Genetic algorithms as a mechanism for modelling gene interactions using time-course data

Algoritmos genéticos como mecanismo para modelar interacciones genéticas utilizando información observada en distintos tiempos

David J. John[1], Kenneth David Meza-Chaves[2]

John, J. D.; Meza-Chaves, K. D. Genetic algorithms as a mechanism for modelling gene interactions using time-course data. *Tecnología en Marcha*. Vol. 31 - Número Especial Movilidad Estudiantil 5. Octubre 2018. Pág 69-87

DOI: https://doi.org/10.18845/tm.v31.i5.4087

1 American. Professor at Wake Forest University, Department of Computer Science. Winston-Salem, North Carolina, United States of America.
Email: djj@wfu.edu

2 Costa Rican, Computing Engineering Student, Tecnológico de Costa Rica, School of Computing. Cartago, Cartago, Costa Rica.
Email: keme0409@gmail.com

Keywords

Gene interaction model; genetic algorithm; Bayesian likelihood; directed acyclic graph.

Abstract

Gene interaction models are weighted graphs derived from replicates of gene abundance time-course data, where each weighted edge is a probability of gene association. These interaction models are a tool to assist biological researchers in understanding gene relationships. Two new genetic algorithms, one fairly traditional and the other based on crossover with infrequent application of a chaotic mutation operator, are developed specifically to produce gene interaction models from sparse time-course abundance data. Both genetic algorithms evolve a new population from a current population of directed acyclic graphs, each representing a Bayesian model for possible gene interaction. The genetic algorithm fitness is the relative posterior probability that a Bayesian model fits the gene abundance replicates. These Bayesian likelihoods are computed using one of three analysis techniques: cotemporal, first order next state and second order next state. The weighted gene interaction models reflect the directed acyclic graphs and their likelihoods present in the final populations of numerous independent genetic algorithm executions. Using a simulated set of genes, these two genetic algorithms find the embedded signals and are consistent across analysis paradigms. Results from a set of biological gene abundance data, from *Arabidopsis thaliana* stimulated by the plant hormone auxin, are modeled.

Palabras clave

Modelo de interacción genética; algoritmos genéticos; probabilidades Bayesianas; grafo acíclico dirigido.

Resumen

Los modelos de interacción genética corresponden a grafos ponderados que se generan a partir de replicaciones de abundancia genética en distintos tiempos de observación, en donde cada arista con peso es una probabilidad de asociación genética. Estos modelos de interacciones son herramientas empleadas por los investigadores en biología para comprender las relaciones genéticas. Dos algoritmos genéticos fueron desarrollados para modelar interacciones genéticas a partir de información de entrada obtenida en experimentos científicos (abundancia genética), en donde un algoritmo sigue un esquema tradicional, mientras que el otro (no tradicional) se basa en la etapa de 'crossover', con la aplicación de la mutación de manera poco frecuente. Ambos algoritmos genéticos evolucionan una población actual conformada por grafos acíclicos dirigidos para producir una nueva población, en donde cada grafo representa un modelo Bayesiano para una posible interacción genética. El 'fitness' empleado en los algoritmos genéticos consiste en la probabilidad relativa posterior en la que un modelo Bayesiano se ajusta a las replicaciones de abundancia genética. Estas probabilidades Bayesianas son calculadas utilizando una de las tres técnicas de análisis: cotemporal, estado siguiente del primer orden y estado siguiente del segundo orden. Los modelos de interacción genética ponderados reflejan los grafos acíclicos dirigidos y sus probabilidades presentados en la última población de cada una de las numerosas ejecuciones independientes del algoritmo genético. Empleando un set de genes simulado, ambos algoritmos genéticos encuentran las señales que se han diseñado y se mantienen consistentes entre paradigmas. También, se presentan modelos empleando un set de datos de la abundancia genética obtenida a partir del estudio de la planta *Arabidopsis thaliana* estimulada por la hormona auxina.

Introduction

Deoxyribonucleic acid (DNA), discovered in 1869 by the scientist Friedrich Miescher, is a molecule that resides mostly in the cell nucleus and contains the genetic information used to create proteins, which are essential for all the biochemical functions that occur in living organisms (Miescher, 1871). The DNA is composed of genes which contain a particular set of instructions to create a protein. In order to make a protein, a process, gene transcription, copies each of the chemical bases into messenger RNA (ribonucleic acid), mRNA. The mRNA moves out of the nucleus and uses cell organelles in the cytoplasm called ribosomes to form an amino acid that finally folds and configures to form the protein. The genes involved in gene transcription interact with each other to produce the protein. The amount of mRNA at a certain time is known as transcript abundance (or protein or metabolite abundance) and is associated with gene activity. Scientists are interested in discovering how the genes interact in the process of creating a protein. Predicting gene interactions can help scientists to discover associations involved in biochemical systems [22], [8].

Many research projects have focused on the development of appropriate mathematics and statistics to describe gene interaction and on the application of computational tools to effectively search for gene interaction models [4], [9], [25], [20]. This work builds upon the approach developed using Bayesian posterior probability and a Metropolis-Hastings search algorithm [20]. From several published papers [20], [19], [21], their methods often predicted true interactions with high probabilities. However, the computation basis, Metropolis-Hastings search, was very time intensive and limited the number of genes that could be analyzed.

The main focus of this paper is to discuss a new computational technique, based on genetic algorithms (GA), as an improvement of the Metropolis-Hastings (MH) approach. This work is a collaboration between Wake Forest University and Tecnológico de Costa Rica. The main reasons to develop this new approach is to significantly improve the time required to produce interaction models, and to support scaling up the number of genes. One of our main goals is to more thoroughly visit the high likelihood portions of the search space so that more accurate posterior probabilities can be found. For analyzing 12 genes together, the MH approach has about a 3-week runtime, while the GA only requires several hours.

Gene Interaction Models

A graph is a mathematical representation of objects connected by some relationship. These objects are better known as nodes and their connections are called edges. There are many types of graphs depending on the context. The edges can be weighted or not. In this project, three different types of graphs are used: the weighted directed graph, the weighted undirected graph and the directed acyclic graph (DAG) [29], [2]. In a weighted directed graph, the edges are directed from one vertex to another, and the edge weight represents the strength of the directed connection. In a weighted undirected graph, the edges are between vertices (with no preferred direction) and the edge weight indicates the strength of the undirected connection. A directed acyclic graph is a directed graph with the additional property that for each vertex there is no non-empty sequence of directed edges leaving and returning to the vertex, i.e. there are no cycles.

Given the genes g_1,…,g_k, a DAG with nodes labelled with g_1,…,g_k is a model for the interaction of these genes. The DAG shown in figure 1 is an interaction model, where every relationship represents the influence between the genes. The directed edges of the DAG capture the notion of one of three gene relationships: cotemporal, first order next state, or second order next state. The cotemporal relationship indicates that the gene abundance satisfies some

identity at each time step. The first order next state relationship indicates the gene abundance are strongly related from one-time step to the next. The second order next state indicates that the gene abundance of the first gene at times t-1 and t are strongly related to the gene abundance of the second gene at time t+1.

