Brock University

Department of Computer Science

Evolution of Stochastic Bio-Networks Using Summed Rank Strategies

Brian J. Ross
Technical Report # CS-11-05
February 2011

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

Evolution of Stochastic Bio-Networks Using Summed Rank Strategies

Brian J. Ross
Dept. of Computer Science
Brock University
St. Catharines, Ontario
Canada L2S 3A1
Email: bross@brocku.ca

***Abstract*—Stochastic models defined in the stochastic pi-calculus are evolved using genetic programming. The interpretation of a stochastic model results in a set of time series behaviors. Each time series denotes changing quantities of components within the modeled system. The time series are described by their statistical features. This paper uses genetic programming to reverse engineer stochastic pi-calculus models. Given the statistical characteristics of the intended model behavior, genetic programming attempts to construct a model whose statistical features closely match those of the target process. The feature objectives comprising model behavior are evaluated using a multi-objective strategy. A contribution of this research is that, rather than use conventional Pareto ranking, a summed rank scoring strategy is used instead. Summed rank scoring was originally derived for high-dimensional search spaces. This paper shows that it is likewise effective for evaluating stochastic models with low- to moderate-sized search spaces. A number of stochastic models with oscillating behaviors were successfully evolved. Attempts on a larger-sized model were not successful. Reasons for its poor performance are likely due to poor choices in feature selection, and too many selected features and channels contributing to a overly difficult search space.**

I. Introduction

Systems biology uses the perspective that biological behaviors emerge from the complex interactions of separate components. Bio-network modeling is one application of systems biology, which involves modeling and simulating biological phenomena on the computer. In recent years, a variety of computational environments and formalisms have been used for simulating these phenomena [1]. Formal systems used for bio-network modeling include Petri nets [2], [3], [4], Bayesian networks [5], [1], P-systems [6], [7], cellular automata [8], [9] and ODE's [10]. All have their particular strengths and weaknesses. However, few comparative analyses of them exist in the literature.

Stochastic process algebras are another formalism for modeling and analysing biological and chemical processes [11], [12], [13]. One such process algebra is the stochastic pi-calculus (or SPI calculus) [14], [15]. It has been used to concisely representing stochastic models for a variety of biological and chemical systems.

Genetic programming (GP) has been used to automatically synthesize models written in the SPI calculus [16], [17]. Given a time series plot of a bio-chemical processes desired behavior, for example, a plot of quantities of proteins over time, GP attempts to evolve a SPI calculus model that produces a similar time-series behavior. The stochastic time series is characterized by a set of statistical feature scores [18], which allows behaviors with different degrees of stochastic noise to be handled. The work in [16] examined the problem using a single-objective fitness framework, while [17] took a multi-objective approach.

TABLE I
SPI-Calculus syntax

$$\begin{aligned} \text{P} &::= 0 \mid P\#P \mid Proc = P \mid \Sigma \\ \Sigma &::= \pi.P \mid delay(t).P \mid \Sigma + \Sigma \\ \pi &::= \text{in(c)} \mid \text{out(c)} \end{aligned}$$

This paper extends earlier research in [17], by considering an alternative means for multi-objective scoring. The ranked sum scoring multi-objective scoring strategy was proposed for use in high-dimensional optimization problems [19]. Recently, other research has shown its suitability for low-dimensional problems as well [20], [21]. Benefits of summed rank over traditional Pareto ranking are that outliers are discouraged, and an improvement over all objectives arises. This paper shows that the summed rank scoring strategy is similarly effective for SPI calculus modeling with GP. However, the appropriate selection of statistical features is critical for success.

Section II briefly reviews the stochastic pi-calculus process algebra. The multi-objective evaluation of stochastic processes is discussed in Section III. The grammatical GP system is described in Section IV. Section V presents some example experiments and results, which are discussed and compared to related work in Section VII. Some summary conclusions end the paper in Section VIII.

II. Stochastic pi-Calculus

The stochastic pi-calculus (SPI) is a process algebra used for the stochastic modeling of concurrent processes [22], [23]. Like the pi-calculus before it [24], SPI has the ability to denote concurrent communications, as well as mobility (dynamically changing networks). SPI is effective for modeling various complex behaviors as seen in chemical and biological systems [12], and a simulation environment is available [15].

A subset of the stochastic π-calculus is used here (Table I). We are primarily interested in the stochastic characteristics of the algebra, and we have no interest in mobility here. Therefore, the process algebra could be called *stochastic CCS* (based on an earlier formalism [25]). The operators used are as follows. A null process is *0*. Sequential ordering is done with ".". Concurrency (#) allows the concurrent interaction of expressions. Process definition (*Proc*=) defines a process that can be called via a "Proc" call (possibly with arguments). Choice (+) permits the stochastic selection of a term. A term is an input (*in(c)*) or output (*out(c)*) communication on a channel c. Each channel c has a quantitative rate assigned to it. A delay $delay(t)$ advances the clock by a stochastically-computed duration indicated by argument t. The notation *K@P* means *P#P#...#P* repeated K times.

