Spring 4-25-2017

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

Safeeul Bashir Safee
University of New Mexico - Main Campus

Safeeul Bashir Safee
Candidate

Computer Science
Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Dr. Melanie E. Moses, Chairperson

Dr. Abdullah A. Mueen

Dr. Tatiana P. Flanagan

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.

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 also would like to thank Dr. Tatiana P. Flanagan for her initial guidance 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, such as ants. During foraging, ants keep track of food sources by using memory (site fidelity) or by communicating 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 it is beneficial for ants recruit nest mates to collect seeds from large clusters. However, we do not know when the recruitment occurs in natural ant population. To explore this question, we used a power law distribution to arrange seeds in piles of different sizes. We observe that simulated ants use pheromone more when the food sources are clumped, and they re-discover the pile more frequently if the food source is large. Simulated ants don't use the pheromone to recruit from the food source that is scattered in the environment. We use simulations to determine how to correlate change points with recruitment events, and then use that relationship to infer recruitment event in in field data. 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.

Contents

List of Figures

List of Figures

List of Tables

List of Algorithms

Glossary

GA	Genetic Algorithm
CPFA	Central Place Foraging Algorithm
Catagory A	Change point detected with 10 seconds of laying pheromone
Catagory B	Change point detected with 11-300 seconds of laying pheromone
Catagory C	Change point detected after 300 seconds of laying pheromone
Catagory D	Change point detected but no pheromone laid

Chapter 1

Introduction

Social insects are species that live in colonies and manifest three characteristics [1]; a. group integration[2], b. division of labor[3] and c. overlapping generations[22]. These creatures have survived many mass level extinction events. Millions of years of genetic evolution helped them to adapt to the environment and to master the strategies of survival.

Their strategies to avoid congestion and to optimize their movements to forage in most efficient ways without any central authority have attracted researchers and scientists[16]. 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 forage during the morning or in the evening sessions during the summer[12, 21]. In this study, we will mostly talk about *Progonomyrmex* species, which are group foragers[20]. Foraging activities of harvester ants depend on many factors such as temperature, light, and availability of seeds[21].

Social insects such as ants sometimes use pheromone to communicate with each

other to forage[13]. The previous study has shown that they follow three strategies to forage[7, 8, 15]. Ants use memory to remember the location of the food source. They communicate with other ants using pheromone, and they perform random walk in search of food.

Foraging depends mostly on how food is distributed in the environment[19, 15]. To analyze the foraging strategies of ants, field experiments have been conducted on three species of *Pogonomyrmex* desert harvester ants. Foods were distributed around 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 called the Central Place Foraging Algorithm(CPFA)[11] was developed by Hecker and Moses. The purpose of CPFA is to collect resources from the spatial environment by using the strategies mentioned above: memory, communication and random walk[5, 7, 15].However, it is not clear which strategies are used under which conditions in real ants[18]. It is possible 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 use a change point detection algorithm to detect that change in foraging rate. Our goal is to detect when ants use pheromones rather than site fidelity. Because field data are noisy and we have no ground truth, we use simulations to test how reliably change point detection is at identifying pheromone use. We then use the method to identify change points and infer pheromone use on the field data.

We used the CPFA to simulate the field experiments. The environment of the CPFA has been designed 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[9, 17, 14]. The change point detection algorithms were tuned to detect change points when the pheromone is laid.

From the simulation, we know exactly when the pheromone was laid and site fidelity was used, thus 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 best algorithm with tuned parameters on the field data.

We observe that in simulation 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 that is scattered in the environment.

1.1 Contribution and Organization

Chapter 2 describes species of the ants that we have used in our experiments, the ant inspired model CPFA and the Genetic Algorithm for tuning the parameters.

Chapter 3 describes the procedure that we have followed to investigate the problem. It includes the simulation environment, parameter settings for genetic algorithm, methods and efficiencies of different approaches of change point algorithms.

Chapter 4 includes the results of different change point detection methods and the result of applying best change point detection algorithm on the field data.

Chapter 5 includes conclusion and future works.

Appendices are included at the end.

Chapter 2

Background

Ants are social insects that have evolved millions of years. Their strategies to find resources for their survival are fascinating. prior works describes several experiments on desert harvester ants at the field to observe how they forage[7]. Three different species of harvester ants are *P. rugosus*, *P. desertorum* and *P. maricopa*.

Our goal was to figure out how these ants find resources from an environment and what is the effect of the distribution of seeds on which foraging behavior is used. Seeds in the fields are distributed in a ring showning figure 2.1. The area of the food distribution is scaled with the colony size of ants. For example, *Desertorum* was the smallest in colony size (77±296), so the ring radius of food was 1.5 to 3 meters. *Rugosus* has a colony size of 1712±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, 1024 seeds can be divided into four pile size categories with 256 seeds in each. One large pile of 256 seeds are placed all together in a random location in the ring. Next 256 seeds are divided into 4 equal sizes of 64 seeds and placed in the ring. Next 256 seeds are equally divided into 16 piles of 16 seeds and placed inside the ring. The remaining seeds are distributed uniformly around the nest.

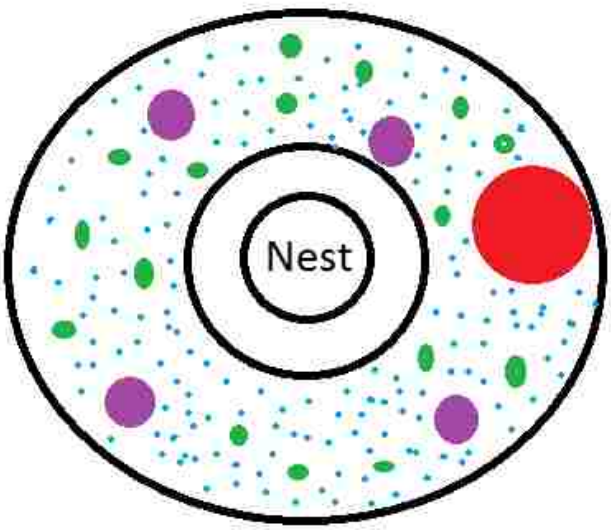


Figure 2.1: Distribution of Seeds in the field experiment for power law distribution with 1024 seeds. Red pile indicates one large pile of 256 seeds. 4 purple piles represent 4 large piles of 64 seeds, green color represents 16 piles of 16 seeds and blue seeds are 256 random seeds

2.2 CPFA

Central Place Foraging Algorithm(CPFA) is an agent-based model where agents are programmed to follow an ant inspired strategy to collect seeds. In CPFA, ants follow three methods to collect seeds[11].

- **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 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.

Whether the agents will use any of three strategies is governed by seven parameters. Table 2.1 represents the seven parameters, that govern CPFA and their initialization values[11].

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

For parameters in a uniform distribution, higher the value of the parameter the higher the probability of that event. For example, "probability of switching to searching" follows a uniform distribution. Higher values of this parameter, create a higher probability that the ant will switch to search from initial departure.

For the exponential distribution, the higher the value of the parameter, the 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.

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. 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 direction to travel and search for resources in that direction. The higher the value of switching to the search, the faster they begin the search process.

The *probability of returning to nest* defines the chances of returning to nest for unsuccessful foraging trip. This parameter actually decides how long the ant will keep searching new places before returning to the nest. Higher the value of the

parameter, mean that ants search shorter before returning to the nest..

The *rate of site fidelity* comes into play when the ants use site fidelity to go to an informed location where that ant remembers that the food already exists. The lower the value of this parameter, the 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 high pheromone stays in the ground for a longer time. The pheromone evaporates faster for the lower value of the parameter.

2.3 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 individual 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.

In our genetic algorithm, the strings that undergo mutation represents the seven parameters of CPFA. Mutation rate is fixed to one percent.

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, in this case 50 generations. The GA can also be terminated by any combination of the termination methods mentioned above. 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**

Chapter 3

Method

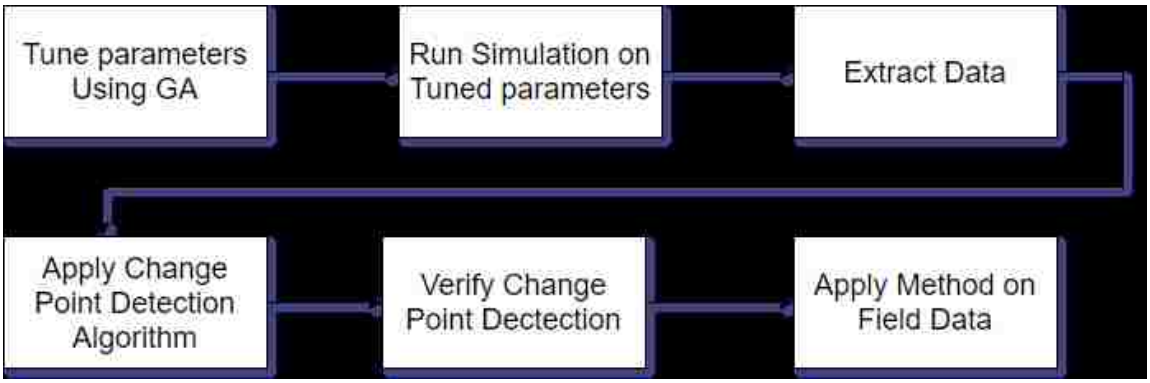


Figure 3.1: Steps of Change Point Analysis

3.1 Setting Simulation Environment

We setup the simulation environment to mimic the field experiments. So we distributed the resources as shown in figure 3.2 . We had three different setups for three different species of ants. The number of agents were 12, 48 and 96. The total duration of each experiment was 90 minutes (We collected data from field experi-

ments 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. 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 eliminated the use of site fidelity. Which means that the probability of using site fidelity is 20, and remaining parameters are evolved using the GA.

2. **Site fidelity Only Parameters:** In this type of experiment we eliminated 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

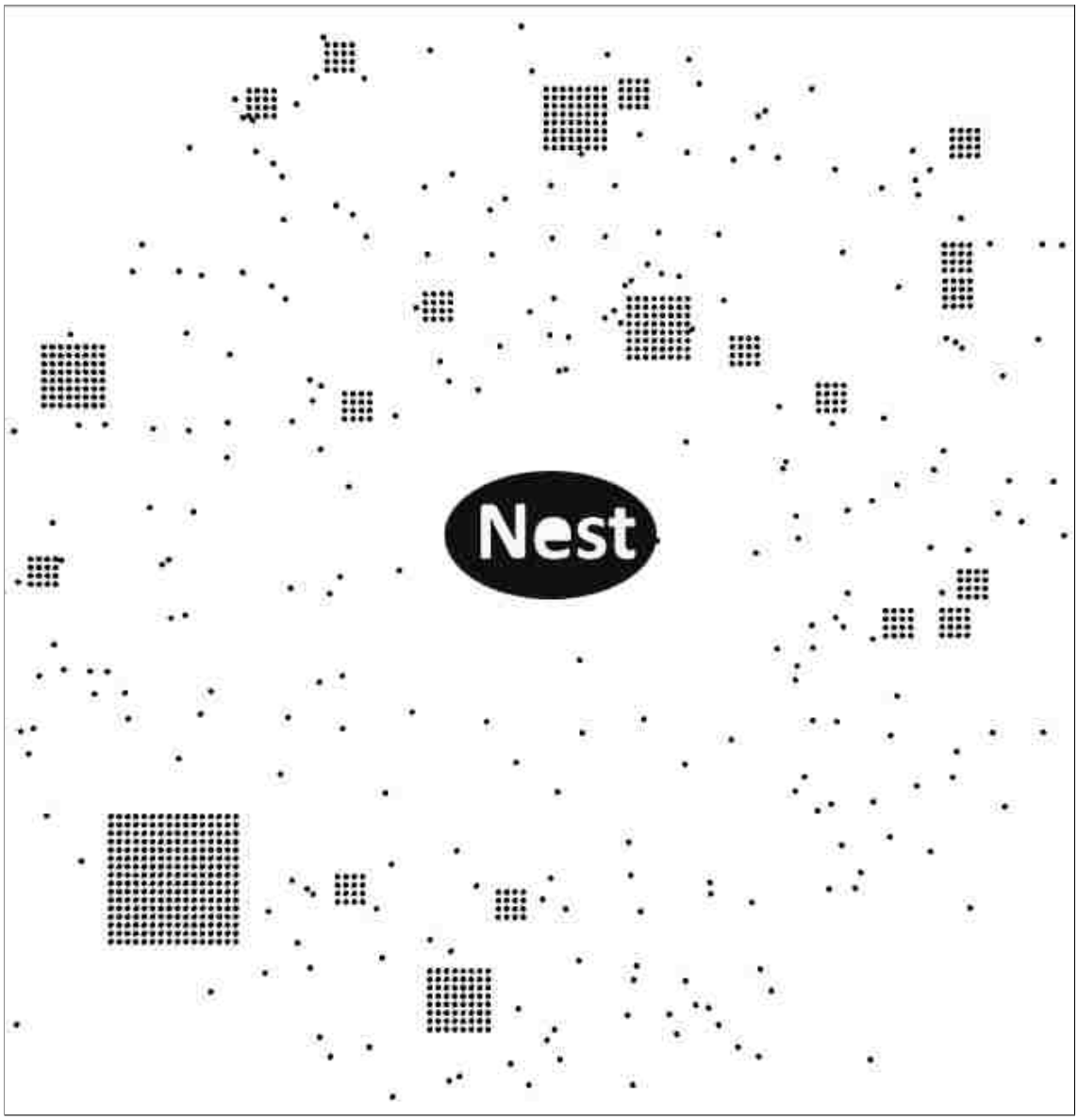


Figure 3.2: Example setup of a simulation environment for *P. rugosus* with 1024 seeds three different types of piles. One large pile of 256 seeds, four piles of 64 seeds and sixteen piles of 16 seeds. 256 random seeds, are distributed uniformly inside the ring.

the actual field experiment condition in which 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 where fitness is defined as maximizing the number of seeds collected in 90 minutes. Initially, we have created a population of one hundred colonies in the 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 value of random seed 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.

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 drop off time for each seed, location of each seed in the arena. 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. Once they discover it, they start to collect seeds from those piles. 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 time-line for each type of distribution. We have calculated foraging rate in overlapping windows, where each

window has a fixed size and slided by a fixed time (such as 10 seconds) to create overlapping windows. Figure 3.3 shows the overlapping window for time-line.

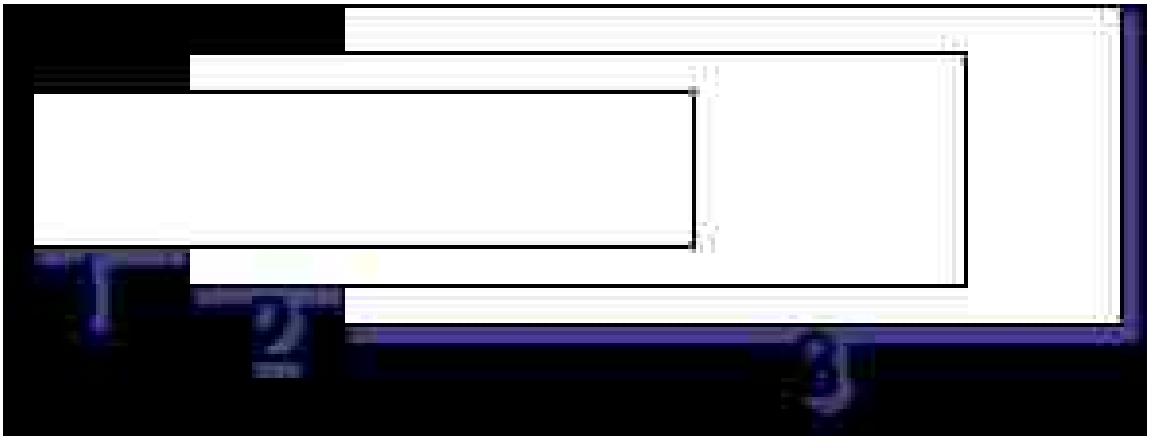


Figure 3.3: An example of overlapping windows of foraging rate.

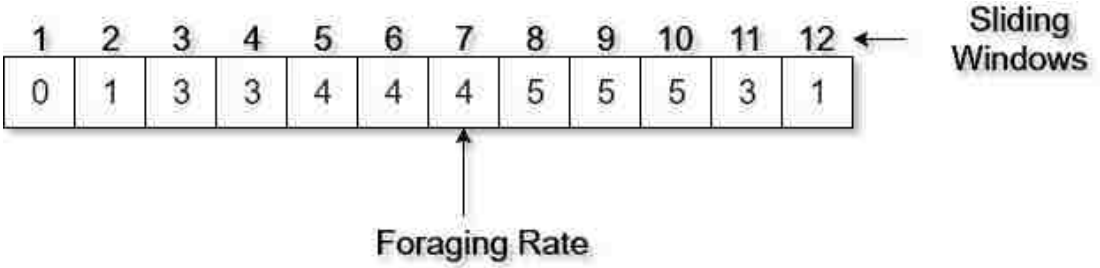


Figure 3.4: An example of a timeline for a distribution where numbers at the top represent the sliding window number. Values in the boxes are the rate of collection of seeds per window.

The length of the window is fixed to average time required for two round trips unless stated otherwise. For the simulated experiments it is 266 seconds. Sliding amount is fixed to 10 seconds. So if the experiment is for 90 minutes (5400 seconds), we kept the length of the sliding windows for 266 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 time-line 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.

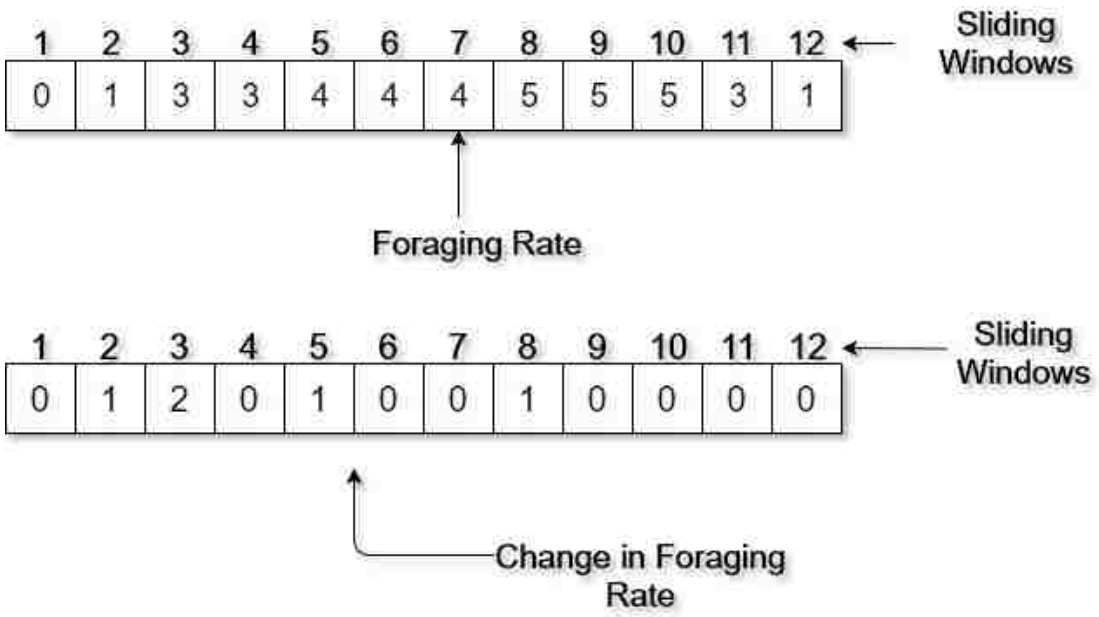


Figure 3.5: An example of timeline and change in foraging rate. The change in foraging rate is calculated by measuring the difference between the timeline windows

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 3.6 and algorithm 3.1 demonstrates how the cumulative sum is calculated.