An important question is how likely a particular DAG fits the replicate abundance data, L(DAG|R_1,…,R_k), i.e. how well does the DAG reflect the abundance measurements. Of course, such a likelihood would depend upon the underlying analysis paradigm (cotemporal, first order next state, or second order next state) applied to the DAG given the time-course gene abundance data (mRNA measurements at a small number of time points). Closed forms for each of these three paradigms have been developed, applied and studied [20], [19], [21].

Examining all possible DAGs to discover exact posterior probabilities of all gene-to-gene interactions is infeasible; the space of all possible DAGs has doubly exponential size as a function of the number of genes [20]. Therefore, heuristic search algorithms must be employed. A further restriction for the DAGs considered in this research is that no child can have more than three parents.

The final product of the analysis is an amalgamated model (weighted graph). The amalgamated model is constructed from a number of composite models (weighted graphs). Each composite model is a reflection of a large number of DAGs. An effective and fast technique which allows efficient sampling of a large DAG space is a genetic algorithm.

Genetic algorithms

The main algorithmic search technique utilized in this novel approach is the genetic algorithm (GA). These are used for heuristically solving difficult search and optimization problems. Basically, the motivation of a GA is to naively mimic natural genetic operations. GAs are stochastic algorithms which embrace the concepts of the genetic inheritance and Darwinian strife for survival [15], [5].

The genetic algorithm vocabulary is influenced by natural genetics. In general terms, a main concept is the individual, or chromosome, that belongs to a population. Every chromosome consists of genes, and every gene represents a characteristic that may be inherited. To avoid confusion, we refer to the individuals (rather than chromosomes) in the populations.

Every individual represents a possible “solution”. The current population of individuals corresponds to a selected sample in the space of potential solutions. To facilitate the search for better solutions, the GA exploits the individuals of the current population with the crossover operator and it explores the search space by applying a mutation operator to current individuals. For this reason, GAs have been quite successfully applied in many arenas [12], [15], [5].

A genetic algorithm starts with an initial population, of fixed size. The basic idea is to iteratively evolve a new population of individuals from the current population. Fundamentally important is a function that compares one individual to another, the fitness function. Ideally, one individual is more fit than another if it represents a better solution (in our case, higher likelihood). The new population is formed by selecting parents from the current population (with selection rates proportional to fitness), exchanging genetic information via the crossover operator yielding children, and finally mutating the children. Crossover, governed by the crossover rate, combines the features from the two parents to form two offspring. Mutation, subject to the mutation rate, arbitrarily alters one or more features of a selected child which introduces extra variability into the new population. As well, the elitism operator can be applied to advance a number of superior individuals from the current population to the new population.

Genetic algorithms excel at finding good solutions for a problem. Typically each generation is obtained in a polynomial algorithm time, as well most space requirements are modest.

Traditional genetic algorithm

First, a traditional genetic algorithm is developed to create weighted gene interactions models from multiple sets of gene abundance data. Every individual in a population is an adjacency matrix, which represents a directed acyclic graph (DAG), and the relative likelihood, L(DAG|R_1,…,R_k). In mathematical terms, an adjacency matrix is a square matrix used to represent a graph, where an entry '1' in the (i,j) cell implies the presence of a directed edge from gene g_i to gene g_j. An entry '0' corresponds to the lack of a corresponding directed edge. Figure 1 illustrates this idea.

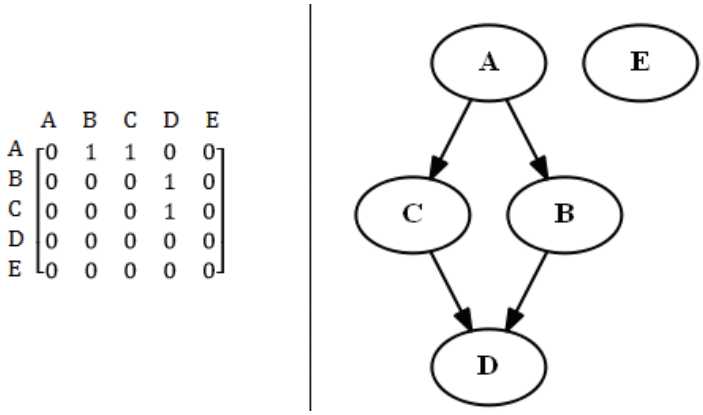

Figure 1. The left panel is an adjacency matrix representing the DAG in the right panel. For example, the '1' entry on row A and column B corresponds to the presence of the directed edge A→B in the DAG. This directed edge indicates that gene A influences the gene B. Furthermore, A influences D indirectly since the edge B→D is present in the DAG.

When creating the initial population, three phases are applied to create the individuals, as seen in figure 2. First, all the individuals are randomly created. Next, six steps identify and keep the top individuals, then replace all the others. Each step in this phase identifies an increasing number of top individuals. Last, three steps keep the majority of top individuals and then mutate the remaining. The purpose of this initialization is to create a starting population where the individuals are of high fitness and the population overall is diverse. Using only random assignment, often the GAs were not efficient in their search.

Given the current population, the GA proceeds to create the new population using selection, crossover and mutation. Before starting selection, an elitist strategy is applied. The top ten percent of the individuals in the current population are inserted directly into the new population.

```
/* INITIAL POPULATION GENERATOR */
N = population size
FUNCTION initial_population_generator
BEGIN
      Randomly create N DAGs, with computed likelihoods, creating POP

      phase_01_percentages = [15, 30, 45, 60, 75, 90]
      FOR PERCENTAGE IN phase_01_percentages DO
            Keep the top PERCENTAGE of unique individuals in CPOP
            Complete the rest of CPOP randomly
            Compute the likelihoods
      ENDFOR

      phase_02_percentages = [60, 70, 80]
      FOR PERCENTAGE IN phase_02_percentages DO
            Keep the top PERCENTAGE of unique individuals in CPOP
            Complete the rest of CPOP, mutating random individuals
            Compute the likelihoods
      ENDFOR
END
```

Figure 2. Pseudocode with the steps for creating the initial population.

Ranked selection is used to choose parents [15]. First, the current population is sorted based on their likelihoods. Then, every individual receives a fitness value based on a uniform and linear mapping of these sorted likelihoods into the intervals between 0.0 and 1.0. The worst ranked individual will have the smallest fitness value, and the best will have the largest fitness. Ranked selection assures that all the individuals have a reasonable chance to be selected. By this, the best individual will have the best chance to be selected but the worst could have a reasonable opportunity to reproduce as well. After this process, based on the fitness based probability, a pair of parents are selected, with replacement.

Governed by the probability of crossover (P_x=0.05), both chosen parents could become the two new offspring or the crossover operator is applied to the parents to produce two new offspring. For the crossover operator, a column in the adjacency matrices is chosen arbitrarily, and then the entries in those columns are exchanged, creating two new individuals. The new individuals are added to the new population. For example, having two parents (A and B), as shown in figure 3, and selecting two arbitrary selected columns, column C and column D, both columns get exchanged to produce two different children. The new children are included in the new population.

