ATHABASCA UNIVERSITY

# MACHINE LEARNING TECHNIQUES TO UNVEIL AND UNDERSTAND *PSEUDOMONAS AERUGINOSA* SURVIVAL MECHANISM IN NUTRIENT DEPLETED WATER

BY

BERTRAND SODJAHIN

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE MASTER OF SCIENCE IN INFORMATION SYSTEMS

FACULTY OF SCIENCE AND TECHNOLOGY
SCHOOL OF COMPUTING AND INFORMATION SYSTEMS

ATHABASCA UNIVERSITY
JULY, 2018

**Approval of Thesis**

The undersigned certify that they have read the thesis entitled

**MACHINE LEARNING TECHNIQUES TO UNVEIL AND UNDERSTAND PSEUDOMONAS AERUGINOSA SURVIVAL MECHANISM IN NUTRIENT DEPLETED WATER**

Submitted by

**Bertrand Cadoukpe Sodjahin**

In partial fulfillment of the requirements for the degree of

**Master of Science in Information Systems**

The thesis examination committee certifies that the thesis
and the oral examination is approved

**Co-Supervisors:**
Dr. Vivekanandan Kumar
Athabasca University

Dr. Shauna Reckseidler-Zenteno
Athabasca University

**External Examiner:**
Dr. Joe Harrison
University of Calgary

November 14, 2018

## Dedication

***To Him by Whom I can do all things***

**Acknowledgments**

My sincere gratitude to my two supervisors, Dr. Vivekanandan Kumar and Dr. Shauna Reckseidler-Zenteno, for their unwavering supports and the great mentoring they have provided me during this thesis. Thank you for all.

I would also like to acknowledge the following organizations for their financial supports through scholarships, awards, and grants: Alberta Innovates Technology Future (AITF), Alberta Advanced Education, and the Canadian Artificial Intelligence Association. The funding provided was not only conducive to my focus on the research work but also helped in my travels nationally and internationally for the dissemination of the research findings at various conferences.

The accomplishment of this milestone would not have been possible without the support of my brothers. To you, I say un *Grand Merci!*

Lastly, to my colleagues of Professor Vive's research group, it was a great experience being part of the team and doing my research side by side with you. Thank you for your feedback and contributions during my presentations.

## Abstract

*Pseudomonas aeruginosa* is a Gram-negative organism that is ubiquitous in the ecosystem and antibiotic resistant. Capable of long-term survival, it is a common cause of hospital-acquired infections. The focus of this thesis is to unveil *P. aeruginosa* genes interactions and identify those that are pivotal to its mechanisms of survival. With unlabeled data collected from *P. aeruginosa* gene expression in response to low nutrient water, a Bayesian Networks Machine Learning methodology was implemented, and a static regulatory network of its survival was modeled. Subsequently, node influence techniques were used to infer a dozen genes as key orchestrators of the survival phenotype. Among these genes, *PA0272* was identified to be the root node in the learned network model. Water survival experiments were conducted in the lab on *PA0272* mutants, and it was interestingly found that their survival declined by 10-fold compared to the wild type PA01; 10-fold or higher being significant.

**Keywords**: *Pseudomonas aeruginosa*, gene expression, Bayesian Network, Machine Learning

## Table of Contents

## List of Tables

## List of Figures

# Chapter I - Introduction

A Gram-negative bacterium, *Pseudomonas aeruginosa* is ubiquitous in the environment and known for its ability to inhabit a number of niches, causing disease in plants, animals, humans. Due to *P. aeruginosa* resistance to antibiotics and its ability to form biofilms, the diseases caused are very difficult to treat [1, 2]. This diverse organism is a common cause of hospital-acquired infections, mostly causing skin infections in burn patients, infections of indwelling devices, and fatal lung infections in patients with cystic fibrosis [2, 3]. According to [4], approximately 40% of mechanically ventilated patients with *P. aeruginosa* pneumonia succumb to their condition. Studies have shown that *P. aeruginosa* may survive for months on hospital surfaces [5] and can be considered a model organism for the study of diverse bacterial mechanisms that contribute to persistence. The presence of a large number of genes, 50% more genes than *E. coli*, permits diversity and adaptability by the organism [6, 7]. Understanding how this organism is able to survive, particularly in water depleted of nutrients, is the primary focus of this thesis.

## 1. Data Description

An existing transposon library of *P. aeruginosa* mutants was utilized. This mini-Tn5-*luxCDABE* transposon mutant library of *P. aeruginosa* PAO1 is a collection of random transposon mutants, each containing a mutation in a different gene. This is the result of insertion of a mini-*Tn5* transposon into the gene, which prevents effective transcription and eventually translation of the gene into a functional protein. Each insertion of the mini-*Tn5* transposon contains the *luxCDABE* operon,

which results in light production as the gene is being transcribed. This allows the determination of gene expression under a variety of conditions. The *luxCDABE* operon is derived from the bacterium *Photorhabdus luminescens*, which is a luminescent marine bacterium [8]. The mini-Tn5-*luxCDABE* library in PAO1 contains 9,000 mutants, of which 2,500 have been mapped and characterized. Of the 2,500 characterized mutants, 1,384 were determined to produce light and determined to be *lux* fusions. This collection of 1,384 mutants was screened for gene expression in water by measuring the luminescence in counts per second (cps) as well as the optical density (OD600) of each well over time points ranging from 8 hours to 2 months. Gene expression data [9] was analyzed by dividing cps/OD600 for each well at each time point and then calculating the fold change of each well at each time point compared to time zero. The gene expression data has 15 columns overall, the first of which is "well_ID" and represents the array identification of the wells in which the mutants have been inoculated. The second column is "gene_name", the third is the "PA_number", the fourth is "product_name", the fifth is "original_well_ID", that is the ID of the well before the transfer, and columns 6 to 15 ($T_4$ – $T_{672}$) represent the ten different time points (sampling times) of the gene expression which represents the ratio of the actual measurement (absolute value) at time $T_i$ ($i>0$) by the value for the same gene at time zero ($T_0$). The procedure of establishing this ratio is known as normalization, that is from the absolute value to a relative value. The second dataset that consisted of 18-timepoint was obtained a couple months after the collection of the first dataset, for an overall 28 data tuples.

## 2. Research Questions

The *P. aeruginosa* genome has a greater genetic and functional diversity [4], which explains its versatility. With 8.4% of its genes predicted to be involved in regulation, this percentage for *P. aeruginosa* is considered to be well above those of all sequenced genomes [6]. Both [4] and [10] discussed a comparative genomic study of various *P. aeruginosa* genomes. This led to the paradigm according to which the bacterium is of a mosaic composition: a *Core Genome*, and an *Accessory Genome*. The former is common to all the strains studied and encodes a certain number of factors, such as enzymes involved in nutrient uptake and metabolism. The second, i.e., the accessory genome, represents about 10% of the genome and varies from strain to strain. Works by [4] and [10] attribute to this latter component the niche-based adaptation of the organism which survives in low nutrient environments. Certain elements of the accessory genome are thought to promote persistence in hostile environments through the encoding of metabolic pathways [11, 12]. In 2010, [4] stated the necessity to acquire knowledge of the accessory genome for the application of *P. aeruginosa* in biotechnology and at a more critical level for the development of effective treatment or prevention of the virulent infections caused by the bacterium. The foregoing provides a substantial rationale and significance to our research project which aims to identify and understand the survival mechanism of *P. aeruginosa* in low nutrient environment. Below are formulated the following main research questions and hypotheses:

Q1: Which computational model accurately characterizes *P. aeruginosa* survival mechanism in low nutrient water?

Q2: As per a well-defined computational model, which regulatory genes primarily contribute to *P. aeruginosa* survival in low nutrient water?

H1: If genes are considered as nodes of a graph, a graphical model would describe the mechanism of survival, unveiling causal node influences, and functional interplays among gene groups.

H2: If the bacterium is considered as a population of genes, not all but only some key genes would operate as regulators, to orchestrate the survival mechanism.

Subsequently, additional questions are posed, some of which are partially broached in the scope of this work, and recommendations are made to further tackle them in future works.

Q3: What are the functional interplays in *P. aeruginosa* survival mechanism in the absence of nutrients?

Q4: Could the bacterium temporal states be computationally inferred?

Q5: Is *P. aeruginosa* survival solely due to accessory genome or of shared responsibility with core genome? And if so, to which extent (%) is each of the two components accountable?

Designing a methodology to answer these questions will lead to the identification of low nutrient response genes, shedding thus light on the survival mechanism. Also, it will be instrumental in elucidating temporal direction of interaction (causality/influence) among *P. aeruginosa* genes. With an existing transposon library

of *P. aeruginosa* mutants, research experiments were designed to collect gene expression data [9]. Machine Learning techniques have been applied on this dataset to infer information concerning the survival mechanism of *P. aeruginosa*, thus making this a multidisciplinary research.

## 3. Thesis Organization

This thesis is organized into five chapters. The first is the introduction to the research topic. It summarizes the problem, describes the dataset and announces the research questions. The second chapter covers literature review and the third traces the research methodology. The fourth chapter shows and discusses the results. The fifth chapter offers conclusions and future works. Lastly, the back matter of the thesis includes an appendix with additional tables.

# Chapter II – Literature Review and Theoretical Framework

## 2.1 Microarray Data and Bases for Processing Techniques

A microarray concept was used in [13] to introduce gene expression analysis and computational genomics. Generally, functional genomics, genome sequencing, and gene profiling or gene expression (to identify genes associated with a certain phenotype) are frequent topics in this area of study. To these topics, [13] does not propose specific solutions per se; rather it introduces the microarray structure, its operation, and the general analysis methods of gene expression data. This includes the representation of gene expression data and the computation of distance for the evaluation of similarity, which is an integral part of clustering methods. Though clustering methods are much more complex in their categorization [14], the literature in [13] distinguishes two major types of clustering: Hierarchical and non-hierarchical. They further highlight key terminologies that are useful in referencing portions of data in the gene expression matrix. The first is *gene expression profile* and corresponds to the cumulative expression levels for a gene across all experimental conditions. The second, *sample expression profile* is the cumulative expression levels for all genes in a single experimental condition. Another interesting point in [13] is the aspect of relating expression data to other biological information for acquiring insight on biological processes and making new findings. Of notable interest is also [15] where a probabilistic model identifies co-regulated genes with their transcriptional regulators and the conditions that influence their regulation.

### 2.2 Machine Learning in Gene Expression Analysis

In [16], a gene is defined as a segment of DNA which encodes for a protein, and gene expression is described as the sequential steps of the DNA transcription into RNA and the translation of the RNA into the associated protein. The expression level of a gene in an organism is therefore measured through the observation of its protein fabrication rate. A microarray allows to capture at once the integral biological activities and to generate high-throughput data useful in the inference of cell regulatory pathways.

Discussing the applications of Machine Learning in biology, [16] distinguish *Supervised Learning* and *Unsupervised Learning*, both of which learn information from data. The supervised learning is a feedback-based learning in which for every input, the learning agent is provided with a target reference. But in the unsupervised learning there is no feedback to the learning algorithm. In gene expression analysis as highlighted by [16], unsupervised learning could be achieved with techniques such as Clustering or Bayesian Networks (BN). The flexibility of Clustering and its intuitiveness make its use common in the domain of bioinformatics [17]. Bayesian Networks on the other end has a probabilistic approach and its application in gene expression microarray data has drawn much attention due to the insight it provides pertaining to the network of interaction among cells that regulate the gene expression. BN was applied for the first time in this area by [18], where a probabilistic network was learned with a *Saccharomyces cerevisiae* data matrix, capturing the joint probability distribution over the expression levels of the genes.

### 2.3 Bayesian Networks and Structure Learning

Bayesian networks (BN) combine two well established mathematical areas: Probability and Graph Theory. Defined as Direct Acyclic Graph (DAG), it is a graphical representation that establishes probabilistic relationships among a finite set of discrete random variables of a given domain [19]. Each node in the graph, as illustrated in Figure 2.1 [20], represents a domain variable. When a node has a directed arrow or arc to another one, the first is said to be the ancestor or parent of the latter. To each arrow in the graph is associated a Conditional Probability Distribution (CPD) of a node $X_i$ given its parent $Pa(X_i)$. Considering a graph $G$ and its associated CPD, the joint distribution of all variables is uniquely represented by the following factorized joint probability distribution:

$$P(X_1, \dots, X_n) = \prod_i P(X_i|Pa(X_i)) \qquad (2.1)$$

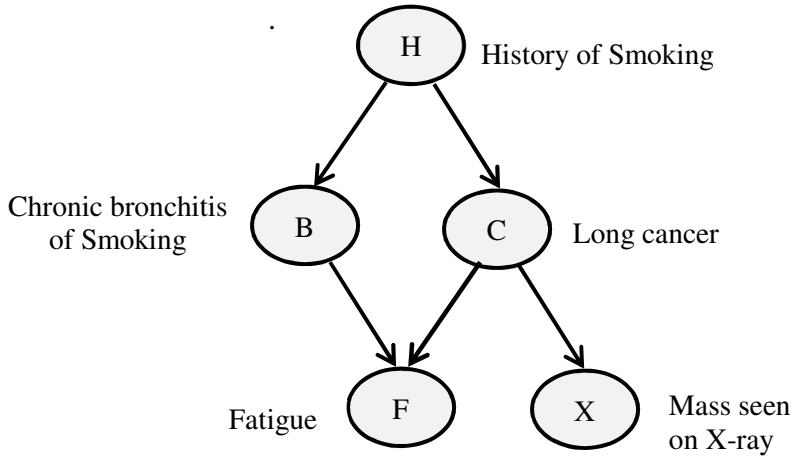



Figure 2.1: An example of Bayesian Networks, excerpted from [20]

The link between two variables in a BN is interpreted as correlation or causation. In the latter case, the network is called *Causal Bayesian Network*, and is adopted in most gene expression analysis for causality inference. Illustrating a causal

influence, [19] presented the manipulation of an independent variable $A$ while observing its effect on a dependent variable $B$. This resulted in different probability distributions of $B$ given the values assigned to $A$, ($P[B|A] = a_1 \neq P[B|A = a_2]$).

In real life applications, the graphs of the BN are initially unknown. Domain knowledge is one of the means for constructing Bayesian Networks. It is achieved in three main steps: the determination of the number and the meaning of the domain variables, the establishment of the influence relationships among the variables, and the computation of the conditional probabilities from the BN structure.

Several domain knowledge-based techniques have been developed [21, 22, 23] to aid the construction of Bayesian networks. But these knowledge-based techniques are not suitable when it comes to applications requiring a complete construction i.e., there is an edge between any pair of nodes in the graph. On the other hand, the method of learning from data [23, 24, 25] can achieve a complete construction [19] and consists of two main learning tasks: Parameters learning and Structure learning. Because the network structure is a priori unknown, learning this structure would theoretically precede the learning of the parameters. However, in practice, the parameters' learning is generally integrated through an inner loop with the structure learning [19]. Learning Bayesian networks structure from data amounts to searching amid the eligible networks, the one that best represents the data. Figure 2.2 illustrates the general overview. Because this search process is NP-complete [26] and therefore is likely to take indeterminate amount of time to complete, heuristic approaches have been proposed to learn graph structure from data [27, 28]. These

structure learning algorithms are categorized in two majors groups: S*core-and-search-based* approach [24, 25, 29] and *constraint-based* approach [30, 31]. From these approaches emerged a third one known as *hybrid method*, consisting in the coupling of the two other methods.

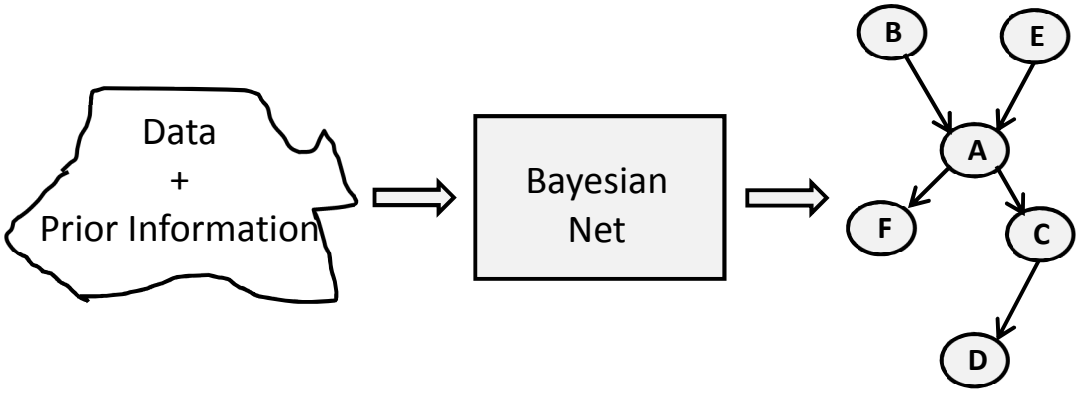



Figure 2.2: Graphical illustration of Bayesian Network Learning [18]

### 2.3.1 Definitions and Theorems

Presented below are key definitions, theorem and remarks to provide some basis for structure learning.

Definition 2.3.1: There is an *uncoupled head-to-head meeting* or *v-structure* at a node X on a chain in a DAG if there is a head-to-head meeting i.e., Y→X← Z at X and no edge between Y and Z.

Theorem 2.3.1: Two DAGs $G_1$ and $G_2$ are *Markov equivalent*[1] if and only if they have the same links (edges without regard for direction) and the same set of uncoupled head-to-head meetings. Also, these DAGs are faithful to the same probability distribution.

Definition 2.3.2: *Markov equivalence classes* are disjoint classes, each containing

[1] Markov equivalence groups DAGs into equivalence classes based on the conditional independencies that they entail [32].

DAGs that are Markov equivalent.

Definition 2.3.3: A *DAG pattern* is defined as a unique graph that is used to represent a given class of Markov equivalence and is characterized by the coexistence of directed and undirected edges.

Remark 2.3.1: In a DAG however, all edges are directed. While for every undirected edge between two nodes in a DAG pattern the corresponding edge in any of the DAGs of the class is directed, the direction could be either way. But for a directed edge in a DAG pattern, the homologous edge is identical for all DAGs in the class.

Remark 2.3.2: DAG pattern, PDAG (Partially Directed DAG) and CPDAG (Complete Partially Directed DAG) all refer here to Markov equivalence class and would be used interchangeably.

### 2.3.2 Score-and-Search-based Method

The score-and-search-based method requires a *search space* to be defined, a *search strategy* to be devised and a *model selection* criterion.

#### a. Search Space

The search space a priori contains all eligible Direct Acyclic Graphs (DAGs) in the domain. This is known as *Network structure search space*. Works by [33] have been instrumental in determining the number of DAGs in a search space given the variables count in the domain under consideration. Formula (2.2) below shows the number of DAGs as a highly exponential function of the number of variables.

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} C_i^n 2^{i(n-i)} f(n-i) \qquad (2.2)$$

When applied to real world domains which usually have substantial number of

variables, enumerating and scoring all possible DAGs is infeasible. An alternative known as *Equivalence classes of network structure search space* that considerably reduces the number of network candidates is the one in which the networks under consideration in the space are Complete Partial DAG (CPDAG) also referred to as DAG pattern [28]. This cardinality reduction is corroborated by [34] that demonstrates that for an increasing number of variables up to *n=10,* there is an asymptotic ratio of number of DAGs over number of CPDAGs:

$$\lim_{n \to 10} \frac{DAGs\_Count}{CPDAGs\_Count} = 3.7 \qquad (2.3)$$

A CPDAG is a single graphical representation of a group of DAGs which are Markov equivalent, and DAGs belonging to same CPDAG class imply exactly the same set of independence statements among the variables in the domain. While search in a space of DAGs results in a DAG as output, the search result for a CPDAGs space is a CPDAG - a group of DAGs. Though [19] notes that Markov equivalent structures cannot model relationship of exact direct influence, this research posits that this could still be achieved by decomposition into steps. Basically, one would first take advantage of the reduced number of network candidates (CPDAGs) in the space for a faster selection of the best CPDAG model. Then the selected CPDAG would be expanded into all its DAGs. This set of DAGs thus shortlisted would be used as the new DAGs space from which finally the best DAG model would be elected. Despite that CPDAG (DAG pattern) is conducive to a reduced search space's cardinality (compared to the corresponding DAGs space), exhaustively enumerating and scoring all CPDAGs is also said to be computationally impracticable for non-small number of variables. Heuristic methods are therefore proposed

to expedite the finding process of the local maximum in the structure space, be it with DAGs or DAG patterns.

A third space, which has been discussed in the next section, is referred to as *Orderings over the network variables search space*. Typically, the variables in the space are ordered prior to the search; and the ordering could either be based on domain knowledge or computationally achieved.

