Brock University

Department of Computer Science

The Evolution of Higher-level Biochemical Reaction Models

Brian J. Ross
Technical Report # CS-10-02
December 2010

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

The Evolution of Higher-level Biochemical Reaction Models

B. J. Ross bross@brocku.ca
Department of Computer Science, Brock University, St. Catharines, ON, L2S 3A1, Canada

Abstract

Computational tools for analyzing biochemical phenomena are becoming increasingly important. Recently, high-level formal languages for modeling and simulating biochemical reactions have been proposed. These languages make the formal modeling of complex reactions accessible to domain specialists outside of theoretical computer science. This research explores the use of genetic programming to automate the construction of models written in one such language. Given a description of desired time-course data, the goal is for genetic programming to construct a model that might generate the data. The language investigated is Kahramanoğullari's and Cardelli's PIM language. The PIM syntax is defined in a grammar-guided genetic programming system. All time series generated during simulations are described by statistical feature tests, and the fitness evaluation compares feature proximity between the target and candidate solutions. Target PIM models of varying complexity are used as target expressions for genetic programming. Results were very successful in all cases. One reason for this success is the compositional nature of PIM, which is amenable to genetic program search.

Keywords

Genetic programming, grammar-guided, biochemical modeling, time-series, statistical features, process algebra.

1 Introduction

Systems biology involves the mathematical and computational modeling of biochemical systems (Bower and Bolouri, 2001; Fisher and Henzinger, 2007; Tkacik and Bialek, 2009). One such application area is functional genomics, which is concerned with the modeling of gene and protein interactions, usually with respect to time. Research is ongoing for new computational tools and techniques that will aid in the automation of effective models for increasingly complex biological phenomena (Markowetz, 2005; Markowetz and Spang, 2007; Chou and Voit, 2009; Hecker et al., 2009). The practical benefits of such tools are many. Given laboratory data such as time-varying protein and enzyme levels measured for biochemical reactions, the ability to automatically determine viable genetic, biochemical, and cellular mechanisms that might give rise to them is a valuable analytical tool. Furthermore, novel discoveries might be found with these tools, since new mechanisms might arise that were not previously considered by humans. When this technology matures, it should be possible to specify to the computer the desired end result of a biological system, for example, particular time-course patterns of proteins. Potential biological models that might give rise to these requirements could be discovered by the computer system, to could be given further consideration by the biologist.

This paper uses genetic program to automatically construct models of biological reactions. A new modeling language by Kahramanogullari and Cardelli (2009) called PIM (Programming Interface for Modeling) is considered. PIM describes biochemical reactions at a high-level, in which basic interactions between species (proteins, genes) and their binding sites are described. The interpretation of PIM models results in a set of time series plots. These plots denote the level of interactions between various components of the biochemical molecules being modeled. The stochastic nature of the language means that separate simulations of the same PIM model can produce different behaviors. In fact, depending on the model, behaviors can vary greatly between simulations. Therefore, time series are characterized by statistical features, which permit a variety of stochastic and chaotic behaviours to be represented. Fitness evaluation then finds a closeness of fit between the statistical features of candidate PIM models and some target behavior of interest. The PIM language itself is implemented within a grammar-guided genetic programming systems. This permits a nearly direct translation between PIM expressions and GP grammar trees. It also permits useful grammatical constraints to be specified, for controlling the search space complexity.

The main motivation behind this research is to investigate the effectiveness of genetic programming in evolving models written in the new generation of biological modeling languages such as PIM. Although the target models investigated here are necessarily limited in scope and realism, being able to successfully reconstruct them with genetic programming shows the promise of this technology in future applications. Automatically synthesizing viable models for real-life laboratory data is an important application area for computational intelligence.

The paper is organized as follows: Section 2 reviews related research that uses genetic programming for biochemical modeling. The PIM language is reviewed in Section 3, and its implementation for the genetic programming system is discussed in Section 4. Model evaluation strategies are outlined in Section 5. Experiments are described in Section 6, and the results are shown in Section 7. A discussion of the results, and comparisons with related work, are given in Section 8. Concluding remarks are given in Section 9.

2 Related Work

Biological network models can be classified as being deterministic or stochastic. Deterministic models are those whose behaviors are determined entirely by the starting environment. In an evolutionary computation setting, the target behavior's determinism can be exploited during fitness evaluation, by performing error fitting between the time series curves of the candidate process and the target. Deterministic bio-networks have been evolved with genetic algorithms, for example, networks represented by Petri nets (Kitagawa and Iba, 2003), MP systems (Castellini and Manca, 2009), and S-systems (Kikuchi et al., 2003). GP infers deterministic metabolic networks from target time series, using an intermediate representation having the form of an artificial electronic circuit (Koza et al., 2000). GP is used to evolve gene regulatory networks taking the form of nonlinear differential equations (Sakamoto and Iba, 2001; Ando et al., 2002; Streichert et al., 2004; Cho et al., 2006; Floares, 2008; Qian et al., 2008). GP has also been used to evolve S-system models (Wang et al., 2007). Banzhaf (2003) develops a conceptual model of artificial regulatory networks using genetic programming. Lee and Yang (2008) combine GP and neural networks to evolve gene regulatory networks.

Stochastic networks incorporate a probabilistic element to the network semantics, to account for the stochastic noise seen in biological systems (Raj and van Oudenaar-

Model	::=	*Sentence ... Sentence*
Sentence	::=	site **on** species *Operator* site **on** species {*Rate*} {*Conditions*} |
		species **becomes** species {*Rate*} |
		species **decays** {*Rate*}
Operator	::=	**associates** | **dissociates** |
		gets phosphorylated | **gets dephosphorylated**
Rate	::=	**with rate** float
Conditions	::=	*Condition ... Condition*
Condition	::=	site **on** species **is bound** | site **on** species **is unbound**

Table 1: PIM syntax

den, 2008). This stochastic noise makes the characterization of behavior less straightforward than with deterministic processes, given the (often significant) variability seen during separate simulations. Genetic algorithms are used to evolve stochastic bio-networks in Drennan and Beer (2006) and Chu (2007). Stochastic oscillating behaviors are evolved using GP in Leier et al. (2006), Imada and Ross (2010) and Ross (2010). Fuzzy rules are evolved with GP in Linden and Bhaya (2007), which although not probabilistic, does account for stochastic behavior.

The problem of evolving stochastic bio-networks is closely related to the more general problem of evolving models for noisy, chaotic time series. Although such time series may be deterministic in origin, techniques for handling noise are often applicable to stochastic models as well. Examples of the use of GP in this area include (Angeline, 1998; Zhang et al., 2004; Rodriguez-Vazquez and Fleming, 2005; Schwaerzel and Bylander, 2006; Borrelli et al., 2006; Zhang et al., 2004).

3 The PIM Language

Process algebra are mathematical languages used to model and analyze concurrency (Hoare, 1985; Milner, 1989). Specialized stochastic process algebra have been derived, such as the stochastic pi-calculus, which have been found suitable for modeling biological networks (Priami, 1995; Blossey et al., 2006; BioSPI, 2010). The stochastic pi-calculus features a quantitative characterization of the state of a model, since quantities of components denoting molecules, proteins, or gene expression levels can be measured and displayed during a simulation. This behavior takes the form of time series plots. This has a correspondence to equivalent time-course behaviors measured in the laboratory.

One disadvantage of process algebra is that they are esoteric formalisms that are challenging to learn and master, even for seasoned computer programmers. The majority of practitioners using them are theoretical computer scientists specializing in process algebra research. Needless to say, most biologists do not have the time nor desire to master such arcane formal languages.

In an effort to make biochemical modeling accessible to those in the natural sciences, higher-level biological modeling languages have been proposed (Blossey et al., 2006; Danos et al., 2007; Guerriero et al., 2007; Dematte et al., 2008; Kahramanogullari and Cardelli, 2009). These languages discard the descriptive generality and computational power of pure process algebras, in favor of domain-oriented languages that describe application problems at a higher level. These languages are designed to naturally express particular classes of problems in biology and chemistry, using domain-based nomenclatures familiar to domain experts.