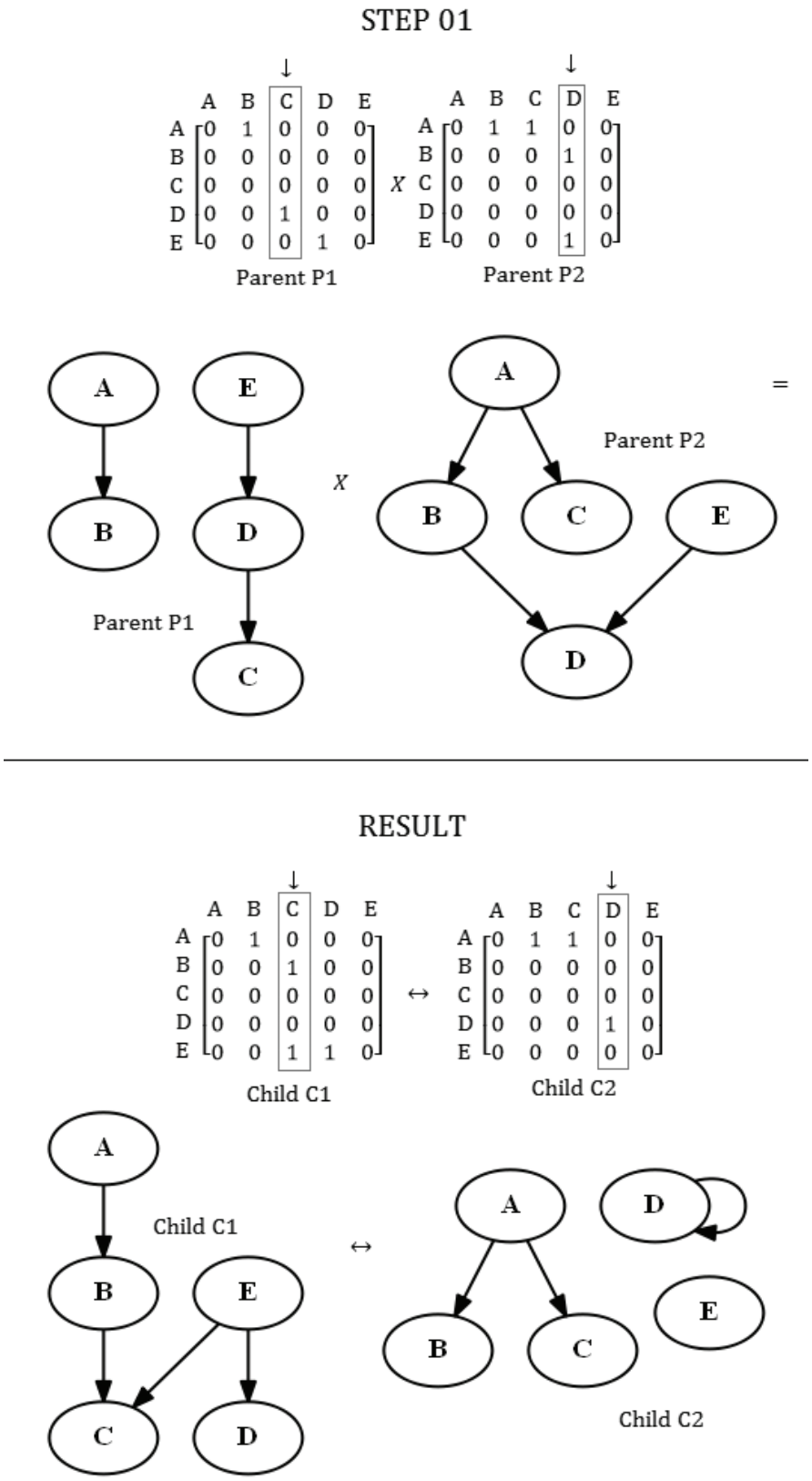

Figure 3. In the image above two parents (A and B) that are going to reproduce, swapping the columns C and D respectively. In the image below, the result of this crossover.

The next mechanism to apply is mutation. Based on probability of mutation (P_m=0.3), this process changes values of randomly selected entries in the adjacency matrix. The affected changes either create new edges or delete existing edges. The figure 4 illustrates this process.

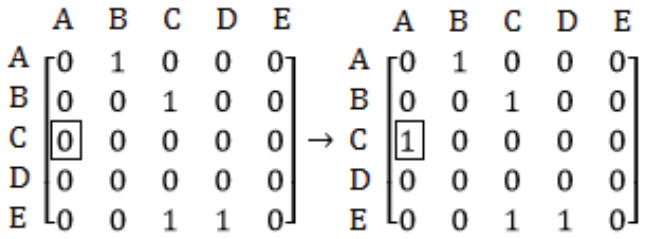

Figure 4. An example of the mutation, changing a random entry in the matrix. A new edge, C→A, is added.

After the application of crossover and mutation, some individuals in the new population may no longer be a DAG, cycles may have been introduced by either crossover or mutation. Because of this, an additional function for repairing the individual with cycles is required. Essentially the repair function iteratively identifies all cycles in the directed graph and removes an edge in the largest cycle until no cycles remain. This is done very efficiently using Tarjan's algorithm [27]. The repair algorithm is shown in figure 5. For example, in the figure 4, after applying the mutation we don't have an individual with the structure of a DAG, so it needs to be fixed (figure 6).

```
/* DAG REPAIRING FUNCTION*/
FUNCTION repairing_function
PARAMETERS
      -  Population to be repaired (POP)
BEGIN
      FOR INDIVIDUAL IN POP DO
      BEGIN
            Break cycles found on the diagonal of the adjacency matrix of
            the INDIVIDUAL

            FOR COLUMN IN INDIVIDUAL DO
            BEGIN
                  count_edges = number of ones in the COLUMN
                  WHILE count_edges > 3 DO
                        Delete random edge in the COLUMN
                  ENDWHILE
            ENDFOR

            Break edges that appear in the most cycles of INDIVIDUAL
            until no more cycles are found
      ENDFOR
END
```

Figure 5. Pseudocode for the repairing operator to eliminate self-loops, enforce the restriction of at most three parents, and break cycles.