### b. Search Strategies

The search strategies range from *brute-force* to *simulated annealing* through the *firsts* (*depth-**first**, width-**first**, best-**first***). Referred to as exhaustive search, the principle of brute-force in Bayesian networks learning consists in comprehensively enumerating all eligible DAGs and respectively assigning a score to each. Thereafter, the DAG with the best score is selected. This rigorous method, only computationally feasible for $n \leq 5$, is considered gold standard for the evaluation of other algorithms [19], notably the heuristic ones.

The general search strategy of heuristic algorithms rests on an estimation of the cheapest path from a start state to a goal state, without laboriously browsing through each state in the space. Among these heuristics is the *K2 algorithm* which operates on a pre-established total causal parents-children ordering variables. This total order not only prevents cycles in structures, it also reduces the search space. At the start, K2 considers that each node has no parents. By increments, it adds parents to the current node until the score of the corresponding network formed cannot be further improved. Where variables are not pre-ordered, a *search over orderings* will be adopted, which is said to be more efficient than search over DAGs [19].

Though in certain situations the network structure is known, and one just needs to learn its parameters, in real world cases however the structure is unknown, and its learning can be generally addressed as an optimization problem in a discrete space. *Greedy search*, which is an intuitive way of tackling such optimization problem, proceeds by choosing a certain structure from the space, as the starting point. It then moves to the neighbor with the highest score and continues so forth until no structure is better than the current (see Algorithm 2.1). This means that the local maximum has been found.

A neighbor in Bayesian network structures is defined as any graph obtained as a result of applying one single operation (add, delete or reverse arc) to the current structure. Literature [19] observes three different considerations for an initial state. This could be a structure with no arcs (where all nodes are independent, or a structure where any pair of nodes is connected), a structure with complete arcs, or a structure randomly selected from the search space. Though [19] observes that both, the "goodness" of the local maximum and the time required to reach that maximum, depend on the initial structure, it also recognizes that there is no formalized knowledge for mapping best initial structure to best local maximum.

In default of such knowledge, running the algorithm several times, with a different initial structure each time, is an alternative. From all the local maxima found, the best model is chosen. In the quest of a good local maximum solution, techniques such as simulated annealing and best-first have been proposed. *Simulated annealing* [35] is a stochastic heuristic search that has the ability to overcome the local

maximum and further to find the global maximum[2]. This is possible because instead of an exclusively deterministic selection of the neighbor with the highest score, simulated annealing concurrently makes systematic neighbor selection with probability *p*, and random selection with a probability *1-p*. *Best-first* on the other hand is simply a systematic highest score-based search that does not involve probability. According to [25], when restarted multiple times with different initial structures, greedy search (refer to Algorithm 2.1) achieves better performance in terms of model and time, over simulated annealing and best-first.

**Algorithm 2.1**: Greedy Search – Excerpted from Bayes Net (2007)

**Input** (Dataset)

1. Generate initial BN, evaluate it and set it as current BN
2. Evaluate the current BN's neighbours
3. If best score of the neighbours > current BN's score,
   {set the neighbour with best score as the current BN and return to step 2}
   else {Stop}

**Output** (BN)

### c. Model Selection

Carvalho [36] categorizes scoring functions for model selection in two groups: *Bayesian scoring functions* and *Information-theoretic scoring functions*. A model is selected based on the result of scoring function whose role is to evaluate the degree to which a generated network represents the dataset *D* under consideration. There are several scoring functions that are classified under the two aforementioned categories. The best Bayesian network is the one that produces the highest score under the chosen scoring function. One of the bases of scoring function is

[2] While the local maximum of a function is its highest value over a local interval, global maximum of a function is its absolute highest value over its entire domain of definition.

Maximum Likelihood[3] (ML), according to which the network that predicts the data with the highest probability is selected, and is given by:

$$\hat{\theta}_G = \arg max_{\theta_G} \{p(D|\theta_G, G^h)\} \qquad (2.4)$$

$G^h$ is the hypothesis of the Bayesian Network from the graphs search space, $\theta_G = (\theta_1, \theta_2, \ldots \theta_n)$ is the vector of parameters where each $\theta_i$ is also a vector of parameters for the variable $X_i$ whose distribution is $p(x_i|Pa_i, \theta_i, G^h)$, and $\hat{\theta}_G$ is the ML among all of the estimated parameters vectors $\theta_G$. [19] noted that in some cases, the application of ML principle can lead to over fitting as the selected model would suit very well the current data but cannot be generalized and would therefore need to be relaxed. To achieve this relaxation, the scoring function is coupled with penalty in order to dissuade complexity in the model selection function.

*Bayesian Information Criterion (BIC)* is one of the well-used functions which while scoring also penalize complexity. It is classified under the Information-theoretic scoring functions and is defined below, where the component preceded by the minus is the penalty. *N* and *d* are respectively the number of data cases and the number of parameters in the network.

$$\boldsymbol{\log p\left(D\middle|\hat{\theta}_G, G^h\right) - \frac{d}{2}\log N} \qquad (2.5)$$

From this expression, it is apparent that the BIC is independent of the prior, and clearly it shows both the expressions of reward and "punishment". In [37] it is noted that the BIC formula is the exact opposite (by minus) of the Minimum Description

[3] Likelihood is the conditional probability estimation of certain parameters, given certain observations; and ML is its maximization. Here, it is the selection, among a list of potential graphs, of the one which when observed, produces the highest probability of predicting the data.

Length (MDL), which is another well-known model selection criterion and is classified as Information-theoretic scoring functions according to [36]. In fact, the original intent of penalty is to reduce complexity, but it backlashes to the point that as result, BIC tends to choose models that are too simple, in order to escape load of "punishment". The other scoring function discussed by [19] is the *BDe metric*, which falls under the Bayesian scoring functions [36]. In this category, the computation is based on the evaluation of the posterior probability of a graph *G* given the data, and it is expressed as:

$$p(G^h|D) = \frac{p(D,G^h)}{P(D)} \qquad (2.6)$$

$P(D)$ is a normalized constant that is independent of the graph structure *G*. As such, $P(D)$ will has no impact the ordering of model options and therefore the relative posterior probability, $p(D, G^h)$, is generally adopted as model selector:

$$p(D, G^h) = p\ (G^h) * p(D|G^h) \qquad (2.7)$$

It is the product of $\boldsymbol{p(G^h)}$, the prior probability of graph $\boldsymbol{G^h}$ and $\boldsymbol{p(D|G^h)}$, the marginal likelihood. The prior could either be provided by an expert or just uniformly set. In the BDe metric which is Bayesian metric with Dirichlet priors and equivalence, networks are grouped by equivalence class and scores are assigned to groups of networks as opposed to individual network. Though Bayesian score is a more accurate criterion, it requires intensive computation [19].

### 2.3.3 Constraint-based-method

Based on Conditional Independence (CI) statistical tests, this method takes a different approach to the Bayesian Network structure learning problem. It proceeds

first by uncovering dependencies and conditional independencies among the variables. Based on three main assumptions [19], the discovered relationships are then used to infer and construct the corresponding BN structure. The first assumption, *causal sufficiency*, assumes that there are no common hidden parents of observed variables in the domain. The *causal Markov* assumption supposes that, given its parents, a variable is independent of all other variables that are not its descendants. Lastly, for a network structure $G$ and its associated probability distribution $P$, the *Faithfulness* assumption requires each verified conditional independence relationship in $P$ to be entailed by causal Markov assumption in $G$. Though with these assumptions, existence of an edge between two nodes is certain, there is no guarantee that all edges in the network would be directed. As result, the BN produced by Constraint-based method is a Partial DAG (PDAG) which is a set of Markov equivalent DAGs.

### 2.3.4 Hybrid Method

This method simply consists in the coupling of Constraint-based and Score-and-search-based methods. [37] suggests constraint-based be used as starting point to generate variables ordering and subsequently learn the actual Bayesian network with Score-and-search-based. This hybrid method is conducive to addressing concerns with datasets presenting large number of attributes and small sample size.

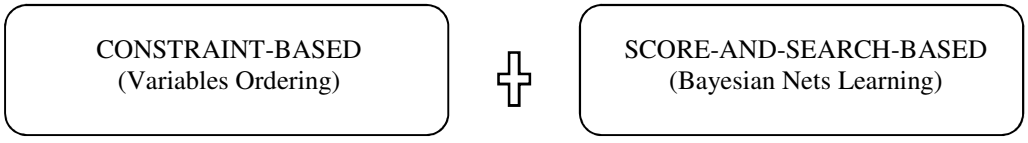



Figure 2.3: Illustration of hybrid method

### 2.4 Bayesian Inference

Bayesian Network is one of the network models that can be used to reason under uncertainty based on the laws of probability. It is a directed acyclic graph model containing nodes that correspond to random variables (discrete or continuous) as well as directed links that connect pairs of nodes. Each node has a conditional probability distribution given its parents, $P(Xi| Parent(Xi))$ which quantifies the effect of the parent on the node. Semantically speaking, Bayesian network can be viewed in two equivalent ways. The first is through the representation of joint probability distribution which is helpful in constructing the network and the second through an encoding of a collection of conditional independence formulations which is conducive to the inference procedures design. So, Bayesian network offers a compact/concise way to represent conditional independence relationship in the domain under modeling consideration and is mostly exponentially smaller than joint distributions that are explicitly enumerated.

Bayesian network helps to compute posterior probability distribution for a set of query variables for some observed event which consists of assigned values to a set of evidence variables. This posterior probability computation is referred to as inference. Based on the principles and techniques used, it could be exact inference or approximate inference [38]. Given the ambition of the first to be exact which turns out to be intractable in general, the latter whose principles and technique are summarized here, is therefore the method to consider as it sacrifices accuracy for computation time. The algorithms of approximate inference in Bayesian Networks are randomized sampling algorithms (Monte Carlo) and used in several branches of

science for the estimation of quantities that are hard to compute to exact values. There are in general two families of such algorithms: Direct sampling methods and the Inference by Markov chain simulation (MCMC Markov chain Monte Carlo) algorithms. In the first method, each sample is generated from scratch. MCMC on the other hand generates each of its samples by applying random change to the preceding sample. Direct sampling algorithms are indeed the simplest sampling methods for Bayesian Network. Basically, events are generated from a network that has no evidence associated with it, and each variable is sampled in turn in topological order, with probability distribution conditioned on values assigned to the variable's parents.

Direct sampling algorithms includes Rejection sampling and Likelihood weighing. In rejection sampling, after samples are generated from prior distribution specified by the network, all those that do not match the observed/given evidence are rejected and the actual estimate of $P(X=x|e)$ is the frequency of $X=x$ among the samples that remain. Likelihood is more efficient in that it only generates events that are consistent with the evidence *e*.

A particular form of MCMC called Gibbs sampling is well suited for Bayesian Networks. It proceeds with an initial state consisting of having evidence variables at their observed values and then generates a next state by randomly sampling a value for one of the none-evidence variables.

Bayesian Networks are globally defined by [28] as "graphical structures for representing the probabilistic relationships among a large number of variables and doing probabilistic inference with those variables". Once the Bayesian Network structure

is learned, the subsequent step is usually the Bayesian Inference. In real life applications, the theoretical scenario of identifying the sample space, determining the probabilities of individual events, defining the random variables and then computing the values of the joint probabilities distributions is not feasible. In such settings, the random variables are first identified and the probabilistic relationships among them are determined. Because the conditional probabilities of interest are usually not those directly computable, Bayesian Inference—which is based on Bayes' Theorem—is used to indirectly infer unknown probabilities given of the probabilities of other events. Practically, inference is determining the likelihood of a feature to be in a particular state. For two events E and F, with $P(E) \neq 0$ and $P(F) \neq 0$, the probability of E given F is given by Bayes' theorem as:

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)} \qquad (2.7)$$

Considering $n$ mutually exclusive and exhaustive events $E_1$, $E_2$, ... $E_3$ such that $P(E_i) \neq 0 \; for \; 1 \leq i \leq n$, for any other event F, Bayes' theorem is generalized as:

$$P(E_i|F) = \frac{P(F|E_i)P(E_i)}{P(F|E_1)P(E_1) \; + \; P(F|E_2)P(E_2) \; + \cdots + \; P(F|E_n)P(E_n)} \qquad (2.8)$$

Bayes' theorem is used when one is unable to directly determine the conditional probability of interest while the "inverse" conditional probability is known. For instance, having at hand—from data or from expert's knowledge—the probability that gene A is expressed given that gene B is expressed, as well as the independent probabilities of the expression of gene A and gene B respectively, one can determine by Bayes' inference (with equation 2.3) the unknown probability of interest, i.e., probability that gene B is expressed given that gene A is expressed.

## Chapter III - Methodology

A gene expression matrix resulting from gene response to low nutrient water is available from [9] and is based on PAO1 mini-Tn5-*luxCDABE* transposon mutant library. Overall 28 timepoints were measured. The attributes are of numerical type and thus confer to this research a quantitative methodology, with some qualitative considerations from Microbiology as background knowledge. The dataset, the hypothesis, the research objective and questions, as well as the literature review, all together contributed in shaping the architecture in Figure 3.1.

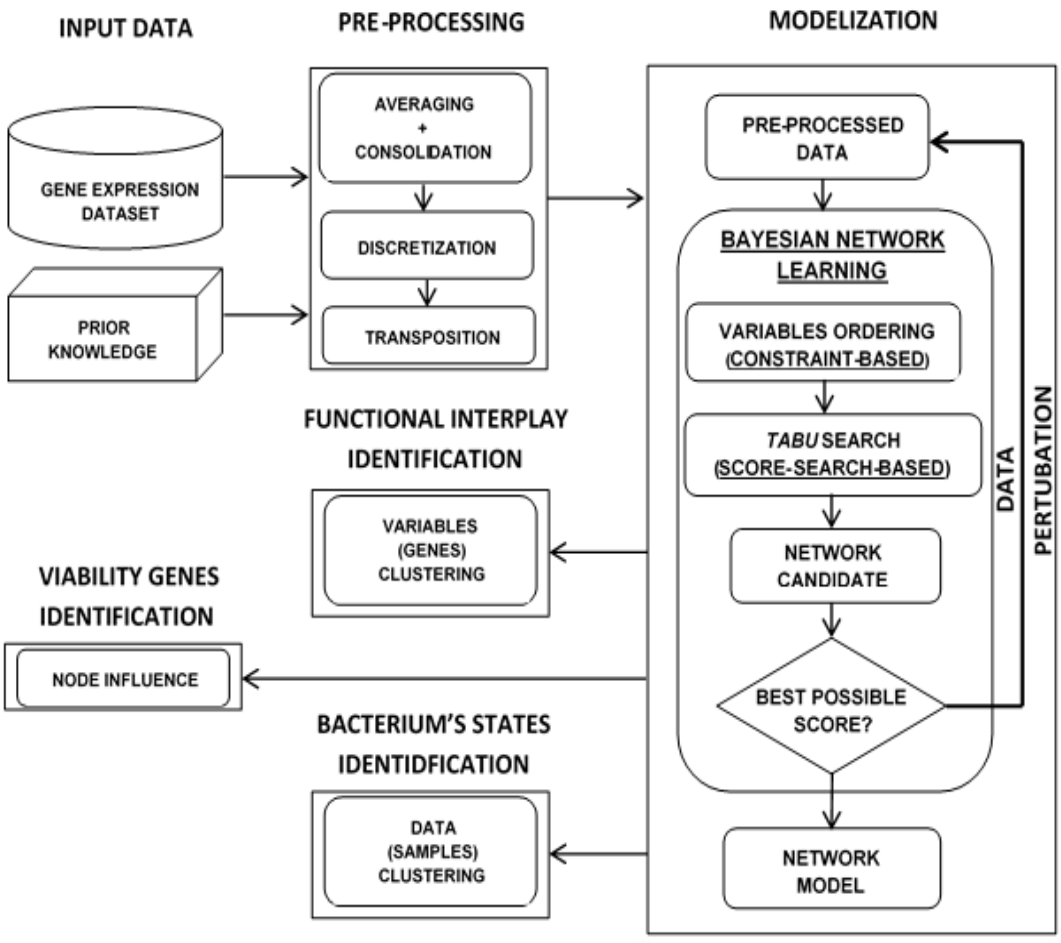



Figure 3.1: Methodology architecture

This architecture has at its center Bayesian Networks, a field of Machine Learning, for its capability to explore, to study, and to construct algorithms that can learn and make predictions and inference from data. Key technical challenges in this domain of research include massive number of variables, small sample size, resulting sparse networks, and also computational complexity [18].

The very first step of this research is knowledge acquisition in Microbiology, particularly about *Pseudomonas aeruginosa* and its genome. Among papers reviewed at this stage, [2, 6, 4, 39, 40] were fundamental, followed by the exploration of literature on genes expression analysis techniques. The publications [13, 16] are found to be foundational references on introductory biology and computer science tools for bio-analysis techniques. They are the bedrock for broader literature acquisition and literature review that was undertaken as presented in Chapter II.

### 3.1 Data Pre-processing

Pre-processing is of essence in data analysis. A misstep at this stage would propagate through the entire analysis and therefore requires attention. There are two datasets, one of which is illustrated in Table 3.1.

Table 3.1: Gene expression matrix—sample with timepoint fold-change values

| **Genes** | **Exprs_T4** | **Exp_T8** | … | **Exprs_T672** |
|---|---|---|---|---|
| PA5398 | 1.926365 | 1.427299 | … | 0.030316 |
| PA5400 | 2.138769 | 1.51678 | … | 0.048796 |
| … | … | … | … | … |
| … | … | … | … | … |
| Tgt | 1.06 | 0.64 | … | 0.73 |

Table 3.2 shows the two datasets that were combined to form the database used in

this research. $D_{10}$, a 10-timepoint was the initial dataset; and $D_{18}$, an 18-timepoint, was collected two month later, as a requirement for additional data samples.

Table 3.2: Comparison of the two datasets

| **Present in 10-timepoint ($D_{10}$)** | **Present in the 18-timepoint ($D_{18}$)** |
|---|---|
| × | emr |
| × | Intergenic point 90844 |
| × | Intergenic point 3418121 |
| × | Intergenic point 3621504 |
| PA0033 | × |
| PA2654 | × |
| PA4103 | × |
| PA4499 | × |
| PA5238 | × |
| PA5346 | × |
| PA5439 | × |
| PA5518 | × |
| × | retS |
| × | sadB |
| spuB-spuC | × |

Visually parsing the data, one can realize that some genes initially expressed become repressed over time and vice versa. This observation suggests that hierarchical "hand passing" exists between genes in their expression in order to ensure the agent survives over time. The following tasks were accomplished to ready the dataset for analysis.

Replicas Value Averaging: Each of the two datasets contains replicas of gene names. For each dataset respective clusters of variables were created with the same gene name. The name was associated to the corresponding cluster. The values of the gene replicas within a cluster were examined. In the case where a replica appears to demonstrate a wildly different expression in contrast to the other ones (Table 3.3), that replica was considered an outlier and therefore eliminated. The gene

expression values for the remaining replicas showing similar expression tendency were averaged for each time point and then labeled with the new cluster name.

Table 3.3: Outlier identification

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PA0497 | 0.23 | … | 0.08 | 0.07 | 0.06 | 0.05 | … | 0.12 |
| PA0497 | 1.40 | … | 0.08 | 0.11 | 0.12 | 0.61 | … | 1.20 |
| **PA0497** | **1.29** | **…** | **957.24** | **2787.51** | **1403.86** | **1.75** | **…** | **1.28** |

Data Integration: The comparison of the two datasets shows that they do not fully match up in gene variable. Several genes are exclusively present in either one. This can be attributed to a strain successfully growing in one trial and not in the other. This difference was captured in Table 3.2 above. For instance, *PA5518* present in $D_{10}$ was absent in $D_{18}$ while *retS* present in $D_{18}$ was absent in $D_{10}$. To consolidate these two data into a single uniform dataset, only the gene variables that are present in both were kept. $V_{D10}$ and $V_{D18}$ being gene variable sets, for each dataset, the resulting consolidated variables set is obtained by intersection and concatenated into $D_U$ (Equation 3.1). This resulted in 971 genes with 28 data samples each, for a (971x28) data matrix.

$$V_{DU} = V_{D10} \cap V_{D18} \qquad (3.1)$$

Discretization: The gene expression values in the original data were normalized respectively to the corresponding value at $T_0$. These normalized values represent genes expression fold change. According to biological importance in the experiment by [9], a gene is considered induced if it expression value is 2-fold or more, repressed if the fold change half or less, and neither expressed nor repressed if the fold change is within the rage ]0.5, 2[. On the basis of these three expression states

of a gene, this research proposes a ternary discretization {1,0,-1}, by assigning 1 to any gene expression value within the range of [2, +∞[, 0 to values in ]0.5, 2[, and -1 to expression values in the interval ]-∞, 0.5]. (See Algorithm 3.1).