The SPI calculus's main feature of interest is its stochastic semantics. The semantics use the Gillespie algorithm [26], a well-known stochastic algorithm used in chemical simulations. Consider the following:

$$(in(x).P_1 + \Sigma_1)|(out(x).P_2 + \Sigma_2)|P_3 \overset{rate(x)}{\rightarrow} P_1|P_2|P_3$$

Here, if the Gillespie algorithm selects $in(x)$ and $out(x)$, then a handshake arises between them. The expression transforms to that on the right-hand side. The quantities of channel x have been used up by this transition, and the availability of x in the overall expression has been reduced. On the other hand, new quantities of other channels (including x) may now be added via the resulting terms P_1 and P_2. When channels denote biochemical substances, such as proteins, enzymes, or other molecules, their quantities can be measured over time, resulting in quantitative time series for process behavior.

In the above, the Gillespie algorithm selects the execution of x stochastically, in a manner similar to that used by Roulette Wheel selection in evolutionary computation [27]. The probability of choosing a channel depends on the quantity of that channel available for communication, as well as the rate of that channel.

III. Stochastic model evaluation

A. Characterizing time series

TABLE II
Statistical feature tests used for time series analysis.

Test	Description
1. μ	Mean.
2. σ	Standard deviation.
3. Kurtosis	Degree of peakness or flatness relative to normal distribution.
4. Serial correlation (sc)	Degree of fit to a white noise model.
5. Chaos	Sensitivity dependence on initial values.
6. Teravirta	Degree of non-linearity.
7. Adjusted frequency	Measure of cyclic activity of possibly varying frequency.

The interpretation of SPI calculus models results in time series output – one time series sequence per channel. Each time series denotes the quantities of active instances for that channel during the course of the simulation. There is a wide variety of possible behaviors for these time series. Some models exhibit highly regular and predictable behaviors, while others result in time series with considerable stochastic noise and variability.

Time series analysis is a long studied discipline [28]. We characterize time series behaviors by sets of univariate statistical feature tests. The tests used here are taken from Imada [18], which were based on ones in [29], [30]. Although a suite of 17 tests are implemented in [18], selected combinations of those in Table II are used here. The tests are implemented in C and R (a statistical package) [31]. See [18] for more details on these tests and their implementation.

To determine the statistical feature characteristics of a target model, the target SPI model is interpreted 1000 times. The mean and standard deviations of the feature values over all the runs was computed. The mean scores are potential feature values to be used for the target model. As done by Imada [18], when examining the feature scores, the *stability* (z score) of each feature f is also calculated:

$$Stability_f = \frac{\mu_f}{\sigma_f}$$

Features having a high stability mean that they vary less, and may be a more accurate characterizations of the target behavior. On the other hand, some feature tests are naturally stable, and hence stability may not be a useful heuristic for them. Although stability analysis may help in selecting suitable features for a model, the overall decision procedure is *ad hoc*. In two experiments, a fortuitous selection of features proved to be effective. However, we were less fortunate in one experiment (Oregonator) in this regard.

B. Multi-objective scoring

The *weighted summed rank* strategy is a scoring strategy presented by Bentley and Wakefield [19]. It was originally proposed for high-dimensional multi-objective problems, where the more conventional Pareto ranking would fail [32]. However, it has since been shown to be effective for low-dimensional problems as well, with the added benefit of discouraging outlier solutions [20].

Consider an individual with a fitness vector $< f_1, ...f_k >$. The objective scores for the individuals in the population is ranked separately, resulting in a rank vector $< r_1, ..., r_k >$ for each individual. Each r_i is the ordered rank of f_i with respect to the rest of the population. For example, the individual with the best f_i score for some objective will have $r_i = 1$ for that objective. Once the fitness ranks are determined, an individual's fitness is computed as:

$$fitness = \sum_{i}^{k} w_i r_i.$$

Lower scores are preferrable. The w_i term is the weight for objective i (default 1). A higher w_i increases the priority of objective i in the score. Note that *fitness* is equivalent to the

average (weighted) rank for the individual. Also note that this strategy converts the problem into a single-objective score. This is done via measuring relative independent performance on each objective, rather than via a combination of absolute scores as done by weighted sums.

A variation of the above is the *normalized summed rank*:

$$fitness = \sum_{i}^{k} w_i \frac{r_i}{R_i}$$

where R_i is the maximum rank value in the population for that objective.

Another variation is the *sum of dominance ranks*. The dominance rank d_i of an individual for objective i is the number of individuals in the population that have a better score on objective i. Once the dominance rank vector $< d_1, ..., d_i >$ is determined, the sum of dominance ranks is computed similarly to the sum of ranks above.

C. Evaluation of GP solutions

The identification of high-performing solutions from GP runs is a challenging task when stochastic processes are being considered. SPI models produce stochastic behaviors when interpreted, and a given SPI expression can generate highly variable behaviors during independent simulations. Furthermore, even targeted solution expressions may occasionally exhibit behaviors that are far removed from the expected norm. Depending on the statistical features used to characterize the behavior, the feature scores themselves may vary significantly.

To find solutions from each experiment, a "wide net" is cast, which examines a large assortment of generated models. When a run terminates after reaching the maximum generation, the top 25 scoring individuals are saved. After all the 20 runs for an experiment have completed, these top 25 scorers from each run are combined together, resulting in 500 candidate models. Each expression is then interpreted 100 times, and the mean statistical feature score is calcuated for each feature of interest. The z-score is determined between the expression score and the corresponding score of the target model. A 95% match is considered a "hit", and this is recorded. This results in a hit vector per candidate solution. The individuals with the highest-performing hit vectors can then be given closer scrutiny as potential solutions for the experiment.

