School of Computing Science,
University of Newcastle upon Tyne

Automatic Parameterisation of Stochastic Petri Net Models of Biological Networks

O. J. Shaw, L. J. Steggles and A. Wipat

Technical Report Series

CS-TR-909

May 2005

Published by the University of Newcastle upon Tyne,
School of Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, UK.

Automatic Parameterisation of Stochastic Petri Net Models of Biological Networks

Oliver Shaw [1]

School of Computing Science
University of Newcastle upon Tyne
Newcastle upon Tyne, UK

Jason Steggles [2]

School of Computing Science
University of Newcastle upon Tyne
Newcastle upon Tyne, UK

Anil Wipat [3]

School of Computing Science
University of Newcastle upon Tyne
Newcastle upon Tyne, UK

Abstract

Stochastic simulations are able to capture the fine grain behaviour and randomness of outcome of biological networks not captured by deterministic techniques. As such they are becoming an increasingly important tool in the biological community. However, current efforts in the stochastic simulation of biological networks are hampered by two main problems: firstly the lack of complete knowledge of kinetic parameters; and secondly the computational cost of the simulations. In this paper we investigate these problems using the framework of stochastic Petri nets. We present a new stochastic Petri net simulation tool *NASTY* which allows large numbers of stochastic simulations to be carried out in parallel. We then begin to address the important problem of incomplete knowledge of kinetic parameters by developing a distributed genetic algorithm, based on NASTY's simulation engine, to parameterise stochastic networks. Our algorithm is able to successfully estimate kinetic parameters to replicate a systems behaviour and we illustrate this by presenting a case study in which the kinetic parameters are derived for a stochastic model of the stress response pathway in the bacterium *E.coli*.

Keywords: Petri nets, Genetic Algorithms, Systems Biology, Kinetic parameters, Stochastic simulation

1 Introduction

Over the past decade whole genome sequencing has revolutionised the biological sciences [16]. A new era of data rich science in biology has arisen based on the whole genome projects and the wealth of post-genomic studies that they facilitate, aimed at understanding the many complex cellular processes that occur in living organisms. Hence there is now an unprecedented amount of data available about biological systems. These new data resources have enabled the systems wide study of biological systems, referred to as *Systems Biology*, in which the goal is to understand how the individual biological parts interact to yield the behaviour of the whole system [13]. One important approach in Systems Biology is the modelling and simulation of a biological networks to help understand and predict the behaviour of these complex systems. The simulation of biological networks are carried out with either deterministic simulators [18], stochastic simulators [29], or hybrid simulators [14], each of which have their own advantages and disadvantages (see Section 2.1). Stochastic simulations have been shown to capture the fine grain behaviour and randomness of outcome of biological networks not captured by deterministic techniques [19] and as such are becoming an increasingly important technique. However, current efforts in the stochastic simulation of biological networks are hampered by two main problems:

(i) First, there is a lack of quantitative data on molecular concentrations and kinetic parameters that are essential to the successful simulation of biological networks [21].

(ii) Secondly there is a large computational cost to stochastic simulation of these networks. A recent review suggested that an average personal computer would take a whole day to simulate 100 minutes of a 100 reaction system [3]. This problem is exacerbated since multiple repetitions of simulations are often required.

In this paper we consider addressing these problems within the framework of stochastic Petri nets [17]. Stochastic Petri nets have been shown to be an appropriate tool for the simulation of biological networks [10,31]. Utilising Petri nets not only allows fast, accurate simulation of a system, it also opens the model up to the wide range of analysis techniques available within the Petri net framework [30,28]. We begin by presenting a new stochastic Petri net simulator *NASTY* (Not Another Simulator Thank You) which is compliant with the Petri Net Markup Language (PNML) [2]. This simulator importantly

[1] Email: o.j.shaw@ncl.ac.uk
[2] Email: l.j.steggles@ncl.ac.uk
[3] Email: anil.wipat@ncl.ac.uk

uses mass action kinetics as a default since this is fundamental to the stochastic simulation of biological networks [5]. It addresses the computational cost of carrying out stochastic simulations by allowing large numbers of stochastic simulations to be performed in parallel and uses a distribution algorithm to ensure effective use is made of all available processing power.

The problem of incomplete kinetic parameter data is at present a major limiting factor in the application of stochastic modelling in Systems Biology [20]. We propose an algorithm for estimating missing parameters based on using a *genetic algorithm* [11] approach. The genetic algorithm we develop is based on evolving a population of individuals that encode different sets of kinetic rates. Each individual in a population can be assigned a *fitness value* [11] that indicates how closely its resulting simulations correlate to a set of training data. This process requires large numbers of multiple simulations and so to overcome the inherent cost of performing stochastic simulations we employ the NASTY simulator engine which distributes simulations over multiple processors. The genetic algorithm allows the population to evolve over time by individuals *mutating* (random rate changes), *crossovers* (two individuals swapping rate information), *cloning* (an individual progressing unchanged to the next generation) and *culling* (removing unfit individuals) [9]. The result is a probabilistic algorithm which allows a population of kinetic rates to evolve to towards solutions that match given training data.

The genetic algorithm we developed appears to successfully estimate solutions for missing rate parameters that replicate the core behaviour of a given system. Further work is required to improve this process and we suggest some future avenues of research in Section 5. We illustrate our simulation tools by presenting a case study in which a stochastic model of the stress response pathway in the bacterium *Escherichia.coli* (*E.coli*) [31] is derived from a set of incomplete data. We validate our resulting model by performing a number of behavioural experiments which we compare to a benchmark model in the literature [31].

The paper is organised as follows. In Section 2 we provide an introduction to the simulation of biological networks and give a brief overview of stochastic Petri nets. In Section 3 we describe the NASTY Stochastic Petri net simulator and discuss in detail the genetic algorithm developed for estimating kinetic parameters. The above tools are illustrated in Section 4 where a detailed case study of deriving a stochastic model of the stress response of the bacterium *E.coli* is presented and validated. Finally, in Section 5 we give our concluding remarks.

