Guilherme Balestieri Bedin

Arquitetura Computacional para Simulação de Redes Metabólicas em Larga Escala

Dissertação submetida a avaliação como requisito parcial para a obtenção do grau de Mestre em Computao Aplicada

Orientador: Prof Dr. Ney Lemke

So Leopoldo
2005

Espaço reservado para folha de aprovação impressa.

Dedico este trabalho aos meus pais, Ramon e Tita.

Agradecimentos

Primeiramente, gostaria de agradecer o meu orientador por ter me convencido a mudar de área e realizar meu trabalho em bioinformática. Além disso, por todos ensinamentos e conselhos durante estes longos dois últimos anos. Também gostaria de agradecer a todo o pessoal do laboratório de bioinformática por terem sido ótimos companheiros de jornada. Em especial a Norma pelas explicações sobre biologia molecular. Gostaria ainda de agradecer a todos os colegas da T&T. Em especial ao pessoal da "*Printers*" e aos sócios da empresa que me proporcionaram algumas liberdades em relação a horário e faltas. E sem elas seria inviável trabalhar e estudar. Por último, mas não menos importante, gostaria de agradecer aos meus familiares e amigos que me acompanharam nesta aventura pelo apoio e compreensão.

Resumo

A quantidade de dados disponíveis a respeito do funcionamento das células vêm aumentando a cada dia. Muitos organismos tiveram suas redes metabólicas caracterizadas. O estudo destas redes possibilita uma melhor compreensão dos processos envolvidos no funcionamento da célula, podendo ser utilizado no desenho racional de drogas e na predição de seus efeitos colaterais. Uma das principais ferramentas para o estudo de redes metabólicas é a simulação computacional, mas quanto maiores e mais complexas forem estas redes maior será o custo computacional para simulá-las. Estes sistemas possuem uma natureza estocástica, podendo seguir caminhos diferentes dependendo das condições do ambiente. A computação das simulações das redes metabólicas usando um modelo estocástico pode ser realizada de forma distribuída, mas seu custo computacionalmente é alto. Em contrapartida, a tecnologia de *grid* de computadores vêm evoluindo e tornando-se uma alternativa aos supercomputadores para processamento de alto desempenho. O uso de *grid* pode viabilizar o estudo de redes metabólicas complexas num tempo aceitável. Neste trabalho descreve-se a implementação de uma arquitetura computacional para a realização destas simulações em larga escala. Esta arquitetura tira proveito das características dos métodos estocásticos de simulação em relação a paralelismo e distribuição. Os métodos implementados foram validados através de comparação com resultados da literatura. Além disso, também foi avaliado o desempenho da arquitetura computacional para a computação de simulações em um ambiente de *grid*. Os resultados obtidos mostram uma diminuição significativa no tempo total de simulação com a utilização da arquitetura proposta.

TITLE: "Computational Architecture for Metabolic Network Large Scale Simulation"

Abstract

The amount of biological data is growing every day. Many organisms had their metabolic networks characterized. These studies permit a better understanding of cell behavior, and can be used to aid rational drug design and prediction of its side effects. One of the main tools to explore metabolic networks is computer simulations, but the size and complexity of networks impacts on the simulation costs. These systems have a stochastic behavior, they can take different paths depending on environmental conditions. Metabolic network simulation using a stochastic model computation can be distributed, but they have a high computational cost. On the other hand, the grid technology is evolving and becaming an alternative to supercomputers on high performance computing. The study of complex metabolic networks can be done on acceptable time by the employment of grids. In this work we describe an implementation of a computational architeture to execute this kind of large scale simulation. This architeture explores charecteristics of the stochastic methods to achieve efficient simulation cost. The implemented methods were validated by comparing its results with ones obtained in the literature. We also evaluate performance of the computational architecture on a grid environment. The results show a significant reduction on simulation costs when using the proposed architecture.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A evolução dos computadores é parte fundamental na mudança que vem ocorrendo nas ciências da vida. A biotecnologia captura o segmento das ciências da vida que combina biologia, química, física, matemática e ciência da computação, aplicando técnicas biológicas desenvolvidas através de pesquisa básica e desenvolvimento de produtos. Da mesma maneira que os computadores mudaram o desenvolvimeto na indústria automotiva, onde simulações paralelas substituíram os protótipos físicos no estudos de colisão de carro, a biotecnologia usa computação paralela para aprimorar drasticamente o entendimento dos blocos de construção da vida, genes e proteínas, habilitando que cientistas testem, categorizem, e simulem os efeitos das novas drogas no corpo humano (ABBAS, 2004).

A bioinformática é um ramo da biotecnologia, onde biologia, ciência da computação e tecnologia da informação se unem para formar uma única disciplina. No começo da "revolução genômica", a bioinformática tratava somente da criação e manutenção dos bancos de dados que armazenavam informação biológica: bioquímica clássica, genomas, experimentos de microarranjos de DNA, análise topológica, descrição dos processos da vida e simulação. Entretanto, uma teoria da célula deve combinar as descrições das estruturas existentes nela e uma descrição computacional e teórica da dinâmica dos processos da vida.

Um dos maiores desafios da bioinformática é tornar toda a informação obtida através de experimentos e simulações compreensíveis em termos biológicos, a fim

de usar estas informações para fazer predições. Além de saber o que está acontecendo no organismo, também almeja-se predizer o que irá acontecer dado uma circunstância específica. Este tipo de poder preditivo somente pode ser alcançado simulando os processos biológicos computacionalmente. Hoje em dia, já é possível replicar comportamentos biológicos através de simulação computacional (KITANO, 2002). Entretanto, estas simulações possuem um alto custo computacional (KIEHL; MATTHEYSES; SIMMONS, 2004).

As simulações computacionais dos processos biológicos são uma ferramenta importante no desenho racional de drogas. Com elas, é possível diminuir o alto custo do desenvolvimento e teste de medicamentos. Outra possibilidade interessante é o tratamento personalizado, onde são coletadas informações biológicas do paciente e usando simulação desenvolve-se uma droga personalizada para ele.

Uma área que está em franca expansão hoje dia é a engenharia metabólica de organismos, também conhecida como vida sintética (GIBBS, 2004), onde se pretende criar *BioBricks*, componentes com funções específicas. Estes componentes podem ser combinados para desenhar ou modificar um organismo para alguma finalidade (máquinas vivas), como por exemplo, modificar uma espécie de bactéria para consumir lixo radioativo. Para criação dos *BioBricks* é necessário o entendimento de vários processos biológicos. Entretanto, quanto mais complexo for o *BioBrick* ou a combinação deles, maior é a necessidade de simular computacionalmente estes sistemas para visualizar e prever o seu comportamento.

O custo computacional de simular processos biológicos está diretamente ligado a complexidade dos sistemas modelados (HASELTINE; RAWLINGS, 2002). As novas aplicações do conhecimento adquirido pela área da bioinformática nos últimos anos vem tornando cada vez maior a necessidade de simular processos biológicos complexos. Muitas vezes o poder computacional pode ser o fator limitante, inviabilizando a simulação do sistema biológico em tempo plausível.

Atualmente, existem métodos de simulação de processos biológicos propícios para paralelização e distribuição do processamento. Nestes métodos a computação

pode ser dividida sem a necessidade de comunicação entre as partes durante o processamento. Nos últimos anos, a tecnologia de *grid* de computadores vem evoluindo e tornando-se uma alternativa aos supercomputadores para processamento de alto desempenho. O uso de *grid* pode viabilizar o estudo de processos biológicos complexos num tempo aceitável, pois permite o compartilhamento colaborativo de recursos em ambientes multi-institucionais (FOSTER, 1999).

A figura 1.1, obtida em (GOBLE, 2002), mostra uma previsão do custo computacional de que várias aplicações da bioinformática irão necessitar até o ano de 2010 e o compara com a lei de Moore (*Moore's Law*). A lei de Moore diz que a capacidade dos processadores dobra a cada 18 meses. Os *metobolic pathways* são representados por redes metabólicas, as quais são simuladas para o estudo e compreensão dos mesmos. Como pode-se ver no gráfico, este tipo de simulação irá necessitar de um grande poder computacional, muito maior do que um computador atual sozinho conseguiria. Novamente, evidencia-se a necessidade de uma tecnologia como a de *grid* para realização deste tipo de simulação.

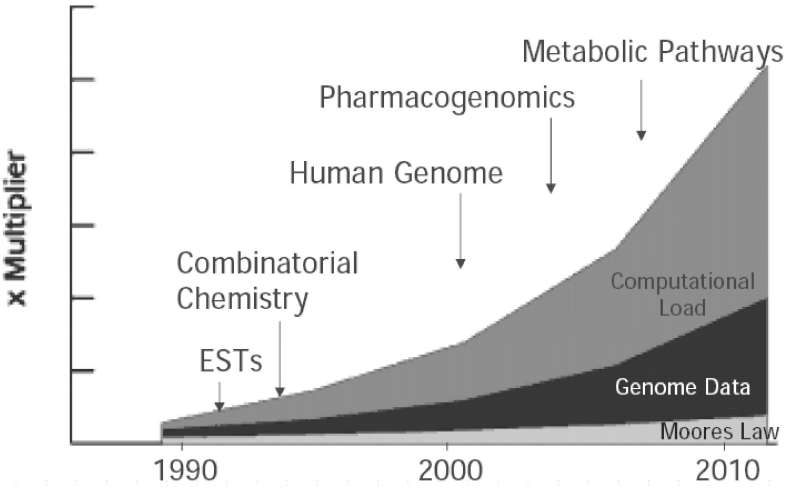


FIGURA 1.1 – Previsão do custo computacional requerido para diversas aplicações da bioinformática comparados com a lei de Moore (GOBLE, 2002).

Neste trabalho será descrita a implementação de uma arquitetura computacional para simulação de redes metabólicas em larga escala. Esta arquitetura objetiva tirar proveito das características deste tipo de simulação em relação a paralelismo

e distribuição. O ambiente alvo desta arquitetura são *grids* de computadores heterogêneos. A arquitetura utiliza o paralelismo intra-nó e entre-nós levando em conta as diferenças entre as máquinas. Por exemplo, em máquinas bi-processadas são usadas *threads* para executar mais de um fluxo de execução do simulador.

As simulações podem ser feitas usando tanto o modelo estocástico, baseado no algoritmo de Gillespie (GILLESPIE, 1977), quanto o modelo determinístico, utilizando o método de Runge-Kutta (PRESS et al., 2002). O modelo estocástico possue um alto custo computacional, mas representa melhor os processos biológicos. Enquanto, o modelo dertminístico possue um custo computacional baixo e não consegue representar alguns processos biológicos. Outra característica importante da arquitetura computacional proposta é a possibilidade da utilização do método híbrido. Nesta abordagem, usa-se o modelo determinístico para simular uma parte da rede metabólica enquanto a outra parte utiliza o modelo estocástico, para diminuir o custo computacional do modelo estocástico mantendo suas características.

