Department of Computer Science

COMSM2202

Research Review

Inferring population dynamics of house-hunting ants

Student:

Christina Pantopoli

Supervisor:

Dr. Oliver Ray

Executive Summary

The behaviour of ant-species *Temnothorax albipennis* has been extensively studied by the Bristol Ant Lab, which led to the discovery of many mathematical models that examine and analyse the process of an ant colony emigration from one nest site to another. These models describe this occurrence by a set of differential equations which detail the development of the different categories of the ants in a colony. Although these mathematical models were developed by the use of different equation discovery models, they do not take into account the relationship between the equations as each equation was created individually.

The aim of the project is to create a differential equation discovery model through the use of genetic algorithms while taking into account the relationship between the equations and then integrate that into a program. The software will be developed on either C or Mat Lab as they offer libraries on genetic algorithms and other algorithms that are used in machine learning to solve equations. Within this project we will also be looking into how credit assignment works on equation discovery systems. The final step aims to utilise the genetic algorithms produced, to generate ODEs (ordinary differential equation) that represent the emigration process of an ant colony and compare them with existing ODEs that were discovered using other models. This project is 80% part II, 10% part I and 10% part III.

Therefore, the key deliverables of this project are:

- the production of a genetic algorithm that outputs the expected equation based on the input and the generation of equations that might depend on the development of another one,
- a prototype software that will implement the genetic algorithm to find equations,
- an evaluation of the algorithm by conducting experiments with different input data and compare the result with the expected equation by applying methods of statistical analysis for instance calculating the margin of error,
- an assessment of the credit assignment of the genetic algorithm - discovering if there are errors, where they occurred and how important they are to the accuracy of the outcome,
- an analysis of the ODEs created by this software and previous software by determining their accuracy

Upon completion, this project will offer development in tracking and understanding the behaviours of this species of ants and in solving sets of equations. The main novelty is that it will provide an equation discovery model that can find sets of equations whereas all the existing ones do not offer that. Moreover, it will help deliver clearer results for the population dynamics of ants during the process of ant emigration. After we run data collected from experiments that observe the ants during emigration, the software will deliver more accurate equations that outline the behaviour of ants. This project will also assist in the improvement of the tracking software for the ants that is used as we could compare the data that the tracking software provides with the data that the ODEs supply and use a statistical analysis to find if the data are precise. If not then the project could enhance the tracking software so it will take into account more variables including other defining conditions.

Contents

1. Introduction and Scope

1.1. Introduction

Finding solutions to problems using algorithms and optimization methods has been long considered a major development in the field of evolutionary computation. Within this project we aim to use these methods to develop a new equation discovery system that will help us learn equations from data in a more accurate way and develop sets of them instead of a single equation each time. Through the development of this system we seek to solve equations that describe the process of an ant colony emigrations from one nest to another and compare them with equations that have been discovered using other equation discovery systems. As a consequence a piece of software will created that will execute the equation discovery system to make it easier for the user to solve mathematical formulas. At the end of the project a statistical analysis will be carried out to evaluate the system by conduction experiments with different inputs and for this part my mathematical background will come in good use.

The added value this project will offer is in the fields of mathematics and biology it will provide an equation discovery model that can find sets of equations and clearer results for the population dynamics of ants during the process of ant emigration. It will also help improve the tracking software for the ants that is used as it could compare the data that the tracking software provides with data that the ODEs from the system will supply and use a statistical analysis to find if the data are precise.

1.2. Motivation

This project was derived from pursuing the creation of equations that can explicitly describe the behaviour of *Temnothorax albipennis* during the emigration process. This species has been extensively been study in the University of Bristol as many useful information can be obtained by observing the behaviours of ants and the decisions they make in different environments.

Fig. 1 Temnothorax albipennis

The Bristol Ant Lab (University of Bristol) has conducted many experiments on this species of ants during the emigration process from one nest to another and each time a parameter changes so as have a better understanding of this process (Computational Ant Lab, n.d.). These experiments take place in the biology department where colonies of ants are held inside boxes which during the experiments are placed inside a fenced surface and the entrance of the box is

opened. Most experiments include the placement of a box that contains a nest and an empty box that will become the new nest after the emigration.

The emigration has to be put into action since the ants will not abandon their current nest if something does not change. The ants are forced to start emigration and search for a new nest by either taking the roof off the box so there is more sunlight in the nest which they do not find preferable so they try to find a nest which is better fit.

For example, one experiment includes obstacles in the route from one nest to the other, another has the initial nest have a see-through roof and many other variables change for each experiment. These experiments are recorded and then the videos are processed through a tracking software which offers data that can be used as input in the equations discovery models.

The tracking software provides data for each category of this species that exists in the colony based on their behaviours for example if the ant searched for a new nest then it is considered a scout, if it gets carried by another ant then it is considered as a passive ant and so on. Finally, after this data are processed through the equation discovery systems they offer mathematical models that describe the movement of the ants during the emigration.

Different equation discovery system were used by other postgraduates and PhD students of the University of Bristol during the period of their thesis and sets of equations were discovered accordingly. The models that have been used so far are Pratt, Planqué and Nutonian Eureqa (Appendix. [1]). Out of the models used, the most capable model is Eureqa as it can discover equations for a variety of data even if the data are extremely complex. However, Eureqa as all the models mention above have the ability to only execute on single equations which set back the discovery of sets of equations.

The outcome of this project will provide the ability to use a system so as to solve multiple equations in parallel and each equation is relevant to each other so their accuracy should also improve. Moreover, it will enable biologists to derive more precise results that describe the behaviour of ants that could also contribute to the improvement of the tracking software that is used to interpret the movements of ants.

1.3. Areas of Research

To achieve the objectives set out, the necessary knowledge of various scientific areas is in needed. These scientific areas can be identified into two main categories: machine learning and equation discovery and they will be covered later on inside this research review.

Indeed Machine learning is an area that will be necessary to implement the project. Within this segment we will describe why machine learning is so important nowadays and the various tools that are used as Machine learning techniques. We will also explain the reason as to why a research on Machine learning was conducted and why tools offered from studying Machine learning will be used for this project.

Machine Learning includes many different types of algorithms and one of them is genetic algorithms. Genetic Algorithms are used to build predictive models which is paramount to the

realization of this project considering we are going to predict equations based on data given. Within this section we will describe why genetic algorithms are the preferred method that will be used for this project and a detailed representation of a normal process it follows. Additionally, we will cover the advantages and disadvantages of using them.

Genetic Programming: In this section we will include information of what is genetic programming and how it is used. Also, we will record some similarities and differences between GAs and GP and more details on how to apply a genetic programming algorithm.

We cannot forget to mention equation discovery systems as the projects aim to creating one. This part will briefly cover some of the equation discovery systems that were already mention. Nutonian Eureqa will be examined in more detail as it is the foundation that will be used to create our own equation discovery system.