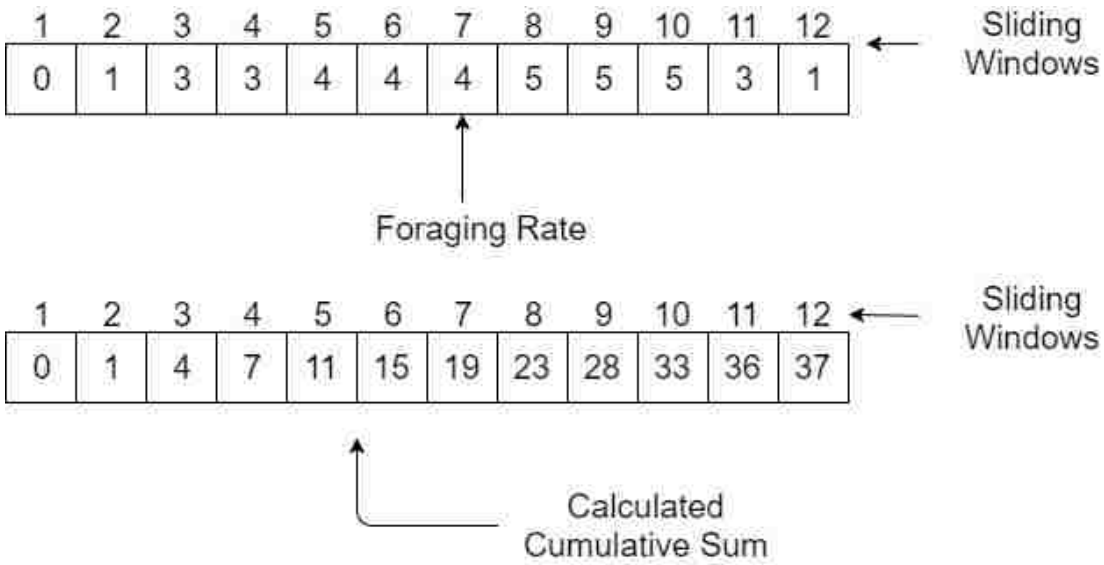


Figure 3.6: This figure demonstrates how the cumulative sum is calculated from the timeline of foraging rate.

Algorithm 3.1 Pseudo code for calculating cumulative sum.

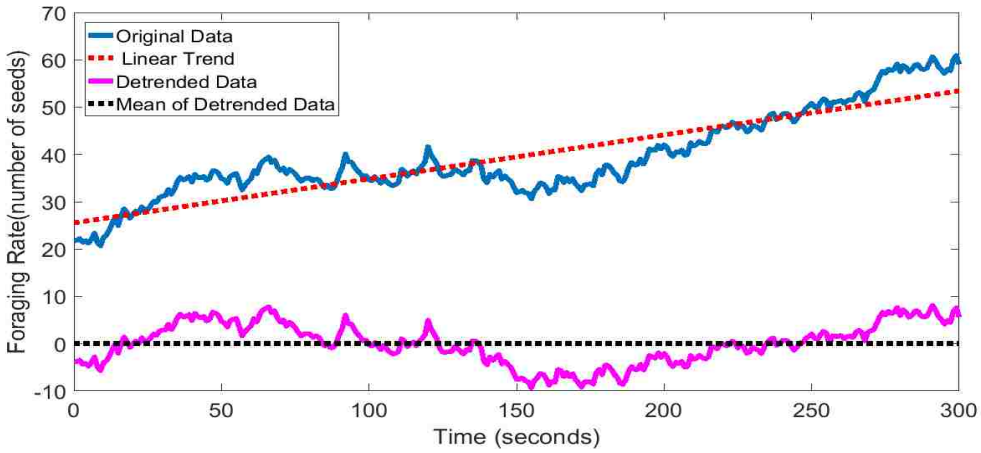
```
1: Sum=0
2: for i=1:Number of Sliding Window do
3:     Sum= Sum + window(i)
4:     CumulativeSum(i)=Sum
5: end for
```

3.5.2 Detrending

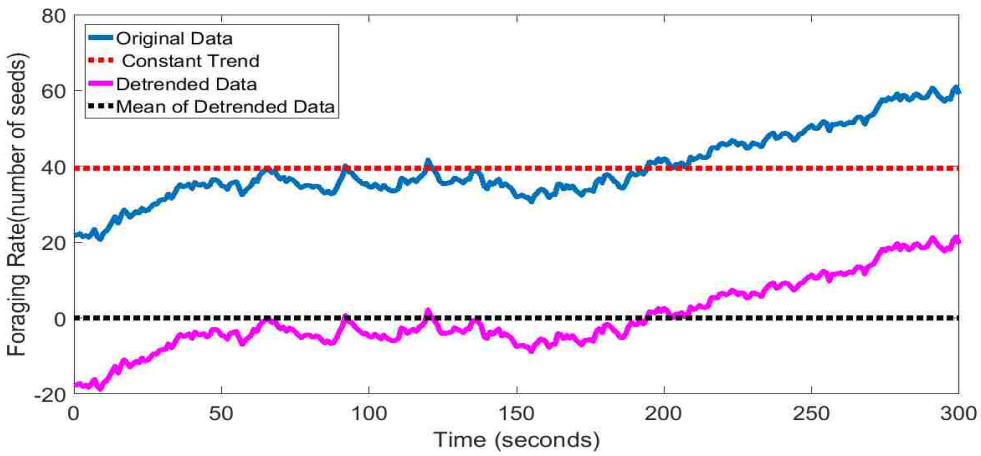
A time series trend is defined as a long-term change in the mean. The removal of a trend in a statistical or mathematical operation of time series is called detrending. It is often applied to remove features which are obsolete or unimportant. In time series analysis, detrending is also used in preprocessing 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 3.7 demonstrates how linear and constant detrending affect the time series.



(a) Linear Detrending



(b) Constant Detrending

Figure 3.7: 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 speed 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 small changes. Thus, 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.

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

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.

- **Catagory A or** ≤ 10**:** Change point detection within 10 seconds of pheromone laying events,
- **Catagory B or** $11 - 300$**:** Change point detections within 11-300 seconds of pheromone laying events,
- **Catagory C or** > 300**:** Change point detections after more than 300 seconds of pheromone laying events and
- **Catagory D or None:** 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 6 different types of the data set. This led us to evaluate the performance of change point detection algorithm for six different methods. After validating six different methods that we have applied to simulation

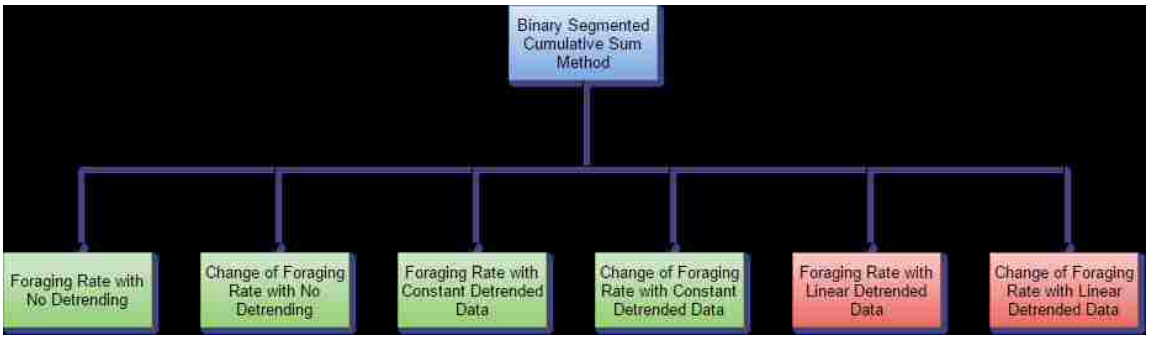


Figure 3.8: Change point analysis on an alternate dataset

data. We select the best method for them based on the performance on six categories stated above, and we apply the best method in our field data.

Chapter 4

Results

4.1 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 change point analysis on *foraging rate*, *change in foraging rate*, *constant detrended foraging rate* , *constant detrended change in foraging rate*, *linear detrended foraging rate* and *linear detrended change in foraging rate*.

Figure 4.1 compares the results of four different methods on a pheromone only simulated environment for *P. rugosus*. We can see that, *constant detrending* detects change points which is closer to the pheromone laying event. The result of *constant detrending with change in foraging rate* is closer to the *constant detrending*, but *linear detrending* and *linear detrending with change in rate* doesn't perform well in this environmental settings.

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

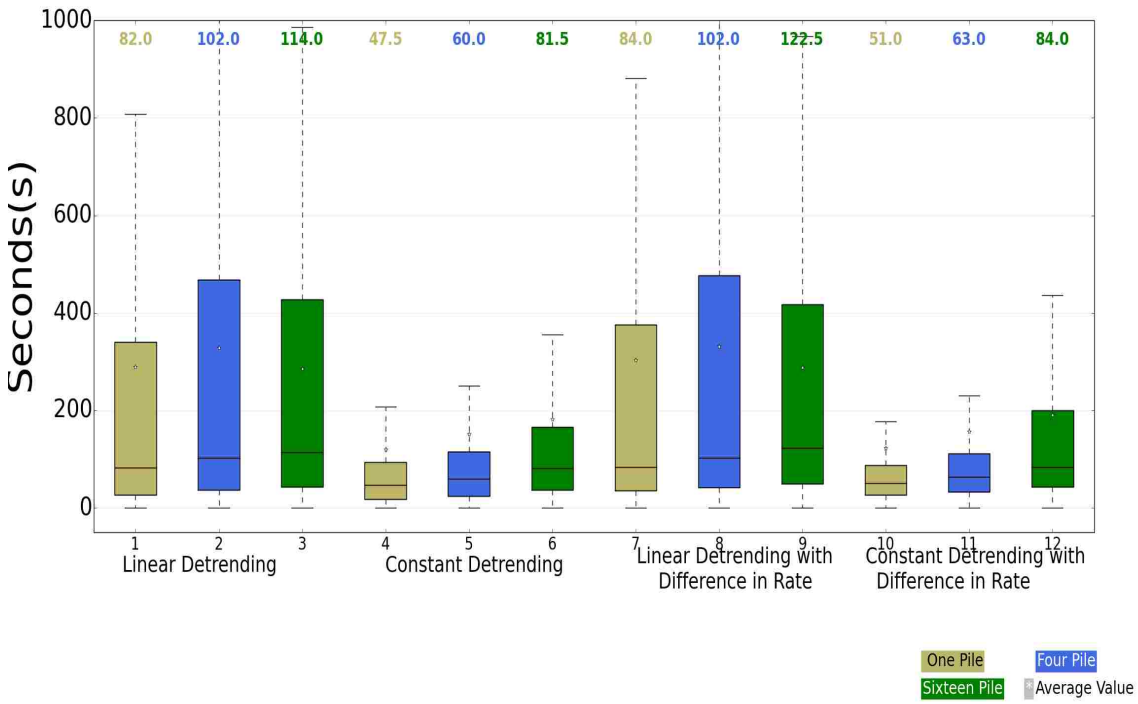


Figure 4.1: Comparison of change point detection method for pheromone only parameters. Outliers are skipped to provide a better indication of differences. The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

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 informative. The detailed figure with the outliers are provided in the appendices.

We compare the result with the change point detection in raw foraging rate. We observe that, the average time difference between change point detection and

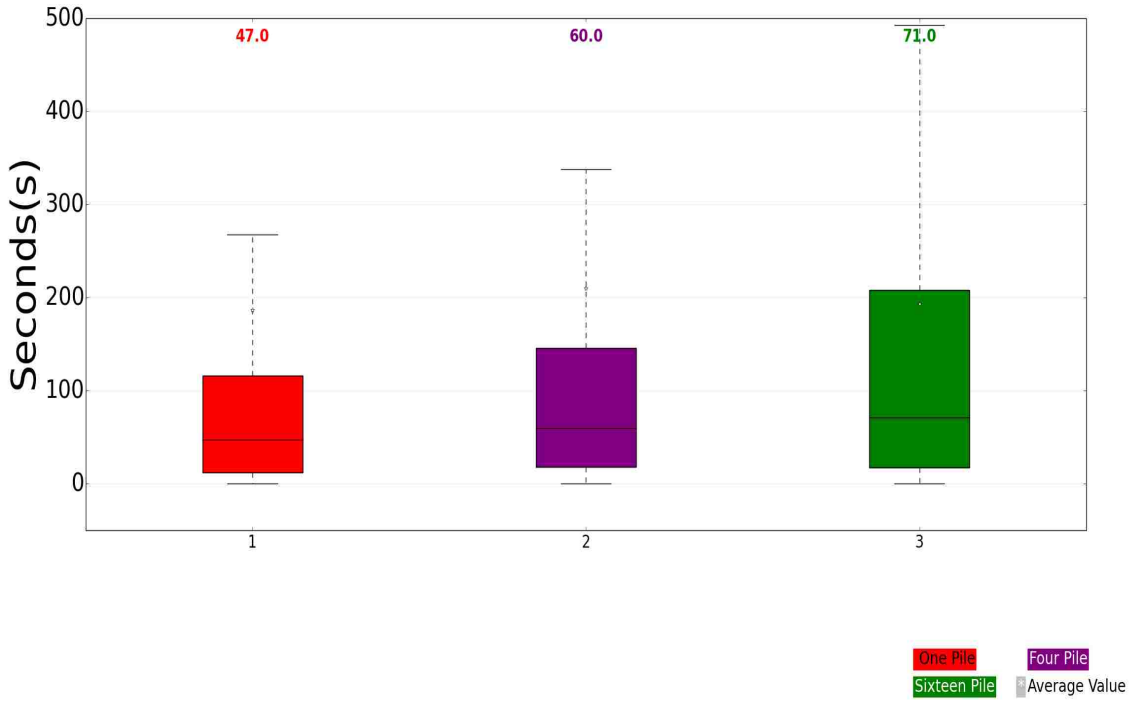


Figure 4.2: Enlarged view of efficiency of change point detection algorithm on raw foraging rate. The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

pheromone laying event is similar to *constant detrending*.

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* , 15% of the time change points are detected within 10 seconds of laying pheromone, whereas, in *linear detrending* , it is only 9%. *Constant detrending* also has significant improvement over *linear detrending* in *Catagory* $11 - 300$. 74% of the time, changes points are detected with eleven to three hundred seconds of laying pheromone while using *constant detrending* , on the other hand, using *linear detrending* it is only 50%. In *catagory* > 300 the error is 6% for *constant detrending* , whereas, it is 21% for *linear detrending*.

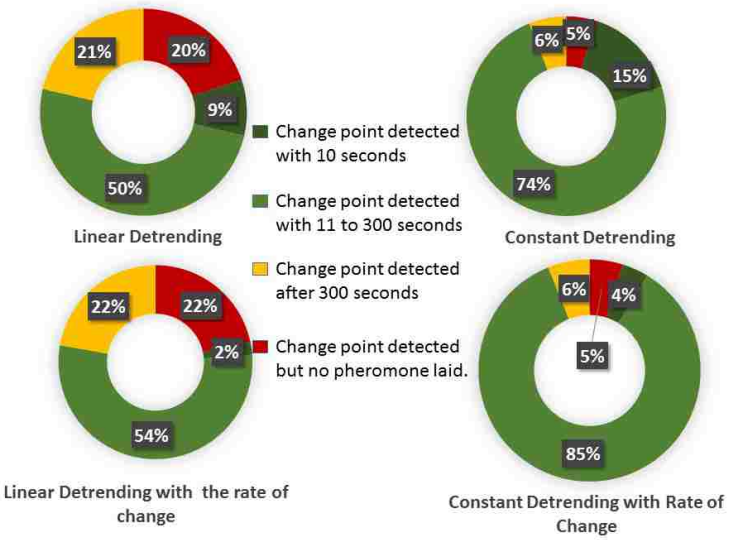


Figure 4.3: Efficiency chart for change point detection methods on a large pile of 256 seeds. This result is generated from the simulated data of pheromone only parameters, configured for *P. rugosus*

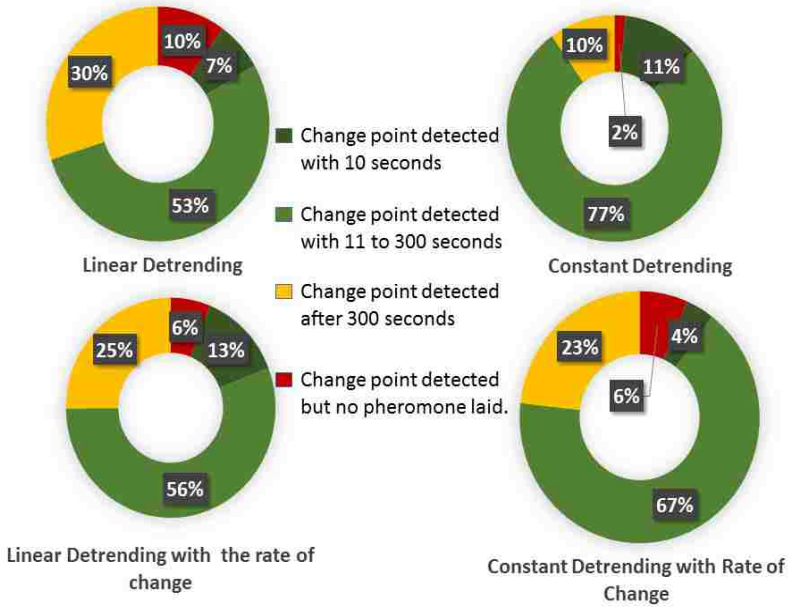


Figure 4.4: Efficiency chart for change point detection methods on four medium piles of 64 seeds. This result is generated from the simulated data of pheromone only parameters, configured for *P. rugosus*

Performance of *constant detrending method* and *constant detrending method with difference in rate* is similar, if we consider *categories* ≤ 10 and *catagory* $11-300$. But *constant detrending* does significantly better than *constant detrending with change of foraging rate* in detecting change points for four piles and sixteen piles in the simulated environment.

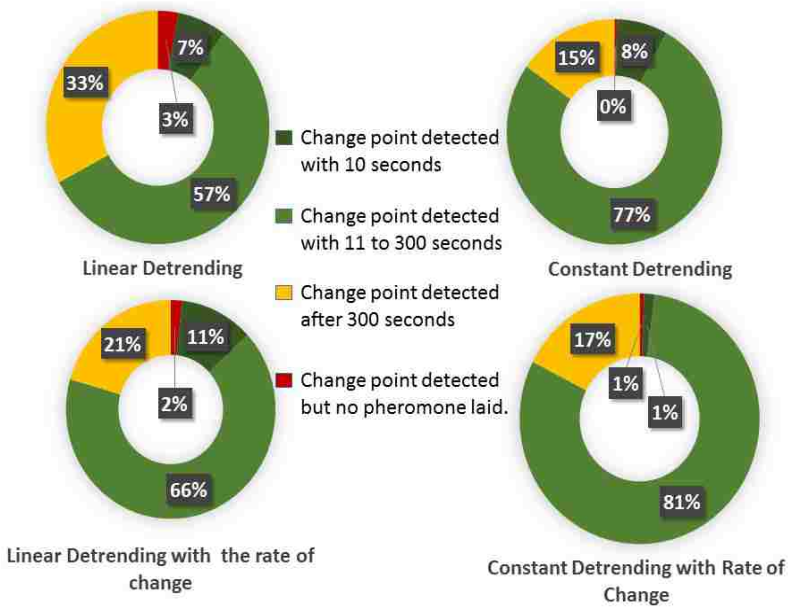


Figure 4.5: Efficiency chart for change point detection methods on sixteen small piles of 16 seeds. This result is generated from the simulated data of pheromone only parameters, configured for *P. rugosus*

From figure 4.4 and figure 4.5 we can observe that, the performance of *constant detrending* combined in *category* ≤ 10 and *catagory* $11-300$ is better than performance of *constant detrending with change in foraging rate* combined in these two categories. Also from figure 4.4 and 4.5 we can see that *constant detrending* does better in *category* > 300 than *constant detrending with change in foraging rate*.

