Learning Petri Net Models of Non-Linear Gene Interactions

Abstract

Understanding how an individual's genetic make-up influences their risk of disease is a problem of paramount importance. Although machine learning techniques are able to uncover the relationships between genotype and disease, the problem of automatically building the best biochemical model or "explanation" of the relationship has received less attention. In this paper, I describe a method based on random hill climbing that automatically builds Petri net models of non-linear (or multi-factorial) disease-causing gene-gene interactions. Petri nets are a suitable formalism for this problem, because they are used to model concurrent, dynamic processes analogous to biochemical reaction networks. I show that this method is routinely able to identify perfect Petri net models for three disease-causing gene-gene interactions recently reported in the literature.

Keywords: Epistasis; Petri net; high order gene-gene interaction; multi-start random hill climbing.

1.0 Introduction

One goal of the biological sciences is to identify the DNA sequence variations in human populations that cause genetic diseases. Unfortunately, for most common genetic diseases such as sporadic breast cancer, this problem is difficult because gene interactions are non-linear (Ritchie et al., 2001). That is, individual genes in isolation may not be statistically significantly correlated with the presence of the disease in a population, but certain combinations of interacting genes are correlated.

Although machine learning and statistical techniques such as multi-dimensional reduction (Ritchie et al, 2001; Moore & Williams 2002) have been applied successfully to find these gene combinations given a database of genetic information, biochemists are also interested in finding "biochemically plausible" models of the causal influence of genes on disease. If a model is biochemically plausible to a

degree, then it may reveal characteristics of the actual biochemical pathways in humans that can aid understanding of the disease. Most existing machine learning and optimisation algorithms, however, are not designed specifically to this end to produce biochemically plausible models. For example, a typical machine learning algorithm might produce a decision tree or a set of rules, neither of which could be considered a biochemical explanation of the phenomenon.

A computational model that is considered biochemically plausible is the Petri net (Goss & Peccoud, 1988; Reisig, 1985). Petri nets are ideal for this purpose because they can be used to model parallel, interacting processes. Additionally, a direct analogy can be made between Petri net components and elements in a biochemical reaction network.

Petri net models are often designed by biochemists by hand. However, this approach is both time-consuming and difficult, and as the complexity of the models increases, it is likely to be infeasible. There is a need, therefore, for an automated solution to this problem (Moore & Hahn, 2003). To date, the problem of Petri net induction has not been addressed in the machine learning literature (to the author's knowledge), even though Petri nets are widely used elsewhere in computer science.

In this paper, the practical problem of automatically finding Petri net models of biochemical interactions is addressed. I start with a statistical model of how genetic variation across three genes influences the risk of disease, and show that a simple multi-start random hill climbing strategy can find a perfect Petri net model of the interaction. This is in direct contrast to earlier work by Moore & Hahn (2003) in which grammatical evolution was shown to find good, but not always perfect, solutions. The approach described here was also used to find a perfect solution to the much more complex four-gene non-linear interaction model for sporadic breast cancer, originally reported in the literature by Ritchie et al. (2001).

The paper is organised as follows. In Section 2.0, a brief introduction to genetics, Petri net biochemical modelling, and multistart random hill climbing is given. In Section 3.0, previous work on the problem of automating Petri net model building is

described. Section 4.0 details my approach to finding Petri net models, and Section 5.0 reports the results. Section 6.0 is the conclusion.

2.0 Background

In this section, I briefly overview the genetics motivating this work, and then introduce the idea of the Petri net and its biochemical analogy. I will also describe the multi-start random hill climbing search algorithm.