One example of a language for biological modeling is the PIM language (Kahramanogullari and Cardelli, 2009) (Figure 1). PIM describes basic reaction events at a high level. For example, sentences describe how specific proteins may interact at particular binding locations, and the conditions under which such interactions proceed. PIM is also compositional, so that models can be refined by adding additional terms and sentences. Once a model is described in PIM, a PIM translator will convert it to a lower-level stochastic pi-calculus system, which can then be simulated by a stochastic pi-calculus interpreter. The time-series curves are normally interpreted as quantities of molecules (components, genes,...) having particular interactions in the system as a whole (more later). PIM simulations therefore exploit the computational advantages of the stochastic pi-calculus, as resulting time-course plots emulate the stochastic and chaotic characteristics seen in real biological systems. However, PIM models are considerably simpler to specify and understand than models written in lower-level process algebra.

The following summary of PIM is necessarily of limited depth; please refer to Kahramanogullari and Cardelli (2009) for details.

In Figure 1, a model is composed of a number of sentences. Each sentence describes a basic reaction event. Reaction events occur on species, which denote specific kinds of molecules (proteins, genes). The locations on species at which events occur are their sites. Species and sites are denoted by problem-specific labels. The following reaction events are possible. An *association* event is when a site on a species binds with a site on another species. *Dissociation* is when such an association is undone. Phosphorylation and its converse are similar, except the association happens with a phosphate on the species and site specified. Transformations occur when a species becomes another species. Degradation is when a species decays. All these events happen at particular frequencies or *rates*. If a rate is specified, a floating-point number species its frequency (higher numbers are more frequent). The default rate is *1.0*. Finally, condition terms can be used, which act as constraints for sentences. Conditions specify that particular sites on a species should be bound or unbound before that event can occur.

An example of a sentence is the following:

```
site a on ProteinA associates site b on ProteinB with rate 2.5
    if site b on ProteinA is bound
```

ProteinA and ProteinB are labels describing specific protein species of interest. Site “a” on ProteinA is identified as a site that may bond with site “b” on ProteinB with a rate of 2.5. This event only happens when site b on ProteinA is in a bound state. Note that site labels are unique to species, and so the two instances of site “b” in the sentence are referring to different locations.

Although the syntax of PIM sentences is very straight-forward, it is still possible that models can be ill-formed. A number of correctness conditions are therefore imposed on sentences in a model, to ensure that the model is definable:

1. Species in conditions are referred to in body of sentence.

2. No contradicting conditions in a sentence.

3. Sites in the conditions are defined elsewhere in the model.

4. Associations refer to unbound species.

5. Dissociations refer to bound species.

6. Transformations refer to unbound species.

7. Conditions do not overlap between sentences.

For example, condition 2 refers to the situation in which two conditions in a sentence are contradictory ("site b on B is bound and site b on B is unbound"), since the meaning cannot be determined. Note that adhering to these conditions does not ensure that the model being written is in any sense correct with respect to the biological phenomenon being studied. Rather, these correctness conditions discourage ambiguities that are detectable via syntactic analysis of the model.

A complete PIM model for some phenomenon of interest consists of multiple sentences. The intension is that the sentences describe all the aspects of interest in a given biochemical reaction. The model will also include a statement that bootstraps the simulation, by specifying the total initial quantities of base species, for example, 1000 copies of two proteins:

$$1000@\ (\ \text{ProteinA}\ |\ \text{ProteinB}\ )$$

An adequate description of PIM's model execution semantics is certainly beyond the scope of this paper. The essence of the semantics is that the sentences contribute to the overall simulation according to the quantities of species, the satisfaction of conditions, and rates of activity. The execution engine (described below) is founded on Gillespie's algorithm, which is well-known in the field of chemical simulations (Gillespie, 1977). The effect of running Gillespie's algorithm on the PIM model is that stochastic behavior arises, with characteristics that are reminiscent of what is seen in laboratory experiments.

The procedure for performing PIM simulations is as follows. The model is read into a PIM translator (Kahramanogullari and Cardelli, 2009; Kahramanogullari, 2010). Presuming the model adheres to the correctness conditions, it is then translated into an equivalent model written in the stochastic pi-calculus. This is then interpreted using SPIM – a stochastic pi-calculus interpreter (Phillips and Cardelli, 2004; Phillips, 2007). SPIM execution results in a set of time-series plots (Figure 4). The number of plots depends on the number of active sites in the model. A species *S* with k sites will result in 2^k plots. Each plot represents a combination of the sites defined on S (in other words, the powerset of the sites). For example, if *S* has 3 sites *a*, *b*, and *c*, then the possible combinations are $\{\epsilon, a, b, c, ab, ac, bc, abc\}$. The plot denotes a time-course measurement of the quantity of instances of a combination of sites being simultaneously in a bound state. The biological interpretation of this plot is that it represents the quantity changes over time of the number of molecules having that particular physical state during a reaction.

4 A grammatical definition of PIM for genetic programming

DCTG-GP is a grammar-guided genetic programming system (Ross, 2001). The key advantage of grammar-guided GP is the ease in which new languages can be defined for genetic programming. Appropriate grammatical constraints to inhibit search are also readily imposed. The system uses a definite clause translation grammar (DCTG) to define target languages. A DCTG is a logic-based attribute grammar, which permits

1	model	::=	sentences
2a	sentences	::=	sentence
2b	sentences	::=	sentence, sentences
3a (assoc, dissoc)	sentence	::=	site, species, site, species, rate, conditions, detach
3b (phos, dephos)	sentence	::=	site, species, rate, conditions, detach
3c (trans)	sentence	::=	species, species, rate
3d (decay)	sentence	::=	species, rate
4a	detach	::=	*null*
4b	detach	::=	rate, conditions
5a	rate	::=	*null*
5b	rate	::=	float
6a	conditions	::=	*null*
6b	conditions	::=	site, species, condtype
7	condtype	::=	bound | unbound

Table 2: Context-free grammar definition of PIM.

syntactic and semantic definitions for a language to be defined together. DCTG-GP is implemented in Prolog (Clocksin and Mellish, 1994).

The context free grammar definition (CFG) of PIM used in the DCTG-GP system is in Table 2. This grammar is used during random GP tree generation and reproduction operations, and ensures the grammatical integrity of all GP expressions evolved during a run. The grammar shown is a simplification of the actual definition file used, with implementation details omitted. The rules for site, species and float are not shown, as they simply return legal values from user-supplied lists or ranges. Note that the DCTG-GP definition does not require the literal syntactic definition of PIM from Table 1. Rather, the CFG represents the argument structures of sentences, which will be represented as branches in the grammatical GP tree. For example, in rule 3c, the relevant information for a transformation sentence is the two species types in the transform, and the optional rate value. In the semantic portion of the DCTG (not included), the actual values for labels and operator types are determined and stored in the GP tree. When a PIM model is to be interpreted for fitness evaluation, internal data structures used in the rules are rewritten when required in the actual PIM syntax, as used by an external PIM translator application.

A few aspects of the grammar in Table 2 should be clarified. Firstly, one result of the correctness conditions in Section 3 is that a dissociation sentence is only possible if its association sentence counterpart has been defined. Therefore, the *detach* reference at the end of rule 3a simplifies the construction of sensible models. When *detach* is invoked in the grammar during tree generation, either detachment (dissociation) will not occur (4a) or will occur (4b). In the latter case, a new PIM sentence will be constructed, with the same site and species pairs in the original rule 3a, but with possibly new conditions. Similar reasoning occurs with dephosphoration in rule 3b. Secondly, the implementation of rules 3a and 3b ensure that conditions use species referred to in

the body of the sentences (see discussion below). Similarly, rule 3c ensures that a transformation is constructive, by using two different species in its arguments. Lastly, there is a maximum of one positive and negative condition in a sentence. Although multiple conditions are easy to define, the experiments in Section 6 use single conditions only. Multiple conditions also increase the time for correctness testing, which is discussed next.