No capítulo 2 é feita uma revisão da parte biológica relevante para este trabalho. O capítulo 3 descreve os modelos e os métodos de simulação de redes metabólicas. Os trabalhos relacionados são discutidos no capítulo 4. O capítulo 5 descreve como foi implementada a arquitetura computacional para simulação de redes metabólicas em larga escala e como usá-la. No capítulo 6 são apresentados os resultados obtidos através da utilização da arquitetura, validação e analise de desempenho. E por último, são tecidas as conclusões no capítulo 7.

Capítulo 2

Redes Metabólicas

Quando considera-se um organismo como um todo, o metabolismo representa todos os processos químicos do mesmo. O metabolismo do organismo pode ser dividido em duas partes, a síntese e a quebra de moléculas orgânicas, denominadas anabolismo e catabolismo respectivamente. Quando considera-se uma substância em particular, o metabolismo é a atividade química envolvendo essa substância num organismo vivo. Já quando refere-se a uma célula viva específica, o metabolismo é o conjunto de todos os processos químicos daquela célula.

A representação esquemática em 2.1 é aplicada igualmente para ecossistemas, para organismos inteiros e assim como para uma única célula. O núcleo central do metabolismo bioquímico é essencialmente o mesmo em todos os organismos existentes hoje em dia.

O metabolismo pode realizar muitas conversões químicas complexas em uma série de pequenos passos, sendo cada passo (reação) catalizado por enzimas. Os conjuntos de reações são chamados de mapas metabólicos e possuem funções catabólicas e anabólicas específicas. O conjunto com todos os mapas metabólicos formam a rede metabólica (LEHNINGER; NELSON; COX, 1993). As próprias enzimas são produtos metabólicos, porém como catalizadores elas estão presentes em pequenas quantidades. Em um mesmo organismo podem existir mais de uma enzima catalizando a mesma reação (isoenzimas). Por outro lado, nenhum organismo realiza todo o conjunto de reações possíveis, tipicamente as células contém de algumas centenas a

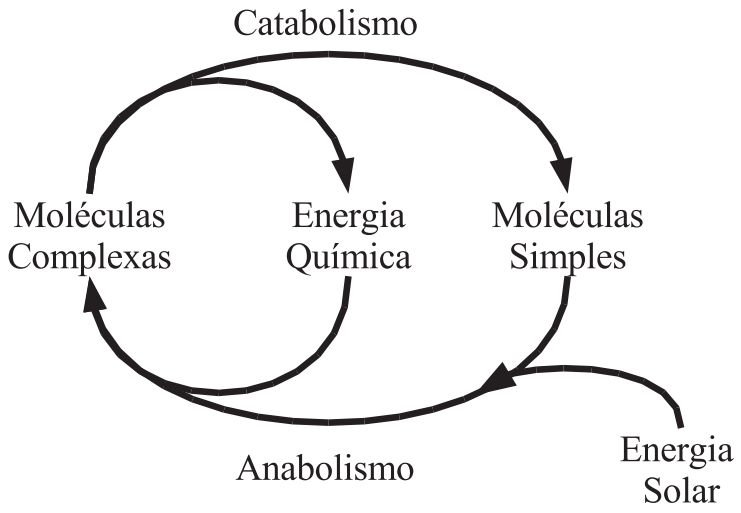


FIGURA 2.1 – Metabolismo

alguns milhares de enzimas (FEEL, 1997).

Neste capítulo será feita uma breve revisão de biologia molecular, tratando dos assuntos pertinentes ao entendimento do que são e para que servem as redes metabólicas. A seção 2.1 apresenta os componentes básicos da biologia molecular, indo desde os aminoácidos até as proteínas. A seção 2.2 fala sobre a regulação da expressão gênica e seus processos como transcrição e tradução. Na seção 2.3 é mostrado como modelar matematicamente as reações bioquímicas, as quais representam uma rede metabólica ou um subconjunto dela. Nas seções 2.4 e 2.5 apresentam-se dois modelos de redes metabólicas clássicos.

2.1 Biologia Molecular

A maioria dos constituintes moleculares dos sistemas vivos são compostos de átomos de carbono unidos covalentemente a outros átomos de carbono e de hidrogênio, oxigênio e nitrogênio. As propriedades especiais de ligação do carbono permitem a formação de uma grande variedade de moléculas. Os compostos orgânicos de massa molecular menor, como aminoácidos, nucleotídeos e monossacarídios, servem como subunidades monoméricas de proteínas, ácidos nucléicos e polissacarídios, respectivamente. Uma única molécula protéica pode ter 1.000 ou mais aminoácidos

e o ácido desoxirribonucléico tem milhões de nucleotídeos (LEHNINGER; NELSON; COX, 1993).

Existem mais de 6.000 diferentes tipos de compostos orgânicos em cada célula da bactéria *Escherichia coli (E. coli)*, incluindo perto de 3.000 proteínas diferentes e um número similar de moléculas de ácidos nucléicos. Em humanos pode haver dezenas de milhares de tipos diferentes de proteínas, assim como muitos tipos de polissacarídeos (cadeias de açucares simples), uma grande variedade de lipídios e muitos outros compostos de peso molecular menor.

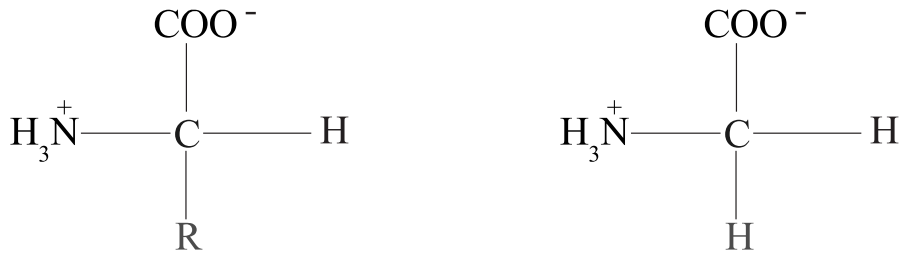
Purificar e caracterizar exatamente todas essas moléculas seria um trabalho incomensurável se não fosse o fato de que cada classe de macromoléculas (proteínas, ácidos nucléicos, polissacarídeos) é composta de um pequeno conjunto de subunidades monoméricas comuns. Estas subunidades monoméricas podem ser unidas covalentemente em uma variedade virtualmente ilimitada de seqüências.

Os ácidos desoxirribonucléicos (DNA) e os ácidos ribonucléicos (RNA) são construídos de apenas quatro diferentes nucleotídeos. As proteínas são compostas de 20 tipos diferentes de aminoácidos. Os quatro tipos de nucleotídeos com os quais todos os ácidos nucléicos são construídos e os 20 tipos de aminoácidos com os quais todas as proteínas são construídas, são idênticos em todos os organismos vivos.

A maioria das subunidades monoméricas com as quais todas as macromoléculas são construídas exercem mais de uma função nas células vivas. Os nucleotídeos servem não apenas como subunidades para os ácidos nucléicos, mas também como moléculas transportadoras de energia. Os aminoácidos são subunidades das moléculas protéicas e também precursores de neurotransmissores, pigmentos e muitos outros tipos de biomoléculas.

2.1.1 Aminoácidos

As subunidades monoméricas relativamente simples das proteínas fornecem a chave para a estrutura de milhares de proteínas diferentes. Todas as proteínas, são construídas com o mesmo conjunto de 20 aminoácidos, ligados covalentemente



(a) Aminoácido (b) Glicina

FIGURA 2.2 – Estrutura geral dos aminoácidos presentes nas proteínas. Tirando o grupo **R**, ou cadeia lateral, esta estrutura é comum a todos os α-aminoácidos das proteínas (exceto a prolina). O carbono α aparece no centro das figuras.

em seqüências lineares características. Dado que cada um destes aminoácidos possui uma cadeia lateral distinta, a qual determina suas propriedades químicas, este grupo de 20 moléculas precursoras pode ser considerado como o alfabeto com o qual a linguagem da estrutura protéica é escrita (LEHNINGER; NELSON; COX, 1993).

Todos os aminoácidos encontrados nas proteínas possuem um grupo carboxila e um grupo amino ligados ao mesmo átomo de carbono, o carbono α (figura 2.2). Eles se diferenciam através de suas cadeias laterais (grupos **R**), os quais variam em estrutura, tamanho e carga elétrica, e influenciam na solubilidade do aminoácido em água.

Os polímeros de aminoácidos, mais conhecidos como peptídios, variam muito de tamanho, desde moléculas pequenas contendo apenas dois ou três aminoácidos até grandes macromoléculas contendo milhares de aminoácidos. Os peptídeos maiores são chamados de proteínas e serão melhor analisados na seção 2.1.2.

Dois aminoácidos podem ser unidos covalentemente através de uma ligação amida substituída, chamada ligação peptídica, para formar um dipeptídio. Tal ligação é formada por remoção dos elementos da água de um grupo α-carboxila de um aminoácido e do grupo α-amido de outro (figura 2.3). Esta ligação peptídica é um exemplo de reação de condensação, uma classe de reação muito comum nas células vivas.

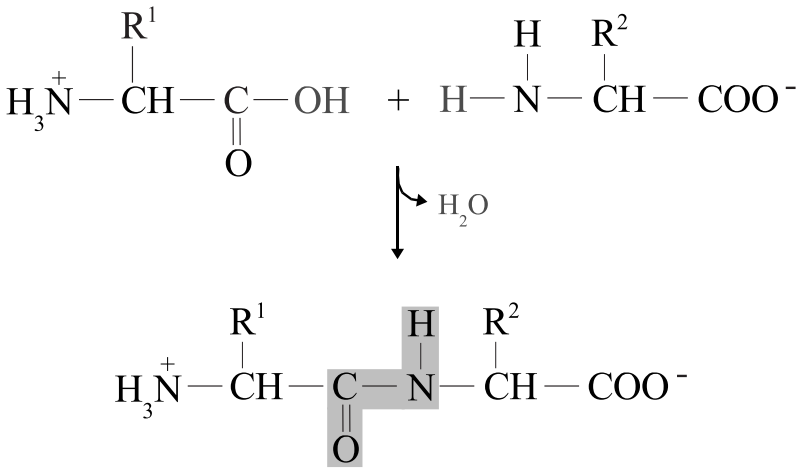


FIGURA 2.3 – Formação de uma ligação peptídica em um dipeptídio, sombreado em cinza.

2.1.2 Proteínas

As células, em geral, contém milhares de proteínas diferentes, cada uma com uma função ou atividade biológica diferente. Estas funções incluem catálise enzimática, transporte molecular, nutrição, mudança de forma (motilidade) do organismo ou célula, papéis estruturais, defesa do organismo, regulação entre muitas outras.

