Seeds in the fields are distributed in a ring. The area of the food distribution is scaled with the colony size of ants. For example, *Desertorum* was the smallest in colony size (77$\pm$296), so the ring radius of food was 1.5 to 3 meters, Rugosus has a colony size of 1712$\pm$174. So the radius of food distribution for Rugosus was in 5-10 meters. The seeds are organized in a power law distribution around the nest.

2.1 Power Law

In power law distribution of food, seeds are distributed into multiple piles of different pile size. For example, for power rank 5 of power law distribution total number of seeds will be 1024. And it is divided into 4 types of 256 seeds in each type. One large pile of 256 seeds are placed all together in a certain position. Next 256 seeds are divided into 4 equal sizes of 64 seeds and placed around the nest. Next 256 seeds are equally divided into 16 piles of 16 seeds and placed around the nest inside the ring. Rest of the seeds are distributed uniformly around the nest.

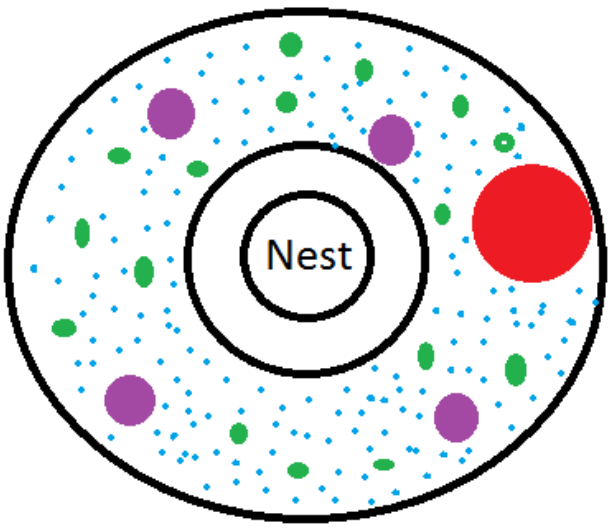


Figure 2.1: Distribution of Seeds in the field experiment for power law distribution with power rank 5. Red Pile indicates one large pile of 256 seeds. 4 purple piles represent 4 large piles of 64 seeds, Green color represents 16 piles of 16 seeds and blue seeds are 256 random seeds

2.2 Field Observation

After the distribution of foods across the nest, we observe their collection of seeds. It is observed that, ants need longer time to discover the large pile with 256 seeds than the piles with small amount of seeds. But once the seeds are discovered, they started recruiting from the piles.

Studies showed that information is transferred among the ants for larger piles with more seeds. We have plotted the collection of seeds from the field experiments and discovered ants walk randomly to collect seeds. Once they get the seed they bring it back to the nest. But if they discover a large food source they share the information with others to recruit from the food source. Once the information is shared more ants come into recruitments and while collecting this food they also share information with other ants.

As soon as ants start recruiting from the food source, their foraging rate goes up. But the food source starts losing the number of seeds. As the number of seeds starts decreasing from the food source the recruitment of amount of ants also starts decreasing.

2.3 CPFA

Based on this observation, an model was proposed to simulate the behavior of ants. It is called Central Place Foraging Algorithm(CPFA). CPFA is an agent-based model where agents are programmed to follow ants strategy to collect seeds. In CPFA Ants follows three methods to collect seeds.

- **Random Walk:** Ants starts walking randomly from the nest. If they find any food they bring it back to nest

- **Use of Internal Memory:** They remember the last position where the food was found and can return to that place for further search of food. This is method is called site fidelity.

- **Use of Pheromone or Communication:** They lay pheromone trails from the food source to the nest so that other ants can follow that trail to collect food from that source.

Initially, a search location is selected for each of the ants. Then ants start traveling to the search site. After reaching the search site, they perform either uninformed random walk to a random location or informed random walk to a known location based on site fidelity or pheromone. If no resource is found, then they return to the nest.

If they find any resource they start to sense the local resource density. Based on the local resource density they decide whether to use site fidelity in future or to lay pheromone. After sensing the resource density, the return to the nest with the seed.

While foraging whether the agents will use any of three strategies is governed by seven parameters. Table 2.1 represents the seven parameters, that governs CPFA and their initialization values.

Parameters	**Initialization Functions**
Probability of Switching to Searching	U(0,1)
Probability of Returning to Nest	U(0,1)
Uniform Search Variation	(0, 4 PI)
Rate of Informed Searched Decay	E(20,0)
Rate of Site Fidelity	E(20,0)
Rate of Laying Pheromone	E(20,0)
Rate of Pheromone Decay	E(20,0)

Table 2.1: Seven parameters and their initialization, that characterizes Central Place Foraging Algorithm

Parameters that are following a uniform distribution, higher the value of the parameter higher the probability of that event. For example, "probability of switching to searching" follows a uniform distribution. Higher the value of this parameter, higher the probability that the ant will switch to search.

For the exponential distribution, higher the value of the parameter, lower the chance of using that feature. For example, if *Rate of Laying Pheromone* is zero, then it means that there is a higher chance of laying the pheromone. On the other hand, a value close to 20 means that chance of using the pheromone is very low.

The *probability of switching to searching* determines the chance of ants to switch to search for resources. When ants were not primed by site fidelity or pheromone information, they select a random location to visit. And search for resources in the pre-determined location. If they cant find any food, they select another new location. The higher the value of switching to the searching parameter, the more they search.

The *probability of returning to nest* defines the chances of returning to nest for unsuccessful foraging trip. Ants determine the position of a search location by either using site fidelity, or pheromone or randomly. When they travel to a particular location, they look for resources. If they cannot find any resource, they select a new location to explore and travels to that place. They keep selecting a new place to explore for a certain period of time. The parameter *probability of returning to nest* comes into play in this scenario. This parameter actually decides how long the ant will keep searching new places before returning to the nest. Higher the value of the parameter, higher the ant will explore the places for resources.

The *rate of site fidelity* comes into play when the ants use site fidelity to go to an informed location where the food already exists. Lower the value of this parameter, higher the chances of following the site fidelity.

The *rate of laying pheromone* is the probability of laying pheromone while returning to nest from a food location. This depends on sensing of local resource density. When the ants collect seeds from a location, they sense the density of resources in nearby areas. Lower the value of *probability of laying pheromone*, higher the chances of laying pheromone.

The *probability of pheromone decay* determines how fast the pheromone will evaporate. When the value of the parameter is low pheromone stays in the ground for longer time. And act as vice versa.

2.4 Genetic Algorithm

Genetic Algorithms are metaheuristic search algorithm inspired from natural selection and evolutionary genetics. As such they represent intelligent exploitation of a random search used to solve optimization problems. The basic techniques of GAs are designed to simulate processes in natural systems necessary for evolution.

The evolution usually starts from a population of randomly generated individuals strings that are analogous to the chromosome that we see in our DNA, and is an iterative process, with the population in each iteration called a generation. GAs simulate the survival of the fittest among individuals over the consecutive generation for solving a problem. Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution. GA proceeds through the solution domain by evaluating the fitness function. The whole process of evolution is divided into three major section-Selection, Crossover & Mutation.

2.4.1 Selection

Selection is an important part of Genetic Algorithm. Without selection, directing the algorithm towards the fitter solution is not possible. During each successive generation, a selection of population is allowed to breed to maintain the population. Individual populations are selected throughout the evaluation of fitness function. Typically, the individual populations with better values of fitness functions are selected. Selection methods differ by the goal of the GA. Some selection methods focus on the fitness of the solution and preferentially selects the best solution. As the evaluation of each function is time-consuming, some methods, only rates the solutions, does not evaluate them. Few strategies of selections are truncation selection, roulette selection, stochastic universal sampling, tournament selection, elitism and rank selection.

2.4.2 Crossover

In genetic algorithm, crossover is a genetic operator which is used to vary the programming of chromosomes to pass it to the next generation. Crossover is a process of taking more than one parent solution and producing child solution from them. There are various methods of crossovers. There are various methods of crossovers. Among them, single point crossover and two point crossovers are mostly used methods.

In *single point crossover*, a point is selected in both of the parents and genes are exchanged at that particular point between the parents.

In *two point crossover* method two different points are selected as head and tail and all the data are swapped in between this head and tail point among the parents.