2 Background

2.1 Simulation of Biological Networks

Simulation of biological networks can be carried out with a number of techniques depending on the assumptions made about the system. There are currently three main methods in use for simulating biological networks: deterministic simulations, carried out with Ordinary Differential Equations (ODE's) [18]; stochastic simulations, carried out with the Gibson-Bruck algorithm [6], the Gillespie algorithm [7] or stochastic Petri nets [17]; and finally, hybrid simulation techniques [14] are appearing which attempt to amalgamate the other two approaches. Currently a large amount of the systems biology community's effort is directed towards deterministic simulation (as exemplified by the large proportion of deterministic models at the Systems Biology Markup Language [12] (SBML) website [29]).

Deterministic simulation techniques such as ODE's assume, among other things, that: (a) concentrations vary deterministically over time and (b) concentrations vary continuously and continually. However these assumptions may not be valid for some important aspects of biological systems. With regard to assumption (a), an analysis of cell protein production (i.e. transcription and translation events) showed that proteins are produced in variable numbers at random time intervals [19]. Importantly these variations can lead to large time differences between successive events in regulatory cascades and subsequently produce probabilistic outcomes in switching between alternative regulatory paths [19]. These stochastic effects may be a source of some of the unexplained phenotypic variations in isogenic populations [19], and deterministic techniques are unable to capture these interesting and important behaviours [32,19]. In stochastic modelling assumption (a) is replaced with (a') "the timing of discrete reactions is random" [5]. Assumption (b) breaks down theoretically at the low molecular concentrations found in single cell based biological systems [5]. For example it is reasonable to suggest there is a continuation of concentrations between 6mols/l and 7mols/l, however this assumption is clearly not valid under low concentrations as there is no midpoint between 10 and 11 molecules. In stochastic modelling assumption (b) is replaced with (b') "concentrations change by discrete numbers of molecules, corresponding to single reaction events" [5].

An ODE based model of a biological system may produce results equivalent to the average of stochastic simulations. This is because ODE's can reproduce the system behaviours at the macroscopic scale, while a stochastic simulator captures more fine grained behaviours at the meisoscopic level [5]. If the behaviour is subject to switching mechanisms between alternate pathways [19] individual stochastic simulations would lead to different system states, for example the switch between lysis or lysogeny [1]. The ODE simulator would not be able to capture this behaviour. Hence there are real practical reasons why stochastic modelling techniques are an appropriate and necessary

method for the simulation of biological networks.

The overall aim of stochastic modelling is to test the understanding we have of the system and to make predictions about the behaviour of a system. However, despite the extensive data collected, there are still practical limitations to the knowledge available for modelling biological systems. In particular, there is a lack of quantitative data on molecular concentrations [21], a lack of kinetic parameters [21], and "unknowns" are often present in a system, such as uncharacterised proteins that may contribute to a systems behaviour [3]. While useful models of non trivial biological systems have been analysed using stochastic techniques, many kinetic parameters have been *"adjusted"* or *"chosen"* by hand for certain system behaviours [1,31]. With most simulation tools relying heavily on user input in the creation and simulation of models, the process of altering kinetic parameters to find the desired behaviour is likely to be a time consuming process. In this paper we address this important issue by developing appropriate tool support to allow the user to utilise distributed computing power, and automatically discover suitable kinetic parameters from which meaningful models can be constructed and analysed.

2.2 Stochastic Petri nets

The theory of Petri nets [25] provides a graphical notation with a formal mathematical semantics for modelling and reasoning about concurrent, distributed systems. As well as being straight forward to interpret visually, Petri nets provide a range of powerful analysis and simulation techniques [17,24] and have been used extensively in Computing Science [26]. A Petri net is a directed bipartite graph and consists of four basic components: *places* which are denoted by circles; *transitions* denoted by black rectangles; *arcs* denoted by arrows; and *tokens* denoted by black dots. A simple example of a Petri net is depicted in Figure 1. The places, transitions and arcs describe the static structure of the Petri net. Each transition has a number of *input places* (places with an arc leading to the transition) and a number of *output places* (places with an arc leading to them from the transition). Note arcs that directly connect two transitions or two places are not allowed. From a biological perspective we normally view places as representing a particular molecular species, with the number of associated tokens on a place representing the amount present, and transitions represent chemical and biological reactions [30,27].

The state of a Petri net is given by the distribution of tokens on places within it, referred to as a *marking*. The *state space* of a Petri net is therefore the set of all possible markings. The dynamic properties of the system are modelled by transitions which can fire to move tokens around the places in a Petri net. Transitions are said to be *enabled* if each of their input places contain at least one token. An enabled transition can *fire* by consuming one token from each of its input places and then depositing one token on each of its output places.

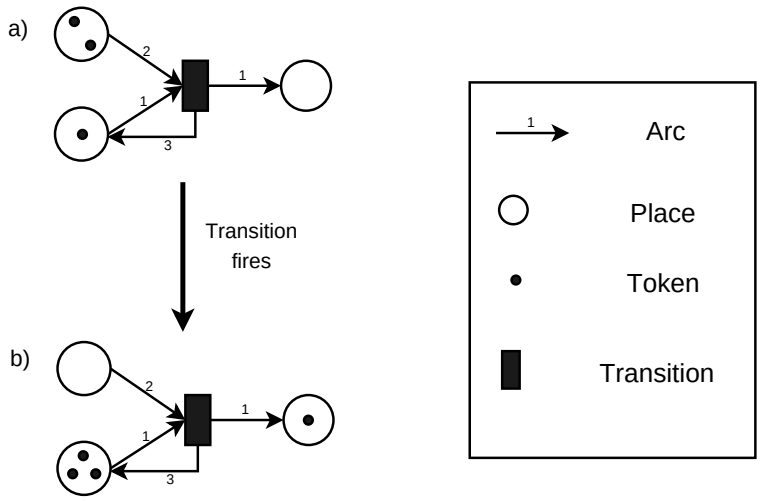


Fig. 1. An illustration of a simple Petri net before and after firing a transition

In stochastic Petri nets the basic framework introduced above is extended to allow time to be considered. We use the Generalised Stochastic Petri Net (GSPN) framework [17] in this paper which associates a rate with each transition to define how many times it will fire per unit of time. In the sequel we fix the unit of time to be seconds. This extension means that the firing of enabled transitions is now dependent on both the current marking and the associated firing rates, enabling us to simulate the timed behaviour of a system.