2.1 Genetics

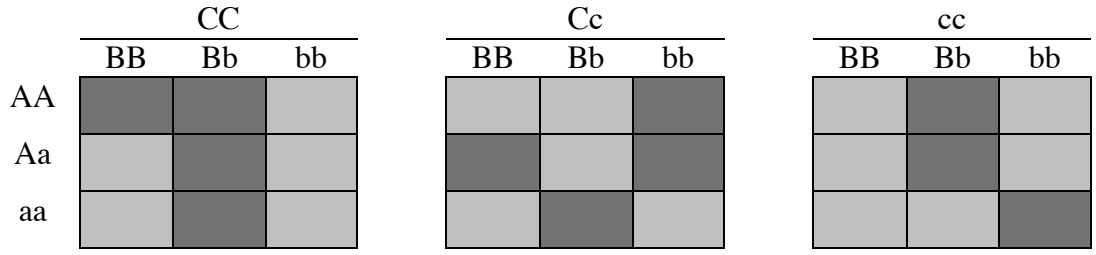
A genotype is defined as a combination of two alleles, one inherited from the mother, the other from the father. For example, if *A* and *a* are the alleles in a biological population, then the possible genotypes in individuals are *AA*, *Aa* or *aa* (the ordering of the alleles is unimportant). A consecutive ordering of genotypes such as these form a DNA sequence.

A disease is considered genetic if the presence of one or more genotypes is statistically correlated to the presence of a disease in a population. That is, there is a non-trivial conditional probability P(D|G) derived from frequency data of an individual having a disease D given a genotype G. This probability function is also called the penetrance (Moore & Hahn, 2003). For example, muscular dystrophy is a disease caused by a mutation to a single genotype. Its penetrance is high because the probability of the disease given the mutation is high. Diseases of this type are also called Mendelian because there is only a single genetic cause.

However, it is diseases that are caused by non-linear combinations of genes rather than Mendelian diseases that are the focus of this paper. Two examples are essential hypertension and sporadic breast cancer (Ritchie et al, 2001; Moore & Williams, 2002). In Figure 1, two competing non-linear models of the influence of genotypes on the risk of essential hypertension are depicted. Both models were discovered by Moore & Hahn (2003) using a genetic algorithm (Goldberg, 1989) to mine human genetic data obtained from an experiment.

The risk of disease in these examples is determined by the presence of three genes in various combinations, but the average penetrance of each single gene is approximately equal. Dark shaded cells represent high risk (more than 5% penetrance) while light unshaded cells represent low risk (less than 5% penetrance). Since each of the three genotypes can have three different values, there are a total of 3^3 or 27 different DNA sequence variations that a single individual could have.

(A) Model 1



(B) Model 2

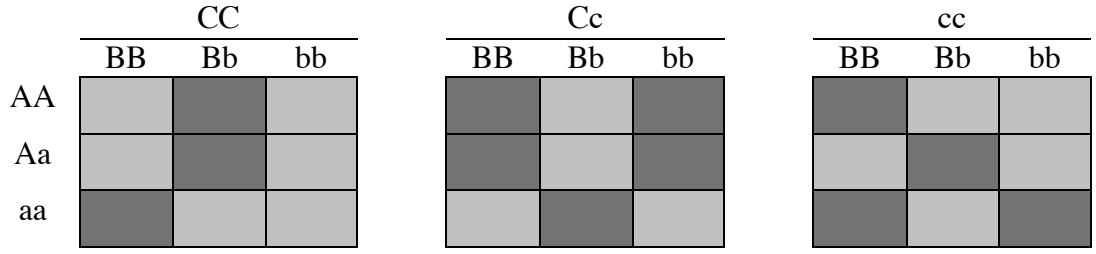


Figure 1: Two three-gene non-linear genetic models of essential hypertension risk originally discovered by Moore & Hahn (2003). Dark shaded cells are high risk; light shaded cells are low risk.

Figure 2 depicts a similar non-linear model of sporadic breast cancer risk. In contrast to the previous two models, which were three-gene models, this is a four-gene model. The data from which this model was derived was recently acquired after a controlled study involving 200 female subjects, and is believed to be the first four-gene non-linear model identified (Ritchie et al., 2001). Again, shaded cells indicate high risk and unshaded cells indicate low risk. The empty cells indicate a sequence variation not found in the sample, and of the 3^4 or 81 possible variations, 52 different sequences were present in the experimental sample. This four-gene interaction model is significantly more challenging to find a biochemically-plausible Petri net model for

than the three-gene models in Figure 1, simply because there is an exponentially larger number of number of variations that must be accounted for.