2. Machine Learning

2.1. Broad Machine Learning Background

Machine Learning is a section of computer science that was developed from trying to improve the performance of computers so as to give them the ability to learn. Machine leaning consists of the study of algorithms and models that increase their performance through experience.

A more formal definition from Tom. M. Mitchell on what machine learning is : "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997).

There are three types of problems and tasks in machine learning and they are divided in supervised learning, unsupervised learning and reinforcement learning.

- Supervised learning: The program is offered the inputs and the befitting outputs and the objective of the program is to create the map that connects the inputs with the outputs.
- Unsupervised learning: No conditions are given to the program, the purpose is to find relationships from the input for example it searches for hidden patterns.
- Reinforcement learning: The program is not directly offered inputs, it is given an objective and it attempts to reach it by performing different tasks.

This project we will be using reinforcement learning as we know what the expected results should be and we offer the algorithm a condition of when to stop.

Machine learning can also be divided in other categories based on the output of the task such as classification problems, regression and other.

- Classification problems is when the outputs are separated into specific categories. For instance one classification problem is a program that would try to predict if a patient could have cancer or not based on the age. Thus the output of this program is yes or no and the input given would be data of previous patients that had cancer and others that did not.
- Regression problems are used for predicting results that are continuous such as finding equations by using linear or non-linear regression.

In our case the category our program will belong to is a regression problem as the outputs keep changing during the execution of the program.

Machine learning algorithms are of great use when trying to solve problems that involve big data sets and trying to discover patterns or irregularities in the data or when trying to make a program adapt to conditions that keep on altering.

The modelling of a machine learning problem includes selecting the appropriate training experience, choosing the type of the target function and how it is represented and deciding on the algorithm for developing the target function from training examples.

2.2. Genetic Algorithms Background

2.2.1. General Information

Humankind has always wanted to develop new techniques and tools so that it would enhance the performance of tasks, as a result scientists believed that evolution could be used in systems as a tool to optimize solutions for technical problems. By observing nature scientists support that natural evolution by the use of reproduction and other genetic operations develops rapidly and in a very complex level (Goldberg and Holland, 1988). This leads to the justification that if it is applied on computational problems the solution of them will be discovered faster and in a more optimized approach.

The idea behind genetic algorithms, was the development of a population with all the possible solutions for a given problem and the application of techniques inspired from genetic evolution and natural selection. In the mid 70's genetic algorithms made their first appearance when J. Holland created a model that could solve optimisation problems with the use of GAs.

GAs are mostly used to find the best solution for a problem, so the search for it starts with a population of initial hypotheses and the hypotheses could be considered as the genes in a chromosome. With the use of natural selection and genetic operators such as mutation and crossover, a new generation of hypotheses (genes) is born by selecting the hypotheses that are evaluated to be most fit. The fitness is calculated with the use of a probabilistic model which calculates the fitness of one hypothesis to the proportional fitness of the other current hypotheses. Natural selection guarantees that the hypotheses with the best fitness will be used to generate the future populations. By using the crossover operator during the process, the GA combines genes from two hypotheses and combines them to create two new offspring which will be better fit than their parents. Genes could be mean bits of strings or a node that contains a part of a program of each hypothesis. Another genetic operator used is mutation which is often executed after crossover, it performs a random change in the new generation of hypotheses which allows for a single child to be produced from a single parent. An impressive achievement is that by combining the crossover and the mutation operator, the population can move from the local optimum it might get deadlocked in, this way the fitness of the new generation gets keep improving until the GA will provide the best hypotheses. GAs have many advantages and disadvantages and some of them will be stated below.

2.2.2. The pros and cons of GAs

There are many reasons why we are going to use GAs:

- They can seek the space of hypotheses that have complex pieces and the effect of each piece on the hypothesis may be demanding to describe with other methods.
- GAs do not require any information on the gradient descend.
- Their behaviour when solving big scale optimization problems is ideal and they offer a solution more accurately and faster since they can search for hypotheses in parallel.
- They can find a hypothesis which is a global solution and has high fitness.

However, GAs are not always suitable as they have some disadvantages:

- GAs do not guarantee to find the local optimum, since if they reach a point where the fitness of the population satisfies a condition, they will stop their evolution.
- The best hypotheses is found by comparing it with the rest hypotheses that belong in the population as a result if the initial is not diverse enough. Even if the GAs run multiple iterations, they will not discover the best hypothesis but just the “best” hypothesis based on the original parents.
- GAs are unable to find solutions when the fitness function returns only true or false as the search will not converge on the answer.

2.2.3. Typical structure of Genetic Algorithms

GA($Fitness, Fitness\_threshold, p, r, m$)

Fitness: A function that assigns an evaluation score, given a hypothesis.

Fitness_threshold: A threshold specifying the termination criterion.

p: The number of hypotheses to be included in the population.

r: The fraction of the population to be replaced by Crossover at each step.

m: The mutation rate.

- *Initialize population:* $P \leftarrow$ Generate p hypotheses at random
- *Evaluate:* For each h in P, compute $Fitness(h)$
- While $[\max_h Fitness(h)] < Fitness\_threshold$ do

Create a new generation, P_S:

1. *Select:* Probabilistically select $(1-r)p$ members of P to add to P_S. The probability $\Pr(h_i)$ of selecting hypothesis h_i from P is given by

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

2. *Crossover:* Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P, according to $\Pr(h_i)$ given above. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s.
3. *Mutate:* Choose m percent of the members of P_s with uniform probability. For each, invert one randomly selected bit in its representation.
4. *Update:* $P \leftarrow P_s$.
5. *Evaluate:* for each h in P, compute $Fitness(h)$

- Return the hypothesis from P that has the highest fitness.

Fig.2 Basic steps of a GA (Mitchell, 1997)

The core body of a GA consists of five steps:

1) Initialization: The initial population is created either by picking random hypotheses or by disrupting an input hypotheses. The initialization of the population is not considered critical since the population will evolve so as to become have more disparate hypotheses based on the optimization parameters.

2) Evaluation: In the second step the fitness of each hypotheses is calculated with the use of the fitness function which is the only criteria of evaluation of the hypotheses. This assessment is used either as a terminate condition or the process of probabilistic choosing which of the current hypotheses are going to be used to create the next generation.

   The fitness function receives a hypothesis and returns a value that represents the degree of fitness for that hypothesis, usually the output is in the margin from 0 to 1 where 1 indicates that the hypothesis meets all the conditions of the fitness function and it is an acceptable solution. The representation of the fitness function depends on the problem and can be from simple to very complex. Even though the ideal fitness function is continuous and monotonous it rarely is, so the next thing we seek is for it to have many local maximum or an isolated global maximum

   The most common approach to select which hypotheses are going to be selected is based on appraising their quality is probabilistically. The decisive point is that the fitness function that is chosen must be accurate and it should provide a computational cost that gives the best results.