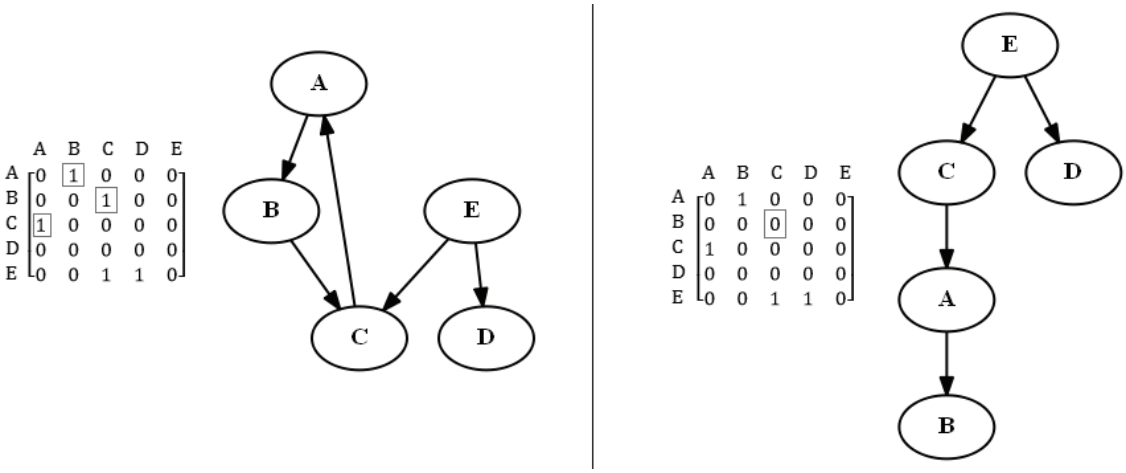

Figure 6. In the left panel, the adjacency matrix for the directed graph that contains a cycle, A→B,B→Cand, with its graph representation. On the right, the same adjacency matrix after applying the repair operator, breaking the edge B→C, and the repaired DAG.

When all the previous functions are applied, a new population results and we are ready to start a new iteration. The total number of iterations, the population size, the crossover and mutation rates are determined a priori. The last individuals of the final generation create a composite model (described on a later section). The figure 7 shows the pseudocode for the algorithm.

```
/* TRADITIONAL GENETIC ALGORITHM */
N = population size
M = amount of generations
BEGIN
      Generate the initial population (POP)
      Calculate the likelihood of every individual
      Sort the population based on the likelihood
      Compute the fitness function on POP

      Set cont = 0
      WHILE cont < M DO
            Put top 10% of individuals in CPOP (Elitism)
            WHILE LENGTH(CPOP) < N DO
                  Choose parents P1 and P2 from POP (Ranked Selection)
                  IF probability of selection is matched DO
                        Perform the Crossover, the two new offspring will
                        be inserted in the children population (CPOP)
                  ELSE
                        The two new offspring will be P1 and P2 and are
                        inserted in the CPOP
                  ENDIF
            ENDWHILE

            Set POP = CPOP
            Perform the Mutation to POP
            Perform the Repairing to POP
            Calculate the likelihood of every individual
            Sort of the population based on the likelihood
            Compute the fitness function on POP

            Set cont = cont + 1
      ENDWHILE
END
```

Figure 7. Pseudocode of the traditional genetic algorithm.

CHC genetic algorithm

A non-traditional genetic algorithm, the CHC genetic algorithm (CHC GA) [3], is implemented specifically for finding gene interaction models. There are several significant differences between this CHC GA and the traditional genetic algorithm discussed previously. First, selection is not based on the fitness function. Second, the CHC GA has a deliberate mechanism to avoid crossover of very similar individuals. Third, the crossover operator is half uniform crossover (HUX), which is a very disruptive form of crossover. Finally, mutation is not regularly applied. Instead, there is a procedure to detect when the population has lost its diversity, which then intensively uses mutation to reinitialize the current population and afterwards continues [28].

There are some concepts that are shared between the CHC and the traditional genetic algorithms. For example, every individual in a population is an adjacency matrix for a DAG and its likelihood. The initial population, the likelihood value calculation and the DAG repairing methods are used exactly as in the traditional genetic algorithm implementation. Every population of individuals has a fixed size.

Selection of the parent pairs is performed randomly from the current population without replacement. Every individual in the current population has an equal opportunity to produce a child. This selection technique is not dependent on the likelihood of the individuals.

The crossover operator computes the Hamming distance[3] between the two parents. A threshold is established, equal to one quarter of the number of significant entries in an adjacency matrix. If the Hamming distance does not exceed the threshold then the parents do not produce the offspring. The algorithm avoids crossover between very similar individuals.

Otherwise, the parents use the HUX crossover operator to produce two children. On this type of crossover, exactly half of the nonmatching entries are swapped [1]. If no children are generated the threshold is reduced and a new pairing of parents begins. When all the children are generated, instead of replacing the current population with the children, the newly created children must compete with the members of the current population for survival, cross-generational competition. More specifically, the parents and children are merged and ranked according to the likelihood values and then the new population is created in an elitist way by selecting the highest ranked individuals to fill the new population. When the threshold becomes negative, the mutation operator is used to reinitialize the current population. In this case, the population is reinitialized and the cycle is repeated, by using the individual with the highest likelihood for creating the replacement population. The new individuals are created by mutating the best individual randomly. The best individual is also added unchanged to this new population, insuring that the next round of crossover cannot converge to a less likelihood.

An example of HUX crossover is seen in figure 8, where the two parents, A and B, who are going to be crossed. In the first step, the entries where the two parents differ are marked and as we can see they differ on four entries. In the second step, we need to randomly choose half of those differing entries in other to do the crossover. In the image, those entries are marked by green squares. The result finally is going to be two different offspring, with the selected entries swapped from their parents.

3 The Hamming distance between two adjacency matrices of equal size refers to the number of entries in which the corresponding symbols (in this case '0' or '1') are different.

