Parallelized Parameter Estimation of Biological Pathway Models *

R Ramanathan[1], Yan Zhang[1], Jun Zhou[1], Weng-Fai Wong[1], and P S Thiagarajan[2]

[1] Department of Computer Science, National University of Singapore, Singapore
{ramanathan, zhangyan, zhoujun, wongwf}@comp.nus.edu.sg
[2] Laboratory of Systems Pharmacology, Harvard Medical School, Boston, USA
psthiagu@gmail.com

Abstract. We develop a GPUs based technique to analyze bio-pathway models consisting of systems of ordinary differential equations (ODEs). A key component in our technique is an online procedure for verifying whether a numerically generated trajectory of a model satisfies a property expressed in bounded linear temporal logic. Using this procedure, we construct a statistical model checking algorithm which exploits the massive parallelism offered by GPUs while respecting the severe constraints imposed by their memory hierarchy and the hardware execution model. To demonstrate the computational power of our method, we use it to solve the parameter estimation problem for bio-pathway models. With three realistic benchmarks, we show that the proposed technique is computationally efficient and scales well with the number of GPU units deployed. Since both the verification framework and the computational platform are generic, our scheme can be used to solve a variety of analysis problems for models consisting of large systems of ODEs.

Keywords: bio-pathway models, ordinary differential equations, graphics processing units, BLTL, statistical model checking, parameter estimation

1 Introduction

We advocate a generic platform-aware technique to study the dynamics of large bio-pathways models. Specifically we focus on a well-established formalism, namely, a system of ordinary differential equations (ODEs) to model the dynamics of the pathway models. For such systems we implement an analysis method on a multi-core platform consisting of a pool of general-purpose graphical processing units (GPGPUs or simply GPUs).

A system of ODEs together with a (initial) set of values for the initial concentrations and the rate constants was formulated by Palaniappan *et al.* [27] as a model of a biochemical network that takes into account cell-cell variability in

* This research was supported by the Singapore MOE grant MOE2013-T2-2-033.

a population. These ODEs systems will be high dimensional with no closed form solutions. To get around this a probabilistic approximation technique accompanied by a statistical model checking procedure was developed by assuming a probability distribution over the set of initial values. Since the present paper is essentially an adaptation of this technique for an efficient GPU based implementation we shall begin with a brief description of the theoretical underpinnings of this technique.

Variations in the initial concentrations of species and kinetic rate constants across a cell population are typical. To cater for these variations, we assume an initial probability distribution over the range of initial values and rate constants. We then suppose that the states of the system are observed at only the discrete time points $\{0, 1, 2, \ldots\}$ (with the unit of time being chosen suitably). In fact we assume that the behavior of the system is of interest only upto a fixed maximum time point t_K. The constant K is chosen based on the application at hand. For instance, in the parameter estimation problem the last time point for which experimental data is available will be used to determine K. We choose bounded linear time temporal logic (BLTL) to specify dynamical properties and the BLTL specifications of interest are designed to respect this time bound. The atomic propositions used in the specifications will assert that the current value of a continuous variable lies in a given interval (with rational end points). This will in effect discretize the value space of the variables as well. We then impose the condition that the vector field associated with the ODEs systems is C^1-continuous which is a justifiable assumption in the context of biochemical reaction networks. As a consequence one can construct a σ-algebra over TRJ^K, the set of trajectories of length K and define a probability measure over this space. However due to the determinacy of the dynamics the probability measure over the space of trajectories will induce a probability measure over the space of measurable subsets of the set of initial states. Moreover the set of initial states of the trajectories that satisfy a BLTL formula will be measurable in this σ-algebra and hence can be assigned a probability value. As a result one has to merely sample the initial states and the parameter values according to the given initial distributions to carry out the sequential hypothesis testing procedure (or any statistical model checking procedure based on BLTL specifications). As an aside, this idea breaks down in the case of hybrid dynamics due to mode switchings and one needs to develop new technical machinery to induce a probability measure over the space of trajectories. A preliminary version of this extension is presented in a paper in the present volume [12].

The theory sketched above is used in [27] to carry out parameter estimation as follows. A conjunction of BLTL formulas describe the available experimental time-course data as well as known qualitative properties. One then deploys the statistical model checking procedure to evaluate the goodness of the current estimates for unknown parameter values. With the help of an evolutionary search strategy one then searches through the parameter space to obtain a good set of parameter values. The estimated values are then validated using test data that

was not made available to the estimation procedure. A similar technique is also applied to do sensitivity analysis [27].

For high dimensional systems with many unknown parameters, one will have to call upon the SMC procedure many times and for each such call one will have to generate sufficiently many trajectories to ensure the termination of the SMC procedure. Consequently the computational cost induced by the repeated executions of the SMC procedure can be quite high. This is the motivation for the GPUs based implementation of the above mentioned parameter estimation procedure developed in this paper.

Obviously one can numerically generate trajectories in parallel on a GPU. Thus it is tempting to take for granted an easy parallel implementation and a corresponding increase in performance. This is however not the case. The memory hierarchy of a GPU and its single-instruction multiple-thread (SIMT) organization of its arithmetic units constitute severe constraints. A naïve implementation will often perform no better than (and in some cases worse than!) a sequential implementation. GPUs are however an attractive candidate since they are available off-the-shelf and can offer performance that is comparable to the more-expensive and less-available multi-core platforms. Furthermore, it is possible to form large pools of GPUs in a scalable and cost effective way using cloud services. Therefore, the effort required to overcome the architectural constraints of GPUs may well be worth it and this is the hypothesis we pursue here.

In simplified terms, the iterative parameter estimation procedure consists of:

(i) Encode the experimental data and known qualitative trends as a BLTL formula φ (as detailed in Section 3).
(ii) Fix the required confidence level and the false positives and negatives rates w.r.t. which one wishes to verify φ.
(iii) Guess a current value for each unknown parameter.
(iv) Evaluate the goodness of these estimated parameters by repeatedly generating trajectories till the statistical test associated with the SMC procedure terminates.
(v) If the outcome is yes then the current estimate is a good one. If not, guess a new set of values using the evolutionary search strategy and iterate.

Thus it is step (iv) which is ripe for parallelization. However just generating a numerical trajectory is not enough. One must evaluate if it satisfies φ which is of course easy to do. However only a small amount of memory will be available in the vicinity of a GPU core. Hence the generated trajectories need to be sent up through a number of levels in the memory hierarchy, each of which is significantly slower than the previous one. This will all but eliminate the performance gains obtained by generating the trajectories in parallel. Hence one must verify whether a generated trajectory satisfies φ on the fly without having to store the whole trajectory. Again this is not difficult to do though one must minimize the amount of intermediate data (typically Boolean combinations of the subformulas of φ that still need to be satisfied) to be kept track of. However the obvious online procedures will involve branching that is based on the current requirements and this will clash with the hardware parallelism available

in GPUs. At the level of a single core, groups of parallel threads called *warps* are scheduled to run the compiled code, which at each step, execute the same machine instruction in a lock-step fashion. This is the heart of GPU's execution model. If two threads in a warp take different branches, the warp will have to be executed twice, once for each branch. This so called *branch divergence* causes severe performance degradation [21]. To avoid this, we construct a *deterministic* automaton-based online model checking technique. This is the main technical construction in the paper. It turns out that it is better to store the automaton (as a look-up table) in the intermediate storage shared by the cores and hence we also implement a standard latency hiding technique to mitigate the data transfer delays between this shared store and the global store (using which the rest of the parameter estimation steps are carried out) during model checking.