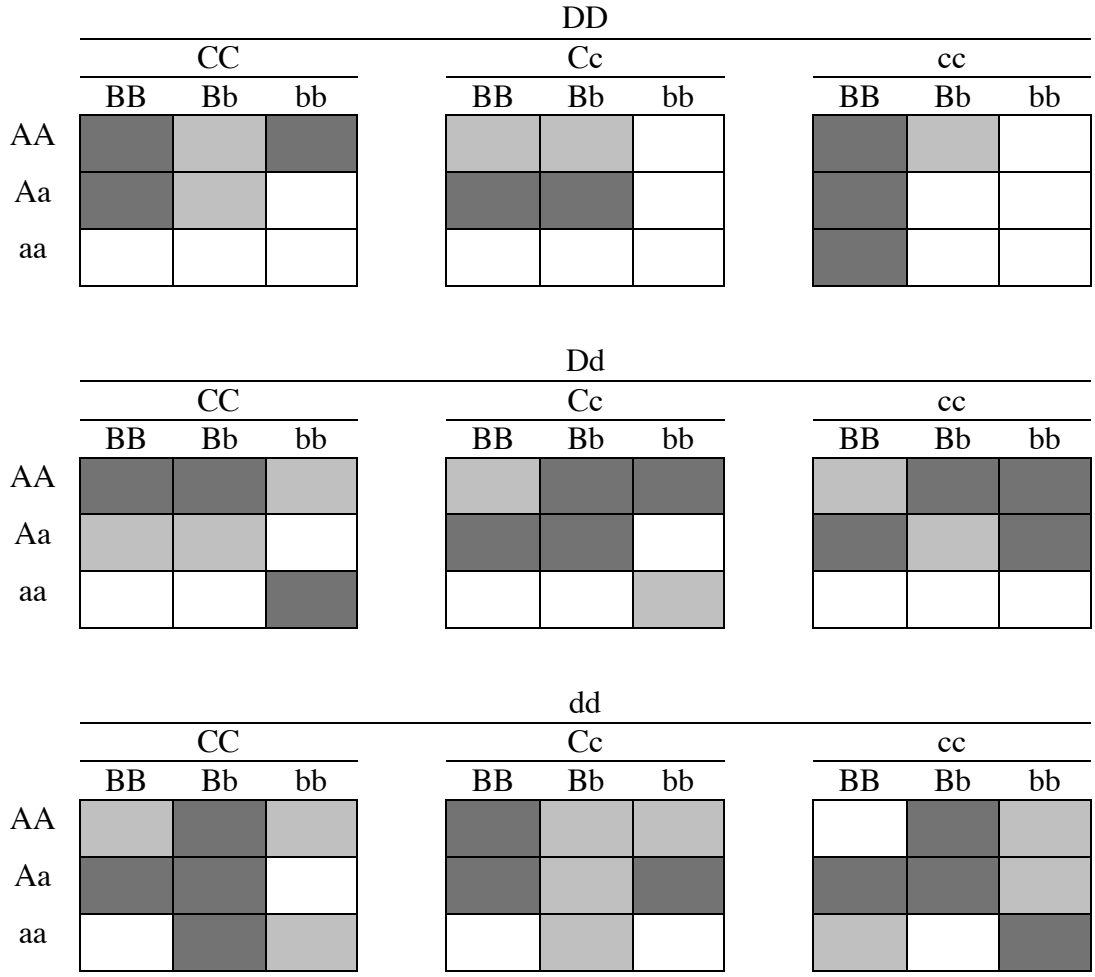


Figure 2: A four-gene non-linear genetic model of sporadic breast cancer first identified by Ritchie et al. (2001). Dark shaded cells are high risk; light shaded cells are low risk. Empty cells represent sequences not present in the experimental data.

2.2 Petri Net Modelling of Biochemical Networks

A Petri net is a computational formalism widely used for modelling distributed and concurrent computer systems (Reisig, 1985). Because Petri nets are intrinsically concurrent, they are also useful to biochemists seeking to model biochemical reaction networks (Goss & Peccoud, 1988). Figures 3 and 4 depict two simple Petri nets. Figure 3 simulates the process of protein synthesis.