Let us now consider in detail the simulation algorithm underlying a GSPN [17]. We begin by using the firing rate of each transition to calculate a *negative exponential probability density function* (pdf) [23,17] which has a "memoryless" property that simplifies the simulation procedures. For a more detailed discussion on the mathematical theory behind this assumption see [23]. A delay to fire is then calculated for each enabled transition by sampling the pdf and all the enabled transitions are then placed in a priority queue. The transition with the smallest delay is then fired (as described above) and the global time updated appropriately. Firing this transition may have possibly enabled or disabled transitions which have input places connected to the fired transition and may also have changed the delay associated with the firing transition since mass action kinetics are used. Thus it is necessary to re-sample those transitions that have had their inputs changed by the firing step and then either insert, move, or remove them in the priority queue as appropriate. This procedure is then repeated until there are no more enabled transitions or some pre–determined stop time has been achieved. Notice that this algorithm relies on local re-sampling, i.e. not having to re–sample the non–effected transitions in the queue. This is made possible due to the "memoryless" property of the negative exponential pdf. The result is a large speed up in the simulation algorithm when compared to the Gillespie algorithm [7]. We note that this algorithm is equivalent to the Gibson-Bruck algorithm [6] and in fact, the dependability tree discussed in [6] is, in effect, the underlying Petri net

structure.

3 Stochastic Simulation and Parameter Discovery

In this section we describe the tools we developed for constructing stochastic Petri net models of biological systems. We begin by giving an overview of a distributed simulation environment for stochastic Petri nets called NASTY developed specifically with simulating biological models in mind. We then consider developing a genetic algorithm which uses the NASTY simulator to perform parameter fitting on incomplete stochastic Petri net models.

3.1 The NASTY Simulator

In order to address the problem of kinetic parameter fitting we first needed an efficient stochastic Petri net simulator to form the core of our tool. Such a simulator would need to be well–suited to modelling biological systems, be able to efficiently perform large numbers of stochastic runs and collate the resulting information. While a number of stochastic Petri net simulators already exist [26] we found that none of these fully met our requirements. We therefore decided to develop a new simulator tool called *NASTY* (Not Another Simulator Thank You). NASTY was implemented in Java and has three main elements: a core stochastic Petri net simulation engine; a user friendly GUI interface for the construction of models; and a distributed job scheduling protocol to allow simulations to be carried out on multiple machines. The NASTY tool was developed to provide a suitable simulation environment for biological networks and to this end uses mass action kinetics as a default. NASTY allows users to utilise the processing power of a large cluster of machines, assuming these machines have a shared file system. The architecture of NASTY is shown in Figure 2. As discussed in Section 1, performing multiple simulations can require a prohibitively large amount of computation time. However, since each individual simulation is an independent job the task is straightforward to parallelise. NASTY makes use of this fact and works by farming out jobs to a large number of servers to make performing multiple runs computationally feasible.

NASTY provides a Java swing GUI which allows the user to build models by hand. The tool is also compliant with the Petri Net Markup Language (PNML) [2], allowing users to import or export models from/to the wide range of existing Petri net tools [26]. Data standards are also emerging in the biological community, for example the *Systems Biology Markup Language* (SBML) [12] is an important standard which is becoming the *lingua franca* of systems biologists. In order to ensure the applicability of our tools we have developed tools for interchanging PNML and SBML models [30]. Hence the tools we present here can be seen as having real practical application for the biological modelling community.

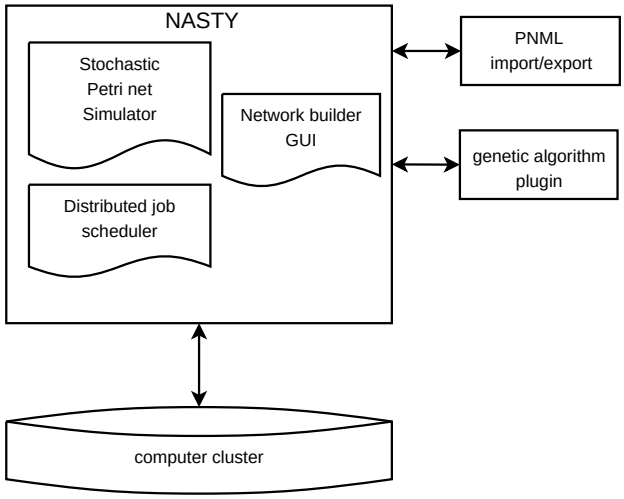


Fig. 2. A simplified view of the NASTY simulators architecture.

3.2 A Genetic Algorithm for Parameter Fitting

In this paper we are interested in taking a stochastic Petri net model and then discovering some or all of the kinetic parameters required to produce a given "gold standard" behaviour. Due to the combinatorial state–space expansions of searching for multiple kinetic parameters, we believe the problem of automatic parameterisation to be NP-complete. Such NP-complete optimisation problems, as typified by the *travelling salesman* problem [4], have no known efficient algorithms for finding exact solutions. Instead heuristic techniques are applied which allow solutions to these hard problems to be estimated. There are numerous heuristic techniques available in the literature, such as *simulated annealing* [15], *tabu searches* [33], and *genetic algorithms* [11]. We choose to develop a *genetic algorithm* here due to the ease with which genetic algorithms can be parallelised and in particular, build on the ideas presented in [11]. We note that applying such heuristic approaches to parameter fitting has, to some extent, been considered with ODE models [22]. However, there does not appear to have been much work in this area for stochastic techniques probably due to the prohibitively large amount of time required to perform the necessary multiple simulations on a single CPU. We are able to address this problem by making full use of the facilities offered by NASTY.

The genetic algorithm approach is based on applying a simplified interpretation of Darwinian evolution in which a population of individuals is allowed to evolve [11]. Each individual represents a "chromosome" that encodes a possible solution to the given optimisation problem. A measure of the correctness of a solution, known as the *fitness function*, can be calculated from the data encoded on the chromosome. The fitness of the individual relative to the population is equivalent to the likelihood that the individual's genes progress to the next generation. The idea is to allow a population of solutions to evolve using techniques analogous to those found in real organisms, such as

"crossovers", "mutations" and "cloning" [11]. Since the evolution of fitter individuals is favoured, the aim is to evolve a population that encodes accurate solutions to the given problem.