**Algorithm 3.1**: Gene Expression Discretization Pseudo Code

**Input** (Continuous Dataset, Prior Knowledge of Expression Level)

If $G_{i,j} \geq 2$ Then $G_{i,j} = 1$

Else if $G_{i,j} \leq 0.5$ Then$G_{i,j} = -1$

Else $G_{i,j} = 0$.

**Output** (Discretized Dataset)

Transposition: The gene expression matrix $G_{n,m}$ of the original dataset has $m$ row vectors (gene expression profile $G_i$) and $n$ column vectors (sample expression profile $G_j$):

$$G_{i} = (g_{i1}, g_{i2}, g_{i3}, \ldots, g_{in}) \qquad (3.2)$$

$$G_j = \begin{pmatrix} g_{1j} \\ g_{2j} \\ g_{3j} \\ \vdots \\ g_{mj} \end{pmatrix} \qquad (3.3)$$

$$G_{n,m} = \begin{pmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{m1} & g_{m2} & g_{m3} & \cdots & g_{mn} \end{pmatrix} \qquad (3.4)$$

Since the focus of this research is on gene regulatory network and causality between genes rather than between experimental conditions, the gene expression profile is transformed into column vector and the sample expression profile is transformed into row vectors. This consists in applying transposition operation on $G_{n,m}$ to obtain the corresponding following transpose matrix:

$$G_{n,m}^{T} = G_{m,n} = \begin{pmatrix} g_{11} & g_{21} & g_{31} & .. & g_{m1} \\ g_{12} & g_{22} & g_{32} & .. & g_{m2} \\ g_{13} & g_{23} & g_{33} & .. & g_{m3} \\ .. & .. & .. & .. & .. \\ g_{1n} & g_{2n} & g_{3n} & .. & g_{mn} \end{pmatrix} \quad (3.5)$$

Table 3.4: Sample of the discretized transpose matrix

| PA5398 | PA5400 | … | Tgt |
|---|---|---|---|
| 0 | 1 | … | 0 |
| 0 | 0 | … | 0 |
| … | … | … | … |
| -1 | -1 | … | 0 |

### 3.2 Learning Strategy of Bayesian Network Model

Though the data $D = \{D_1, D_2 \ldots, D_N\}$ is available as a set of temporal tuples, at this stage, each time point (Tuple) is treated as an independent observation and not as time series data. In this section, we focus on learning and constructing a *static* Bayesian Network $B = (G, \Theta)$, $G$ being the graph and $\Theta$ the set of parameters characterizing the network. Each node in this BN represents a gene (e.g., *Tgt*), characterized by a conditional probability table that specifies its probability distribution, conditioned on its parents' values. This distribution over the gene variables ($G_i$) is defined by the *chain rule* identity as in Equation 3.6 [38].

$$P(G_1 = g_1 \ldots, G_n = g_n) = P(g_n | g_{n-1} \ldots, g_1) P(g_{n-1} | g_{n-2} \ldots, g_1) \ldots P(g_2 | g_1) P(g_1)$$
$$= \prod_i P(G_i = g_i | g_{i-1}, \ldots, g_1)) \quad (3.6)$$

BN is very suitable for its factorial representation of states. Its probabilistic approach is of interest since gene expression is intrinsically a stochastic phenomenon which is generally affected by high noise level [41]. Further, [41] states that BN can be used for causal reasoning i.e., *P(effect|cause)*, diagnostic reasoning i.e.,

*P(cause|effect)*, combination of both, or even for "explaining away". The fundamental of Bayesian networks is Bayes' Rule (Equation 3.7) that enables both causal and diagnostic reasoning.

$$P(cause|effect) = \frac{P(effect|cause)P(cause)}{P(effect)} \qquad (3.7)$$

In a diagnostic model graph, the links are directed from symptom to cause while they are directed from cause to symptom in a causal model graph. This latter representation requires fewer dependencies specification than the former in which additional dependencies are specified on one hand between causes that are actually independent and on the other hand between symptoms that occur separately [38]. In areas such as medical diagnosis, doctors usually have the causal information i.e., *P(symptoms|disease)* and infer diagnosis, which is *P(disease|symptom)*. As cited by [38] on page 517, [42] shows that expert physicians preferably adopt causal rules over diagnostics rules in their probability judgment. Therefore, in this thesis work, the evidence/symptom is *bacteria survival in a certain environment* with *a gene or group of genes* being the probable cause. This requires the derivation of *P(bacteria survival in a certain environment | a group of genes)* and *P(a group of genes | bacteria survival in a certain environment)*.

The graph model will be learned from data, based on causal rules. The model construction learning task consists of finding a set of parent genes for each gene, in a way that the resulting network is acyclic and its score minimal [43]. Figure 3.2, as discussed in [36], summarizes the standard skeleton of the learning methodology.

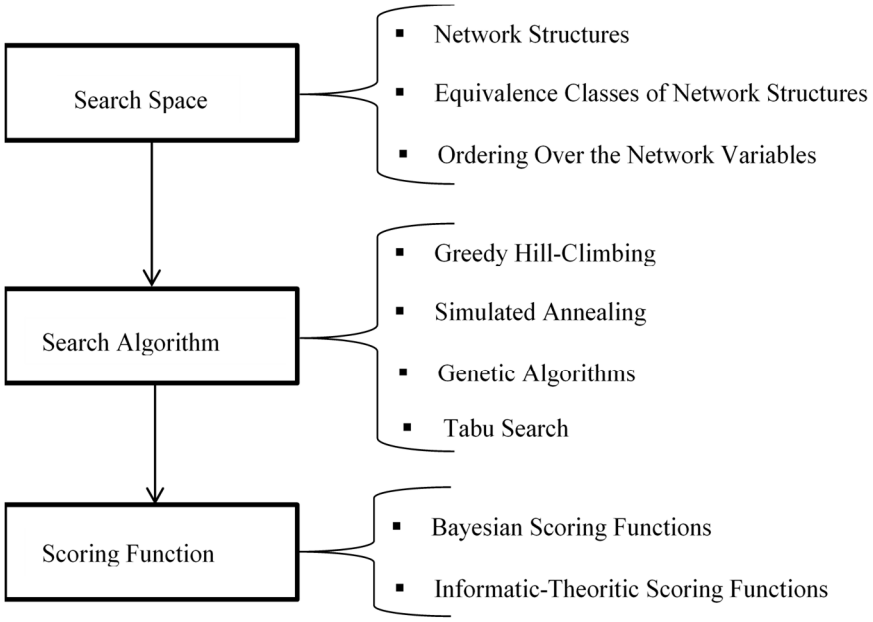



Figure 3.2: Standard Bayesian Network learning methodology

### 3.2.1 Choice of the Search Space

Table 3.5, excerpt from [41], presents the types of structure learning problems along with corresponding applicable methods. As in many real-world cases, the structure for this research is unknown and is to be learned. The preprocessed data shows a full observability, thus pointing to the type "Unknown Full". Therefore, *Search through model space* has been selected as the learning method. This corresponds to the *Network Structure* option in Figure 3.2 above.

Table 3.5: Learning methods based on learning problems

| **Structure/Observability** | **Method** |
|---|---|
| Known, Full | Sample statistics |
| Known, Partial | EM or gradient ascent |
| Unknown, Full | Search through model space |
| Unknown, Partial | Structural EM |

### 3.2.2 Choice of the Search Strategy and Model Selection

Constraint-based algorithms use statistical analysis and learn the network's structure with conditional independence tests (e.g., $\chi^2$ test) to determine the existence of edges between node variables. This results in a unique model based on categorized information. Score-based on the other hand examines all possible structures in the space and assigns a score that corresponds to the measure of the "goodness" of the Bayesian Network's evaluation of a given dataset. As far as efficiency is concerned, constraint-based methods are said to do well with large number of variables dataset ($\cong 1000$ genes as variables) while score-based algorithms give more accurate results with small sample size dataset (here $\cong 30$ gene expression time points). Furthermore, the Bayesian score-based method presents more advantages over constraint-based. For instance, it can overcome incorrect conditional independency categorical decisions, by model averaging. It also deals with missing data and is able to find models that constraint-based is incapable of detecting [28]. The third method known as *Hybrid method* consists in the combination of both methods afore-discussed and is generally suggested as the alternative that offers the advantages from both of its constituents. In addition, these constituents are particularly and respectively in adequacy with the two main dimensions of the dataset (i.e., large number of variables and small sample size). Consequently, it follows that this research adopts the *hybrid method*: Constraint-based to obtain variables ordering and network skeleton, and Score-based to identify arcs direction.

a. Constraint-Based: Genes Ordering

Depending on the domain and the availability of background information, prior knowledge is primarily used to complete variables ordering. Considering domains with small number of variables, [37] suggests that grouping the variables into generic classes such as symptoms or diseases is a practical approach to reduce the orderings cardinality of the domain, without recourse to highly greedy heuristics. In fact, [44] has shown this in a medical application with 10 nodes, dividing them into "blocks." But when expert's knowledge is unavailable, and the number of variables is very large, ordering is quasi unachievable with brute force search through all $n!$ possibilities in the quest of the best combination Order-Network. This is the case here, as this research deals with a large number of variables ($\cong 1000$ genes), and to date, there are no available knowledge on existing pre-established genes ordering of *P. aeruginosa*. The authors of [37] implemented and successfully tested ***CB***, an algorithm that uses a Conditional Independence (CI) test to suggest a total order of the nodes which is then used by a Bayesian algorithm to learn and construct the best network.

b. Score-and-search-based: Taboo Search

According to [38], search in general is a process of finding a series of actions whose execution would lead to the goal of solving a given problem. Based on the application and the environment in which a search agent is operating, there are different types of search. The two main currents of search algorithms are *classical informed search* which is an offline search, and *online search*. Informed search, referred to as Heuristic, is a category of search algorithms that uses goal-based agent,

precisely, a problem-solving agent. Uninformed/Blind search is its counterpart. Unlike this, informed search uses problem-specific knowledge plus the problem's definition information. This is in the form of an evaluation function *f(n)* which estimates the distance/cost to the goal. This function is the source of the efficiency of informed search over uninformed search, in the sense that it is guided by the knowledge pertaining to where to look for potential solutions.

Basically, for every node *n* in the search space, the function *f(n)* quantifies the desirability (cost effectiveness or more promising state) to expand the node and uses the results to decide which node to expand next. A typical example of informed search is *A* search*, the most widely known form of *Best-first search.* This search minimizes the cost *f(n)* which combines the cost *g(n)* to reach a node, and *h(n)* the cost to go from the node to the goal, with $f(n) = g(n) + h(n)$. While informed search is classified as offline problem-solving search with known, observable, discrete, deterministic environment, and complete solution computation prior to execution in real world, the Online Search agent in contrast interleaves computation and action and operates in an originally unknown state space, with nondeterministic environment, thus requiring precepts.

Online search is appropriate for exploration problem, using its action as experiment for learning purpose [38]. Typical example of online search is a robot agent that is placed in a new building and tasked to explore it in order to build a map that it can use to get from point A to B. Beyond the classical informed search, there is *local search*. The term local here alludes to the neighborhood within which search is

operated. It proceeds to its quest of solutions by evaluating and modifying current state instead of moving systematically through path exploration from the initial state. It relaxes the assumptions made in informed search and is more appropriate to problems in which only solution matters and not the path and cost that lead to it.

Within this family of local search are the following notable algorithms: *Genetic Algorithms (GA)* from evolutionary biology and *Simulated Annealing (SA)* from statistical physics. Both are *Hill-Climbing* (Figure 3.3) based search which continuously moves to uphill states in the direction of increasing value and stops when it attains a peak where none of its neighbour's states has a higher value. The concern with such algorithm is that it never makes "downhill" and is condemned to be incomplete as it can get stuck on local maximum while global maximum exists.

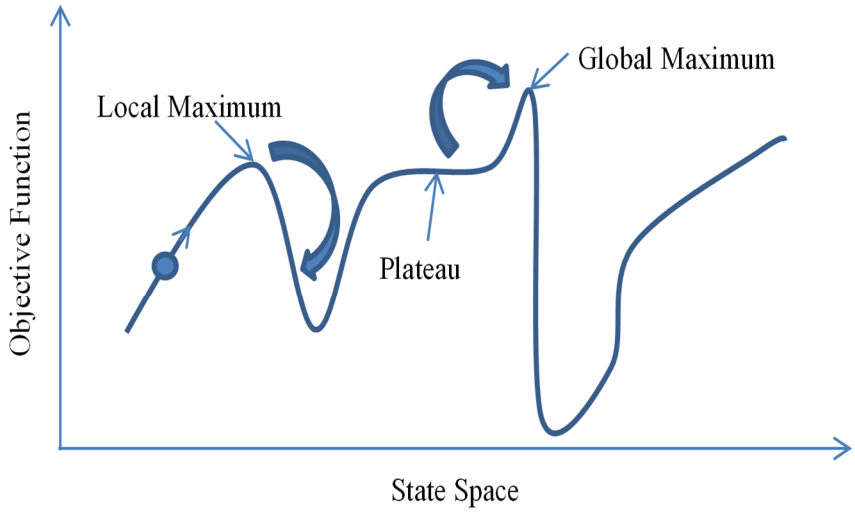



Figure 3.3: Graphical illustration of local maximum issue with hill-climbing

Though extremely inefficient, a purely *random walk* with uniform and random moves among a set of successors, is complete [35]. This research does not employ GA, a stochastic hill-climbing search in which new states are generated by mutation

and by crossover, combining pairs of states from the population of states. The core technique of simulated annealing on the other hand allows it to overcome local minimum concerns. Since hill-climbing algorithm can get stuck in a local maximum/minimum (i.e., incomplete), it is combined with the random walk to take advantage of its completeness and to make simulated annealing algorithm overcome the local minimum concern of hill-climbing. Simulated annealing is inspired from metallurgy annealing which is the tempering/hardening of metal or glass by heat to a higher temperature and then proceeding to a gradual cooling down allowing the material to reach low-energy crystalline state. Another perspective of the core technique in hill climbing is gradient descent in terms of cost minimization, which is similar to the task of getting a Ping-Pong ball into the deepest fissure in a bumpy surface. By letting the ball roll, it will come to rest at a local minimum. Now by shaking the surface, the ball can be bounced out of that local minimum with the trick being, to shake it hard enough as not to get it out of the reach of the global minimum. In summary, the procedure is to start by shaking the surface hard (or raise the temperature to a higher level in case of metallurgy) and then progressively reduce the shaking force (lowering temperature). Assuming a minimization objective, and considering the stochastic aspect of simulated annealing, the probability at the start to move from current state to another one of higher value is near zero. But as it evolves, the probability increases for a move to a higher neighbour value, to avoid being stuck at a local minimum.

Practically, simulated annealing is a metaheuristic method used in discrete and very large search space environment. According to [33], the cardinality of DAGs search

space for a BN selection is function of the number of nodes (Formula 2.2).

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} C_i^n 2^{i(n-i)} f(n-i)$$

$$f(2) = 3$$

$$f(3) = 25$$

$$f(5) \cong 29 \times 10^3$$

$$f(10) \cong 4.2 \times 10^{18}$$

$$f(47) \cong 9.0 \times 10^{376}$$

$$\boldsymbol{f(1000)} \cong ??? \times \mathbf{10}^{???}$$

For instance, with *n=10* i.e., $4.2 \times 10^{18}$ as search space size, a brute force approach will require years of computation time, even on a supercomputer [43]. Given $n \approx 1000$ genes in this research, searching within this myriad of possible space DAGs can only be effectively tackled as an optimization problem. With this huge cardinality search space problem, the methodology must be grounded on heuristics, specifically metaheuristic. Like simulated annealing, Taboo Search is a metaheuristic iterative algorithm for combinatorial optimization. It considers a potential solution candidate to a problem and aspires for possible improved solution in the immediate neighbourhood. Though it inherits from the general local search the tendency to get stuck in a suboptimum or on plateaus, it escapes this trap by accepting solutions worse than the current. The other key aspect to which it owes its name is the *prohibition* (taboo in *Polynesian*) for the algorithm to go back to previously explored solutions within a defined time frame or number of iterations. Cycle is thus prevented through the maintenance of a *Taboo list*. Taboo search has proven successful

in a wide range of applications, from resource planning to molecular engineering, through biomedical analysis. Several comparatives studies have demonstrated that it outperforms simulated annealing and GA. For instance [45] experimentally proved that Taboo Search presents the best performance for the quality of solution and the quality of the solution subspace. It is said to require less implementation complexity and parameters tuning. This is corroborated by [46] in their 2006 empirical study. Considering all these, this research adopts Taboo search.

c. MDL as Scoring functions for learning Bayesian networks

Table 3.6 captures the elimination process in the selection of scoring function. The dataset available for this research is raw data with unknown underlying prior probability distribution. According to [36] and [47], all *Bayesian Scoring* functions require prior probability distribution this restricting the selection to Information-Theoritic scoring functions.

Table 3.6: Summary of scoring functions for learning BN

| **Scoring Functions** | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Name | | PPDD[4] | BoEC[5] | Formula | Decomposable /Score-Equiv. | Network Tendency |
| Bayesian Scoring | BD | | ✓ | × | - | - | - |
| | K2 | | ✓ | × | - | - | - |
| | BDe | | ✓ | × | - | - | - |
| | BDeu | | ✓ | × | - | - | - |
| | | | | | | | |
| Info-theoretic Scoring | LL[6] | AIC | × | ✓ | $LL(B\|T) - \|B\|$ | Yes/Yes | Over-fitting |
| | | BIC | × | ✓ | $LL(B\|T) - \frac{1}{2}\log(N)\|B\|$ | Yes/Yes | Under-fitting |
| | | MDL | × | ✓ | $-(BIC)$ | Yes/Yes | Trade-off |
| | | NML | × | ✓ | $LL(B\|T) - \boldsymbol{C}_N(B_G)$ | No/Yes | - |
| | MIT | | × | ✓ | - | Yes/No | - |

[4] Prior Probability Distribution Dependant
[5] Based on Encoding Compression
[6] Log-Likelihood

AIC stands for Akaike Information Criterion, and BIC for Bayesian Information Criterion. Both have the disavantage respectively of over-fitting and under-fitting while MDL, which stands for Minimum Description Length, offers a trade-off. Though MDL and BIC are additive inverses, their respective derivation principles are different. In addition to the trade-off advantage, [47] justified the use MDL in the presence of raw data, as it does not require known prior distribution assumption. NML—Normalized Minimum Likelihood—which is based on MDL, is not decomposable over network structure and involves an exponential sum over all possible data of size N, with no hope for efficiently computing $\boldsymbol{C}_N(B_G)$ [36]. Because decomposability is a key computational requirement factor, overall it comes down to MDL and MIT—Mutual Information Test. Though score-equivalence is not demanded for this research, MDL has been opted over MIT which is not score-equivalent.

As per formula column in Table 3.6, MDL has two components. The first is the number of bits to represent the model i.e., graph + probabilities in the Bayesian Network *B*. The minimum value of this component corresponds to the simplest network, which is the fully unconnected network, all genes being independant from each other. The second component is the number of bits to represent the learning data *D*, given *B*. In other words, it is the likelihood of data *D* given the model. The minimum value for this latter component represents a fully connected notwork. The goal of MDL is to minimize the sum of its two components. Practically this amounts to finding a trade-off between the two terms.

### 3.3 Data Perturbation

To further ensure that the search methodology escapes local minimum during the learning process, Taboo search is supplemented with Data Perturbation, where random noise is added to the weight of each observation in the dataset. This provides a reasonable confidence of finding the optimal network, i.e., the most compact representation of the joint probability distribution over the 954 genes.

### 3.4 Summary of the Algorithm

Algorithm 3.2 below summarizes the overall BN learning methodology.