If we look at the performance of the four methods over these categories in phermone only simulated environments, for *P. maricopa* and *P. desertorum* we observe

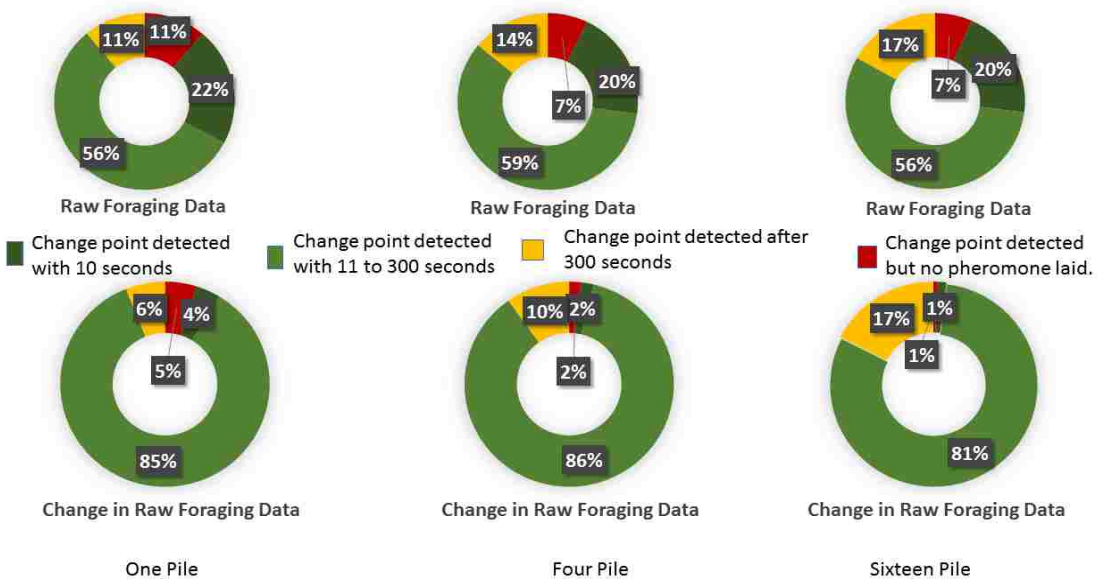


Figure 4.6: Efficiency chart for change point detection methods on foraging rate and change in foraging rate. This result is generated from the simulated data of pheromone only parameters, configured for *P. rugosus*

similar pattern of performances. The results for *P. maricopa* and *P. desertorum* are provided in the appendices.

Figure 4.6 shows efficiency of change point detection algorithm on *raw foraging rate* and *change in foraging rate* in pheromone only experiments. We observe that, the performance is better for *change in foraging rate*.

Figure 4.7 shows the comparison of four different methods on a simulated environment with both memory and communication for *P. rugosus*. From this figure it is clearly visible that *constant detrending* does better in detecting the change points than any other methods. As a matter of fact, *constant detrending* does better in memory plus communication environment than the communication only environment.

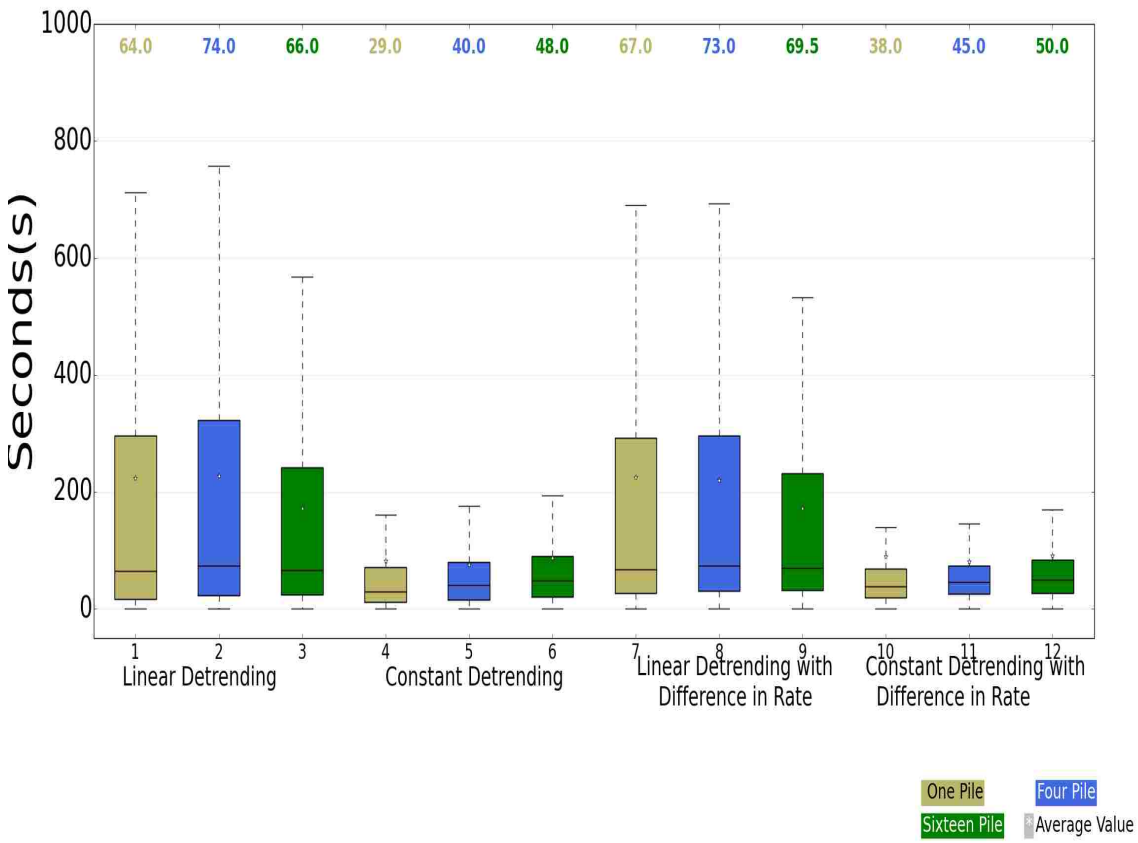


Figure 4.7: Comparison of change point detection methods without outliers for *pheromone plus sitefidelity* parameters . The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

We have analyzed the efficiencies of all four methods for communication plus memory environmental settings. We observe that *constant detrending* and *constant detrending with change in foraging rate* does significantly better than the other two methods. But in *category* > 300 and catagory None, *constant detrending* does better than *constant detrending with change in rate*. We observe that the error is 0% for *constant detrending*, whereas it is 1% for *constant detrending with change in foraging rate*. *Constant detrending* also surpasses *constant detrending with change in foraging*

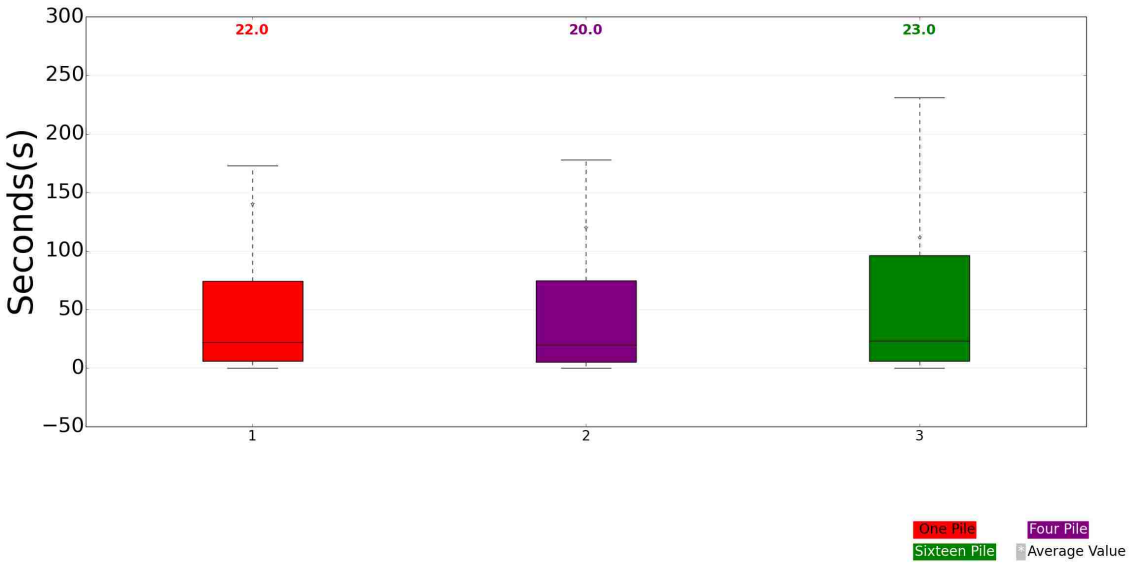


Figure 4.8: Enlarged view of efficiency of change point detection algorithm on *raw foraging rate* on simulation data of 12 ants. The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate. Compared to figure 4.7 change point detection on raw data detects changes fastest.

rate in the combined result of *category* ≤ 10 and *category* $11 - 300$. We observe, the similar pattern of performances for four piles and sixteen piles.

Figure 4.8 shows the efficiency of change point detection algorithm on foraging rate. We observe, time to detect the change points is better than *constant detrending*.

Figure *4.9(a)* is the efficiency chart for *foraging rate with no detrending* for one large pile on pheromone plus site fidelity environment. The result of other piles for *P. rugosus* have similar pattern. We have mentioned those in appendices. In figure *4.9(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

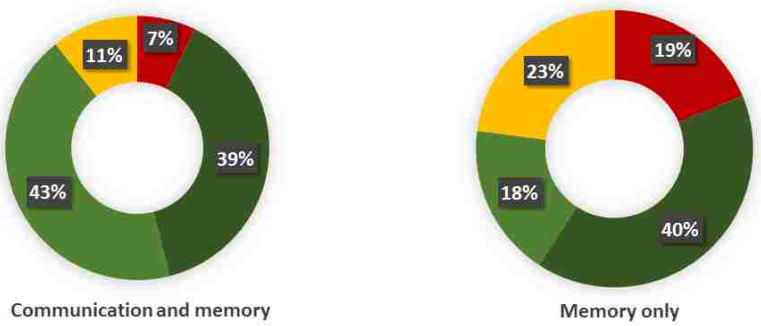


Figure 4.9: Figure on the left is the efficiency chart of the change point detection algorithm on *raw foraging rate with no detrending* for one large pile of 256 seeds. The data is generated from the simulation of pheromone plus site fidelity parameters of *P. rugosus*. Figure on the right is the efficiency chart of change point detection algorithm on *raw foraging rate with no detrending* for one large pile of 256 seeds. The figure is generated by analyzing the simulated data of sitefidelity only parameters 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* also method detects change points in site fidelity only environment. 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 4.9(b), we have observed that 58% 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.

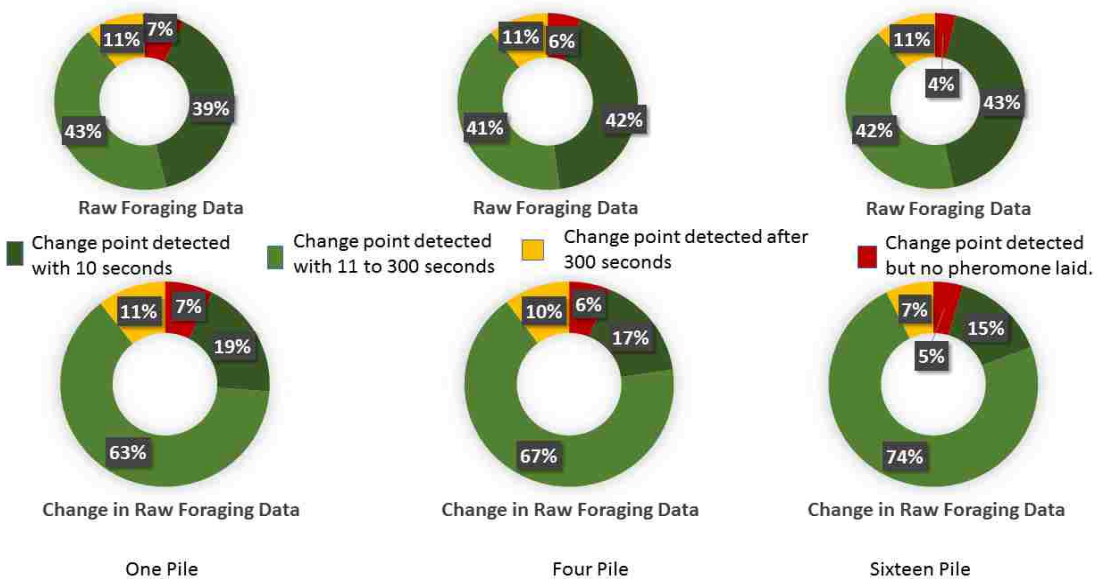


Figure 4.10: Efficiency of change point detection algorithm for detecting pheromone using *raw foraging rate* and *change in raw foraging rate*, in memory and communication parameters.

To validate which method is better in detecting only pheromone but no site-fidelity event, we have also calculated the time difference between site fidelity and followed by a change point detection event for memory plus communication environment. From figure 4.10 we observe that, only for few experiments it detects change points for the use of memory. This data is generated by applying the change point detection algorithm on *foraging rate* and *change in foraging rate*. We observe similar pattern for *constant detrending* method. Although we miss some of the change points when we use the *foraging rate with no detrending* or *change in foraging rate with no detrending*, it is better because we do not detect much of the site fidelity events in this method.

We have varied the number of ants to observe the change in the foraging rate in the simulation. We have performed the simulation with 96 ants and 48 ants. We

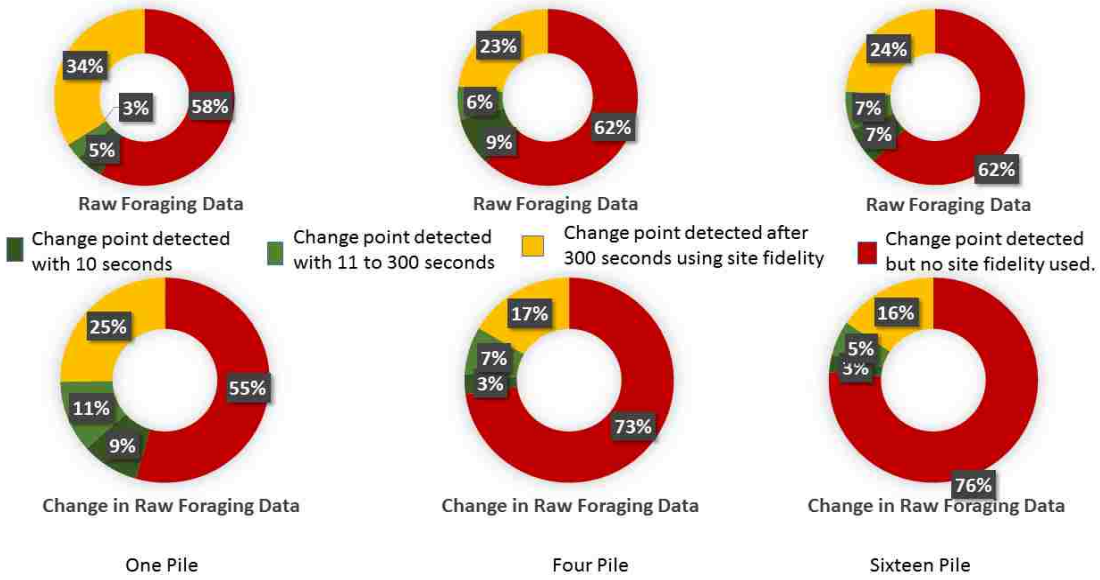


Figure 4.11: Efficiency of change point detection algorithm on raw data for detecting site fidelity in memory and communication parameters.

have observed that in the simulation with 96 ants, they collect more random seeds than the seeds that are clumped. A reason for this is when the number of ants is too large, the rate of collision increases between them while they try to forage from a clumped food source. The performance is little better when we ran the experiment with 48 ants. They tend to collect seeds from the clumped food distribution.

We have applied the change point detection algorithm on the foraging rate of these two types of simulations, keeping the parameters for change point detection algorithm same. We observe that the average time difference between a pheromone laying event followed by a change point detection algorithm increases for 48 ants. And it is increased more for 96 ants.

We ran the simulation for 96 and 48 ants with site fidelity only parameters, we observe that they don't use the site fidelity often to recruit. Even though they use, it does not effect in their foraging rate. Although we have detected a few change points

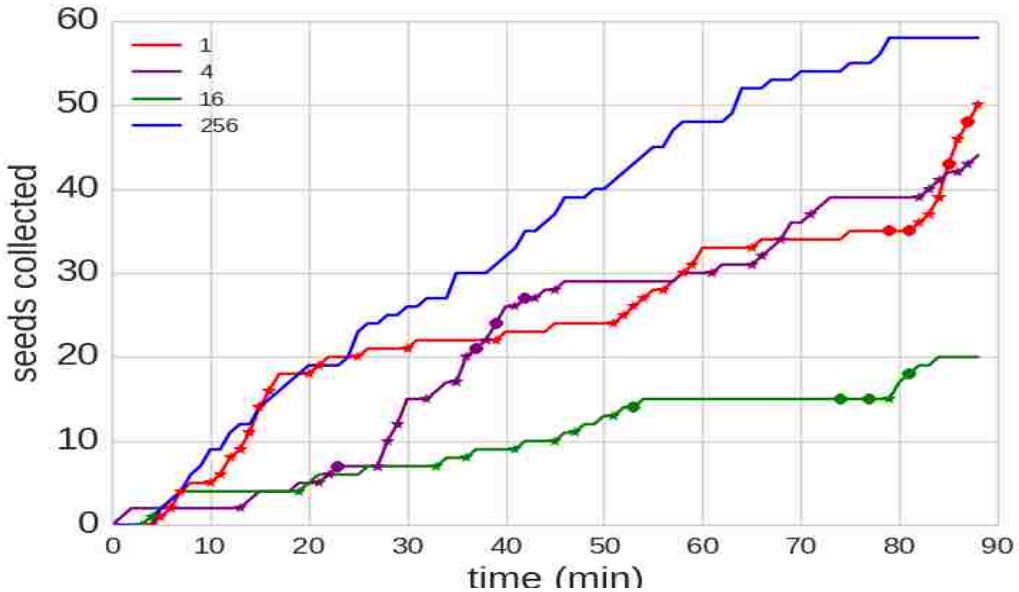


Figure 4.12: A plot of seed collection from simulation of 12 simulated ants. The stars in the foraging data represent the pheromone laying event and circles represent detection of change points using the binary segmented cumulative sum method on raw data.

in their foraging rate, the accuracy of the algorithm is very low in the simulation for size fidelity only parameters. The results are included in the appendices.

The change point detection algorithm on raw foraging data detects the pheromone laying event fastest in pheromone only experiments. It also detects pheromone in memory and communication only environment. However, It rarely detects change points for sitefidelity in memory and sitefidelity experiments. It also detects change points in memory only experiments because sitefideity is used so often in these experiments.

Because it is the best at detecting changepoints, we use the binary segmented cumsum change point detection algorithm on raw data to detect recruitment in the field data.

4.2 Results from Field Data

We applied the change point detection algorithm on the change of foraging rate in field data. The parameters are kept same as simulations for the change point detection method. We are able to detect change points in 10 experiments out of 11 for *P. desertorum*. For *P. maricopa*, we detected change points in 10 experiments out of 11 and for *P. rugosus* we are able to detect change points in 11 experiments out of 13. Table 4.2 shows the detail of the results for field data.