In our genetic algorithm each possible solution to a parameter fitting problem will be represented by a single "chromosome", where each kinetic parameter is representative of a "gene" [11]. We model these chromosomes simply as vectors of floating point numbers. The fitness of each individual (chromosome) is then calculated by simulating the stochastic Petri net model using the individual's rates and assessing how closely the results match the required behaviour. The resulting genetic algorithm is presented below in pseudo code form:

Algorithm 1 The Genetic Algorithm

Initialise Population P_0
for $g = 0$ to MAX **do**
  for Solutions $s \in P_g$ **do**
    Simulate s to calculate its fitness
  end for
  Create new empty population P_{g+1}
  while $Size(P_{g+1}) < (Size(P_g) - CLONES)$ **do**
    Select $s1$ and $s2$ from P_g using fitness values
    Crossover $s1$ and $s2$ to produce $s3$ and $s4$
    Add $s3$ and $s4$ to P_{g+1}
  end while
  while $Size(P_{g+1}) < Size(P_g)$ **do**
    Select s from P_g using fitness values
    Insert s into P_{g+1}
  end while
  for Solutions $s \in P_{g+1}$ **do**
    if $Random() < MUTES$ **then**
      Mutate s within P_{g+1}
    end if
  end for
end for

The algorithm starts with the initialisation phase in which an initial population P_0 of individuals is created by randomly selecting rates. The population is then simulated to allow the fitness of each individual to be assessed. This involves simulating each individual a number of times to obtain an average of its stochastic time trajectories and this is done efficiently by farming out the simulation tasks to the server pool. The resulting average time trajectory is then compared to a gold standard to obtain a fitness score for the individual. The fitness of the whole population is then calculated by summing the individual fitness scores.

The next step is to begin the selection process for the next population. In

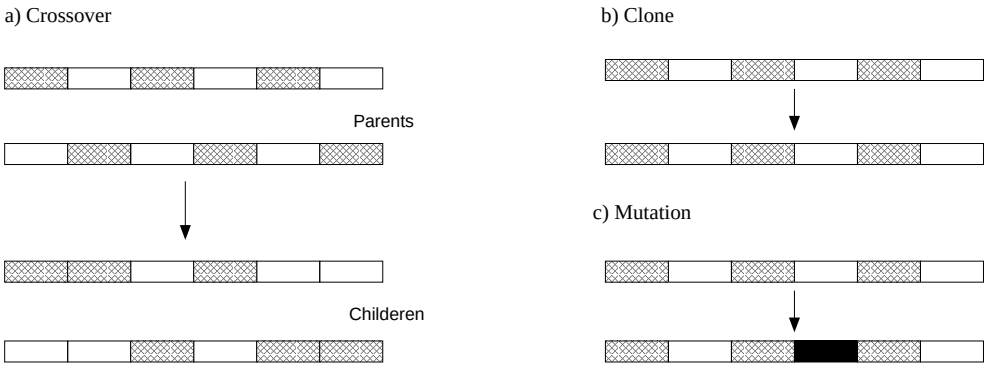


Fig. 3. Possible fates of solutions during the evolution process. a) Crossover, two parents producing two children. b) Cloning, an individual is passed directly to the next generation. c) Mutation, a single gene (kinetic rate) is changed.

our approach a random roulette wheel based technique [9] is used to probabilistically select individuals which are then subject to one of three fates [9]: *crossover*; *cloning*; and *culling*. During crossover two individuals are selected and these then "breed" to produce two new children by randomly selecting genes from the two parents (see Figure 3.a)). These children then pass into the new population. Individuals may also be selected to progress unchanged to the next population and we refer to this as cloning (see Figure 3.b)). Any individual not selected for cloning or crossover has effectively been culled and will not appear in the new population. The number of individuals cloned is governed by the constant *CLONES* in the algorithm above. Note that using this approach the fittest individual may not survive and conversely, the least fit may. This in fact is an important point since it helps prevent the population getting stuck in a local minima.

Once the makeup of the next generation is decided the mutation phase begins. Here individuals are selected randomly to be subjected to a single random gene mutation, as shown in Figure 3.c). The number of mutations applied is controlled by the threshold constant *MUTES* which sets the probability of performing a mutation. These mutations introduce new genes into the population, giving the potential for more varied solutions to be considered. After the mutation phase is completed a new population emerges. The whole process above is then repeatedly applied until a pre-defined number *MAX* of populations have been generated.

3.3 Assumptions for Kinetic Parameterisation

To allow a starting point for the automatic parameterisation of stochastic networks, some additional assumptions were necessary. Along with the assumptions made in [31], two additional assumptions were made. Firstly, it was assumed that the starting concentrations of the system were well understood. This assumption reduces the search space of the problem, allowing the implementation of this experimental process. We hope to implement the

auto scanning of these parameters in future work. Note this assumption is not necessarily unrealistic in our biological setting where many of the molecular amounts considered tend to a steady state [31]. When applied to gene regulatory networks (as we do in Section 4) there are certain places, such as those representing physical DNA molecules, where the initial amounts are practically fixed. Our second assumption was that the network topology was correct. Again this assumption is reasonable since biologists normally have a good idea of the network topology underlying their biological system of interest. They are able to perform a variety of laboratory experiments which provide qualitative information on the system structure and behaviour, such as those described in [31].

4 Case Study: Stress Pathway in *E.coli*

In this section we demonstrate and evaluate our parameter fitting tool on a bench mark case study of a gene regulatory network, namely the *Escherichia coli* σ-32 stress response pathway [31].

4.1 The σ-32 Stress Response Pathway

A model of the *Escherichia coli* σ-32 stress response pathway was selected to validate and test our approach. This model has been published in detail previously by Srivastava *et al* [31] providing the basis for a useful bench mark case study. Briefly, the σ-32 stress response system of *E.coli* allows the organism to respond to situations that may jeopardise the organisms survival. The responses to stress generally involves the coordinated regulation of genes whose products have functions such as protecting essential cellular machinery from damaging environmental factors, facilitating the use of alternate energy sources and inducing the organism to move away from the source of the stress. The coordination of this response is centred around a type of protein called a sigma factor, in this case σ-32. The idea is that increased levels σ-32 are able to switch on around 30 genes that encode the production of other proteins that alleviate stress, termed σ-32 induced proteins. Sets of genes that are co-regulated in this fashion are termed *regulons*. Free σ-32 protein can combine with RNA polymerase (to form Eσ-32) to induce the σ-32 regulon. The level of σ-32 in the cell is modulated in response to an input to the pathway which senses stressful conditions and induces the production of σ-32 from its parent gene (*rpoH*). The constant accumulation of σ-32 is prevented by a protein degradation pathway which is an important regulatory mechanism in this pathway. In *E.coli* this degradation of σ-32 occurs via a protein produced from the *ftsH* gene. However, in order to be degraded rapidly σ-32 must be complexed with the protein products of other genes, which are themselves members of the σ-32 regulon. In this study we refer to this complex as the J-Comp-σ-32 complex.