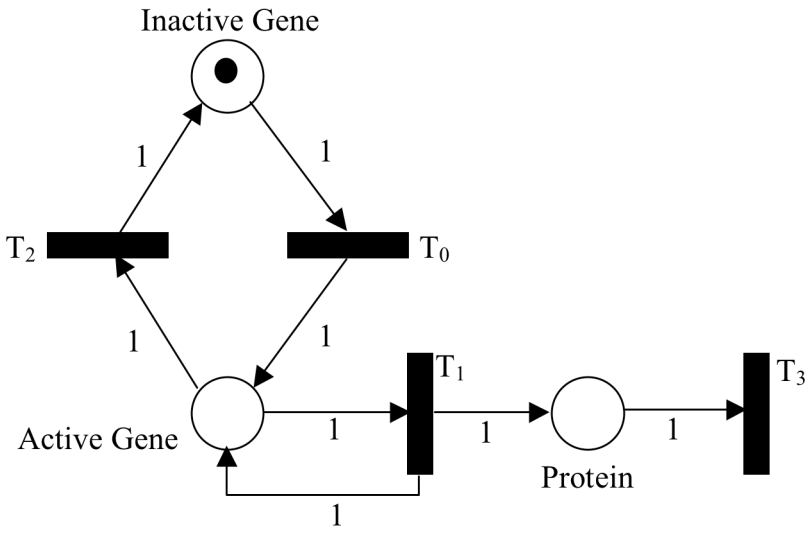


Figure 3. A simple Petri net model of protein synthesis from Goss & Peccoud (1988).

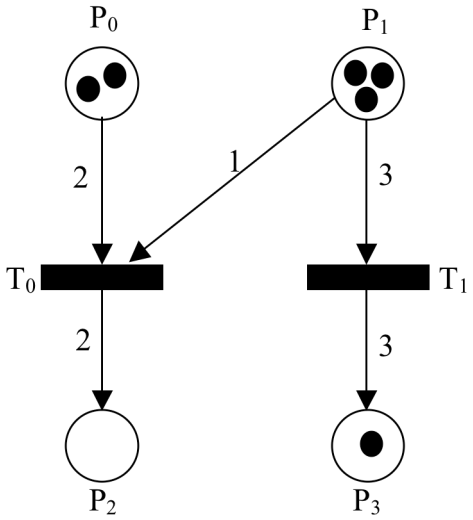


Figure 4. A Petri Net with two transitions enabled. The maximum capacity of each place in this network is four tokens.

Petri nets are essentially directed graphs, with tokens that "move around" the graph as reactions take place. There are two types of nodes in a Petri net graph: places, depicted as circles, and transitions, which are narrow black rectangles. Arcs may only be directed from place to transition (in which case they are referred to as input arcs), or transition to place (making them output arcs). The implication of this is that a Petri net is always bipartite.

Each place and each arc is also assigned a non-negative integer. In the case of places, this integer specifies the maximum number of tokens (i.e. the maximum capacity) of the place. In the case of arcs, it is called a weight, and it controls the rate at which tokens can move from place to place via the transitions.

Petri nets are executable in that tokens move around the network from place to place as transitions fire. A place can have zero or more tokens present at a discrete point in time, and the distribution of tokens at that time is called a marking. The presence of a token at a place is indicated by a black dot inside the circle representing a place. To illustrate, the marking of the Petri net depicted in Figure 3 is (1,0,0) and the marking of the Petri net in Figure 4 is (2,3,0,1).

A transition is enabled and can "fire" if (i) the number of tokens at each input place is at least equal to the weight on the arc connecting the place to the transition (i.e. there are sufficient input tokens to be consumed), and (ii) there is sufficient unused capacity at each output place to receive the number of tokens as specified on the output arc. The process of firing a transition involves removing tokens from each input place equal to the input arc weights, and adding tokens at each output place equal to the output arc weights. Transitions may have no input or no output places, in which case they can act as either infinite sources or as infinite sinks of tokens.