3) Natural selection: In this step the hypotheses with the best fitness score are put together in a subset in no particular order. However, the hypotheses which are not fit enough are discarded from the population. The number we select and dispose is not random, we use one of the variables passed as input in the GA to compute the proper amount in each case. In the end the best fit hypotheses are added to the population of the new generation. This process is based on the Darwinian theory of evolution (Forrest, 1993; Mitchell, 1997).

   Hypotheses can be selected or replaced depending on many different methods. The method mention in Figure 2 in the selection process is called fitness proportionate selection and it computes the ratio of its fitness to the sum of the fitness of all the hypotheses, which shows the probability that a hypothesis will be selected. Another method is the tournament selection where the population is divided into pairs and then the most fit of the two remains becomes a parent for the next generation whereas the other one gets changed. This method offers a more diverse population. An additional method is rank selection where the hypotheses are ranked according to their fitness and there are chosen based on their rank (De Jong, 1993).

   When selecting the hypotheses that will be used for the new population, one should be in caution to avoid crowding which happens when a hypothesis is greatly better than the others so the population instinctively carries characteristics only from that hypothesis. This leads to the population becoming less diverse and it slows down the progress of the GA.

4) Crossover: In the fourth step a genetic operation is performed on our population. A certain amount of hypotheses are picked based on their fitness and placed into pairs and they are combined so as to produce two children by taking information from each parent. The children are then placed into the population of the generation. It is considered that the children are better than their parents since they carry characteristics from both of them.

   There are different methods to apply crossover based on the crossover mask that will be chosen to use. The crossover operation continues being applied to the new population. However, the crossover operator might generate an offspring which is not suitable for the population for instance by combining the characteristics by the parents an individual is created

that is problematic, as a result the crossover operator has to be designed so that it constraints the development of children that should exist in the new population (Forrest, 1993; Eiben and Smith, 2003).

5) Mutation: In the final step another genetic operator is executed which is the mutation operator. This operator produces one child by on parent by changing random characteristics of the parent. This step offers the possibility to generate solutions that do not exist in the population.

At this point, the population contains hypotheses that have better quality than the previous one and the steps from 1 to 5 are repeated as long as the GA is terminated. The GA can be terminated depending on many conditions:

- There is a fixed number of iterations that the GA is supposed to perform.
- The degree of suitability of the best solution is greater than a given threshold.
- Finding a solution that .meets the minimum standard.
- If better results are not generated after a number of iterations that means it has reached a local optimum and the operation should come to an end.

2.2.4. Convergence and Update

The method selection of parents that will be used in the crossover operation significantly affects performance of the GA. Two problems that often occur during the selection process are premature convergence and slow convergence. Convergence is the prevalence of a solution or small variations of that solution in a large percentage of the population, if the GA is efficient then the population will converge after enough iterations to the global optimum. In premature closure, the population rapidly converges around a solution that is a local optimum. This phenomenon occurs in cases where a hypothesis in the population has better fitness than the rest, this also can be observed by the fitness function which will show very sharp peaks and intense local maxima (Eiben and Smith, 2003; SWATI, 2015). To address this problem many techniques have been discovered such as scaling, remapping the fitness function and many others. In the case of slow convergence the exact opposite transpires after a huge number of iterations the population still does not converges and the only way to work this out is by changing the fitness function to give a more intense fluctuations. Previously, I mention that some of the original solutions are placed in the new population along with the new developed children this method is called partial-renewal and it allows the children to compete with their parents so that the best one prevails.

2.2.5. Problem Domains

It is not specific where it is appropriate to use GAs for an optimization problem. Some scientists claim that there is no better method for optimization other than GAs even though there similar methods that exist such as gradient-descent, Monte-Carlo, Simplex – Simplex optimization and many others that can perform the same task. The key is to use the knowledge of the system which is going to be optimized in order to choose the best technique that finds the best solution faster.

When there is not enough knowledge on the calculation of the gradient descent then GAs are preferred. Furthermore, another reason why GAs are preferred is their versatility. GAs can use the knowledge of system so as to customize itself for example if the program is caught in a local optimum then it can change the mutation operator. So while there is no guarantee that GAs can behave better than other methods for an application, they do not fail to achieve a satisfactory solution to the optimization problem. Another feature GAs have is that they do not directly optimize their variables but they optimize their performance.

For this project the conditions are optimal to use GAs since the population that will be used will not be too complex and the calculation of the global optimum is not a necessity as other methods can be used to accomplish the same target.

2.3 Genetic Programming Background

Genetic programming (GP) is often inaccurately thought of as a type of GA when as a matter of fact it is an implementation of GAs where the inputs are pieces of programs. There many programs that can be optimized by the use of GP since a great deal of computer programs can be genetically developed. GP is identical to GAs in the fact that they can run in parallel and the most effort during their execution is given to the calculation of the fitness function.

The most common representation of a program that is handled by a GP is a tree structure therefore the programs that are usually used with GP are from programming languages that can be represented in trees. GP can also be applied to programs that do not express trees, one example of this is linear genetic programming.

2.3.1. GP Population

The first step that is taken to create a GP is the initialization of the population. Similarly to GAs, GP works with the same structure used for GAs as they have an initial population. However a GPA's (genetic programming algorithm) population is compromised from programs which are created by functions constructed by the user, variables and constants that will be used to build a program by combining parts of the population (Mitchell, 1997). The program itself is described as a tree and the functions contained in it correspond to nodes in the tree. As mentioned the population is produced by combining primitive arguments that are given, to accomplish the initialization of the population we can either do it randomly or by using certain inquiries. Three methods have been discovered that create a random initial population and the most common one is the ramped half-and-half method (Affenzeller, 2009; Eiben and Smith, 2003; Koza, 1992; Brameier and Banzhaf, 2007). Researchers continue their work in finding new methods to initialize a population since depending on the method used the GP offers different results. An initial maximum length for the trees is given D_{max}, also the set of functions F is also provided so as to create the original population. The three methods that have been discovered so far are:

- Full method: All the trees that get built have the same depth D_{max}, the depth is the number of the branches it takes to move from the beginning until the root node not the sum of the nodes contained in a tree. For each tree, one of the functions F is chosen randomly and based on the number of arguments the function chosen needs, the fitting amount of branches is attached to that node. Before adding the function to the tree, a check is carried out to control that that the depth of the tree will be within limits even after the addition of the new tree. As a result a function F is chosen if depth $d < D_{max}$ or terminal T is chosen if $d = D_{max}$.
- Grow method: When using this method the trees have various lengths but the depth of any tree cannot exceed D_{max}. The development of each tree begins with the randomly or probabilistically selecting an arguments that belongs the space that is a combination of functions and termination nodes, $F \cup T$ if $d < D_{max}$ that means even though each branch of the tree can have different length the maximum length of the whole tree is D_{max}.