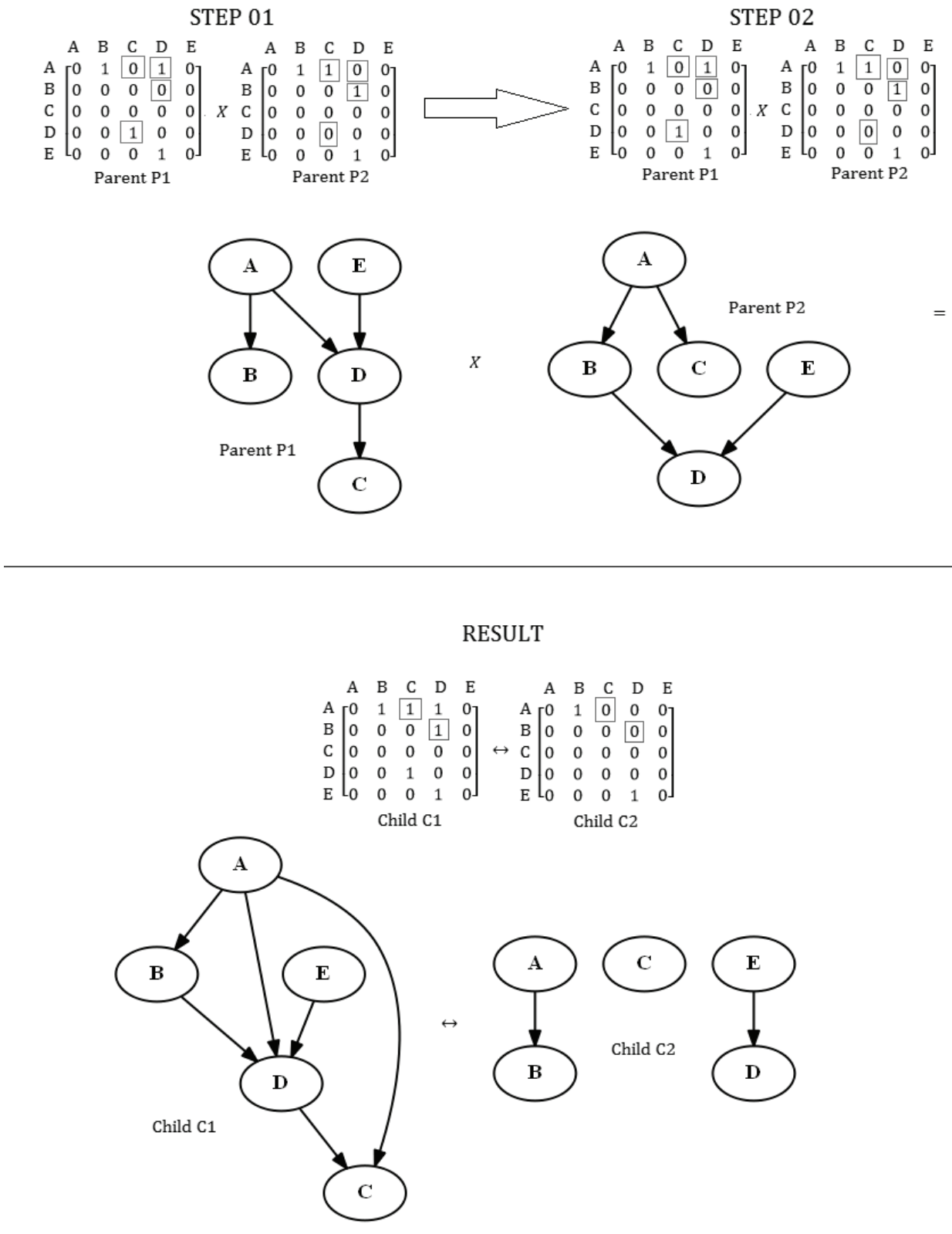

Figure 8. An example of HUX crossover in CHC genetic algorithm. The first step search for the bits where the two parents differ (marked in red). The second step, randomly choose half of these bits to be swapped (marked in green). In the image below, the result of this crossover.

The CHC genetic algorithm only uses mutation when the population has stagnated. When the threshold value drops to zero there have been several generations without any new offspring accepted into the population. Figure 9 illustrates the CHC implementation.

```
/* CHC GENETIC ALGORITHM */
L = length of individual = rows * columns
N = population size
M = amount of generations
BEGIN
      Generate the initial population (POP)
      Set threshold = L/4

      Set cont = 0
      WHILE cont < M DO
            FOR i = 0 TO N/2 DO /* Selection */
                  Choose parents P1 and P2 randomly from POP without
                  replacement
                  IF Hamming_Distance(P1, P2) > threshold DO
                        Perform HUX Crossover, the two new offspring will
                        be inserted in the children population (CPOP)
                  ENDIF
                  Perform the Repairing to CPOP
            ENDFOR
            IF LENGTH(CPOP) == 0 DO
                  Set threshold = threshold - 1
            ELSE
                  Set POP = best N from POP + CPOP
                  Perform the Repairing to POP

            ENDIF
            IF threshold < 0 DO
                  Create the new population (Mutation is applied here)
                  Set threshold = L/4
            ENDIF

            Set cont = cont + 1
      ENDWHILE
END
```

Figure 9. Pseudocode of the CHC genetic algorithm.

Model Building Methodology

Gene interaction models, based on replicates of gene transcription data, are constructed by a genetic algorithm based on either cotemporal, first order next state, or second order next state likelihood which results in a specific fitness function. At the conclusion of the execution of the genetic algorithm, the unique DAGs from the final population, *FP*, are the ingredients for the composite model. The composite model is a weighted directed or undirected graph, corresponding to next state or cotemporal analysis, where the weight of any edge e is:

$$w(e) = \frac{\sum_{d \in FP} X(e \in d) L(d|R_1, R_2, R_3)}{\sum_{d \in FP} L\,(d|R_1, R_2, R_3)} \quad (1)$$

where $\chi(e \in d)$ is a 0/1 function with the value of 1 if and only if the edge e belongs to the DAG d. All the edge weights have values between 0.0 and 1.0, inclusive.

Edges that consistently appear in the DAGs or appear in DAGs with large likelihood tend to have high weight in the composite model. Similarly, edges that consistently do not appear in DAGs or appear in DAGS with low likelihood tend to have low weight in the composite model.

For each of the three analysis paradigms, twelve composite models are created. The twelve composite models consistently have much in common, however, not unexpectedly, often are not identical. Each of these twelve provides an important glimpse into the gene interactions. After computing the union of the unique DAGS from the final populations which form the twelve composite models, this new population (UFP) is the basis for the amalgamated model. The weights of the edges of the amalgamated model are computed exactly as those for the composite model where the final population FP is now UFP. Each weighted edge in the amalgamated model is the relative probability of gene interaction between the two vertices incident with the edge. There will be three amalgamated models, one for each analysis paradigm. Just as with the creation of the composite models, the weights of the edges in the amalgamated models are not based on the frequency of occurrence of the DAGs, but exclusively on the computed Bayesian likelihoods of the unique DAGs in the final population UFP. The amalgamated models are the result of many observations of the genetic algorithm measuring how well DAGs in the chromosome populations fit the replicates of gene abundance data.

Two types of experiments are required. First, it is necessary to validate the algorithmic technique through the use of engineered data. In this simulated data, a specific signal is placed. The modelling technique must be able to accurately discover this signal. Successful capture of this signal will verify algorithmic correctness. Second, the modeling algorithm will be applied to time course gene transcription data collected from Arabidopsis thaliana. The various cotemporal and next state models will suggest likely gene interactions.

Results

Using engineered gene data

Many different engineered data sets were examined as part of the protocol to establish the reliability of the modeling algorithms. This is necessary before applying the algorithm to biological data to generate gene interactions predictions. We discuss one of these using engineered data, which is representative of the others.

In this simulated data set, three replicates of time course gene transcription are used with 12 different genes. The names of these genes are BE1, …, BE12. Each replicate is based on simulated abundance measurements taken at 10 different times. These data were generated in a cotemporal context; hence, the best modeling results are expected when generating cotemporal gene interaction models. However, when using the next state contexts, we expect consistency between the next state models, even between the traditional and CHC genetic algorithms.

The cotemporal signal placed in the replicates was generated as follows: BE1, ..., BE4 are highly correlated (ρ = 0.9), BE5, ..., BE8 are highly correlated (ρ = 0.9), and BE9, ..., BE12 are highly correlated (ρ = 0.9). A multivariable Gaussian random number generator was used to engineer the relationships and to introduce variation in each of three replicates. An expected cotemporal model from this data is shown in figure 10.

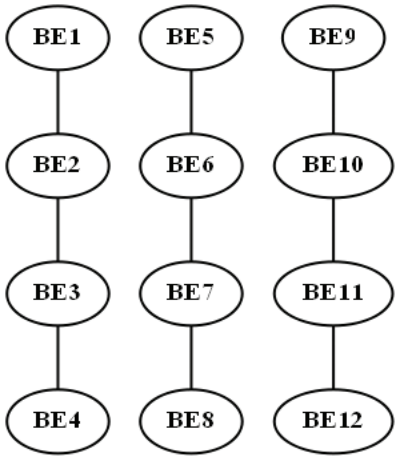

Figure 10. A graph representation of the expected cotemporal model for the engineered data set.