We have evaluated the performance of our GPU based parameter estimation procedure on a number of bio-pathway models drawn from the biomodels data base. Specifically we have applied our method to the EGF-NGF pathway, the segmentation clock pathway and the thrombin-dependent MLC-phosphorylation pathway. Our results show that one can hope to achieve significant performance gains especially by deploying a pool of GPUs. The present implementation can be further optimized and a similar strategy can be followed to solve the sensitivity analysis problem. As pointed out in the concluding section we also feel that other analysis problems concerning bio-pathways can be tackled using the approach developed in this paper.

1.1 Related Work

The problem of efficiently generating numerical trajectories on GPUs for large systems of ODEs has been studied in literature [22,34,25]. In our implementation, we adopt Liu *et al.*'s approach that encompasses a heterogeneous group of GPU threads where the memory-access threads and the trajectory computing threads are separated into different warps to achieve latency hiding and hence scalability.

Efficient methods for model checking probabilistic systems have been studied [8,20,29,7,32]. The statistical model checking (SMC) approach initiated by Younes and Simmons [33] based on the sequential probability ratio test proposed by Wald [31] has turned out to be a fruitful one and is adopted here. SMC usually involves checking whether an individual trace satisfies a given temporal specification. When the specification is a BLTL formula, this is known as *BLTL path checking*. Kuhtz and Finkbeiner show that the path checking problem can be parallelized by unrolling the BLTL formulas into Boolean circuits [18]. Barre *et al.* adopt the MapReduce framework [10] to verify a single large trace using distributed computing [3]. However, it is not clear how these methods can be implemented on a GPU-based platform.

On the other hand, Barnat *et al.* take an automata-theoretic approach to parallel model checking of a restricted class of multi-affine ODE systems [1,2]. The ODE model dynamics is first approximated as a rectangular abstraction automaton and a given LTL property is translated into a Büchi automaton that represents its negation. A parallel model checker then looks for an accepting

cycle in the product automaton by symbolically exploring the state space. But this approach tends to over-approximate the model dynamics. Oshima *et al.* present a FPGA-based framework for the checking of BLTL specifications with applications on partial differential equations [26]. Their method also involves a Büchi automaton construction but requires a large set of trajectories to be stored in the hardware before a property can be verified. In contrast our online method is based on GPUs, which we believe are more accessible and scalable. Further our focus is on ODE systems that arise in bio-pathway models.

In recent years, statistical model checking has become a building block to solve complex problems. David *et al.* apply SMC using analysis of variance (ANOVA) to find the optimal set of parameters of a network of stochastic hybrid automata [9]. Jha *et al.* show how the parameter synthesis problem for stochastic systems can be approached using statistical model checking [16]. In this paper, we focus on efficient parallelization techniques for traditional analysis tasks based on SMC, especially parameter estimation [5]. In particular, we realize the approach proposed by Palaniappan *et al.* [27] on a GPU-based platform. The key new ingredient is a novel *deterministic* online BLTL path checking procedure that fits in with the requirements of the GPU platform.

The paper is organized as follows. First Section 2 introduces the ODE dynamics and the syntax and semantics of BLTL. Section 3 formulates the online verification problem, and describes our automata-theoretic solution to this problem. In the subsequent Section 4, we develop the GPU based solution to the parameter estimation problem with the online verification procedure serving as the kernel. In the subsequent Section 5 we perform a number of performance case studies, and in Section 6 we summarize and point to future research directions.

2 Background

2.1 ODEs and Trajectories

In the present setting, a bio-chemical network is modeled as a system of ODEs. Assume that there are n molecular species $\{x_1, x_2, \ldots, x_n\}$ involved in the network. For each x_i, an equation of the form $\frac{dx_i}{dt} = f_i(\mathbf{x}, \Theta_i)$ describes the kinetics of the reactions that produce and consume x_i where $\mathbf{x}$ is the concentrations of the molecular species taking part in the reactions. Θ_i consists of the rate constants governing the reaction. Each x_i is a real-valued function of time $t \in \mathbb{R}$. We assume in this section that all rate constants are known. In Section 4, it will become clear how unknown rate constants are handled while solving the parameter estimation problem.

To capture the cell-to-cell variability regarding the initial states, we define for each variable x_i an interval $[L_i^{init}, U_i^{init}]$ with $L_i^{init} < U_i^{init}$. The actual value of the initial concentration of x_i is assumed to fall in this interval. We set $INIT = \prod_i [L_i^{init}, U_i^{init}]$. In what follows, we let $\mathbf{v}$ to range over $\mathbb{R}^n$.

We represent our system of ODEs in the vector form, $\frac{d\mathbf{x}}{dt} = F(\mathbf{x}, \Theta)$ with $F_i(\mathbf{x}, \Theta) := f_i$. In the setting of bio-chemical networks, the expressions in f_i will

model kinetic laws such as mass-action and Michaelis-Menten's [17]. Moreover, the concentration levels of the various species will be bounded and the behavior of the system will be of interest only up to a finite time horizon. Hence we assume in this paper that f_i is Lipschitz-continuous for each i . As a result, for each $\mathbf{v} \in \mathit{INIT}$ the system of ODEs will have a unique solution $\mathbf{X}_{\mathbf{v}}(t)$ [15]. We are also guaranteed that $\mathbf{X}_{\mathbf{v}}(t)$ is a C^0-function (i.e., continuous function) [15] and hence measurable.

For convenience, we define the flow $\Phi : \mathbb{R}_+ \times \mathbf{V} \to \mathbf{V}$ for arbitrary initial vectors $\mathbf{v}$ as $\mathbf{X}_{\mathbf{v}}(t)$. Intuitively, $\Phi(t, \mathbf{v})$ is the state reached under the ODE dynamics if the system starts at $\mathbf{v}$ at time 0. We work with $\Phi_t : \mathbf{V} \to \mathbf{V}$ where $\Phi_t(\mathbf{v}) = \Phi(t, \mathbf{v})$ for every t and every $\mathbf{v} \in \mathbf{V}$. Again, Φ_t is guaranteed to be a C^0-function (in fact $1 - to - 1$) and Φ_t^{-1} will also be a C^0-function.

In our applications, given the nature of the experimental data, the states of the system will be observed only at discrete time points and only within a finite time horizon. Hence by choosing a suitable unit of time we will assume that the states of the systems are observed at the time points $0, 1, \ldots$. A *trajectory* is a finite sequence $\tau = \mathbf{v}_0\mathbf{v}_1 \ldots \mathbf{v}_k$ such that $\mathbf{v}_0 \in \mathit{INIT}$ and $\Phi_1(\mathbf{v}_j) = \mathbf{v}_{j+1}$ for $0 \leq j < k$. We let TRJ denote the set of finite trajectories which model the dynamics of the ODEs system.

2.2 Time-bounded Linear Temporal Logic

In order to encode the dynamical properties of TRJ, we will use formulas in bounded time linear temporal logic *(BLTL)*. An atomic proposition is of the form $(x_i \geq v)$ or $(x_i \leq v)$ with $v \in \mathbb{R}$. The proposition $(x_i \geq v)$ is interpreted as "the current concentration level of x_i is greater than or equal to v". A finite set of atomic propositions, AP, is assumed to be given for a bio-pathway model.