To illustrate the process of firing, in Figure 3, transition T_0 is enabled because its input arc weight is 1 and there is one token at the "Inactive Gene" place. If T_0 fires, a token will be removed from "Inactive Gene" and added to "Active Gene" . This in turn enables transition T_1 which, if it fires, consumes a token from the "Active Gene" place but immediately replenishes it because of the self-loop. It also adds a token to the "Protein" place. The marking at this point would be (0,1,1). Similarly, in Figure 4, if transition T_0 fires, the marking will change from (2,3,0,1) to (0,2,2,1). Alternatively, transition T_1 is also enabled and could fire instead, in which case the marking would become (2,0,0,4).

Deadlock occurs if two transitions are both enabled, but the cannot both fire because of a limited number of tokens. Figure 4 illustrates this nicely: both transitions T_0 and

T_1 are enabled, but if one fires it would consume sufficient tokens at P_1 to disable the other. There are various strategies for resolving deadlock, but here I will be concerned only with the simplest strategy of giving the transitions an arbitrary priority.

An analogy between Petri nets and biochemical networks is described in detail by Goss and Peccoud (1988). The idea is that places represent molecular species, and tokens correspond to individual molecules in a biochemical network. A marking, therefore, is the distribution or concentration of molecules over molecular species at a particular point in time. Transitions correspond to chemical reactions, implying that input places are the reactants and output places are the products of the reactions. The arc weights are reaction rates, and an enabled transition means that a reaction is possible although it may not necessarily occur. In this context Petri nets can be considered biochemically plausible.

2.3 Multi-Start Random Hill Climbing

Multi-start random hill climbing is one of the simplest yet most effective search strategies for problem solving. In terms of evolutionary computation, it corresponds to a (1, 1) Evolutionary Strategy (Beyer, 1994) in which a single individual is always replaced by a single, better, individual, if one is found. For those unfamiliar with the approach, the basic idea is that there is a value function to be maximised. The search is initialised with a randomly generated individual (a Petri net in this case), and random changes or "mutations" are made. Each change is only accepted if it increases the value of the Petri net. By this continual process, the value is expected to increase with time until, ideally, it reaches the maximum possible value at which point the search can stop.

One well-known problem with single-start hill climbing is the presence of local optima. In other words, a hill climbing strategy may lead the search to a point where all further mutations fail to increase the value function, but the current value is not maximised. Further mutations in this situation are futile. To overcome this problem, the search is simply restarted from a new randomly generated individual if it is detected that a local optima has been reached. I assume that a local optima has been

reached if no improvements are made after a fixed number of attempts, given by a constant called the NO_IMPROVEMENT_LIMIT. The total number of evaluations globally, over all the runs, is also recorded, and the search is terminated if either an individual with maximum value is found, or the total number of evaluations exceeds a certain maximum, global limit.

3.0 Related Work

This work builds on research reported by Moore & Hahn (2003), who originally identified the non-linear models depicted in Figure 1 and subsequently devised a grammatical evolution-based approach to find near-perfect Petri nets explaining these models.

The key idea behind Petri net modelling in the context of disease-causing genes is that biochemical reactions produce concentrations of a particular toxic substance directly related to the risk of disease. In particular, if the concentration of this toxic substance exceeds a certain amount, then a high risk assignment is made. In terms of the analogy with Petri nets, this implies that one place is a "target" or output place (representing the toxic substance), and if the number of tokens present at the place is greater than some discrete fixed threshold, then a high-risk assessment is made. Petri nets that make identical assessments for each DNA sequence variation as specified by the non-linear models in Figures 1 and 2 can be considered perfect.

One critical issue facing Moore & Hahn was how to select which aspects of the Petri nets were to be fixed *a priori*, which were to be subject to mutation but independent of genetic variation, and which were to be subject to mutation and dependent on genetic variation. A Petri net can vary along many possible dimensions, such as the number of places, the number of transitions, the maximum place capacities, the arc weights and directions, the initial token markings, the number of transitions to be fired, the firing delays, and so on. They decided that the best approach was to let the search algorithm itself select how the variables and their dependencies would be determined. This resulted in quite a vast search space, but the researchers had a 110-CPU Linux-based parallel cluster on which to perform their experiments.