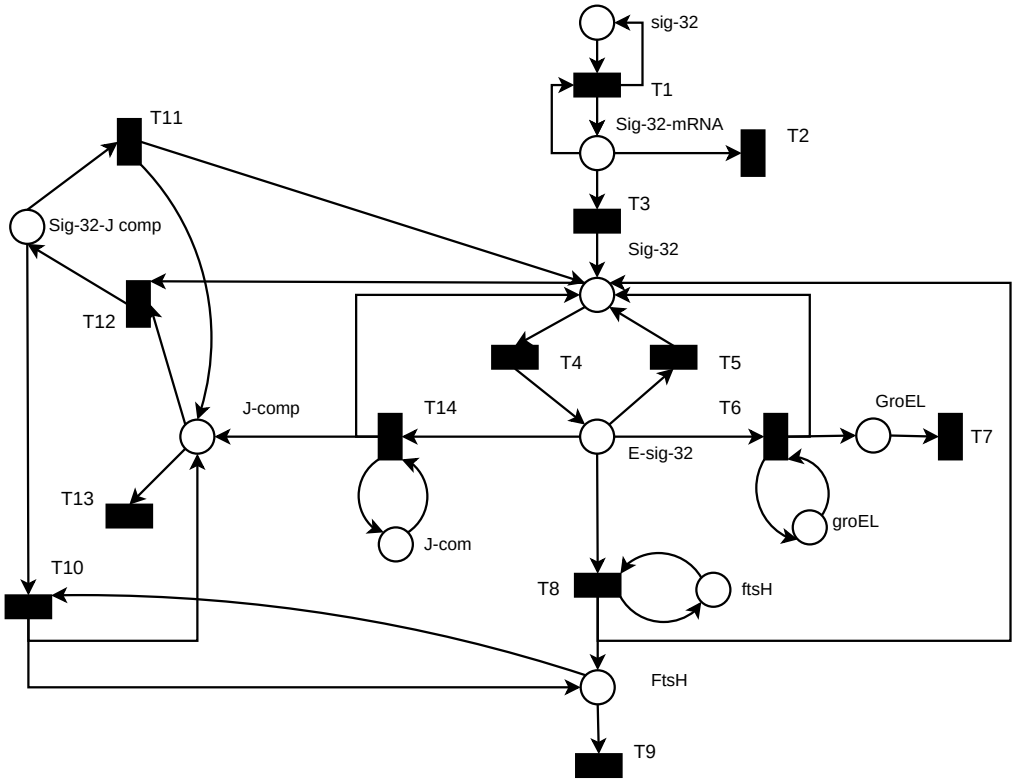
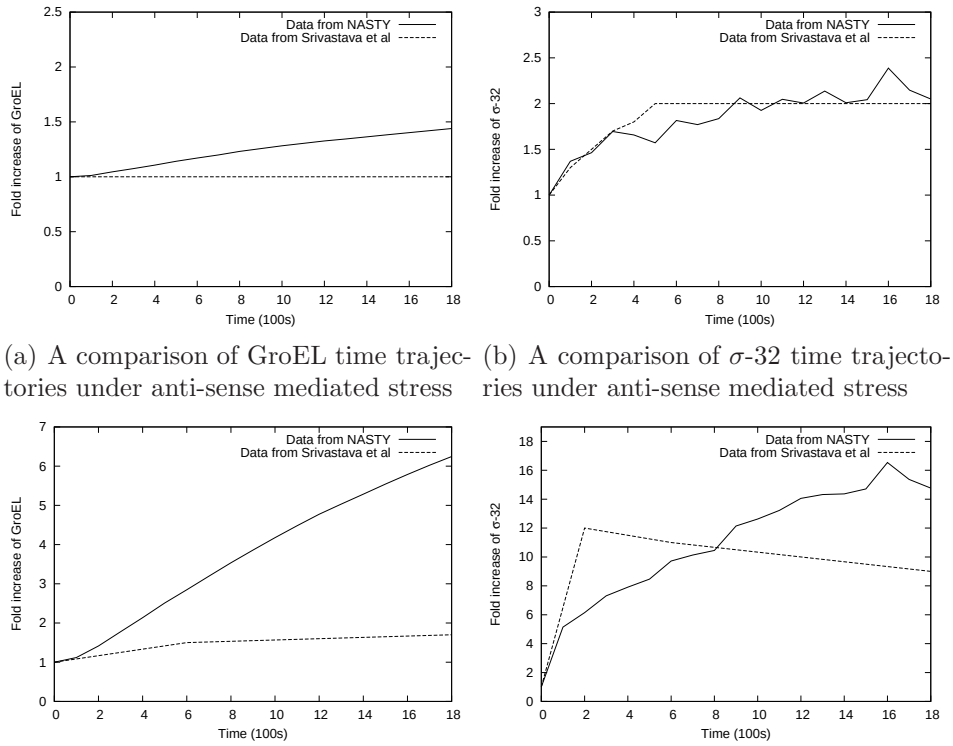


Fig. 4. A Petri net representation of the σ-32 stress response pathway of *E.coli*, from Srivastava *et al* [31]

4.2 Petri Net Based Modelling of the σ-32 Stress Response Pathway

The Petri net model of the above regulatory network used in this paper is depicted in Figure 4. The external input to the model is provided by a mechanism that detects stressful conditions, although for clarity, the details of the complex signal transduction systems that act as a sensor mechanisms for stress have been abstracted. Essentially, detection of a stressful condition is assumed to alter the rate of the transition $T3$ increasing the production of the σ-32 protein (labelled Sig-32 in Figure 4) through the translation and transcription of the σ-32 gene (labelled sig-32 in Figure 4). The model includes transitions representing the interaction of σ-32 with RNA polymerase ($T4$ and $T5$), induction of the protein degrading enzyme ftsH ($T8$), and production of the J-complex ($T14$) and association of the J-complex with σ-32 ($T12$). In our model J-Comp-σ-32 protein is degraded via transition $T10$. The protein GroEL is a known member of the σ-32 regulon which is not involved in the direct regulation of σ-32. We have included it in our model since the level of GroEL can be used to provide an accurate indication of the induction of the σ-32 stress response regulon.