P. desertorum			
Experiment	One	Four	Sixteen
DP15_070709	4	2	2
DP10_0606	3	3	2
DP14_0606	2	2	2
DP15_0604	2	0	2
DP15_0617	1	2	2
DPC_070109	2	1	2
DP15_070809	3	2	2
DP16_0610	2	2	2
DPB_070809	2	2	2
DP9_0603	0	3	2

P. maricopa			
Experiment	One	Four	Sixteen
MP8_0609	3	3	4
MP9_0528	2	2	1
MP9_0602	1	2	2
MP9_0611	2	2	1
MP9_0620	2	2	2
MPE_24Jul09	2	2	2
MP4_0527	2	2	2
MP7_0620	1	2	2
MP8_0527	1	1	1
MPX_0624	2	2	3

P. rugosus			
Experiment	One	Four	Sixteen
RP6_0609	2	2	2
RP7_0610	3	2	1
RP10_0527	2	2	2
RP10_0528	1	2	2
RP10_0604	2	0	1
RP12_0602	2	2	2
RP14_0610	2	1	2
RP16_0612	2	2	2
RP19_0618	3	2	2
RS23_0821	2	2	2
RP17_0617	1	3	2

Table 4.1: 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 10 experiments out of 11. For *P. maricopa* it is 10 out of 11. For *P. rugosus* it is 11 out of 13. The numbers conclude how many times change points are detected for a pile type in each experiment.

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. In the field experiments we were also able to detect change points for most of the experiments. This suggests all three species are either using pheromones for all pile sizes or they exclusively use

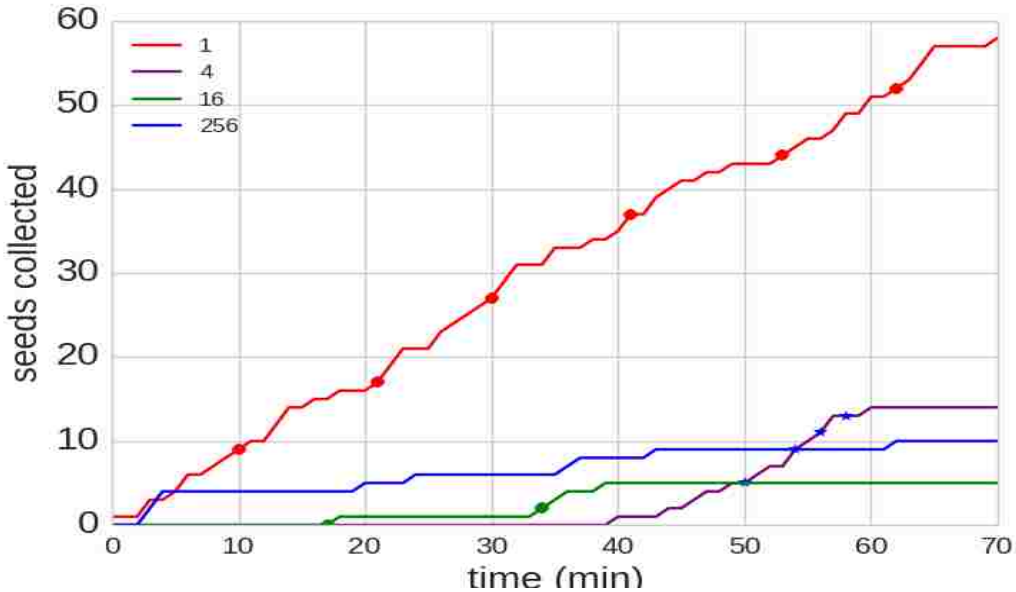


Figure 4.13: A plot of seed collection from one of the field experiment of *P. desertorum* with change points indicate with circles.

sitefidelity, but use it in large amounts that generate change points.

Chapter 5

Conclusion

An enormous amount of research has been performed to analyze the foraging strategies of ants. Scientists performed many biological experiments to discover what strategies ants follow to forage from the environment.

Although we have talked about the combination of three strategies for foraging, it was unclear which of the strategies ants follow for recruiting from large food sources. Several indoor experiments on harvester ants which used high definition cameras and chemical detectors have shown that they lay pheromone trails for other ants to follow.

As the amount of pheromone laid by the ants are very small, it was difficult to identify the existence of chemicals in the outdoor field experiments. So, we used this mathematical approach to detect the pheromone laying events of ants in the outdoor field experiments.

In simulations, we could detect change points in every single experiment. Since we can detect change points in the majority of the field experiments, we infer that all three species of ants either use pheromone to recruit from large clumped food

sources or exclusively uses sitefidelity which generates change points.

In *memory only* simulation environment ants used sitefidelity extensively, which resulted in the drastic change in their foraging rate. As a result, we can see picks and hikes in their foraging rate which was detected by our change point detection algorithm. But when we have tuned the simulations to use both memory and communication, we observe ants did not use memory much, even if they use, it didnt affect their foraging rate, so it is only detected for pheromone.

We have only used binary segmented cumulative sum methods on raw data for detecting the change points. In future, we can try to observe and find the pattern of changes in the foraging rate when the pheromone is laid, and use the statistical data from the simulation to infer the pheromone laying events in the field experiment data.

Chapter 6

Appendicies

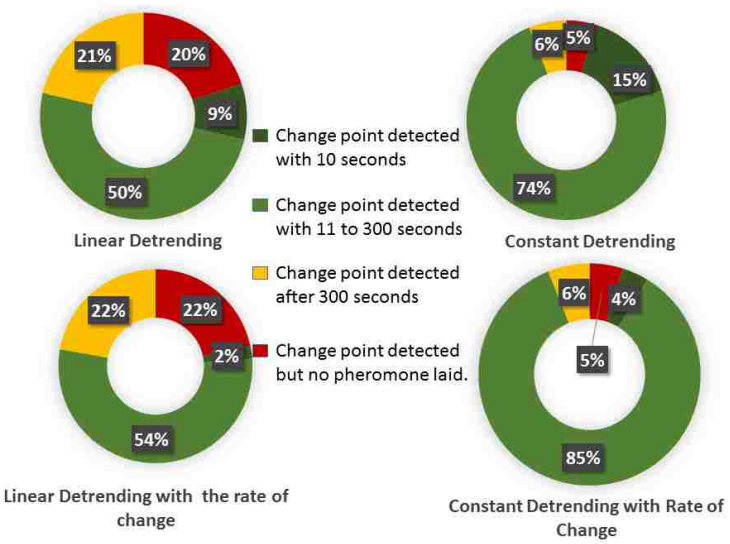


Figure 6.1: Efficiency chart for *P. rugosus*, one pile, communication only setting.

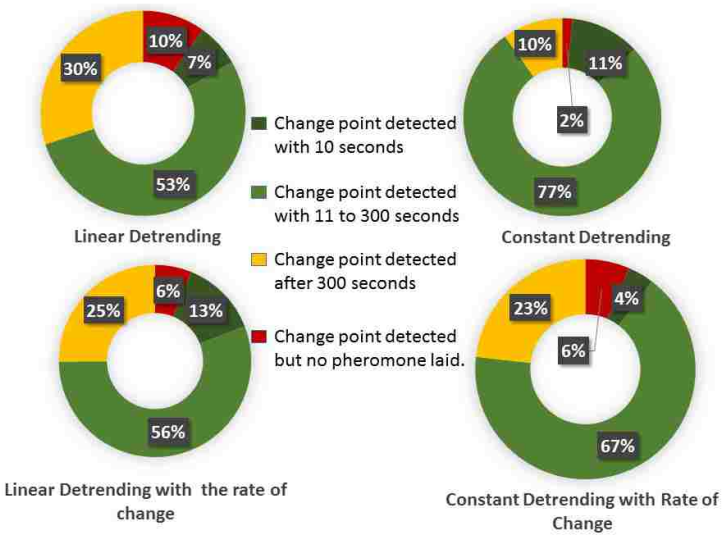


Figure 6.2: Efficiency chart for *P. rugosus*, four pile, communication only setting.

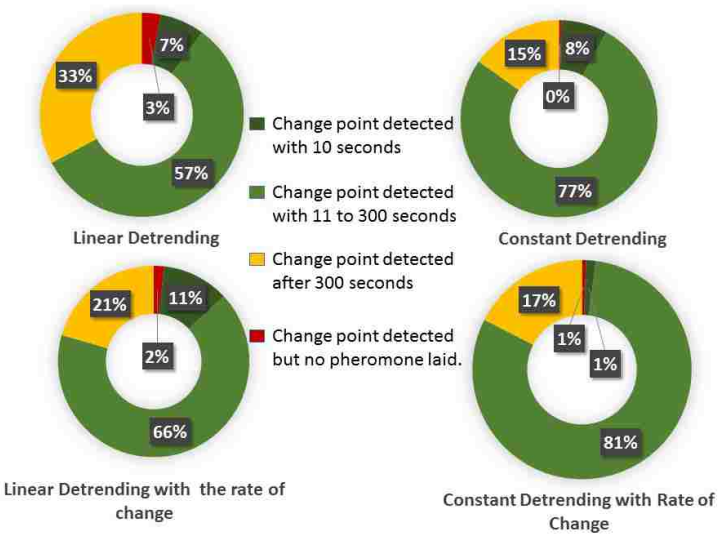


Figure 6.3: Efficiency chart for *P. rugosus*, sixteen pile, communication only setting.

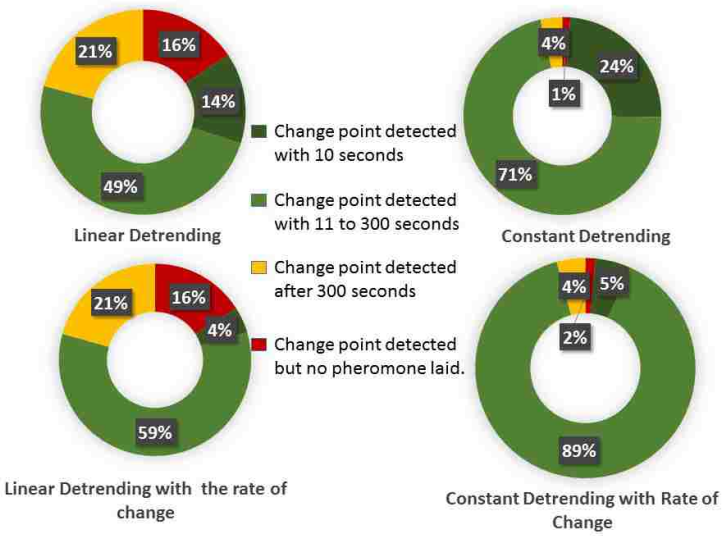


Figure 6.4: Efficiency chart for *P. rugosus*, one pile, memory and communication combined setting.

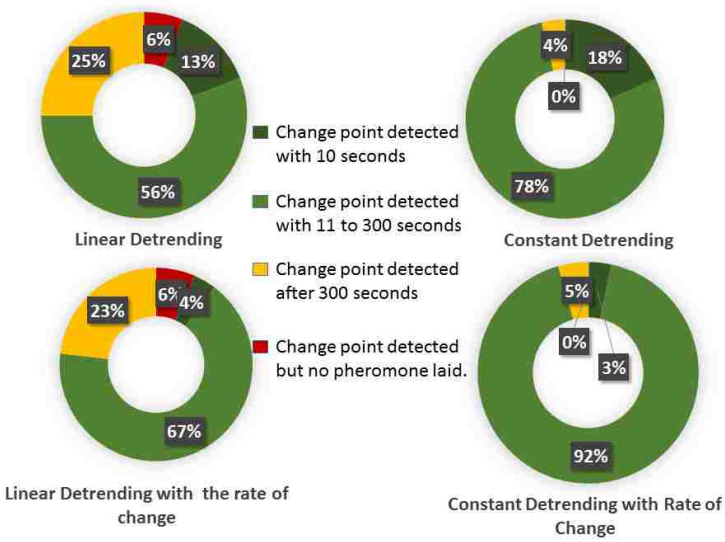


Figure 6.5: Efficiency chart for *P. rugosus*, four pile, memory and communication combined setting.

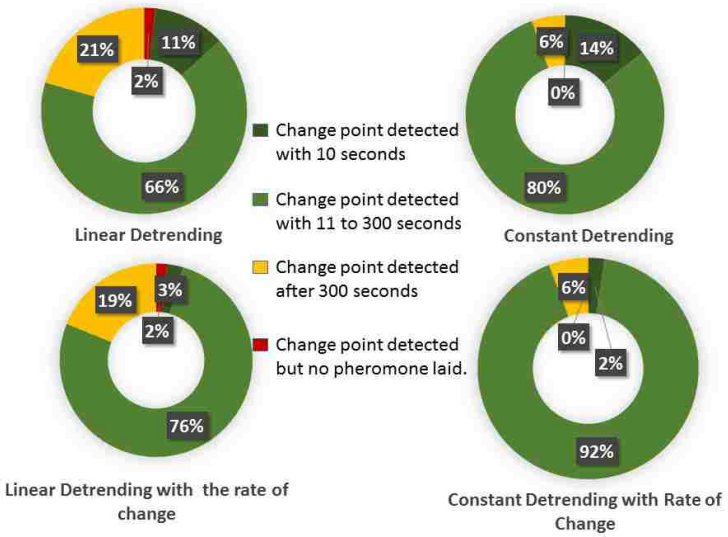


Figure 6.6: Efficiency chart for *P. rugosus*, sixteen pile, memory and communication combined setting.

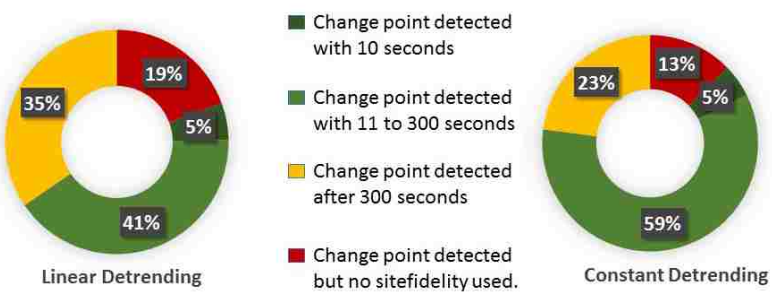


Figure 6.7: Efficiency chart for *P. rugosus*, one pile, memory only setting.

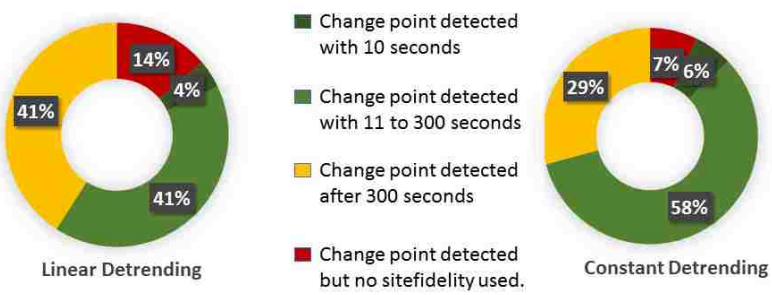


Figure 6.8: Efficiency chart for *P. rugosus*, four pile, memory only setting.

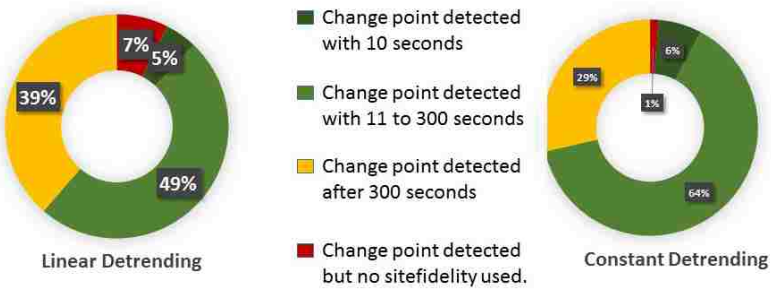


Figure 6.9: Efficiency chart for *P. rugosus*, sixteen pile, memory only setting.

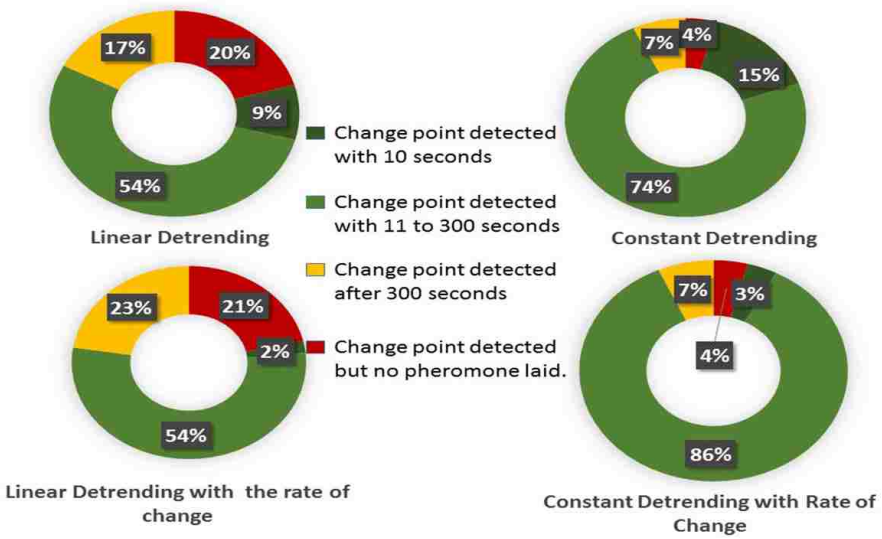


Figure 6.10: Efficiency chart for *P. desertorum*, one pile, communication only setting.

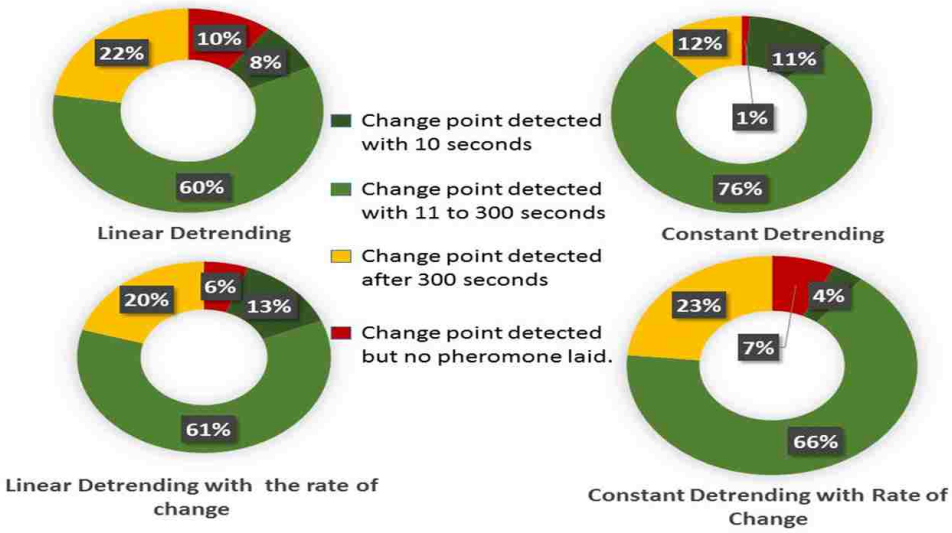


Figure 6.11: Efficiency chart for *P. desertorum*, four pile, communication only setting.

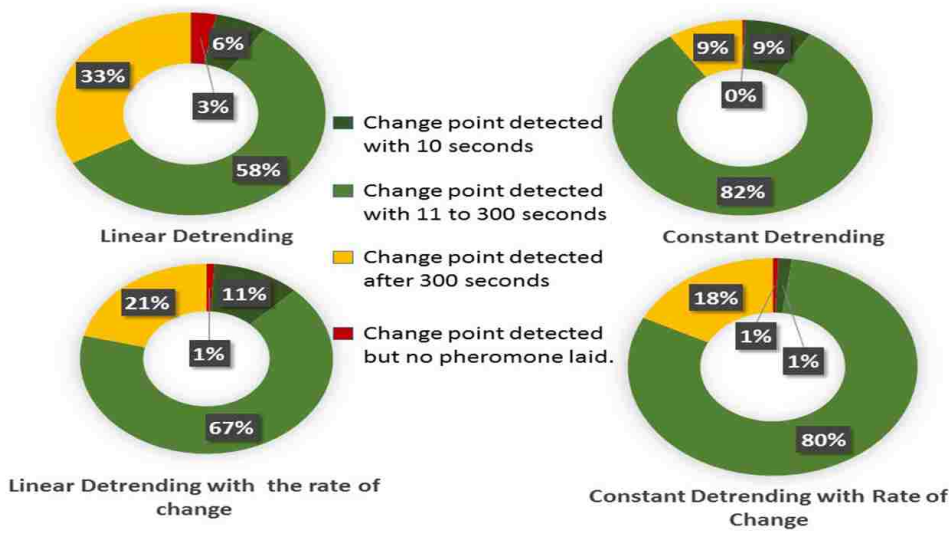


Figure 6.12: Efficiency chart for *P. desertorum*, sixteen pile, communication only setting.