As proteínas consistem de cadeias polipeptídicas muito longas, tendo de 100 a mais de 2.000 resíduos de aminoácidos unidos por ligações peptídicas. Algumas proteínas têm várias cadeias polipeptídicas, as quais são chamadas de subunidades. As suas diferentes funções resultam das diferenças em composição e seqüência de seus aminoácidos. Um polipeptídio com uma seqüência específica de aminoácidos enovela-se em uma estrutura tridimensional única, esta estrutura por sua vez determina a função da proteína.

Quando comparam-se proteínas com funções similares em diferentes espécies observa-se que freqüentemente as seqüências de aminoácidos delas são similares. A seqüência de uma proteína particular não é absolutamente fixa mesmo dentro de uma espécie. Estima-se que na espécie humana cerca de 20 a 30% das proteínas são polimórficas, tendo seqüências de aminoácidos variáveis na população. Muitas dessas variações de seqüência tem pouco ou nenhum efeito sobre a função da proteína.

2.1.3 Nucleotídeos, DNA e RNA

Os nucleotídeos são compostos ricos em energia que direcionam os processos metabólicos em todas as células. Eles também funcionam como sinais químicos, elos importantes nos sistemas celulares que respondem a hormônios e a outros estímulos extracelulares, e são componentes estruturais de vários cofatores enzimáticos e de intermediários metabólicos.

Os três componentes característicos dos nucleotídeos são: uma base nitrogenada, uma pentose e um fosfato. As bases nitrogenadas são derivadas de dois compostos ancestrais, as pirimidinas e as purinas. As estruturas fundamentais dos ácidos nucléicos são os nucleotídeos. Uma molécula de DNA é uma cadeia composta de unidades de nucleotídeos repetidas. Os dois ácidos nucléicos DNA e RNA contém duas bases púricas principais, adenina (A) e guanina (G). Eles possuem também duas pirimidinas principais, uma delas é a citosina (C) e a outra no DNA é a timina (T) e no RNA a uracila (U).

Em 1953, Watson e Crick (WATSON; CRICK, 1953) propuseram o modelo tridimensional para a estrutura do DNA, que consiste de duas cadeias helicoidais de DNA que se enrolam ao redor do mesmo eixo formando uma dupla hélice que gira no sentido da mão direita. As bases púricas e pirimídicas de ambas as fitas estão empilhadas dentro da dupla hélice. Cada base de uma fita está pareada no mesmo plano com uma base de outra fita.

Na estrutura Watson-Crick as duas cadeias ou fitas da hélice são antiparalelas, suas ligações fosfodiéster $5', 3'$ correm em direções opostas. As duas cadeias polinucleotídicas antipararelas da dupla hélice não são idênticas nem na composição nem na seqüência de bases. Elas são complementares entre si. Toda vez que aparecer adenina numa cadeia, timina será encontrada na outra. Similarmente, onde se encontrar guanina numa cadeia encontraremos citosina na outra.

Os ácidos desoxiribonucléicos (DNA), ácidos ribonucléicos (RNA) e proteínas possuem funções diferentes. Na divisão celular o DNA é replicado, então cada célula filha mantém a mesma informação genética da célula mãe. O RNA age como um

meio entre o DNA e as proteínas. Somente uma cópia de DNA está presente na célula. Por outro lado, podem existir múltiplas cópias do mesmo pedaço de RNA, permitindo que as células produzam uma grande quantidade da mesma proteína. Nos eucariotos (organismos com núcleo), o DNA se localiza dentro do núcleo. As moléculas de RNA são produzidas dentro do núcleo e então transportadas para fora dele, onde são traduzidas em proteínas.

2.1.4 Genes

A compreensão atual do gene tem evoluído consideravelmente no último século. Um gene é definido no sentido biológico clássico como uma porção de cromossomo que determina ou afeta um único caráter ou fenótipo (propriedade visível), por exemplo, a cor dos olhos. Entretanto, há também uma definição molecular, um gene é um segmento do material genético que determina ou codifica uma enzima: hipótese de um gene-uma-enzima. Mais tarde, este conceito foi ampliado para um gene-uma-proteína, pelo fato de que nem todas as proteínas possuem funções enzimáticas.

A definição bioquímica atual de um gene é um pouco mais precisa. Em algumas proteínas de cadeias múltiplas todas as cadeias polipeptídicas são idênticas, caso em que elas todas podem ser codificadas pelo mesmo gene. Outras apresentam duas ou mais espécies de cadeias polipeptídicas diferentes, cada uma com uma seqüência de aminoácidos distinta. Desta forma a relação gene-proteína é mais precisamente descrita pela frase "um-gene-um-polipeptídio".

Entretanto, nem todos os genes são expressos na forma de cadeias polipeptídicas. Alguns genes codificam as diferentes espécies de RNAs. Os genes que codificam quer polipeptídios quer RNAs são conhecidos como genes estruturais: eles codificam a seqüência primária de algum produto gênico final, como uma enzima ou um RNA estável. O DNA também contém outros segmentos ou seqüências que tem uma função puramente reguladora. Seqüências reguladoras fornecem sinais que podem denotar o início e o fim de genes estruturais, participar no ligar e desligar da transcrição dos genes estruturais ou funcionar como pontos de iniciação para replicação

ou recombinação.

2.2 Regulação da Expressão Gênica

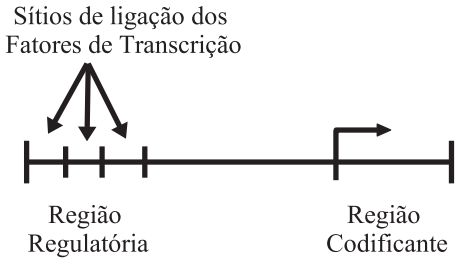
A regulação da expressão gênica é um componente crítico na regulação do metabolismo celular, na orquestração e manutenção das diferenças estruturais e funcionais que existem nas células durante o desenvolvimento. Dado o alto custo da síntese de proteínas, a regulação da expressão gênica é essencial para a célula fazer o melhor uso da energia disponível.

O dogma central da biologia molecular afirma que a informação é armazenada no DNA, transcrita para o RNA mensageiro (mRNA) e traduzida em proteínas. Este cenário torna-se mais complexo quando consideramos a ação de certas proteínas regulando a transcrição. Estas proteínas são chamadas de fatores de transcrição (FTs), elas provem uma via de regulação da expressão de mRNAs.

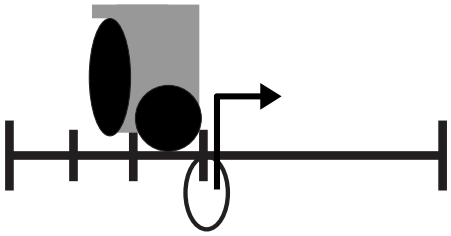
O processo biológico da expressão gênica é um rico e complexo conjunto de eventos que levam do DNA a proteínas funcionais através de vários passos intermediários. Os passos chaves são transcrição e tradução, os quais serão descritos nas seções 2.2.1 e 2.2.2 respectivamente.

2.2.1 Transcrição

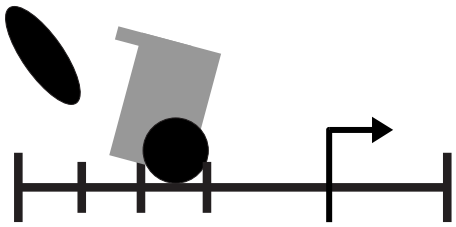
A transcrição é um conjunto de eventos complicados que geram RNA mensageiro (mRNA) a partir do DNA. Considere um gene, como o mostrado na figura 2.4(a). O gene possui uma região codificante e uma região regulatória. A região codificante é a parte do gene que codifica uma proteína, esta é a parte que será transcrita em mRNA e traduzida em proteína. A região regulatória é a parte do DNA que controla a expressão do gene. Ela contém sítios de ligação para os fatores de transcrição, os quais se ligam no DNA e afetam o início da transcrição. Em procariotos, a região regulatória é tipicamente curta (10-100 bases) e contém sítios para um número de fatores de transcrição. Em eucariotos, a região regulatória pode ser longa (mais de 100.000 bases, (BOWER; BOLOURI, 2001)), e contém sítios de



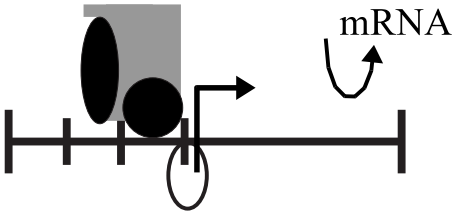
(a) Um único gene pode ser ativado ou desativado por várias combinações de proteína.



(b) O complexo FT habilita a ligação da RNAP.



(c) Ativação ou desativação dos complexos FT ligados ao DNA regulatório



(d) A RNAP faz a cópia do DNA para mRNA.

FIGURA 2.4 – Em cinza está representado a RNAP e em preto os fatores de transcrição. (a) ligação dos fatores de transcrição, (b) formação do complexo de transcrição, (c) ligação da RNA polimerase e (d) iniciação da transcrição.

ligação para múltiplos fatores de transcrição. Os fatores de transcrição podem agir aumentando ou diminuindo a expressão do gene.

Tipicamente, os FTs não se ligam sozinhos, eles se ligam em complexos como o da figura 2.4(b). Uma vez ligados ao DNA, o complexo FT habilita a ligação da RNA polimerase (RNAP) na região antes da região codificante (*upstream*), como na figura 2.4(c). A RNAP forma o complexo de transcrição que separa as duas fitas do DNA, formando um complexo aberto, e então ela se move em uma fita de DNA, passo a passo (base por base), e transcreve a região codificante em mRNA, como na figura 2.4(d).

FIGURA 2.5 – A tradução consiste no movimento do ribossomo pelo mRNA, um códon (três bases) por vez, traduzindo cada códon de mRNA em um aminoácido da proteína final.

2.2.2 Tradução

A tradução acontece no citoplasma (eucariotos) onde o mRNA se liga com os ribossomos, formando um complexo de macro-moléculas cuja função é criar proteínas. O ribossomo se move pelo mRNA de três em três bases, cada combinação de três bases, ou códon, é traduzida em um dos vinte aminoácidos (figura 2.5).

A função do ribossomo é ler a estrutura linear do mRNA e traduzir em uma seqüência linear de aminoácidos. Conforme este processo acontece, a seqüência linear de aminoácidos vai se dobrando até a estrutura tridimensional da proteína. Uma proteína pode fazer a conformação por si mesmo, ou pode requerer a assistência de outras proteínas.