This model of the regulation of the σ-32 regulon has been employed since it has been shown to successfully replicate the behaviour of the biological system [31], as determined by laboratory based studies, and it is of sufficient size and



(a) A comparison of GroEL time trajectories under anti-sense mediated stress

(b) A comparison of σ-32 time trajectories under anti-sense mediated stress

(c) A comparison of GroEL time trajectories under ethanol stress

(d) A comparison of σ-32 time trajectories under ethanol stress

Fig. 5. The fold increase of token numbers for GroEL and σ-32 under stress conditions compared to zero stress. Results from NASTY and Srivastava [31]

complexity to validate our genetic algorithm. However, in our hands some modifications to the model were required in order to supplement the information given in [31]. The initial concentrations of entities in the model were not specifically listed, and thus these were estimated by hand from indications given in the paper. In addition, we assume that the DNA and mRNA molecules that encode σ-32 are outputs of the translation, $T1$ and transcription, $T3$ reactions respectively. With these modifications we were able to recreate the behaviour of the model as described by Srivastava [31].

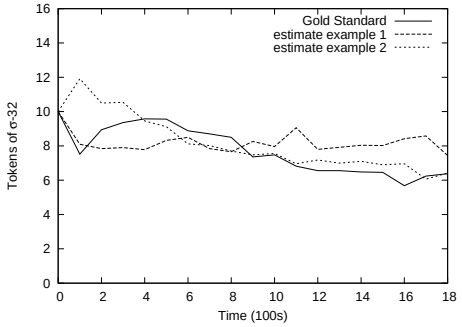
Three particular experiments were selected from those described in [31] to illustrate our approach. These three experiments involved altering the transcription rate $T3$. Under no stress $T3 = 0.007$, under anti-sense mediated ethanol stress $T3 = 0.02$ and finally under ethanol stress $T3 = 0.15$. Both σ-32 and GroEL were monitored under these conditions. The protein σ-32 was measured to provide a indication of level of the stress inherent in the pathway. GroEL is a product of the σ-32 pathway, but is not directly involved in σ-32 regulation hence the level of GroEL gives an indication of whether the σ-32 regulon has been induced.

Initially a zero stress situation was simulated in the NASTY tool. The model was simulated 50 times, and the average value was used to compare the fold increase obtained under stress conditions. The results from our simula-

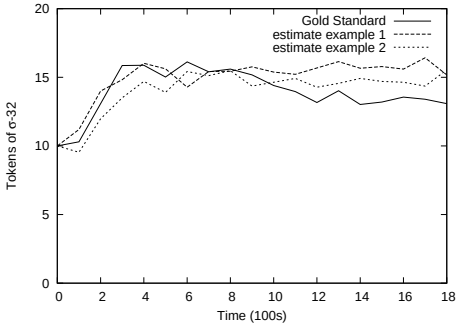
tion under these conditions appear to match well with the results from [31] (data not shown). Next, the model was simulated with the translation rate $T3$ adjusted to the levels for anti-sense mediated stress and ethanol stress situations. These results were compared with the results in Srivastava [31], and are shown in Figure 5. In general, the correlation between the results obtained from NASTY and those from Srivastava is very good. The small disparities apparent are assumed to be due to the lack of clear initial amounts of molecules detailed in [31]. However, these disparities do not impact on subsequent studies with respect to parameterisation of the network, since the simulation results for zero stress, anti-sense mediated stress and ethanol stress using our model simulated in NASTY were used as the "gold-standard" from which the fitness of solutions were evaluated.

4.3 Performance with One Time Trajectory

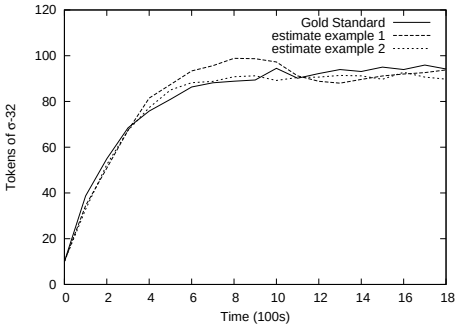
We carried out investigations to determine the performance of the genetic algorithm using the gold standard time trajectories obtained from the NASTY tool. Initially, experiments were carried out using the time trajectory of a single protein σ-32, as a "gold-standard" to evaluate the fitness of solutions in



(a) σ-32 under zero stress, p1=0.441, p2=0.001

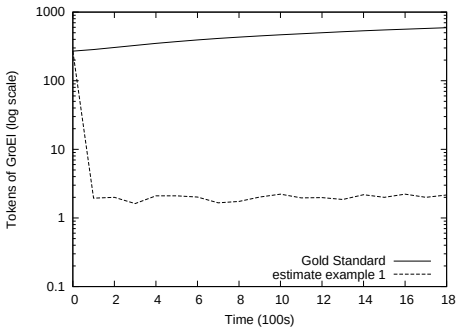


(b) σ-32 under anti-sense mediated stress, p1=0.001, p2=0.001

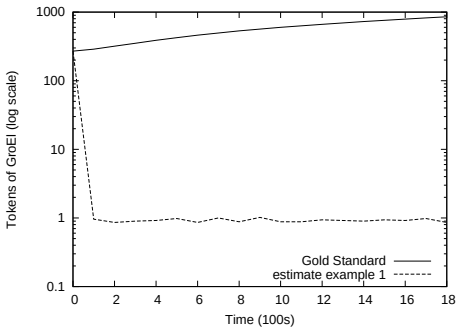


(c) σ-32 under ethanol stress, p1=0.001, p2=0.001

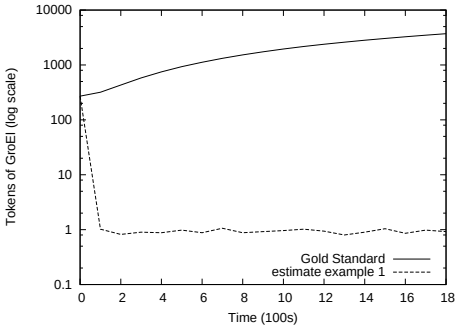
Fig. 6. A comparison of selected results obtained from the genetic algorithm, using a single protein's time trajectories (σ-32) to evaluate fitness, compared against the gold standard time trajectories (Pearson correlation p values included).