IV. Grammatical GP

TABLE III
SPI GRAMMAR

Expr ::=	Processes	
Processes ::=	($proc_0 \equiv$ Choice) # ($proc_1 \equiv$ Choice)	
Choice ::=	Term | Choice + Choice	
Term ::=	Delay(i) | Pi | Pi.Pi | Pi.Proc(i) | Pi.Pi.Proc(i) | Delay(i).Pi.Proc(i) | Pi.Delay(i).Proc(i) | Pi.(Proc(i) # Proc(i)) | Delay(i).(Proc(i) # Proc(i)) | (Proc(i) # Proc(i))	
Pi ::=	in(c) | out(c)	
Delay(i) ::=	delay(time)	($time = 10^{(i \bmod 7)-4}$)
Proc(i) ::=	$proc_j$	($j = i \bmod 2$)

Table III shows a CFG grammar used to implement the SPI-calculus in the GP system. This grammar is used for the Lotka-Volterra and Repressilator experiments. The variable i (used in Proc(i) calls and delay terms) is a terminal representing an ephemeral integer. The variable c (in the Pi rule) is an ephemeral channel label. The rule *Expr* defines two processes, $proc_0$ and $proc_1$. Each process is a choice expression. The terms in these expressions consist of a variety of patterns of communications (Pi), stochastic delays (*delay(time)*), and calls to the processes. The *Term* rule definition, albeit not very elegant, results in trees with reduced depth. The rule for delays uses the integer argument to create a floating-point delay value, selected uniformly from the set $\{0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$. In this way, the integer argument represents a (discrete) logarithmic distribution of delay values. Compare this to using a uniformly distributed float variable, which would be biased against smaller values.

The Oregonator experiment uses a variation of the grammar in Table III, in which either 2 or 3 processes are possible (*Processes*), and no delay terms are used (*Term*).

The GP system used is the DCTG-GP system [33]. This is a Prolog-based GP system that uses definite clause translation grammars (DCTGs) – a form of logic-based attribute grammar.

V. Experiments

A. Lotka-Volterra Model

$$Proc_0 = in(c1).Proc_0$$
$$\mathbf{Proc_1} = \mathbf{out(c1)}.((\mathbf{Proc_1} \# \mathbf{Proc_1}) + \mathbf{in(c2)})$$
$$\mathbf{Proc_2} = \mathbf{out(c2)}.((\mathbf{Proc_2} \# \mathbf{Proc_2}) + \mathbf{delay(10.0)})$$
$$Lotka = Proc_0 \# (500@Proc_1) \# (500@Proc_2)$$
$$rate(c1) = 10.0$$
$$rate(c2) = 0.01$$

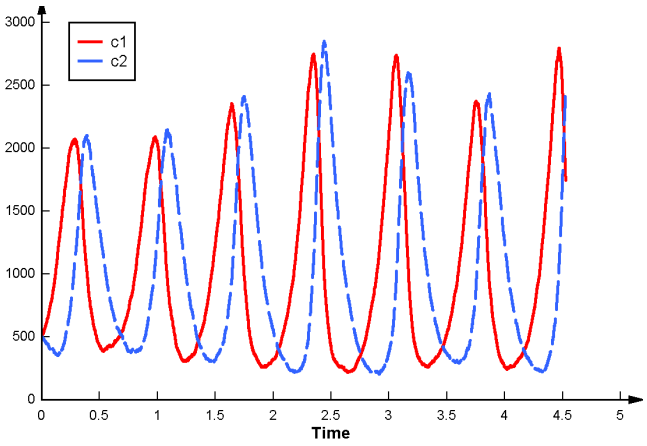


Fig. 1. Lotka-Volterra target model and plot

The first process studied is the Lotka Voltera Equation [34]. This is a dynamic model of predator-prey relationships, in which the number of predator and prey species varies over time. The model shows the dynamic relationship between predators and prey population levels in an environment. It is

TABLE IV
STATISTICAL FEATURES FOR LOTKA-VOLTERRA

Feature	c1	c2
σ	630.8	630.1
sc	6.94	7.03

TABLE V
SPI EVALUATION PARAMETERS FOR LOTKA-VOLTERRA.

Parameter	Value
MOP strategy	sum of dominance
Rank weights	no
Stream filter	50
Max time	5.0
Max ticks	400,000
Log delays	yes

an example of an autocatalytic reaction, as the product of the reaction (increasing predator or prey population) will act as a positive feedback for continued increases of species levels. This continues until an untenable population level is reached, which results in a decline in levels. The effect is a pair of oscillating curves.

The SPI model for the Lotka Volterra model is shown in Figure 1, and is taken from [15]. The expression in boldface is the SPI expression to be reconstructed with GP, while the rest of the SPI model is supplied to GP as a wrapper expression. The graph shows the varying predator and prey species levels over time. The two curves exhibit both regularity (period) and irregularity (magnitude). The curves are correlated with each other (although multivariate statistics such as correlation are not used here). The target SPI model was simulated 1000 times, and the average values for a number of statistical features were determined. The set of features selected for the GP runs are shown in Table IV. Both channels exhibit essentially the same behavior, and differences shown are due to statistical variance.

Parameters used for evaluating the Lotka-Volterra models are shown in Table V. Many of these parameters are chosen with the particular behavioral characteristics of the model in mind. For example, limits of 5.0 simulation time units and 400,000 simulation iterations are used. Delay term transitions are counted as events in the stream. The resulting time series is filtered by selecting one out of every 50 consecutive values in the series. The MOP strategy used here is the sum of the (unweighted) dominance rank score.