2.2.3 *Exons/Introns* e *Splicing*

Em procariotos, a região codificante é contínua, mas em eucariotos ela é tipicamente dividida em várias partes. Cada parte codificante é chamada de *exon*, e as partes entre os *exons* são chamadas de *introns* (figura 2.6(a)). Entre a transcrição e a tradução, nos eucariotos, o mRNA precisa ser movido fisicamente de dentro do núcleo para fora dele. Como parte desse processo, os *introns* são removidos, essa edição é chamada de *splicing*.

Em alguns casos, existem *splicings* alternativos, isto é, o mesmo pedaço de DNA pode ser editado de diferentes maneiras para formar proteínas diferentes (figura 2.6(b)). Em eucariotos, após o processo de *splicing* e transporte para o citoplasma o mRNA está pronto para ser traduzido. Em procariotos, os processos de transcrição

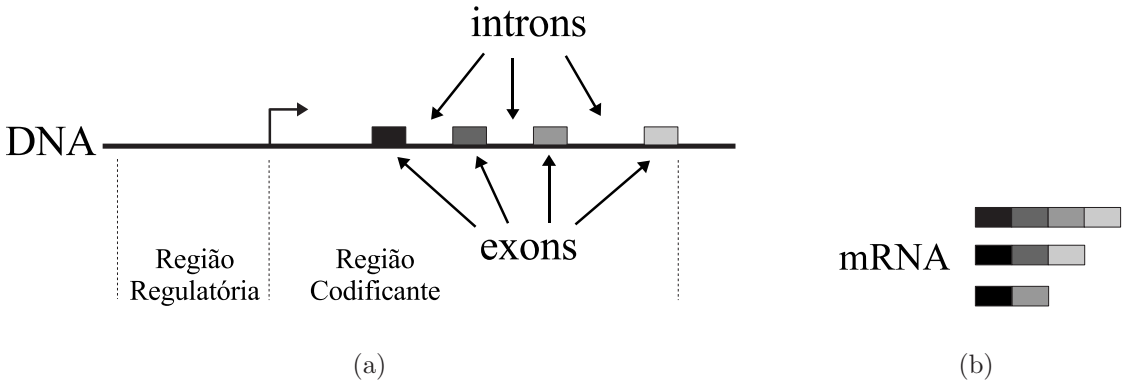


FIGURA 2.6 – (a) O DNA eucariotico possui *exons* e *intros*. (b) Diferentes seqüências de mRNA podem ser geradas a partir do mesmo gene.

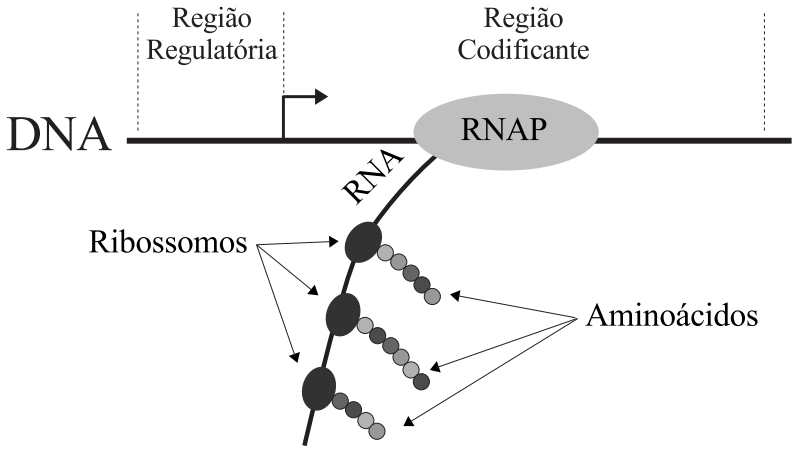


FIGURA 2.7 – No DNA dos procariotos os processos de transcrição e tradução ocorrem concomitantemente. Os aminoácidos traduzidos formam as proteínas no final da tradução.

e tradução ocorrem concomitantemente (figura 2.7).

2.3 Reações Bioquímicas

As reações químicas são a linguagem canônica da modelagem biológica, também conhecidas como reações bioquímicas. Elas provêm uma notação unificada para expressar qualquer processo químico complexo. Considere a reação

$$n_a A + n_b B \xrightarrow{k} n_c C + n_d D \tag{2.1}$$

neste exemplo, n_a moléculas da espécie A reagem com n_b moléculas da espécie B e se transformam em n_c moléculas das espécies C e n_d moléculas da espécie D. Os termos à esquerda da seta são chamados de reagentes e os termos à direita de produtos. Pode haver um número arbitrário de reagentes e produtos em uma reação química. O valor k, em cima da seta, é a taxa na qual a reação ocorre, e está relacionado com fatores como temperatura e volume. Os termos n são chamados coeficientes estequiométricos e usualmente são números naturais pequenos.

A determinação das taxas das reações k que fazem parte do metabolismo é um tópico de pesquisa clássico em bioquímica. O estudo das rotas bioquímicas através de modelos cinéticos e simulação em computador dependem do conhecimento das taxas das reações. O objetivo final do estudo das rotas bioquímicas é entender a dinâmica de uma célula viva em termos da interação entre seus componentes moleculares (KIERZEK, 2002).

As reações bioquímicas acontecem constantemente em todos os organismos vivos e a maioria delas envolvem proteínas chamadas enzimas, as quais agem como eficientes catalizadores. As enzimas reagem com compostos específicos chamados substratos. Por exemplo, a hemoglobina é uma enzima nas células do sangue vermelho e o oxigênio é o substrato com o qual ela reage. As enzimas exercem um papel importante na regulação dos processos biológicos como ativadoras ou inibidoras de uma reação. Para entender as reações bioquímicas estudamos a cinética das enzimas, a qual consiste basicamente das taxas das reações, do comportamento temporal e das condições que influenciam os vários reagentes (MURRAY, 2002).

Na seção 2.3.1 será analisada em detalhe uma reação enzimática simples. Enquanto nas seções 2.3.2 e 2.3.3 serão mostrados outros tipos e exemplos de reações enzimáticas mais complexas, para as quais pode-se utilizar o mesmo tipo de análise matemática usado na seção 2.3.1.

2.3.1 Reação Enzimática Básica

Uma das reações enzimáticas mais simples, proposta por Michaelis e Menten (MICHAELIS; MENTEN, 1913), envolve um substrato S reagindo com a enzima E para formar o complexo SE que por sua vez é convertido no produto P e na enzima E. Esta reação enzimática pode ser representada esquematicamente por

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} SE \xrightarrow{k_2} P + E \tag{2.2}$$

Onde k_1, k_{-1} e k_2 são constantes associadas com as taxas da reação. O símbolo da seta dupla $\rightleftharpoons$ indica que a reação é reversível enquanto o da seta simples $\rightarrow$ indica que a reação somente acontece em uma direção. O mecanismo geral é a conversão do substrato S, por catálise com a enzima E, em um produto P. Mais detalhadamente, uma molécula de S se liga a uma molécula de E para formar uma de SE, a qual eventualmente produz uma molécula de P e uma molécula de E.

A *Lei da Ação das Massas* afirma que a taxa de uma reação é proporcional ao produto das concentrações dos reagentes. Denotam-se as concentrações dos reagentes em (2.2) pelas letras minúsculas

$$s = [S], \quad e = [E], \quad c = [SE], \quad p = [P], \tag{2.3}$$

onde [ ] tradicionalmente representa concentração. A *Lei da Ação das Massas* aplicada a (2.2) leva a uma equação para cada reagente e a um sistema de equações diferenciais não linear (vide seção 3.1)

$$\frac{ds}{dt} = -k_1 es + k_{-1} c, \tag{2.4}$$

$$\frac{de}{dt} = -k_1 es + (k_{-1} + k_2)\, c, \tag{2.5}$$

$$\frac{dc}{dt} = -k_1 es - (k_{-1} + k_2)\, c, \tag{2.6}$$

$$\frac{dp}{dt} = k_2 c. \tag{2.7}$$

Os k's, chamados taxas, são constantes de proporcionalidade na aplicação da *Lei da Ação de Massa*. Por exemplo, a primeira equação para s simplesmente estabelece que a taxa na qual a concentração $[S]$ varia é formada por uma perda na taxa proporcional a $[S]\,[E]$ e um ganho na taxa proporcional a $[SE]$.

Para completar a formulação matemática são necessárias as condições iniciais, as condições do início do processo de conversão de S para P, então

$$s\,(0) = s_0, \quad e\,(0) = e_0, \quad c\,(0) = 0, \quad p\,(0) = 0. \tag{2.8}$$

As soluções das equações (2.4), (2.5), (2.6) e (2.7) dadas as concentrações iniciais (2.8) e as taxas das reações fornece as concentrações dos reagentes e dos produtos em função do tempo. Nos problemas de cinética das reações preocupa-se somente com concentrações não negativas.

A equação (2.14) é desacoplada das demais e gera o produto

$$p\,(t) = k_2 \int_0^t c(t')dt' \tag{2.9}$$

quando $c\,(t)$ é determinado, então só é necessário preocupar-se (analiticamente) com as três primeiras equações (2.4), (2.5) e (2.6).

O mecanismo da enzima E em (2.2) é uma catálise, ele facilita o acontecimento da reação. A concentração total da enzima E, livre mais combinada, é constante. Esta lei de conservação para a enzima também aparece na adição da segunda equação (2.5), e livre, e na terceira equação (2.6), c combinada, tem-se

$$\frac{de}{dt} + \frac{dc}{dt} = 0 \quad \Rightarrow \quad e\,(t) + c\,(t) = e_0 \tag{2.10}$$

usando as condições iniciais (2.8). Com isto, o sistema de equações diferenciais

ordinárias fica somente com duas equações, uma para s e outra para c,

$$\begin{aligned} \frac{ds}{dt} &= -k_1 e_0 s + (k_1 s + k_{-1})\, c, \\ \frac{dc}{dt} &= k_1 e_0 s - (k_1 s + k_{-1} + k_2)\, c, \end{aligned} \tag{2.11}$$

com as condições iniciais

$$s\,(0) = s_0, \quad c\,(0) = 0. \tag{2.12}$$

O modo usual de tratar essas equações é assumir que o estágio inicial da formação do complexo, c, é muito rápido e depois disso ele permanece em equilíbrio. Ou seja, $dc/dt \approx 0$ no caso da segunda equação em (2.11) obtém-se c em termos de s,

$$c\,(t) = \frac{e_0 s\,(t)}{s\,(t) + K_m}, \qquad K_m = \frac{k_{-1} + k_2}{k_1} \tag{2.13}$$

substituindo na primeira equação em (2.11) temos

$$\frac{ds}{dt} = -\frac{k_2 e_0 s}{s + K_m} \tag{2.14}$$

onde K_m é chamado constante de *Michaelis*. Considera-se tradicionalmente que a enzima está presente em pequenas quantidades comparado com os substratos, e com isso assume-se que as concentrações dos substratos não muda significantemente durante a fase transiente inicial. Neste caso a dinâmica (aproximada) é governada por (2.14) com condição inicial $s = s_0$. Esta aproximação é conhecida como estado estacionário. Resolvendo (2.14) com a condição inicial em $s\,(t)$ obtém-se uma solução implícita,

$$s\,(t) + K_m \ln s\,(t) = s_0 + K_m \ln s_0 \tag{2.15}$$

Agora substituindo a solução implícita (2.15) na equação em (2.13) obtém-se a expressão para o complexo $c\,(t)$. Mas isto não satisfaz a condição inicial em $c\,(t)$ na

equação (2.12). Entretanto, pode-se assumir que esta é uma aproximação razoável para quase todo o tempo, e de fato para muitas situações experimentais é suficiente. Existem duas escalas de tempo envolvidas neste sistema: a do transiente inicial perto de $t = 0$ e a outra escala de longo prazo quando a concentração do substrato muda significantemente, período o qual o complexo enzimático é razoavelmente aproximando por (2.13) com $s(t)$ da equação (2.15).