Another critical issue is how to represent a Petri net. Since Moore & Hahn chose to use a recent evolutionary optimisation technique known as grammatical evolution (O'Neill & Ryan, 2001), they had to be able to represent a Petri net as a bit string. This required them to construct a context-free grammar defining the space of all possible Petri nets as well as their genetic variations. Terminals in the grammar represented the basic units of a Petri net such as places, arcs, or transitions, and also encoded whether or not the units depended on a DNA sequence variation (and if so, which variation), or if they were completely independent of sequence variation.

After multiple runs of grammatical evolution on their parallel cluster, a perfect solution was found for Model 2 (Figure 1B), but no perfect solution was ever found for Model 1 (Figure 1A). The best solution for Model 1 misclassified disease risk for a single sequence variation. Additionally, the researchers attempted to find a good Petri net solution for the four-gene model in Figure 2, but they were unsuccessful. The main problem was apparently the exponentially larger search space.

4.0 My Approach

In contrast to the previous work, I believe that better Petri net optimisation can be achieved by severely restricting the parameter space being searched. Moore & Hahn (2003) allowed for the possibility of variation on almost all facets of their Petri nets. In other words, different genotypes could determine different arc weights, connectivity, place capacities, and so on. Nearly every variable related to the Petri net was potentially gene dependent.

The opposite approach is to start from the assumption that genes may only influence a very small portion of the Petri net, with the rest of the network being independent of genetic variation. This should result in a significantly smaller search space, with a correspondingly increased chance of finding an optimal solution if one exists inside the search space.

I chose, for the experiments reported here, to let only the initial distribution of tokens or molecules across the network (i.e. the initial marking) be genotype dependent, with the remainder of the variables (such as the structure of the network and the arc weights) being genotype independent. It could be argued that this is analogous to a situation where genes have the dual function of both producing molecules such as proteins, and also regulating the production rates of other genes. I have also globally fixed the number of places and the number of transitions to further reduce the search space, as runs in which the number of places and/or transitions were allowed to vary unequivocally resulted in poor performance. This means that the search may need to be run several times, with different numbers of places or arcs each time, before an optimal solution is found.

The basic value function to be maximised is the performance of the Petri net on the task of modelling an accumulation of a particular toxic substance that causes a disease. This is represented simply as the number of tokens at a designated place. If the number of tokens exceeds a fixed threshold, defined as 5% of the maximum capacity of the place after a certain amount of time (the same rule as used by Moore & Hahn, 2003), then the risk assignment is high; otherwise it is low. This is the same rule as used by Moore & Hahn (2003). For the model to be perfect, therefore, all high risk genetic variations should lead to high risk assignments; all low risk variations should avoid such an assignment. For the three-gene models, the maximum value is therefore 27 (being 27 correct assignments) and for the four-gene model it is 52.

How does the genotype, then, influence the initial marking? The basic approach is to assign a quantity of tokens to each place for each genetic variation. For example, if a gene can have values *AA*, *Aa* or *aa*, and the maximum place capacity is 10 tokens, then the quantity for *AA* could be 10 tokens, the quantity for *Aa* might be 5 tokens, and the quantity for *aa* may be 0 tokens. The quantities are completely arbitrary and different values could have been used; the key point is to distinguish between different genotypes by assigning them different quantities of tokens for the initial marking. To take this a step further, if a similar encoding scheme is used for all the genes, then a complete DNA sequence such as *AabbCC* leads to an initial marking of

(5, 0, 10) over the first three places of the network, uniquely specifying the entire portion of the genotype being considered.

One additional place (which, in these experiments, is not one of the places used to encode the input) is designated to be the output place representing the "toxic" or disease-causing substance. After firing the network a number of times given the initial marking, this will be the place where the concentration of tokens is measured in order to determine the risk assessment. There are, therefore, a minimum of four required places for the three-gene models in Figure 1 (being three places encoding the input and one output place), and five required places for the four-gene models. This contrasts to Moore & Hahn's approach, which found Petri net models with only one place.

The focus of the multi-start random hill climbing, then, is on mutating the connectivity and arc weights between the places and transitions. In other words, the search space is the set of possible reactions and reaction rates between molecular species, with the initial concentrations of molecular species being determined genetically.