The GP parameters used for this and the other experiments are in Table VI. Most of the parameters are standard in the GP literature (eg. [35]). Of note is that the initial randomized population is oversampled. A total of 4000 trees are generated, and the fittest 1500 scoring trees in it are selected as the starting population. All the GP trees in the population are unique. (Syntactically and behaviorally identical models can arise after translation to SPI, however.)

TABLE VI
GP PARAMETERS FOR ALL EXPERIMENTS

Parameter	Value
Runs	20
Solutions per run	25
Initial popn.	4,000
Population	1,500
Unique popn.	yes
Max tree depth (init)	8
Max tree depth	12
Prob. crossover	90%
Prob. internal crossover	85%
Prob. terminal mutation	75%

B. Repressilator

$$\mathbf{Gene(a,b)} = \mathbf{delay(0.1)}.(\mathbf{Prot(a,b)}\#\mathbf{Gene(a,b)}) + \mathbf{in(a)}.\mathbf{delay(0.0001)}.\mathbf{Gene(a,b)}$$
$$\mathbf{Prot(a,b)} = \mathbf{out(b)}.\mathbf{Prot(a,b)} + \mathbf{delay(0.001)}$$
$$Repressilator = Gene(x,y)\#Gene(y,z)\#Gene(z,x)$$
$$rate(x) = rate(y) = rate(z) = 1.0$$

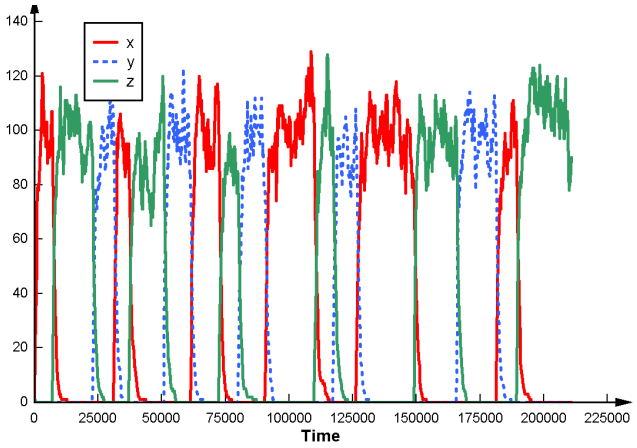


Fig. 2. Repressilator target model and plot

TABLE VII
STATISTICAL FEATURES FOR REPRESSILATOR

Feature	Value
μ	34.42
σ	40.27
sc	16.09
chaos	0.029
freq	11694.0

The repressilator is a synthetic gene regulatory network [12]. The SPI model in 2 shows 3 $Gene(a,b)$ terms. When the a argument is strengthened in quantity (probability) in the system, the b argument quantity is inhibited. Conversely, a reduction in a results in an increase in b. When this logic gate behavior is placed in a feedback loop of 3 genes, the irregular

TABLE VIII
SPI EVALUATION PARAMETERS FOR REPRESSILATOR.

Parameter	Value
MOP strategy	sum of ranks
Rank weights	freq=3 (rest=1)
Stream filter	3
Max time	200,000
Max ticks	20,000
Log delays	yes

oscillations in the given plot arise. These oscillations are much more irregular and noisy than those in Figure 1. The period and duration of each channel's plot is determined by stochastic influences in the system. A detailed discussion of stochastic models of the Repressilator using SPI are in [12].

The irregular nature of the repressilator makes it a good candidate for description by statistical features. As is often the case, the problem becomes one of selecting effective features to use. This experiment uses the features in Table VII. An intention here is to see whether evaluating one of the three channels will be sufficient for inferring the Repressilator model.

The GP parameters used are in Table VI. The SPI parameters are in Table VIII. Here, a sum of ranks scoring is used, where the frequency rank uses a weight of 3 (the rest use 1). See Section V-A for a description of the other parameters.

C. Oregonator

TABLE IX
STATISTICAL FEATURES FOR OREGONATOR

Feature	c1	c3	c5
μ	165.2	87.2	173.5
σ	172.0	189.5	339.4
kurt	-0.21	3.68	3.00
tera	0.24	1.80	1.88
freq	1.00	0.63	0.72

TABLE X
SPI EVALUATION PARAMETERS FOR OREGONATOR.

Parameter	Value
MOP strategy	normalized sum of ranks
Rank weights	freq=2 (rest=1)
Stream filter	25
Max time	3.5
Max ticks	250,000

Like the Lotka-Volterra model, the Oregonator model (Figure 3) is an autocatalytic reaction [36]. The SPI expression is based on one in [15]. It is more complex than the previous SPI models. The example plot shows three oscillating curves for 3 channels. The first cycle is irregular from the other cycles, as the system is moving from its initial state at that time. Afterwards, the oscillating curves are fairly regular.

$$
\begin{aligned}
&P_0 = in(c1).P_0\\
&P_1 = in(c3).P_1\\
&P_2 = in(c5).P_2\\
&\mathbf{P_3 = out(c2) + out(c4) + in(c4)+}\\
&\qquad \mathbf{out(c3).(P_5\#P_3\#P_3)}\\
&\mathbf{P_4 = out(c1).P_3 + in(c2)}\\
&\mathbf{P_5 = out(c5).P_4}\\
&Oregonator = P_0\#P_1\#P_2\#P_3\#P_4\#P_5\\
&rate(c1) = rate(c) = 2.0\\
&rate(c3) = 52.0\\
&rate(c4) = 0.008\\
&rate(c5) = 26.0
\end{aligned}
$$