2.3.2 Cinética do Substrato Suicida

O mecanismo baseado em inibidor, ou cinética do substrato suicida, foi representado por (WALSH; WANG, 1978),

$$\begin{array}{lllll} E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} X & \xrightarrow{k_2} & Y & \xrightarrow{k_3} E + P \\ & & \downarrow & k_4 & \\ & & E_i & & \end{array} \tag{2.16}$$

onde E, S e P representam enzima, substrato e produto, respectivamente, X e Y intermediários enzima-substrato, E_i enzima inativa. Neste sistema, Y pode escolher um entre dois caminhos, $E + P$ com taxa k_3 ou E_i com taxa k_4. A razão destas taxas, k_3/k_4, é chamada taxa de partição (r). Os dois caminhos são considerados irreversíveis durante a escala de tempo da reação. O S é conhecido como substrato suicida, pois ele se liga ao sítio ativo da enzima mas ao invés de se converter em P ele irá se converter em E_i inativando a enzima e agindo como um inibidor da reação.

O sistema de equações diferenciais que representa as reações (2.16) é

$$\begin{aligned}
\frac{d[S]}{dt} &= -k_1[E][S] + k_{-1}[X], \\
\frac{d[E]}{dt} &= -k_1[E][S] + k_{-1}[X] + k_3[Y], \\
\frac{d[X]}{dt} &= k_1[E][S] - k_{-1}[X] - k_2[X], \\
\frac{d[Y]}{dt} &= k_2[X] - k_3[Y] - k_4[Y], \\
\frac{d[E_i]}{dt} &= k_4[Y], \\
\frac{d[P]}{dt} &= k_3[Y].
\end{aligned} \tag{2.17}$$

2.3.3 Fenômeno Cooperativo

Na reação 2.2 uma molécula de enzima combina-se com um substrato e a enzima possui um único sítio de ligação. Existem muitas enzimas que possuem mais de um sítio de ligação para moléculas de substratos. Por exemplo, a hemoglobina (Hb), proteína que carrega o oxigênio (O_2) nas células do sangue vermelho, possui quatro sítios de ligação para as moléculas de oxigênio. Uma reação entre uma enzima e um substrato é dita cooperativa se uma única molécula de enzima pode depois de ligar uma molécula de substrato em um sítio, ligar outra molécula de substrato em outro sítio. Este é um fenômeno comum.

Outro comportamento importante é quando a ligação de uma molécula de substrato em um sítio de uma enzima com vários sítios de ligação afeta a atividade de ligação nos outros sítios da enzima. Esta interação indireta entre os sítios é chamada alosteria, ou efeito alostérico. Se a ligação de um substrato em um sítio aumenta a atividade em outro sítio o substrato é chamado de ativador, e se diminui de inibidor.

O modelo consiste em uma enzima E que se liga ao substrato S para formar um complexo C_1. O complexo C_1 não somente se quebra para formar o produto P e a enzima E de novo como ele também pode-se combinar com outra molécula de substrato para formar o complexo duplo C_2. O complexo C_2 quebra-se e forma o

produto P e o complexo simples C_1. O mecanismo de reação para este modelo é

$$\begin{aligned} E + S &\underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C_1 \xrightarrow{k_2} E + P \\ S + C_1 &\underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} C_2 \xrightarrow{k_4} C_1 + P. \end{aligned} \tag{2.18}$$

E o sistema de equações diferenciais que o representa é

$$\begin{aligned} \frac{d[S]}{dt} &= -k_1[E][S] + k_{-1}[C_1] - k_3[S][C_1] + k_{-3}[C_2], \\ \frac{d[C_1]}{dt} &= k_1[E][S] - k_{-1}[C_1] - k_2[C_1] - k_3[S][C_1] + k_{-3}[C_2] + k_4[C_2], \\ \frac{d[C_2]}{dt} &= k_3[S][C_1] - k_{-3}[C_2] - k_4[C_2], \\ \frac{d[E]}{dt} &= -k_1[E][S] + k_{-1}[C_1] + k_2[C_1], \\ \frac{d[P]}{dt} &= k_2[C_1] + k_4[C_2]. \end{aligned} \tag{2.19}$$

2.4 Modelo da Cinética Intracelular Viral

Um modelo simples da rede metabólica de um vírus, como o da figura 2.8, foi desenvolvido para explorar as diferenças entre as implementações do modelo determinístico e estocástico no trabalho (SRIVASTAVA; YOU; YIN, 2002). Os componentes estudados são os ácidos nucléicos virais e uma proteína estrutural viral (*estrutura*). O ácidos nucléicos são classificados como genômico (*gen*) ou fita molde (*fita*). O genoma, que pode ser DNA, RNA fita positiva, RNA fita negativa ou outra variante, é o veículo pelo qual a informação genética do vírus é transportada.

O genoma segue um de dois destinos. No primeiro, ele pode ser modificado, tanto por integração no genoma do hospedeiro ou algum outro tipo de processamento, por exemplo transcrição reversa, para formar a fita molde. A fita molde refere-se a forma de ácido nucléico que é transcrito e está envolvida na catálise da síntese de cada componente viral. No segundo destino, o genoma pode ser empacotado junto com proteínas estruturais para formar descendentes do vírus.

Uma seqüência comum de eventos na replicação viral envolve o aumento da

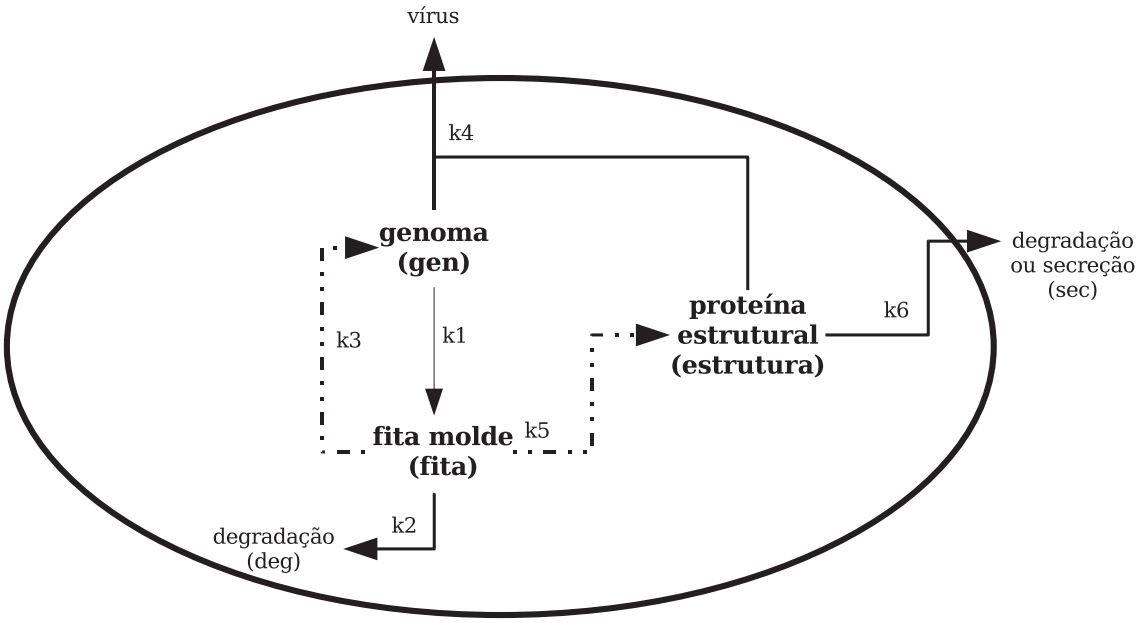


FIGURA 2.8 – Modelo do ciclo de replicação viral. A *fita* somente é usada para catalise da síntese do *gen* e da *estrutura*. As reações catalíticas são representadas por linhas pontilhadas.

concentração de fitas moldes do vírus depois da infecção, seguido da produção de descendentes do vírus. Os vírus de DNA, por exemplo, inicialmente produzem um baixo nível de proteínas não estruturais que catalizam e aumentam o número de moléculas de fita molde. Uma vez que o número de moléculas de fita molde esteja suficientemente alto, proteínas estruturais são sintetizadas e incorporadas nas partículas dos descendentes.

O tempo dos eventos de replicação podem estar ligados a fatores externos. Neste modelo, o chaveamento entre aumento do genoma viral e produção de vírus é regulado através da cinética da ação de massa. Pois, o chaveamento pode ser matematicamente alcançado através da dinâmica de alguns componentes virais, sem nenhuma dependência explícita do hospedeiro. Entretanto, a dependência implícita do hospedeiro é refletida nos valores dos parâmetros cinéticos do modelo. Por exemplo, se em uma determinada célula hospedeira um certo aminoácido está escasso, pode-se refletir este fato usando taxas menores para a síntese de proteínas.

O modelo de rede empregado foi condensado, no sentido de que muitas reações individuais foram combinadas numa única reação. Por exemplo, a síntese de proteínas

TABELA 2.1 – Parâmetros Cinéticos do Modelo: Cinética Intracelular Viral

Parâmetro	**Valor**
k_1	0.025 day^{-1}
k_2	0.25 day^{-1}
k_3	1.0 day^{-1}
k_4	7.5×10^{-6} $\text{moléculas}^{-1}\text{day}^{-1}$
k_5	1000 day^{-1}
k_6	1.99 day^{-1}

estruturais requer que o DNA viral seja transcrito em mRNA e o mRNA seja traduzido numa proteína estrutural. Para o modelo viral, assume-se que estas reações podem ser combinadas e caracterizadas usando uma taxa única como parâmetro.