(a) GroEL under no stress, p=0.08



(b) GroEL under anti-sense mediated stress, p=0.959



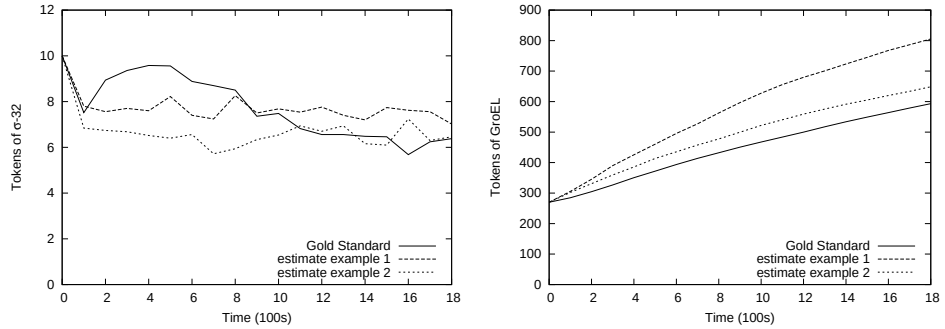
(c) GroEL under ethanol stress, p=0.87

Fig. 7. A comparison of selected results obtained from the genetic algorithm, using a single protein's time trajectories (σ-32) to evaluate fitness, compared against the gold standard time trajectories (Pearson correlation p values included).

the genetic algorithm. The results for these experiments are shown in Figure 6. When compared to the σ-32 gold standards, results obtained from the Genetic algorithm displayed a highly significant similarity. To determine how well these results matched the behaviour of the system more globally, the solutions obtained from the genetic algorithm were compared to the "gold-standard" GroEL time trajectories. These results are shown in Figure 7. Results from these comparisons displayed a poor match between the genetic algorithms solutions and the "gold–standards". This was an interesting and surprising result as the genetic algorithm had found solutions with extremely high fitness, almost maximal, for σ-32, whilst being extremely inaccurate in predicting the behaviour of GroEL, another part of the same system. This indicates that our fitness function may need to be refined and we now consider this.

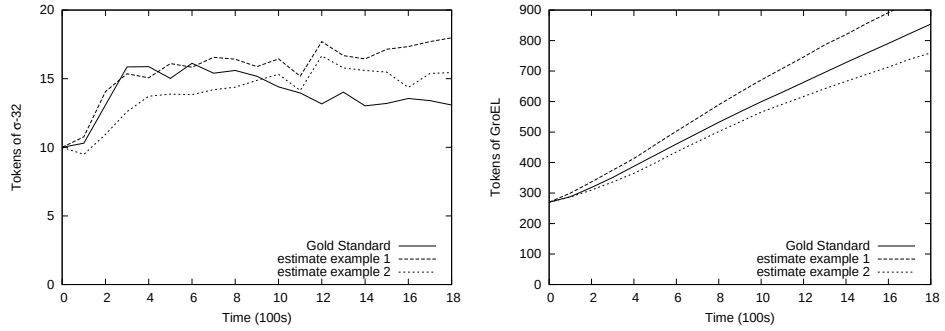
4.4 Performance with Two Time Trajectories

To resolve the situation above, the genetic algorithm was programmed to utilise the "gold-standard" time trajectories for both σ-32 and GroEL. A second set of experiments was then carried out using a similar approach of altering the transcription rate T3 as described above. The estimated trajectories



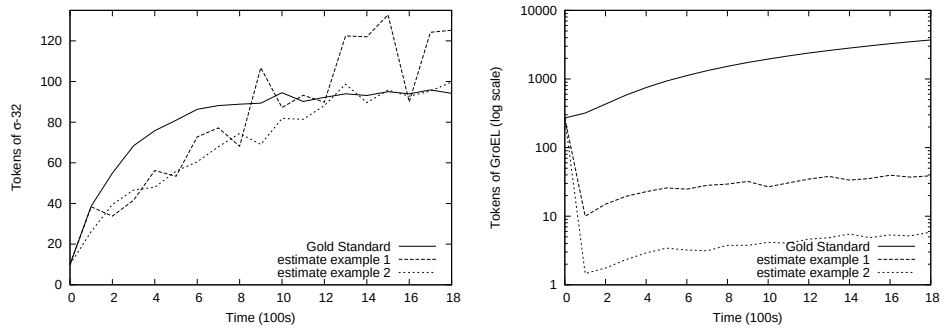
(a) σ-32 under no stress, p1=0.02, p2=0.27

(b) GroEL under no stress, p1=0.001, p2=0.001



(c) σ-32 under anti-sense mediated stress, p1=0.001, p2=0.05

(d) GroEL under anti-sense mediated stress, p1=0.001, p2=0.001



(e) σ-32 under ethanol stress, p1=0.001, p2=0.001

(f) GroEL under ethanol stress, p1=0.193, p2=0.4

Fig. 8. Solutions obtained from the genetic algorithm using the time trajectories of two proteins (σ-32 and GroEL) to evaluate fitness, compared with the gold-standards (Pearson correlation p values included)

obtained by the genetic algorithm in this case are shown in Figure 8. The correlation of the estimated trajectories with the "gold-standard" for both σ-32 and GroEL were very high in all three stress situations. The exception to this case was GroEL under ethanol stress. Here the genetic algorithm estimated a suitable solution with regard to σ-32 in terms of both quantitative and qualitative behaviour. However the solution with regard to GroEL matched qualitatively but was not quantitatively accurate.

5 Discussion

Stochastic simulations are becoming an increasingly important tool in Systems Biology as they are able to capture the fine grain behaviour and randomness of outcome of biological networks missed by deterministic techniques [19,31]. In this paper we have considered developing tools to help enable the stochastic simulation of biological networks. Using the framework of stochastic Petri nets we have attempted to address two of the main hurdles faced by biologists when using stochastic simulation: the high computational cost of performing simulations; and the lack of kinetic parameter data available.