The amalgamated cotemporal model shown in figure 11 captures exactly the engineered cotemporal signal. This model was generated by the traditional and CHC algorithms, where both algorithms found the same signals but with different probabilities, matching with the model in the figure 10.

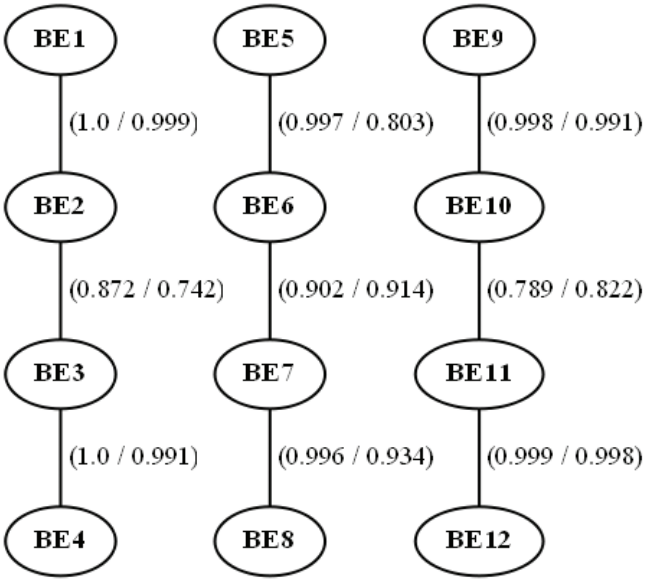

Figure 11. The amalgamated model of the traditional and CHC genetic algorithms using the cotemporal paradigm (with cutoff value 0.7). Both genetic algorithms generated similar gene interactions with different probabilities between the edges. The probabilities correspond to the next format: (traditional / CHC).

Even though the data was engineered exclusively with a cotemporal signal, there certainly can exist first order next state and second order next state signals in the data. In figure 12 the second order next state models strongly suggest the existence of a signal. The traditional and CHC models six edge (B11→B5, B6→B2, B6→B4, B6→B2, B9→B4, B8→B7). There are six directed edge in the traditional, not CHC, model (BE6→BE1, BE3→BE5, BE5→BE1, BE9→BE2,

BE11→BE8, BE12→BE8). The six directed edges found in both models shows strong agreement. Similarly, but not shown, there is significant agreement between the first order next state models.

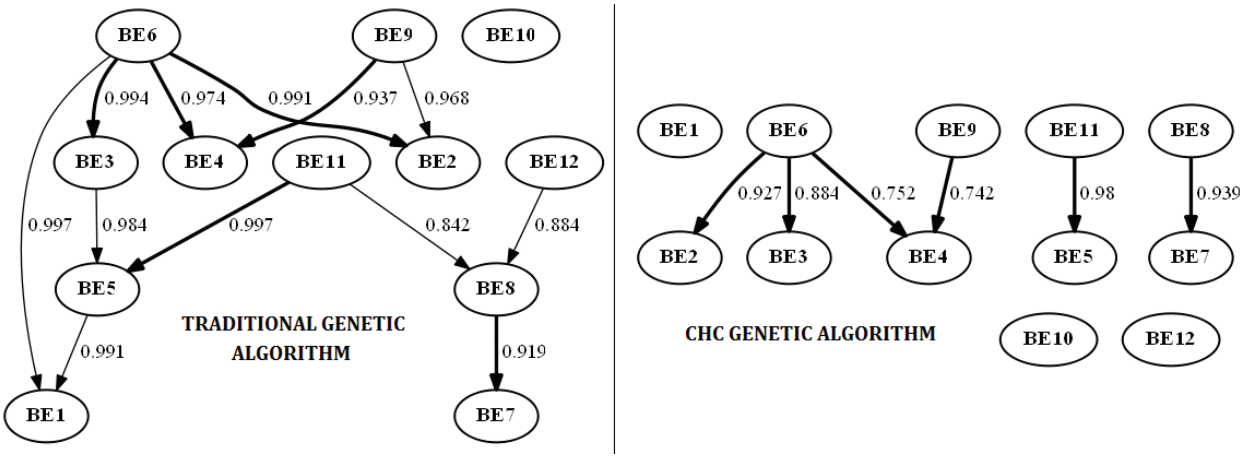

Figure 12. Amalgamated (with cutoff value 0.7) second order gene interactions from the engineered data set using the traditional and CHC genetic algorithms. Information from twelve composite models were integrated to create these amalgamated models. The bold edges represent the similar edges between the models.

These models illustrate the diversity of prediction models created by the genetic algorithms. There are many similarities between the cotemporal/next state models, but there will be variation between the models. The model variation is due to data differences in the replicates and to the stochastic nature of the genetic algorithms.

Using arabidopsis gene data

The genetic-algorithm based methods are applied to an experimentally generated set of transcript abundance data from roots of Arabidopsis thaliana after auxin treatment [10]. The goal of this analysis tests whether our probabilistic amalgamated models will identify relationships that have been identified through experimental approaches, and whether they can identify interesting new relationships that then can be examined experimentally. The gene abundance data was derived from work in the laboratory of Professor Gloria Muday (Department of Biology, Wake Forest University).

This analysis focused on a set of 12 Arabidopsis thaliana's genes from a large time-course microarray experiment examining the effect of the plant hormone indole-3-acetic acid (IAA) on changes in transcript abundance [10]. To generate this data, triplicate samples of roots of Arabidopsis thaliana were either mock-treated or treated with 1 μM IAA for 0, 0.5, 1, 2, 4, 8, 12, or 24 hours [10]. This data set identified over 1200 transcripts that showed consistent, reproducible, and significant enhancement or reduction after IAA treatment [10]. This set of 12 genes were selected previously [21]. Seven genes (IAA1, IAA2, IAA3, IAA14, IAA16, IAA18 and IAA19) are members of the AUX/IAA family, a group of transcriptional repressor proteins [11], [23], [24]. There are also genes encoding two transcriptional activator proteins linked to root development, ARF19 and WRKY23 [6], [17], and signaling proteins implicated in auxin regulated processing including a G-protein (XLG1), a kinase (PINOID), and a regulator of cell cycle progression, cyclin (CYCB2) [7], [26].

In the figure 12, results obtained using both genetic algorithms (traditional and CHC) are shown. When using the cotemporal paradigm, the program finds 9 signals (IAA3—PINOID, CYCB2—PINOID, PINOID—IAA14, IAA14—IAA18, IAA18—XLG1, WRKY23—IAA16, IAA2—ARF19, ARF19—IAA19, IAA19—IAA1) in both models. Besides, when using the first order next state paradigm, the program finds only 1 signal between the models (IAA16→ARF19). Finally, when using the second order next state, the 4 signals, WRKY23→PINOID, WRKY23→IAA1, ARF19→IAA16, IAA19→IAA14, are found among models. The figure 13 illustrates a combined model, only showing the signals detailed above.