A BLTL formula is defined as follows. First, every atomic proposition, as well as the Boolean constants *true* and *false*, is a BLTL formula. If ψ_1 and ψ_2 are BLTL formulas, $\neg\psi$ and $\psi_1 \vee \psi_2$ are BLTL formulas. Also, if ψ is a BLTL formula, so is $X\psi$. Finally, if ψ_1 and ψ_2 are BLTL formulas, and t is a positive integer then $\psi_1 U^{\leq t} \psi_2$ is a BLTL formula. The derived propositional connectives $\wedge, \supset$, etc. and the temporal operators $G^{\leq t}$ and $F^{\leq t}$ are defined in the usual way.

Let $\tau = \mathbf{v}_0\mathbf{v}_1 \ldots \mathbf{v}_k$ be a trajectory and $0 \leq j \leq k$. The semantic relation $\tau, j \models \psi$ is defined as follows.

- $\tau, j \models (x_i \geq v)$ iff $\mathbf{v}_j(i) \geq v$. The clause $\tau, j \models (x_i \leq v)$ is defined similarly.
- $\neg$ and $\vee$ are interpreted in the usual way.
- $\tau, j \models X\psi$ iff $j < k$ and $\tau, j + 1 \models \psi$.
- $\tau, j \models \psi_1 U^{\leq t} \psi_2$ iff there exists t' such that $t' \leq t$ and $j + t' \leq k$ and $\tau, j + t' \models \psi_2$. Further, $\tau, j + t'' \models \psi_1$ for every $0 \leq t'' < t'$.

We say that τ is a *model* of ψ if $\tau, 0 \models \psi$.

The rationale of choosing BLTL instead of a more sophisticated logic is two-fold. First, relevant properties of bio-pathway models, especially in the context of parameter estimation and sensitivity analysis are linear time properties defined

over a bounded time horizon. Second, BLTL has enough expressive power to characterize properties relating to bio-pathway models while being a very simple temporal logic to work with. Hence we choose BLTL over other commonly-used formalisms, such as continuous stochastic logic and metric temporal logic.

3 Online Model Checking Procedure

The problem of BLTL path checking involves determining whether a BLTL formula is satisfied by a trajectory. According to the BLTL semantics, it is easy to see that the truth value of a BLTL formula can be decided by trajectories with finite length. Online BLTL path checking requires only the current valuation of the atomic propositions as input. At each step, it evaluates the BLTL formula under the current valuation and generates a new formula that represents the "obligation" in the following step. The procedure terminates when the formula under consideration becomes either true or false, indicating a satisfaction or falsification of the original formula.

Such an algorithm can be easily implemented on CPUs. On the other hand, to achieve a good performance on GPUs one must address the problem of branch divergence, which occurs when two GPU threads choose different code segments under the evaluation of a condition as illustrated in the following example.

Example 1 (Branch Divergence). Consider BLTL formula $\phi = F^{\leq 8} G^{\leq 5} p$, where p is an atomic proposition. Expanding ϕ, we get $\phi = \left(p \wedge XG^{\leq 4} p\right) \vee XF^{\leq 7} G^{\leq 5} p$. Notice that if the current valuation is $\sigma_1 = \{p \mapsto \mathit{false}\}$, ϕ is reduced to $\phi_1 = F^{\leq 7} G^{\leq 5} p$; if it is $\sigma_2 = \{p \mapsto \mathit{true}\}$, ϕ is reduced to $\phi_2 = G^{\leq 4} p \vee F^{\leq 7} G^{\leq 5} p$.

Now we initiate two GPU threads to check whether ϕ is satisfied for two different trajectories. Naively, we implement each thread as *if* σ_1 *then check* ϕ_1 *else check* ϕ_2. Branch divergence happens when the two trajectories take different valuations. Since GPU stream processors require that each GPU thread executes identical instructions, the two threads will process both ϕ_1 and ϕ_2 and simply discard the unrelated part, resulting in a 50% loss of performance. □

3.1 Automaton-based BLTL Path Checking

To better utilize the parallelism of GPUs, we introduce an automaton-based BLTL path checking algorithm. Given a BLTL formula ψ, it is well-known that there exists a positive integer K that depends only on ψ such that for any trajectory τ whose length is greater than K, one needs to examine only a prefix of length K to determine whether τ is a model of ψ [4]. The online procedure we shall construct examines τ as it is being generated (through numerical simulation) in a lock-step fashion. Instead of generating a trajectory of length K at once, it incrementally simulates the ODE model and checks whether the current trajectory satisfies the formula ψ.

It is convenient to focus on the sequence of truth values of the atomic propositions induced by a trajectory. Let us call such a sequence *AP-sequence*. Given

a trajectory $\tau = \mathbf{v}_0\mathbf{v}_1 \dots \mathbf{v}_k$, its induced AP-sequence is denoted as τ_{ap}, which is the sequence $P_0P_1 \dots P_k$ where for $0 \leq i \leq k$:

$(x_j \bowtie v) \in P_i$ iff $\mathbf{v}_i(j) \bowtie v$, $\bowtie \in \{\leq, \geq\}$.

We now wish to construct a deterministic automaton for ψ that accepts (rejects) an AP-sequence iff it is (not) a model of ψ.

As the first step, we replace the time constants mentioned in ψ by symbolic variables and manipulate these variables separately. To this end, we define the formula $sym(\psi)$ inductively as follows.

- $sym(\psi) = \psi$ if ψ is an atomic proposition;
- $sym(\neg\psi) = \neg sym(\psi)$ and $sym(\psi_1 \vee \psi_2) = sym(\psi_1) \vee sym(\psi_2)$;
- $sym(X\psi) = X sym(\psi)$;
- $sym(\psi_1 U^{\leq t} \psi_2) = sym(\psi_1) U^{\leq x_\alpha} sym(\psi_2)$ where $\alpha = \psi_1 U^{\leq t} \psi_2$.

Thus the subscript assigned to the symbolic variable is the sub-formula in which the time constant appears. Often for convenience we will index these variables by integers rather than concrete formulas. Thus $sym(F^{\leq 8} p \vee G^{\leq 3} q)$ will be typically represented as $F^{\leq x_1} p \vee G^{\leq x_2} q$. We refer to $sym(\psi)$ as a *symbolic* BLTL formula.

For a BLTL formula ψ, we now define the automaton $\mathcal{A}_\psi = \langle S_\psi, 2^{AP_\psi}, \rightarrow, s_{in}, \mathcal{F} \rangle$, where S_ψ is the set of states, AP_ψ is the set of atomic propositions that appear in ψ, $\rightarrow \subseteq S_\psi \times 2^{AP_\psi} \times S_\psi$ is the transition relation (to be defined below) , $s_{in} \in S_\psi$ is the initial state and $\mathcal{F} \subseteq S_\psi$ are the final states.

Let $\phi_{in} = sym(\psi)$ and CL be the least set of formulas that contains the sub-formulas of $sym(\psi)$ and satisfies:

If $\psi_1 U^{\leq x} \psi_2$ is in CL then $X\psi_1 U^{\leq x} \psi_2$ is also in CL.

We let BC denote the Boolean combinations of formulas in CL. A state of the automaton is a triple of the form (ϕ, Y, V), where $\phi \in BC$, Y is the set of variables that appear in ϕ, and V is a valuation that assigns a *positive* integer to every variable in Y. We define $s_{in} = (\phi_{in}, Y_{in}, V_{in})$, where Y_{in} is the set of the symbolic variables that appear in ϕ_{in}, and V_{in} assigns to each variable in Y_{in} the corresponding value in ψ. More precisely, if x_α is in Y_{in} and $\alpha = \psi_1 U^{\leq t} \psi_2$ then $V_{in}(x_\alpha) = t$. $\mathcal{F} = \{(true, \emptyset, \emptyset), (false, \emptyset, \emptyset)\}$.