Para desenvolvimento deste modelo também assume-se: (1) os outros componentes necessários para as reações ocorrerem estão disponíveis em nível constante; (2) depois de ser infectada, uma célula não pode ser infectada novamente, mesmo que a infecção inicial seja degradada; (3) reagentes estão distribuídos de maneira homogênea no espaço; (4) na infecção inicial a célula possui um número configurado anteriormente de fitas molde viral; (5) as células são diferenciadas e não se dividem.

A tabela 2.1 contém os parâmetros cinéticos usados nas simulações do modelo. As equações diferenciais que representam o modelo são:

$$\frac{d\left[gen\right]}{dt} = k_3\left[fita\right] - k_1\left[gen\right] - k_4\left[gen\right]\left[estrutura\right], \tag{2.20}$$

$$\frac{d\left[fita\right]}{dt} = k_1\left[gen\right] - k_2\left[fita\right], \tag{2.21}$$

$$\frac{d\left[estrutura\right]}{dt} = k_5\left[fita\right] - k_6\left[estrutura\right] - k_4\left[gen\right]\left[estrutura\right]. \tag{2.22}$$

As reações bioquímicas que representam o modelo descrito na figura 2.8 são:

$$gen \xrightarrow{k_1} fita \tag{2.23}$$

$$fita \xrightarrow{k_2} deg \tag{2.24}$$

$$fita \xrightarrow{k_3} gen \tag{2.25}$$

$$gen + estrutura \xrightarrow{k_4} virus \tag{2.26}$$

$$fita \xrightarrow{k_5} estrutura \tag{2.27}$$

$$estrutura \xrightarrow{k_6} sec \tag{2.28}$$

onde as reações 2.25 e 2.27 são catalíticas. Isto implica que elas não precisam ter nenhuma molécula de fita molde para reagirem. Além disso, quanto mais moléculas de fita molde existirem, maior será a chance delas acontecerem, fazendo com que o número de moléculas de genoma e estrutura sempre aumente.

2.5 Modelo do fago λ

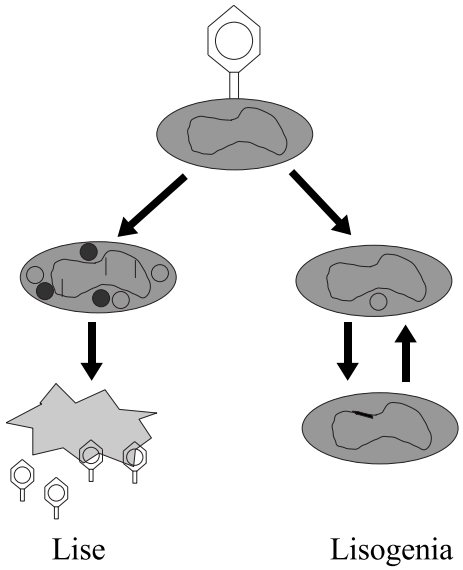


FIGURA 2.9 – O ciclo de vida do fago λ. O fago injeta seu DNA na *E. coli* alvo, podendo se replicar e dissolver a célula hospedeira (lise), ou integrar seu DNA no DNA da célula hospedeira (lisogenia).

O fago λ é um vírus que infecta as células da *Escherichia coli*. Na figura 2.9 está representado duas vias possíveis de desenvolvimento do vírus. Ele pode tanto se replicar e dissolver (lise) a célula hospedeira, liberando cerca de 100 descendentes, quanto integrar seu DNA ao DNA bactériano e formar uma lisogenia. No último caso, o vírus irá se replicar passivamente toda vez que a bactéria se replicar. Uma lisogenia também possui imunidade contra futuras infecções do fago λ, protegendo-se do próprio vírus que poderia destruir a lisogenia através de outra infecção que resultasse em lise. Sobre as condições certas, uma lisogenia pode ser induzida, isto é, o DNA viral se remove do DNA bacteriano replicando-se normalmente e dissolvendo a célula hospedeira (BOWER; BOLOURI, 2001).

O fago λ vem sendo estudado exaustivamente, pois, ele é um dos organismos mais simples com sistema de múltiplos estados finais. Seu genoma foi seqüenciado e os detalhes de quais genes são expressados, quando e em qual quantidade, tendo como hospedeiro a *Escherichia coli* foram estudados. Como resultado o fago λ é muito bem compreendido. Ele também é um organismo que exibe comportamento probabilístico ou estocástico, onde algumas células infectadas acabam em lise e outras em lisogenia. Por isto, ele é o sistema apropriado para demonstrar o comportamento estocástico da regulação gênica (BOWER; BOLOURI, 2001). Alguns dos principais trabalhos que estudam a cinética do metabolismo do λ através de simulação são (ARKIN; ROSS; MCADAMS, 1998), (GIBSON; BRUCK, 2000), (BOWER; BOLOURI, 2001) e (SANTILLáN; MACKEY, 2004).

Na figura 2.10 são mostradas as duas fitas de DNA do fago λ, sendo que o tamanho dos genes, promotores, operadores, terminadores e DNA não estão em escala. Os genes estão representados por retângulos em cinza claro (N, CRO, CI, CII, CIII, P, Q), os promotores por retângulos cinza escuro (PR, PRM, PL, PRE), os operadores por retângulos com linhas diagonais (OR_1, OR_2, OR_3, OE_1, OE_2, OL_1, OL_2), os terminadores por triângulos (TL_1, TL_2, TR_1, TR_2) e os sítios de utilização-N por círculos (NUTR e NUTL). Na fita+ do DNA a RNA polimerase caminha no sentido da esquerda para a direita e na fita− do DNA no sentido da

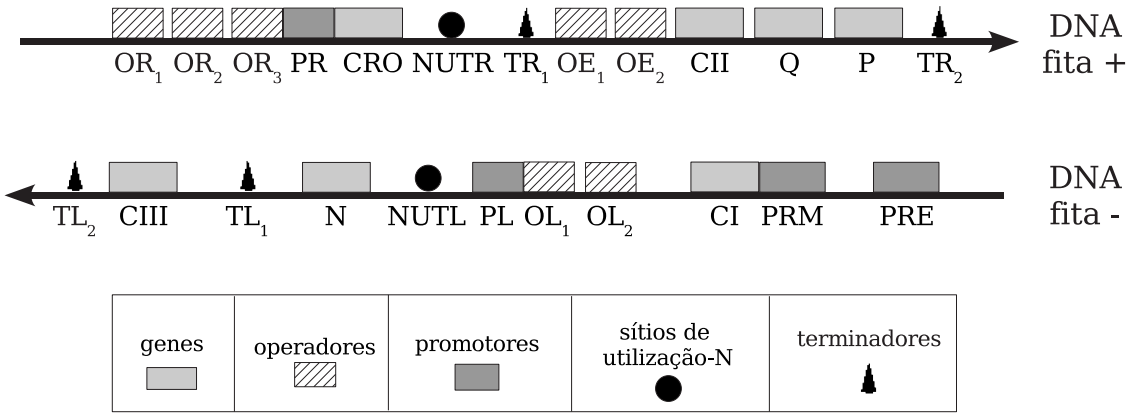


FIGURA 2.10 – DNA do fago λ

direita para a esquerda.

A RNA polimerase pode iniciar a transcrição se ligando a qualquer um dos promotores. Uma vez iniciada a transcrição em um determinado promotor, a RNA polimerase caminha pela fita do promotor na direção correspondente da fita. Quando a RNA polimerase atinge o final de um gene é produzida uma molécula de mRNA, mas ela continua colada ao DNA, caminhando sobre o mesmo. Ao passar por um terminador, a RNA polimerase possui uma probabilidade de sair do DNA. Quando a RNA polimerase passa pelos sítios de utilização-N ela caminha mais devagar e a proteína N pode ou não se ligar à RNA polimerase, mudando a probabilidade da mesma sair do DNA nos terminadores.

Capítulo 3

Simulação de Redes Metabólicas

Nas seções seguintes, serão apresentados alguns dos métodos de simulação existentes para os sistemas de reações bioquímicas acopladas capazes de simular os modelos descritos na seções 2.4 e 2.5. A seção 3.1 apresenta o modelo de simulação contínua, a seção 3.2 os modelos de simulação discreta e seção 3.3 o modelo de simulação híbrida.

3.1 Simulação Contínua ou Determinística

Realizando suposições como um número grande de reagentes e reações rápidas é possível transformar qualquer sistema de reações químicas em um sistema de equações diferenciais ordinárias não lineares, onde as variáveis são concentrações (BOWER; BOLOURI, 2001). Uma maneira de fazer isto é através da cinética das reações, considerando as taxas de transição entre todos os estados microscópicos.

Neste modelo o estado é uma lista das concentrações de cada espécie. Assume-se que estas concentrações são contínuas, e variam no tempo de acordo com equações diferenciais. Considere a seguinte reação química:

$$A \xrightarrow{k_1} A' \tag{3.1}$$

ela representa a reação que ocorre na taxa $k\,[A]$, por exemplo, à medida que a

concentração de A ($[A]$) aumenta existe um acréscimo linear na taxa em que A' é criado. Mais precisamente,

$$\frac{d[A]}{dt} = -k[A] \tag{3.2}$$
$$\frac{d[A']}{dt} = k[A'] \tag{3.3}$$

conforme a reação acontece $[A]$ diminui e $[A']$ aumenta. A taxa da reação depende somente dos reagentes, sendo que em reações com mais de um reagente a taxa da reação é a constante k vezes o produto das concentrações de cada reagente. Por exemplo,

$$A + B \xrightarrow{k} AB \tag{3.4}$$

gera as equações diferenciais:

$$\frac{d[A]}{dt} = -k[A][B] \tag{3.5}$$
$$\frac{d[B]}{dt} = -k[A][B] \tag{3.6}$$
$$\frac{d[AB]}{dt} = k[A][B] \tag{3.7}$$

Nas reações que possuem algum reagente com coeficiente estequiométrico diferente a equação diferencial correspondente tem que levar isto em conta, elevando a concentração do reagente à potência do valor do coeficiente estequiométrico e multiplicando pelo valor do coeficiente estequiométrico. Por exemplo,

$$2A \xrightarrow{k} B \tag{3.8}$$

gera as seguintes equações diferenciais:

$$\frac{d[A]}{dt} = -k \times 2 \times [A]^2 \tag{3.9}$$
$$\frac{d[B]}{dt} = k \times 2 \times [A]^2 \tag{3.10}$$

onde a taxa de decréscimo de A e de acréscimo de B são iguais a duas vezes a concentração de A ao quadrado.