The grammar defines grammatically correct PIM models. Nevertheless, these models might not respect the correctness conditions in Section 3. A few grammar rules (3a and 3b) ensure correct species references in conditions, which addresses condition 1. However, other errors can arise, and incorrect models will be uninterpretable. There are a number of possible strategies for dealing with this:

1. *Evolution*: The fitness function can penalize erroneous models.
2. *Grammar*: Incorporate the correctness conditions into the grammar.
3. *Expression correction*: Correct errors before evaluating models.

Using evolution is the simplest approach, as it requires little effort to implement. Of course, this may be hugely detrimental to evolutionary effectiveness, since a majority of models may be erroneous. Almost the entire population would have equally terrible fitness scores, and effective search would be impossible. Defining a more sophisticated grammar seems appealing, especially since the correctness conditions are syntactically detectable. Unfortunately, this is very difficult to implement with the context-free grammar used in DCTG-GP. It would require a large number of state values to be passed between grammar rules, and the resulting complexity of the grammar would be prohibitive.

Expression correction is the most practical solution to the correctness issue. Expression correction is applied to the initial PIM model generated from a GP tree, to correct violations of the correctness conditions. This is done immediately before model interpretation during fitness evaluation. The corrections are not saved back into the tree, nor retained by the GP system. Because the correction procedures are deterministic, however, an erroneous expression will always have the identical corrections applied to it. Deterministic correction is important for evolution, because it provides stable chromosomes for search. The correction procedures also apply some expression editing that is outside the scope of the correctness conditions, in order to improve the likelihood of suitable models being generated by GP.

A number of parameters are defined by the user, to help specify the desired size and complexity of the evolved PIM model. The user supplies a list of species labels (S), and a list of site labels (L). The assumption is that any desired model evolved by GP should contain all the species in S and sites in L. Thus these lists should be the minimum-sized sets of labels required, assuming this is known. An optional minimum connectivity list (C) can be supplied. It indicates the minimum number of distinct sites that should ideally be used by each species. If C is not given, correction steps using it will be skipped. Note that the connectivity in generated models may not adhere to those in C, and may use fewer species and sites than in S and L. However, models will never use more species and sites than are given in these lists.

The following expression corrections are applied to a model, in the order given:

1. *Correct missing sites*: Each species is inspected. If its minimum connectivity specified in C has not been met, then duplicate site references within it are replaced with missing sites.

2. *Correct bad conditions*: For each condition in a sentence, its site should be one that is paired with the species in another sentence. If not, it is corrected to be so. If the correction cannot be done, that condition is removed. (Note that the grammar ensures the correctness of species in conditions.)

3. *Reduce overlapping conditions*: If one sentence S_1 has conditions that are a subset of those from another sentence S_2, then remove sentence S_1. Should the deleted S_1 have a rate specified, and S_2 not have a rate, then S_2 inherits the rate from S_1.

After application of these steps, conditions 1 through 6 in Section 3 are respected by the model. Condition 7, however, is only partially addressed by the overlapping condition reduction (item 3). Overlap reduction removes the majority of overlaps seen in the experiments, especially since single terms in conditions are used. Although a complete overlap reduction algorithm is possible, it would be expensive to implement for the little gain obtained. Therefore, when a model with overlapping conditions is not corrected by the overlap reduction step, fitness evaluation will penalize it.

5 Model evaluation

5.1 Statistical feature evaluation

PIM models exhibit stochastic behaviours, and a model may generate different time series behaviors during each simulation. In fact, some models can produce very chaotic, noisy behaviors. Consequently, standard techniques such as measuring the sum of errors between target and candidate time series are unlikely to be effective.

A more effective strategy is to use statistical features to characterize time series behaviors (Imada, 2009; Imada and Ross, 2010). A suite of different statistical feature tests taken from the literature were implemented (Nanopoulos et al., 2001; Wang et al., 2006). However, the use of too many feature tests is not advisable. Extraneous features will result in an intractably large search space. Some features impart little germane information about a desired behavior, and their inclusion will act as distractive decoys to the search. Note that feature selection for time series is a non-trivial and open research problem in data mining and time series analysis (Liu and Motoda, 2007). A solution to the problem would be of significant interest to stock market investors.

After studying the feature characteristics for the target models used in this paper, the model of interest were adequately characterized by a small set of basic feature tests: average, standard deviation, and skew. Average and standard deviation are well-known. Skew is the measure of asymmetry of a distribution of data values. A negative skew means the left tail is longer; a zero skew means the distribution is symmetric. Examining the target plots in Figure 2, these three features were determined to be adequate for describing the shapes of the curves. Although other feature tests may perhaps be more appropriate, it is not a goal here to determine the optimal set of features to use.

The next issue to consider is the determination of which time-series plots from a PIM model simulation are actually necessary for fitness evaluation. As discussed in Section 3, a species having k sites results in 2^k separate time-series curves. Considering that each curve is also denoted by multiple feature tests, a combinatorial increase in evaluation criteria can result. The selection of particular time-series curves to use for an experiment is subjective and problem-specific, and depends on the complexity of the model being considered, and the behavior of the curve in question. As a rule of thumb, it is important to evaluate curves that map to the complexity of the model, so that GP will evolve a model with adequate numbers of species and sites. For example,

if 3 sites are required in the model, then selecting a few curves that denote combinations of these three sites is sensible. Fewer curves are preferable as well, to keep the search space tractable.

Some PIM models may be erroneous (overlapping conditions), and therefore uninterpretable. Since they have no time-series plots to evaluate, they are assigned a high penalty value. However, other PIM models may be correct and interpretable, but do not have the correct number of species, or level of site connectivity. Models that are too small will generate fewer time plots than the target model, and some specific plots to be evaluated might not exist. In such cases, a penalty will be assigned for each missing plot. On the other hand, when models happen to be larger than the target model, penalties will not be assigned, and we will rely on feature matching by GP to hone towards the desired solutions.

The method by which scores and penalties are assigned during fitness evaluation is the topic of the next section.

5.2 Fitness score assignment

A multi-objective approach to fitness evaluation is used here (Coello et al., 2007). The interpretation of a PIM model results in a number of time series curves, and each curve is then assigned some feature values that characterize it. The goal is to construct a PIM model whose set of feature values closely matches those of a target behavior. However, statistical features can vary widely in range and sensitivity. Time series curves can also differ substantially between each other. A multi-objective evaluation strategy prevents the need to manually reconcile the set of feature values with one another.

An approach to multi-objective scoring called *sum of weighted ranks* is used. It was originally proposed by Bentley and Wakefield (1997) for handling high-dimensional multi-objective problems. Consider a population of size N having members $M^1, ..., M^N$. Each M^j has a computed feature vector $< F_1^j, ..., F_K^j >$, where there are K features of interest. A target behavior is characterized by a feature vector $<T_1, ..., T_K>$. The absolute errors are first calculated for the population:

$$E^j = < |F_1^j - T_1|, \ ..., \ |F_K^j - T_K| > \qquad 1 \leq j \leq N$$

If any feature value is missing due to a missing time series plot, then a penalty value is inserted for that absolute error. Then the ranks of the error are determined for the population. A rank is a relative integer ordering of values within one objective, where the smallest magnitude value is assigned rank 1, the second is assigned rank 2, and so on. Tied magnitudes receive the same rank. This results in rank scores for each individual:

$$R^j = < \ r_1, \ ..., \ r_K \ >$$

where $1 \leq r_i \leq N$. The sum of ranks score S^j for each individual in the population is then:

$$S^j = \sum_{i=1}^{K} w_i r_i$$

where w_i is a user-supplied weight value. If the weights are omitted, then the default weight $w_i = 1$ is used. These S^j scores are then used as fitness values by evolution, where smaller values are preferred. The sum of ranks is equivalent to an *average* of ranks, except that a final scaling by the number of ranks is not performed. When a tournament selection is performed, as done here, both approaches are equivalent in their effect.