Next we define the the transition relation $\rightarrow$ of $\mathcal{A}$. Let (ϕ, Y, V) and (ϕ', Y', V') be states and $P \subseteq AP_\psi$ be a set of atomic propositions. Then $(\phi, Y, V) \xrightarrow{P} (\phi', Y', V')$ is a transition iff the following conditions are satisfied.

- Suppose $\phi = p$ is an atomic proposition. If $p \in P$, then $\phi' = true$; otherwise, $\phi' = false$. In either case $Y' = V' = \emptyset$.
- Suppose $\phi = \neg\varphi$, and there exists a transition $(\varphi, Y, V) \xrightarrow{P} (\varphi', Y'', V'')$. Then $\phi' = \neg\varphi'$, $Y' = Y''$ and $V' = V''$.
- Suppose $\phi = \phi_1 \vee \phi_2$, and there exist transitions $(\phi_1, Y_1, V_1) \xrightarrow{P} (\phi_1', Y_1', V_1')$ and $(\phi_2, Y_2, V_2) \xrightarrow{P} (\phi_2', Y_2', V_2')$. Then $\phi' = \phi_1' \vee \phi_2'$, $Y' = Y_1' \cup Y_2'$, and $V'(x_i) = V_i'(x_i)$ for $x_i \in X_i$, $i \in \{1, 2\}$.
- Suppose $\phi = X\varphi$. Then $\phi' = \varphi$ and $Y' = Y$ and $V' = V$.
- Suppose $\phi = \phi_1 U^{\leq x_\alpha} \phi_2$, and there exist transitions $(\phi_1, Y_1, V_1) \xrightarrow{P} (\phi_1', Y_1', V_1')$ and $(\phi_2, X_2, V_2) \xrightarrow{P} (\phi_2', Y_2', V_2')$. Then $\phi' = \phi_2' \vee (\phi_1' \wedge X\varphi)$ where $\varphi = \phi_2$

if $V(x_\alpha) = 1$. Furthermore $Y' = Y'_1 \cup Y'_2$ and V' restricted to Y'_1 is V'_1 and V' restricted to Y'_2 is V'_2. If $V(x_\alpha) > 1$ then $\varphi = \phi_1 U^{\leq x_\alpha} \phi_2$. Furthermore $Y' = Y'_1 \cup Y'_2 \cup \{x_\alpha\}$ while V' restricted to Y'_1 is V'_1 and V' restricted to Y'_2 is V'_2. In addition $V'(x_\alpha) = V(x_\alpha) - 1$.

The set of states S_ψ is given inductively:

$s_{in} \in S_\psi$. Suppose $s \in S_\psi$ and $s \xrightarrow{P} s'$. Then $s' \in S_\psi$.

It is easy to show that this automaton has the required properties. Moreover its number of states is bounded by $\ell + \Sigma_{x \in X_{in}} V_{in}(x)$ where ℓ is the number of appearances of the X operator in ψ.

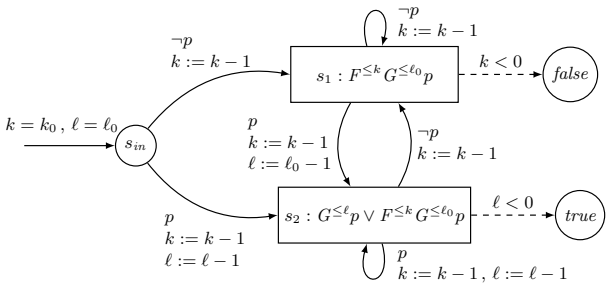

Fig. 1. Automaton for the BLTL formula $F^{\leq k_0} G^{\leq \ell_0} p$ with $k_0 = 8$ and $\ell_0 = 5$.

Example 2. Consider the BLTL formula $\psi = F^{\leq k_0} G^{\leq \ell_0} p$, where $k_0 = 8$ and $\ell_0 = 5$ are constants. Fig 1 shows a fragment of the automaton A_ψ. To avoid clutter we have not explicitly shown the symbolic variables and their valuations. The dashed arcs indicate that the input states will transit to the corresponding final states given proper valuations of atomic propositions. □

4 Parameter Estimation

Often times the values of many of the rate constants appearing in the ODEs and the initial concentrations of the species will be unknown. One will have to learn them using limited experimental data. Solving this parameter estimation problem is the crucial first step towards the analysis of ODEs based bio-pathway models. Here we derive a parallel extension of the method developed by Palaniappan *et al* [27]. This will lead to a GPU implementation of a solution to this crucial problem.

For convenience, we shall assume –as done in the previous section– all the initial concentrations are known but that their nominal values can vary over a cell population. The parameter estimation procedure searches through the value space of the unknown parameters to determine the "best" combination of values that can explain the given data and predict new behaviors [24]. The key

step in this procedure is to determine the fitness-to-data of the current set of parameter values. We use BLTL to encode both experimental time series data and known qualitative trends concerning the dynamics of the pathway. We then develop a parallel statistical model checking procedure (SMC) to determine the goodness of the given set of parameter values, while taking into account that these values can fluctuate across the population of cells that the data is based on. This procedure will numerically generate trajectories in parallel and use our online model checking method to determine if the current trajectory satisfies the given specification. Subsequently, we use a global optimization strategy known as SRES [28] to choose a new set of candidate parameter values according to the SMC based score assigned to the current set.

4.1 Statistical Model Checking

Consider an ODE-based model of a pathway and the associated notations developed in the previous section. In addition, let $\Theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$ be the set of all rate constants. To capture cell-to-cell variability we assume that the range of values for each θ_j is $[L^j, U^j]$ for $1 \leq j \leq m$. We shall present the SMC procedure while assuming that all the rate constants -as interval values- are known. They are to be viewed as the current guess of the values of the unknown parameters.

An implicit assumption is that the value of a rate constant, when fixed initially, does not change during the time evolution of the dynamics, although this value can be different for different cells. To verify whether the ODE system satisfies a property $Pr_{\geq r}\psi$, where ψ is a BLTL formula and r $\in$ [0,1), we use a statistical model checking procedure based on Younes and Simmons' method [33]. Assume that we are given a distribution (usually uniform) over $INIT$ and $\Pi_j[L^j, U^j]$. Then the notion $Pr_{\geq r}\psi$ standing for "the probability of a trajectory chosen randomly according to the given distribution over $INIT$ and $\Pi_j[L^j, U^j]$ satisfying the formula ψ is $\geq$ r"- can be precisely defined [27].

Accordingly, we test the alternative pair of hypotheses : $H_0 : p \geq r + \delta$ and $H_1 : p \leq r - \delta$ where p is the standard probability measure of the set of trajectories that meet the specification ψ and δ is the user defined indifference region. α and β signify the type-I error and type-II error bounds respectively. We use Wald's Sequential Probability Ratio Test (SPRT) [31] for the sequential hypothesis test. In SPRT, random samples are drawn iteratively and we update the SPRT ratio, q_m at the end of each round

$$q_m = \frac{[r-\delta]^{(\sum_{i=1}^{m} y_i)}[1-[r-\delta]]^{(m-\sum_{i=1}^{m} y_i)}}{[r+\delta]^{(\sum_{i=1}^{m} y_i)}[1-[r+\delta]]^{(m-\sum_{i=1}^{m} y_i)}} .$$

The variables $y_1, y_2 \ldots$ signify a sequence of Bernoulli random variables which correspond to the set of trajectories with an assigned value of 1 if a trajectory k satisfies the property ψ or 0, if it does not satisfy. When sufficient samples are drawn, the test terminates. Otherwise, the test proceeds to draw more samples until the statistical guarantee defined by the error bounds and the indifference region are met. We define our stopping criterion as follows: We accept the null

hypothesis H_0 if $q_m \geq \widehat{A}$ and accept the alternate hypothesis H_1 if $q_m \leq \widehat{B}$. Otherwise, we update q_m and sample a new random trajectory. For the thresholds of the sequential hypothesis test, we set $\widehat{A} = \frac{1-\beta}{\alpha}$ and $\widehat{B} = \frac{\beta}{1-\alpha}$.