Para resolvermos o sistema de equações diferenciais precisamos das condições iniciais além das próprias equações. A maioria dos sistemas de equações diferenciais correspondentes às reações químicas acopladas que representam algum sistema da regulação gênica ou outra parte do metabolismo são muito complicados para serem resolvidos analiticamente.

3.2 Simulação Discreta ou Estocástica

A evolução de um sistema de reações químicas acopladas pode ser feita utilizando um modelo estocástico. A probabilidade de uma dada molécula da espécie A reaja com uma molécula da espécie B num curto espaço de tempo dt é calculada por $k_1 dt + o\,(dt)$. Considere, por exemplo, o seguinte conjunto de reações:

$$A + B \xrightarrow{k_1} C \tag{3.11}$$

$$B + C \xrightarrow{k_2} D \tag{3.12}$$

$$D + B \xrightarrow{k_3} A \tag{3.13}$$

Uma possível maneira de calcular a probabilidade de uma reação ocorrer seria marcar cada molécula da espécie A com $A_1, A_2, ..., A_{\#A}$, cada molécula da espécie B com $B_1, B_2, ..., B_{\#B}$, cada molécula da espécie C com $C_1, C_2, ..., C_{\#C}$ e cada molécula da espécie D com $D_1, D_2, ..., D_{\#D}$. Após todas as moléculas terem recebido uma marca existirão $(\#A) \times (\#B)$ cópias distintas da reação (3.11), $(\#B) \times (\#C)$ cópias distintas da reação (3.12) e $(\#D) \times (\#B)$ cópias distintas da reação (3.13). Onde cada cópia da reação (3.11) terá a mesma probabilidade de acontecer, assim como cada cópia das reações (3.12) e (3.13).

Neste modelo estocástico consideramos que as reações estão ocorrendo dentro de soluções. Este fato permite agrupar as reações por espécies de moléculas, pois, a posição onde as reações ocorrem não importa. Então, as $(\#A) \times (\#B)$ cópias da

reação (3.11) podem ser agrupadas em uma única reação, a qual possui densidade de probabilidade $k_1 \times (\#A) \times (\#B)\, dt + o\,(dt)$.

O estado do sistema é definido pelo número de moléculas de cada espécie, o qual muda discretamente toda vez que uma reação é executada. Por exemplo, o estado $E = (\#A, \#B, \#C, \#D)$ se transformará no estado $E' = (\#A - 1, \#B - 1, \#C + 1, \#D)$ se a reação (3.11) for executada. A probabilidade disto acontecer é dada por:

$$P\,(E', t + dt|E, t) \;\; = \;\; a_1 dt + o\,(dt) \tag{3.14}$$

onde, a probabilidade de estar no estado E e ir para o estado E' no próximo instante de tempo dt é $a_1 dt + o\,(dt)$.

Usualmente a maneira de lidar com o modelo estocástico é criar uma variável para cada estado $(\#A, \#B, \#C, \#D)$ possível. Então, a probabilidade de estar em um estado X é uma função do tempo condicionada ao estado inicial e tempo inicial do sistema:

$$P(X, t|X_0, t_0). \tag{3.15}$$

O tempo entre eventos de reações Y, do sistema nos estados X é exponencialmente distribuído com a taxa característica dada pela densidade de probabilidade, $a_r\,(X)$. Intuitivamente, o sistema vai para o estado X no tempo t depois de ter passado pelo estado $X - Y_r$, no tempo $t - \Delta$ e realiza a transição para X com probabilidade $a_r\,(X - Y)$. Ou, ele já está no estado X com a probabilidade de nenhuma reação acontecer durante o período $[t, t_d t)$. Escrevendo na forma diferencial, obtemos a Equação Mestra Química:

$$\frac{\partial P\,(X, t|X_0, t_0)}{\partial t} = \sum_{r=1}^{R} \left[\alpha_r(X - Y_r) P(X - Y_r, t|X_0, t_0) - \alpha_r(X) P(X, t|X_0, t_0)\right]. \tag{3.16}$$

A simplicidade desta expressão esconde a dificuldade encontrada em obter soluções analíticas.

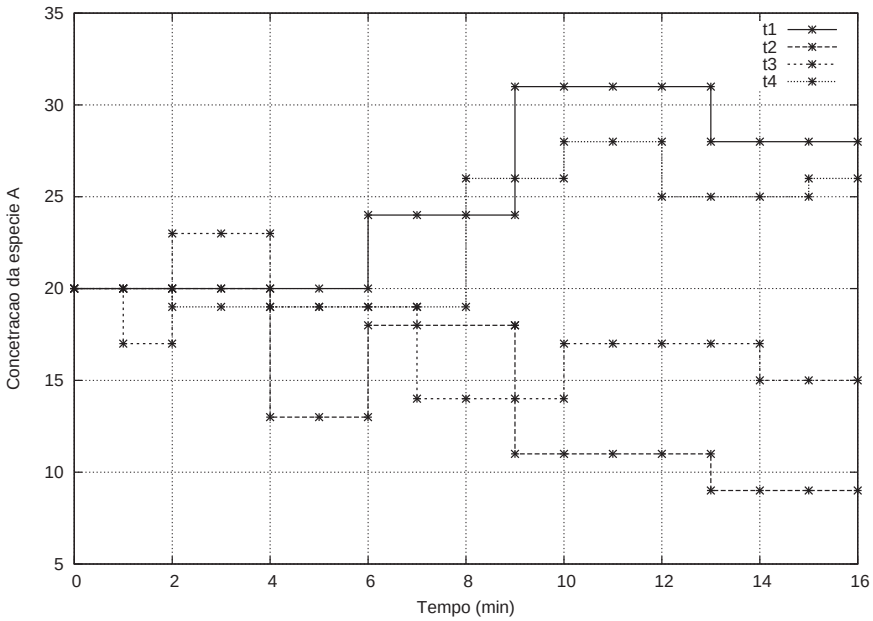


FIGURA 3.1 – Trajetórias exemplo da espécie A.

As taxas das reações podem diferir entre o modelo discreto e determinístico. Para as reações na forma $X \xrightarrow{k} produto$, a taxa determinística k_{det} é igual à taxa estocástica k_{est}, $k_{det} = k_{est}$. Nas reações na forma $X + Y \xrightarrow{k} produto$ as taxas são $k_{est} = k_{det}/(A_g V_{cel})$, onde A_g é o número de Avogado e V_{cel} o volume da célula. Por último, para as reações no formato $X + X \xrightarrow{k} produto$ as taxas ficam $k_{est} = k_{det}/(2A_g V_{cel})$.

3.2.1 Algoritmos de Gillespie

No método estocástico exato considera-se o problema de gerar uma única trajetória, uma seqüência possível de reações. Ao contrário da Equação Mestra Química que resolve um sistema de equações para a probabilidade de todas as trajetórias possíveis, a geração de uma única trajetória é relativamente simples, sendo necessária apenas a seqüência das transições de estados e os tempos nos quais elas ocorreram. No gráfico da figura 3.1 mostramos quatro trajetórias exemplo para a espécie A.

Um modo eficiente de gerar trajetórias é escolher as reações e tempos de acordo com suas distribuições de probabilidades corretas. Então, a probabilidade deste

algoritmo gerar uma dada trajetória é exatamente a probabilidade calculada pela Equação Mestra Química para esta trajetória.

É possível criar um algoritmo que escolhe as reações e tempos de acordo com suas distribuições de probabilidades corretas mesmo quando não for possível escrever a Equação Mestra Química completa e resolvê-la. Para achar o número médio de moléculas de um composto em determinado tempo pode-se executar a simulação diversas vezes gerando várias trajetórias, e calcular a média de moléculas do composto num determinado tempo.

Gillespie (GILLESPIE, 1977) apresentou dois algoritmos estocásticos exatos, onde a cada período de tempo o sistema está em um único estado exato. Uma transição consiste em executar uma reação, tendo no máximo r (número de reações) possíveis transições a partir de um estado. O segredo está em escolher a reação que será executada usando geradores de números randômicos com a distribuição de probabilidade correta.

3.2.2 Método Direto

O método direto calcula explicitamente qual reação ocorre e em qual tempo, por isto, ele é chamado direto. Para um sistema em um determinado estado este método responde as seguintes questões:

- Qual será a próxima reação a ocorrer ?
- Quando ela irá ocorrer ?

Estas questões podem ser respondidas probabilisticamente especificando a densidade de probabilidade $P(r,t)$ para a próxima reação r que ocorre no tempo t (GIBSON; BRUCK, 2000). Pode ser mostrado que

$$P(r,t)\,dt \;=\; a_r \exp\left(-t\sum_j a_j\right) dt \tag{3.17}$$

no algoritmo 3.2.1 apresenta-se os passos para calcular qual será a próxima reação e quando ela irá ocorrer usando as densidades de probabilidades das reações.

Algoritmo 3.2.1

1. Inicializa, determina o tempo (T) e o número de moléculas de cada composto.

2. Calcula a função densidade de probabilidade a_i, para todas as reações.

3. Escolhe r de acordo com a distribuição:

$$Pr\,(r) \;=\; \frac{a_r}{\sum_j a_j} \tag{3.18}$$

4. Escolhe t de acordo com uma distribuição exponencial:

$$P\,(t) \;=\; \left(\sum_j a_j\right) \exp\left(-t \sum_j a_j\right) dt \tag{3.19}$$

5. Altera o número de moléculas para refletir a execução da reação r e o tempo para $T = T + t$.

6. Volte para o passo 2.

Ao iniciar a simulação, é definido qual será o tempo inicial e a quantidade de moléculas cada composto possui. Após realizada a inicialização, executa-se até atingir o tempo final de simulação os passos descritos a seguir. Primeiramente, calculam-se as densidades de probabilidade de todas as reações, guardando a soma total delas (DP_{total}). Após, gera-se dois números randômicos (NR_1 e NR_2), tirados de uma distribuição uniforme entre $(0, 1)$. Escolhe-se a reação que acontecerá usando o algoritmo da roleta. Em seguida, realiza-se a reação escolhida e avança-se a simulação em

$$\frac{-\log(NR_1)}{DP_{total}}$$

onde, NR_1 é número randômico que não foi usado no algoritmo da roleta.

A escolha de uma reação usando o algoritmo da roleta é feita da seguinte maneira. Calcula-se um valor aleatório entre $(0, DP_{total})$ através da formula $DP_{total} \times NR_2$. Percorre-se todas as reações somando na variável DP_{soma} o valor da densidade de probabilidade da reação corrente, se DP_{soma} for maior que $DP_{total} \times NR_2$ escolhe-se a reação corrente para reagir.

3.2.3 Método da Primeira Reação

O método da primeira reação é assim chamado por realizar a reação com menor tempo estimado t_i, tempo que cada reação ocorreria se nenhuma outra ocoresse primeiro. Apesar dos métodos da primeira reação e direto parecerem diferentes, foi provado em (GILLESPIE, 1977) que eles são equivalentes. No algoritmo 3.2.2, r será a reação com o menor tempo estimado e t será o tempo estimado da reação r.