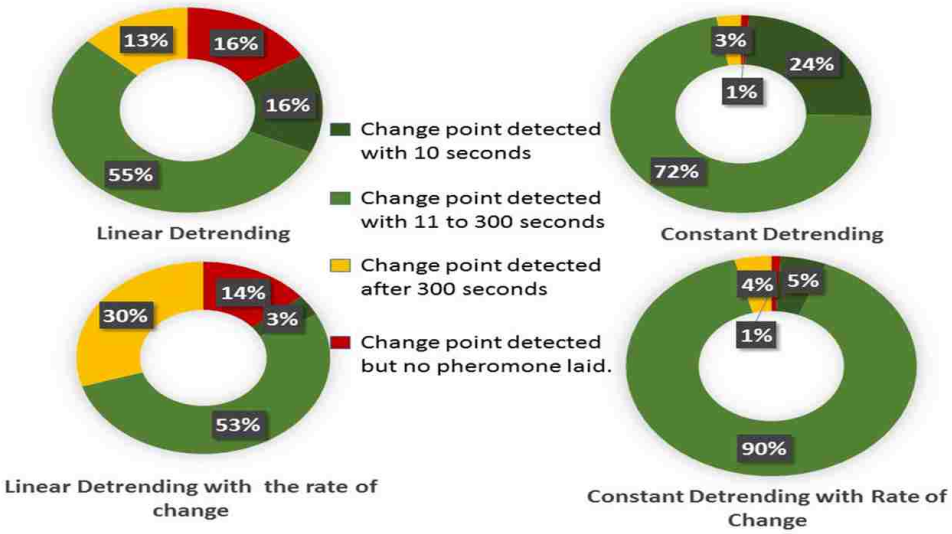


Figure 6.13: Efficiency chart for *P. desertorum*, one pile, memory and communication combined setting.

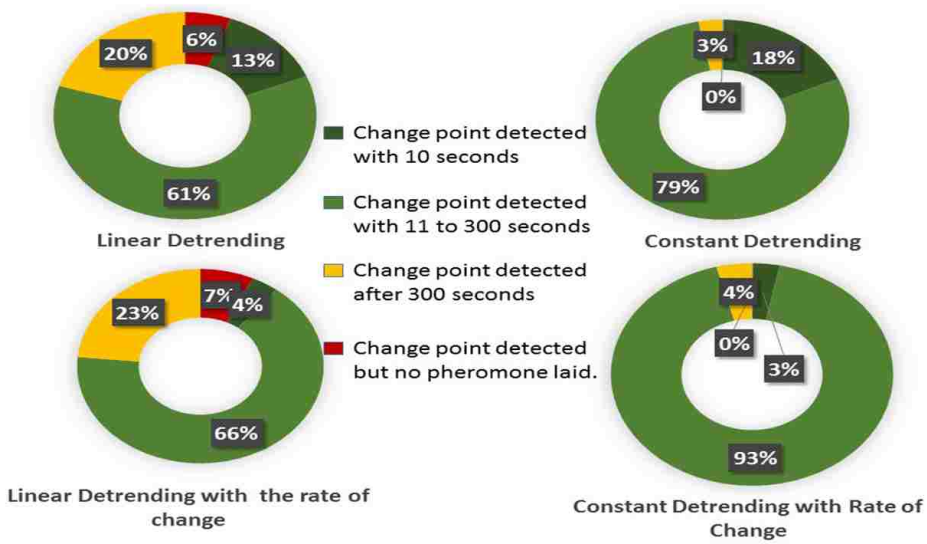


Figure 6.14: Efficiency chart for *P. desertorum*, four pile, memory and communication combined setting.

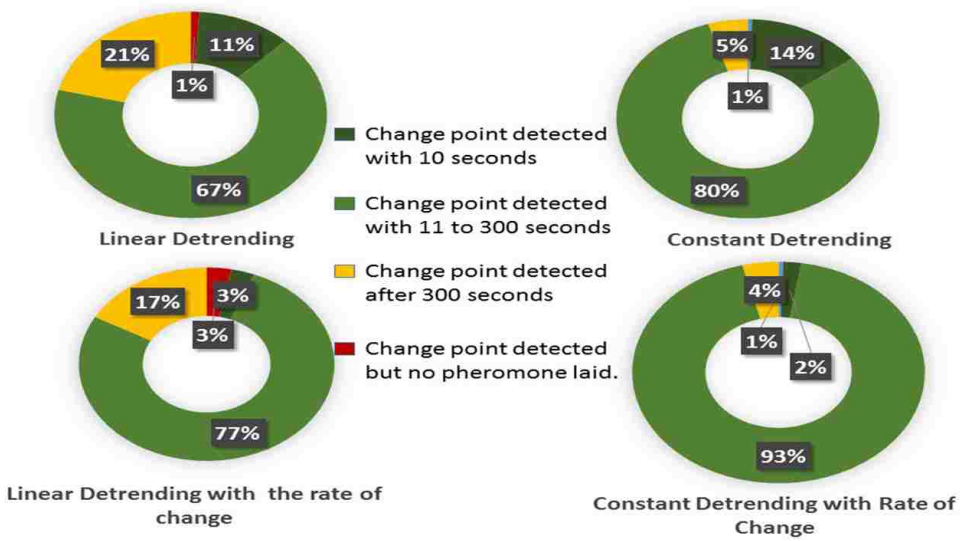


Figure 6.15: Efficiency chart for *P. desertorum*, sixteen pile, memory and communication combined setting.

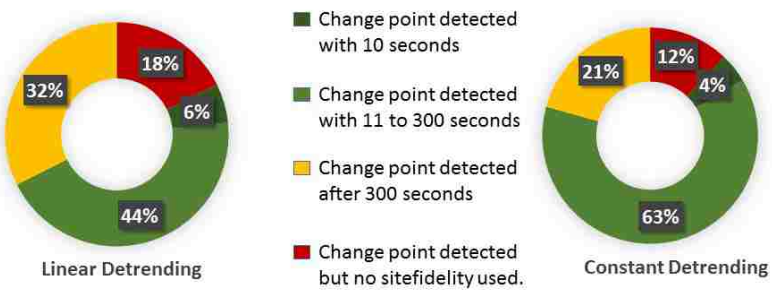


Figure 6.16: Efficiency chart for *P. desertorum*, one pile, memory only setting.

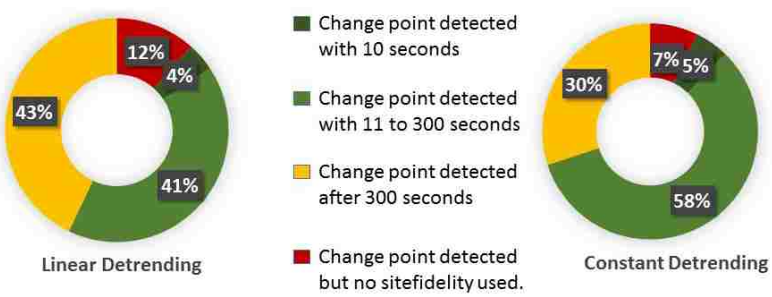


Figure 6.17: Efficiency chart for *P. desertorum*, four pile, memory only setting.

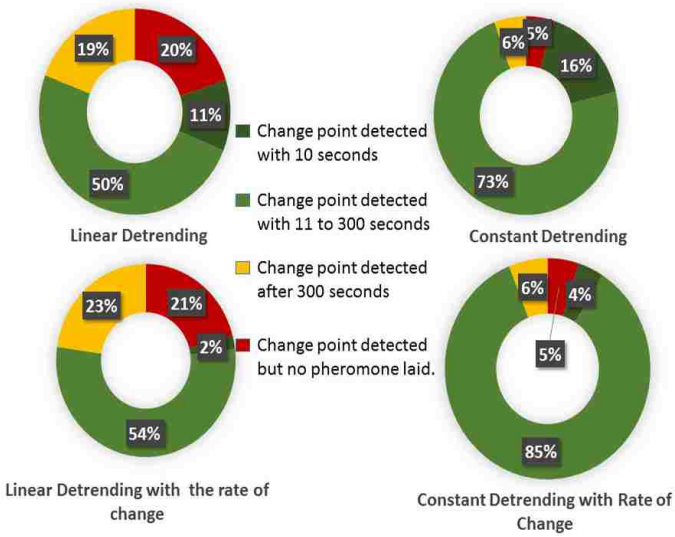


Figure 6.18: Efficiency chart for *P. maricopa*, one pile, communication only setting.

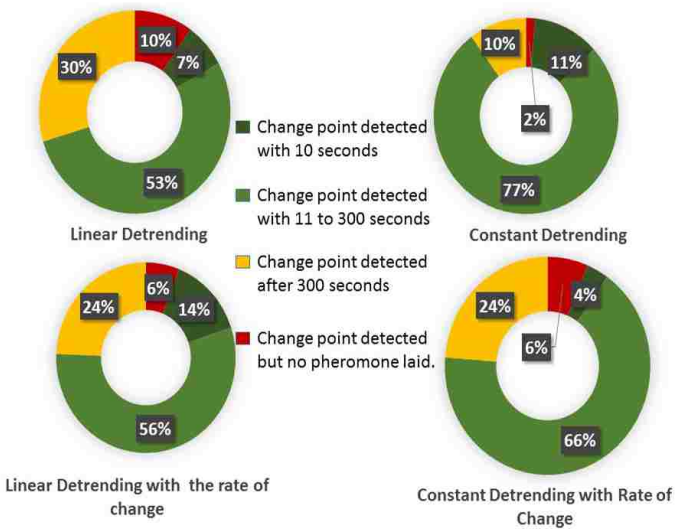


Figure 6.19: Efficiency chart for *P. maricopa*, four pile, communication only setting.

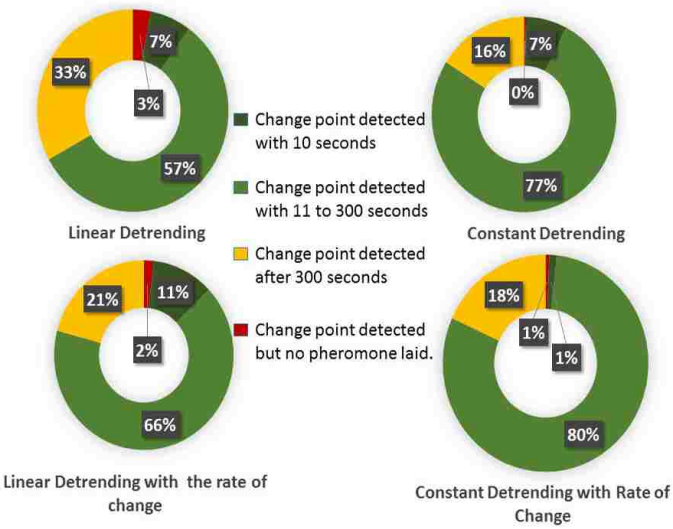


Figure 6.20: Efficiency chart for *P. maricopa*, sixteen pile, communication only setting.

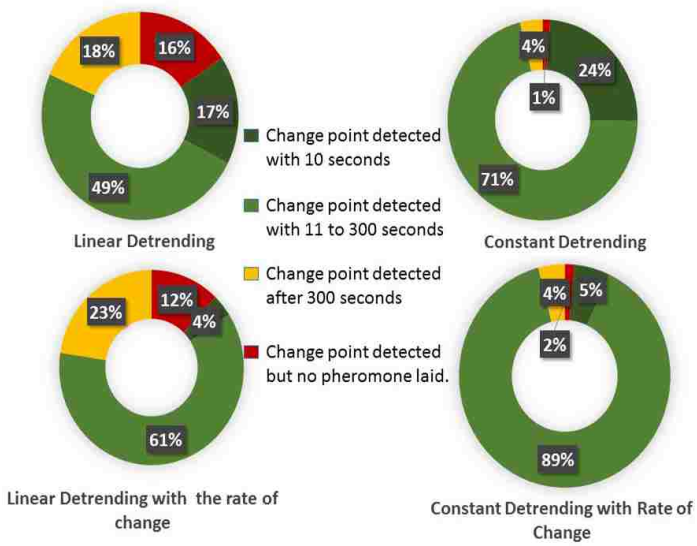


Figure 6.21: Efficiency chart for *P. maricopa*, one pile, memory and communication combined setting.

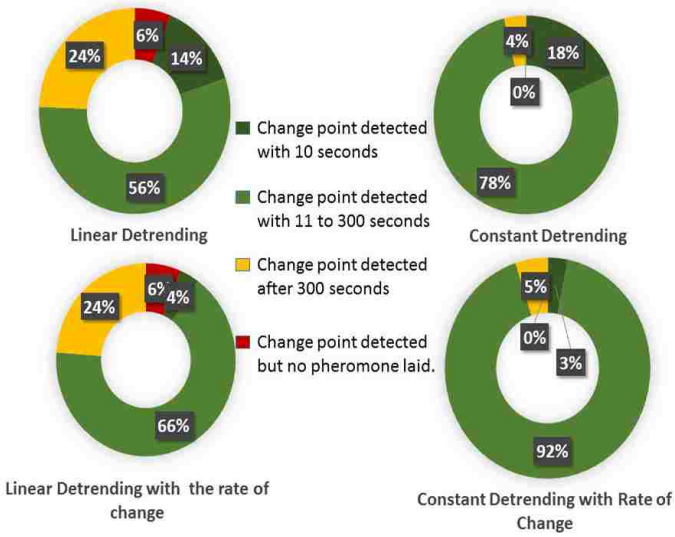


Figure 6.22: Efficiency chart for *P. maricopa*, four pile, memory and communication combined setting.

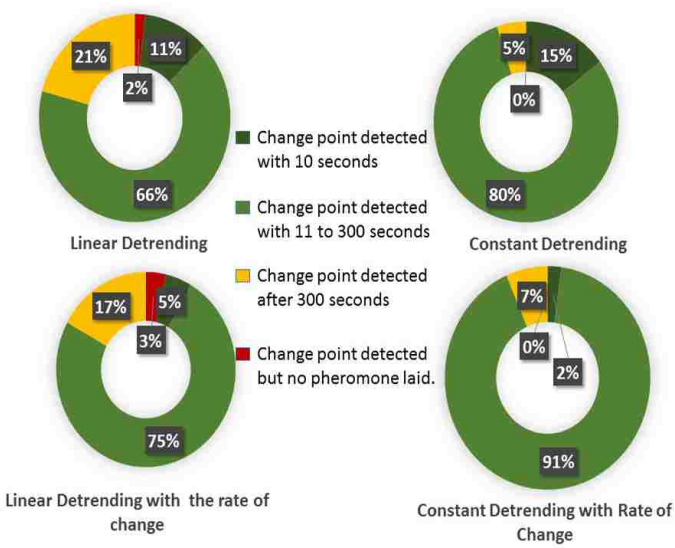


Figure 6.23: Efficiency chart for *P. maricopa*, sixteen pile, memory and communication combined setting.

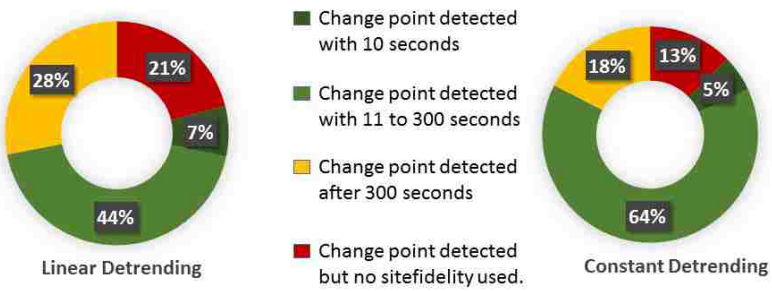


Figure 6.24: Efficiency chart for *P. maricopa*, one pile, memory only setting.

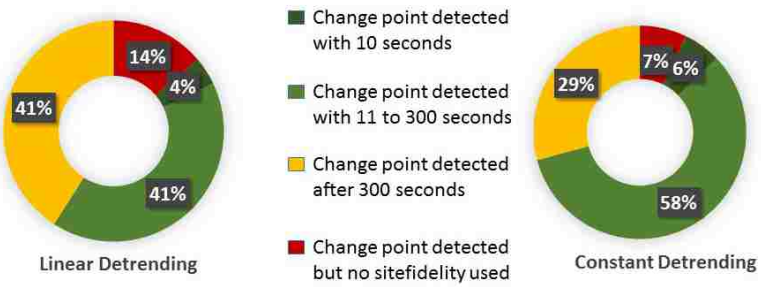


Figure 6.25: Efficiency chart for *P. maricopa*, four pile, memory only setting.

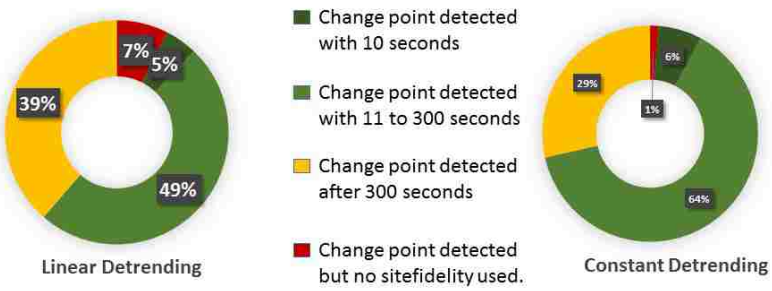


Figure 6.26: Efficiency chart for *P. maricopa*, sixteen pile, memory only setting.