2.4.3 Mutation

Mutation is a genetic operator used to maintain the diversity between the generations of a population. It is more similar to the biological mutations. Mutation is more like bit flip operation where in terms of the genome we alter the value of one or more gene. Using mutation a solution can be achieved which is entirely different from the previous solution. Thus we can achieve better solutions by doing mutations in the chromosomes. We can control the intensity of mutation by setting the probability of mutation for the chromosomes.

With some low probability, a portion of the new individuals will have some of their bits flipped. Its purpose is to maintain diversity within the population and inhibit premature convergence. Mutation alone induces a random walk through the search space. Mutation and selection create a parallel, noise-tolerant, hill-climbing algorithms.

The process is followed until a common termination condition is reached. This termination condition can be either a solution which fulfills minimum criteria. GA can also be terminated once it reaches a desired number of generation, or if it reaches the maximum fitness. Evaluation of fitness functions also takes lots of time. So sometimes the GA is bound to a strict time limit, where The genomes or chromosomes are collected after completion of GA run for a certain time. The GA can also be terminated by any combination of the termination methods mentioned above. In Short, the GA Algorithm can be defined as *algorithm 2.1*

Algorithm 2.1 Genetic Algorithm at a glance.

1: Initialize the population randomly
2: Determine the fitness of the first generation
3: **while** Desired solution is obtaind **do**
4: Select elite population with the best fitness
5: Create a new population by crossover and mutation among the elite population
6: Evaluate fitness of the population
7: **end while**

Although GA is very effective in finding the solution in a large special domain, it has some drawbacks. If the length of the chromosome increases, the solution space may increase exponentially, which may increase the time to reach the solution domain. Also, GA mutation and crossover in an undesired region can produce useless genomes which can push the solution to some local maxima. Besides, evaluating repeated genome in the same generation can increase the evaluation time. GA's also can't solve efficiently where the fitness function measure is a boolean value. Evolving the GA fitness function in first come-first serve manner can increase the bottleneck of the problem.

Chapter 3

Method

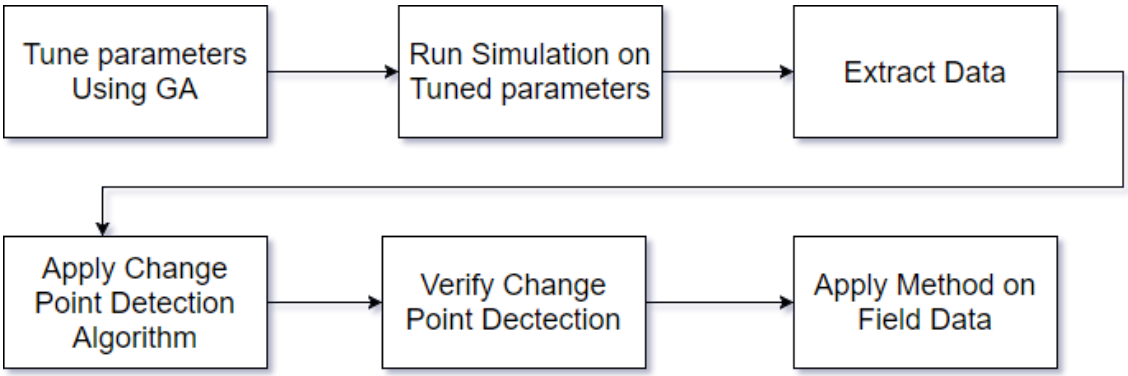


Figure 3.1: Steps of Change Point Analysis

3.1 Setting Simulation Environment

We tried to setup the simulation environment as like as the field experiment environment. So we distributed the resources in Donut Shapes. We had three different setups for three different species of ants. For *Rugosus* the distribution of the resources was in power law. Total 1024 seeds are distributed in 4 different densities.

256 seeds are piled up in one single pile. There were 4 piles of 64 seeds in each. 16 piles had 16 seeds each and rest of the 256 seeds were distributed uniformly among the nest in the donut shape ring. In the field experiment, the area of the donut shape was proportional to the colony size. In here we tried to keep the area of the donut shape ring similar to the field experiment. The inner radius of the ring was 5 meter and the outside radius was 10 meter for *P. Rugosus*. We kept the number of agents up to 12. The total duration of each experiment was 90 minutes (We collected data from field experiments for 90 minutes only). The total arena size was 20×20 meter. We kept the arena into this size and bounded the agents to search in this arena. The setup is varied for *Maricopa* and *Desertorum*. Table 3.1 represents the environmental setup of simulations for *P. Rugosus*, *P. Maricopa* and *P. Desertorum*.

Species	**Number of Seeds**	**Radius of Seed Distribution**
P. Rugosus	1024	5-10 meter
P. Maricopa	128	1-3 meter
P. Desertorum	128	1-3 meter

Table 3.1: Environmental Setup of simulation for three species

3.2 Tuning Parameters using Genetic Algorithm

To analyze the data, we have tuned the parameters of CPFA. As stated above in the background study, enormous amount of parameters for CPFA can be used to evaluate the fitness. But we have used the genetic algorithm to achieve the optimum set of parameters. We have divided the simulations into three categories to tune the GA for three different environments.

1. **Pheromone Only Parameters:** For this type we have restricted the use of

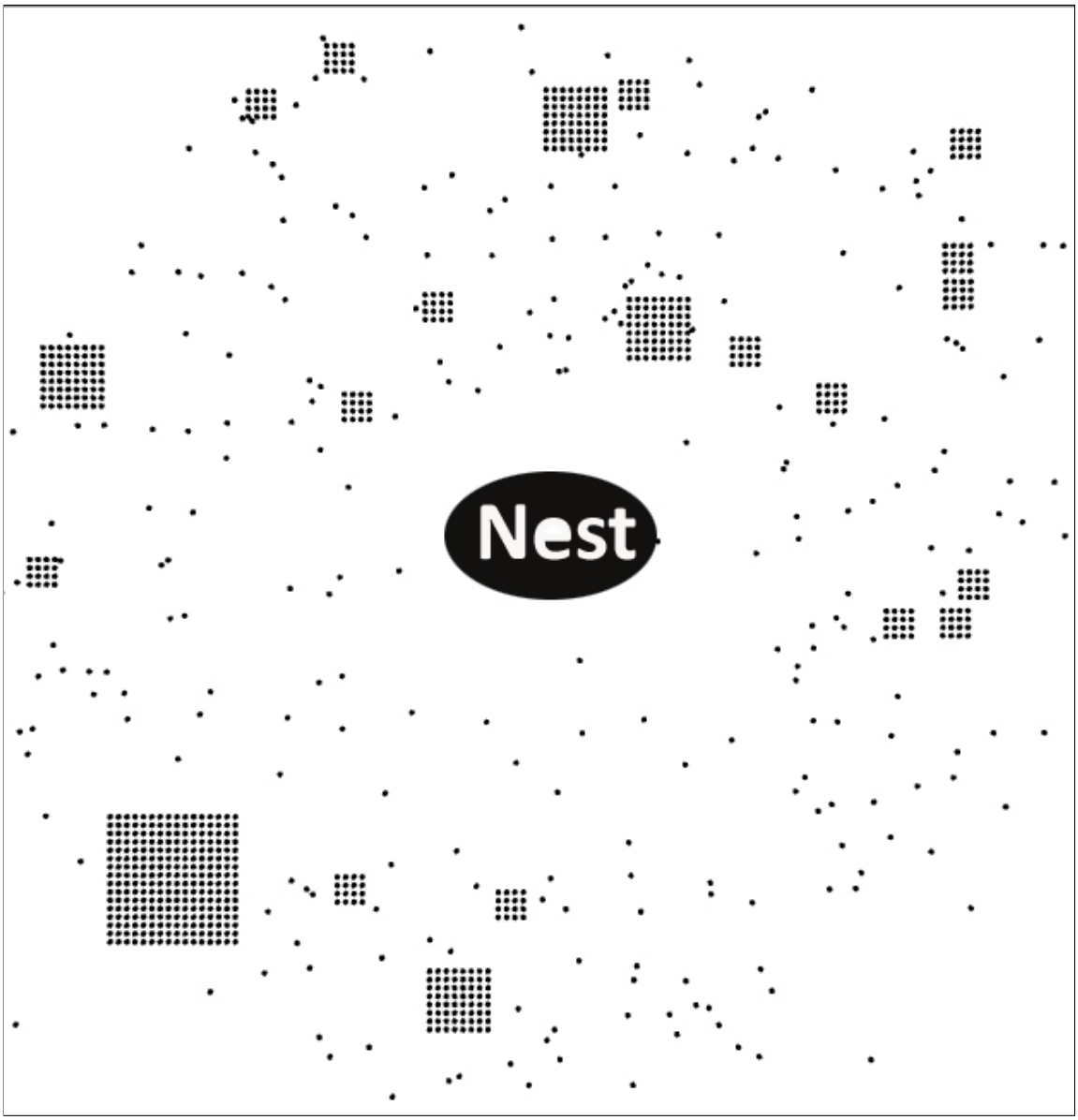


Figure 3.2: Example setup of a simulation environment for *P. Rugosus* with total 1024 seeds. The setup showing three different types of piles. One large pile of 256 seeds, Four piles of 64 seeds and sixteen piles of 16 seeds. 256 random seeds are distributed uniformly inside the ring.

site fidelity. Which means that the probability of using site fidelity is 20. Rest of the parameters evolved using the GA.