Fitness vector				Rank vector	Pareto Rank	Sum ranks Sum	Sum ranks Rank	Normalized Sum	Normalized Rank
(1,	9,	5,	4)	(1, 1, 2, 2)	1	6	1	0.63125	1
(2,	100,	4,	8)	(2, 2, 1, 3)	1	8	2	0.875	2
(10,	9,	9,	10)	(3, 1, 4, 4)	2	12	4	2.05	4
(16,	100,	8,	4)	(4, 2, 3, 2)	2	11	3	1.5	3
(16,	9,	500,	0)	(4, 1, 5, 1)	2	11	3	0.875	2
		max rank:		(4, 2, 5, 4)					

Table 3: Examples of rank scoring. No weights are used.

A possible option is to *normalize* the ranks before the summation step:

$$SN^j = \sum_{i=1}^{K} \frac{w_i r_i}{max_i}$$

where max_i is the maximum rank found for each objective. This can make contributions to the final score more uniform amongst the individual ranks.

An example of score calculations using sum of ranks is shown in Table 3. For comparison, the commonly used Pareto ranking is included (Goldberg, 1989). The "Rank" columns show the relative fitness ordering arising with each scoring scheme. In all cases, the final scores are not influenced directly by the raw scores in the fitness vector. Rather, the ranks are calculated from the relative ordering of scores between different fitness vectors. Using tournament selection, the actual final score values do not matter, and only their relative ordering (rank) is pertinent. Note that there are only two Pareto ranks for the set of fitness vectors. When more objectives are defined, most vectors will be undominated with Pareto ranking. This adversely affects selective pressure for evolution. In comparison, the plain and normalized sum of ranks create a more diverse range of scores. The normalized ranks also differ from the plain version in terms of overall relative ordering. More dramatic differences between them can arise with larger populations.

6 Experiments

6.1 Target models

The five PIM models used as targets are in Table 4. Models 2 and 4 are taken from (Kahramanogullari and Cardelli, 2009), which model Fcγ receptor phosphorylation during phagocytosis. Models 1 and 3 are modifications of 2, and model 5 is a modification of 4. The intention is for model 1 to be the simplest base model, with models 2 through 5 showing graduated increases in complexity. The modifications of models 2 and 4 are not intended to be biologically motivated.

A very simplified explanation of the biochemical reaction being modeled is as follows. Phagocytosis is a cellular process by which particles are engulfed by a cell membrane to form a phagosome, or internal cellular compartment (Wikipedia, 2010). There are 3 basic steps in the reaction (Figure 1), in which (i) an unbound particle (ii) connects or binds to a phagocyte surface, which results in a (iii) triggering of the phagocytosis. It is these binding events that are described in the PIM models. In model 2 in Table 4,

Model 1:	site f on FcR associates site i on IgG site y on FcR gets phosphorylated if site f on FcR is bound site z on FcR gets phosphorylated if site f on FcR is bound
2, 3:	site f on FcR associates site i on IgG with rate 2.0 site y on FcR gets phosphorylated {**with rate 4.0**} if site f on FcR is bound site z on FcR gets phosphorylated if site f on FcR is bound
4, 5:	site f on FcR associates site i on IgG with rate 2.0 site y on FcR gets phosphorylated {**with rate 4.0**} if site f on FcR is bound site z on FcR gets phosphorylated if site f on FcR is bound site s on FcR associates site sr on Src if site f on FcR is bound site s on FcR dissociates site sr on Src

Table 4: Target PIM models. Models 3 and 5 include the boldface rate term.

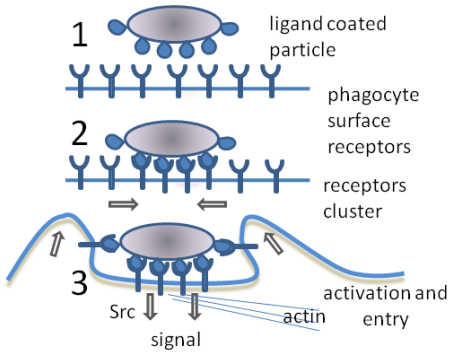
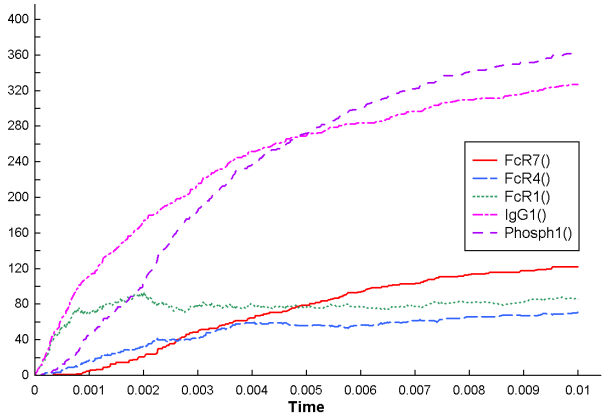


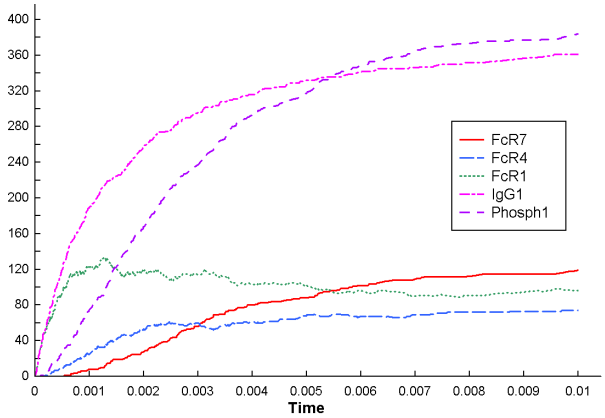
Figure 1: Phagocytosis in three steps. (i) Unbound phagocyte surface receptors. (ii) Binding of receptors. (iii) Phagocytosis is triggered and the particle is surrounded. (Illustration by GrahamColm at en.wikipedia.org, redistributed under Creative Commons license).

the first sentence describes step (ii) in Figure 1, while sentences 2 and 3 describe step (iii). A key point in the PIM model is the identification of particular sites in the species, and whether they are in bound or unbound states. The reaction is further refined in model 4, by showing an intermediate binding step that occurs with phosphates before the actual phagocytosis. Again, the state of the sites on the species participating in the reaction is critical.

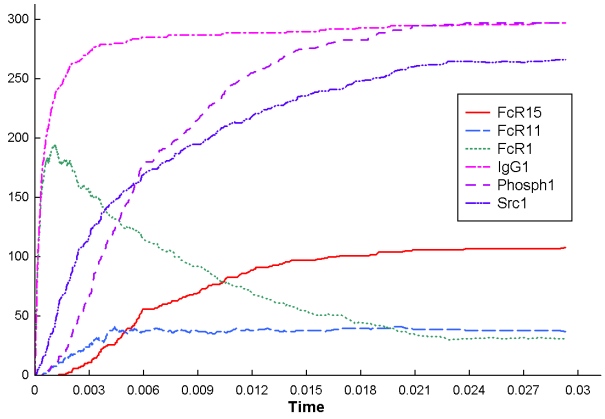
Examining Table 4, model 1 is the simplest example, and involves species FcR, IgG, and Phosph. (Phosph arises with phosphorylation statements). A maximum of 3 sites are used. An example plot is given in Figure 2, and shows the 5 curves to be measured during fitness evaluation. Although there are a maximum of 14 curves possible for model 1, three of them were null, and two were duplicates. As discussed in Section 3, the generated curves map to combinations of simultaneously bound sites. The FcR7 curve represents the binding of sites f, y, and z. Therefore, selecting this curve for evaluation helps establish the complexity of species FcR – it requires 3 bound sites in



Model 1



Model 2



Model 4

Figure 2: Plots for models 1, 2, and 4. Plots are from single simulations.

order for the FcR7 curve to be defined. FcR4 denotes the binding of sites f and y, and FcR1 is the lone binding of f. IgG1 and Phosph1 represent those species with their single sites bound. Since no rates are included, the default rate of 1.0 is used throughout. Models 2 and 3 are identical in structure to model 1, except that they include additional rate terms. An example plot for model 2 is also in Figure 2. The inclusion of the single rate term causes a noticeable change to most of the curves, when compared to those of model 1.