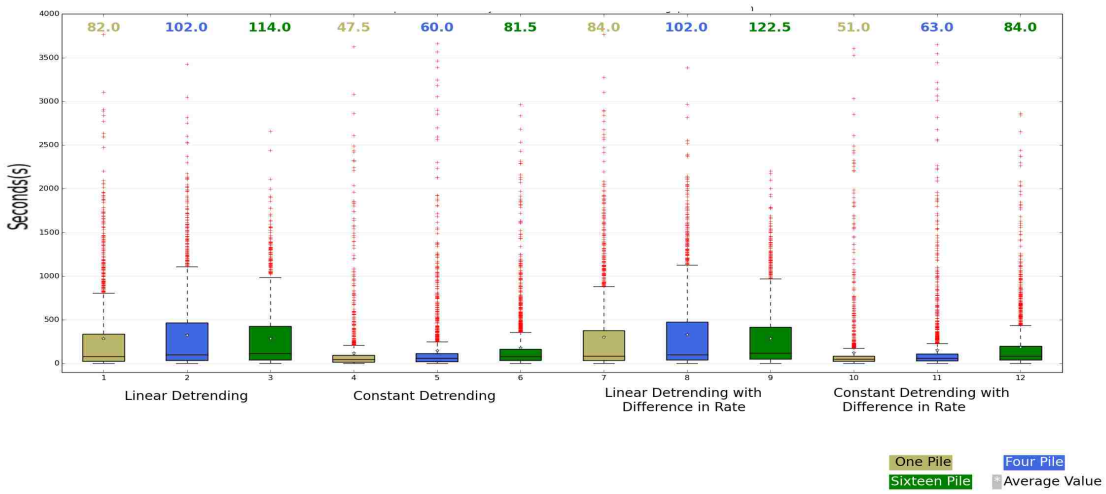


Figure 6.27: Comparison of change point detection methods for pheromone only parameters for *P. rugosus*. The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

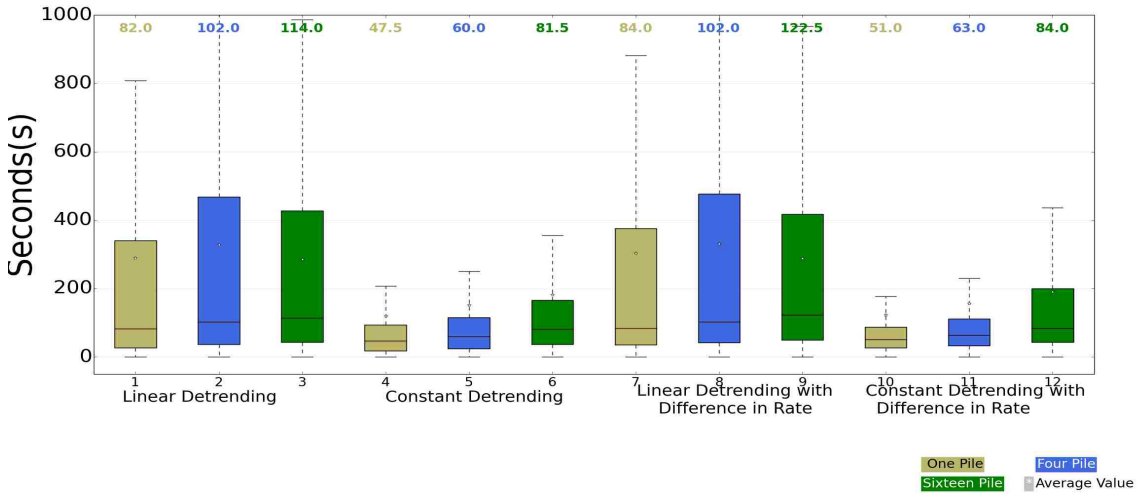


Figure 6.28: Comparison of change point detection method without outliers for pheromone only parameters *P. rugosus*.The time in seconds represents the difference between a pheromone laying event followed by a detection of change point in the foraging rate.

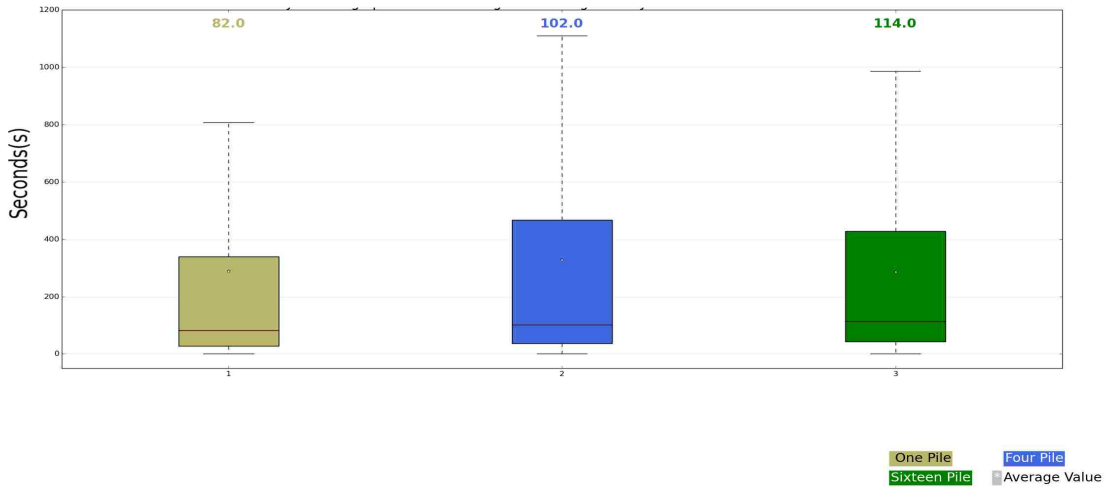


Figure 6.29: Efficiency of Linear Detrending method on data of *P. rugosus* for pheromone only settings

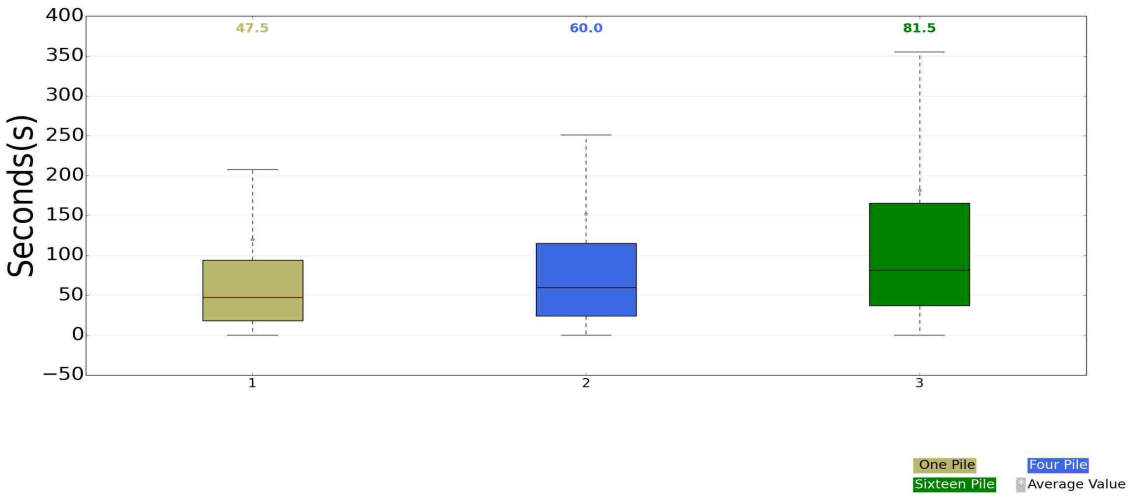


Figure 6.30: Efficiency of Constant Detrending method on data of *P. rugosus* for pheromone only Settings.

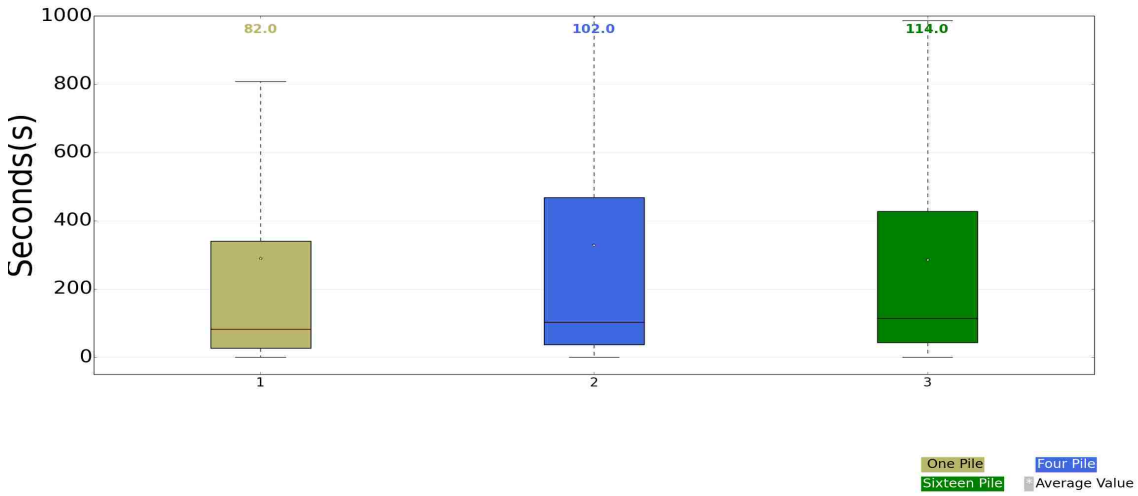


Figure 6.31: Efficiency of Linear Detrending with rate of change method on data of *P. rugosus* for pheromone only settings.

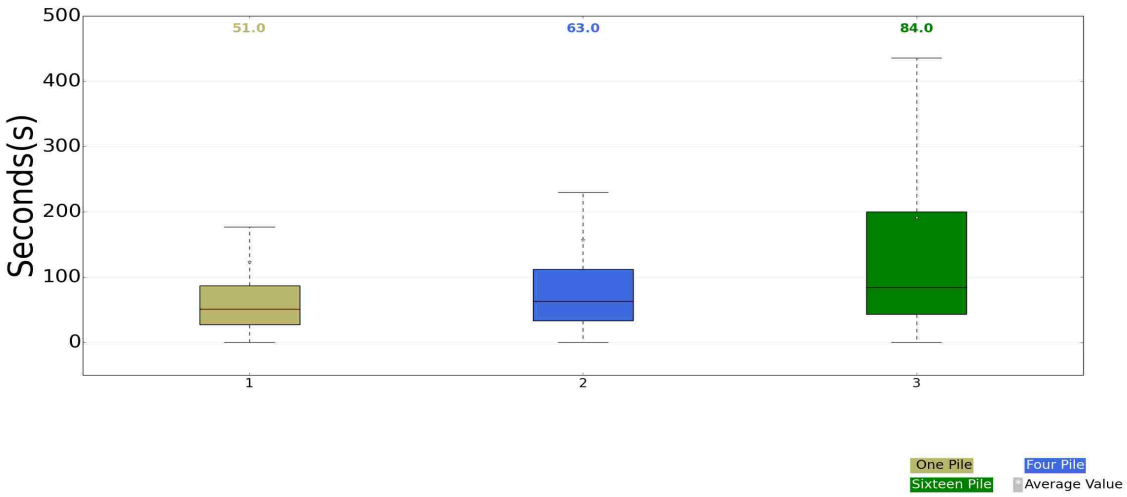


Figure 6.32: Efficiency of Linear Detrending with rate of change method on data of *P. rugosus* for pheromone only settings

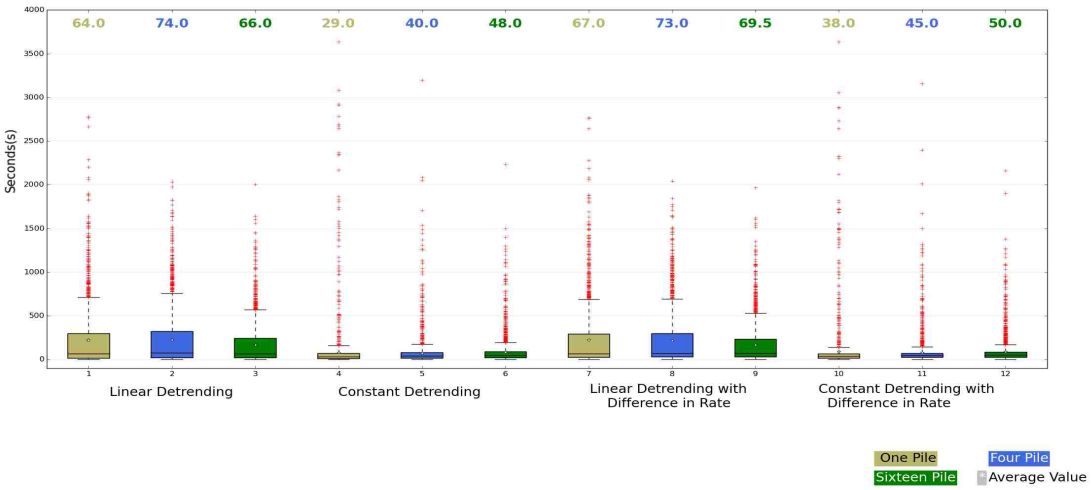


Figure 6.33: Comparison of all methods on data of *P. rugosus* for both combination of communication and memory

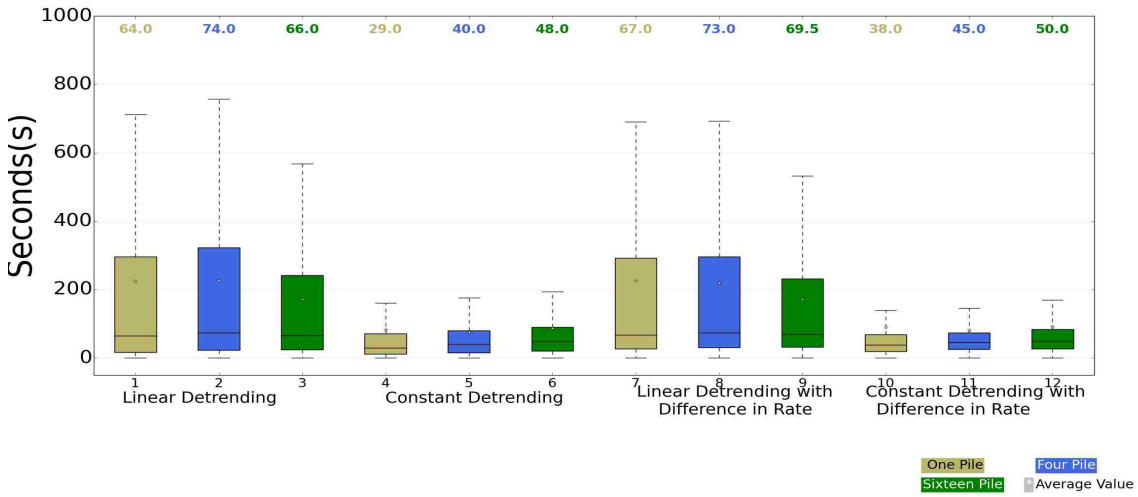


Figure 6.34: Comparison of all methods without outliers on data of *P. rugosus* for both combination of communication and memory

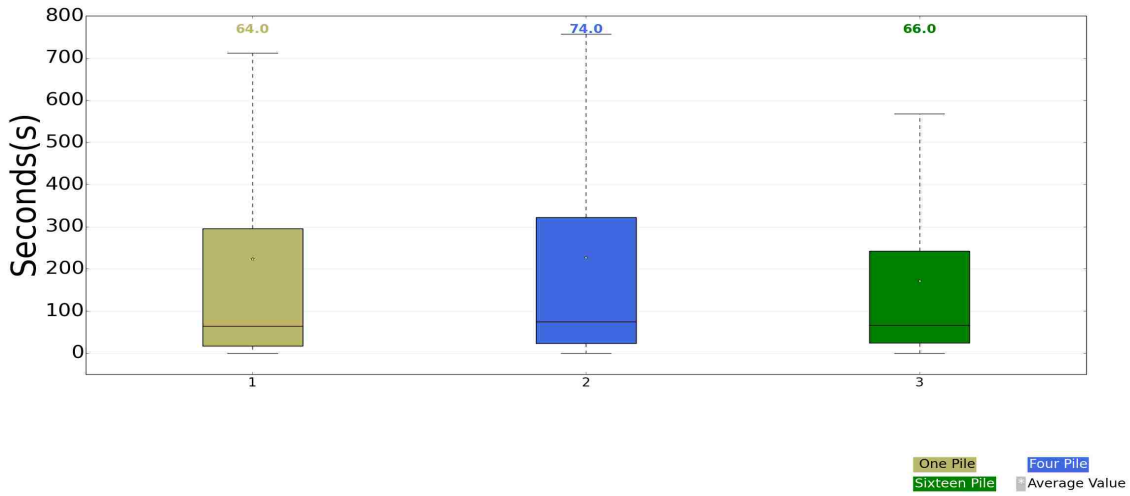


Figure 6.35: Efficiency of linear detrending on data of *P. rugosus* for combination of communication and sitefidelity

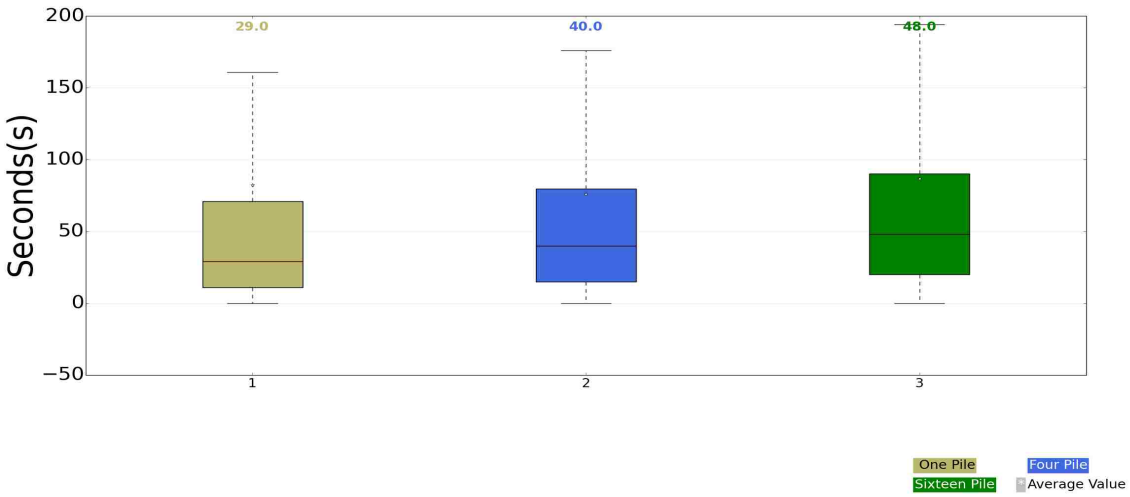


Figure 6.36: Efficiency of constant detrending methods on data of *P. rugosus* for both combination of communication and memory

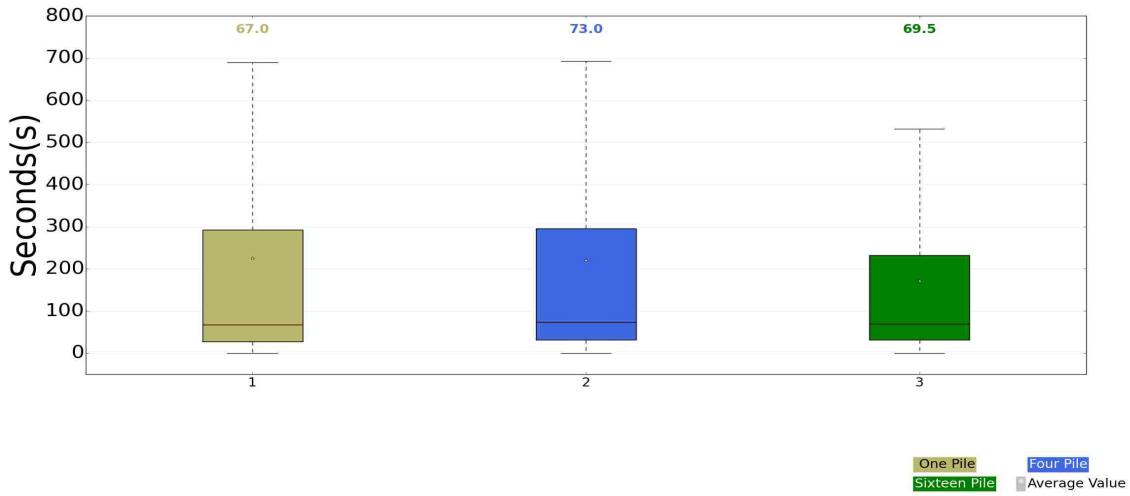


Figure 6.37: Efficiency of linear detrending method with rate of change in foraging rate on data of *P. rugosus* for both combination of communication and memory