**Algorithm 3.2**: Hybrid search (Nodes ordering + Taboo Search + Data Perturbation)

```
Input: pre-processed Dataset D(n genes, N data samples)
   [G1 ... Gn] <- generate best node order with heuristic
         CI test over n! possible nodes ordering space;
   For i = n...2 {{Pa(Gi) <- k parents ∈ {Gi-1...G1} :
         P(D| Pa(Gi)) is max & P(Gi|Gi-1 ...G1) = P(Gi| Pa(Gi));}
         For j = 1...k { Edges[] <- Connect(Pa(Gi)j -> Gi);
           CPT(Gi)<- P(Gi| Pa(Gi)j;}//CPT = Cond. Prob. Table
   gInitCandidate <- graph({G1 ... Gn},Edges[],CPT(Gi));
   f(n) <- #(Candidates in search space) function of n;
      H(f(n))<- TabooListSize from Heuristic evaluation H;
   TDP(H(f(n))<- Taboo(H(f(n))) + DataPerturbation(D);
Output: gBest = TDP(H(f(n)), gCandidate, MDL))
```

First, the data is preprocessed, and the resulting *n* gene variables are ordered by heuristic conditional independence test over *n!* possible nodes ordering space. Then, in the order thus established, each gene $G_i$ (as a graph node) respectively is assigned a set of *k* parents denoted $P_a(G_i)$ and chosen from $\{G_{i-1}...G_1\}$. For a parents set $P_a(G_i)$ to be assigned as such to a gene $G_i$, the probability of the data $D$ given $P_a(G_i)$ should be the maximum possible $(P(D|\,Pa(Gi))_{max})$ and the probability of the gene $G_i$ given the set of all the other genes that precede it should be equal

to the probability of that gene $G_i$ given his parent set, i.e.; $P(G_i|G_{i-1} \ldots G_1) = P(G_i| P_a(G_i))$. A directed link connects each parent from the parents set to $G_i$, and a Conditional Probability Table (CPT) is associated to the gene $G_i$. The CPT for a gene $G_i$ is an array of its respective probabilities given each of its *k* parents. The graph thus formed by the connections between genes is the initial Bayesian Network candidate which is optimized through *Taboo* search by heuristic evaluation. The number of all possible candidates for the search is a function *f(n)* of the number of genes *n* as in formula 2.2. The Taboo search, a combinatorial optimization—combined with Data Perturbation—considers, in the immediate neighborhood of the current graph candidate, a potential solution candidate that yields a better score according to the MDL metric. A neighbor is any graph obtained by one single operation, i.e., adding, deleting or reversing an arc on the current graph structure. At the end of the iterative heuristic search, the optimal Bayesian Network is returned as output.

### 3.5 Inference from Learned Model: Identifying Low Nutrient Response Genes

*Pseudomonas aeruginosa* survival in water was measured several times and it is known that it survives in water without nutrients for a very long time, even up to 8 weeks [5]. Some researchers [48] have tested 148 bacterial strains including *P. aeruginosa*—a Gram-negative organism—and found that almost all the Gram-negative bacteria survived in water for at least 30 weeks and up to 16 years. This research had access to gene expression from *P. aeruginosa* existence in nutrient depleted water for over 4 weeks period, and it had been experimentally shown that the

bacterium was alive at one-month timepoint [9]. Also, survival experiments were completed up to 3-4 months whereby the concentration of cells was very close to the concentration used at the start of the experiment ($1 \times 10^7$ colony forming units/milliliter).

A colony starts from a single cell. It is generally difficult to measure single cells, so the samples had been plated and colonies counted on a petri plate. It had been shown that there was some variability within the population of cells and this had been demonstrated by flow cytometry whereby the characteristics of single cells could almost be determined with fluorescent dyes. With this, it had also been shown that the cells were dormant [9], but not dead over time, and 28 timepoints were recorded in the current data.

Due to the variability afore mentioned some cells that were likely dead near the beginning of the time course appeared to transition and adapt to being more dormant over time. It had been proven that the cells were still alive and viable as well, even though they are not metabolically active or replicating. So, one could hypothesize low, medium, and high life states for the cells. High state would likely be growing and replicating cells, which was not much observed in water. Given it was experimentally proven that *PA* was alive (but possibly at various stages e.g., high, low, dormant) the phenotype embodied in the data is "survival". Having learned the BN, the next step is to identify major contributors to the survival phenotype.

Among all nodes in the network, the strongest genes that, in one way or another contribute to the survival of *P. aeruginosa* are determined. Norsys [49] terms the

nodes in a network that are at the very top (root) as "predispositions", to whom the likelihood of an observed phenotype is attributed. All root nodes are considered as potential cause of survival and from these the strongest would be identified. Also, any non-root nodes appearing to be pivotal in the maintenance of the observed phenotype will also be considered as potential cause of survival. To achieve this, the notion of node influence has been applied. The weight of a node's influence in a network is directly related to the strength of its incoming and outgoing arc(s).

In [50] two ways of assessing the strength or importance of an arc were discussed:

- Conditional probability tables in the definition of the model are used to determine the importance/thicknesses [51] of the arcs. In this, the magnitude of the influence transmitted by a directed edge from node *A* to node *B* is given by:

$$\max_k[\max_i[P(B \geq b_k|a_i) - P(B \geq b_k|a_0)]] \qquad (3.8)$$

- *Kullback-Leibler Divergence* measure [52] defined by equation 17:

$$D_{KL}(P(X)||Q(X)) = \sum_X P(X)\log_2\frac{P(X)}{Q(X)} \qquad (3.9)$$

This latter measure is a generalization, to the abstract case, of the definition of information according to *Claude Shannon*. On a broader scale, it concerns with the evaluation of statistical distance or divergence between populations. In Bayesian statistics, it is used to evaluate the amount of information gained, from a prior distribution to a posterior distribution. This was used in [53] to evaluate the strength of an arc in a network and is named *Arc Force*. It compares a current network *P* with a hypothetically identical network *Q* but minus the arc whose force is under

evaluation. *Mutual Information* is a particular case of arc force when the descendant node only has one parent. Also arc force supersedes mutual information because in its computation it operates with global network distribution as opposed single bivariate relationship. While [51] provides a good view of how two directly connected variable nodes interact, the relevance of the information rendered is not only just local but also static in that observations or indirect influences are not considered. The Arc Force on the other hand is dynamic, factoring in observations and indirect influence. *Arc Force*, i.e., node Force, has been adopted to evaluate the genes' participating *force* in the regulatory network of the survival. Technically, given a directed edge $E_{i->j}$ between two genes, $P(G)$ the probability distribution of our learned network model, and $Q(G) = P(G) - E_{i->j}$, then the $Arcforce(E_{i->j}) = D_{KL}$ (Equation 3.10) is the measure of information gained from $Q(G)$ to $P(G)$. In summary, the identification of low nutrient response genes approach consisted first in searching through the graph to identify all root nodes, i.e., all nodes with no parents but having children, denoted as $R_G$. Then for each gene $G_i$ in the entire network, its $force(G_i)$ was considered as the sum of all its associated $Arcforce(E_{i->j})$. With $S_G$, the set of the top strongest genes in *force*, the viability genes set has been inferred as: $V_G = R_G \cup S_G$. The choice of using both root node and node force as criteria is based on the hypothesis that there exists a hierarchical "hand passing" among genes in their expression to ensure *P. aeruginosa* survival.

### 3.6 Development Environment and Configurations

A list of development/analysis platforms have been explored and four of them have been tested as shown in Table 3.7. Though Netica could handle a large number of

variables, it cuts off any variable number beyond 250. The choice criteria of the four platforms are also summarized in Table 3.7.

Table 3.7: Summary of analysis platform selection

| Implementation Platforms | BN | DBN | Learning | Inference | Scalability #variables | Visualization & interactivity |
|---|---|---|---|---|---|---|
| WEKA | ✓ | × | × | × | × | × |
| R Studio | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| Netica | ✓ | ✓ | ✓ | ✓ | ✓* | ✓* |
| BayesiaLab | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

BayesiaLab is a virtual lab environment platform, with Bayesian Networks at its center. It supports research process through modeling, analysis, simulation and optimization. Developed by a pair of French Professors in early 2000s, it provides the scientists and researchers with litheness in manipulation and movement between the different tasks of the overall investigative work. The most valuable and determinant aspect of BayesiaLab for this research is the impeccable visualization and interactive capabilities it features. This research uses BayesiaLab for development and analysis.

### 3.6.1 Determining optimal number of parents per node

To determine the analysis' parameter pertaining to the number of precursors (parents) per gene, *Theorem 3.5.1* [54] has been used.

**Theorem 3.5.1:** *In an optimal Bayesian network based on MDL scoring function, each variable has at most* $log\left(\frac{2N}{log\,N}\right)$ *parents, N being the number of data points.*

The number of data samples in this research is *N=28*. If $P_a(G_i)$ is the parents set for a gene $G_i$, the cardinality of $P_a(G_i)$ should respect the following inequality:

$$|P_a(G_i)| \leq log\left(\frac{2 * 28}{log\ 28}\right) \cong 1.60 \qquad (3.10)$$

This means that the scores for any parents set size larger than "1" will not be computed given that here one parent in the parents set is guaranteed to be suboptimal.

### 3.6.2 Data Import and Variables Elimination

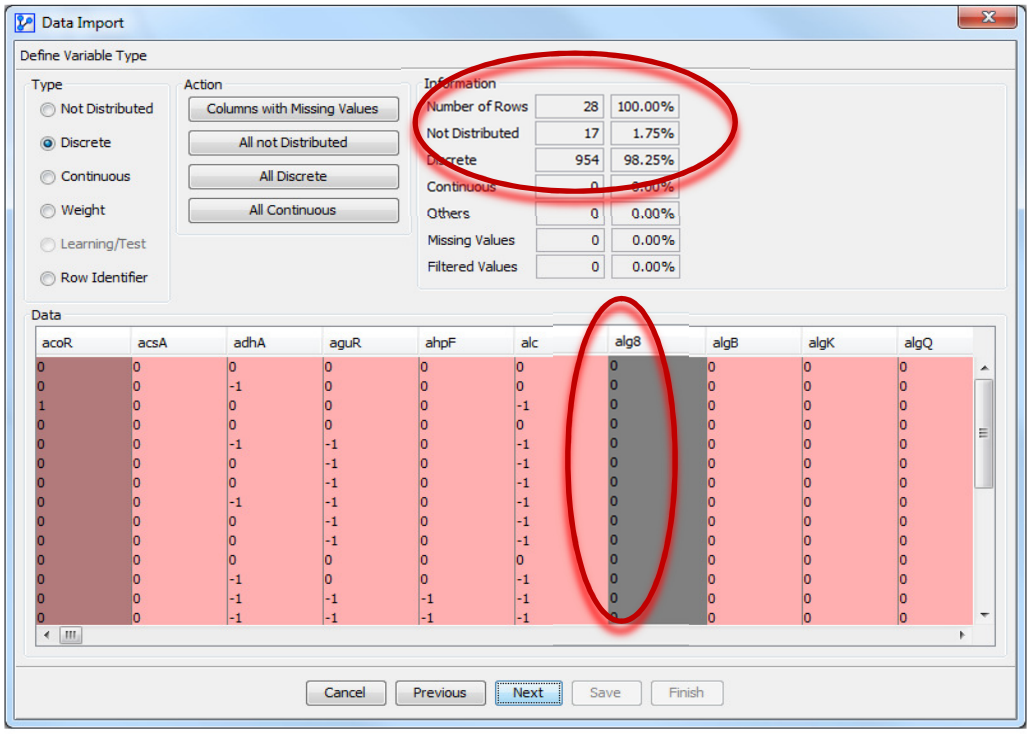



Figure 3.4: Data import screen capture

After the dataset has been imported as illustrated in Figure 3.4, a group of 17 genes was eliminated because they were not distributed in the data. As one can observe for *alg8*, each of its 28 discretized values is equal to zero, and therefore not distributed. As result the 971 gene variables initially input have been brought down to

954. Table 3.8 shows the possible discretized states of a few gene variables.

Table 3.8: Discretized states of some gene variables

| Nodes 954 | | | |
|---|---|---|---|
| *cyoB* | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| *aspA* | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| PA0272 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| | | 1 | 1 |
| PA5507 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| | | 1 | 1 |
| *cupC2* | Discrete | **States** | **Aggregates** |
| | | 0 | 0 |
| | | 1 | 1 |
| PA0377 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| PA3265 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| *cysW* | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| | | 1 | 1 |
| PA4715 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| | | 1 | 1 |
| PA5270-PA5271 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| | | 1 | 1 |
| PA0752 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| PA5001 | Discrete | **States** | **Aggregates** |
| | | -1 | -1 |
| | | 0 | 0 |
| ..... | ..... | ... | ... |

### 3.6.3 Networks Structural Coefficient Setting

The formula of the MDL scoring function in BayesiaLab [52] is given by:

$$MDL(B,D) = \alpha DL(B) + DL(D|B) \qquad (3.11)$$

In this equation (3.11), the factor α represents BayesiaLab structural coefficient. *DL(B)* and *DL(D|B)* respectively are the structural complexity of the network graph, and the adequacy of Bayesian network to the data. Associated to the structural complexity term of MDL, the parameter (α) which ranges in [0-150], is by default set to 1. Smaller α implies greater complexity requirement for the graph. Conversely, the complexity requirement is lower for higher α values.

The structural learning commenced with α =1. Due to this complexity stringency and a very small 28 data samples coupled with the 954 variables, the algorithm ran continuously for 5 days till it was aborted. The experimental selections of values then were set in the interval of α ϵ [3-150] that produced the least complex graph i.e., fully unconnected graph, for structure learning. This was followed an increment of structural coefficient value from 1 to *1.2884025* as shown in Figure 3.5.

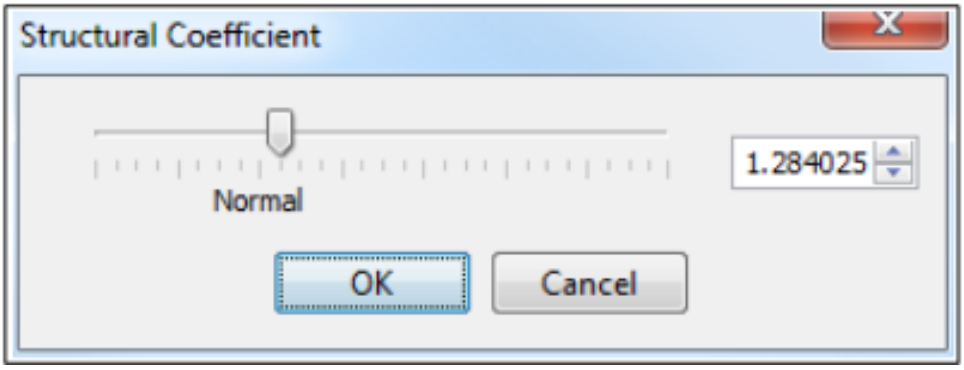



Figure 3.5 Structural coefficient selection

## 3.7 Explorative Approach to the Additional Research Questions

### 3.7.1 Exploring Functional Modules in *P. aeruginosa* Survival

Genes expression is not only an inherent stochastic phenomenon, but also is hierarchical in nature. This led to the postulation of the existence of higher-level "latent" variables—not directly measured in the gene expression trials—that would represent genes functional modules. To investigate these modules, by *Multiple Clustering*, relationships were further analyzed within groups of genes that made up the learned survival mechanism model. From the learned BN model, through *Variable Clustering*, groups of strongly connected genes were computationally segregated and then it was posited that some hidden common causes, respectively, were the underlying factors of the strong intra-cluster connections obtained in the gene clusters. Thereafter, for each cluster, *Data Cluster* was used to induce a latent variable that is to represent the common cause. The newly obtained variables—*Factor_i*—were then hypothesized to be the genes functional modules (e.g., 107 in Table A.2) and are to be further studied with domain experts in future works.

### 3.7.2 Exploring *P. aeruginosa* States

The dataset of this research is the collection of the representation of the expression level of *P. aeruginosa* genes. So, the expression level is a factor that characterizes the gene's state as an entity. This perspective instigated the speculation of the existence of an overarching factor which would characterize the bacterium as a higher-level entity. Namely a factor which by computation would indicate the bacterium' states over time (e.g., active or dormant). These two states were assumed to be the only states, followed by data clustering (See Figure A.3 and Table A.3).

# Chapter IV – Results, Evaluations and Discussions

## 4.1 Learned Network Model and Analysis

The computation time to learn the BN model lasted ~ 300 hours on a 4 GB RAM Windows computer. Figure 4.1 shows the overall BN learned from data and Figure 4.2 presents its evaluation. As it can be observed, the network is fragmented with one fragment conspicuously standing out (see green oval in Figure 4.1). The other fragments include very small networks of 2-10 nodes and almost 70% of the genes are unconnected and regarded as noise. One of the original assumptions was that not all the genes would participate in the survival phenotype. That is, not all genes would be included in the model. Initial plans included the use of PCA for dimensionality reduction, which at a closer look was not adequate for the purpose.

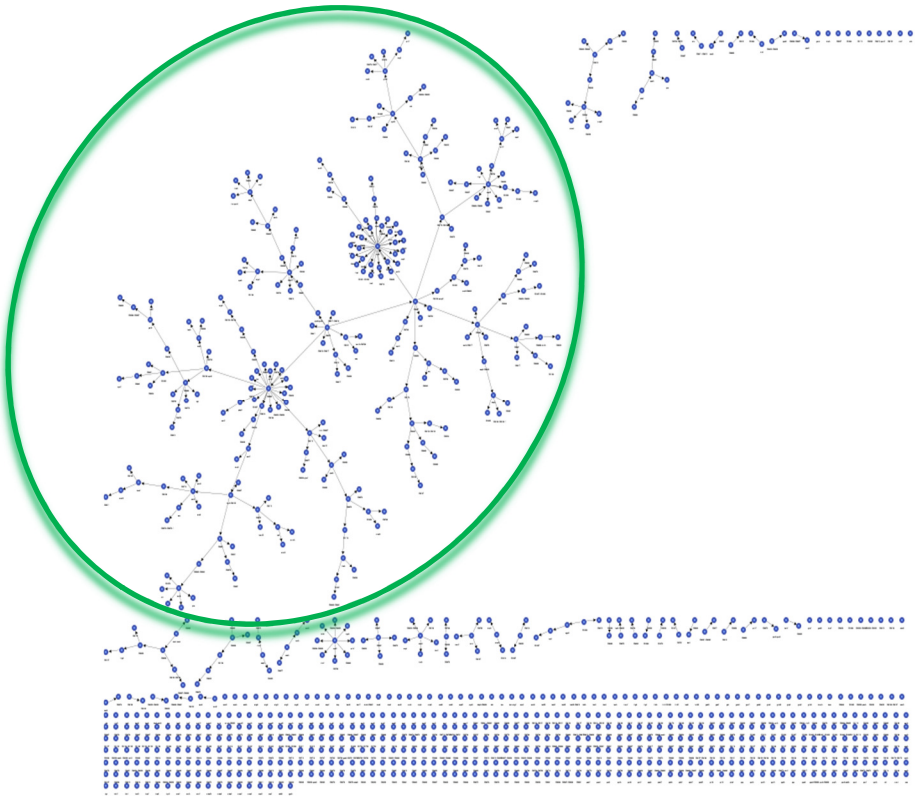


Figure 4.1: *P. aeruginosa* overall learned Bayesian Network—fragmented

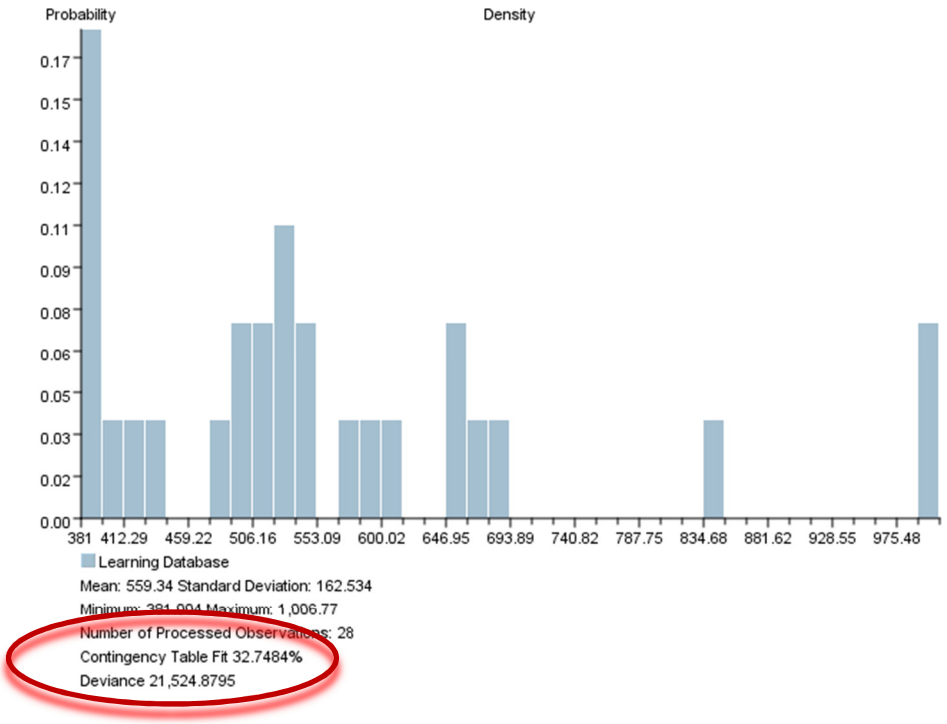



Figure 4.2: Evaluation of the overall network with the Contingency Table Fit criterion

With the Bayesian learning, there were ~25% of the overall 954 genes in the main network. Gene expression networks are generally sparse [18], i.e., only a small number of genes directly affect each other. So, the BN learning reduced the dimensionality to ~ 250 informative gene variables in the actual *P. aeruginosa* survival mechanism model (Figure 4.3). In most applications of gene expression, the number of data instances is relatively small. Because of the statistical challenge of large number of variables and small number of samples, one cannot discriminate among all possible models, since the small amount of data is not enough to identify one single most probable model. Though the study in [18], which is based on *score-and-search* method, uses 48 more data samples (76 instances for 800 genes) than in this research, the hybrid approach adopted here theoretically compensates for that difference—constraint-based methods do well on large number of variables

dataset while score-based gives more accurate results on small sample size dataset. Even so, one should note that larger sample sizes confer more reliability to models.