2. **Site fidelity Only Parameters:** In this types of experiment we restricted the use of pheromone. For this case ants can only use site fidelity and random walk to collect resources

3. **Using Both site fidelity and pheromone:** This environment represents the actual field experiment condition. Here agents use both site fidelity and pheromone along with the random walk.

For each type of environment, we have tuned the parameters to obtain maximum fitness using genetic algorithm. Initially, we have created a population of one hundred robotic swarms in simulated environment. Each swarm's foraging strategy is randomly initialized using the parameter setting mentioned in table 2.1. The best genome is selected for crossover and mutation for next generation. We continued this process until we obtain the best fitness genome or parameter set. The GA is terminated either when the parameters are converged, or it reaches to generation 50.

For each swarm in a population, fitness is tested for four different random seeds. The random seed is the variable which controls the variables of a simulation. After evaluation of each random seed for one parameter set, we have calculated the average seed collection to define the fitness of that particular swarm. These random seeds are basically numbers which are fixed for each generation. For each generation, we have selected four different numbers for the random seeds and then evaluated all the parameters for those values.

For GA we can confine particular parameters to a certain range of values by setting this in evolver.cpp file. For Example, for "Pheromone Only" environment we have restricted the values of site fidelity to $(20, 20)$ where each value inside the parenthesis represents upper bound and lower bound.

To calculate the fitness for each parameter set it takes evaluating the fitness function for four times due to four different random seeds, which means for each

generation it needs evaluating the objective function for 400 times. So over 50 generations, it will need 2000 evaluations of the objective function. This can take a lot of time if we perform the evaluation sequentially.

To remove this bottleneck, we have used multi-threading of genetic algorithm by evaluating multiple objective functions simultaneously. We have used GA Lib Genetic Algorithm Package. The software for this work used the GAlib genetic algorithm package, written by Matthew Wall at the Massachusetts Institute of Technology. The MPI version was written by Andrew Rasmussen https://github.com/andyras/GAlib-mpi/blob/master/LICENSE who modified the code from https://github.com/BORJA/GAlib-mpi. The evolver.cpp file is used to initialize the GA parameters and pipe the parameters to the GA Lib. Detail of initial parameter settings of genetic algorithm for three different setup is given in table 3.2.

CPFA Parameters	**Pheromone Only**	**Sitefidelity Only**	**All Parameters**
Probability of Switching to Searching	U(0,1)	U(0,1)	U(0,1)
Probability of Returning to Nest	U(0,1)	U(0,1)	U(0,1)
Uniform Search Variation	(0, 4 PI)	(0, 4 PI)	(0, 4 PI)
Rate of Informed Searched Decay	E(20,0)	E(20,0)	E(20,0)
Rate of Site Fidelity	E(20,20)	E(20,0)	E(20,0)
Rate of Laying Pheromone	E(20,0)	E(20,20)	E(20,0)
Rate of Pheromone Decay	E(20,0)	E(20,20)	E(20,0)

Table 3.2: Initialization of seven parameters of CPFA for three different environments

3.3 Generating Data Set for Analysis

Once the parameters are tuned for three different environments, we have generated the data for our analysis using these parameter sets. For each of the experiment, we have extracted pick up and drop off time for each seed, location of each seed in the arena. Pick up time represents the time when the seed is picked by the ant from the location of the seed and drop time is when it is dropped off at the nest. We have tagged each ant with distinct ID. For each seed, we also have extracted which ant has collected that seed. Also, we have tracked when the pheromone is laid, and followed, and when the site fidelity is followed. We have assigned distinct ID number to each pile so that when a pheromone trail is laid we can track which pile the trail is coming from.

For each type of environment, we have simulated 500 experiments and generated data mentioned above. We varied the value of random seed for each experiment while keeping the CPFA parameters constant for a particular environment. We also varied the position of seeds for each experiment. Each experiment was performed for 90 minutes.

While generating the data for "pheromone only parameters", We did not extract any site fidelity data, because we tuned all the parameters not to use site fidelity data. Similarly, for "site fidelity only" experiments we did not extract any pheromone data as there was no pheromone. We have collected both site fidelity data and pheromone data when we have used both methods together for collecting resources.

3.4 Analyzing The Foraging Data

After generating all the data from the simulation we have tried to observe how the ants collect seeds from different food distributions. We have observed that it takes

some time for them to discover the larger piles. But once they discover it, they start to collect seeds from those piles. And they use site fidelity and pheromone for this recruitment. Once they start collecting this seeds we see an increase in their foraging rate. So we tried to detect those changes in their foraging rate by applying the change point detection algorithm.

3.4.1 Creating Timeline for each type of distribution

We have studied each experiment separately to analyze the change in their foraging rate. For each experiment, we studied foraging rate for each type of pile individually. To study foraging rate for each pile we have created a timeline for each type of distribution.

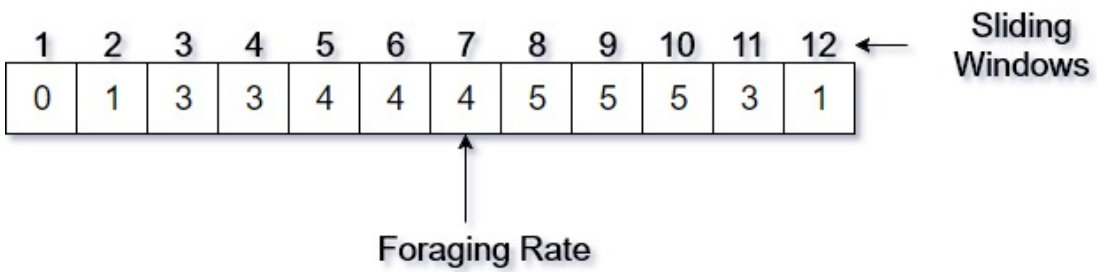


Figure 3.3: An example of a timeline for a distribution where numbers at the top represent the sliding window number. Values in the boxes are the rate of collection of seeds per window.

For example, we have divided the whole time of the experiments into sliding windows. Where each sliding window represents the rate of collection of seeds for that time. The length of the sliding window is 60 seconds for *P. Rugosus*. And then we slided the window by 10 seconds so that we can get the rate of collecting seeds by that window. So if the experiment is for 90 minutes (5400 seconds), we kept the length of the sliding windows for 60 seconds and slided it by 10 seconds, we get total 540 sets of data where we calculated their foraging rate for a particular pile.

Once we have created the timeline for each experiment for a particular distribution of seeds we used Change point detection algorithm to detect the change in the rate of collection of seeds. Another method we have created the timeline is by taking into consideration the change in the rate of foraging. In this method for creating the timeline instead of foraging rate, we take into consideration the change in foraging rate.