Representation is also an important issue in any search problem. Poor representations can easily cause the search to fail. In contrast to the graph representation depicted in the figures above, and also the bit-string representation used in the paper previously discussed, I have chosen to use the simplest possible representation of a Petri net as an array of integers called an incidence matrix. Only Petri nets with self-loops cannot be defined in this way (Reisig, 1985).

To elaborate, an incidence matrix is a fixed-size table with places along one dimension and transitions along the other. Each place P_i and transition T_j has an entry $<P_i,T_j>$ in the matrix defining the presence or absence of a directed arc. If $<P_i, T_j>$ is positive, then the arc is directed from P_i to T_j (i.e. P_i is an input of T_j). If $<P_i, T_j>$ is negative, the the arc is directed from T_j to P_i (P_i is an output place for T_j). In both cases, the absolute value of $<P_i, T_j>$ also indicates the weighting on the arc. A value of

zero indicates that there is no arc between T_j and P_i. To illustrate, Figure 5 depicts an incidence matrix for the Petri net shown in Figure 4.

	$\mathbf{T_0}$	$\mathbf{T_1}$
$\mathbf{P_0}$	2	0
$\mathbf{P_1}$	1	3
$\mathbf{P_2}$	-2	0
$\mathbf{P_3}$	0	-3

Figure 5. Incidence matrix for the Petri net in Figure 4.

This incidence matrix representation of Petri nets is used in the system described here. In each case, the maximum place capacities, the total number of transitions and places (i.e. the size of the table), and the maximum and minimum values of the weights are globally fixed, but they can be varied manually between runs.

A mutation operator was also defined so that new variants "close" to the current best network could be generated. Again, the simplest strategy was utilised: a single, randomly selected, entry in the incidence matrix was randomly changed to another value between the minimum and maximum arc weight. This effectively allowed a single mutation to modify, add, delete, or invert an arc.

In order to resolve conflict between multiply-enabled (i.e. deadlocked) transitions, an arbitrary delay ordering of transitions was assumed. That is, each transition T_m has a unique index *m*, such that transition T_m will always fire before transition T_n whenever $m<n$. This results in a deterministic Petri net.

In the next section, the results of the experiments are discussed. All of the simulations were performed on a single MacOSX G4 Powerbook with 640MB of RAM and an 867 Mhz G4 processor. The Petri net simulation code and the search algorithms were written in C++.

5.0 **Results**

In each run of the multi-start hill climber, the maximum place capacity was set to 10, and the minimum and maximum weights for each arc were set to -6 and 6 respectively. The NO_IMPROVEMENT_LIMIT constant for deciding whether or not to restart the hill climb was set to the number of places multiplied by the number of transitions. All of these values were found by trial-and-error to be the most effective, although they are by no means likely to be the only combinations of values that could have solved the problem.

It was also found that a slight modification to the multi-start random hill climbing improved the search efficiency. Rather than accepting only mutations that increase value, mutations were accepted as long as they did not decrease value. This effectively means that when the search reaches a "plateau" in which all similar Petri nets have the same value, a random walk strategy is employed to explore the plateau.

5.1 Three-Gene Interaction Models

When learning Model 1, trial-and-error led to the number of places and transitions were being fixed at 5. Of the five places, three represented the initial substances derived form the DNA sequence variations, one represented an intermediate substance, and the last place represented the toxic disease-causing substance. A total of 60,007 evaluations were required, and there were 60 re-starts of the hill climber. A perfect model was found after 30.38 seconds. In the case of Model 2, 8 transitions were required, but this time only 4 places were needed. There were 49 restarts for a total of 41,151 evaluations which took 39.94 seconds. In Figure 6, solutions for both Models 1 and 2 are shown in incidence matrix form.