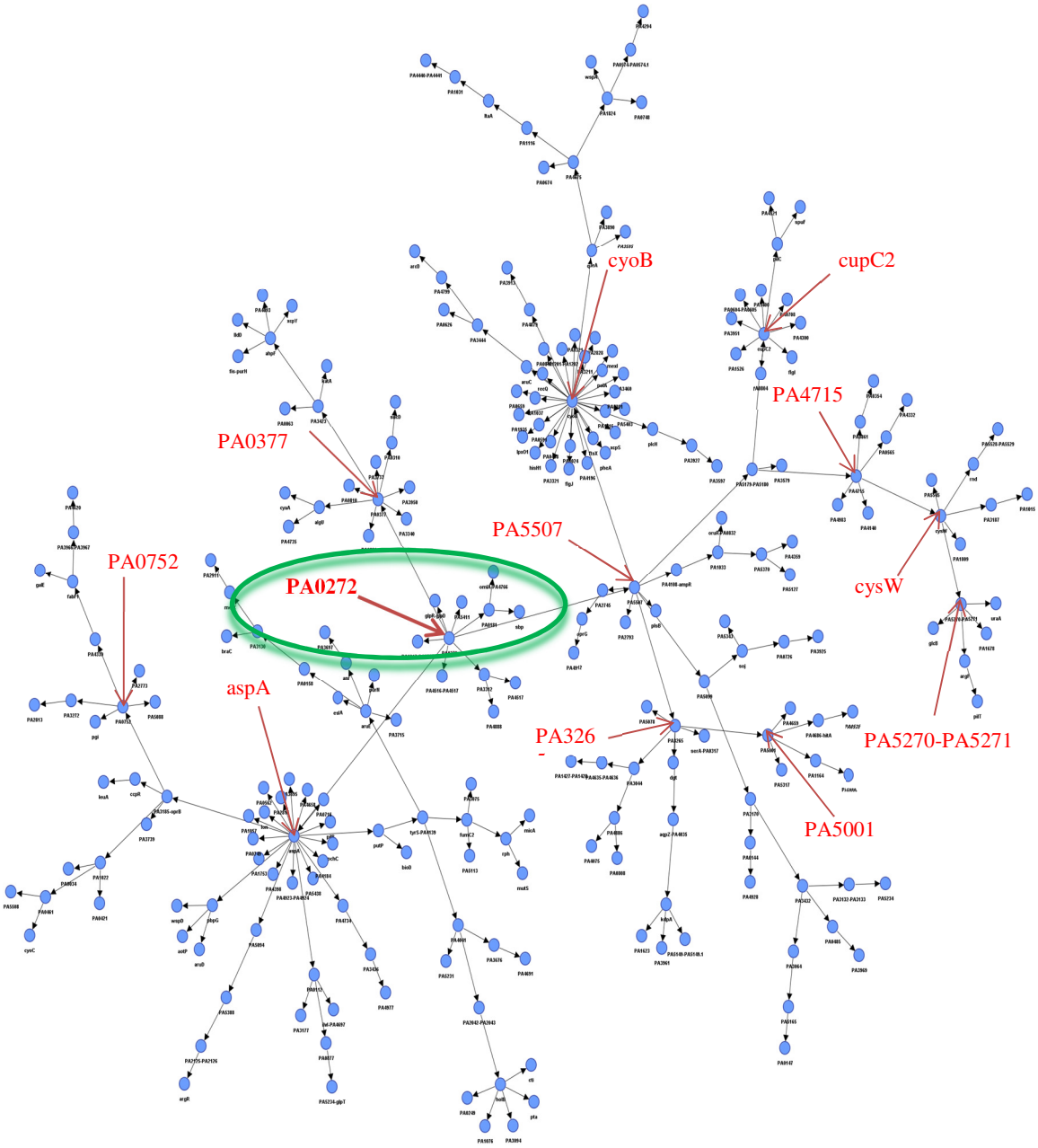



Figure 4.3: *P. aeruginosa* BN learned model—node is gene and link is dependency.

Contingency Table Fit (CTF) is the measure of the degree of fit between network's joint probability distribution and data. The better the network represents the data,

the closer the CTF is to 100 %. Deviance is calculated from the difference between the mean log-likelihoods of network and of the data. The smaller this value, the closer the network represents the database used.

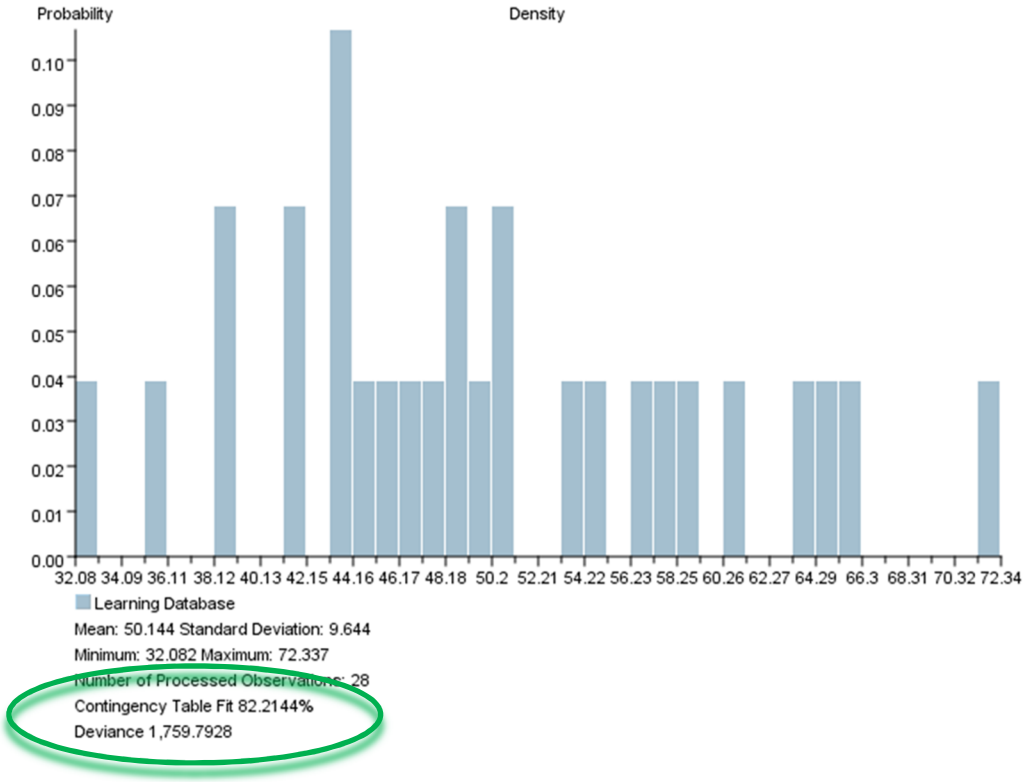



Figure 4.4: Evaluation of the goodness of the survival mechanism with Contingency Fit criterion network

Figure 4.4 shows the evaluation of the goodness of the learned BN model (Figure 4.3). CTF = 82.21% and Deviance = 1,7590.7928 for the learned BN, both of which are in sharp contrast with the respective corresponding values of 32.74% and 21,524.8795 (Figure 4.2) for the global network that comprises all fragmented genes network (Figure 4.1). This shows that the optimal graph with the 249 genes is a good representation of the dataset as opposed to the global fragmented network.

### 4.2 Low Nutrient Genes Identification: Results and Discussions

The methodology established in section 3.4 concerning the root node(s) and the node force was applied. Seeking for root node(s) in the network of Figure 4.3 identified one single node as root (***PA0272***). Below in decreasing order (Table 4.1) is the first dozen most influential nodes of the network according to their nodes force score. These genes are computationally inferred as key orchestrators of *P. aeruginosa* viability in low nutrient water. Table A.1 shows the next 4 dozen of genes.

Table 4.1: Top 12 genes inferred as *P. aeruginosa* contributors to survival

| | Gene | Node Force | Gene Type | Description/Function |
|---|---|---|---|---|
| 1 | *cyoB* | 29.15 | Protein Coding | cytochrome o ubiquinol oxidase subunit I, energy generation |
| 2 | *aspA* | 20.41 | Protein Coding | aspartate ammonia-lyase |
| 3 | PA0272 | 10.72 | Protein Coding | transcriptional regulator, gene regulation |
| 4 | PA5507 | 10.11 | Protein Coding | hypothetical protein |
| 5 | *cupC*2 | 8.61 | Protein Coding | chaperone CupC2, pilus assembly and biofilm formation |
| 6 | PA0377 | 7.79 | Protein Coding | hypothetical protein |
| 7 | PA3265 | 6.31 | Protein Coding | Small molecule transporter |
| 8 | *cysW* | 6.12 | Protein Coding | sulfate transporter |
| 9 | PA4715 | 5.59 | Protein Coding | aminotransferase |
| 10 | PA5270-PA5271 | 5.14 | Protein Coding | hypothetical protein |
| 11 | PA0752 | 4.77 | Protein Coding | hypothetical protein |
| 12 | PA5001 | 4.52 | Protein Coding | cell surface-sugar biosynthetic glycosyl-transferase, LPS synthesis |

The genes are ordered from the top most influential to the least. All of these genes are protein coding, similar to the case of *S. cerevisiae* in [18]. Four of these genes, i.e., 33%, are hypothetical proteins which are characteristic of accessory genes. This particular gene, *PA0272*, is actually the only one found to be a root node, i.e., node at the top of the network's hierarchy. A look up of its functional description—in the genomics expert knowledge database—showed that it is a transcriptional

regulator which is an orchestrator of gene activity, i.e., bacterium's life. It is an originator and directly turns on many other genes (Figure 4.3). Though *cyoB* is not a root node, it has the highest force (29.15 bits) and is reported to be an oxygen active site. Also, *cupC2* appears to be involved, to a certain extent, in the formation of biofilms—results of complex clusters of microorganisms surrounded by a protective and adhesive matrix—that provide resistance to antibiotics.

### 4.3 Experimental Validation of our Results

The results were subjected to validation in the Microbiology lab at the University of Calgary under the supervision of Dr. Shauna Reckseidler-Zenteno and Dr. Shawn Lewenza, Professors at Athabasca University, and also at the University of Calgary. This experimental verification involved a fastidious procedure that required significant amount of time, and only few genes could be tested at a time. *PA0272*, a transcriptional regulator, is an originator—root node in the learned network model—according to the computational analysis. It turned on *algU* which interestingly encodes for a transcriptional regulator of the polysaccharide alginate and other polysaccharide genes involved in survival. *PA0272* was tested for survival in water according to the procedure below:

- The strains (mutant PA0272, wild type PAO1) were inoculated into Luria Broth (LB) and grown overnight at 37°C with shaking at 250 rpm for ~ 18 hrs.
- The next day, 100 µl of the overnight culture was inoculated (sub-cultured) into 3ml of fresh LB and incubated for 3 hours at 37°C and 250 rpm to an optical density ($OD_{600}$) of 0.5 to obtain cells in the mid-log phase of growth.

- One ml of cells from the mid-log culture of each strain was obtained and centrifuged at 13,000 rpm for 3 min. The supernatant was removed, and the cell pellet was re-suspended in 1 ml sterile distilled water (sdH2O). the cells were then centrifuged again; the supernatant was removed, and the cells were re-suspended again in 1 ml sdH2O. This wash step was repeated 1 more time (3 times in total) to remove any nutrients remaining from the LB.
- One ml of the washed cells (concentration of 5 x 108 cfu/ml) were inoculated into 9 ml of sdH2O for a final concentration of approximately 5 x $10^7$ cfu/ml. The water samples were incubated at room temperature and bacterial quantitation was performed by serial dilutions and plating on LB agar. The bacterial quantitation was performed by taking 100 µl of the sample and preparing 10-fold serial dilutions in a series of tubes followed by plating of dilutions $10^{-4}$, $10^{-5}$, and $10^{-6}$ on LB agar plates.

The quantitation was performed at Time 0 to verify the number of bacteria initially added to the water. Thereafter it was then performed every week for a number of weeks to determine the survival of each strain in water, a low nutrient environment. The samples were prepared in triplicate and in some cases where more volume of sample was required, a higher volume of cells and water was used, maintaining the concentration at approximately $10^7$ cfu/ml. Since 0.5 is equal to 5 x 108 colony forming units per ml (cfu/ml) and cells are put in water for survival experiments at a concentration of 1 x $10^7$ cfu/ml cells were diluted 1:10 in the final sample. Figure 4.5 below shows the experiment results of water survival for PA01 vs *PA0272* mutants. According to this, *PA0272* survival declined by 10-fold compared

to wild type (PA01) after ~1 month of incubation in water. Overall it was 10 to 100-fold reduced, 10-fold or greater being significant by student's T-test[7]. The reduction here is in reference to survival essays and not gene expression essays. A reduction in survival is of interest because it is an indication of a mutation in the gene, i.e., the gene is non-functional and therefore is needed for survival. In other words, if the gene was not mutated and was functional then the organism would survive properly. This is a common assumption in molecular biology. A reduction or change that occurs when a gene is mutated means that the gene contributes to the phenotype. In this research results, *PA0272* was predicted to be correlated to the expression of other genes when PAO1 is in water. Based on this, *PA0272* is assumed to have an important role in water survival. Thus, having the strain die in water because it is carrying a mutation in that gene means that *PA0272* must be required for survival.

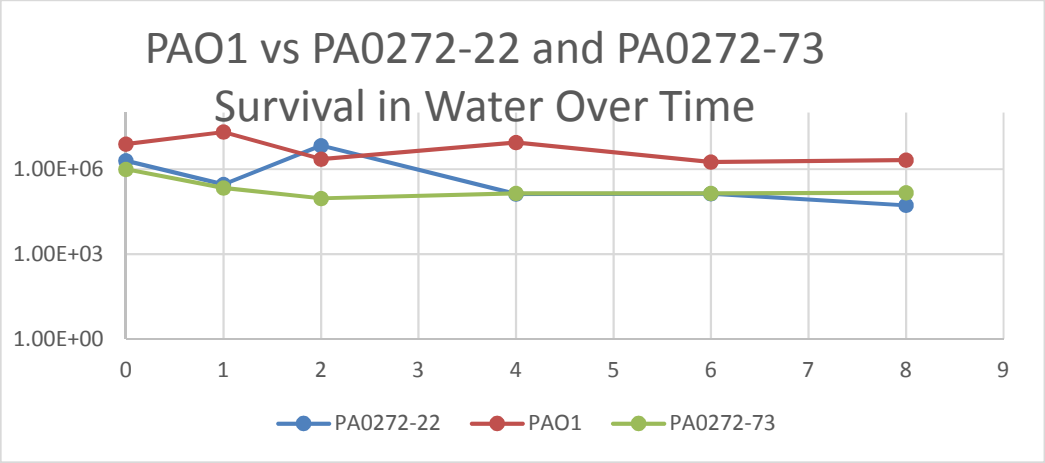



Figure 4.5: Graph of survival test in water over time of PA0272 vs wild type PA01

[7] Inferential statistics used to gauge the significance of difference between the means of 2 groups.

# Chapter V – Conclusions and Further Research

## 5.1 Conclusions

The thesis research concerned *Pseudomonas aeruginosa*, an environmental bacterium with long-term survival, which resists antibiotics and is a common cause of severe nosocomial infections. The pervasiveness of this agent and the high-level death risk that it represents instigated the overarching objective of this research—investigate the mechanism undergirding *P. aeruginosa* survival in a water medium depleted of nutrient.

A 28 timepoints gene expression matrix dataset, resulting from *P. aeruginosa* genes response to low nutrient water was used. The collection of the data was based on PAO1 mini-Tn5-*luxCDABE* transposon mutant library.

This research started with the study of the data and Microbiology literature review, particularly on *P. aeruginosa* and its genome. *P aeruginosa* has a great genetic and a rich functional diversity, which both justify its versatility. Some comparative genomic studies in the literature revealed that *P aeruginosa* genome is of mosaic composition, consisting of Core genome and Accessory genome. While the former is common to almost all strains, the latter, which is about 10% of the entire genome, varies between strains and is designated as the niche-based adaptation of the organism. Interests in the literature have been expressed to further the knowledge of this accessory genome. The hypothesis of this research is that *P aeruginosa* is capable of long-term survival in water due to the presence of particular genes which encode for protein that facilitate persistence. This research explored a computational model that would well characterize the survival mechanism of this organism, and identify potential genes involved in the survival mechanism.

The computational approach of this research is based on Machine Learning and

began with the focus on the exploration of gene expression analysis literature which asserted that the phenomenon of gene expression is intrinsically stochastic. This led to the selection of Bayesian Networks for its probabilistic nature, to analyze this probabilistic phenomenon. The literature study continued further with Bayesian Networks learning and construction, as well as heuristic search techniques. Upon these rests the research methodology.

With this methodology, an optimal *P. aeruginosa* survival mechanism model has been established through a probabilistic graph model that was learned and constructed solely from the gene expression data. The graphical representation permits an easy visual reading of the network of interaction among the genes that regulate the gene expression of *P. aeruginosa*. The model revealed that *PA0272*, a transcriptional regulator, is a root node and therefore an originator. This gene has a directed arc to a*lgU* which interestingly is also a transcriptional regulator of the polysaccharide alginate and other polysaccharide genes involved in survival.

From this model, node influence techniques were applied to infer a dozen distinct genes, as the principal orchestrators of *P. aeruginosa* viability maintenance in low nutrient water. All the genes inferred are found to be of protein coding type. Also, their associated functional descriptions were identified which biologically aligned with the bacterium's survival. The computational findings are supported by the theory, and the results have been experimentally lab-tested for validation. It is found that the survival of *PA0272* in water was 10 to 100-fold reduced compared to PA01, the wild type; 10-fold or greater being significant.

### 5.2 Additional Works Recommendations

The functional interplay involved in *P. aeruginosa* survival mechanism was also explored to determine the influence of certain combinations of gene states on specific states of the bacterium. Regarding the functional interplay, by *Multiple Clustering* technique, 33 clusters (Figure A.1) were made on one hand and 107 clusters (Figure A.2 and Table A.2) were made on the other, which suggest that these could be the representations of the functional modules involved in the survival mechanism. In fact, [6] found that 45.8% of *P. aeruginosa* ORFs[8] are genes with unknown functions and were able to assign 54.2% to 25 functional categories derived from those used for *E. coli* functional classification. On this basis, one can postulate that when all the functions are identified, the number of functional categories in *P. aeruginosa* would be between 33 and 107. This research recommends further comparative studies between the dendrograms of the 33-cluster and the 107-cluster to check and verify if each cluster indeed carries out a function. Also, one can computationally determine the optimal number of functional categories within the range 33-107.

As far as the bacterium states exploration is concerned, by *Data clustering*, a factor *Factor_0* (Figure A.3 and Table A.3) has been obtained, to represent the bacterium as an entity. The states of this factor are denoted C1 and C2 as in Table A.3. In this case as well, this research recommends further studies to investigate if *Factor_0* can be a computational representation of the bacterium and see if a relation can be inferred between its states and the experimental states of the *P. aeruginosa*.

[8] Open Reading Frame: It is the portion of a reading frame that can be translated.

### 5.3 Future works with Inductive Logic Programming

According to [38], *Decision-tree-learning* and *Inductive Logic Programming* (ILP) are both predictive models based on learning techniques and are very useful in AI (Artificial Intelligence) especially in Machine Learning. While attribute-based, the first, though still among the most powerful, is the simplest. The second is an integration of inductive methods with great potentials of first-order logic representation. It focuses on representing hypotheses as logic programs. One of the key differences between these two models is the background knowledge. Albeit in certain cases of decision-tree-learning, prior knowledge could be incorporated, in ILP which presents a rigorous approach to the general knowledge-based inductive learning, prior knowledge is integral part of its execution, namely its hypothesis determination through the entailment equation ($Background \wedge Hypothesis \wedge Descriptions \models Classifications$). Because ILP is relational and based on first-order theories, it accomplishes successful learning in domains where attributes-based, i.e., Decision-tree-learning, hardly succeeds. An example of such domain is protein folding which intrinsically rests on relationships between objects. Another example is that decision-tree-learning will flounder in the resolution of a problem involving binary predicate that would require a transformation into unary predicate (attribute-based); this is characteristic of decision-tree-learning which is unable to learn relational predicates. ILP being based on such types of predicates, it has a wider range of problem solving applications. The decision tree is a tree-based learning representation which starts and gradually grows until it fits the data observations. Similarly, ILP begins with general rules and progressively tightens up to suitably be consistent

with the data. Overall, ILP outperforms decision-tree (and others), and this comes from its ability represent relations and also to use prior/background knowledge. Its human friendly readable rules made it to be adopted in various disciplines notably bioinformatics which is in fact we are dealing with here. Now, beyond the identification of genes responsible for the persistence of *Pseudomonas aeruginosa*, it will be of interest to know what other knowledge can be gained from the data. And we suggest the power of induction logic programming (ILP). Its rules are easily interpreted by human and this makes it popular and well accepted in domains beyond computer science. Work by [55] is a good example of the power of ILP which they used to include experiment design and therefore devising an autonomous scientist which discovered new knowledge about functional genomic of yeast. So ILP consideration to investigate unknown environmental factors (stimuli, temperature, aeration, PH, biological etc.…) shaping *P. aeruginosa* survival mechanisms is suggested for future work.