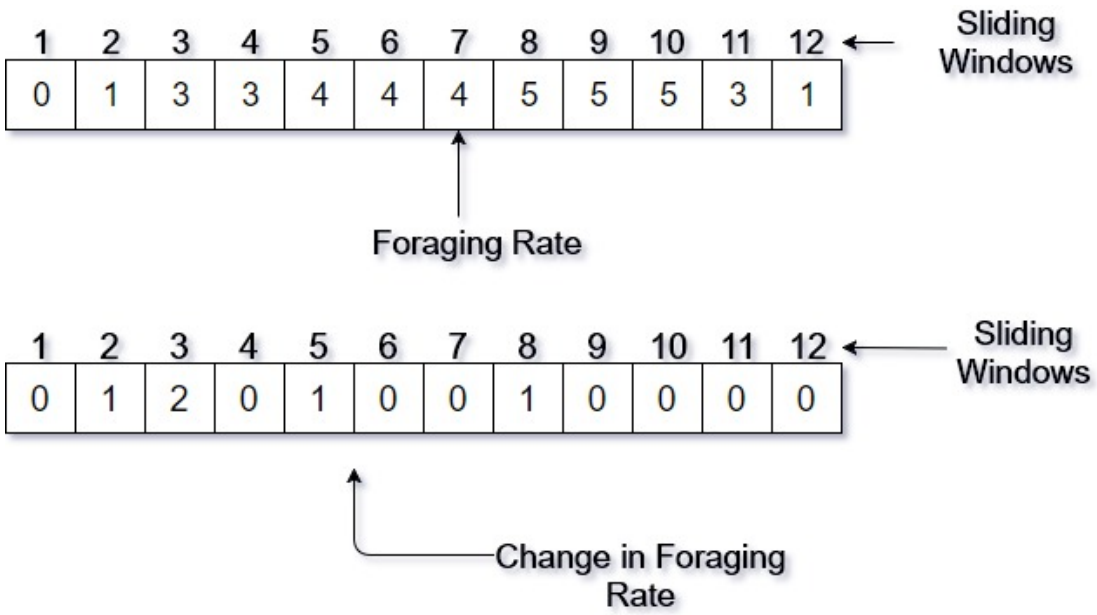


Figure 3.4: An example of timeline and change in foraging rate. The change in foraging rate is calculated by measuring the difference between the timeline windows

The length of the window and the sliding amount for the window is different for each species. We have systematically varied this two parameters to fine tune the change point detection algorithms. Values for each of the parameters will be discussed further in the result section.

3.5 Change Point Detection Algorithm

The change point detection algorithm is divided into two parts. First part is the adding rate of collecting seeds to calculate the cumulative sum and detrend for smoothing. And the second part is applying the change point detection algorithm. We have used binary segmented cumulative sum method to determine the change points.

3.5.1 Calculating the Cumulative Sum

The calculation of cumulative sum is basically adding the foraging rate in each window. Figure 3.5 and algorithm 3.1 demonstrates how the cumulative sum is calculated.

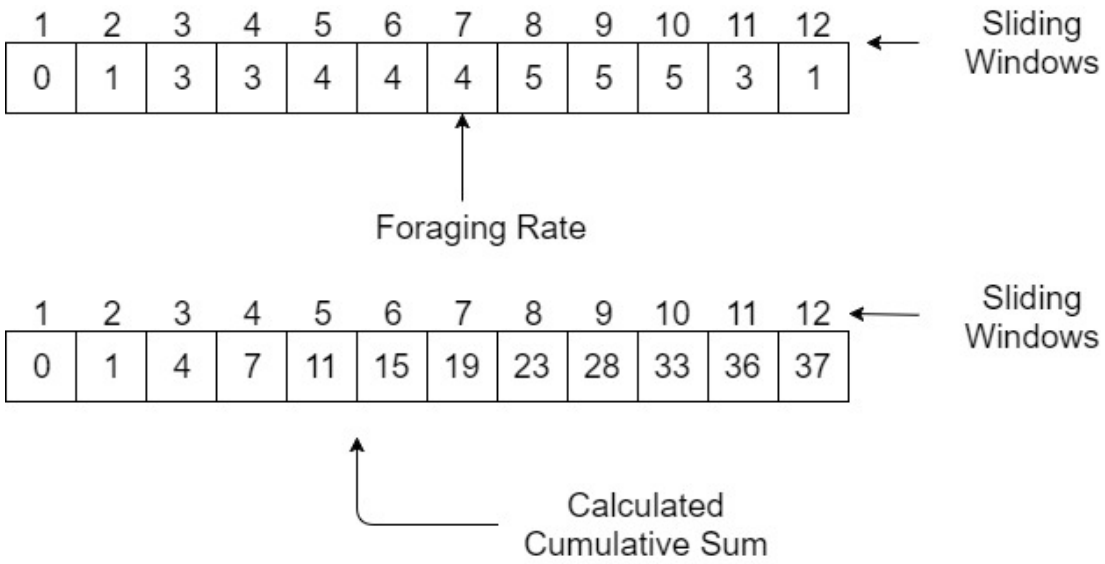


Figure 3.5: This figure demonstrates how the cumulative sum is calculated from the timeline of foraging rate.

Algorithm 3.1 Pseudo code for calculating cumulative sum.

```
1: Sum=0
2: for i=1:Number of Sliding Window do
3:     Sum= Sum + window(i)
4:     CumulativeSum(i)=Sum
5: end for
```

3.5.2 Detrending

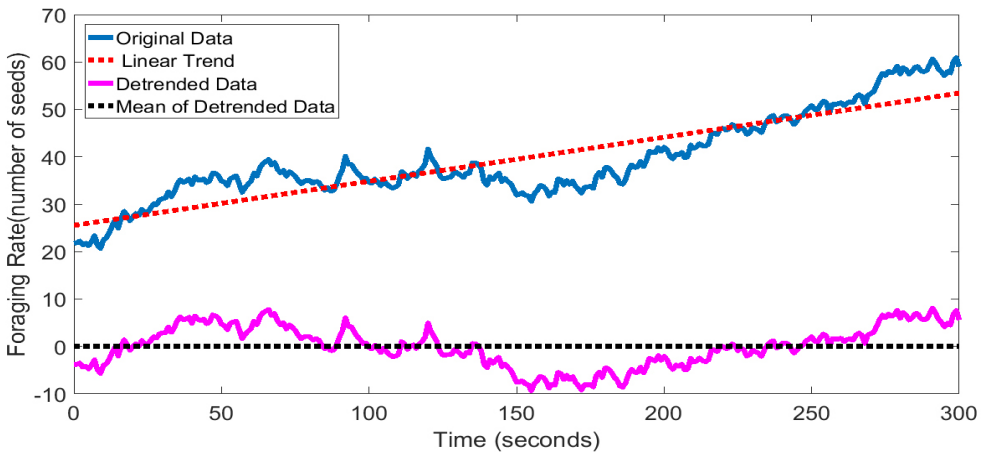
In time series trend means a lazy and gradual change in properties over the whole interval of the event. It is sometimes implicitly defined as a long-term change in the mean. It can also be referred as a change in some statistical properties. Usually, periodic and seasonal components and abrupt fluctuations and other parts were studied separately. Modern analysis techniques frequently treat the series without such routine decomposition, it is still required to consider the trend separately.

The removal of a trend in a statistical or mathematical operation of time series is called detrending. It is often applied to remove features which are obsolete or unimportant. In time series analysis, detrending is also used in preprocessing step to prepare data set for further analysis.

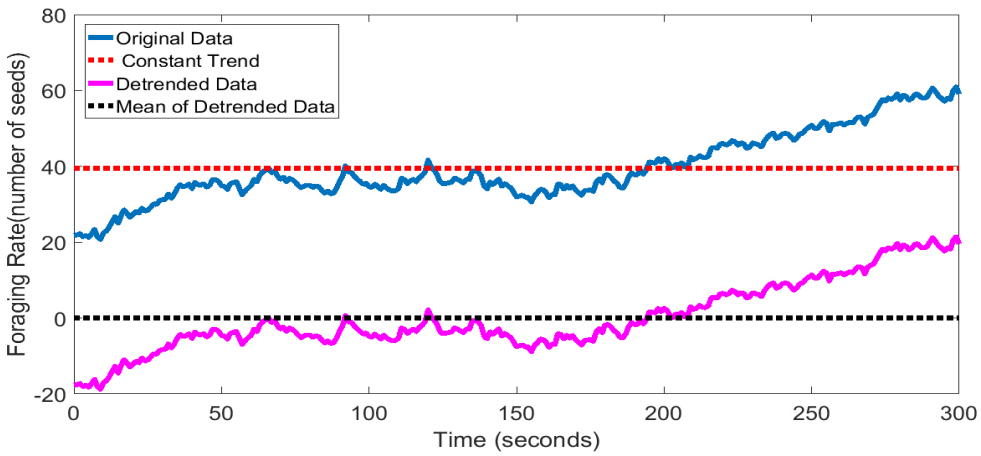
There are several methods of detrending. Linear trends in mean can be truncated by subtracting a least-square-fit straight line. Different procedures are used for more complicated trend. For example, the cubic smoothing spline is commonly used in dendrochronology to fit and remove ring-width trend that might not be linear, or not even monotonically increasing or decreasing over time. It is important to understand the effect of detrending on spectral properties of time series before trying to remove the trend from the time series.