Model 4 is a more expressive and complex model. With the addition of Src, it uses a total of 4 species. FcR now has 4 sites, which means that a maximum of 16 curves are possible for FcR alone. As before, the FcR curves are selected to establish its inherent complexity. For example, FcR15 denotes the simultaneous binding of all 4 of its sites, while FcR11 represents the binding combination of f, s, and y. The other curves are similar to those in the simpler models above. Model 5 is like model 4, except for its second rate term (in boldface).

To run the models in a SPIM simulator, models 1, 2, and 3 are run with 400 copies of FcR, IgG, and Phosph, while models 4 and 5 use 200 copies of these and the Src species.

Prior to using GP, each target model was simulated 1000 times, and mean statistical features were calculated (Table 5). Standard deviations for these values were also computed (not shown), to be used during z-score analyses.

6.2 Genetic programming parameters

Table 6 lists the genetic programming parameters used in the experiments. Only the parameters differing from experiment 1 are shown for experiments 2 through 5. Although most parameters are well known in the GP literature (eg. (Koza, 1992)), a few require explanation. The initial random population is overpopulated. A total of 4000 random trees are generated, and the elite 1500 are culled from it. Koza's ramped tree generation is used to create the initial population. Full trees are generated at a rate of 0.95, and grow at a rate of 0.05. Grow trees tend to produce unfeasibly small PIM models, and so they are not generated often. A unique population option is used. This ensures that all the GP trees in the population are unique throughout the run. This does *not* guarantee that translated PIM models are unique, however. The elite 5 individuals in every generation are migrated to the next generation, after which they are re-evaluated and assigned new feature vectors. A large penalty value is used for missing time sequences, due to model size mismatches or PIM correctness issues. This ensures that such sequences will cause their respective ranks to have high enumerations (lower fitness in the rank). The PIM-specific parameters are discussed in Section 4. Note that although the IgG connectivity of 2 in model 1 is overestimated, it has neglible effects on performance.

7 Results

The evaluation of stochastic models from multi-objective evolutionary algorithms is a tricky task. Unless criteria is established to evaluate quality, it is difficult and time-consuming to evaluate the multitudes of solutions generated. The problem is compounded with stochastic systems such as PIM models: stochastic models generate varying behaviors during different interpretations. Even a perfect solution expression may occasionally exhibit behavior deviating outside the norm.

Given these challenges, experimental results were evaluated as follows. From each GP trial, 25 of the top scoring solutions are saved. (These expressions have low normal-

Model	Curve	Avg	Std Dev	Skew
1	FcR1	74.9	13.1	-2.76
	FcR4	47.6	17.8	-1.37
	FcR7	68.6	40.7	-0.38
	IgG1	238.7	78.6	-1.19
	Phosph1	232.4	114.2	-0.66
2	FcR1	96.6	16.5	-1.73
	FcR4	60.3	20.2	-1.66
	FcR7	72.6	38.7	-0.61
	IgG1	290.0	76.3	-1.72
	Phosph1	265.8	114.9	-0.94
3	FcR1	64.4	9.45	-2.19
	FcR4	126.3	36.6	-2.0
	FcR7	84.2	36.4	-1.09
	IgG1	289.3	76.5	-1.71
	Phosph1	309.2	110.3	-1.41
4	FcR1	50.5	17.2	0.26
	FcR11	19.6	8.39	-1.12
	FcR15	38.6	23.4	-0.42
	IgG1	157.2	35.9	-2.00
	Phosph1	117.0	62.5	-0.60
	Src1	106.2	44.7	-0.93
5	FcR1	50.4	17.2	0.27
	FcR11	39.4	15.3	-1.29
	FcR15	45.3	23.7	-0.74
	IgG1	157.2	35.8	-2.10
	Phosph1	134.5	63.7	-0.86
	Src1	106.2	44.6	-0.94

Table 5: Feature values for models. All values are the averages from 1000 interpretations of each model.

ized sum of ranks scores.) This yields a total of 750 individuals from the 30 trials for a single experiment. All the solution trees are converted, with correctness editing, into PIM expressions. Duplicate PIM models are removed. (Note that renaming of site labels within expressions can cause them to avoid deletion, even though they are behaviorally identical.) The remaining "unique" PIM models are then evaluated. Each model is interpreted 100 times, and its mean feature scores are calculated. These scores are then compared to the target features, using a statistical significance analysis. The z-scores between the solution and target feature scores are computed. When a feature score between the solution and target match at a 95% level of significance, a "match" is recorded. The result is a vector of match indicators, one per objective, which tell whether each feature value matches the target at a 95% significance level.

The above z-score analysis should be considered a heuristic aid for identifying potential high-quality solutions, rather than a fool-proof, absolute means for automatically ascertaining solution quality. Firstly, the z-score calculation requires an adequate number of iterations to be undertaken for calculation of average feature scores, both for solutions and the target models. The number of iterations required depends upon the

	Model		
Parameter	1	2, 3	4, 5
Trials	30		
Solns per trial	25		
Generations	25	40	60
Init. pop. size	4000		
Population size	1500		
Rate full, grow	0.95, 0.05		
Rate cross, mut	0.75, 0.25		
Rate term. mutation	0.75		
Ramped tree depth range	4-6		
Reprod. tree depth	6	7	8
Tournament size	3		
Unique population	yes		
Elitism	5		
Float range	$1.0 \leq f < 5.0$		
Fitness	sum ranks (norm.)		
Penalty	10^7		
Sites	3	3	4
Species/Connectivity	FcR/3, IgG/2	FcR/3, IgG/1	FcR/4, IgG/1, Src/1

Table 6: GP Parameters

expression being evaluated, and this is not easy to establish for arbitrary PIM models generated by the GP system. In addition, the use of z-scores implicitly assumes that the feature scores being analyzed exhibit a normal distribution. When either of the above assumptions do not hold, errors such as false negatives may arise. Therefore, we also perform a visual examination of solution expressions, to verify whether a solution is a syntactic match with the target model.

Tables 7 and 8 summarizes the z-score analysis of the quality of solutions for all the experiments. "Unique solutions" denotes the number of models remaining after duplicate PIM expressions are deleted. "Objective hits" are the numbers of expressions having k separate feature hits (similarity with corresponding target feature at 95% significance level). The percentages are included in parentheses. Since duplicate expressions between runs are eliminated, the objective hits values do not account for the number of times in which separate trials produce the identical solutions. Therefore, the "Runs with hits" column is included, to show the number of times separate trials evolved the target model expression. This was determined by visually inspecting the 25 solutions generated in each of the 30 trials per experiment. This task was aided by only considering solutions that have the same number of sentences as the target model in question. Although this aids manual inspection significantly by reducing the number of expressions to view, it does mean that potentially viable solutions with "intron" (non-functional) sentences will not be considered.

The experiments for models 1, 2 and 3 show that good solutions are plentiful, and exact hits with the target model are common. Model 1 was fairly trivial for GP to find, and was often seen in early generations. The solutions for model 2 do not perform as well as for model 1, as the inclusion of its rate term complicates the search space.

Model	Unique solutions (max 750)	Objective hits (%) 0	1-5	6-10	11-14	15	Trials with hits (max 30)
1	248	8 (3.2)	23 (9.3)	47 (19.0)	122 (49.2)	49 (19.8)	30
2	714	1 (0.1)	101 (14.1)	373 (52.2)	219 (30.7)	20 (2.8)	25
3	730	5 (0.7)	363 (49.7)	314 (43.0)	46 (6.3)	2 (0.3)	20

Table 7: Solution quality for models 1, 2 and 3. Hits refer to number of objectives matching target objective at 95% significance level, after interpreting solution 100 times and target 1000 times. Total of 15 objectives.