### 5.4 Future works with Dynamic Bayesian Networks

In this thesis, static Bayesian Networks, which describe a probability distribution over fixed variables, has been implemented. Dynamic Bayesian Networks (DBN) are an extension of static BN to a temporal dynamic process. The next step in furthering the understanding of *P. aeruginosa* survival is to establish a temporal genes interactions regulatory network that will give deeper insight on how the ineluctable time factor affects the mechanism. This will require a prior network model $B_o$ that specifies the distribution over gene expression at time $T_0$ and a transition network $B_T$ that would specify the transition model.

## REFERENCES

[1] F. Jorgensen, M. Bally, V. Chapon-Herve, G. Michel, A. Lazdunski, P. Williams, and G. S. A. B. Stewart, "RpoS-dependent stress tolerance in *Pseudomonas aeruginosa*," Microbiology, vol. 145, no. 4, pp. 835–844, Jan. 1999.

[2] S. Lewenza, R. K. Falsafi, G. Winsor, W. J. Gooderham, J. B. McPhee, F. S. Brinkman, R. E. Hancock, "Construction of a mini-Tn5-*luxCDABE* mutant library in *Pseudomonas aeruginosa* PAO1: a tool for identifying differentially regulated genes", Genome research, vol. 15, no. 4 pp. 583-589, 2005.

[3] J. A. Driscoll, S. L. Brody, and M. H. Kollef, "The Epidemiology, Pathogenesis and Treatment of *Pseudomonas aeruginosa* Infections," Drugs, vol. 67, no. 3, pp. 351–368, 2007.

[4] V. L. Kung, E. A. Ozer, and A. R. Hauser, "The Accessory Genome of *Pseudomonas aeruginosa*," Microbiology and Molecular Biology Reviews, vol. 74, no. 4, pp. 621–641, 2010.

[5] A. Kramer, I. Schwebke, and G. Kampf, "How long do nosocomial pathogens persist on inanimate surfaces? A systematic review," BMC Infectious Diseases, vol. 6, no. 1, 2006.

[6] C. K. Stover, et al., "Complete genome sequence of *Pseudomonas aeruginosa* PAO1, an opportunistic pathogen", Nature, vol. 406, no 6799, pp. 959-964, 2000.

[7] M. C. Wolfgang, B. R. Kulasekara, X. Liang, D. Boyd, K. Wu, Q. Yang, C. G. Miyada, and S. Lory, "Conservation of genome content and virulence determinants among clinical and environmental isolates of *Pseudomonas aeruginosa*," Proceedings of the National Academy of Sciences, vol. 100, no. 14, pp. 8484–8489, 2003.

[8] M. K. Winson, S. Swift, P. J. Hill, C. M. Sims, G. Griesmayr, B. W. Bycroft, and G. S. Stewart, (1998). Engineering the *luxCDABE* genes from *Photorhabdus luminescens* to provide a bioluminescent reproter for constitutive and promoter probe pasmids and mini-*Tn5* constructs. FEMS microbiology letters, *16(2)*, 193-202.

[9] S. Lewenza, J. M. Abboud, K. Poon, M. Kobryn, I. Humplik, J. R. Bell, L. Mardan, S. L. Reckseidler-Zenteno. *Pseudomonas aeruginosa* displays a dormancy phenotype during long-term survival in water. PloS one. 2018 Sep 20; 13(9); https://doi.org/10.1371/journal.pone.0198384.

[10] C. S. Mesquita, P. Soares-Castro, and P. M. Santos, "*Pseudomonas aeruginosa*: phenotypic flexibility and antimicrobial resistance," Microbial pathogens and strategies for combating them: science, technology and education, vol. 1. A. Méndez-Vilas, Formatex Research Center, pp. 650–665, 2013.

[11] E. Aguilar-Barajas, M. I. Ramírez-Díaz, H. Riveros-Rosas, and C. Cervantes, "Heavy Metal Resistance in Pseudomonads," Pseudomonas, pp. 255–282, 2010.

[12] J. Campos-García, "Metabolism of Acyclic Terpenes by Pseudomonas," Pseudomonas, pp. 235–253, 2010.

[13] M. M. Babu, "An Introduction to microarray data analysis," Computational genomics: Theory and application, R P. Grant, Horizon Bioscience, pp. 225-249, 2004.

[14] J. Han, M. Kamber, and J. Pei, (2011). Cluster Analysis: Basic Concepts. *Data mining: concepts and techniques* (pp. 443-495). Elsevier.

[15] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman, "Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data," Nature Genetics, vol. 34, no. 2, pp. 166–176, November 2003.

[16] M. Molla, M. Waddell, D. Page, and J. Shavlik, "Using Machine Learning to Design and Interpret Gene-Expression Microarrays," AI Magazine, 22-Mar-2004.

[17] J. H. Do and D. K. Choi, "Clustering approaches to identifying gene expression patterns from DNA microarray data" Molecules and cells, vol. 25, no. 2, pp. 279-288, 2008.

[18] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian Networks to Analyze Expression Data," Journal of Computational Biology, vol. 7, no. 3-4, pp. 601–620, 2000.

[19] Bayes Net, September, 2007. Retrieved October 26, 2015, from http://www.bayesnets.com/

[20] G. Cooper, (1999). An overview of the representation and discovery of causal relationships using Bayesian networks. Computation, causation, and discovery, 4-62.

[21] M. Druzdzel and L. V. D. Gaag, "Building probabilistic networks: Where do the numbers come from?", IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 4, pp. 481–486, 2000.

[22] D. Heckerman, "Probabilistic similarity networks," Networks, vol. 20, no. 5, pp. 607–636, 1990.

[23] S. Nadkarni and P. P. Shenoy, "A causal mapping approach to constructing Bayesian networks," Decision Support Systems, vol. 38, no. 2, pp. 259–281, 2004.

[24] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," Machine Learning, vol. 9, no. 4, pp. 309–347, 1992.

[25] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," Machine Learning, vol. 20, no. 3, pp. 197–243, 1995.

[26] D. M. Chickering, "Learning Bayesian Networks is NP-Complete," Learning from Data Lecture Notes in Statistics, pp. 121–130, 1996.

[27] C. Su, A. Andrew, M. Karagas, and M. E. Borsuk, "Overview of Bayesian network approaches to model gene-environment interactions and cancer

susceptibility" Doctoral dissertation, International Environmental Modelling and Software Society (iEMSs), 2012.

[28] R. E. Neapolitan, "More structure learning", Learning Bayesian networks. Upper Saddle River, NJ: Pearson Prentice Hall, pp. 617-645, 2004.

[29] D. M. Chickering, "Optimal structure identification with greedy search", The Journal of Machine Learning Research, vol. 3, pp. 507-554, 2003.

[30] J. Pearl, and T. S. Verma. "A Theory of Inferred Causation." In Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference on..., KR'91, Cambridge, MA, April 22-25, 1991, vol. 2, p. 441. Morgan Kaufmann Pub, 1991.

[31] P. Spirtes, C. N. Glymour, and R. Scheines, Causation, prediction, and search. Cambridge, MA: MIT Press, 2000.

[32] R. E. Neapolitan, Learning Bayesian networks. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[33] R. W. Robinson, "Counting unlabeled acyclic digraphs," Lecture Notes in Mathematics Combinatorial Mathematics V, pp. 28–43, 1977.

[34] Gillispie, S. B., & Perlman, M. D. (2001, August). Enumerating Markov equivalence classes of acyclic digraphdels. In Proceedings of the Seventeeth conerence on Uncertainty in artificial intelligence (pp. 171-177). Morgan Kaufmann Publishers Inc.

[35] S. J. Russell and P. Norvig, "Beyond Classical Search", Artificial intelligence: a modern approach. Upper Saddle River: Prentice-Hall, pp. 120-160, 2010.

[36] A. M. Carvalho, Scoring functions for learning Bayesian networks. Inesc-id Tec. Rep., 2009.

[37] M. Singh and M. Valtorta, "An Algorithm for the Construction of Bayesian Network Structures from Data," Uncertainty in Artificial Intelligence, pp. 259–265, 1993.

[38] S. J. Russell and P. Norvig, Artificial intelligence: a modern approach. Upper Saddle River: Prentice-Hall, 2010.

[39] C. S. Mesquita, P. Soares-Castro, and P. M. Santos, "Pseudomonas aeruginosa: phenotypic flexibility and antimicrobial resistance," Microbial pathogens and strategies for combating them: science, technology and education, vol. 1. A. Méndez-Vilas, Formatex Research Center, pp. 650–665, 2013.

[40] M. V. Grosso-Becerra, C. Santos-Medellín, A. González-Valdez, J. L. Méndez, G. Delgado, R. Morales-Espinosa, and G. Soberón-Chávez, (2014). Pseudomonas aeruginosa clinical and environmental isolates constitute a single population with high phenotypic diversity. *BMC genomics*, 15(1), 318.

[41] K. Murphy and S. Mian, "Modelling gene expression data using dynamic Bayesian networks", Technical report, Computer Science Division, University of California, Berkeley, CA, vol. 104, 1999.

[42] A. Tversky and D. Kahneman, "Causal schemata in judgements under uncertainity." In D. Kahneman, P. Slovic, and A. Tversky (Eds), Judgement Under Uncertainity: Heuristics and Biases. Cambridge University Press, 1982.

[43] S. Ott, S. Imoto, and S. Miyano, "Finding Optimal Models For Small Gene Networks," In Pacific symposium on Biocomputing, vol. 9, pp. 557-567, 2003.

[44] S. L. Lauritzen, B. Thiesson, and D. J. Spiegelhalter, "Diagnostic systems by model selection: a case study," Selecting Models from Data Lecture Notes in Statistics, pp. 143–152, 1994.

[45] H. Youssef, S. M. Sait, and H. Adiche, "Evolutionary algorithms, simulated annealing and tabu search: a comparative study," Engineering Applications of Artificial Intelligence, vol. 14, no. 2, pp. 167–181, 2001.

[46] M. A. Arostegui, S. N. Kadipasaoglu, and B. M. Khumawala, "An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems," International Journal of Production Economics, vol. 103, no. 2, pp. 742–754, 2006.

[47] W. Lam and F. Bacchus, "Learning Bayesian Belief Networks: An Approach Based On The Mdl Principle," Computational Intelligence, vol. 10, no. 3, pp. 269–293, 1994.

[48] C. H. Liao and L. M. Shollenberger, Survivability and long-term preservation of bacteria in water and in phosphate-buffered saline. Letters in applied microbiology. 2003 Jul;37(1):45-50.

[49] Norsys (2007), Netica API: programmer's library, reference manual. Retrieved from http://norsys.com/netica-j/NeticaJ_415/docs/NeticaAPIMan_C.pdf

[50] Koiter, J. R. (2006). *Visualizing inference in Bayesian networks* (Doctoral dissertation, University of Pittsburgh).

[51] Elvira Consortium. (2002, November). Elvira: An environment for creating and using probabilistic graphical models. *In Proceedings of the first European workshop on probabilistic graphical models* (pp. 222-230).

[52] S. Kullback, and R. A. Leibler, (1951). On information and sufficiency. The annals of mathematical statistics, 22(1), 79-86.

[53] S. Conrady, L. Jouffe, "Bayesian Networks and BayesiaLab: A Practical Introduction for Researchers", Franklin, TN: Bayesia USA, 2015.

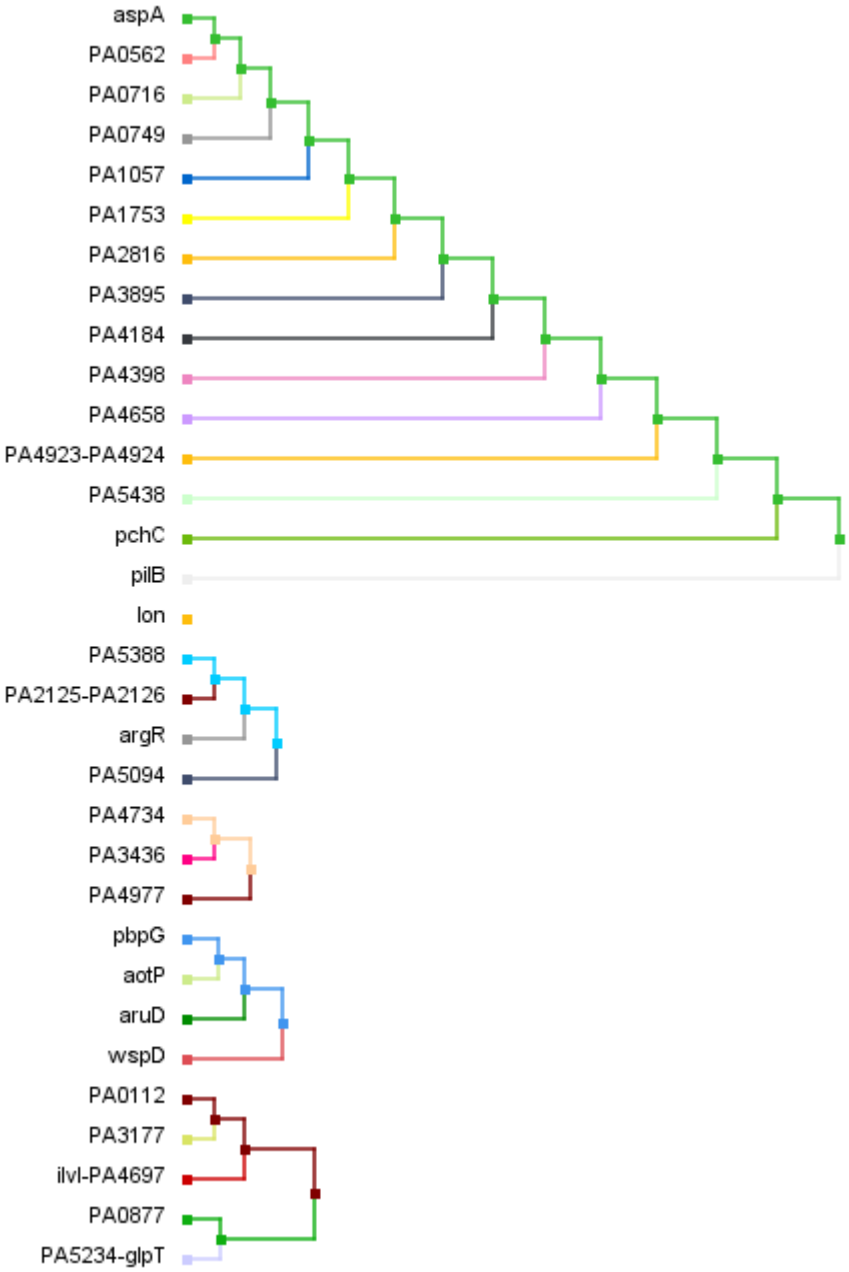
[54] C. Yuan, B. Malone, and X. Wu, (2011, July). Learning optimal Bayesian networks using A* search. *In IJCAI proceedings-international joint conference on artificial intelligence* (Vol. 22, No. 3, p. 2186).

[55] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, and A. Sparkes, (2009). The automation of science. Science, 324(5923), 85-89.

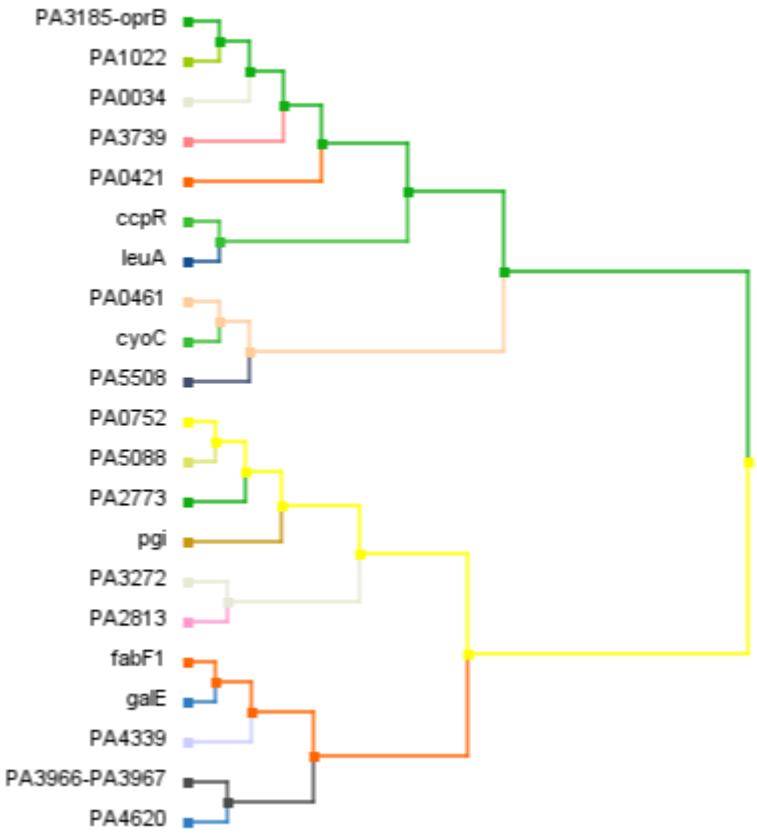
## Appendix – Additional Tables and Figures