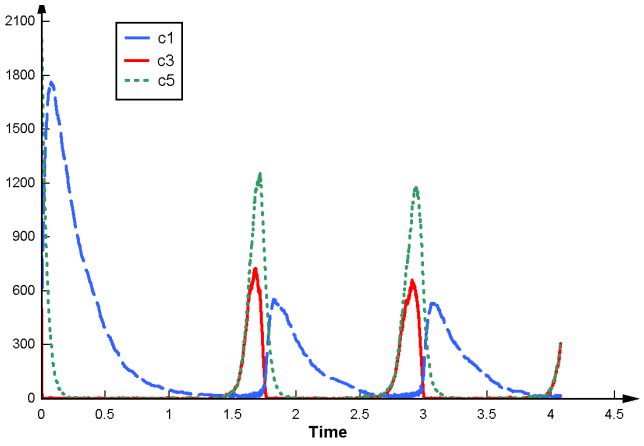


Fig. 3. Oregonator target model and plot

The Oregonator experiment uses statistical features for 3 channels corresponding to the behavior plot (Table IX). As usual, these are mean values computed from 1000 simulations of the target model. Since each curve in Figure 3 shows noticeable differences in shape, the feature values have corresponding differences. The GP parameters used are in Table VI, and the SPI parameters are in Table X.

VI. RESULTS

TABLE XI
SUMMARY OF SOLUTION EXPRESSION RESULTS. TOTAL 500 SOLUTIONS PER EXPERIMENT (20 RUNS).

	# Exact solutions	# Unique expressions	# Duplicated expressions
Lotka-Volterra	25	64	436
Repressilator	24	161	339
Oregonator	0	439	62

Table XI summarizes the results of the runs. Examining the data for the Lotka-Volterra experiment, the solution was found 25 times during the 20 runs. Because GP trees duplication is not allowed in a population, these copies of solutions can be produced during separate runs, or after conversion to SPI calculus expressions after a single run. From the total pool of

TABLE XII
NUMBER OF UNIQUE SOLUTIONS HAVING GIVEN NUMBER OF FEATURE HITS. MAXIMUM POSSIBLE HITS: LOTKA 4, REPRESSILATOR 5, OREGONATOR 15.

	Total unique solutions	Total hits 0	1	2	3	4	5
Lotka-Volterra	64	44	14	3	2	1	-
Repressilator	161	62	68	14	16	0	1
Oregonator	439	76	108	130	83	36	6

500 candidate solutions, 64 unique SPI-calculus expressions were generated, while the remaining 436 models were duplicates of these 64 expressions. Similarly, the repressilator runs also successfully found the exact solution multiple times. The Oregonator experiment, however, was unsuccessful in finding the target model. It also resulted in the greatest variety of solutions (439), probably due to a lack of convergence to behaviorally strong niches in the search space.

Table XII show the spread of feature hits amongst the unique solutions from each experiment (see the discussion of feature hits in Section III-C). These histograms show the quality of solutions from the runs after duplicate expressions are removed. Although there are many solution expressions with low numbers of overall feature hits, this might indicate that the generation of 25 solutions per run is excessive. The table indicates that evolution is honing in on feature characteristics with a high degree of precision. The ability of a candidate solution to be within a feature score of the target model within a 95% significance level is fairly impressive. When this is done with multiple features simultaneously, evolution is having a positive effect.

The performance of the Oregonator hits is mediocre. Of the total of 15 feature tests used, 6 expressions manage to obtain hits on 6 of the features simultaneously.

The performance charts in Figure 4 plot the average error per generation for the best performer and population average, for the Lotka-Volterra and Oregonator experiments. The plots are averaged over 20 runs. The error (Y axis) uses a log scale. In the Lotka-Volterra plots, the top 2 curves are for the standard deviation feature scores, while the bottom 2 are serial correlation. The Oregonator show curves for all 15 features (5 per channel) that were tested. In the "best" plots in (a) and (c), there is a general downward trend in some curves, although others do not show noticeable improvement. This is best explained in the population average plots in (b) and (d). A characteristic of sum of rank scoring is that the performance of some features will be sacrificed, should that promote the improvement of the best performing models in the population. In chart (d), most Oregonator features incur an increase in error during the first 10 generations – directly opposite to that seen in the best performers in chart (c). The bottom set of features in the plot, however, improve throughout the run, and many of the initially sacrificed features then begin improving as well.