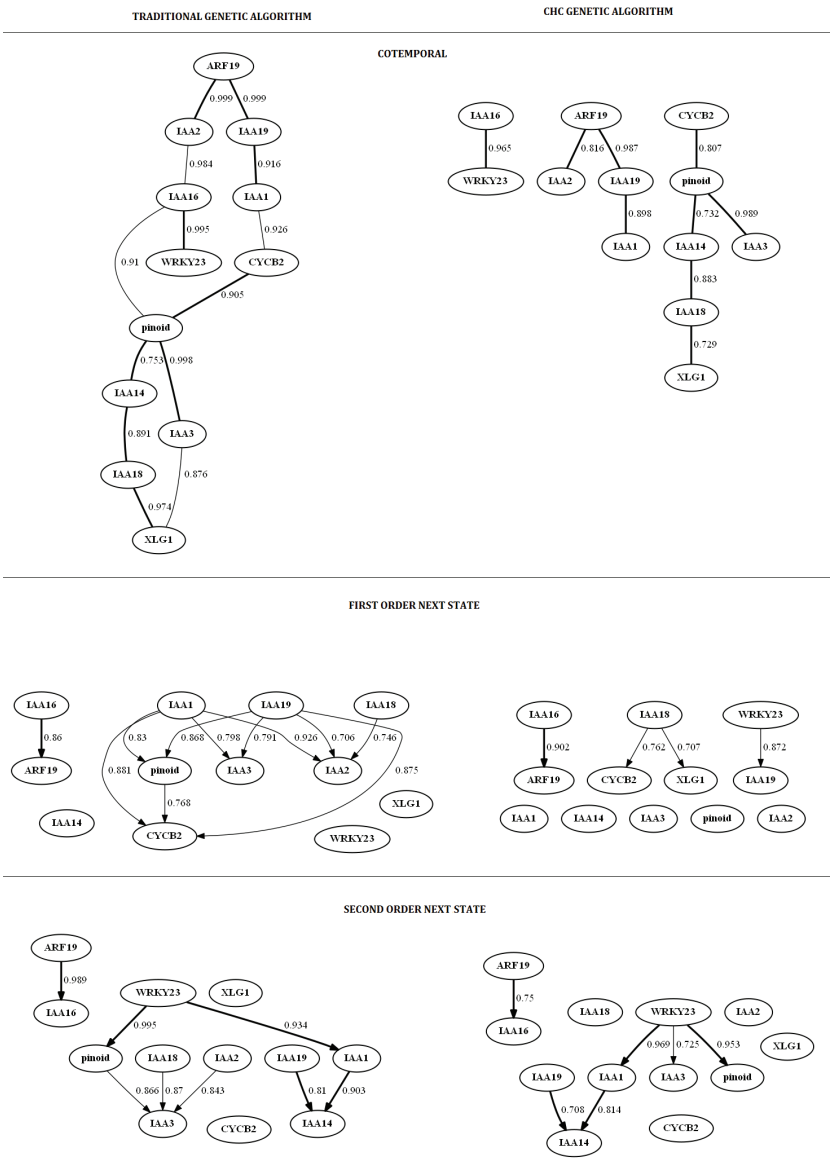

Figure 13. Amalgamated models with cutoff value 0.7, when using the traditional and CHC genetic algorithms in the three different modelling paradigms using real Arabidopsis thaliana time-course data. Information from twelve composite models were integrated to create each of the amalgamated models. The bold edges represent the similar edges between the models on each paradigm.

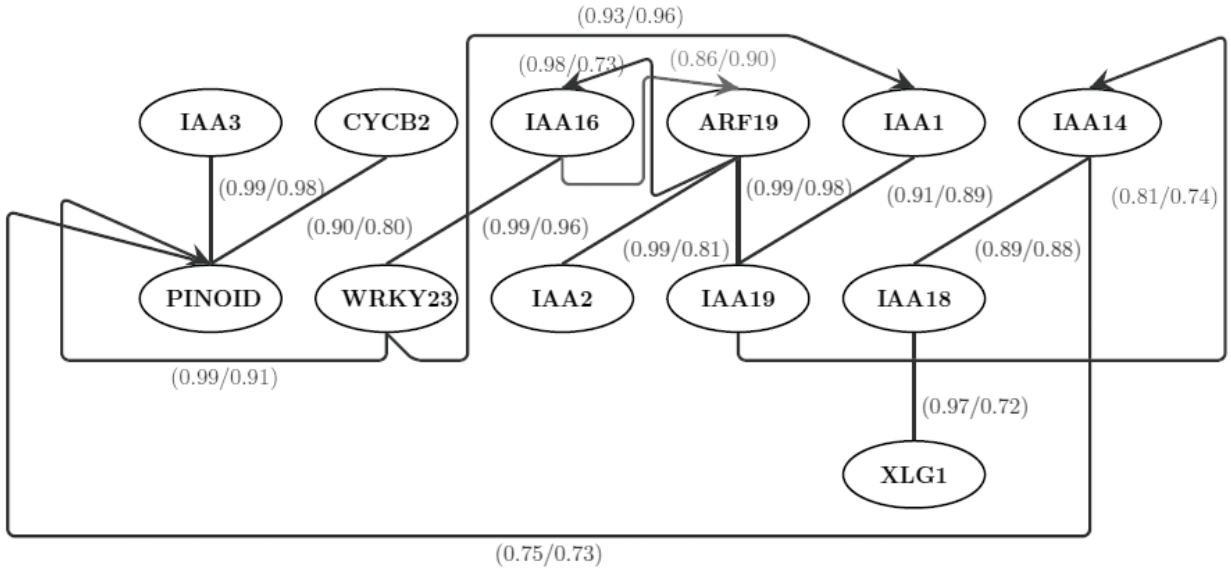

Figure 14. Gene interactions, combining the edges that are similar in the cotemporal (brown), first order next (red) state and second order next state (blue) using the traditional and CHC genetic algorithms with the Arabidopsis thaliana data. The probabilities correspond to the next format: (traditional / CHC).

Implementation details

This project was developed using the programming multiparadigm language python, in its version 3. Different libraries for python were used in all the development, including: bigfloat, networkx, numpy, scipy, graphviz and pydotplus. These libraries simplified the general development of the two different genetic algorithms.

This project can be found in the next links:

- Traditional Genetic Algorithm: https://github.com/keme040994/SGA_Project
- CHC Genetic Algorithm: https://github.com/keme040994/CGA_Project

The project was developed based on a modular technique using different packages to keep the main functions separated. The multiple operators programmed for the main functions can be found in different specific files in the packages.

Conclusions

As main conclusions for this project in the next paragraphs we discuss about the accomplished goals of this new approach and general facts, to understand the relevance of this project.