(A) Model 1

	T_0	T_1	T_2	T_3	T_4
P_{GeneA}	-1	0	1	-5	-6
P_{GeneB}	0	5	-1	2	-6
P_{GeneC}	3	-1	-5	1	-6
$P_{Inter.}$	-2	2	-3	3	-1
P_{Toxic}	4	-6	-2	-4	-5

(B) Model 2

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
P_{GeneA}	1	-3	-4	-5	4	0	-6	5
P_{GeneB}	2	-3	0	-6	3	6	4	-4
P_{GeneC}	0	-1	-2	3	-3	-5	3	2
P_{Toxic}	2	0	-1	4	5	-5	-3	0

Figure 6: Perfect solutions for Models 1 and 2 from Figure 1.

5.2 Four-Gene Interaction Models

The four-gene model from Figure 2 represents a search space exponentially harder than the three-gene models. However, random hill-climbing was still able to find a perfect model. The perfect model was expectedly larger than the previous two with 7 places and 12 transitions, and it was found after a 38,899 evaluations but only 2 restarts. The total CPU time spent was 2.01 minutes.

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
P_{GeneA}	-3	0	4	-5	5	1	4	6	5	-1	0	-3
P_{GeneB}	-1	-3	1	2	0	3	2	5	1	0	-3	5
P_{GeneC}	0	3	-6	1	-1	-2	2	0	0	-3	3	0
P_{GeneD}	-5	2	4	-6	5	5	5	-6	0	0	3	-4
P_{Inter1}	2	-1	-3	-6	-6	-5	-6	-3	-4	-5	-2	-5
P_{Inter2}	-1	3	-5	-3	-2	-1	-5	-3	0	-5	-6	2
P_{Toxic}	3	1	-1	-5	-5	-6	-5	-4	2	-6	-6	2

Figure 7. A perfect Petri net for the 4-gene problem depicted in Figure 2.

6.0 Conclusion

Having demonstrated that random hill-climbing is sufficient for learning Petri nets with deterministic dynamics, the assumption of determinism now needs to be addressed. In other words, is it more realistic for multiply-enabled transitions to fire stochastically, rather than according to an arbitrary delay order? Should "noise" be added to the initial markings and the transition outputs, so that the net's local behaviour is not completely predictable? The answer to these questions ultimately will depend on the application and the degree of simulation "granularity" that is required; if stochastic Petri nets are preferred for some applications, then a more powerful search strategy may be required to discover them. This is an open and important question for future research to address.

To conclude, I have described and evaluated an efficient method of learning Petri net models of non-linear gene interactions. This method could be embedded in a much larger system in which a user first of all mines genetic data using statistical methods (in order to discover the gene combinations responsible for a particular disease), and then produces a Petri net-based explanation or model of how the genes may lead to the disease. Previous work used a more complex grammatical evolution approach that failed to find solutions for all of the known cases of non-linear interactions appearing in the literature and discussed here; I have shown that a simpler hill-climbing based strategy combined with an incidence matrix-based representation is more effective.

Acknowledgements

The author would like to thank Dr Tony Smith, Department of Computer Science, University of Waikato, New Zealand, for reading an early draft of this paper and providing useful feedback.

References

Beyer, H. 1994. Towards a theory of evolutionary strategies: the (mue, lambda)-theory. *Evolutionary Computation* 2(4): 381-407.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goss P. and Peccoud J. 1988. Quantitative Modelling of Stochastic Systems in Molecular Biology by Using Stochastic Petri Nets. *Proceedings of the National Academy of Sciences*, 95: 6750-6755.

Moore J. and Hahn L. 2003. Petri Net Modelling of High-Order Genetic Systems using Grammatical Evolution. *BioSystems* 72: 177-186.

Moore J. and Williams S. 2002. New Strategies for Identifying Gene-Gene Interactions in Hypertension. *Annals of Medicine*, 34: 88-95.

O'Neill M. and Ryan C. 2001. Grammatical Evolution. *IEEE Transactions on Evolutionary Computation* 5: 349-358.

Reisig, W. 1985. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag.

Ritchie M., Hahn L., Roodi N., Renee Bailey L., Dupont W., Parl F., and Moore J. 2001. Multifactor-Dimensionality Reduction Reveals High-Order Interactions among Estrogen-Metabolism Genes in Sporadic Breast Cancer. *American Journal of Human Genetics*, 69:138-147.