Despite the failure of the Oregonator runs to find the target solution, one solution in Figure 5 showed oscillating behavior somewhat similar to the desired behavior. This expression had

$$\mathbf{P_3} = \mathbf{out(c1).out(c1).P_3 + out(c5).out(c5)+ out(c3).(P_4 \# P_3) + in(c5).P_5}$$
$$\mathbf{P_4} = \mathbf{out(c5).out(c1) + in(c1).out(c1)+ out(c4).P_4 + out(c3).(P_4 \# P_5)}$$
$$\mathbf{P_5} = \mathbf{out(c3).P_4}$$

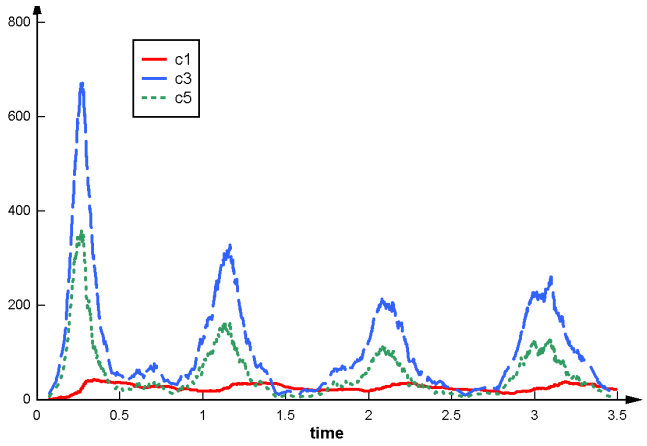


Fig. 5. Oregonator solution and plot

feature score hits on 4 objectives: c3 kurt, c3 freq, c5 tera, c5 freq. It is possible that the matches to the adjusted frequency scores resulted in this oscillation.

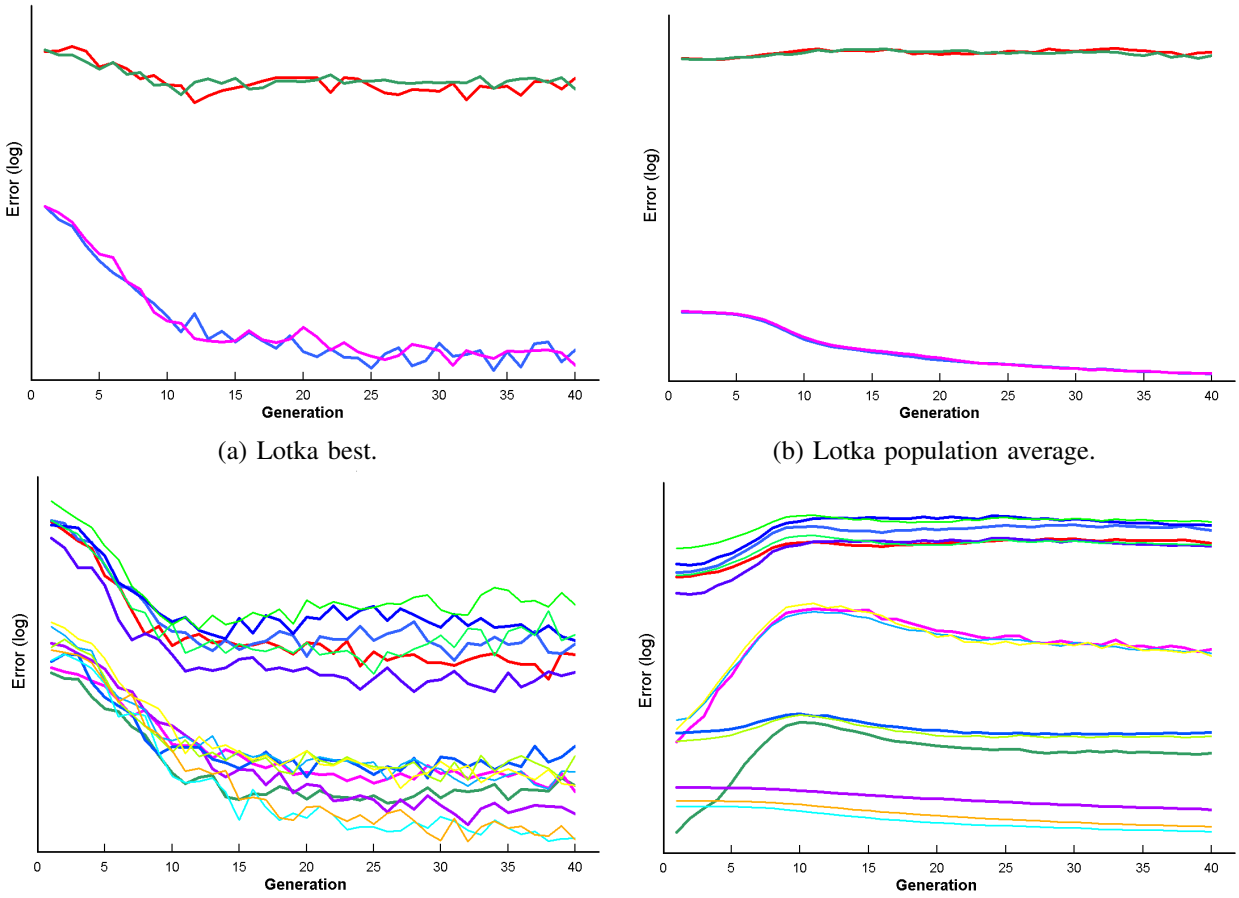
VII. DISCUSSION

The positive results of the Lotka-Volterra and repressilator experiments shows that an effective selection of statistical features is critical for success of model evolution. One observation is that all the solutions that are behaviorally equivalent to the target (100% hits) are also syntactic matches to the target expression. In other words, the features effectively characterized the target model behavior. Additionally, in the case of the repressilator, only one channel was required to characterize it.