Before applying the change point detection algorithm, we have applied detrending

algorithm to remove the trend from the time series. We used linear detrending and constant detrending to observe the effect of detrending in our time series data. Linear detrending removes the linear trend from the data where constant detrending removes the mean from the data. Figure 3.6 demonstrates how linear and constant detrending affects the time series.



(a) Linear Detrending



(b) Constant Detrending

Figure 3.6: An example of applying linear and constant detrending on the cumulative sum of a timeline from one simulated CPFA experiment.

3.5.3 Binary-Segmented Cumulative Sum

Binary Segmentation is one of the most established search method used for detecting the change point. This method extends any single change point method to multiple change points by iteratively repeating the method on different subsets of the sequence.

To perform binary segmentation, we first apply the chosen single change point detection method to the entire data set, if no change point is found then we are done. If a change point is detected, call this τ, then the data is split into two segments, timeline$[1:\tau]$ and timeline$[\tau+1:n]$. We then apply the single change point method to the two segments and repeat iteratively. We stop when no more change points are detected.

Binary segmentation is a very fast algorithm with complexity $O(n \log n)$ to detect the changes. But the major disadvantage of its computational correctness is that it gives us only an approximation of changes. It is not guaranteed that the binary segmentation method will find us the optimum solution. Also due to iterative nature of this algorithm, it may not detect changes in small changes. Which is why to verify the how well this method is performing, we have verified the results with the simulated data. The pseudo code for the binary segmentation algorithm is given in Algorithm 3.2.

3.6 Verification of Change Points

As we have simulated data, and we know when the pheromones and site fidelities are used in simulations, we can certainly verify how efficient our change points detection algorithms are. So to check how efficient is our algorithms to detect change points, we divided the detection of change points into 4 categories.

Algorithm 3.2 Pseudocode for Binary Segmented Mean Cumulative Sum

Input:A set of data of the form ($value_1$,$value_2$,$value_3$...)
    A test statistic $\tau(.)$
    An estimator of the changepoint position $\tau(.)$
    A rejection threshold β
Initialize: Let $C = \phi$, and $S = [1 : n]$
while $S \neq \phi$ **do**
    Choose an element of S
    Denote this element as $[s, t]$
    if $\tau(ys : t) < \beta$ **then**
        remove$[s, t] from S$
    end if
    if $\tau(ys : t) \geq \beta$ **then**
        remove$[s, t] from S$
        calculate $r = \tau(ys : t) + s - 1$,
        add r to C
        if $r \neq s$ **then**
            add $[s, r]$ to S
        end if
        if $r = t - 1$ **then**
            $[r + 1, t]$ to S
        end if
    end if
end while

- **Catagory A:** Change point detection within 10 seconds of pheromone laying events,

- **Catagory B:** Change point detections within 11-300 seconds of pheromone

laying events,

- **Catagory C:** Change point detections after more than 300 seconds of pheromone laying events and
- **Catagory D:** Change point detected but no pheromone laying events has happened.

3.7 Applying the best method on Field Data

We applied change point detection on 4 different types of the data set. This lead us to evaluate the performance of change point detection algorithm for four different methods. After validating four different methods that we have applied to simulation

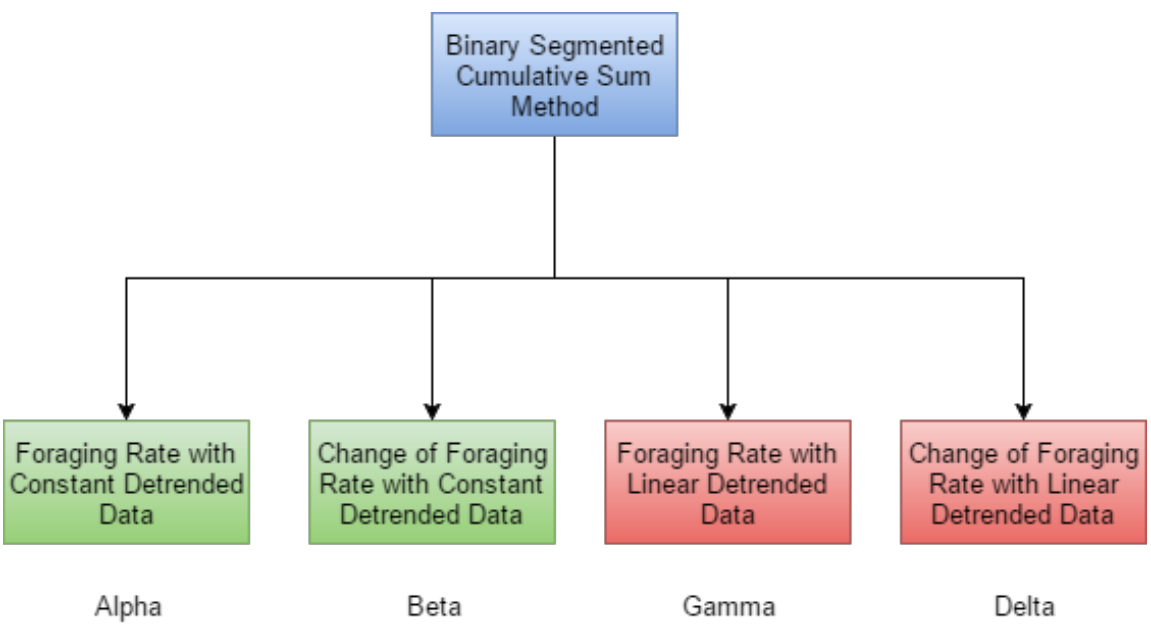


Figure 3.7: Four Different Change point Detection Method

data. We select the best method for them based on the performance on four categories stated above. And apply the best method in our field data.

Chapter 4

Results

4.1 Parameter setup for Change Point Detection Methods

As we know that our change point detection algorithms depends on the values of 3 different parameters- sliding window size, sliding amount and number of change points, we have systematically varied these parameters to obtain the optimum efficiency of change point detection methods for all three species. To tune the sliding window size for each species, we have varied it from 40-120 seconds with an interval of 10 seconds. We have varied the sliding amount from 10-30 seconds with an interval of 10 seconds. For each of the species we have observed the simulations visually to track down, how many times ants discover a particular type of pile and noted that number. We put that data on the algorithm as a statistical parameter.

After comparing the efficiency of all this settings we have tuned this parameters for each of the species. Table 4.1 describes the parameter settings of simulations for each of the species

Species	**Pile**	**Window Size**	**Sliding Amount**	**Mean Number of Change Points**
P. Desertorum	1	80	20	6
	4	80	20	4
	16	80	20	2
P. Maricopa	1	70	20	2
	4	70	20	5
	16	70	20	4
P. Rugosus	1	60	10	4
	4	60	10	4
	16	60	10	4

Table 4.1: Environmental Setup of simulation for three species

4.2 Results from Simulation

We configured the simulation settings as close as we can with the experimental setup of *P. Rugosus*, *P. Maricopa* and *P. Desertorum*. Then we have compared the result of method *alpha*, *beta*, *gamma* and *delta*. based on the time between a pheromone laying event followed by the detection of change point.

Figure 4.1 compares the results of four different methods on a pheromone only simulated environment for *P. Rugosus*. We can see that, *Method* α detects change points which is closer to the pheromone laying event. The result of *Method* β is closer to the *Method* α, but Method γ and Method δ doesn't perform well in this environmental settings.