- Half-and-half method: This technique is a combination of the full and grow methods where 50% of the population is created using the full method and the rest is developed by using the grow method using more than one maximum depth.

The initial population consists of a number of trees and throughout the execution of the system a new generation of population is produced by applying genetic operators that have been mentioned in the previous section. During the selection process the fitness function for each program is really the result of executing the program with the data given.

2.3.2. Crossover

If the crossover operator is applied on genetic programming algorithm then when we have choose two parent programs and a random node of each parent is picked and then those nodes get replaced with each other. This type of crossover is called one-point crossover where the same crossover point is selected form the parents and then those nodes are switched.

There are other forms of the crossover operator for GP, these are n-point crossover, uniform crossover, context-preserving crossover, size-fair crossover, ripple crossover. The uniform crossover (subtree crossover) is very similar to how uniform crossover works in GAs, during uniform crossover the two children are developed by randomly choosing a point in each parent, in the common region of the two programs if they have different shapes, and then they exchange each other. Figure 3 and Figure 4 display two parent trees with various lengths as a result with the use of a uniform crossover, two crossover points are randomly selected by starting from the base nodes that are included in the common of the two parents.

Uniform crossover offers more diversity in the population because it fuses pieces of smaller programs to other programs. For instance, if the node that gets picked from one parent is at the bottom of the common area and it is a function then the nodes it consists of are also transferred to the other parent, the node that is selected is also called a crossover fragment (Koza, 1992; Poli, Langdon and McPhee, 2008).

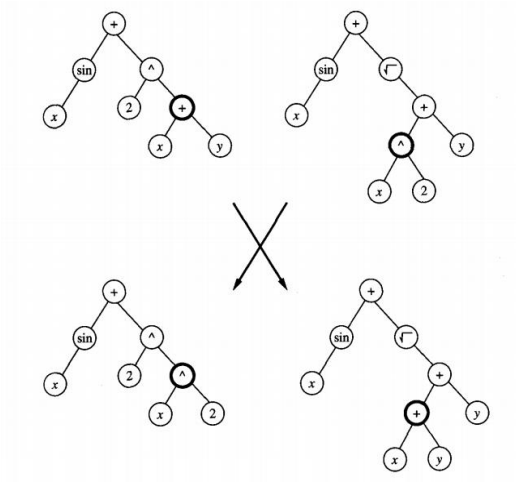


Fig. 3 Crossover operation (Mitchell, 1997)

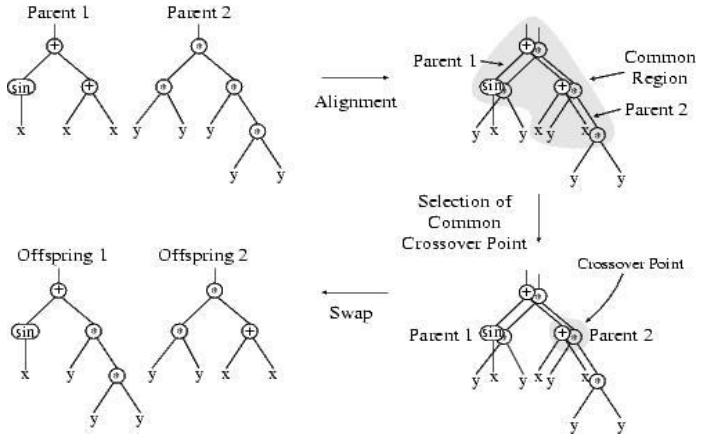


Fig. 4 Example of one-point crossover between parents of different sizes and shapes. (Poli, Langdon and McPhee, 2008)

However, if the parents have varying sizes and shapes then the children will also have different sizes and there is also the possibility that the length of the children will keep on increasing if measures are not taken to prevent this from happening. This incident is called bloat and one of the preventative methods that has been discovered is known as parsimony pressure. This technique suggests that individuals in the population which are comprised from many subtrees should have their fitness function reduced so as to minimize their possibility of being selected as parents for the new population (Eiben and Smith, 2003).

Furthermore, the mutation operator in GP operates the same way as in GAs, it is applied by randomly choosing a node and altering it into something from the initial functions or variables. As mentioned there are two genetic operators that can be applied crossover and mutation, in contrast to GAs, GP can use either of these two operators at the same time. The figure below is taken from Eiben and Smith, 2003 and it displays the creation of a new generation during the process of a GA and the process of GP.

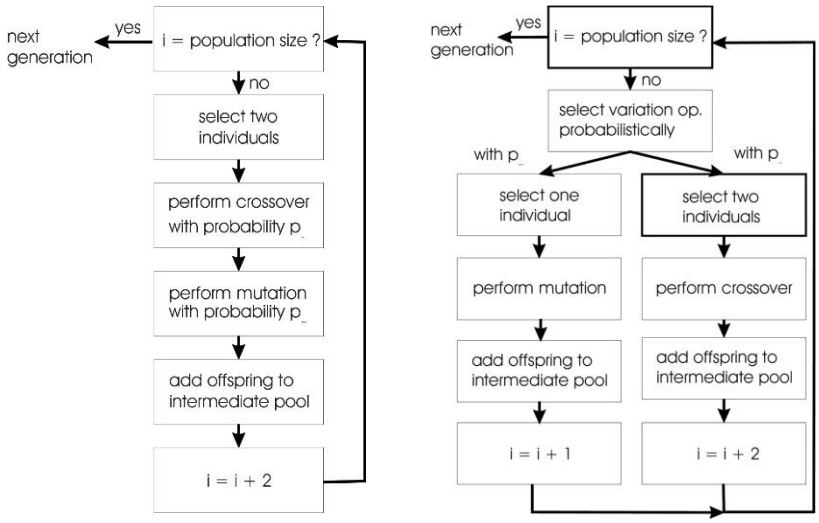


Fig. 4 GA loop Vs GP loop

2.3.3. Parallelization in GP

As previously disclosed GAs can search in parallel for solutions, GPAs can also search in parallel. For this project we will use GPAs since when GPAs search side by side for solutions the program is divided into smaller sections and this causes it to reduce the time each iteration takes. This feature of GP will be very useful for the implementation of our system considering that it will try to build equations from data and we want that to be in the shortest time possible. This section includes the methods that are used to apply parallelization in GP. With regard to the way a GPA can run in parallel, it can be divided to two main categories, physically distributed and logically distributed populations. These two categories can then be divided into smaller subcategories which will be described in more detail to understand which will be more suitable for this project (Koza, 1992; Poli, Langdon and McPhee, 2008).