4.2 The GPU Implementation

In this section, we first describe the design of our online method that overcomes the stringent memory restrictions imposed by the GPU platform to evaluate large number of trajectories as they are numerically generated. We then discuss how the SMC procedure is implemented in our setting using latency hiding.

Our online approach uses the automaton constructed in Section 3, which eliminates the need for handling different formulas explicitly. Recall that running an automaton $\mathcal{A}$ is equivalent to evaluating the corresponding BLTL formula under a series of valuations at different time points until a final state is reached. To efficiently implement this on GPU, branch divergence should be avoided as much as possible. Our solution is to index states, variables and the atomic propositions as defined in Section 3, and encode the transitions and the operations on the valuations into an array A_T. This array represents transitions and operations on the valuations, in which each row corresponds to an input state, and each column to an atomic proposition. Each element of the array consists of an output state and the operations on the valuations associated to the transition. Each GPU thread has access to A_T which is pre-computed and stored in the shared memory. A step in the run of the automaton is performed by all threads of a warp executing in lock-step updating the state and the variables according to A_T. Note that dummy self-loops for the terminal states are added so that once one of them is reached, the automaton stays there forever. This avoids explicit checking for termination, which induces branch divergence.

Example 3. For the fragment of the automaton A_ψ defined in Fig 1, the array

$$A_T = \begin{array}{c} \\ s_{in} \\ s_1 \\ s_2 \\ \top \\ \bot \end{array} \begin{array}{c} \begin{array}{cc} \sigma_1 & \sigma_2 \end{array} \\ \begin{bmatrix} s_1, a_{01} & s_2, a_{02} \\ s_1, a_{11} & s_2, a_{12} \\ s_1, a_{21} & s_2, a_{22} \\ \top, \{\} & \top, \{\} \\ \bot, \{\} & \bot, \{\} \end{bmatrix} \end{array}$$

encodes the automaton, where $\sigma_1 = \{p \mapsto \mathit{false}\}$ and $\sigma_2 = \{p \mapsto \mathit{true}\}$, and a_{ij} updates the set of time variables for the jth transition out of the ith state.

Our code generation scheme for the multi-thread based numerical simulation of an ODEs system is similar to the method developed by Hagiescu *et al.* [13]. During simulation, we generate a number of blocks of trajectories in parallel where the blocks are distributed across a number of GPU cores. At each time step, for each trajectory, we update the current state of the constructed deterministic automaton. We also periodically check if all the trajectories in a given

block have hit a final state in the automaton. When this is the case we update this state information for all the trajectories in the block to the global memory.

If threads from other warps are also scheduled for such long latency global memory accesses, the memory access delay due to control flow divergence will impact performance. To get around this, we use a latency hiding technique where by the global memory accesses are pre-fetched by threads in a separate warp at the same time as when the other threads carry out the numerical integration.

At the global memory level we first pick the terminal state of a trajectory belonging to a block uniformly at random and use it to update the current SPRT score. When the SMC procedure reaches a decision we stop the concurrent numerical integration.

4.3 Parameter Estimation

As the first step, we describe how experimental data can be encoded as BLTL formulas. To do so we first mildly extend the syntax of BLTL with the formulas of type $\psi_1 U^t \psi_2$ with the semantics: ψ_1 will hold exactly up to t time units from now at which point ψ_2 will hold. The construction of the automaton presented in Section 3 can be easily extended to handle this case. Assume, without loss of generality, that $O \subseteq \{x_1, x_2, \ldots, x_k\}$ is the set of variables for which experimental data is available, and which has been allotted as training data to be used for parameter estimation. Assume $\mathcal{T}_i = \{\tau_1^i, \tau_2^i, \ldots, \tau_{T_i}^i\}$ are the time points at which the concentration level of x_i has been measured and reported as $[\ell_t^i, u_t^i]$ for each $t \in \mathcal{T}_i$. The interval $[\ell_t^i, u_t^i]$ is chosen to reflect the noisiness, the limited precision and the cell-population based nature of the experimental data. For each $t \in \mathcal{T}_i$, we define the formula $\psi_i^t = \mathcal{F}^t(i, \ell_t^i, u_t^i)$. Then $\psi_{exp}^i = \bigwedge_{t \in \mathcal{T}_i} \psi_i^t$. We then set $\psi_{exp} = \bigwedge_{i \in O} \psi_{exp}^i$. In case the species x_i has been measured under multiple experimental conditions, the encoding scheme is extended in the obvious way.

Often qualitative dynamic trends will be available – typically from the literature – for some of the molecular species in the pathway. For instance, we may know that a species shows transient activation, in which its level rises in the early time points, and later falls back to initial levels. Similarly, a species may be known to show oscillatory behavior with certain characteristics. Such information can be described as BLTL formulas that we term to be *trend* formulas. We let ψ_{qlty} to be the conjunction of all the trend formulas. We assume $\Theta_u = \{\theta_1, \theta_2, \ldots, \theta_K\}$ as the set of unknown parameters. For convenience we will assume that the other parameter values are known and that their nominal values do not fluctuate across the cell population. We will also assume nominal values for the initial concentrations and the range of their fluctuations of the form $[L_i^{init}, U_i^{init}]$ for each variable x_i. Again, for convenience, we fix a constant δ'' so that if the current estimate of the values of the unknown parameters is $\mathbf{w} \in \prod_{1 \leq j \leq K} [L^j, U^j]$ then this value will fluctuate in the range $[\mathbf{w}(j) - \delta'', \mathbf{w}(j) + \delta'']$. Setting $L_{init,\mathbf{w}}^j = \mathbf{w}(j) - \delta''$ and $U_{init,\mathbf{w}}^j = \mathbf{w}(j) + \delta''$ we define $INIT_{\mathbf{w}} = (\prod_i [L_i^{init}, U_i^{init}]) \times (\prod_j [L_{init,\mathbf{w}}^j U_{init,\mathbf{w}}^j])$. The set of trajectories $TRJ_{\mathbf{w}}$ is defined accordingly.

To estimate the quality of $\mathbf{w}$, we run our parallel SMC procedure –using $INIT_{\mathbf{w}}$– to verify $P_{\geq r}(\psi_{exp} \wedge \psi_{qlty})$. Depending on the outcome of the test for the various conjuncts in the specification, we assign a score to $\mathbf{w}$ using an objective function detailed below. This evaluation is done at the global memory level. We then iterate this scheme for various values of $\mathbf{w}$ generated using a suitable search strategy. For each such $\mathbf{w}$ we launch a fresh instance of the parallel SMC procedure on the GPU network. Using a cloud service, one can launch as many parallel sets of SMC procedures as there are GPU instances available.