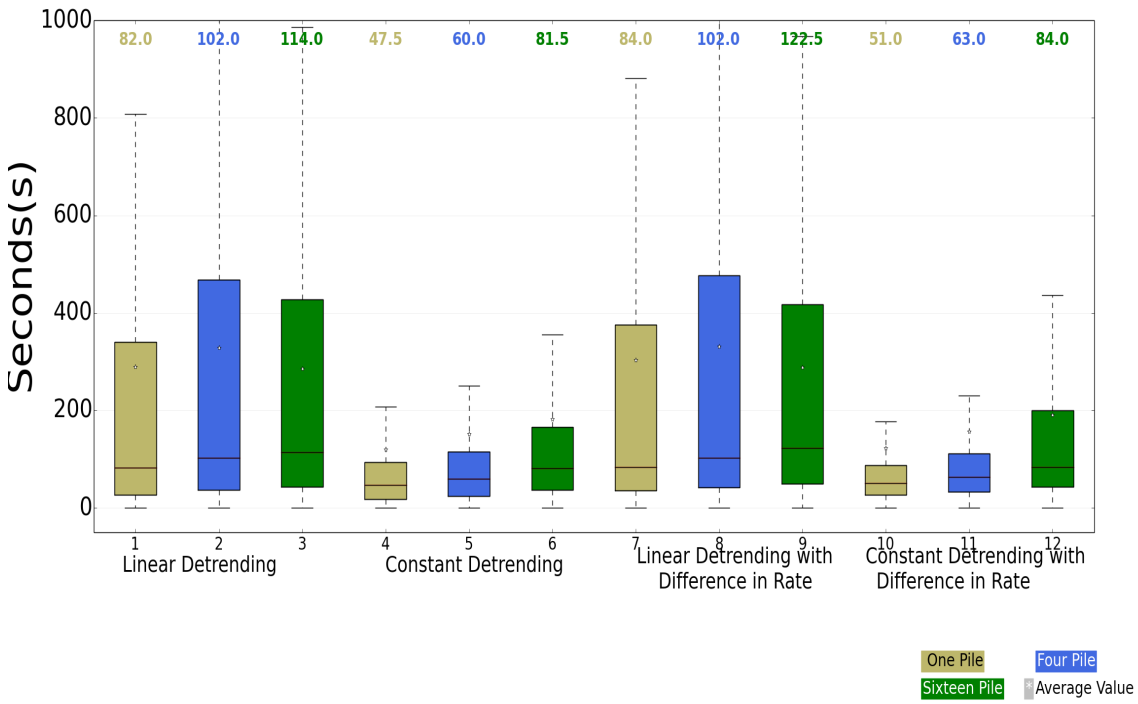


Figure 4.1: Comparison of change point detection method for pheromone only parameters. Outliers are skipped to provide a better perception of differences. The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

We have illustrated the result of *constant detrending* method in figure 4.2. The average difference of pheromone laying event followed by a change point detection for *constant detrending* method is 47.5, 60 and 81.5 for one pile, four large piles and sixteen piles. It is also observed that, the range of time difference between a pheromone laying event followed by a change point detection event is also significantly reduced.

We observe similar phenomenon for *P. Maricopa* and *P. Desertorum*. The box plots of comparison for these two species are provided in the Appendices. The outliers are removed from the figure to make the figure more visible. The detailed figure with

the outliers are provided in the appendices.

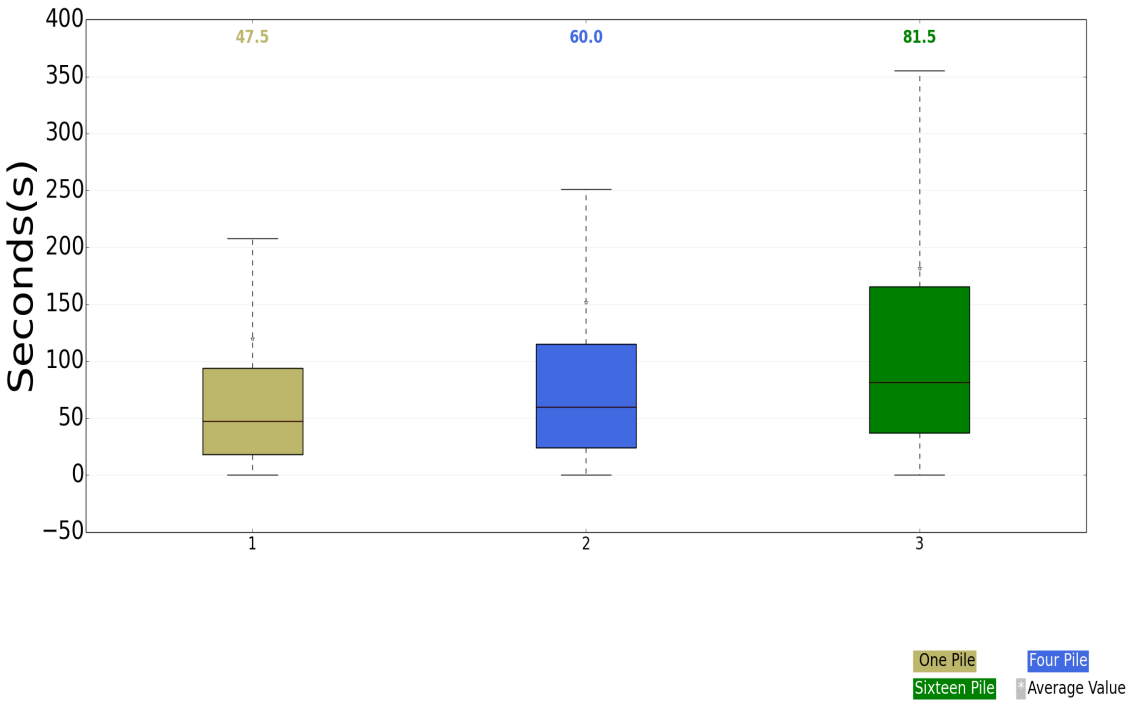


Figure 4.2: Enlarged view of efficiency of *constant detrending* method. The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

We have also evaluated four different methods based on the four categories mentioned in section 3.7. As we can see from figure 4.3 in *constant detrending* method, 15% of the time change points are detected within 10 seconds of laying pheromone, whereas, in *linear detrending* method, it is only 9%. *Constant detrending* method also has significant improvement over *linear detrending* method in *Catagory B*. 74% of the time, changes points are detected with eleven to three hundred seconds of laying pheromone while using *constant detrending* method, on the other hand, using *linear detrending* method it is only 50%. In *catagory C* the error is 6% for *constant detrending* method, whereas, it is 21% for *linear detrending* method.

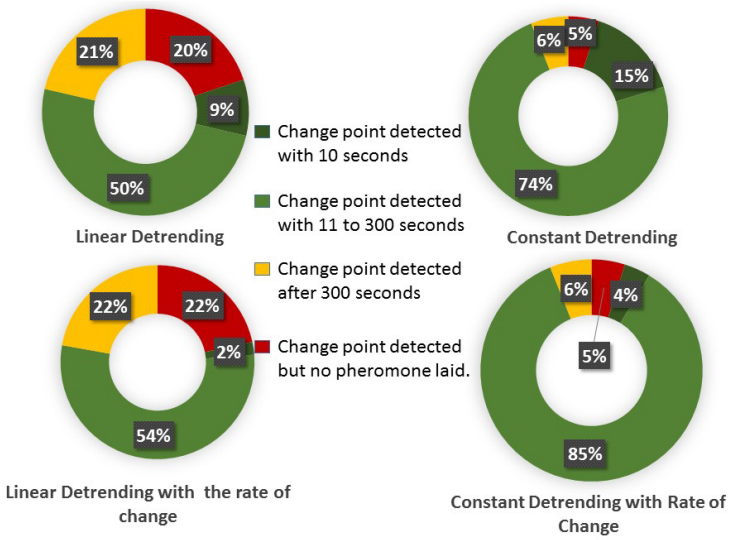


Figure 4.3: Efficiency chart for Change Point Detection Methods on a large pile of 256 seeds. This result is generated from the simulated data of pheromone only environment, configured for *P. Rugosus*

Performance of *constant detrending method* and *constant detrending method with difference in rate* is similar. if we consider categories A and B. But *Method* α does significantly better than *Method* β in detecting change points for four piles and sixteen piles in the simulated environment.

From figure 4.4 and figure 4.5 we can observe that, the performance of *Method* α combined in category A and B is better than performance of *Method* β combined in these two categories. Also from figure 4.4 and 4.5 we can see that *Method* α does better in category C than *Method* β.

If we look at the performance of the four methods over these categories in pheromone only simulated environments, for *P. Maricopa* and *P. Desertorum* we observe similar pattern of performances. The results for *P. Maricopa* and *P. Desertorum* provided in the appendices.