There are many ways to run parallel GP either by running the system on parallel hardware or by building the GP in way that could divide the workload for instance by splitting the population into

sub populations and running the algorithm for each one and then exchange a number of individuals from one population to another. The method mentioned above is called coarse-grained method and it increases the variety in each subpopulation since when each population is developed it contains certain characteristics and then when the GP migrates trees from one population to another, it also carries these characteristics in the new population. This method can be developed so it can run on multiple processors or on one processor.

Another the method that is shown below in Fig.3 is parallelization where multiple fitness functions are calculated and it is also called master-slave GP (Poli, Langdon and McPhee, 2008). However, to use this method the user must make sure that if the calculation of the fitness function takes more or less time for some individuals in our population, then the time you apply the selection of the parents and the genetic operators are not affected.

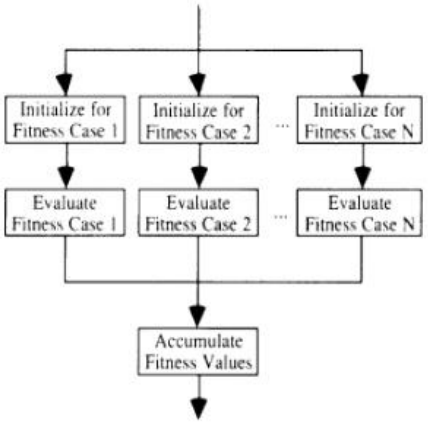


Fig. 5 Parallelization at the level of fitness cases (Koza, 1992)

Various GPAs were discovered that implement parallelization in some of the methods mentioned, some of the tools that apply those methods are IIP (independent and identical processing), SPMD (Single Program Multiple Data), FPGAs (Field programmable gate arrays), CAGE (CellulAr GEnetic Programming tool) (Koza, 1992; Poli, Langdon and McPhee, 2008; Folino, Pizzuti and Spezzano, 2002; SWATI, 2015; Shonkwiler, 1993; Miller et al., 2001; Miller et al., 2001).

The ability of the GPA converging to solution is determined by the primitive functions and variables supplied and also on the data that the fitness function is supplied with to calculate it. As a consequence when developing the system through this project, special consideration should be taken on which variables to use and many attempts will be made with different inputs so as to provide the best system.

3. Equation Discovery Systems

This section of the research review will provide some general information on the various equation discovery systems that exist and a better understanding of the Nutonian Eureqa system as it is the model that is most suitable to what this project aims to achieve. With the development of technology and economy more data are created every day and processing this data and finding patterns in them is becoming increasingly popular. As a result, many systems have been created that take this data and provide equations that represent this data, these systems are called equation discovery systems. The development of such systems begun from the 1970s and have continued to develop until now with different tools as new techniques are being also developed.

The logic behind equation discovery systems is that they utilize genetic algorithms to search through the data and map out the primitive functions given so that the combination of them represents a relationship that fits the data. Equation discovery systems determine the structure that the given function and constants should be put into (Equation discovery - Introduction, 2000).

3.1. Generic Equation Discovery Systems

During the end of the 90s many equation discovery systems (EDSs) were designed such as COPER in 1986, FAHRENHEIT in 1989 and 1992, ABACUS in 1990 and many others. However these systems do not solve differential equations which is what this project aims to do so they will not be reviewed (Equation discovery - Introduction, 2000). A quote from Todorovski and Džeroski, “Different equation discovery systems explore different spaces of possible equation structures” indicates that an EDS can be constructed so as to solve various equations and it can be constrained so as to only create solutions that fit the data (Todorovski and Džeroski, n.d.).

The first ever developed EDS was BACON.1 in 1987 it is considered to be the standard that was used when new EDSs were created. There are 6 versions of the BACON discovery system that were built and each one is better refined than the previous. In 1993 LAGRANGE was created and it was able to find differential and ordinary equations that had more than two parameters. By experiments operated it was discovered that the LAGRANGE system could not handle noisy data that led to errors in the equations generated. Another system that was evolved based on the LAGRANGE system is the LAGRAMGE, where for this system the inputs from the user describe the likely solution. However because the solutions were based on the inputs it limited the process of the system, since if the input given could not produce a solution the system would not converge. Nevertheless, these systems were a good example of EDS and their future potential.

3.2. Nutonian Eureqa

According to information collected, the most accurate, advanced and widely used EDS is Nutonian Eureqa. Nutonian Eureqa does not have the same problems that the previous EDS has shown as it can manage noisy data and the user does not have to supply a specific grammar for the system and that is why it will be consider to be the model of what this project is trying to achieve.

Nutonian Eureqa applies symbolic regression instead of linear and non-linear regression to search the space of mathematical expressions to create a structure that of these expressions that fits the data. Symbolic regression does not attempt to create an equation of a specific structure

but tries to build an implicit equation that combines the primitive functions supplied but also changes the form of the equation. As mentioned in the genetic programming section, a genetic algorithm requires an initial population and in the case of symbolic regression the population is created by randomly merging mathematical expressions such as constants, mathematical operators, analytic functions and state variables (Schmidt and Lipson, 2009). The symbolic regression algorithm operates the same way a GPA does. As a result new equations are developed by applying natural selection on the current population and then applying genetic operators on them. The fitness function that is used calculates the mean squared error of the data that is provided with the data that gets calculated by the new equations so only the equations that are more accurate remain in the population. As previously stated a GA concludes its course based on a threshold that is provided by the user which is either a time limit or the number of iterations that it should execute or when median of the mean squared error of the population reaches a certain margin space. At this point, when Nutonian Eureqa terminates it provides a set of possible equations that solve the data supplied.

In the paper written from Schmidt and Lipson several ideas that were used to develop the algorithm that Nutonian Eureqa uses are discussed. One of this ideas mentioned is something that was also previously stated in the conclusion of the GP section, which is that based on the data given and the primitive functions, variables and constants provided by the user the system will find different solutions to the problem. As a result if the inputs supplied are not appropriate for the solution the algorithm is looking for, then the system might not be able to find the accurate solution and it will attempt to find an approximation of the solution. In the paper an example is given where they conducted an experiment and in the functions given they discarded the cosine and sine operations and provided data that could be represented with those operations and the solution the system gave was the Taylor series which is the closest approximation to the cosine function.