The objective function is formed as follows. Let J^i_{exp} $(= T_i)$ be the number of conjuncts in ψ^i_{exp}, and J_{qlty} the number of conjuncts in ψ_{qlty}. Let $J^{i,+}_{exp}(\mathbf{w})$ be the number of formulas of the form ψ^t_i (a conjunct in ψ^i_{exp}) such that the statistical test for $P_{\geq r}(\psi^t_i)$ accepts the null hypothesis (that is, $P_{\geq r}(\psi^t_i)$ holds) with the strength $(\frac{\alpha}{J}, \beta)$, where $J = \sum_{i \in O} J^i_{exp} + J_{qlty}$. Similarly, let $J^+_{qlty}(\mathbf{w})$ be the number of conjuncts in ψ_{qlty} of the form $\psi_{\ell,qlty}$ that pass the statistical test $P_{\geq r}(\psi_{\ell,qlty})$ with the strength $(\frac{\alpha}{J}, \beta)$. Then $\mathcal{G}(\mathbf{w})$ is computed via:

$$\mathcal{G}(\mathbf{w}) = J^+_{qlty}(\mathbf{w}) + \sum_{i \in O} \frac{J^{i,+}_{exp}}{J^i_{exp}} \tag{1}$$

Thus the goodness to fit of $\mathbf{w}$ is measured by how well it agrees with the qualitative properties as well as the number of experimental data points with which there is acceptable agreement. To avoid over-training the model, we do not insist that every qualitative property and every data point must fit well with the dynamics predicted by $\mathbf{w}$.

The search strategy to evolve candidate parameters will use the values $\mathcal{G}(\mathbf{w})$ to traverse the parameter value space. Global search methods such as Genetic Algorithms (GA) [11], and Stochastic Ranking Evolutionary Strategy (SRES) [28] are computationally more intensive than local methods, but are much better at avoiding local minima. In practice, one usually maintains a *population* of parameter value vectors in each round, and a round is usually called a *generation*. We use the SRES strategy in our work since it is known to perform well in the context of pathway models [24]. The particular choice of search algorithm, however, is orthogonal to our proposed method.

5 Experimental Evaluation

We applied our scheme to three models - EGF-NGF, segmentation clock, and thrombin dependent MLC phosphorylation pathway taken from the Bio-Models database [19]. The GPU implementation was based on CUDA 5.0 run-time and tested on four Nvidia Tesla K20m GPUs with 4.8GB global memory, clocked at 706 MHz each. We compare the performance of our algorithm with that of a CPU based implementation on a PC with 3.4Ghz Intel Core i7 processor with 8 GB of memory. The model checker and the numerical solver for the CPU implementation were written in C++. The numerical solver uses the SUNDIALS CVODES package [14] for numerical integration. For the cloud implementation,

we ported our single node implementation to 25 Amazon Web Service (AWS) cloud `g2.8xlarge` GPU nodes. Each node has two Intel Xeon E5-2670 CPU of 8 cores and four Nvidia GK104 GPUs with 60GB host memory and 4GB global memory on each GPU device. The nodes are connected by the AWS Enhanced Networking and communicate using CUDA-aware OpenMPI. The Nvidia GK104 GPUs have 1,536 cores clocked at 797 MHz each with 4 GB global memory and a memory bandwidth of 160 GB/s. We first verified a few properties on each of the three pathway models. Using our parallelized SMC framework, we then performed parameter estimation of these models.

5.1 Case Studies

Thrombin dependent MLC-phosphorylation pathway Two proteins - actin and myosin coordinate the contraction of muscular tissues. Contraction of endothelial cells is caused by Myosin Light Chain (MLC) phosphorylation-dependent actin-myosin interactions. Thrombin is one agonist which can stimulate this MLC phosphorylation. We simulated the model for 1,000 seconds and the sequential hypothesis test was done every 20 time points. For this pathway, the Runge-Kutta-Chebyshev [30] and the Runge-Kutta-Fehlberg solvers were used for numerically integrating the set of stiff ODEs on the GPU.

Sustained response of phosphorylated MLC: Starting from a low value, the concentration of phosphorylated MLC (MLC^*) reaches a high value and shows sustained activation. This property was verified to be false. It is reported [23] that transient phosphorylation of MLC is regulated by MLC kinase.

Our online procedure achieves significant speedup compared to an offline GPU based model checker which would first simulate a sufficiently large number of trajectories, store them in the global memory and then finally transfer the data to the host to carry out the model checking procedure on the CPU. On a single GPU setting, we achieved approximately 4.6X speedup to verify the hypothesis that phosphorylated MLC does not show sustained activation.

$$P_{\geq 0.9}(([MLC^* \leq 1]) \wedge F^{\leq 5}(G^{\leq 20}([MLC^* \geq 3])))$$

EGF-NGF Pathway The EGF-NGF signaling pathway [6] captures the differential response of the neuro-endocrine cell line PC12 to two growth hormones, EGF and NGF. EGF induces cell proliferation while NGF stimulates cell differentiation. This specific behavior is attributed to a downstream signalling cascade mediated by Erk. Simulation time was set to 61 minutes assumed to be observable at every minute.

Transient ERK activation: We checked whether starting from a low value, the concentration of activated Erk reaches a high value and then begins to fall. This behavior was confirmed. m denotes million in the following encoding:

$$P_{\geq 0.9}([0 \leq Erk^* \leq 0.22m] \wedge F^{\leq 10}([0.48m \leq Erk^* \leq 0.56m]) \wedge F^{\leq 20}(G^{\leq 30}([0.22m \leq Erk^* \leq 0.48m])))$$

Bio-pathway model	$\|x\|$	$\|\Theta\|$	$\|\Theta_u\|$	T	$\|\lambda\|$	$\|G\|$
EGF-NGF	32	48	20	61	200	100
Segmentation clock	16	75	39	200	200	300
Thrombin	105	197	100	1000	100	500

Table 1. Parameter estimation setup and model specifications

Segmentation clock pathway Somite formation in vertebrate embryos is controlled by a biological clock - the segmentation clock network - which periodically activates the segmentation genes. This segmentation clock pathway is said to be driven by coupled oscillations in the Notch, Wnt and FGF signalling pathways. The ODEs system was assumed to be observable at every 5 minutes and the total simulation time was set to 200 minutes.

Oscillations: We formulated the oscillations observed in the concentration profile of Dusp6-mRNA as a BLTL property and verified the property to be true.

$$P_{\geq 0.9}([Dusp6 - mRNA \leq 1] \wedge (F^{\leq 10}([Dusp6 - mRNA \geq 5.5] \wedge F^{\leq 10}([Dusp6 - mRNA \leq 1] \wedge F^{\leq 10}([Dusp6 - mRNA \geq 5.5])))))$$

5.2 Parameter Estimation

To determine the best suitable parameters, for each model we assumed a number of kinetic rate parameters to be unknown. We first synthesized experimental time series data for a limited number of species measured at specific time points and divided them into training and test data. To generate these data points, we simulated a large number of random trajectories on the GPU by sampling initial concentration from a ± 5 % range around the nominal values of each species. We also encoded the dynamic trends of a few species as properties in BLTL. Later, for each BLTL property, its respective symbolic automaton was constructed. We allowed 0.5% parameter variability around the current estimate of parameters in each iteration of the search procedure. Table 1 presents the models' specifications and SRES parameters. For each model, we have listed the number of variables ($|x|$), the number of kinetic parameters ($|\Theta|$), the number of unknown parameters ($|\Theta_u|$), simulation time points (T), total number of individuals ($|\lambda|$) and the number of generations ($|G|$). Fig 2, 3, 4 show the fit to data of the simulation profiles with the estimated parameters for both training and testing data.