The impressive performance of the Lotka-Volterra and repressilator runs is also due to the sum of ranks strategy. Previous attempts used Pareto ranking were not as effective at finding repressilator solutions [17]. The tendency of Pareto domination to allow outlier solutions is detrimental to solution quality. Initial attempts that used single-objective statistical weighting were even less effective [16].

The reasons for the failure of the Oregonator experiment may be due to a number of factors:

1) The features selected were inappropriate.
2) An excessive number of features and channels were used, resulting in a difficult search space.
3) The time limit for the simulation was too short. It was purposefully minimized due to the lengthy simulation times encountered. This gave too small a window for GP to evolve the desired oscillating behavior.
4) The SPI calculus expression complexity in the target warrants further grammatical constraints.



(a) Lotka best.

(b) Lotka population average.

(c) Oregonator best.

(d) Oregonator population average.

Fig. 4. Error plots for Lotka-Volterra and Oregonator. Averaged over 20 runs.

On the other hand, the existence of similar oscillating behavior in one Oregonator model shows that there is the potential for success.

There is very little research applying computational intelligence techniques to process algebra. Earlier work evolved CCS expressions [37], [38], which were neither stochastic nor time-based in nature. More complex bio-network models are effectively evolvable if higher-level modeling languages are considered. Imada used GP with a logic gate language, implemented on top of the SPI-calculus [18]. Target models were characterized with statistical features (this paper borrows some of them), and multiple feature scores were combined with statistically weighting. A number of gene regulatory networks were succesfully evolved with GP. Another example is the PIM bio-modeling algebra, which is also implemented in the SPI calculus [39]. The techniques of this paper were applied to PIM, and a number of models were successfully evolved [40].

Related work using evolutionary computation and bio-modeling includes work in Petri nets [41], [42], S-systems [43], and nonlinear differential equations [44], [45]. The work by Koza *et al.* using GP to evolve metabolic networks is particularly interesting [46]. Also related is work in using GP to evolve models of noisy time series [47], [48], [49], [50].

VIII. Conclusion

This research shows a technique by which modest-sized bio-network models expressed in the SPI calculus can be automatically evolved with GP. The combination of grammatical constraints, statistical feature tests, and multi-objective summed rank scoring was shown to be effective for evolving two example stochastic models. One of the models, the repressilator, has a considerable degree of noise as well. The use of sum of ranks was found to be superior to Pareto ranking done in earlier work on the repressilator [17].

A more complex model, the Oregonator, proved to be a challenge. The main issue with it is conjectured to be the use of too many inappropriate statistical features. With further refinements, it is hoped that the Oregonator and similar models will be tractable.

Current research is examining the issue of feature selection in more detail. More systematic feature selection strategies are being investigated [51]. Other forms of feature tests for time series are being considered [28]. A more detailed examination of grammatically-constrained subsets of the SPI

calculus practical for bio-network modeling is a longer-term project being undertaken.

ACKNOWLEDGEMENT

This research supported by NSERC Operating Grant 138467.

REFERENCES

[1] D. Wilkinson, *Stochastic Modelling for Systems Biology*. Chapman & Hall/CRC, 2006.

[2] P. Baldan, N. Cocco, A. Marin, and M. Simeoni, "Petri Nets for Modelling Metabolic Pathways: a Survey," *Natural Computing*, vol. 9, no. 4, pp. 955–989, 2010.

[3] C. Chaoulya, "Petri Net Modelling of Biological Networks," *Briefings in Bioinformatics*, vol. 8, no. 4, pp. 210–219, 2007.

[4] M. Peleg, D. Rubin, and R. Altman, "Using Petri Net Tools to Study Properties and Dynamics of Biological Systems," *Journal of the American Medical Informatics Association*, vol. 12, no. 2, pp. 181–199, 2005.

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

[6] G. Paun, *Membrane Computing: An Introduction*. Springer, 2002.

[7] M. Perez-Jiminez and F. Romero-Campero, "P Systems: a new computational modelling tool for systems biology," *Transactions in Computational Systems Biology VI*, pp. 176–197, 2006.

[8] A. Deutsch and S. Dormann, *Cellular Automaton Modeling of Biological Pattern Formation*. Birkhauser, 2005.

[9] G. Ermentrout and L. Edelstein-Keshet, "Cellular Automata Approaches to Biological Modeling," *J. Theoretical Biology*, vol. 160, pp. 97–133, 1993.

[10] R. Schwartz, *Biological Modeling and Simulation*. MIT Press, 2008.

[11] BioSPI, "The biospi project," 2010, last accessed July 9, 2010. [Online]. Available: http://www.wisdom.weizmann.ac.il/ biospi/

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

[13] A. Regev, E. Panina, W. Silverman, L. Cardelli, and E. Shapiro, "BioAmbients: An abstraction for biological compartments," *Theoretical Computer Science*, vol. 325, no. 1, pp. 141–167, 2004.

[14] C. Priami, "Stochastic pi-Calculus," *The Computer Journal*, vol. 38, no. 7, pp. 579–589, 1995.

[15] A. Phillips, "The stochastic pi machine (spim)," uRL: http://research.microsoft.com/ aphillip/spim/. Last accessed Feb 14, 2011. [Online]. Available: http://research.microsoft.com/ aphillip/spim/

[16] B. Ross and J. Imada, "Evolving Stochastic Processes Using Feature Tests and Genetic Programming," in *Proc. GECCO 2009*, 2009.