Table A.1: List of the next 4 dozen genes with higher node force

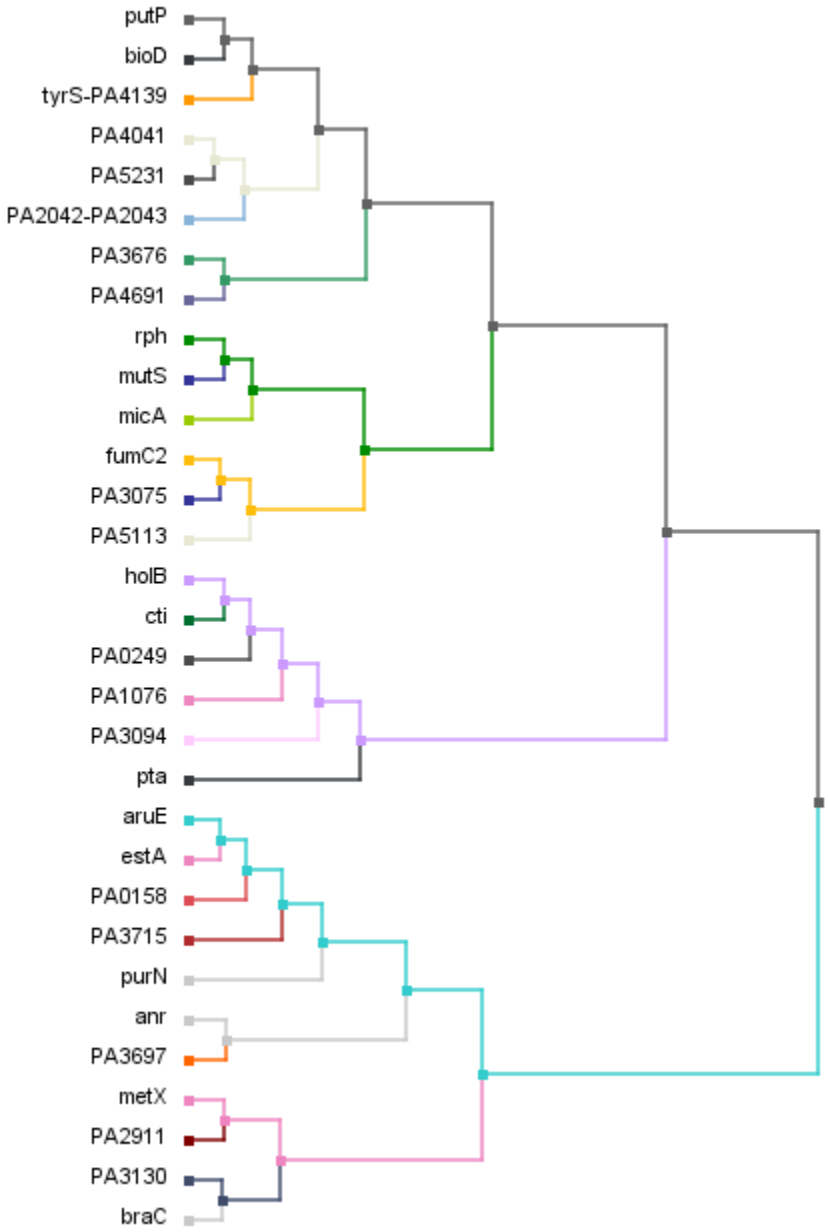
| | Gene | Arc Force |
|---|---|---|
| 13 | PA5179-PA5180 | 4.404549 |
| 14 | *ahpF* | 4.196446 |
| 15 | *aruE* | 4.192155 |
| 16 | *holB* | 4.061651 |
| 17 | PA3185-oprB | 3.614821 |
| 18 | PA4041 | 3.446179 |
| 19 | *queA* | 3.386628 |
| 20 | PA0112 | 3.322634 |
| 21 | *kdpA* | 3.064861 |
| 22 | PA3423 | 3.060869 |
| 23 | PA4675 | 3.047076 |
| 24 | *pbpG* | 2.959627 |
| 25 | PA4886 | 2.955684 |
| 26 | PA5099 | 2.909747 |
| 27 | PA3432 | 2.888731 |
| 28 | PA3044 | 2.872886 |
| 29 | PA3170 | 2.870031 |
| 30 | PA1022 | 2.781305 |
| 31 | tyrS-PA4139 | 2.780358 |
| 32 | *fumC2* | 2.696253 |
| 33 | PA3312 | 2.684043 |
| 34 | PA1824 | 2.669281 |
| 35 | PA0181 | 2.608916 |
| 36 | *algU* | 2.581298 |
| 37 | *fabF1* | 2.432385 |
| 38 | *pilC* | 2.413293 |
| 39 | PA3444 | 2.267929 |
| 40 | PA3130 | 2.248786 |
| 41 | PA1033 | 2.225171 |
| 42 | *soj* | 2.14114 |
| 43 | PA0461 | 2.140365 |
| 44 | PA4635-PA4636 | 2.123005 |
| 45 | PA5370 | 2.081975 |
| 46 | aqpZ-PA4035 | 1.996317 |
| 47 | PA4734 | 1.951847 |
| 48 | PA3187 | 1.951017 |
| 49 | *putP* | 1.928699 |
| 50 | PA4108-ampR | 1.909877 |
| 51 | *rph* | 1.904199 |
| 52 | *argF* | 1.846952 |
| 53 | PA0310 | 1.810984 |
| 53 | PA2125-PA2126 | 1.807017 |
| 55 | PA5388 | 1.803333 |
| 56 | PA11.64 | 1.788807 |
| 57 | PA0565 | 1.768052 |
| 58 | PA0144 | 1.766551 |
| 59 | PA2745 | 1.716449 |


aspA
PA0562
PA0716
PA0749
PA1057
PA1753
PA2816
PA3895
PA4184
PA4398
PA4658
PA4923-PA4924
PA5438
pchC
pilB
lon
PA5388
PA2125-PA2126
argR
PA5094
PA4734
PA3436
PA4977
pbpG
aotP
aruD
wspD
PA0112
PA3177
ilvI-PA4697
PA0877
PA5234-glpT

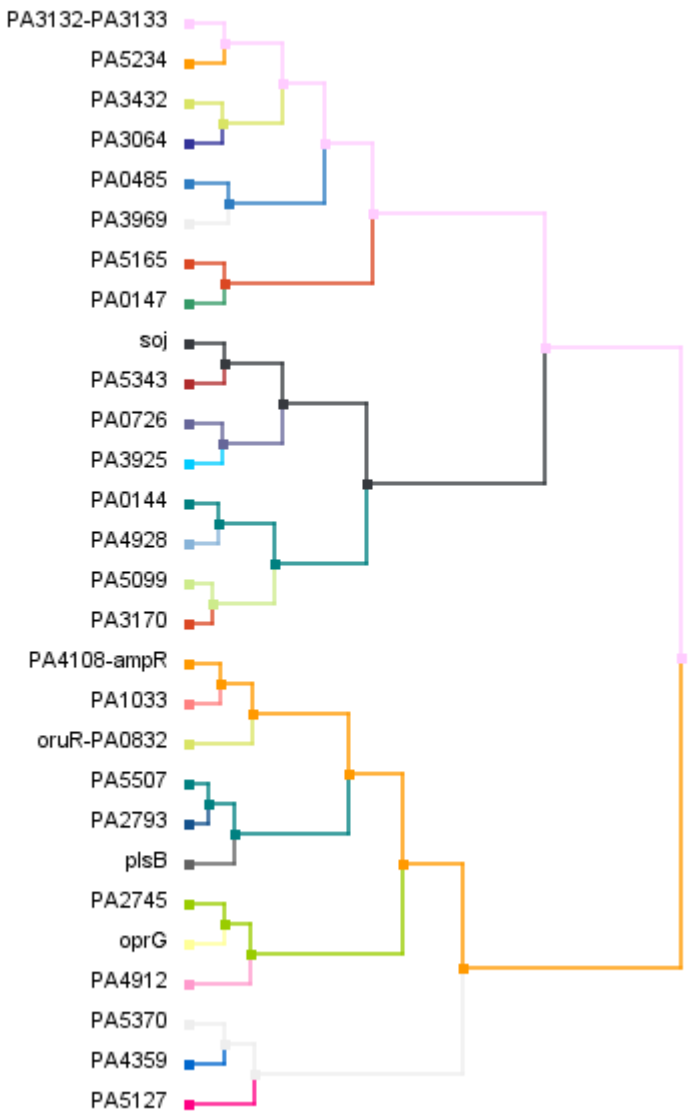
33-cluster (a)


PA3185-oprB
PA1022
PA0034
PA3739
PA0421
ccpR
leuA
PA0461
cyoC
PA5508
PA0752
PA5088
PA2773
pgi
PA3272
PA2813
fabF1
galE
PA4339
PA3966-PA3967
PA4620

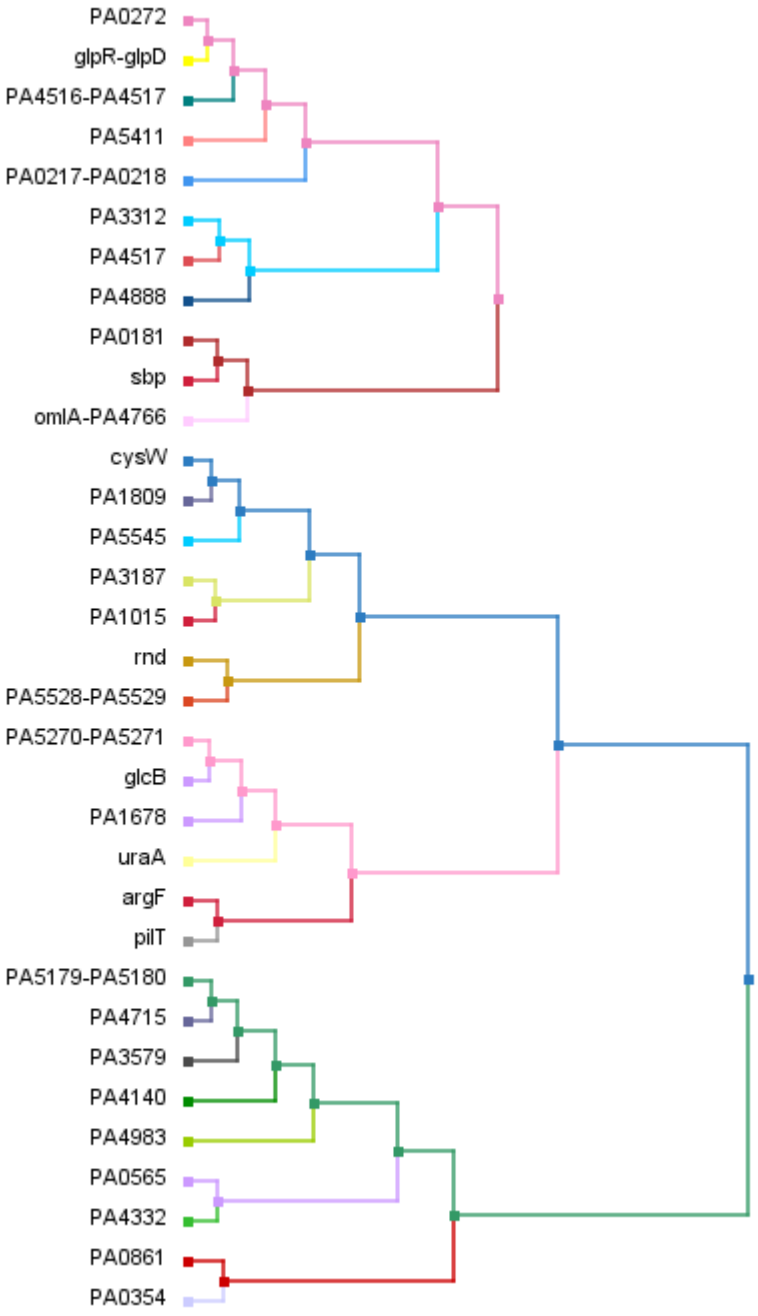
33-cluster (b)


putP
bioD
tyrS-PA4139
PA4041
PA5231
PA2042-PA2043
PA3676
PA4691
rph
mutS
micA
fumC2
PA3075
PA5113
holB
cti
PA0249
PA1076
PA3094
pta
aruE
estA
PA0158
PA3715
purN
anr
PA3697
metX
PA2911
PA3130
braC

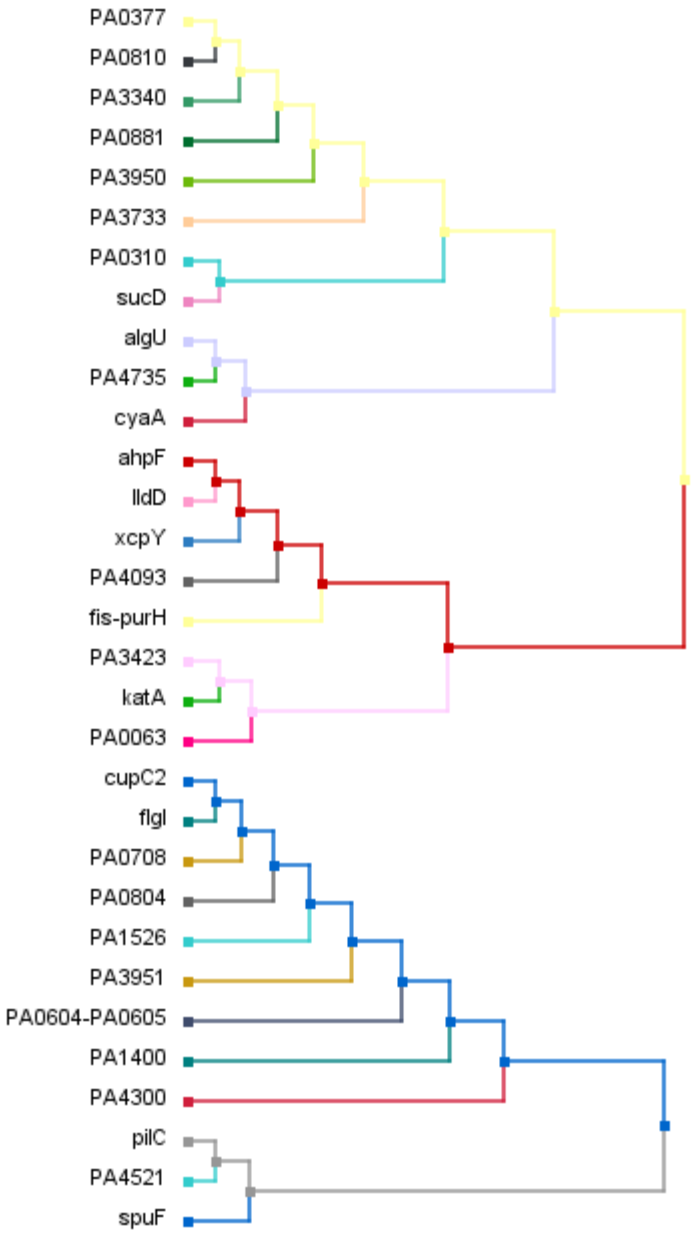
33-cluster (c)


PA3132-PA3133
PA5234
PA3432
PA3064
PA0485
PA3969
PA5165
PA0147
soj
PA5343
PA0726
PA3925
PA0144
PA4928
PA5099
PA3170
PA4108-ampR
PA1033
oruR-PA0832
PA5507
PA2793
plsB
PA2745
oprG
PA4912
PA5370
PA4359
PA5127

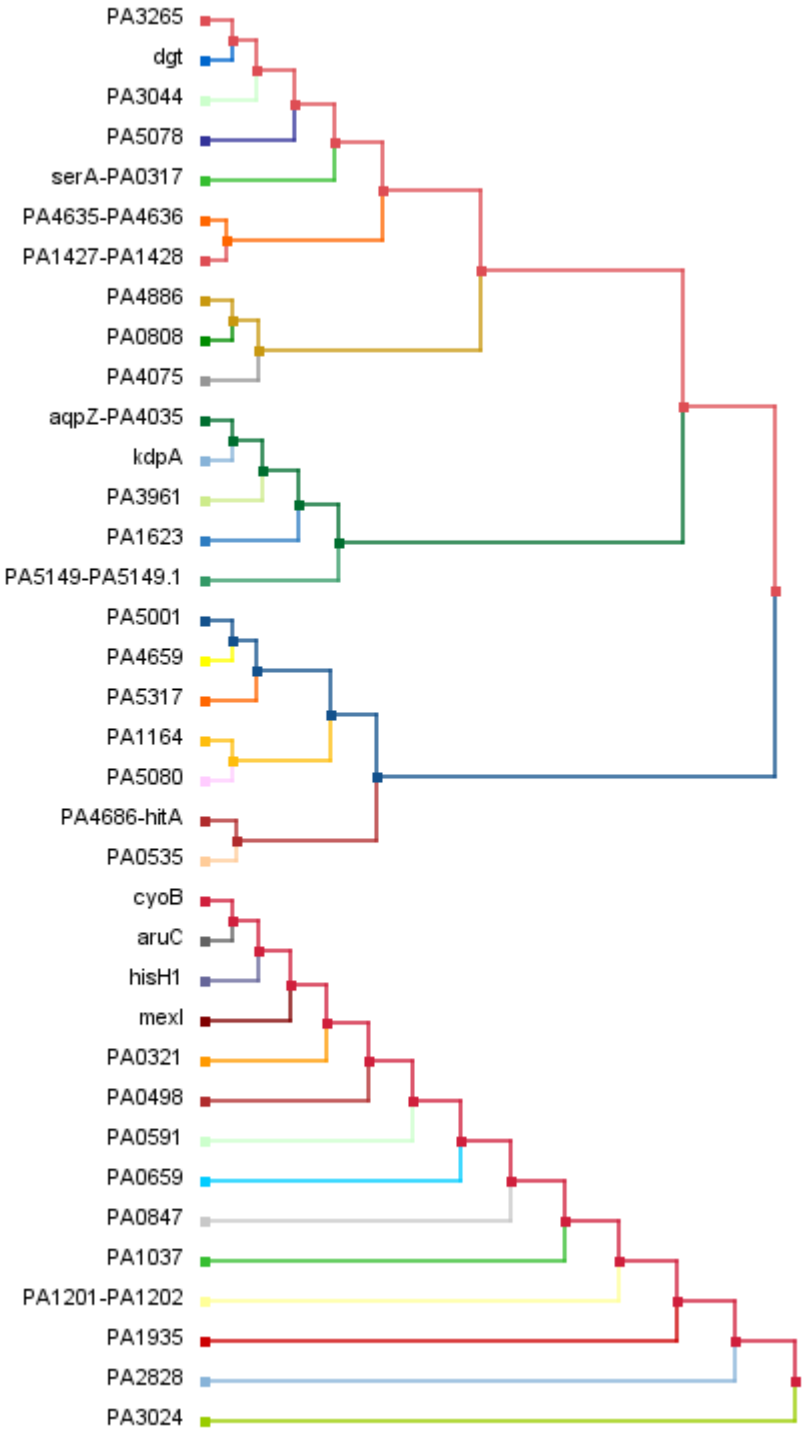
33-cluster (d)


PA0272
glpR-glpD
PA4516-PA4517
PA5411
PA0217-PA0218
PA3312
PA4517
PA4888
PA0181
sbp
omlA-PA4766
cysW
PA1809
PA5545
PA3187
PA1015
rnd
PA5528-PA5529
PA5270-PA5271
glcB
PA1678
uraA
argF
pilT
PA5179-PA5180
PA4715
PA3579
PA4140
PA4983
PA0565
PA4332
PA0861
PA0354

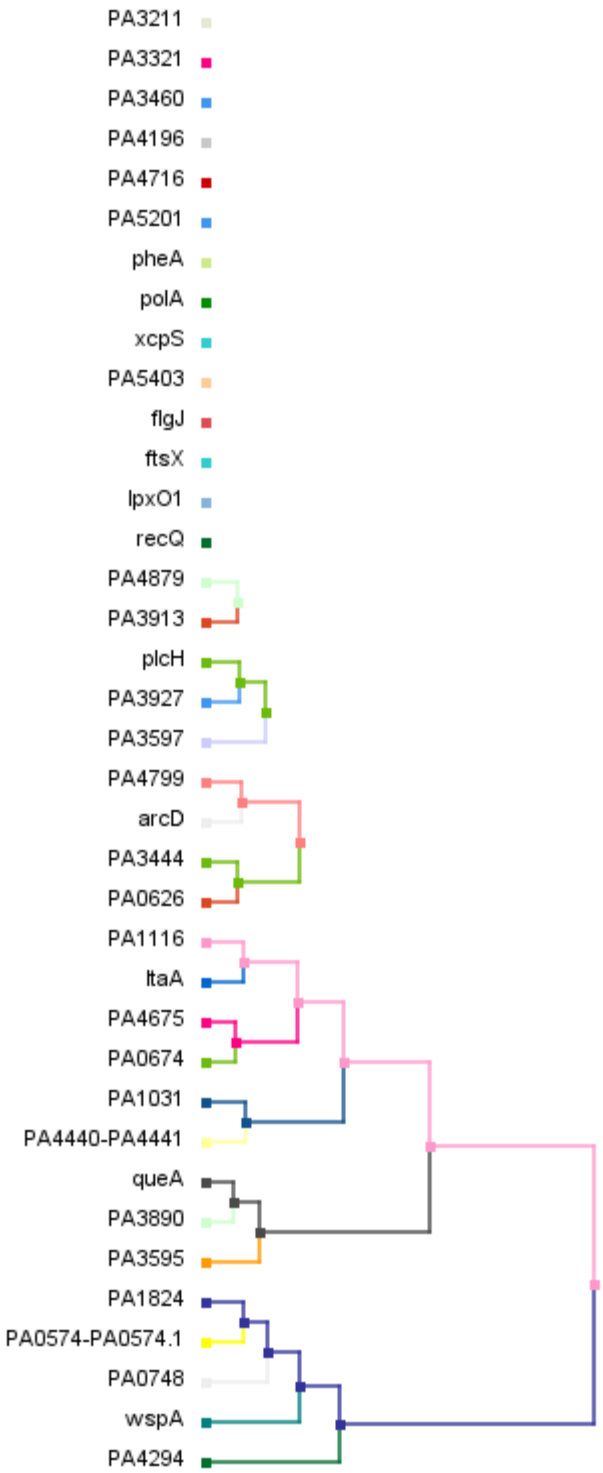
33-cluster (e)


PA0377
PA0810
PA3340
PA0881
PA3950
PA3733
PA0310
sucD
algU
PA4735
cyaA
ahpF
lldD
xcpY
PA4093
fis-purH
PA3423
katA
PA0063
cupC2
flgI
PA0708
PA0804
PA1526
PA3951
PA0604-PA0605
PA1400
PA4300
pilC
PA4521
spuF

33-cluster (f)