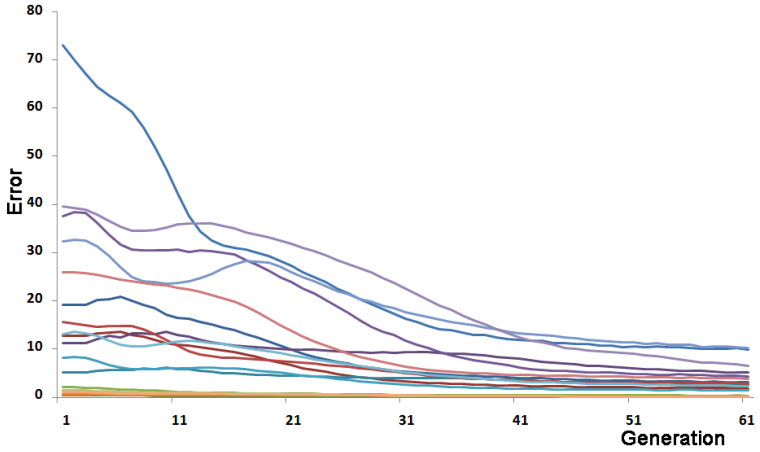
Model	Unique solutions (max 750)	Objective hits (%) 0	1-6	7-12	13-17	18	Runs with hits (max 30)
4	714	4 (0.6)	204 (28.6)	402 (56.3)	104 (14.6)	0 (0.0)	7
5	734	4 (0.5)	357 (48.6)	343 (46.7)	30 (4.1)	0 (0.0)	4

Table 8: Solution quality for models 4 and 5. Total of 18 objectives.

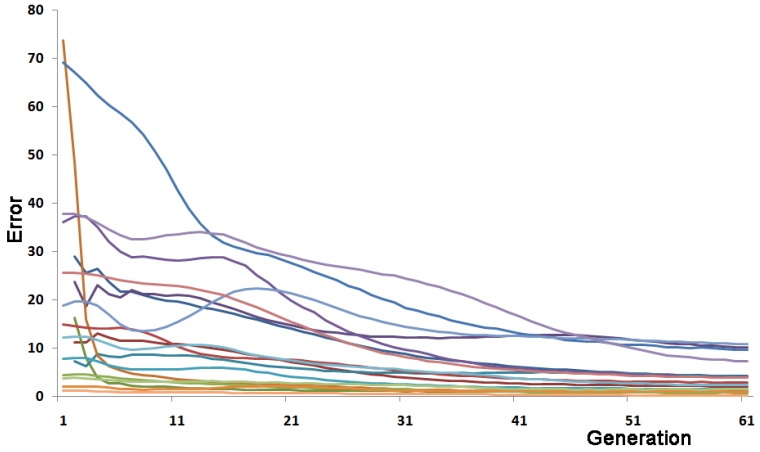
Model 3's additional rate term again results in a more challenging search. Exact feature score hits were not as common. An inspection of the solution expressions, however, showed that many solutions were virtually identical to the target expression, except for expected variations in rate values. For example, in the 30 trials for model 3, there were 57 solution expressions that matched the target model, except that the second rate term used a value f in the range $3.5 \leq f \leq 4.1$. The sensitivity of the stochastic interpreter means that a rate of 3.6 (rather than the target rate of 4.0) may result in feature values that are not considered hits with the target model at a 95% significance level, using our hit analysis scheme.

The results for models 4 and 5 dramatically reflect their complexity. Furthermore, the z-score analysis met some difficulty when analyzing one of the behavior curves. False negatives arose from the z-scores when analyzing the IgG1 skew feature. In fact, even the target model did *not* show a match with itself with respect to this feature. It is likely that this feature is either not normally distributed, or requires many more iterations. Therefore, the number of hits under the 18 column is zero, even though the exact expressions for models 4 and 5 were found multiple times.

Figure 3 shows the average population performance of the runs for models 4 and 5. The curves denote the average error in the population per generation, for each of the 18 objectives, averaged over all 30 trials. The errors have largely converged in the population by the 50th generation. One exception is the curve for "Src average",



Model 4



Model 5

Figure 3: Performance plots for models 4 and 5. Average error per generation, for each objective (18 total), averaged over 30 trials.

	FcR7			FcR4			FcR1			IgG1			Phosph1		
	avg	sd	sk	avg	sd	sk	avg	sd	sk	avg	sd	sk	avg	sd	sk
1	65	66	77	69	65	68	75	77	87	88	88	86	71	72	75
2	83	75	70	70	83	80	35	70	76	20	54	30	35	80	41
3	67	38	41	21	36	59	11	47	68	14	48	32	26	42	32

Table 9: Models 1, 2 and 3 hit rates for features (%)

	FcR15			FcR11			FcR1			IgG1			Phosph1		
	avg	sd	sk	avg	sd	sk	avg	sd	sk	avg	sd	sk	avg	sd	sk
4	58	55	60	65	73	76	44	41	62	13	30	13	39	67	44
5	39	30	27	12	20	55	52	49	71	19	38	28	16	57	22

Table 10: Models 4 and 5 hit rates for features (%)

which still shows a steady improvement during the last 10 generations in both plots. Note that it is not useful to plot the errors of the population's best performer, as there usually exists an individual that shows superlative performance in a feature. Such models, however, are rarely superlative in multiple features simultaneously – at least, until later generations.

Different statistical features exhibit varying degrees of difficulty with respect to evolvability. Consider the individual feature hit rates for the models in Tables 9 and 10. These rates are the hits (95% significance) found for all solutions for each experiment. The tables show a general reduction in hit rates as models increase in complexity. Model 1 is fairly successful in all features, while lower success can be seen in models 2 and 3. The IgG1 features become especially challenging to realize in models 2 and 3. Similar trends can be seen with models 4 and 5.

It is interesting to consider evolved solutions that might be viable alternatives to the target models in Table 4. Two alternative models for model 5 are shown in Table 11. The syntactic differences (besides rate variations) are shown in boldface. Although the biological feasibility of these models will not be considered here, it is clear that they have many expression similarities with the target model. In model A1, an extra dephosphorylation sentence is included. This sentence may be an instance of "genetic programming bloat". Upon examining the rest of A1, site f on FcR is unbound during the early stage of the simulation. After that time, site f in FcR will remain bound to IgG, as there is no mechanism to unbind it. Therefore, this particular sentence may have no impact on behavior after the initial portion of the simulation. This is seen in the z-score hit vector for A1, in which only one feature (skew of FcR1) does not match the target.

The alternative model A2 in Table 11 also has a few minor expression differences with the target model. It showed feature hits on all but 3 features. One expression difference is that a dephosphorylation replaces model 5's dissociation. This was commonly seen throughout the experiments, and shows the similarity of dissociation and dephosphorylation expressions.

Trials took anywhere from wall-clock times of 2.5 hours (model 1) to 5 hours (models 4, 5) per trial. GP runs were done under Linux, on time-shared 3.2 GHz AMD Phe-

A1:	site f on FcR associates site i on IgG with rate 2.01 site y on FcR gets phosphorylated with rate 3.79 if site s on FcR is bound site z on FcR gets phosphorylated if site s on FcR is bound **site z on FcR gets dephosphorylated with rate 4.93 if site f on FcR is unbound** site s on FcR associates site sr on Src if site f on FcR is bound site s on FcR dissociates site sr on Src **if site y on FcR is bound**
A2:	site f on FcR associates site i on IgG with rate 1.99 **if site s on FcR is unbound** site y on FcR gets phosphorylated with rate 3.78 if site s on FcR is bound site z on FcR gets phosphorylated if site s on FcR is bound site s on FcR associates site sr on Src if site f on FcR is bound site **y** on FcR **gets dephosphorylated with rate 2.18 if site z** on FcR is bound

Table 11: Two alternate solutions for model 5 having high performance.

nom II quad cores with 4GB RAM, and 2.66 GHz Intel i7 quad cores with 6 GB RAM. This is respectable performance, given that most of the system software is running in interpreted environments (DCTG-GP is running in Sicstus Prolog (SICS, 2010), and PIM and SPiM are running in OCAML (INRIA, 2010)). The interpretation of PIM models and subsequent analysis of time series can be computationally intensive.