[17] ——, "Using Multi-objective Genetic Programming to Synthesize Stochastic Processes," in *Genetic Programming - Theory and Practice VII*, R. Riolo, U.-M. O'Reilly, and T. McConaghy, Eds. Springer, 2010.

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

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

[20] S. Bergen and B. Ross, "Evolutionary Art Using Summed Multi-objective Ranks," in *Genetic Programming - Theory and Practice VIII*. Springer, May 2010.

[21] R. Flack, "Evolution of architectural floor plans," Master's thesis, Department of Computer Science, Brock University, 2010.

[22] C. Priami, A. Regev, E. Shapiro, and W. Silverman, "Application of a stochastic name-passing calculus to representation and simulation of molecular processes," *Information Processing Letters*, vol. 80, pp. 25–31, 2001.

[23] A. Phillips and L. Cardelli, "A Correct Abstract Machine for the Stochastic Pi-calculus," in *Proc. Bioconcur'04*, 2004.

[24] R. Milner, *Communicating and Mobile Systems: the Pi-calculus*. Cambridge University Press, 1999.

[25] ——, *Communication and Concurrency*. Prentice Hall, 1989.

[26] D. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem*, vol. 81, pp. 2340–2361, 1977.

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

[28] C. Chatfield, *The Analysis of Time Series: An Introduction*. Chapman and Hall/CRC, 2004.

[29] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," in *Information processing and technology*. Commack, NY, USA: Nova Science Publishers, Inc., 2001, pp. 49–61.

[30] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," *Data Min. Knowl. Discov.*, vol. 13, no. 3, pp. 335–364, 2006.

[31] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2007. [Online]. Available: http://www.R-project.org

[32] C. C. Coello, G. Lamont, and D. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. Kluwer, 2007.

[33] B. Ross, "Logic-based Genetic Programming with Definite Clause Translation Grammars," *New Generation Computing*, vol. 19, no. 4, pp. 313–337, 2001.

[34] Wikipedia, "Lotka-volterra equation," 2011, last accessed Feb 14, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Lotka-Volterra_equation

[35] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[36] Wikipedia, "Oregonator," 2011, last accessed Feb 15, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Oregonator

[37] B. Ross, "The Evolution of Concurrent Programs," *Applied Intelligence*, vol. 8, no. 1, pp. 21–32, January 1998.

[38] ——, "Pairwise Sequence Comparison and the Genetic Programming of Iterative Concurrent Programs," in *Proc. Genetic Programming 1998*, J. K. *et al*, Ed. Morgan Kaufmann, 1998, pp. 338–343.

[39] O. Kahramanogullari and L. Cardelli, "An Intuitive Automated Modelling Interface for Systems Biology," in *DCM'09*, 2009, pp. 1–18.

[40] B. Ross, "The Evolution of Higher-level Biochemical Reaction Models," Brock U., Dept. of Computer Science, Tech. Rep. CS-10-02, December 2010.

[41] J. Kitagawa and H. Iba, "Identifying Metabolic Pathways and Gene Regulation Networks with Evolutionary Algorithms," in *Evolutionary Computation in Bioinformatics*, G. Fogel and D. Corne, Eds. Morgan Kaufmann, 2003, pp. 255–278.

[42] J. Moore and L. Hahn, "Systems Biology Modeling in Human Genetic Using Petri Nets and Grammatical Evolution," in *Proc. GECCO 2004*, K. D. et al., Ed. Springer, 2004, pp. 392–401, lNCS 3102.

[43] H. Wang, L. Qian, and E. Dougherty, "Inference of Gene Regulatory Networks using S-System: A Unified Approach," in *CIBCB 07*, G. Volkert, Ed. IEEE, 2007, pp. 82–89.

[44] A. Floares, "Automatic Inferring Drug Gene Regulatory Networks with Missing Information Using Neural Networks and Genetic Programming," in *WCCI 2008*, J. Wang, Ed. IEEE, 2008, pp. 3078–3085.

[45] L. Qian, H. Wang, and E. Dougherty, "Inference of noisy nonlinear differential equation models for gene regulatory networks using genetic programming and kalman filtering," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3327–3339, July 2008.

[46] J. Koza, W. Mydlowec, G. Lanza, J. Yu, and M. Keane, "Reverse Engineering and Automatic Synthesis of Metabolic Pathways from Observed Data using Genetic Programming," Stanford Medical Informatics, Tech. Rep. SMI-2000-0851, 2000.

[47] A. Borrelli, I. De Falco, A. Della Cioppa, M. Nicodemi, and G. Trautteura, "Performance of genetic programming to extract the trend in noisy data series," *Physica A: Statistical and Theoretical Physics*, vol. 370, no. 1, pp. 104–108, 2006.

[48] Y. Jin and J. Branke, "Evolutionary Optimization in Uncertain Environments - A Survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[49] K. Rodriguez-Vazquez and P. J. Fleming, "Evolution of mathematical models of chaotic systems based on multiobjective genetic programming," *Knowledge and Information Systems*, vol. 8, no. 2, pp. 235–256, Aug. 2005.

[50] W. Zhang, G. Yang, and Z.Wu, "Genetic Programming-based Modeling on Chaotic Time Series," in *Proc. 3rd Intl Conf. on Machine Learning and Cybernetics*. IEEE, 2004, pp. 2347–2352.

[51] H. Liu and H. Motoda, *Computational Methods of Feature Selection*. Chapman and Hall/CRC, 2007.