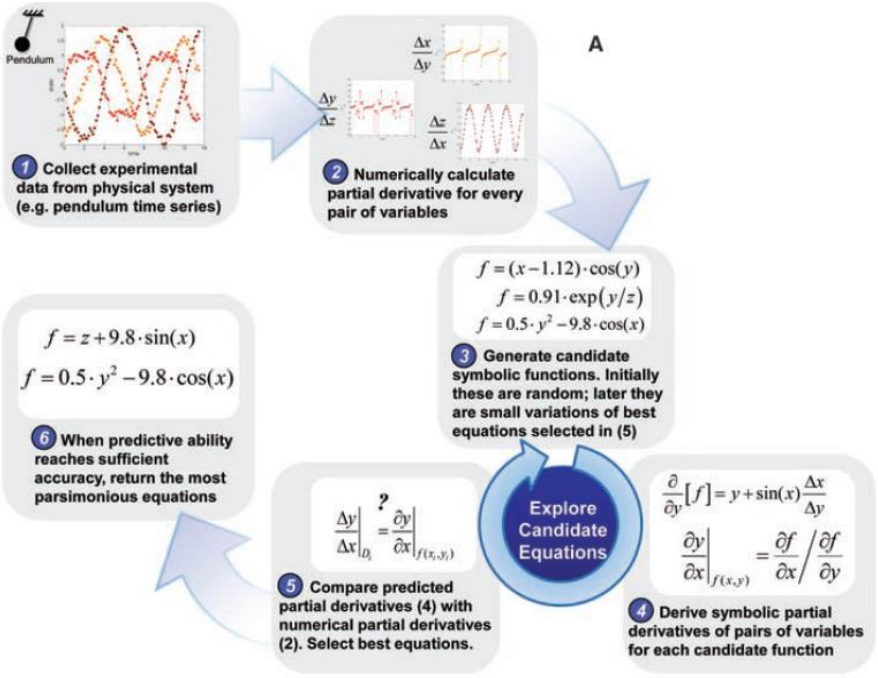


Fig. 6 Procedure of Nutonian Eureqa (Schmidt and Lipson, 2009)

A quote from Schmidt and Lipson “More precisely, the conservation equation should be able to predict connections among derivatives of groups of variables over time, relations that we can also readily calculate from new experimental data. One instance of such a metric is the partial derivatives between pairs of variables.” (Schmidt and Lipson, 2009). This signifies that the solutions found using this type for search criteria will provide be more accurate.

Another concept that Nutonian Eureqa utilizes and would be a useful tool for this project is that the system does not provide one single solution to the data but a set of solutions that fit the data quite accurately. This way the user has a number of solutions to choose from that differ from each other. Within Nutonian Eureqa the equations are picked based on an accuracy-parsimony Pareto front where the inverse of the number of parameters in the equation is considered to be the parsimony which represent how complex the equation is. Nutonian Eureqa provides solution that retain a balance between their complexity and their accuracy.

4. Conclusion

Included in this research review is a number of fields relevant to the implementation of this project. Initially the aim of this project was introduced and the motivation behind it. In conclusion in this part of the research review there will be a summary of what was mentioned and how it is relevant with the implementation of the project.

Considering how fast the world evolves and how technology flourishes, more and more data are generated and researches aspire to find the dynamics of them so as to represent something coherent. Therefore many systems were manufactured to attempt to find patterns and relationships within data and so the field of machine learning was developed.

Based on the information stated in Machine Learning section and the problem this project is going to solve, this project is considered as unsupervised learning as the system will search by itself to find connections between the provided functions to solve the problem. To create this system symbolic regression will be used as it is currently the most appropriate tool to solve differential equations and because it does not constraint the design of the solutions in one specific format.

Symbolic regression employs the functionality of genetic algorithms and genetic programming thus the basic information of how a GA performs were introduced and some details on what problems GAs can solve. An important aspect that this project wishes to use is the parallelization of GP that can be applied that will increase the speed the system finds solutions. Additionally, some characteristics of the most advanced equation discovery system were brought up with the ambition of integrating them in the software that will be developed through this project.

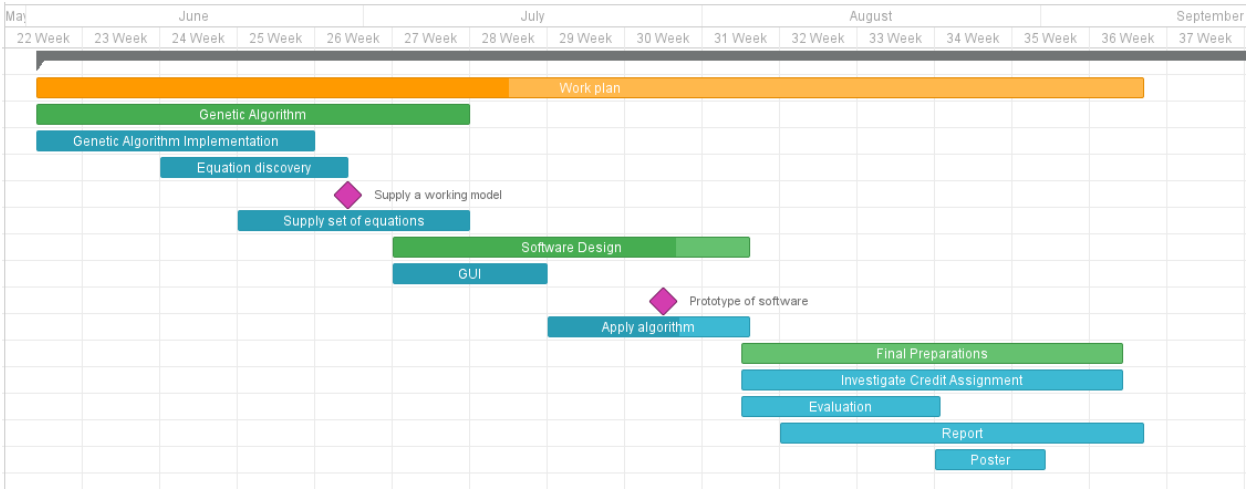
In conclusion this project aims to improve finding mathematical formulae that represent data and specifically data that describe the emigration process of the *Temnothorax albipennis* ants and all the areas disclosed correlate with accomplishing this project.

5. Work Plan

The work plan has been divided into two parts. The first part is a Gantt chart which describes the expected timeline and the time periods of each task for the project. The second part provides a description of the activities of the tasks and milestones contained within the Gantt chart.

5.1. Expected Timeline

	Task name	Start date	End date
	Project Develpomemt	01/06/16	09/09/16
1	Work plan	01/06/16	09/09/16
1.1	Genetic Algorithm	01/06/16	11/07/16
1.1.1	Genetic Algorithm Impleme	01/06/16	27/06/16
1.1.2	Equation discovery	13/06/16	30/06/16
1.1.3	Supply a working model	30/06/16	30/06/16
1.1.4	Supply set of equations	20/06/16	11/07/16
1.2	Software Design	04/07/16	05/08/16
1.2.1	GUI	04/07/16	18/07/16
1.2.2	Prototype of software	28/07/16	28/07/16
1.2.3	Apply algorithm	18/07/16	05/08/16
1.3	Final Preparations	04/08/16	08/09/16
1.3.1	Investigate Credit Assignm	04/08/16	08/09/16
1.3.2	Evaluation	04/08/16	22/08/16
1.4	Report	08/08/16	09/09/16
1.5	Poster	22/08/16	31/08/16



5.2. Description of Tasks

1. Genetic Algorithm

1.1.1 Genetic Algorithm Implementation

During this tasks, the examination of many different genetic algorithms will be directed so as to try and figure out which will the most beneficial for our project. This stage will involve finding which fitness function to use and all the parameters that are needed to create a genetic algorithm.