5.3 Performance

First, Fig 5 illustrates the speed-up of our parallel implementation against a serial implementation for the EGF-NGF and the segmentation clock –non stiff ODEs based– bio-pathway models. For the CPU based implementation, it can be seen that for verifying properties with a high degree of confidence $r \geq 0.9, \alpha = \beta = \delta = 0.01$, it took 17.3 hours. On the other hand, the parallel implementation took just 40 minutes for predicting the best parameter set for the EGF-NGF

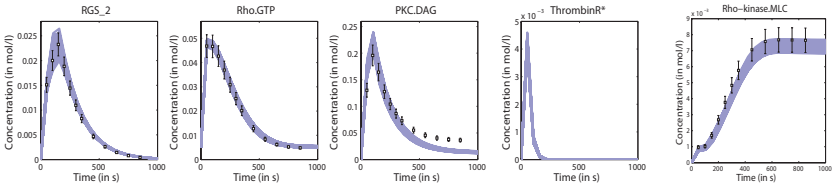

Fig. 2. Parameter estimation of the thrombin pathway. (a) training data (b) test data

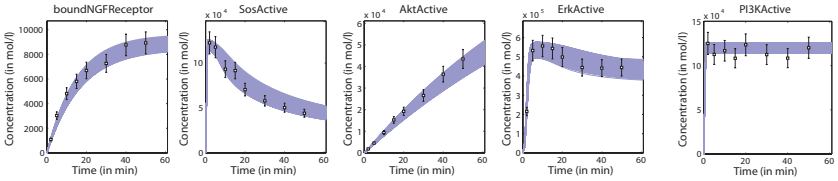

Fig. 3. Parameter estimation of the EGF-NGF pathway. (a) training data (b) test data

model on a 4-GPU node for a range of SPRT parameters, a 24.6× speed-up. Fig 6 shows the time taken for the SRES search for every combination of the desired probability of satisfaction of the properties, type-I and type-II error bounds, indifference-region for the thrombin pathway. Note that for the GPU implementation, the underlying stiff solver is based on an explicit Runge-Kutta method while the CPU based implementation uses a highly optimized CVDense stiff solver from the SUNDIALS CVODES package. In this setting, the GPU based implementation was able to achieve approximately 5x performance over a serial implementation which would run for an excruciatingly long time of 23.2 days for the strongest set of SPRT parameters.

Next, Table 2 shows the performance of our parameter estimation exercise on a range of parallel architectures. The SPRT parameters were set to

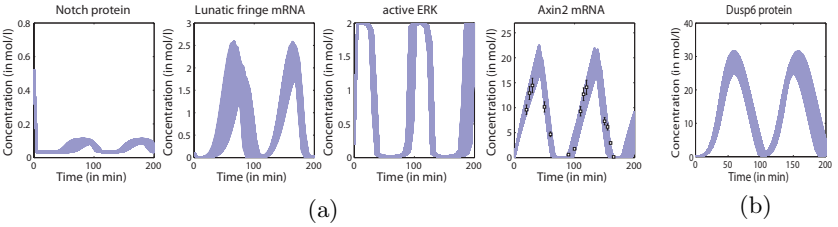

Fig. 4. Parameter estimation of the segmentation clock pathway. (a) training data (b) test data

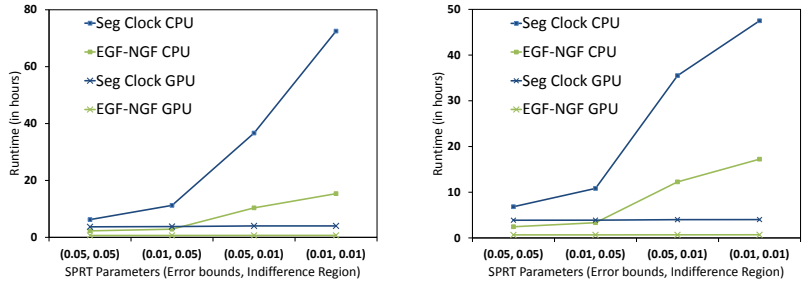

Fig. 5. Parameter estimation performance vs SPRT parameters for EGF-NGF and segmentation clock models. (a) properties with probability ≥ 0.8 (b) probability ≥ 0.9

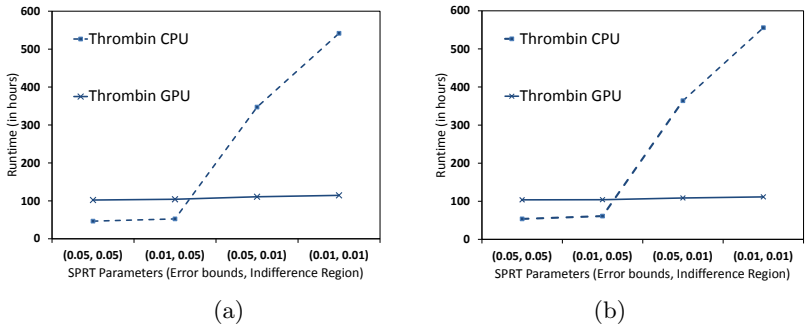

Fig. 6. Parameter estimation performance vs SPRT parameters for thrombin model. (a) properties with probability ≥ 0.8 (b) probability ≥ 0.9

$\alpha = \beta = \delta = 0.01$ and $r = 0.9$. For every generation in our single node parallel implementation, we divided the total number of individuals across 4 GPUs equally. For the cloud based implementation, the set of individuals were divided across 100 GPU instances in 25 machines with 4 GPUs per node.

For the 4-GPU server implementation, we achieved a maximum speed up of 24.6x over the serial implementation for the EGF-NGF pathway. Furthermore, on the cloud, for the same pathway, we were able to estimate the best parameters in 3 minutes. For the segmentation clock pathway, the 4-GPU implementation based SRES search took 4 hours, a speed-up of approximately 11.9x over the CPU implementation. Finally, if one were to compute the best suitable parameters for the thrombin model, it would take around 23.2 days using a CPU based implementation! Even a single-GPU node based implementation would take approximately 4.7 days. The cloud based implementation on the other hand would be able to estimate the parameters in about 5 hours. Note that this is an approximate estimate of the time taken for the thrombin model and we did not actually run the estimation procedure due to the prohibitive computational cost.

Biopathway model	CPU [hr]	1-GPU node [hr]	4-GPU node [hr]	100-GPU cloud [hr]	4-GPU node over CPU
EGF-NGF	17.22	2.8	0.69	0.05	24.6x
Segmentation clock	47.5	16.12	4.01	0.45	11.9x
Thrombin	-	-	111.1	5	5x

Table 2. Performance of our scheme across different architectures

Bio-pathway model	40-GPUs Time[s]	80-GPUs over 40-GPUs	100-GPUs over 40-GPUs
EGF-NGF	445.28	1.62x	2.36x
Segmentation clock	3864.74	1.74x	2.35x

Table 3. Strong scaling performance of the cloud based implementation

Finally, Table 3 presents the strong scaling performance of our parameter estimation method applied on the EGF-NGF and the segmentation clock pathway models on the cloud. Our method achieves near perfect strong scaling when all the individuals in each round of the SRES procedure are launched on unique instances on the cloud.

6 Conclusion

In this paper we proposed a technique for studying the dynamics of large biopathways models that utilizes the power of commodity graphics processors. In particular, starting with a bio-pathway model consisting of a system of ordinary differential equations, we have a parallel, online procedure for checking if the trajectories of this model satisfy a bounded linear temporal logic formula. Our procedure works around various architectural constraints of the graphics processor execution model to achieve significant performance both on local systems as well as in the cloud. We believe that this opens the door for studying large bio-pathway models in a scalable and cost-effective manner.