We began by presenting a new stochastic Petri net simulator NASTY, which was specifically tailored to biological simulations (i.e. used mass action kinetics by default and was compatible with the biological markup language SBML). This simulator addressed the cost of performing stochastic simulations by employing a distributed job scheduler which allowed simulations to be carried out efficiently over a large cluster of machines. We then considered developing a parameter fitting tool based on a genetic algorithm [11] implementation. The resulting heuristic tool combines the inherent parallelism in genetic algorithms with NASTY's distributed processing power to ensure the large number of multiple stochastic simulations required can be efficiently performed.

We illustrated our parameter fitting tool by presenting a case study in which the kinetic rates were derived for a stochastic Petri net model of the stress response pathway in the bacterium *Escherichia.coli* (*E.coli*) [31]. The initial results from our case study were promising, though they indicated that more work is needed to refine our techniques. Suitable kinetic parameters were found for a small part of the system when using only a single protein as the "gold–standard" time trajectory to evaluate the fitness of solutions. However, these solutions were not suitable for the larger system as a whole, e.g. the associated time trajectory for GroEL proved to be inaccurate. To address this problem we adjusted the genetic algorithm to use the time trajectories for two proteins as the "gold-standard" for parameterising the network. Here the genetic algorithm tool derived more generic rates for zero stress, anti-sense mediated stress and ethanol stress which matched both the time trajectories for proteins σ-32 and GroEL, though the GroEL time trajectory for ethanol stress matched only qualitatively.

Our investigation has shown the potential for using heuristic techniques for addressing the difficult problem of parameter fitting in stochastic biological models. More work is now needed to refine these techniques to make them of real practical use to experimental biologists. We are currently investigating improving the "breeding" and "fitness" components of the tools. We are also considering the use of qualitative Petri net models to provide important insights and constraints for the parameter fitting process. Other approaches, such as applying the *tau leap* method [8] to reduce simulation times by ap-

proximating solutions are also likely to prove valuable here.

Acknowledgments. We are very grateful to M. Pocock, M. Koutny and C. Harwood for many useful discussions and comments on this work. We would also like to thank BBSRC for providing financial support for O. J. Shaw during this work via grant 02/B1/X/08298.

References

[1] A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected *E. coli* cells. *Genetics*, 149:1633–1648, 1998.

[2] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber. The Petri net markup language: Concepts, technology, and tools. In *Proceedings of the 24th International Conference on Applications and Theory of Petri Nets (ICATPN 2003), Eindhoven, The Netherlands, June 23-27, 2003 — Volume 2679 of Lecture Notes in Computer Science / Wil M. P. van der Aalst and Eike Best (Eds.)*, pages 483–505. Springer-Verlag, June 2003.

[3] D. Endy and R. Brent. Modelling cellular behaviour. *Nature*, 409(6818):391–395, 2001.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, 1979.

[5] M.A. Gibson. *Computational Methods for Stochastic Biological Systems*. PhD thesis, California Inst. Technology, 2000.

[6] M.A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry*, 104:1876–1889, 2000.

[7] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[8] D.T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[9] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Pub. Co., 1989.

[10] P.J.E. Goss and J. Peccoud. Quantitative modelling of stochastic systems in molecular biology by using stochastic Petri nets. *Proceedings of the National Academy of Sciences of the United States of America*, 95(12):6750–6755, 1998.

[11] J.H. Holland. *Adaptation in natural and artificial systems*. 1975.

[12] M. Hucka, A. Finney, et al. The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[13] R. Iyengar, W. Wolkenhauer, W. Kolch, K. Cho, and U. Kilngmuller. Opening editorial. *Systems Biology*, 1(1):1, 2004.

[14] T.R. Kiehl, R.M. Mattheyses, and M.K. Simmons. Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316–322, 2004.

[15] S. Kirkpatrick, C. D. Gelatt jnr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[16] F. Kunst, N. Ogasawara, et al. The complete genome sequence of the gram-positive bacterium bacillus subtilis. *Nature*, pages 249–256, 1997.

[17] M.A. Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli, and G. Franceschinis. An introduction to generalized stochastic Petri nets. *Microelectronics and Reliability*, 31(4):699–725, 1991.

[18] W. Marwan. Theory of time-resolved somatic complementation and its use to explore the sporulation control network in physarum polycephalum. *Genetics*, 164:105–15, 2003.

[19] H.H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, 94(3):814–819, 1997.

[20] H.H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269(5224):650–656, 1995.

[21] H.H. McAdams and L. Shapiro. A bacterial cell-cycle regulatory network operating in time and space. *Science*, 301(5641):1874–1877, 2003.

[22] P. Mendes and D.B. Kell. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 14(10):869–883, 1998.

[23] I. Mitrani. *Probablisitc Modelling.* Cambridge University Press, 1998.

[24] T. Murata. Petri nets - properties, analysis and applications. *Proceedings of the Ieee*, 77(4):541–580, 1989.

[25] J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1981.

[26] Petri Nets World. Petri nets world home page, http://www.daimi.au.dk/ petrinet/, 2004.

[27] J.W. Pinney, D.R. Westhead, and G.A. McConkey. Petri net representations in systems biology. *Biochem Soc Trans*, 31(6):1513–5, 2003.

[28] V. N. Reddy, M. N. Liebman, and M. L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Computers in Biology and Medicine*, 26(1):9–24, 1996.

[29] SBML. The SBML homepage, at http://www.sbml.org, 2004.

[30] O.J. Shaw, A. Koelmans, L.J. Steggles, and A. Wipat. Applying Petri nets to systems biology using XML technology. *In Proceedings of the Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets, Satellite event of the 25th International Conference on Application and Theory of Petri Nets, Bologna, Italy*, pages 11–25, 2004.

[31] R. Srivastava, M.S. Peterson, and W.E. Bentley. Stochastic kinetic analysis of the escherichia coli stress circuit using sigma(32)-targeted antisense. *Biotechnology and Bioengineering*, 75(1):120–129, 2001.

[32] R. Srivastava, L. You, J. Summers, and J. Yin. Stochastic vs. deterministic modeling of intracellular viral kinetics. *J Theor Biol.*, 218(3):309–321, 2002.

[33] S. Zolfraghari and M. Liang. Comparative study of simulated annealing, genetic algorithms and tabu search for solving comprehensive machine-grouping problems. *International Journal of Production Research*, 40:2141–2158, 2002.