1.1.2 Equation discovery System

After the decision of the GA is finalized, the researcher will try to create one that can solve equations based on the data given. Firstly, we will try to solve equations that are very simple such as linear equations and then move on to more complex equations like differential equations which is our goal.

Supply a working Model - Milestone: At this point an algorithm should have been developed that if given certain data will be able to solve a number of equations and it's in the process to supply more than one.

1.1.4. Supply set of equations

The next step will be to develop a system based on the algorithm, in order to enable it to solve more than one equation each time and the solution of each equation to be depended on the development of the other.

1.2. Software design

1.2.1. GUI

At this point, the programming language to use will be decided based on all the capacities needed to create the software. The languages which that are considered are Mat Lab, C programming, R and JAVA. The programing language should offer genetic algorithms libraries and other options that will come in mind later on. The features that this software demands are straightforward as the working of it is mostly based on just input and output.

1.2. Apply algorithm

This stage concerns the design the implementation of the algorithm created so far. The program should be competent that it takes data from the user and the options needed for the algorithm to run and then supply the equations formed on the data.

Prototype of software - Milestone: This milestone will be reached once a piece of software integrates the genetic algorithm or even just takes the data input and has an interface.

1.3. Final Preparations

1.3.1. Investigate Credit Assignment

This task includes the investigation of any errors in the system and finding the importance of each one and how it affects the output of the system and if it can be fixed.

1.3.2. Evaluation

At this point, there will be an evaluation of the system. The system will be given data on different mathematical expressions and the output will then be evaluated by comparing it with the outputs that other systems offered with the same data and also evaluate it by calculating the margin error of each expression. For example, the system will receive data that represent a specific equation and then the expected output of the system will be an equation that can predict the same data and a mean squared error can be calculated between the data given by the new system and the data that are given from the correct equation.

1.4. Report

After all the data is collected from the credit assignment and the evaluation then writing the thesis is the next task. This is the final step of the project and it includes writing in more detail what techniques were used, what were the results of the evaluation and the credit assignment. Also, it will also include the genetic algorithm that will be created and the process of developing it.

6. Risk Analysis and Contingency Plan

In this section of the research review, the risks that might occur in the development of the project and affect the completion of t are listed. Included in this list is an analysis of their likelihood, their severity and the preventative measures that will be taken to either stop them from occurring or to resolve them if they do happen.

6.1. Limited Time / Operational Feasibility

Having no background in machine learning, it may take longer time to implement the algorithms needed to create the equation discovery model. I do not have previous knowledge in genetic algorithms or genetic programming so that might bring the development of the expected equation discovery model to a halt.

Contingency plan: Use the non-allocated time to catch up to any obstacles that might occur and consult with an expert if possible to receive advice.

Severity: Medium

Likelihood: 50%

6.2. Illness

I might get sick during the development of the project or for some unforeseen reasons I might be unable to work for a period of time and that will affect my progress.

Contingency plan: I have left small periods of time in the work plan as empty just in case I get sick and just to be more flexible with the work plan. Anything more serious that will take a more extensive period of time would be deem by the university as an exception.

Severity: Low

Likelihood: 20%

6.3. Impossibility in creating a discovery equation model for a set of equations

This is the risk with the highest impact on the project. Without starting the process of creating the genetic algorithm, that will be the base of the system, it is difficult to say the likelihood of this occurring.

Contingency plan: The focus of the project will alter in performing credit assignment in the equation discovery model that will be developed at that point.

Severity: High

Likelihood: 40%

7. Bibliography

[1] - Affenzeller, M. (2009). Genetic algorithms and genetic programming. Boca Raton: CRC Press.

[2] - Brameier, M. and Banzhaf, W. (2007). Linear genetic programming. New York: Springer.

[3] - Computational Ant Lab. (n.d.). Computational Ant Lab. [online] Available at: https://sites.google.com/site/computationalantlab/home [Accessed 3 May 2016].

[4] - De Jong, K. (1993). Genetic algorithms are NOT function optimizers. Foundations of genetic algorithms, 2, pp.5-17.

[5] - Discoverlife.org. (2016). Temnothorax albipennis (Curtis, 1854) -- Discover Life. [online] Available at: http://www.discoverlife.org/mp/20q?search=Temnothorax+albipennis [Accessed 3 May 2016].

[6] - Eiben, A. and Smith, J. (2003). Introduction to evolutionary computing. New York: Springer.

[7] - Equation discovery - Introduction. (2000). Equation discovery - Introduction. Available at: http://www-ai.ijs.si/~ljupco/ed/ [Accessed 10 May 2016].

[8] - Folino, G., Pizzuti, C. and Spezzano, G. (2002). Improving Induction Decision Trees with Parallel Genetic Programming. In: In Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop, pp.181-187.

[9] - Forrest, S. (1993). Genetic algorithms: principles of natural selection applied to computation. Science, 261(5123), pp.872-878.

[10] - GBIF.ORG. (2016). R: a language and environment for statistical computing. Available at: http://www.gbif.org/resource/81287 [Accessed 3 May 2016].

[11] - Goldberg, D. and Holland, J. (1988). Genetic Algorithms and Machine Learning. Machine Learning, 3(2/3), pp.95-99.

[12] - Gorges-Schleuter, M. (1990). Explicit parallelism of genetic algorithms through population structures. Parallel Problem Solving from Nature, Volume 496, pp.150-159.

[13] - Grefenstette, J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. Mach Learn, 3(2-3), pp.225-245.

[14] - Jantke, K. and Shinohara, A. (2001). Discovery science. Berlin: Springer, pp.390-398.

[15] - Koza, J. (1992). Genetic programming. Cambridge, Mass.: MIT Press.

[16] - Miller, J., Tomassini, M., Lanzi, P., Ryan, C., Tettamanzi, A. and Langdon, W. (2001). Genetic Programming: 4th European Conference. In: Genetic Programming: 4th European Conference. EuroGP 2001 Lake Como, Italy: Springer.

[17] - Mitchell, T. (1997). Machine Learning. New York: McGraw-Hill.

[18] - Poli, R., Langdon, W. and McPhee, N. (2008). A Field Guide to Genetic Programming. [S.L.]: Lulu Press (lulu.com), 2008.

[19] - Raedt, L. and Flach, P (Eds.). (2001). Machine learning. Berlin: Springer-Verlag, pp.478-490.

[20] - Schmidt, M. and Lipson, H. (2009). Distilling Free-Form Natural Laws from Experimental Data. Science, 324(5923), pp.81-85.