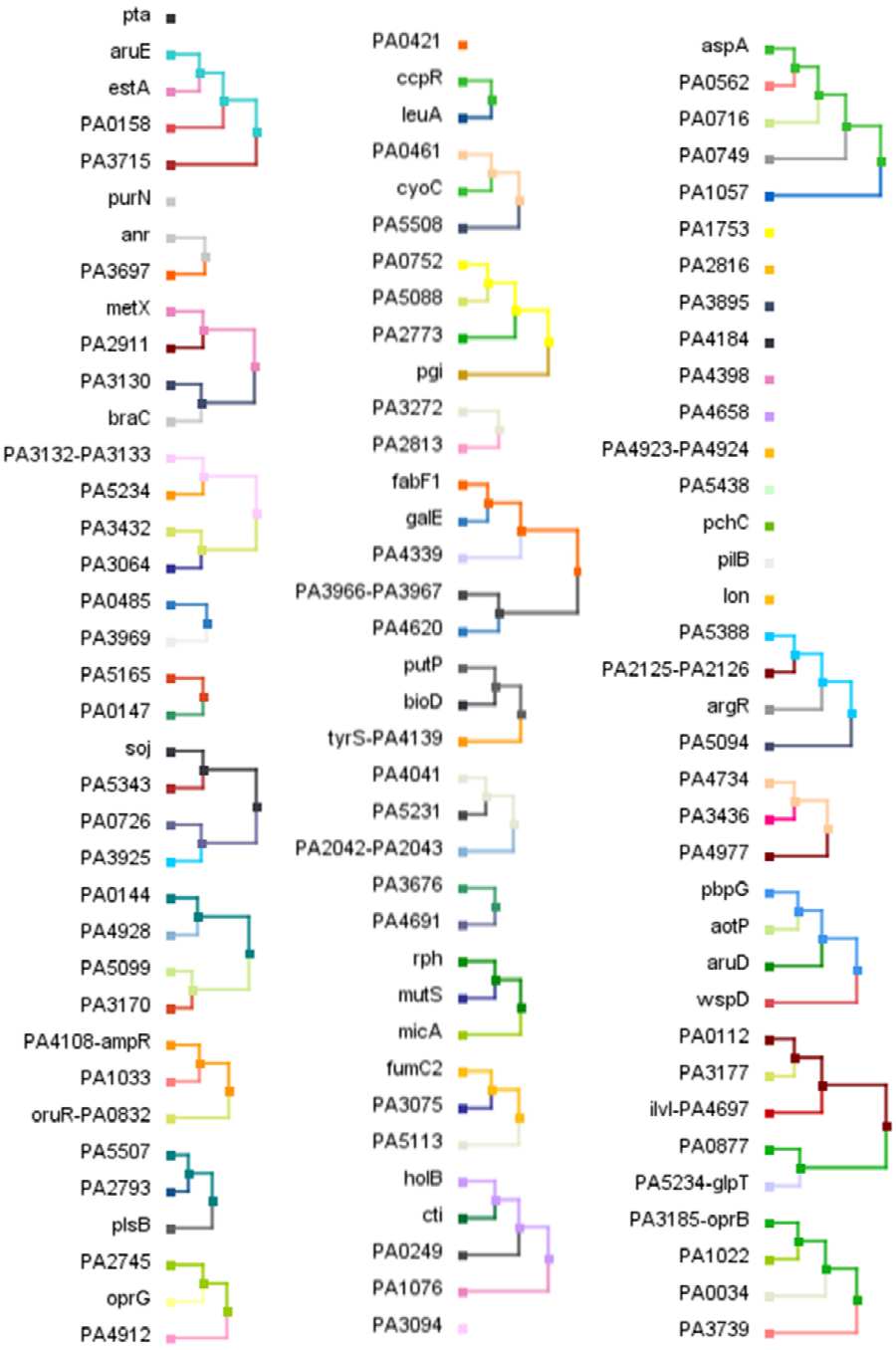

PA3265
dgt
PA3044
PA5078
serA-PA0317
PA4635-PA4636
PA1427-PA1428
PA4886
PA0808
PA4075
aqpZ-PA4035
kdpA
PA3961
PA1623
PA5149-PA5149.1
PA5001
PA4659
PA5317
PA1164
PA5080
PA4686-hitA
PA0535
cyoB
aruC
hisH1
mexI
PA0321
PA0498
PA0591
PA0659
PA0847
PA1037
PA1201-PA1202
PA1935
PA2828
PA3024

33-cluster (g)

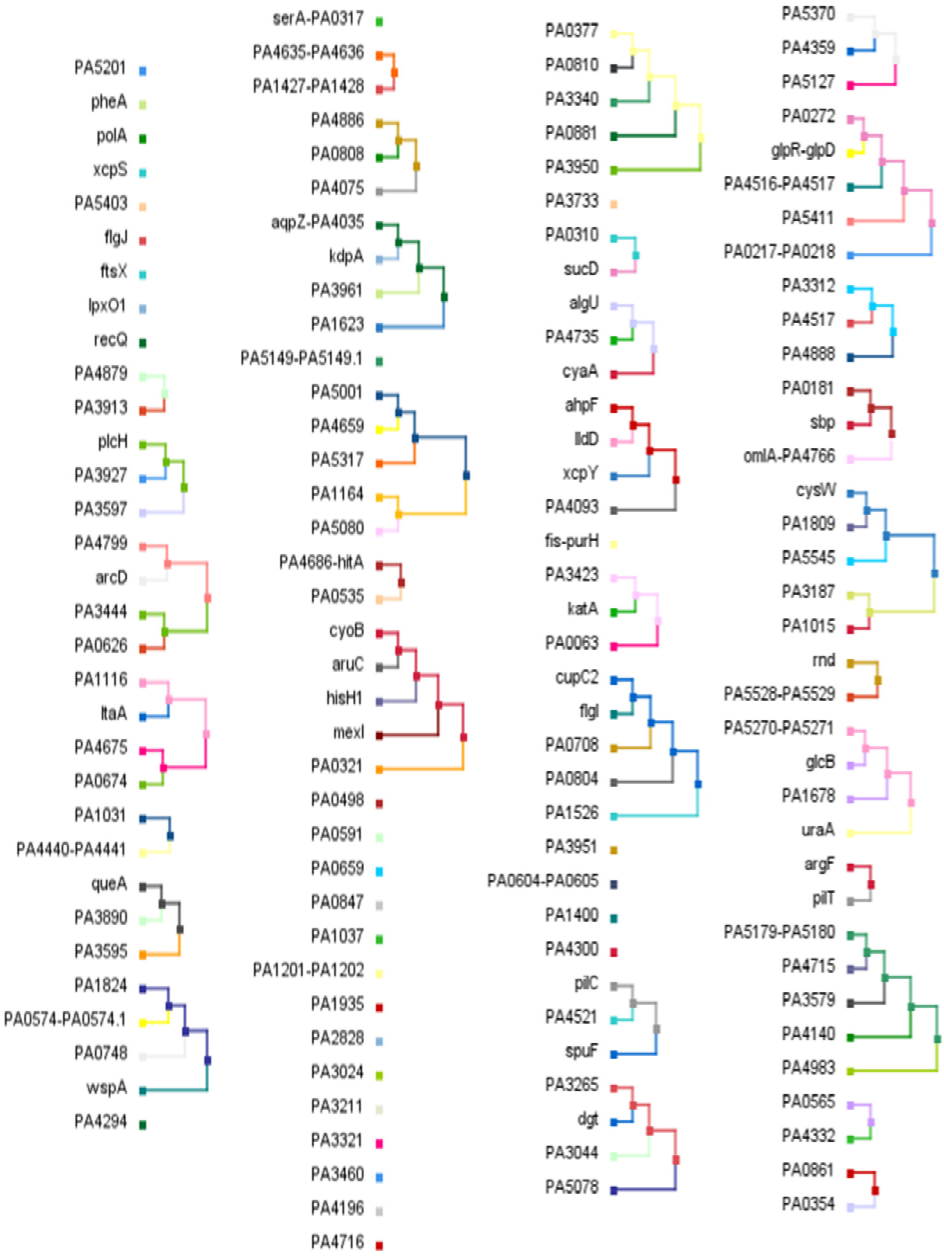



33-cluster (h)

Figure A.1: 33-cluster dendrogram of *P. aeruginosa* possible functional interplay




107-clusters (a)




107-clusters (b)

Figure A.2: 107-cluster dendrogram of *P. aeruginosa* possible functional interplay

Table A.2: 107-cluster suggested functional genes classification list

| Classes | Nodes |
|---|---|
| [Factor_0] | PA0112 |
| | PA0877 |
| | PA3177 |
| | ilvI-PA4697 |
| | PA5234-glpT |
| [Factor_1] | PA5001 |
| | PA1164 |
| | PA4659 |
| | PA5317 |
| | PA5080 |
| [Factor_2] | fabF1 |
| | PA3966-PA3967 |
| | galE |
| | PA4339 |
| | PA4620 |
| [Factor_3] | PA0377 |
| | PA3340 |
| | PA0810 |
| | PA3950 |
| | PA0881 |
| [Factor_4] | PA4715 |
| | PA5179-PA5180 |
| | PA3579 |
| | PA4140 |
| | PA4983 |
| [Factor_5] | cupC2 |
| | PA0804 |
| | PA1526 |
| | flgI |
| | PA0708 |
| [Factor_6] | cysW |
| | PA3187 |
| | PA1809 |
| | PA1015 |
| | PA5545 |
| [Factor_7] | cyoB |
| | mexI |
| | hisH1 |
| | aruC |
| | PA0321 |
| [Factor_8] | aspA |
| | PA0749 |
| | PA1057 |
| | PA0716 |
| | PA0562 |
| [Factor_9] | PA0272 |
| | PA5411 |
| | glpR-glpD |
| | PA4516-PA4517 |
| | PA0217-PA0218 |
| [Factor_10] | pbpG |
| | aotP |
| | aruD |
| | wspD |
| [Factor_11] | PA1022 |
| | PA3185-oprB |
| | PA0034 |
| | PA3739 |
| [Factor_12] | PA1824 |
| | wspA |
| | PA0748 |
| | PA0574-PA0574.1 |
| [Factor_13] | aruE |
| | estA |
| | PA0158 |
| | PA3715 |
| [Factor_14] | kdpA |
| | aqpZ-PA4035 |
| | PA3961 |
| | PA1623 |
| [Factor_15] | PA0726 |
| | soj |
| | PA3925 |
| | PA5343 |
| [Factor_16] | holB |
| | PA0249 |
| | PA1076 |
| | cti |
| [Factor_17] | PA3432 |
| | PA3132-PA3133 |
| | PA3064 |
| | PA5234 |
| [Factor_18] | PA0752 |
| | PA5088 |
| | PA2773 |
| | pgi |
| [Factor_19] | PA3444 |
| | PA4799 |
| | PA0626 |
| | arcD |
| [Factor_20] | PA4675 |
| | PA1116 |
| | PA0674 |
| | ltaA |
| [Factor_21] | PA3130 |
| | metX |
| | braC |
| | PA2911 |
| [Factor_22] | PA2125-PA2126 |
| | PA5388 |
| | argR |
| | PA5094 |
| [Factor_23] | PA3265 |
| | dgt |
| | PA3044 |
| | PA5078 |
| [Factor_24] | ahpF |
| | lldD |
| | xcpY |
| | PA4093 |
| [Factor_25] | PA5270-PA5271 |
| | glcB |
| | PA1678 |
| | uraA |
| [Factor_26] | PA3170 |
| | PA0144 |
| | PA5099 |
| | PA4928 |
| [Factor_27] | PA5370 |
| | PA4359 |
| | PA5127 |
| [Factor_28] | PA3436 |
| | PA4734 |
| | PA4977 |

| | |
|---|---|
| [Factor_29] | PA1033 |
| | PA4108-ampR |
| | oruR-PA0832 |
| [Factor_30] | putP |
| | bioD |
| | tyrS-PA4139 |
| [Factor_31] | PA3423 |
| | katA |
| | PA0063 |
| [Factor_32] | rph |
| | mutS |
| | micA |
| [Factor_33] | oprG |
| | PA2745 |
| | PA4912 |
| [Factor_34] | PA0461 |
| | cyoC |
| | PA5508 |
| [Factor_35] | fumC2 |
| | PA3075 |
| | PA5113 |
| [Factor_36] | pilC |
| | PA4521 |
| | spuF |
| [Factor_37] | PA3312 |
| | PA4517 |
| | PA4888 |
| [Factor_38] | PA3927 |
| | plcH |
| | PA3597 |
| [Factor_39] | PA0181 |
| | sbp |
| | omlA-PA4766 |
| [Factor_40] | algU |
| | PA4735 |
| | cyaA |
| [Factor_41] | PA4041 |
| | PA5231 |
| | PA2042-PA2043 |
| [Factor_42] | queA |
| | PA3890 |
| | PA3595 |

| | |
|---|---|
| [Factor_43] | PA4886 |
| | PA0808 |
| | PA4075 |
| [Factor_44] | PA5507 |
| | plsB |
| | PA2793 |
| [Factor_45] | PA5528-PA5529 |
| | rnd |
| [Factor_46] | PA0485 |
| | PA3969 |
| [Factor_47] | PA2813 |
| | PA3272 |
| [Factor_48] | PA4440-PA4441 |
| | PA1031 |
| [Factor_49] | PA3697 |
| | anr |
| [Factor_50] | PA3676 |
| | PA4691 |
| [Factor_51] | PA0147 |
| | PA5165 |
| [Factor_52] | PA0354 |
| | PA0861 |
| [Factor_53] | leuA |
| | ccpR |
| [Factor_54] | PA4879 |
| | PA3913 |
| [Factor_55] | sucD |
| | PA0310 |
| [Factor_56] | PA0535 |
| | PA4686-hitA |
| [Factor_57] | pilT |
| | argF |
| [Factor_58] | PA0565 |
| | PA4332 |
| [Factor_59] | PA1427-PA1428 |
| | PA4635-PA4636 |
| [Factor_60] | fis-purH |
| [Factor_61] | flgJ |
| [Factor_62] | ftsX |
| [Factor_63] | lon |
| [Factor_64] | lpxO1 |
| [Factor_65] | PA0421 |

| | |
|---|---|
| [Factor_66] | PA0498 |
| [Factor_67] | PA0591 |
| [Factor_68] | PA0604-PA0605 |
| [Factor_69] | PA0659 |
| [Factor_70] | PA0847 |
| [Factor_71] | PA1037 |
| [Factor_72] | PA1201-PA1202 |
| [Factor_73] | PA1400 |
| [Factor_74] | PA1753 |
| [Factor_75] | PA1935 |
| [Factor_76] | PA2816 |
| [Factor_77] | PA2828 |
| [Factor_78] | PA3024 |
| [Factor_79] | PA3094 |
| [Factor_80] | PA3211 |
| [Factor_81] | PA3321 |
| [Factor_82] | PA3460 |
| [Factor_83] | PA3733 |
| [Factor_84] | PA3895 |
| [Factor_85] | PA3951 |
| [Factor_86] | PA4184 |
| [Factor_87] | PA4196 |
| [Factor_88] | PA4294 |
| [Factor_89] | PA4300 |
| [Factor_90] | PA4398 |
| [Factor_91] | PA4658 |
| [Factor_92] | PA4716 |
| [Factor_93] | PA4923-PA4924 |
| [Factor_94] | PA5149-PA5149.1 |
| [Factor_95] | PA5201 |
| [Factor_96] | PA5403 |
| [Factor_97] | PA5438 |
| [Factor_98] | pchC |
| [Factor_99] | pheA |
| [Factor_100] | pilB |
| [Factor_101] | polA |
| [Factor_102] | pta |
| [Factor_103] | purN |
| [Factor_104] | recQ |
| [Factor_105] | serA-PA0317 |
| [Factor_106] | xcpS |

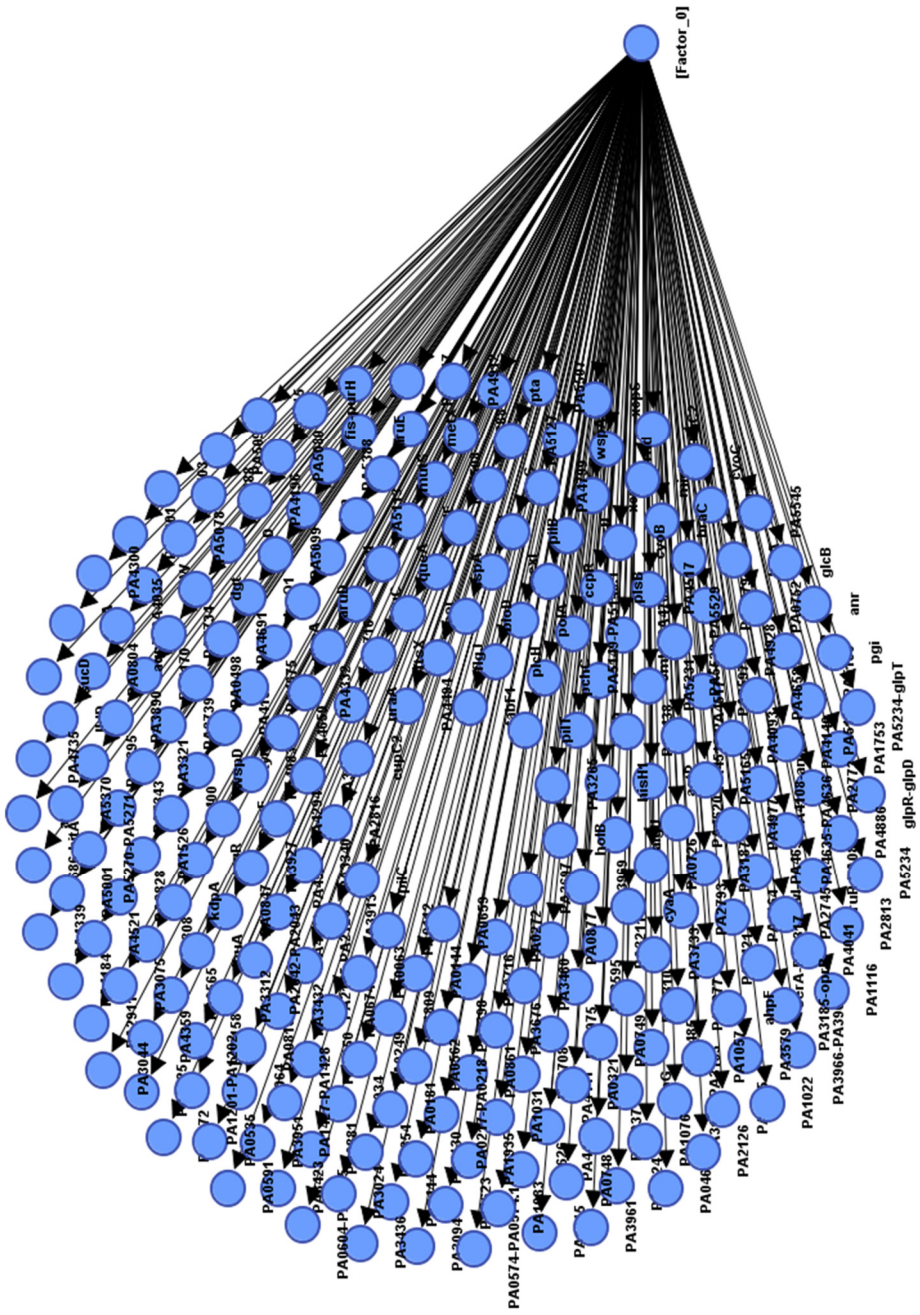



Figure A.3: Data cluster of the 249 genes, with *Factor_0* as computational representation of *P. aeruginosa*

Table A.3: The 2-states targeted clustering evaluations information

| Target: [Factor_0] | | |
|---|---|---|
| Value | C1 (-0.6753) | C2 (-0.1013) |
| Gini Index | 39.08% | 60.39% |
| Relative Gini Index | 99.47% | 99.47% |
| Lift Index | 1.4876 | 1.9073 |
| Relative Lift Index | 100.00% | 100.00% |
| ROC Index | 100.00% | 100.00% |
| Calibration Index | 100.00% | 100.00% |
| Binary LogLoss | 0 | 0 |
| **Statistics** | | |
| R | 1 | |
| R2 | 1 | |
| RMSE | 0 | |
| NRMSE | 0.00% | |
| Overall Precision | 100.00% | |
| Mean Precision | 100.00% | |
| Overall Reliability | 100.00% | |
| Mean Reliability | 100.00% | |
| Overall Relative Gini Index | 99.47% | |
| Mean Relative Gini Index | 99.47% | |
| Overall Relative Lift Index | 100.00% | |
| Mean Relative Lift Index | 100.00% | |
| Overall ROC Index | 100.00% | |
| Mean ROC Index | 100.00% | |
| Overall Calibration Index | 100.00% | |
| Mean Calibration Index | 100.00% | |
| Overall LogLoss | 0 | |
| Mean Binary LogLoss | 0 | |
| **Occurrences** | | |
| Value | C1 (-0.6753) (17) | C2 (-0.1013) (11) |
| C1 (-0.6753) (17) | 17 | 0 |
| C2 (-0.1013) (11) | 0 | 11 |
| **Reliability** | | |
| Value | C1 (-0.6753) (17) | C2 (-0.1013) (11) |
| C1 (-0.6753) (17) | 100.00% | 0.00% |
| C2 (-0.1013) (11) | 0.00% | 100.00% |
| **Precision** | | |
| Value | C1 (-0.6753) (17) | C2 (-0.1013) (11) |
| C1 (-0.6753) (17) | 100.00% | 0.00% |
| C2 (-0.1013) (11) | 0.00% | 100.00% |