8 Discussion

The results are encouraging. All target models were reconstructed, and high-performing alternative models arose. The solution hit counts (final column) in Tables 7 and 8 are conservative, in that we did not account for generated target models with intron code, nor determine specific alternative models that were behaviorally equivalent to the targets.

Although results were positive, there are many aspects of the experimental design that could be improved. The feature hit rates in Tables 7 and 8 suggest that a problem scale ceiling is quickly being reached. Scalability can definitely be improved with feature space reduction, and a more strategic selection of statistical features and target curves. It is not our goal to determine the most optimal and effective behavioral features of a target model. The initial random population contained many non-productive models, even after model correction. If the population were to be of a higher quality, better overall performance would arise during evolution. More sophisticated grammar definitions and correction steps could help, for example, by enforcing minimum species counts in models. When multiple rates are required in a solution, one aspect of search becomes that of numerical parameter optimization. Although GP theoretically performs this during evolutionary, the optimization of numeric values benefits with specialized numeric reproduction operators, for example, mutation with Gaussian perturbation. A separate local search phase, perhaps at the end of a run, can also fine-tune numeric values.

This research is the latest in a series of investigations in stochastic model synthesis with GP. GP has been used to evolve models in the stochastic pi-calculus (Ross, 2007; Ross and Imada, 2009a,b; Ross, 2010), which is the underlying language on which PIM is built. A fundamental issue using a low-level process algebra like the stochastic pi-calculus is that it is inherently GP-unfriendly. Although compositionality is touted as being a primary strength of process algebras, they are only compositional with re-

spect to higher-level process definitions, and are decidedly *not* compositional at the level of algebraic sub-expressions in which processes are constructed. When used in genetic programming, models are highly brittle and are easily destroyed by reproduction operations such as crossover. This results in an extremely difficult search space to navigate, and poses a natural handicap to problem scalability. An algebraic modeling language akin to the pi-calculus was explored with GP by Leier et al. (2006). It is likely that similar scalability limitations will arise with their approach.

This research shows that GP is effective when higher-level modeling languages are considered. Similarly, GP was found to be very capable when used with a logic gate gene regulatory language (Imada, 2009; Imada and Ross, 2010). Like PIM, the underlying semantics of the logic gate language is rooted in the stochastic pi-calculus, and so stochastic simulations are possible. The logic gate language is also very compositional, which makes it amenable to refinement by GP, to an even greater degree than with PIM. It is difficult to construct arbitrary logic gate networks that are not productive and analyzable, while unproductive models are more frequent with PIM in early generations. As done here, Imada's work with logic gates characterizes time series with statistical features. A difference is that she combines multi-objective scores into a single-objective weighted sum using statistical weighting, based on the target model's statistical characteristics. This was shown to be effective in problems of up to 17 features – the same general problem size as in this paper. Although a comparative study of statistical weighting and summed rank scoring is required, it was found by Ross (2010) that pi-calculus networks were more successfully constructed with the sum of ranks approach.

Koza *et al.*'s use of GP to evolve metabolic networks shares similarities with this research (Koza et al., 2000, 2003). They construct metabolic networks by obtaining time series data from the simulator, and matching the data with target time series. One major difference is that their models are deterministic: simulations started in the same conditions will always produce the same deterministic output. This is a great benefit for fitness evaluation, as the time series plots can be directly compared to each other using error fitting. This is not feasible with the noisy, stochastic behaviors considered here.

A popular means for representing gene regulatory network models is with sets of differential equations. Many papers can be found that evolve such models with genetic programming (Sakamoto and Iba, 2001; Ando et al., 2002; Streichert et al., 2004; Cho et al., 2006; Floares, 2008; Qian et al., 2008). It is beyond the goals of this paper to make a qualitative comparison between differential equations and algebraic PIM models. It can be argued that PIM models are human-oriented, and hence may be more instructive to biologists attempting to understand the mechanics of a biological network model. However, more detailed comparisons of these styles of biological models is necessary.

9 Conclusion

The use of GP as a tool for constructing high-level network models of biological phenomena is promising. As more sophisticated bio-network simulation languages are developed, their use as target languages for GP should be considered. We posit that the evolution of complex bio-network phenomena will depend upon the use of higher-level biological simulation languages, such as PIM. Low-level process algebra will unlikely permit a great deal of scalability beyond the modest sized systems currently handled in (Ross and Imada, 2009b). PIM models are translated by the PIM compiler into lower-level stochastic pi-calculus expressions, which are in turn interpreted by the SPIM in-

terpreter. It is important to realize that these intermediate pi-calculus translations are usually very complex and large, and definitely exceed the size and complexity of pi-calculus systems previously generated with GP. This was also seen in Imada and Ross (2010) with the logic-gate language.

Work is continuing on using GP with other bio-simulation languages (Danos et al., 2007; Guerriero et al., 2007; Dematte et al., 2008).

Means for simplifying the search space using feature reduction strategies and improved statistical features are also being considered. Simpler and more well-behaved feature spaces will allow more complex biological models to be handled.

Acknowledgements: Thanks to Ozan Kahramanoğullari for generously sharing his PIM software, and for his helpful advice; to Cale Fairchild, for help with system issues; and to Janine Imada, for the use of her well-written statistical feature analysis code.

References

Ando, S., Sakamoto, E., and Iba, H. (2002). Evolutionary modeling and inference of gene network. *Inf. Sci*, 145(3-4):237–259.

Angeline, P. (1998). Evolving Predictors for Chaotic Time Series. In *Proc. SPIE: Application and Science of Computational Intelligence*, volume 3390, pages 170–180.

Banzhaf, W. (2003). Artificial Regulatory Networks and Genetic Programming. In Riolo, R. and Worzel, B., editors, *Genetic Programming Theory and Practice*, pages 43–61. Kluwer.

Bentley, P. and Wakefield, J. (1997). Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *Soft Computing in Engineering Design and Manufacturing*. Springer Verlag.

BioSPI (2010). The biospi project. URL: http://www.wisdom.weizmann.ac.il/ biospi/. Last accessed July 9, 2010.

Blossey, R., Cardelli, L., and Phillips, A. (2006). A Compositional Approach to the Stochastic Dynamics of Gene Networks. *Trans. in Comp. Sys. Bio (TCSB)*, 3939:99–122.

Borrelli, A., De Falco, I., Della Cioppa, A., Nicodemi, M., and Trautteura, G. (2006). Performance of genetic programming to extract the trend in noisy data series. *Physica A: Statistical and Theoretical Physics*, 370(1):104–108.

Bower, J. and Bolouri, H. (2001). *Computational Modeling of Genetic and Biochemical Networks*. MIT Press Kaufmann.

Castellini, A. and Manca, V. (2009). Learning Regulation Functions of Metabolic Systems by Artificial Neural Networks. In *Proc. GECCO 2009*, pages 193–200. ACM Press.

Cho, D.-Y., Cho, K.-H., and Zhang, B.-T. (2006). Identification of biochemical networks by S-tree based genetic programming. *Bioinformatics*, 22(13):1631–1640.

Chou, I.-C. and Voit, E. (2009). Recent developments in parameter estimation and structure identification of biochemical and genomic systems. *Mathematical Biosciences*, 219:57–83.

Chu, D. (2007). Evolving genetic regulatory networks for systems biology. In Srinivasan, D. and Wang, L., editors, *Proc. CEC 2007*, pages 875–882, Singapore. IEEE Press.

Clocksin, W. and Mellish, C. (1994). *Programming in Prolog (4th ed)*. Springer-Verlag.

Coello, C. C., Lamont, G., and Veldhuizen, D. V. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, 2 edition.