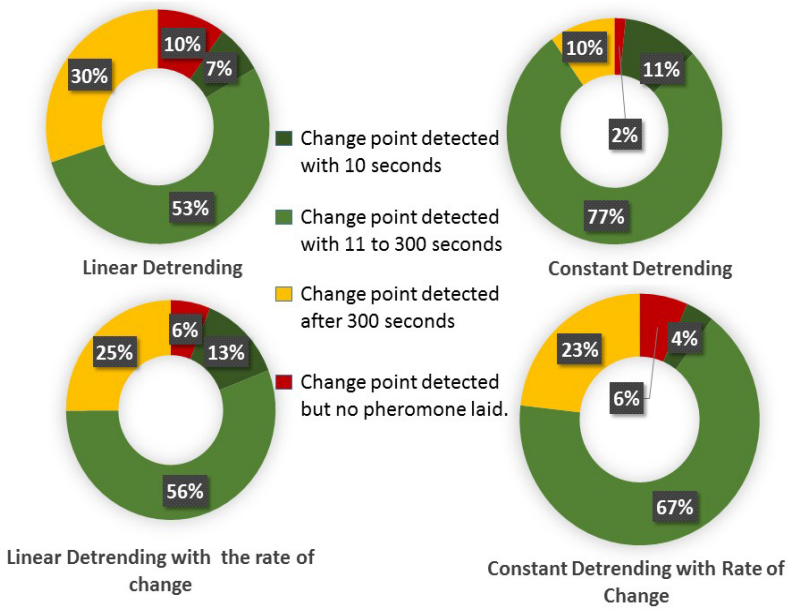


Figure 4.4: Efficiency chart for Change Point Detection Methods on four medium piles of 64 seeds. This result is generated from the simulated data of pheromone only environment, configured for *P. Rugosus*

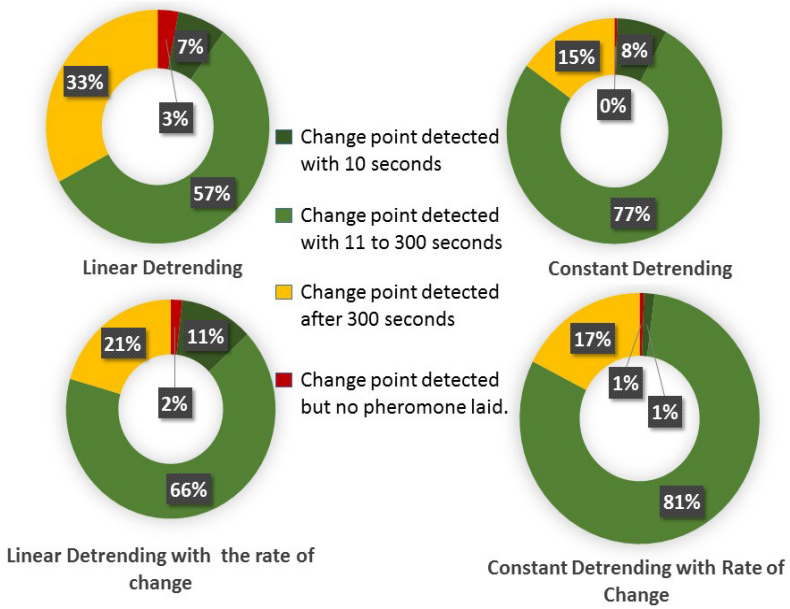


Figure 4.5: Efficiency chart for Change Point Detection Methods on sixteen small piles of 16 seeds. This result is generated from the simulated data of pheromone only environment, configured for *P. Rugosus*

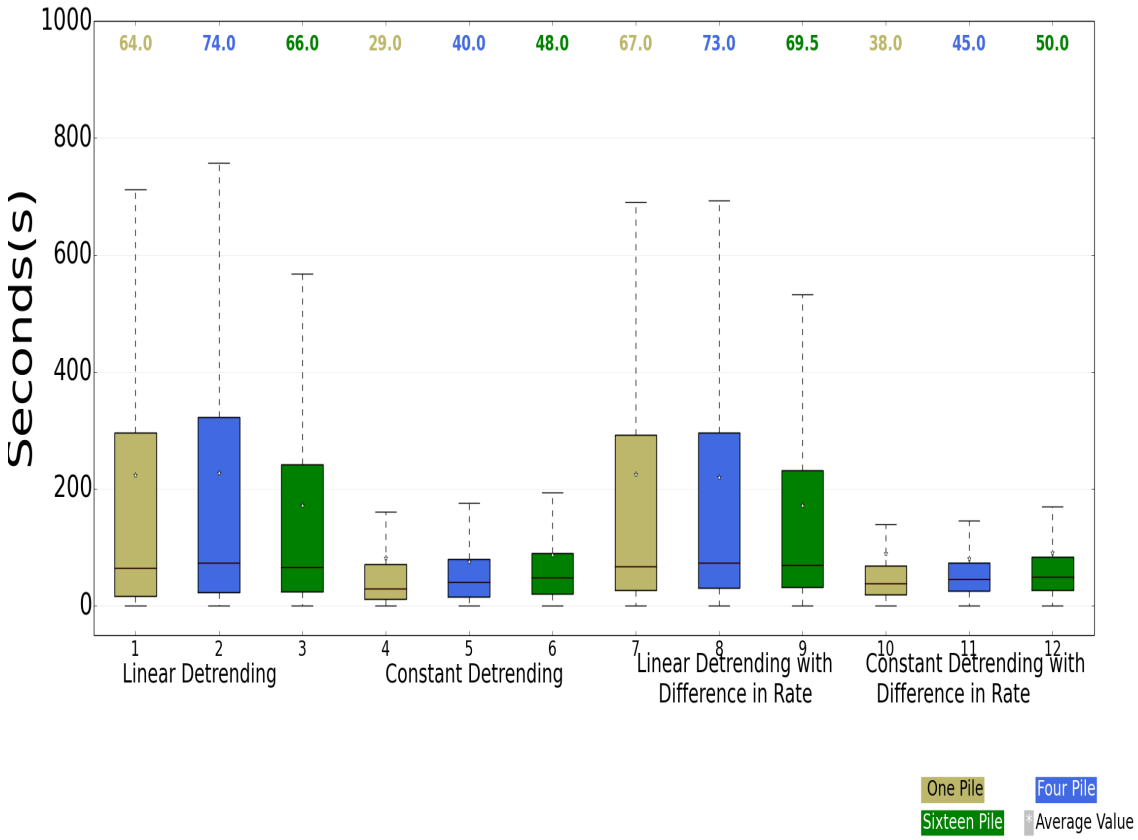


Figure 4.6: Comparison of Change Point Detection Methods without outliers for *pheromone plus sitefidelity* environment . The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

Figure 4.6 shows the comparison of four different methods on a simulated environment with all parameters for *P. Rugosus*. From this figure it is clearly visible that *Method* α does better in detecting the change points than any other methods. As a matter of fact, Method α does better in memory plus communication environment than the communication only environment.

We have analyzed the efficiencies of all four methods for communication plus

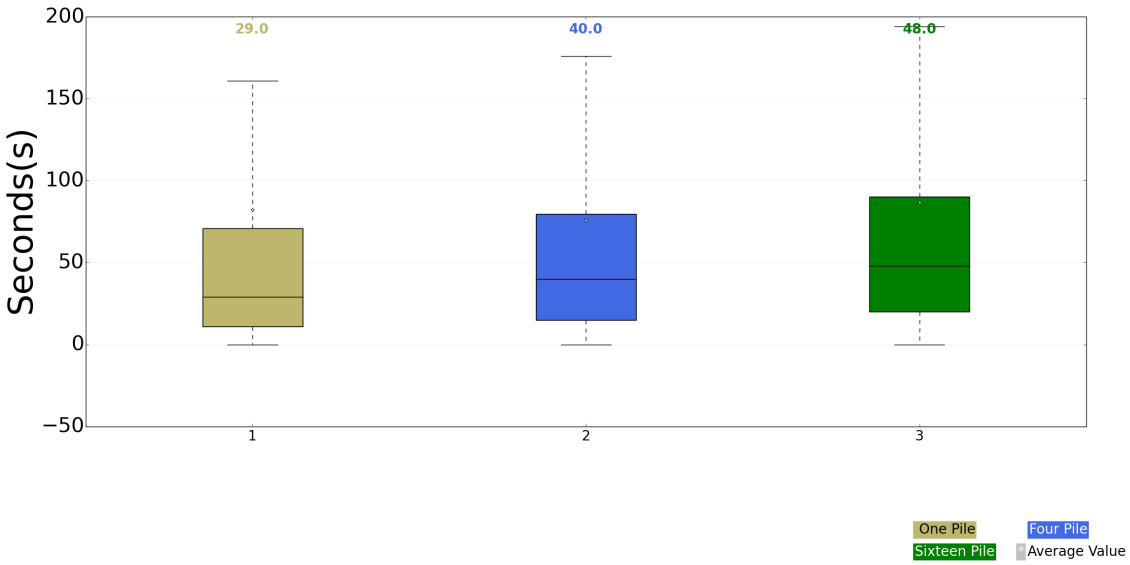


Figure 4.7: Enlarged view of efficiency of*Method* α. The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

memory environmental settings. We observe that *method* α and *method* β does significantly better than the other two methods. But in *category* C and D *method* α does better than *method* β. We observe that the error is 0% for *method* α, whereas it is 1% for *method* β. *Method* α also surpluses *method* β in the combined result of category A and B. We observe, the similar pattern of performances for four piles and sixteen piles.