Both genetic algorithms, the traditional GA and CHC GA are fairly effective at determining composite/amalgamated models in terms of capturing the signal. The genetic algorithms are very fast compared to the last approach, using the Metropolis-Hastings algorithm, when it takes several hours to display results instead of hundreds. The results displayed by the traditional GA and the CHC GA are also very consistent.

To maintain a legal population, we required and initialization and repairing operators. Also, the development environment, using python3 as the programming language with some libraries, provided and effective and portable platform.

As a future work, we hope to scale up the number of genes, in order to find more signals in the gene interactions. Another important to accomplish is to make the genetic algorithms more flexible, like providing a user interface or an easier method to use the platform.

Acknowlegments

Los autores expresan sus agradecimientos al Consejo Nacional de Rectores (CONARE), por el apoyo financiero, y al Tecnológico de Costa Rica, en especial al personal de la Escuela de Computación, a la Vicerrectoría de Investigación y Extensión y a la Dirección de Cooperación Internacional.

A su vez, se agradece todo el apoyo de parte del personal del Departamento de Ciencias de la Computación (Department of Computer Science), Departamento de Matemáticas y Estadísticas (Department of Mathematics and Statistics) y el Departamento de Biología (Department of Biology) de Wake Forest University, en Carolina del Norte, Estados Unidos de América.

Bibliography

[1] Chawdhry, P. K., Roy, R. & Pant, R. K. (1998). Soft Computing in Engineering Design and Manufacturing. London, UK: Springer.

[2] Epp, S. S. (2004). Discrete Mathematics with Applications (3rd Edition). Boston, MA, USA: Brooks/Cole.

[3] Eshelman, L. (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In G. J. E. Rawlins. (Ed.), Proceedings of the First Workshop on Foundations of Genetic Algorithms (pp 265-283). San Mateo, CA, USA: Morgan Kaufmann.

[4] Friedman, N., Linial, M., Nachman, I. & Pe'er, D. (2000). Using Bayesian networks to analyze expression data. Journal of Computational Biology, Vol. 7(3), 601–620.

[5] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Boston, MA, USA: Addison-Wesley.

[6] Grunewald, W., Smet, I. D., Lewis, D. R., Lofke, C., Jansen, L., Goeminne, G., … Beeckman, T. (2012). Transcription factor WRKY23 assists auxin distribution patterns during Arabidopsis root development through local control on flavonol biosynthesis. Proceedings of the National Academy of Science USA, Vol. 109, 1554–1559.

[7] Heo, J. B., Sung, S. & Assmann, S. M. (2012). Ca2+-dependent GTPase, extra-large G protein 2 (XLG2) promotes activation of DNA-binding protein related to vernalization 1 (RTV1) leading to activation of floral integrator genes and early flowering in Arabidopsis. Journal of Biological Chemistry, Vol. 287, 8242–8253.

[8] Jones, N. C. & Pevzner, P. A. (2004). An Introduction to Bioinformatics Algorithms. Cambridge, MA, USA: MIT Press.

[9] Krämer, N., Schäfer, J. & Boulesteix, A. L. (2009). Regularized estimation of large-scale gene association networks using graphical Gaussian models. BMC Bioinformatics, Vol. 10(384), 1-24.

[10] Lewis, D. R., Olex, A. L., Lundy, S. R., Turkett, W. H., Fetrow, J. S. & Muday, G. K. (2013). A kinetic analysis of the Auxin transcriptome reveals cell wall remodeling proteins that modulate lateral root development in Arabidopsis. The Plant Cell, Vol. 25, 3329–3346.

[11] Liscum, E. & Reed, J. W. (2002). Genetics of AUX/IAA and ARF action in plant growth and development. Plant Molecular Biology, Vol. 49(3-4), 387–400.

[12] Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. Charlotte, NC, USA: Springer-Verlag.

[13] Michniewicz, M., Zago, M. K., Abas, L., Weijers, D., Schweighofer, A., Meskiene, I., … Friml, J. (2007). Antagonistic regulation of PIN phosphorylation by PP2A and PINOID directs auxin flux. Cell, Vol. 130(6), 1044–1056.

[14] Miescher, F. (1871). Ueber die chemische Zusammensetzung der Eiterzellen. Medicinisch-chemische Untersuchungen, Vol. 4, 441–460.

[15] Mitchell, M. (1996). An Introduction to Genetic Algorithms. Cambridge, MA, USA: MIT Press.

[16] Norris, J. L., Patton, K. L., Huang, S., John, D. J. & Muday, G. K. (2015). First and Second Order Markov Posterior Probabilities on Multiple Time-Course Data Sets. Proceeding of the Southeast Conference of the IEEE, April 2015, 1-8. doi: 10.1109-SECON.2015.7132880

[17] Okushima, Y., Fukaki, H., Onoda, M., Logis, A. T. & Tasaka, M. (2007). ARF7 and ARF19 regulate lateral root formation via direct activation of LBD/ASL genes in arabidopsis. Plant Cell, Vol. 19(1), 118–130.

[18] Patton, K. L. (2012). Bayesian interaction and associated networks from multiple replicates of sparse time-course data (Master's Thesis). Wake Forest University. Winston-Salem, NC, USA.

[19] Patton, K. L., John, D. J. & Norris, J. L. (2012). Bayesian probabilistic network modeling from multiple independent replicates. BMC Bioinformatics, Vol. 13(9), 1–13.

[20] Patton, K. L., John, D J.., Norris, J. L., Lewis D. & Muday, G. (2013). Hierarchical Bayesian system network modeling of multiple related replicates. BMC Bioinformatics, Vol. 7, 803-812.

[21] Patton, K. L., John, D. J., Norris, J. L., Lewis D. R. & Muday, G. K. (2014). Hierarchical Probabilistic Interaction Modeling for Multiple Gene Expression Replicates. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 11(2), 336-346.

[22] Pevsner, J. (2003). BioInformatics and Functional Genomics. Hoboken, NJ, USA: Wiley.

[23] Reed, J. W. (2001). Roles and activities of Aux/IAA proteins in Arabidopsis. Trends in Plant Science, Vol. 6(9), 420–425.

[24] Rouse, D., Mackay, P., Stirnberg, P., Estelle, M. & Leyser, O. (1998). Changes in auxin response from mutations in an AUX/IAA gene. Science, Vol. 279(5355), 1371–1373.

[25] Schäfer, J. & Strimmer, K. (2005). An empirical Bayes approach to inferring large-scale gene association networks. Bioinformatics, Vol. 21, 754–764.

[26] Sukumar, P., Edwards, K. S., Rahman, A., DeLong, A. & Muday, G. K. (2009). PINOID kinase regulates root gravitropism through modulation of PIN 2-dependent basipetal auxin transport in Arabidopsis thaliana. Plant Physiology, Vol. 150(2), 722–735.

[27] Tarjan, R. E. (1972). Depth-first search and linear graph algorithms. SIAM Journal on Computing, Vol. 1(2), 146-160.

[28] Whitley, D. (1994). A Genetic Algorithm Tutorial. Statistics and Computing, Vol. 4, 65-85.

[29] Wilson, R. J. (2010). Introduction to Graph Theory (5th Edition). London, UK: Prentice-Hall.