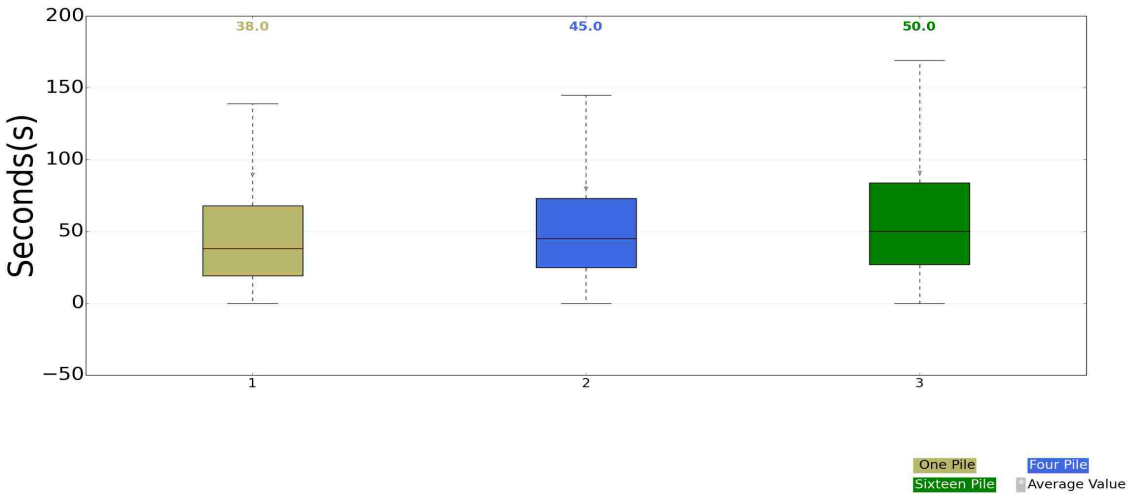


Figure 6.38: Efficiency of constant detrending method with rate of change in foraging rate on data of *P. rugosus* for both combination of communication and memory

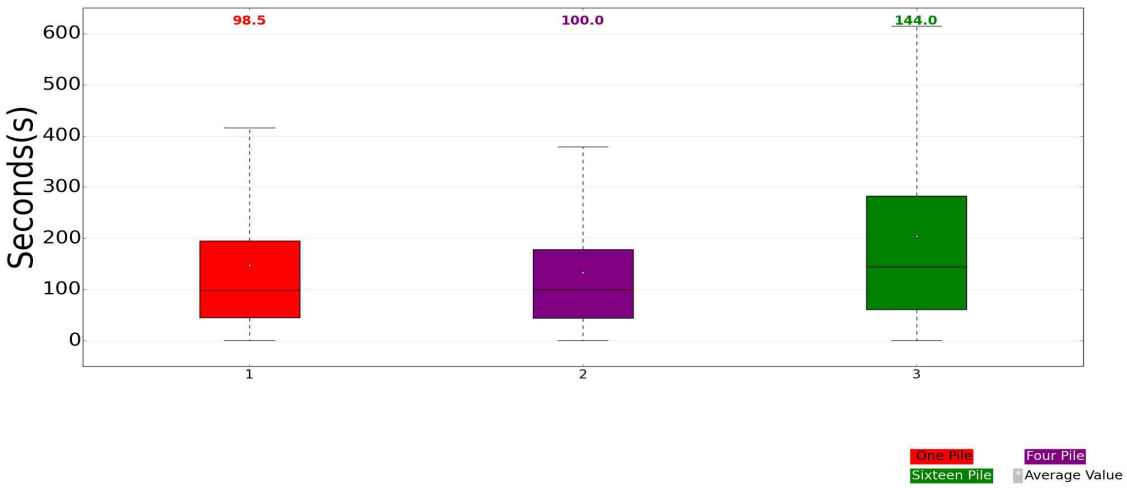


Figure 6.39: Efficiency of change point detection algorithm on foraging rate for 96 ants for combination of communication and memory

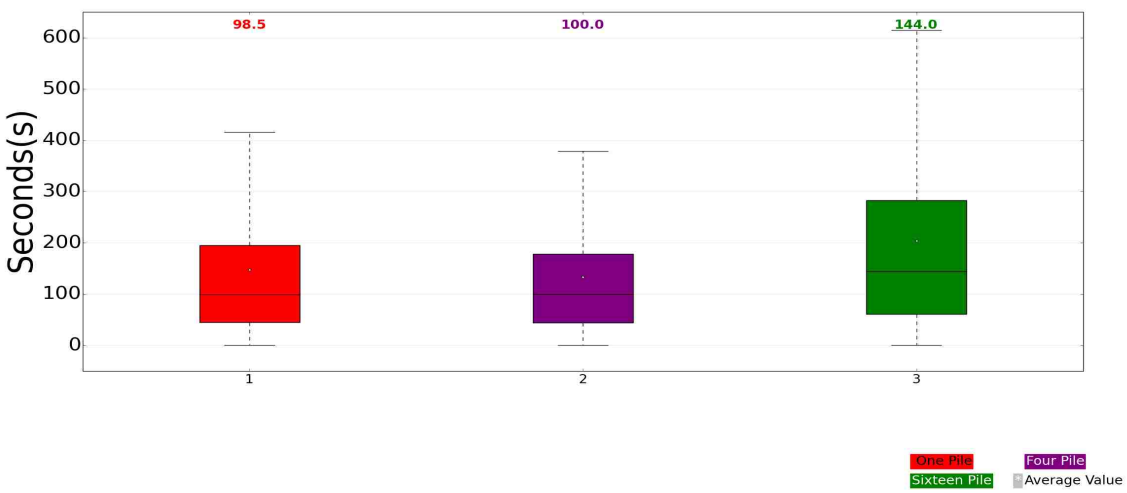


Figure 6.40: Efficiency of change point detection algorithm on change in foraging rate for 96 ants for combination of communication and memory

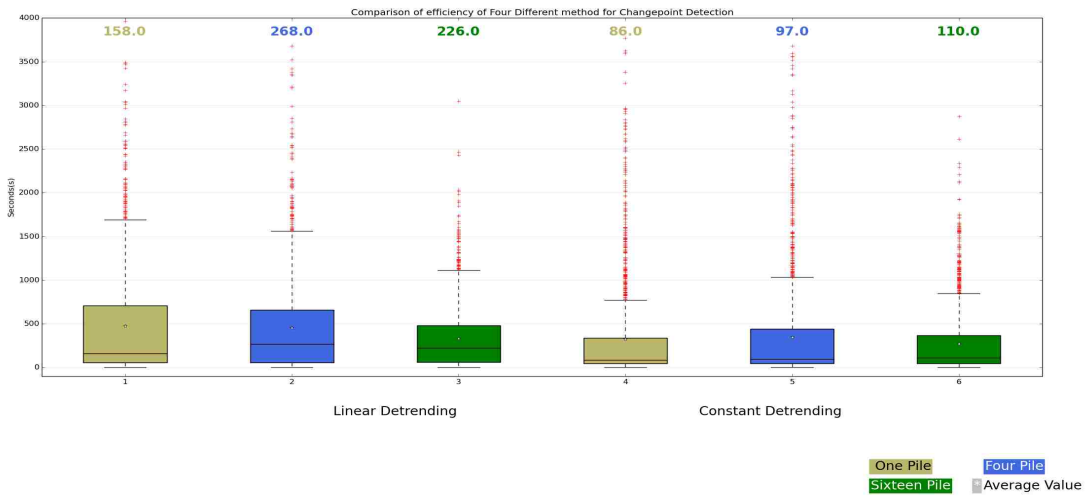


Figure 6.41: Comparison of all methods on data of *P. rugosus* for memory only parameters

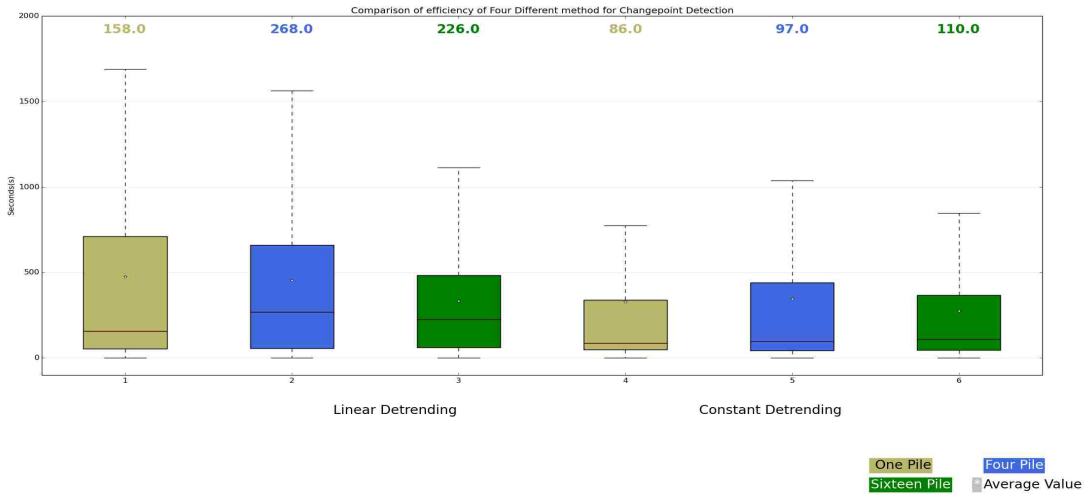


Figure 6.42: Comparison of all methods without outliers on data of *P. rugosus* for memory only parameters

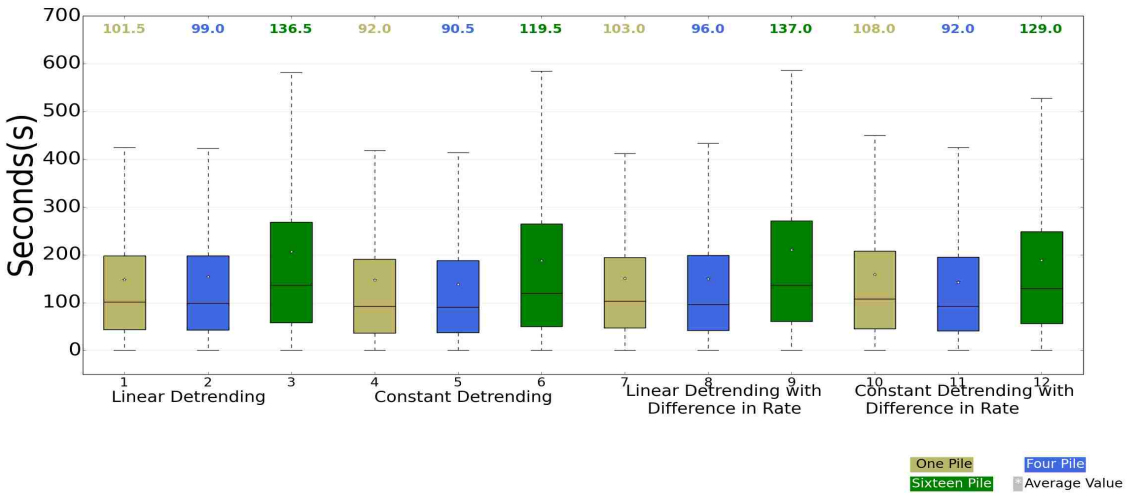


Figure 6.43: Efficiency of all methods with 96 ants for both combination of communication and memory

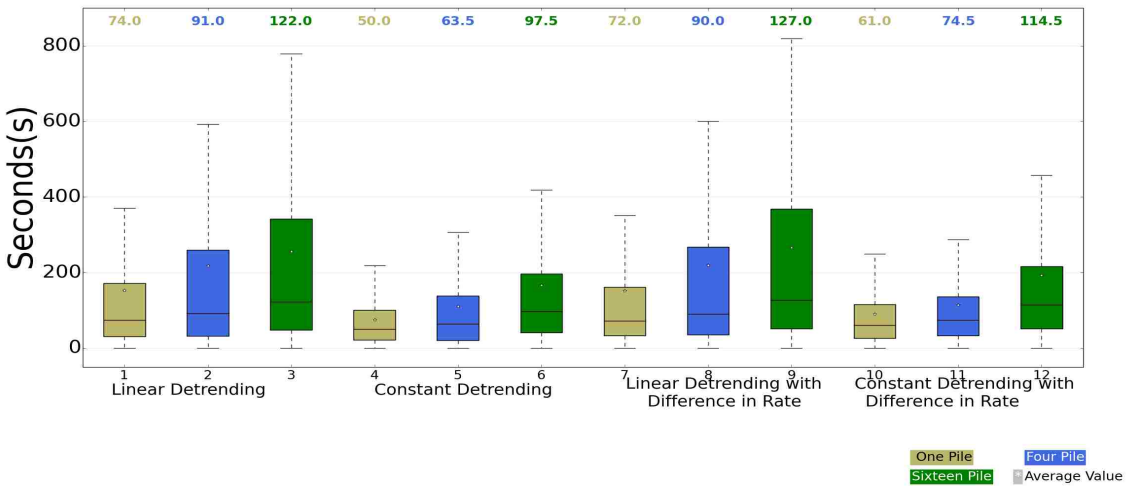


Figure 6.44: Efficiency of all methods with 48 ants for both combination of communication and memory

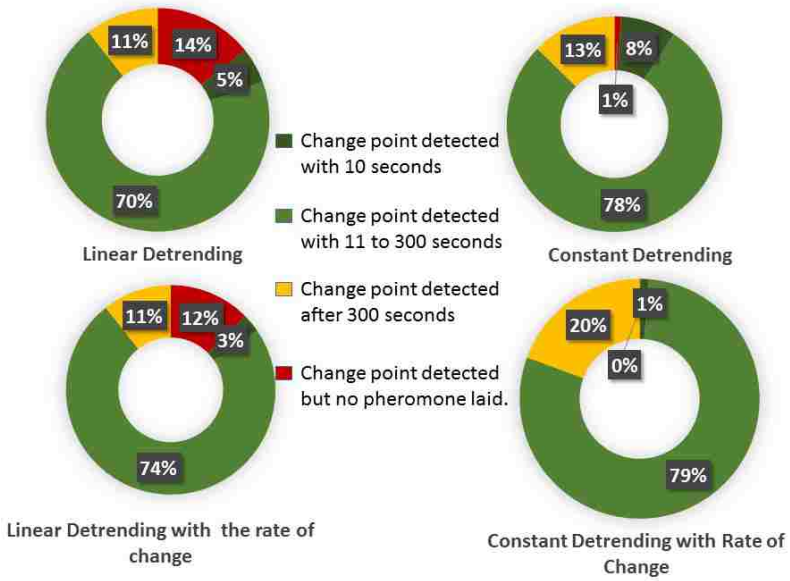


Figure 6.45: Efficiency chart for 96 ants, one pile, memory and communication setting.

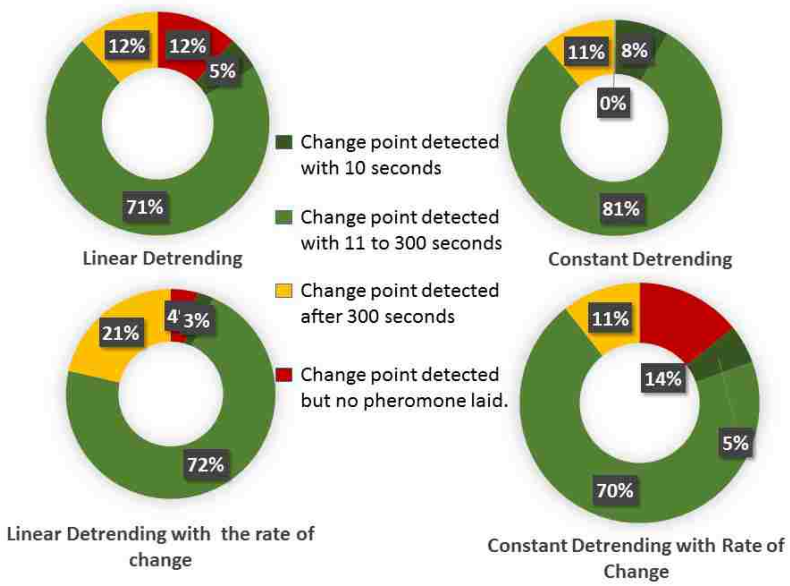


Figure 6.46: Efficiency chart for 96 ants, four pile, memory and communication setting.

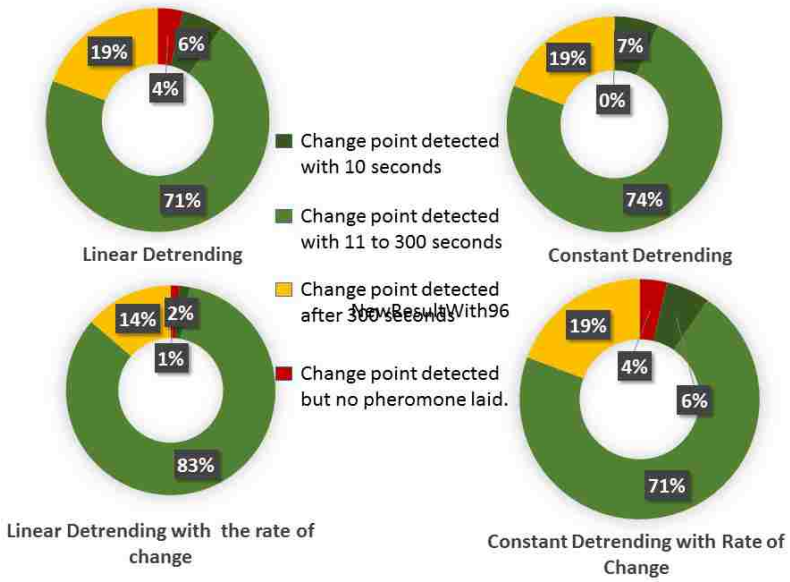


Figure 6.47: Efficiency chart for 96 ants, sixteen pile, memory and communication setting.

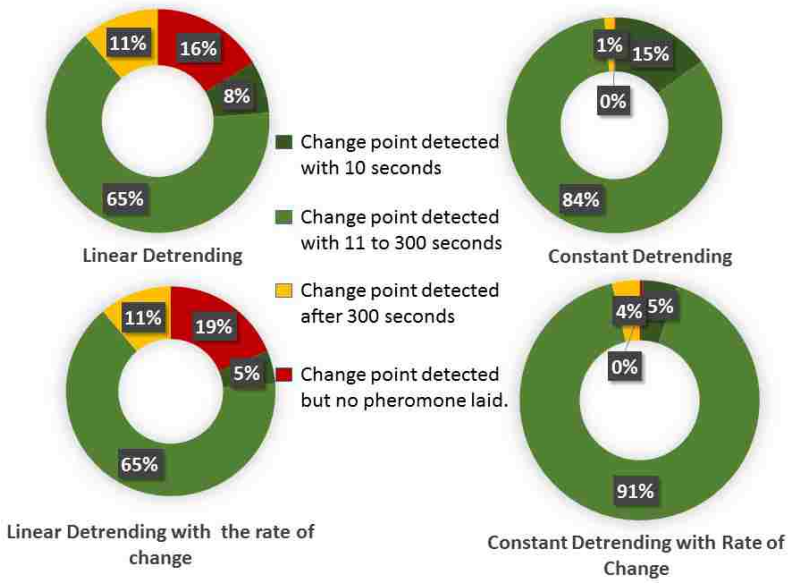


Figure 6.48: Efficiency chart for 48 ants, one pile, memory and communication setting.

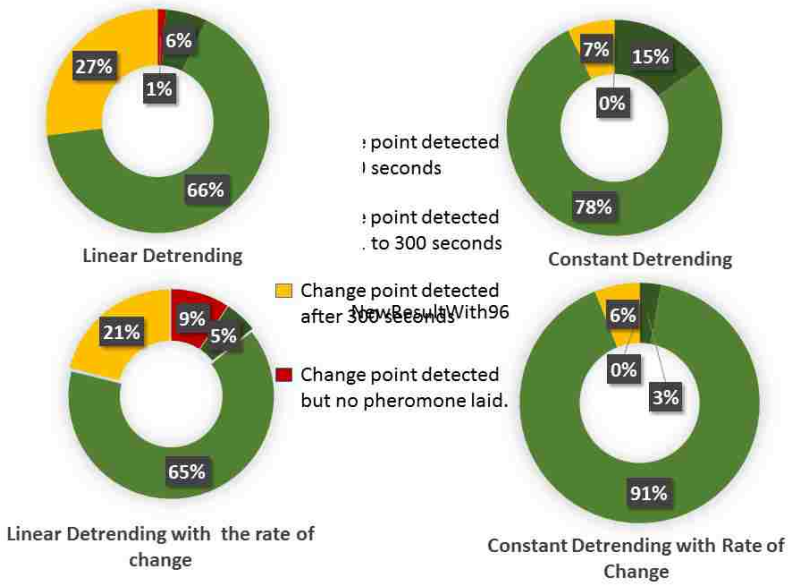


Figure 6.49: Efficiency chart for 48 ants, four pile, memory and communication setting.