Figure *4.8(a)* is the efficiency chart for *method* α for one large pile on pheromone plus site fidelity environment. We skipped the results of other piles for *P. Rugosus* as those have similar patterns of result. In figure *4.8(b)* we demonstrate our model's efficiency in detecting the change points when only site fidelity is used for *P. Rugosus* on one large pile of 256 seeds.

In memory only environment when no pheromone is used, agents use memory

(a) (b)

Figure 4.8: 4.8(a) is the efficiency chart of the *constant detrending* method for one large pile of 256 seeds. The data is generated from the simulation of pheromone plus site fidelity environment of *P. Rugosus*. 4.8(b) is the efficiency chart for the constant detrending method for one large pile of 256 seeds. The figure is generated by analyzing the simulated data of sitefidelity only environment for *P. Rugosus*.

extensively. For this reason, when they start collecting seeds, their foraging rate goes up because of the extensive use of internal memory. Which is why *constant detrending* method detects change points in site fidelity only environment also. So we have also measured the efficiency of all methods by measuring the difference between a use of site fidelity and followed by a change point. From figure 4.8(b), we have observed that 59% of the time a change point is detected after the first use of site-fidelity.

The performance rating is similar for *P. Maricopa* and *P. Desertorum*. Results for these two species are provided in the appendices.

4.3 Results from Field Data

We applied binary segmented cumulative sum change point detection method with constant detrending on the field data. The parameters are kept same as simulations for the change point detection method. We are able to detect change points in 6 experiments out of 11 for *P. Desertorum*. For *P. Maricopa*, we detected change

points in 7 experiments out of 11 and for *P. Rugosus* we are able to detect change points in 12 experiments out of 13. Table 4.2 shows the detail of the results for field data.

P. Desertorum			
Experiment	One	Four	Sixteen
DP15_070709	6	4	2
DP10_0606	6	4	2
DP14_0606	6	4	1
DP15_0604	6	0	2
DP15_0617	6	4	1
DPC_070109	6	4	2

P. Maricopa			
Experiment	One	Four	Sixteen
MP8_0609	2	5	4
MP9_0528	2	0	0
MP9_0602	2	5	4
MP9_0611	2	4	1
MP9_0620	2	5	4
MPE_24Jul09	2	5	4
MPX_0624	2	5	4

P. Rugosus			
Experiment	One	Four	Sixteen
RP6_0609	4	4	4
RP7_0610	4	4	4
RP10_0527	4	4	3
RP10_0528	4	4	4
RP10_0604	3	0	3
RP12_0602	4	4	4
RP14_0610	4	4	0
RP14_0610	4	4	0
RP16_0612	4	4	4
RP19_0618	4	4	4
RS23_0821	4	4	4
RP17_0617	4	4	4

Table 4.2: This table represents the number of change points detected in each of the field experiment of *P. Desertorum*, *P. Maricopa* and *P. Rugosus*. For *P. Desertorum*, we are able to detect change points in 6 experiments out of 11. For *P. Maricopa* it is 7 out of 11. For *P. Rugosus* it is 12 out of 13. The numbers conclude how many times change points are detected for a pile type in each experiment.

From table 4.2 we can see that *P. Desertorum* is using much more pheromone for the large pile. It is discovering the pile more frequently than any other species. But the reason of having more change points in the foraging data of *Desertorum* is, it

discovers and looses pile very frequently. *Maricopa* also detects the large pile. from the data is clearly visible that it does not looses the pile often.

We ran 500 simulated experiment to generate the data for each species. And we are able to detect change points in all of the experiments. This is because agents are programmed to forage despite the situations that harvester ants face. Foraging of harvester ants depends on many factors, including the weather, availability of resources and so many. As a result in some of the experiments we were not been able to detect enough amount of foraging from the ants.

Chapter 5

Conclusion

5.1 Overview

The classic approach to proving a theorem is some really difficult mathematics. For the theory of relativity, I asked grandpa Al exactly how he proved it. He gave me a few hints, including some stuff about rest mass and big electro-motive force. I think he is really smart.

5.2 Conclusions

I conclude that this is a really short thesis.

Chapter 6

Appendicies

6.1 Overview

The classic approach to proving a theorem is some really difficult mathematics. For the theory of relativity, I asked grandpa Al exactly how he proved it. He gave me a few hints, including some stuff about rest mass and big electro-motive force. I think he is really smart.

6.2 Conclusions

I conclude that this is a really short thesis.

References

[1] Social insect. https://www.britannica.com/animal/social-insect.

[2] Carl Anderson and Daniel W McShea. Individual versus social complexity, with particular reference to ant colonies. *Biological reviews*, 76(2):211–237, 2001.

[3] Samuel N Beshers and Jennifer H Fewell. Models of division of labor in social insects. *Annual review of entomology*, 46(1):413–440, 2001.

[4] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. Inspiration for optimization from social insect behaviour. *Nature*, 406(6791):39–42, 2000.

[5] Matthew Collett. How desert ants use a visual landmark for guidance along a habitual route. *Proceedings of the National Academy of Sciences*, 107(25):11638–11643, 2010.

[6] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[7] Tatiana P Flanagan, Kenneth Letendre, William R Burnside, G Matthew Fricke, and Melanie E Moses. Quantifying the effect of colony size and food distribution on harvester ant foraging. *PloS one*, 7(7):e39427, 2012.

[8] Piotr Fryzlewicz et al. Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281, 2014.

[9] Deborah M Gordon and Alan W Kulig. Founding, foraging, and fighting: colony size and the spatial distribution of harvester ant nests. *Ecology*, 77(8):2393–2409, 1996.

[10] Dongxiao He, Jie Liu, Dayou Liu, Di Jin, and Zhengxue Jia. Ant colony optimization for community detection in large-scale complex networks. In *Natural*

Computation (ICNC), 2011 Seventh International Conference on, volume 2, pages 1151–1155. IEEE, 2011.

[11] Joshua P Hecker and Melanie E Moses. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence*, 9(1):43–70, 2015.

[12] Richard J Hobbs. Harvester ant foraging and plant species distribution in annual grassland. *Oecologia*, 67(4):519–523, 1985.

[13] Duncan E Jackson and Francis LW Ratnieks. Communication in ants. *Current biology*, 16(15):R570–R574, 2006.

[14] Tomasz Kukulski, Laila Hübbert, Martina Arnold, Bengt Wranne, Liv Hatle, and George R Sutherland. Normal regional right ventricular function and its change with age: a doppler myocardial imaging study. *Journal of the American Society of Echocardiography*, 13(3):194–204, 2000.

[15] W Narzt, U Wilflingseder, G Pomberger, D Kolb, and H Hörtner. Self-organising congestion evasion strategies using ant-based pheromones. *IET Intelligent Transport Systems*, 4(1):93–102, 2010.

[16] ANDREW JHON Scott and M Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, pages 507–512, 1974.

[17] Peter Tarasewich and Patrick R McMullen. Swarm intelligence: power in numbers. *Communications of the ACM*, 45(8):62–67, 2002.

[18] James FA Traniello. Foraging strategies of ants. *Annual review of entomology*, 34(1):191–210, 1989.

[19] Walter G Whitford. Foraging in seed-harvester ants progonomyrmex spp. *Ecology*, 59(1):185–189, 1978.

[20] Walter G Whitford and George Ettershank. Factors affecting foraging activity in chihuahuan desert harvester ants. *Environmental Entomology*, 4(5):689–696, 1975.

[21] Edward O Wilson and Bert Hölldobler. Eusociality: origin and consequences. *Proceedings of the National Academy of Sciences of the United States of America*, 102(38):13367–13371, 2005.