[21] - Shonkwiler, R. (1993). Parallel Genetic Algorithms. ICGA, pp.199-205.

[22] - SWATI, (2015). PARALLEL GENETIC ALGORITHM. IARJSET - Science, Engineering and Technology, 03(06), pp.270-276.

[23] - Todorovski, L. and Džeroski, S. (n.d.). *Using domain knowledge on population dynamics modeling for equation discovery*.

8. Appendix

[1] - Equations that describe the behaviour of ants discovered by different models.
Taken from: Collerton, J. (2015). *Learning Population Dynamics in Ant Colonies*. Postgraduate. University of Bristol.

- Pratt Model

$$\frac{dS}{dt} = -\sum_{j=1}^{M} \mu_j S - \sum_{j=1}^{M} \lambda_j I(R_j, S)$$

$$\frac{dA_i}{dt} = \mu_i S + \lambda_\iota I(R_i, S) - \sum_{j \neq i} (p_{ji} A_j - p_{ij} A_i) - k_i A_i$$

$$\frac{dR_i}{dt} = k_i A_i + \sum_{j \neq i} (p_{ji} R_j - p_{ij} R_i)$$

$$I(R_i, S) = \begin{cases} R_i & if\ R_i < T\ and\ S > 0 \\ 0 & otherwise \end{cases}$$

$$\frac{dP_i}{dt} = \varphi_i . J(R_i, P_0)$$

$$J(R_i, P_0) = \begin{cases} 0 & if\ R_i < T\ or\ P_0 = 0 \\ R_i & otherwise \end{cases}$$

Parameter	Definition
N	Colony population.
p	Proportion of colony population of consisting of active ants.
λ	Rate of recruitment via tandem runs, per ant.
φ_i	Rate of recruitment via transports, per ant at site i.
μ	Rate of discovery of new sites, per ant, per site.
k_1	Probability per min that an assessor at site 1 begins to recruit.
k_2	Probability per min that an assessor at site 2 begins to recruit.
p_{12}	Rate of switching allegiance from site 1 to site 2 per ant.
c	Time to move an item from site 1 to site 2 after old nest is empty.
M	The number of sites of varying quality
P_0	The number of passive items remaining in the old nest.
T	recruiters

8. Appendix

- Planqué Model

$$\frac{dA}{dt} = \mu A + l(\lambda, R, Q, A)$$

$$\frac{dS}{dt} = \mu A - kS - fr(\lambda, R, Q, S)$$

$$\frac{dR}{dt} = kS + l(\lambda, R, Q, A) + fr(\lambda, R, Q, S)$$

$$\frac{dP}{dt} = -(1-f)c(\varphi, R, Q, P)$$

$$\frac{dC}{dt} = (1-f)c(\varphi, R, Q, P)$$

$$l(\lambda, R, Q, A) = \begin{cases} \lambda \dfrac{RA}{R+A} & if\ R < Q \\ 0 & otherwise \end{cases}$$

$$c(\varphi, R, Q, P) = \begin{cases} \varphi \dfrac{RP}{R+P} & if\ R \geq Q \\ 0 & otherwise \end{cases}$$

$$r(\lambda, R, Q, A) = \begin{cases} \lambda \dfrac{RA}{R+A} & if\ R \geq Q \\ 0 & otherwise \end{cases}$$

Parameter	Definition
N	Colony population.
F	Fraction of active ants.
Q	Colony quorum Threshold.
f	Fraction of post-quorum reverse tandem running time.
φ	Rate at which passive ants are carried to a new nest.
μ	Rate of discovery of new sites, per ant, per site.
λ	Rate at which ants following tandem runs become recruiters.
A	The number of active ants in the old nest.
S	The number of scouting ants in the old nest.
R	The number of recruiting ants in the old nest.
P	The number of passive ants in the old nest.
C	The number of carried ants in the old nest.

8. Appendix

- Nutonian Eureqa

Original Function	Eureqa Derived Function
$\frac{dS}{dt} = -2\mu S - [\begin{cases} \lambda R_1 & if\ R_1 < T\ and\ S > 0 \\ 0 & otherwise \end{cases} + \begin{cases} \lambda R_2 & if\ R_2 < T\ and\ S > 0 \\ 0 & otherwise \end{cases}]$	$\frac{dS}{dt} = -2\mu S - [1.02 \begin{cases} \lambda R1 & R_1 < T\ and\ S > 0 \\ 0 & otherwise \end{cases} + 1.06 \begin{cases} \lambda R_2 & R_2 < T\ and\ S > 0 \\ 0 & otherwise \end{cases}]$
$\frac{dA_1}{dt} = \mu S + \begin{cases} \lambda R_1 & if\ R_1 < T\ and\ S > 0 \\ 0 & otherwise \end{cases} - p_{12}A_1 - k_1A_1$	$\frac{dA_1}{dt} = \mu S - A_1 \begin{cases} p_{12} & \\ p_{12} - S\mu[\![P_1 > 0]\!] - 0.62(k_1\lambda t)[\![S > 0]\!] & Q + k_1 - A_1k_1 \leq R1 \\ & Q + k_1 - A_1k_1 > R1 \end{cases} - k_1A_1$
$\frac{dA_2}{dt} = \mu S + \begin{cases} \lambda R_2 & if\ R_2 < T\ and\ S > 0 \\ 0 & otherwise \end{cases} + p_{12}A_1 - k_2A_2$	$\frac{dA_2}{dt} = \mu S + \begin{cases} 1.04\lambda R_2 + \lambda^2 & [P_2 < 0.001S] \\ 0 & otherwise \end{cases} + p_{12}A_1 - 1.003A_2k_2 + p_{12}[\![p_{12} > P_2]\!]$
$\frac{dR_1}{dt} = k_1A_1 - p_{12}R_1$	$\frac{dR_1}{dt} = k_1A_1 - p_{12}R_1$
$\frac{dR_2}{dt} = k_2A_2 + p_{12}R_1$	$\frac{dR_2}{dt} = k_2A_2 + p_{12}R_1$
$\frac{dP_1}{dt} = \phi \begin{cases} 0 & if\ R_1 < T\ or\ P_0 = 0 \\ R_1 & otherwise \end{cases}$	$\frac{dP_1}{dt} = \phi \begin{cases} R_1 & if\ R_1 \geq T\ or\ P_0 > 0 \\ 0 & otherwise \end{cases}$
$\frac{dP_2}{dt} = \phi \begin{cases} 0 & if\ R_2 < T\ or\ P_0 = 0 \\ R_2 & otherwise \end{cases}$	$\frac{dP_2}{dt} = \phi \begin{cases} R_2 & if\ R_2 \geq T\ or\ P_0 > 0 \\ 0 & otherwise \end{cases}$