Danos, V., Feret, J., Fontana, W., Harmer, R., and Krivine, J. (2007). Rule-Based Modelling of Cellular Signalling. In Caires, L. and Vasconcelos, V., editors, *CONCUR 2007*, pages 17–41. Springer-Verlag LNCS 4703.

Dematte, L., Priami, C., and Romanel, A. (2008). The BlenX Language: A Tutorial. In Bernardo, M., Degano, P., and Zavattaro, G., editors, *SFM 2008*, pages 313–365. Springer-Verlag LNCS 5016.

Drennan, B. and Beer, R. (2006). Evolution of repressilators using a biologically-motivated model of gene expression. In et al., L. R., editor, *Artificial Life X: Proc. Tenth Intl. Conf. on the Simulation and Synthesis of Living Systems*, pages 22–27. MIT Press.

Fisher, J. and Henzinger, T. (2007). Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249.

Floares, A. (2008). Automatic Inferring Drug Gene Regulatory Networks with Missing Information Using Neural Networks and Genetic Programming. In Wang, J., editor, *WCCI 2008*, pages 3078–3085. IEEE.

Gillespie, D. (1977). Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem*, 81:2340–2361.

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

Guerriero, M., Heath, J., and Priami, C. (2007). An Automated Translation from a Narrative Language for Biological Modelling into Process Algebra. In Calder, M. and Gilmore, S., editors, *CMSB 2007*, pages 136–151. Springer-Verlag LNCS 4695.

Hecker, M., Lambeck, S., Toepfer, S., van Someren, E., and Guthke, R. (2009). Gene regulatory network inference: Data integration in dynamic models - A review. *Biosystems*, 96(1):86–103.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice–Hall.

Imada, J. (2009). Evolutionary synthesis of stochastic gene network models using feature-based search spaces. Master's thesis, Department of Computer Science, Brock University.

Imada, J. and Ross, B. (2010). Evolutionary synthesis of stochastic gene network models using feature-based search spaces. *New Generation Computing*. In press.

INRIA (2010). OCAML. Last accessed July 1, 2010.

Kahramanogullari, O. (2010). Pim - spim. http://www.doc.ic.ac.uk/ ozank/pim.html. Last accessed June 18, 2010.

Kahramanogullari, O. and Cardelli, L. (2009). An Intuitive Automated Modelling Interface for Systems Biology. In *DCM'09*, pages 1–18.

Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., and Tomita, M. (2003). Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics*, 19(5):643–650.

Kitagawa, J. and Iba, H. (2003). Identifying Metabolic Pathways and Gene Regulation Networks with Evolutionary Algorithms. In Fogel, G. and Corne, D., editors, *Evolutionary Computation in Bioinformatics*, pages 255–278. Morgan Kaufmann.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Koza, J., Keane, M., Streeter, M., Mydlowec, W., Yu, J., and Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.

Koza, J., Mydlowec, W., Lanza, G., Yu, J., and Keane, M. (2000). Reverse Engineering and Automatic Synthesis of Metabolic Pathways from Observed Data using Genetic Programming. Technical Report SMI-2000-0851, Stanford Medical Informatics.

Lee, W.-P. and Yang, K.-C. (2008). Applying Intelligence Computing Techniques to Modeling Biologial Networks from Expression Data. *Genomics, Proteomics and Bioinformatics*, 6(2):111–120.

Leier, A., Kuo, P., Banzhaf, W., and Burrage, K. (2006). Evolving noisy oscillatory dynamics in genetic regulatory networks. In *et al.*, P. C., editor, *EuroGP 2006*, volume 3905 of *LNCS*, pages 290–299. Springer.

Linden, R. and Bhaya, A. (2007). Evolving fuzzy rules to model gene expression. *Biosystems*, 88(1-2):76–91.

Liu, H. and Motoda, H. (2007). *Computational Methods of Feature Selection*. Chapman and Hall/CRC.

Markowetz, F. (2005). A bibliography on learning causal networks of gene interactions. http://www.molgen.mpg.de/~ markowet/docs/network-bib.pdf. Last accessed April 1, 2005.

Markowetz, F. and Spang, R. (2007). Inferring Cellular Networks - a Review. *MBC Bioinformatics*, 8:1–17.

Milner, R. (1989). *Communication and Concurrency*. Prentice Hall.

Nanopoulos, A., Alcock, R., and Manolopoulos, Y. (2001). Feature-based classification of time-series data. In *Information processing and technology*, pages 49–61. Nova Science Publishers, Inc., Commack, NY, USA.

Phillips, A. (2007). The stochastic pi machine (spim). URL: http://research.microsoft.com/ aphillip/spim/. Last accessed March 1, 2007.

Phillips, A. and Cardelli, L. (2004). A Correct Abstract Machine for the Stochastic Pi-calculus. In *Proc. Bioconcur'04*.

Priami, C. (1995). Stochastic pi-Calculus. *The Computer Journal*, 38(7):579–589.

Qian, L., Wang, H., and Dougherty, E. (2008). Inference of noisy nonlinear differential equation models for gene regulatory networks using genetic programming and kalman filtering. *IEEE Transactions on Signal Processing*, 56(7):3327–3339.

Raj, A. and van Oudenaarden, A. (2008). Nature, Nurture, or Chance: Stochastic Gene Expression and Its Consequences. *Cell*, 135:216–226.

Rodriguez-Vazquez, K. and Fleming, P. J. (2005). Evolution of mathematical models of chaotic systems based on multiobjective genetic programming. *Knowledge and Information Systems*, 8(2):235–256.

Ross, B. (2001). Logic-based Genetic Programming with Definite Clause Translation Grammars. *New Generation Computing*, 19(4):313–337.

Ross, B. (2007). Using Genetic Programming to Synthesize Monotonic Stochastic Processes. In *Proceedings CI-2007*.

Ross, B. (2010). Using Normalized Ranked Sums to Evolve Stochastic Networks. In preparation.

Ross, B. and Imada, J. (2009a). Evolving Stochastic Processes Using Feature Tests and Genetic Programming. In *Proc. GECCO 2009*.

Ross, B. and Imada, J. (2009b). Using Multi-objective Genetic Programming to Synthesize Stochastic Processes. In *Genetic Programming - Theory and Practice*.

Sakamoto, E. and Iba, H. (2001). Inferring a system of differential equations for a gene regulatory network by using genetic programming. In *Proc. CEC 2001*, pages 720–726. IEEE Press.

Schwaerzel, R. and Bylander, T. (2006). Predicting currency exchange rates by genetic programming with trigonometric functions and high-order statistics. In Cattolico, M., editor, *GECCO 2006*, pages 955–956. ACM.

SICS (2010). SICStus Prolog 4. Last accessed July 1, 2010.

Streichert, F., Planatscher, H., Spieth, C., Ulmer, H., and Zell, A. (2004). Comparing genetic programming and evolution strategies on inferring gene regulatory networks. In et al., K., editor, *GECCO-2004*, volume 3102 of *LNCS*, pages 471–480, Seattle, WA. Springer-Verlag.

Tkacik, G. and Bialek, W. (2009). Cell Biology: Networks, Regulation and Pathways. In Meyers, R., editor, *Encyclopedia of Complexity and Systems Science*. Springer-Verlag.

Wang, H., Qian, L., and Dougherty, E. (2007). Inference of Gene Regulatory Networks using S-System: A Unified Approach. In Volkert, G., editor, *CIBCB 07*, pages 82–89. IEEE.

Wang, X., Smith, K., and Hyndman, R. (2006). Characteristic-based clustering for time series data. *Data Min. Knowl. Discov.*, 13(3):335–364.

Wikipedia (2010). Phagocytosis. Last accessed July 2, 2010.

Zhang, W., Yang, G., and Z.Wu (2004). Genetic Programming-based Modeling on Chaotic Time Series. In *Proc. 3rd Intl Conf. on Machine Learning and Cybernetics*, pages 2347–2352. IEEE.