We have used the parameter estimation problem to illustrate the applicability of our method which consists of a parallel SMC procedure whose core is a deterministic online model checking procedure that determines if the trajectory under construction satisfies a given BLTL formula. Many analysis questions can be tackled by assuming a distribution over the set of initial values of the concentrations and parameter values which will then induce a probability to the set of trajectories satisfying a given BLTL formula. For instance, sensitivity analysis of a model can be carried out in this fashion as shown in [27] and we are currently constructing a GPU based implementation using the framework presented here.

In model reconstruction (i.e. constructing a dynamic model to explain the currently available data and hypotheses) one usually ends up with a population of models with different structures. One can apply our scheme to evaluate the maximal likelihood estimates over this population using our online model checking method. Indeed with sufficient GPU units available one can evaluate the quality of a large number of these models in parallel using our method. One can also explore the parameter landscape to identify regions most likely to induce the desired pathway responses to chosen stimuli. Our future work will involve exploring such issues in the context of model learning.

References

1. Barnat, J., Brim, L., Cerná, I., Drazan, S., Fabriková, J., Láník, J., Safránek, D., Ma, H.: Biodivine: A framework for parallel analysis of biological models. In: Proceedings Second International Workshop on Computational Models for Cell Processes, COMPMOD 2009, Eindhoven, the Netherlands, November 3, 2009. pp. 31–45 (2009)
2. Barnat, J., Brim, L., Ceska, M., Lamr, T.: Cuda accelerated ltl model checking. In: Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on. pp. 34–41. IEEE (2009)
3. Barre, B., Klein, M., Soucy-Boivin, M., Ollivier, P.A., Hallé, S.: Mapreduce for parallel trace validation of ltl properties. In: Runtime Verification. pp. 184–198. Springer (2013)
4. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without bdds. In: Intl. Conf. on Tools and Algorithms for the Analysis and Construction of Systems (TACAS'99). vol. 1579. Springer (1999)
5. Bortolussi, L., Sanguinetti, G.: Learning and designing stochastic processes from logical constraints. In: Quantitative Evaluation of Systems, pp. 89–105. Springer (2013)
6. Brown, K.S., Hill, C.C., Calero, G.A., Myers, C.R., Lee, K.H., Sethna, J.P., Cerione, R.A.: The statistical mechanics of complex signaling networks: nerve growth factor signaling. Physical biology 1(3), 184 (2004)
7. Bulychev, P., David, A., Larsen, K.G., Mikučionis, M., Poulsen, D.B., Legay, A., Wang, Z.: Uppaal-smc: Statistical model checking for priced timed automata. arXiv preprint arXiv:1207.1272 (2012)
8. Clarke, E.M., Faeder, J.R., Langmead, C.J., Harris, L.A., Jha, S.K., Legay, A.: Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In: Computational Methods in Systems Biology. pp. 231–250. Springer (2008)
9. David, A., Du, D., Larsen, K.G., Legay, A., Mikučionis, M.: Optimizing control strategy using statistical model checking. In: NASA formal methods, pp. 352–367. Springer (2013)
10. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
11. Goldberg, D.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley (1989)
12. Gyori, B.M., Liu, B., Paul, S., Ramanathan, R., Thiagarajan, P.: Approximate probabilistic verification of hybrid systems. In: Hybrid Systems Biology. Springer (2015)
13. Hagiescu, A., Liu, B., Ramanathan, R., Palaniappan, S.K., Cui, Z., Chattopadhyay, B., Thiagarajan, P., Wong, W.F.: Gpu code generation for ode-based applications with phased shared-data access patterns. ACM Transactions on Architecture and Code Optimization (TACO) 10(4), 55 (2013)
14. Hindmarsh, A., Brown, P., Grant, K., Lee, S., Serban, R., Shumaker, D., Woodward, C.: SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM T. Math. Software 31(3), 363–396 (2005)
15. Hirsch, M., Smale, S., Devaney, R.: Differential equations, dynamical systems, and an introduction to chaos. Academic Press (2012)
16. Jha, S.K., Langmead, C.J.: Synthesis and infeasibility analysis for stochastic models of biochemical systems using statistical model checking and abstraction refinement. Theoretical Computer Science 412(21), 2162–2187 (2011)

17. Klipp, E., Herwig, R., Kowald, A., Wierling, C., Lehrach, H.: Systems biology in practice: concepts, implementation and application. Wiley-VCH, Weinheim (2005)
18. Kuhtz, L., Finkbeiner, B.: Efficient parallel path checking for linear-time temporal logic with past and bounds. arXiv preprint arXiv:1210.0574 (2012)
19. Le Novere, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J., Hucka, M.: BioModels Database: A free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acids Res. 34, D689–D691 (2006)
20. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: Runtime Verification. pp. 122–135. Springer (2010)
21. Lindholm, E., Nickolls, J., Oberman, S., Montrym, J.: Nvidia tesla: A unified graphics and computing architecture. IEEE micro (2), 39–55 (2008)
22. Liu, B., Hagiescu, A., Palaniappan, S.K., Chattopadhyay, B., Cui, Z., Wong, W.F., Thiagarajan, P.: Approximate probabilistic analysis of biopathway dynamics. Bioinformatics 28(11), 1508–1516 (2012)
23. Maedo, A., Ozaki, Y., Sivakumaran, S., Akiyama, T., Urakubo, H., Usami, A., Sato, M., Kaibuchi, K., Kuroda, S.: Ca^{2+}-independent phospholipase A2-dependent sustained Rho-kinase activation exhibits all-or-none response. Genes Cells 11, 1071–1083 (2006)
24. Moles, C.G., Mendes, P., Banga, J.R.: Parameter estimation in biochemical pathways: A comparison of global optimization methods. Genome Res. 13(11), 2467 –2474 (2003)
25. Murray, L.: Gpu acceleration of runge-kutta integrators. Parallel and Distributed Systems, IEEE Transactions on 23(1), 94–101 (2012)
26. Oshima, K., Matsumoto, T., Fujita, M.: Hardware implementation of bltl property checkers for acceleration of statistical model checking. In: Proceedings of the International Conference on Computer-Aided Design. pp. 670–676. IEEE Press (2013)
27. Palaniappan, S.K., Gyori, B.M., Liu, B., Hsu, D., Thiagarajan, P.: Statistical model checking based calibration and analysis of bio-pathway models. In: Computational Methods in Systems Biology. pp. 120–134. Springer (2013)
28. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. Evolutionary Computation, IEEE Transactions on 4(3), 284–294 (2000)
29. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: Computer Aided Verification. pp. 266–280. Springer (2005)
30. Verwer, J., Hundsdorfer, W., Sommeijer, B.: Convergence properties of the runge-kutta-chebyshev method. Numerische Mathematik 57(1), 157–178 (1990)
31. Wald, A.: Sequential tests of statistical hypotheses. The Annals of Mathematical Statistics 16, 117–186 (1945)
32. Younes, H.L., Kwiatkowska, M., Norman, G., Parker, D.: Numerical vs. statistical probabilistic model checking. International Journal on Software Tools for Technology Transfer 8(3), 216–228 (2006)
33. Younes, H.L., Simmons, R.G.: Statistical probabilistic model checking with a focus on time-bounded properties. Information and Computation 204(9), 1368–1409 (2006)
34. Zhou, Y., Liepe, J., Sheng, X., Stumpf, M.P., Barnes, C.: Gpu accelerated biochemical network simulation. Bioinformatics 27(6), 874–876 (2011)