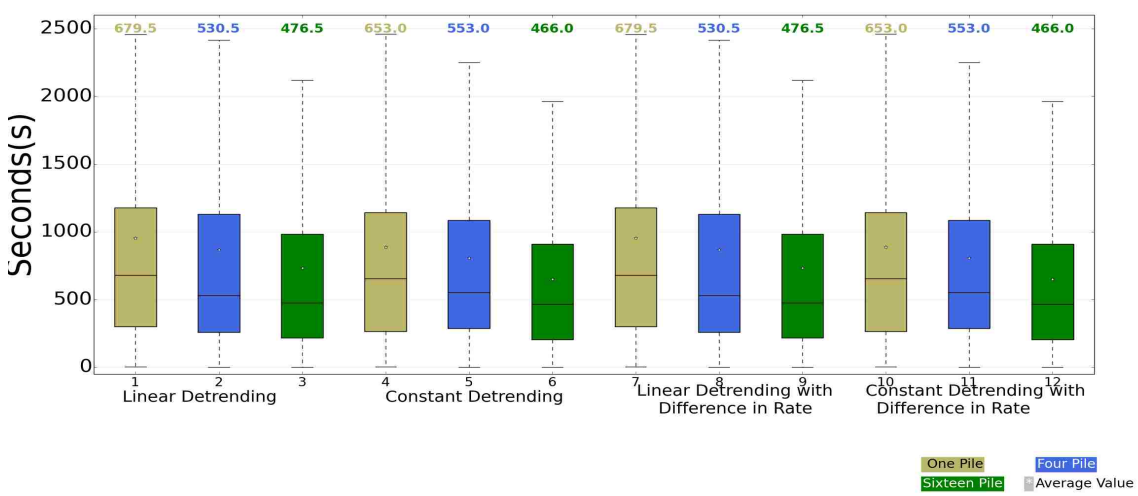


Figure 6.50: Efficiency of all methods with 96 ants for memory only parameters

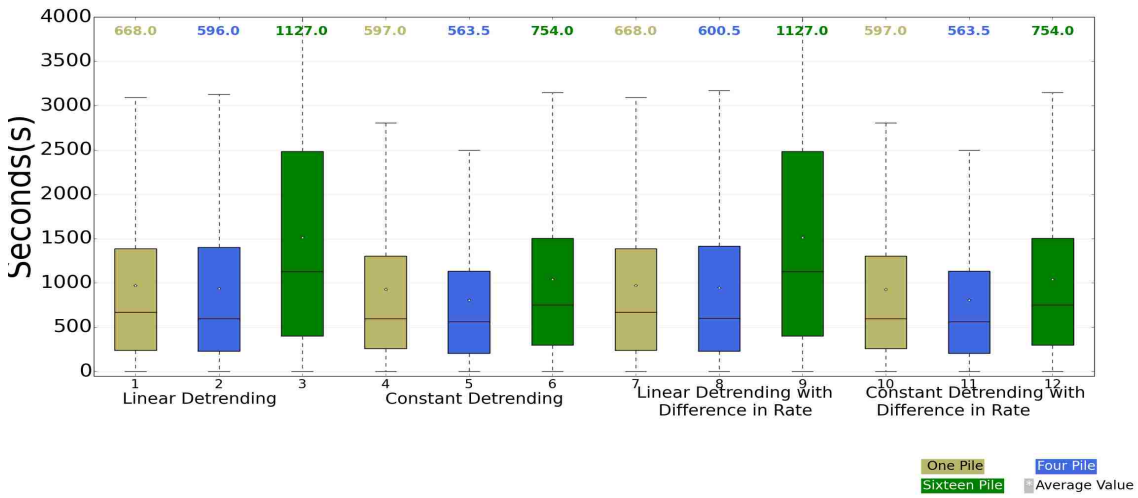


Figure 6.51: Efficiency of all methods with 48 ants for memory only parameters

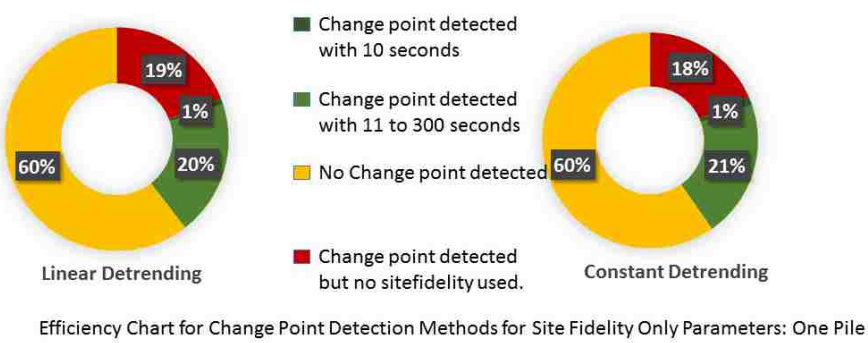


Figure 6.52: Efficiency chart for simulation with 96 ants, one pile, memory only setting.

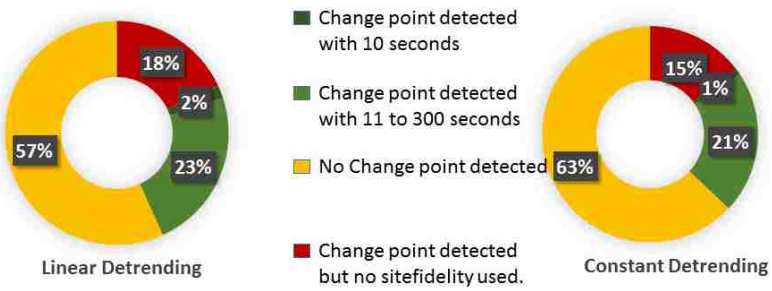


Figure 6.53: Efficiency chart for simulation with 96 ants, four pile, memory only setting.

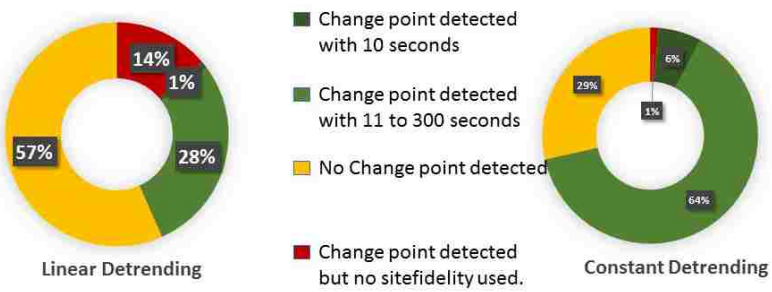


Figure 6.54: Efficiency chart for simulation with 96 ants, sixteen pile, memory only setting.

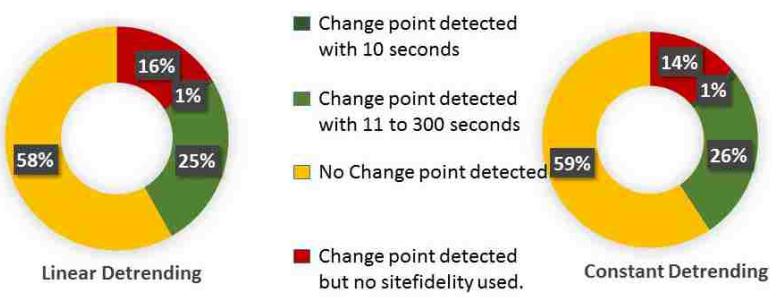


Figure 6.55: Efficiency chart for simulation with 48 ants, one pile, memory only setting.

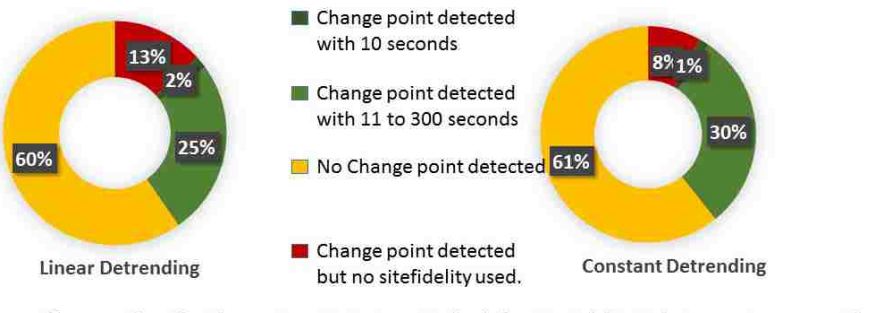


Figure 6.56: Efficiency chart for simulation with 48 ants, four pile, memory only setting.

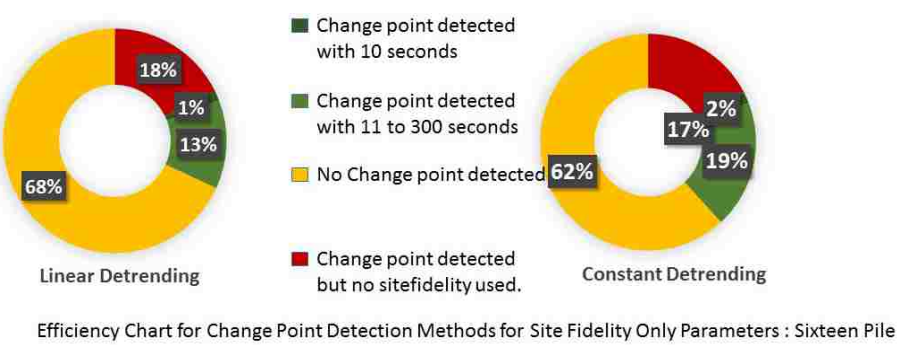


Figure 6.57: Efficiency chart for simulation with 48 ants, sixteen pile, memory only setting.

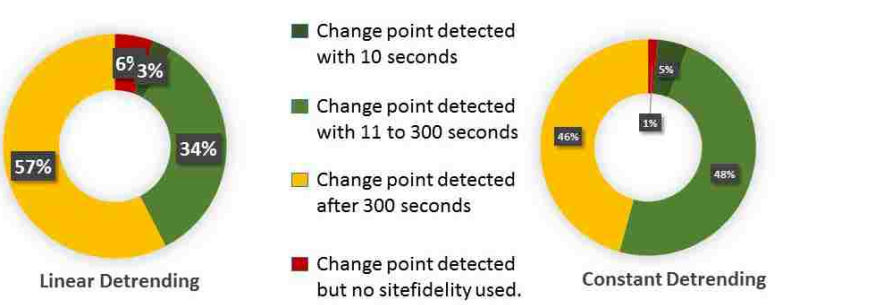


Figure 6.58: Efficiency chart for *P. desertorum*, sixteen pile, memory only setting.

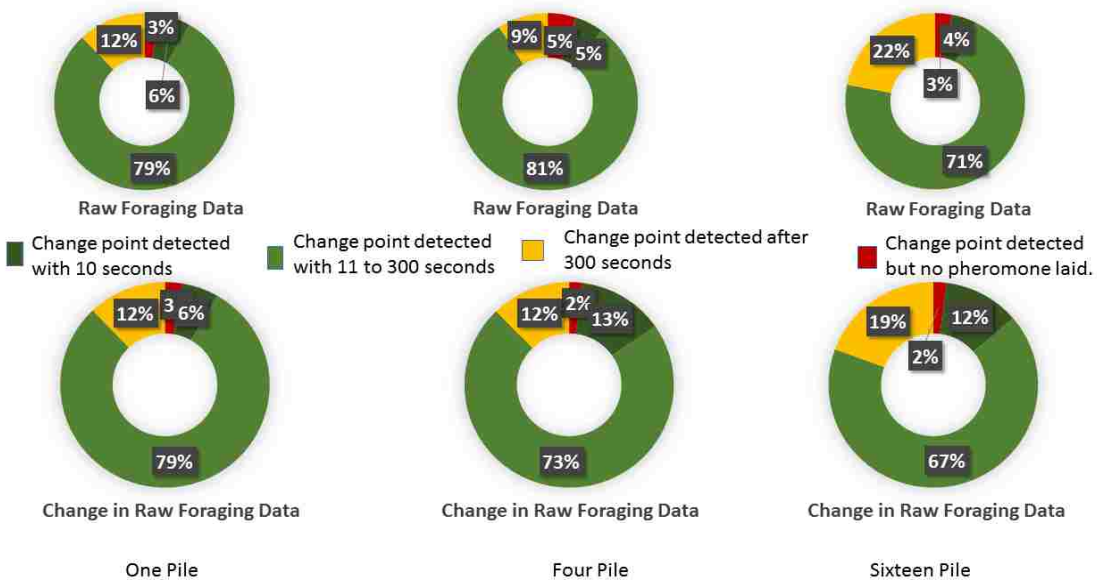


Figure 6.59: Efficiency of change point detection algorithm on foraging rate and change in foraging rate for memory plus communication parameters.

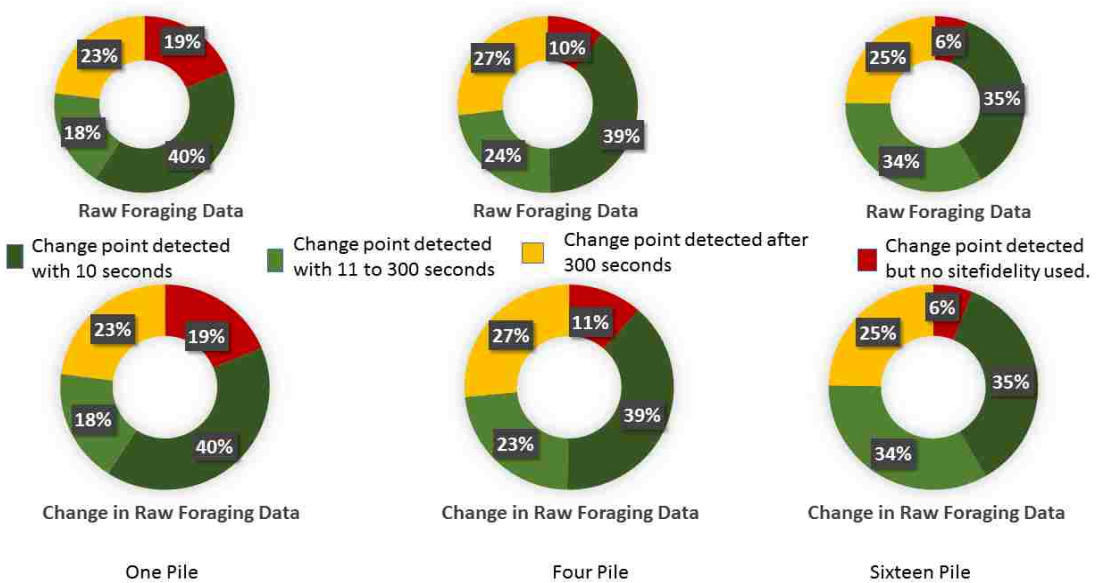


Figure 6.60: Efficiency of change point detection algorithm on foraging rate and change in foraging rate for memory only parameters.

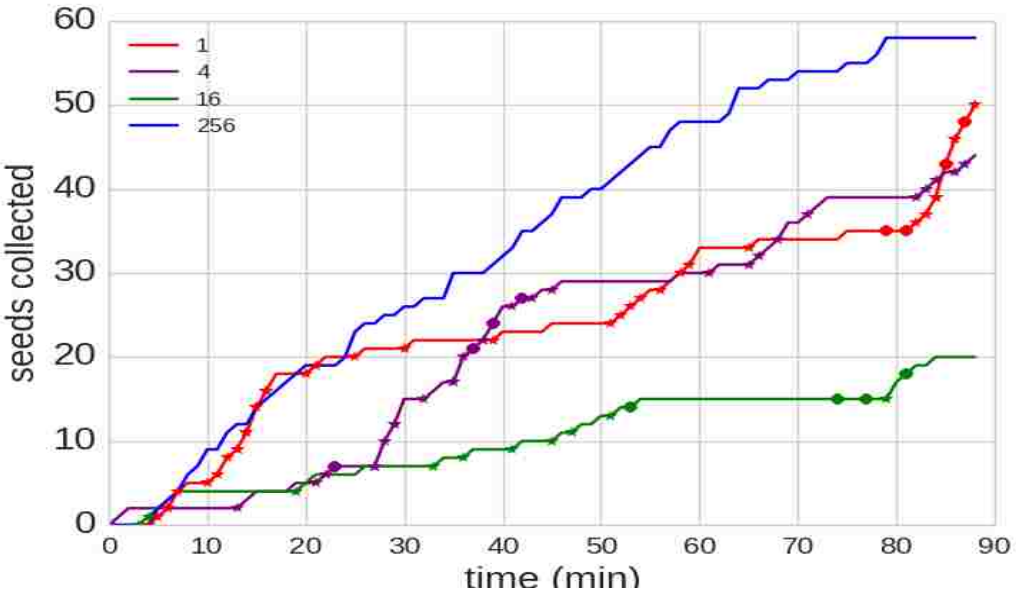


Figure 6.61: A plot of seed collection from simulation of 96 ants.

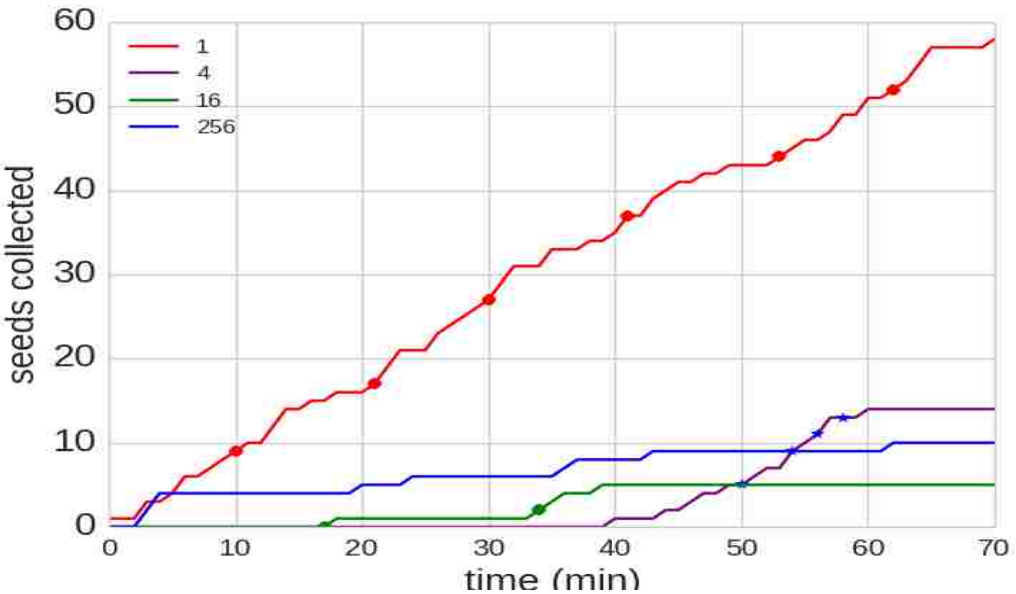


Figure 6.62: A plot of seed collection from one of the field experiment of *P. desertorum* with change points on the collection rate.

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] Tatiana Paz Flanagan, Kenneth Letendre, William Burnside, G Matthew Fricke, and Melanie Moses. How ants turn information into food. In *Artificial Life (ALIFE), 2011 IEEE Symposium on*, pages 178–185. IEEE, 2011.

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

[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] Kenneth Letendre and Melanie E Moses. Synergy in ant foraging strategies: memory and communication alone and in combination. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 41–48. ACM, 2013.

[16] 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.

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

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

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

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

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

[22] 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.