Algoritmo 3.2.2

1. Inicializa, determina o tempo (T) e número de moléculas de cada composto inicial.

2. Calcula a função densidade de probabilidade a_i , para todas as reações.

3. Calcula o tempo estimado t_i para cada reação de acordo com uma distribuição exponencial com parâmetro a_i:

$$t_i = \frac{1}{a_i} \log\left(\frac{1}{rand}\right) \tag{3.20}$$

   onde $rand$ é um número randômico sorteado usando uma distribuição uniforme.

4. Atribua a r a reação com o menor t_i.

5. Atribua a t o valor de t_r.

6. Volte para o passo 2.

Ao iniciar a simulação, é definido qual será o tempo inicial e a quantidade de moléculas cada composto possui inicialmente. Após realizada a inicialização, executa-se até atingir o tempo final de simulação os passos descritos a seguir. Para cada reação, calcula-se a sua densidade de probabilidade (DP) e gera-se um número randômico (NR_1), através de uma distribuição uniforme entre $(0, 1)$. Após, calcula-se o tempo estimado da reação dado por:

$$\frac{1}{DP} \times \log\left(\frac{1}{NR_1}\right),$$

guardando o valor do menor tempo estimado e o identificador da sua reação correspondente. Depois de ter passado por todas reações, estará guardado o identificador da reação de menor tempo estimado e o seu respectivo tempo estimado. Então, realiza-se a reação de menor tempo estimado e avança-se a simulação para o tempo atual mais o tempo estimado para esta reação acontecer.

3.3 Simulação Híbrida

Os métodos exatos descritos nas seções anteriores aumentam o seu custo computacional em função do número de eventos de reações, ou seja, o número de reações que acontecem na simulação (GIBSON; BRUCK, 2000). Conforme sistemas mais complexos são modelados maior será capacidade computacional necessária. Mais precisamente, os métodos de Gillespie possuem complexidade $O(rE)$, onde r é o número de reações e E o número de eventos de simulação. O método híbrido proposto em (HASELTINE; RAWLINGS, 2002) resolve este problema para o caso em que as reações possuem velocidades com ordem de magnitude diferente. Este método divide o sistema em dois conjuntos de reações, o "rápido" e o "lento". As reações rápidas são aproximadas deterministicamente, enquanto as reações lentas são tratadas estocasticamente. Estas aproximações podem diminuir o custo computacional significantemente.

Nesta seção será descrito o algoritmo híbrido sem se preocupar com a for-

mulação matemática do problema de misturar os resultados determinísticos e estocásticos, que foi analisada em (HASELTINE; RAWLINGS, 2002).

O estado do sistema x modela a ocorrência de cada reação irreversível. O modelo da ocorrência de uma reação é consistente com o modelo de espécies de moléculas

$$n = n_0 + \nu^T x, \tag{3.21}$$

onde, assumindo que existem m ocorrências das reações e p espécies químicas: x é o estado do sistema em termos da ocorrência (um vetor-m), n é o número de moléculas de cada espécie (um vetor-p), n_0 o número inicial de moléculas de cada espécie (um vetor-p) e ν é a matriz estequiométrica (uma matriz-$m \times p$). Por exemplo,

$$A + B \xrightarrow{k_1} C + D \tag{3.22}$$

$$C \xrightarrow{k_2} 2A \tag{3.23}$$

$$D \xrightarrow{k_3} 3B \tag{3.24}$$

a matriz que representa estas reações é

$$\nu = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} -1 & -1 & 1 & 1 \\ 2 & 0 & -1 & 0 \\ 0 & 3 & 0 & -1 \end{array}\right] \end{array} \tag{3.25}$$

e

$$n = \begin{bmatrix} n_A \\ n_B \\ n_C \\ n_D \end{bmatrix} \qquad n_0 = \begin{bmatrix} n_A^0 \\ n_B^0 \\ n_C^0 \\ n_D^0 \end{bmatrix}. \tag{3.26}$$

Com isto, temos

$$\begin{bmatrix} n_A \\ n_B \\ n_C \\ n_D \end{bmatrix} = \begin{bmatrix} n_A^0 \\ n_B^0 \\ n_C^0 \\ n_D^0 \end{bmatrix} + \begin{bmatrix} -1 & -1 & 1 & 1 \\ 2 & 0 & -1 & 0 \\ 0 & 3 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{3.27}$$

que é equivalente a

$$\begin{aligned} n_A &= n_A^0 + 0 \\ n_B &= n_B^0 + 3 \\ n_C &= n_C^0 + 0 \\ n_D &= n_D^0 + 1 \end{aligned} . \tag{3.28}$$

3.3.1 Método Híbrido

O algoritmo para resolver a aproximação descrita nesta seção, usando simulação de Monte Carlo (NEWMAN; BARKEMA, 1999) escolhe randomicamente o próximo tempo (τ) e a próxima reação que irá reagir (μ) de sua densidade de probabilidade, da mesma maneira que o método direto.

Inicializa-se o sistema, atribuindo o tempo t igual a zero e o número espécies n igual a n_0. As reações são divididas em dois subconjuntos de x conforme o critério de partição definido para o modelo. O primeiro conjunto conterá as reações que ocorrem poucas vezes e o segundo as que ocorrem muitas vezes: o vetor-$(m-l)$ y e o vetor-l z, respectivamente. Ou seja, o subconjunto y representa as reações que ocorrem poucas vezes e o subconjunto z as reações que ocorrem muitas vezes.

O algoritmo do método híbrido 3.3.1 contém duas fases distintas, uma quando o critério de partição não foi atingido e outra quando foi. Na primeira, ele fica executando do passo 2 até o passo 4, usando Gillespie para o conjunto x todo, enquanto não atingir o critério de partição. Assim que ele atinge o critério ele troca de fase e começa a executar os passos 5 até 9, usando Gillespie para as reações lentas (y) e integrando as reações rápidas (z), enquanto o critério de partição for válido.

O critério de partição do estado x em reações "lentas" e "rápidas" deve ser intuitivo, segundo (HASELTINE; RAWLINGS, 2002). É recomendado manter pelo

menos duas ordens de magnitude de diferença entre as velocidades das reações dos dois subconjuntos y e z.

Algoritmo 3.3.1

1. Se o critério de partição foi atingido, vá para o passo 5.

2. Calcule: (a) as densidades de probabilidade das reações, $r_k = a_k(x)$, e (b) a densidade probabilidade total, $r_{tot} = \sum_{k=1}^{m} r_k$.

3. Escolha dois números randômicos, p_1 e p_2, de uma distribuição uniforme entre $(0, 1)$. Seja $\tau = -\log(p_1)/r_{tot}$, escolha j que

$$\sum_{k=1}^{j-1} r_k < p_2 \leq \sum_{k=1}^{j} r_k. \tag{3.29}$$

4. Seja $t \leftarrow t + \tau$. Seja $n \leftarrow n + \nu_j^T$, onde ν_j é a linha j de ν. Vá para o passo 1.

5. Para o subconjunto y, calcule (a) as densidades de probabilidade das reações, $r_k^y = a_k^y(y, z)$, e (b) a densidade de probabilidade total, $r_{tot}^y = \sum_{k=1}^{m-l} r_k^y$.

6. Escolha dois números randômicos, p_1 e p_2, de uma distribuição uniforme entre $(0, 1)$.

7. Seja $\tau = -\log(p_1)/r_{tot}$. Integre o subconjunto z no intervalo $[t, t + \tau)$ para determinar $\hat{\nu}_z = (v^z)^T[z(t + \tau) - z(t)]$. Seja $t \leftarrow t + \tau$ e $n \leftarrow n + \hat{\nu}_z$.

8. Calcule novamente as densidades de probabilidade das reações r_k^y e a densidade de probabilidade total $r_{tot}^y(t)$. Escolha j que

$$\sum_{k=0}^{j-1} r_k^y < p_2 r_{tot}^y(t) \leq \sum_{k=0}^{j} r_k^y. \tag{3.30}$$

9. Seja $n \leftarrow n + (\nu_j^y)^T$, onde ν_j^y é a linha j da ν^y. Vá para o passo 1.

Na inicialização da simulação é configurado o tempo inicial e máximo da mesma e a quantidade de moléculas de cada composto. Enquanto não for atingido o tempo máximo de simulação executam-se as seguintes etapas. Calcula-se a densidade de probabilidade de todas as reações, verifica se alguma reação pode ser processada no modelo determinístico através do critério de partição. Se nenhuma reação satisfazer o critério, segue-se como no método direto até avançar o tempo de simulação. Caso alguma reação satisfaça o critério, calcula-se a soma das densidades de probabilidade das reações "lentas" (DP_{lentas}). Depois, geram-se dois números randômicos (NR_1 e NR_2), tirados de uma distribuição uniforme entre $(0, 1)$. Então, é calculada via equações diferenciais a evolução da simulação do tempo atual até o tempo atual mais daqui a quanto tempo a reação lenta irá acontecer

$$\frac{-\log(NR_1)}{DP_{lentas}}$$

e calcula-se novamente DP_{lentas}. Escolhe-se segundo o algoritmo da roleta a reação que irá reagir do conjunto das reações "lentas". Reage-se a reação escolhida e avança-se a simulação para o tempo em que a reação lenta irá acontecer, calculado anteriormente.

Capítulo 4

Trabalhos Relacionados

Durante os últimos dois anos, período da confecção deste trabalho, muitos trabalhos que se relacionam a este surgiram. Tanto com relação a novos métodos de simulação possíveis de serem implementados no simulador, como novos simuladores, ferramentas e ambientes de simulação. Para citar alguns exemplos (DHAR et al., 2004b), (KIEHL; MATTHEYSES; SIMMONS, 2004), (GILLESPIE; PETZOLD, 2003) e (YOU; HOONLOR; YIN, 2003).

Neste capítulo, na seção 4.1 apresenta-se o tipo de computação de alto desempenho que será usado pelo simulador deste trabalho. Na seção 4.2 são apresentados os principais métodos de simulação estocástica de redes metabólicas existentes atualmente que não foram implementados neste trabalho. Já a seção 4.3 mostra alguns dos simuladores que realizam uma função similar ao simulador deste trabalho e algumas das principais iniciativas para simulação de uma célula completa, nas quais poderia se encaixar a arquitetura descrita neste trabalho como um módulo.

4.1 Computação de Alto Desempenho

Uma das definições de *Grid* que se adapta à maneira como este é utilizado pelo simulador deste trabalho é: computação em *Grid* habilita o compartilhamento de recursos distribuídos geograficamente por organizações virtuais que perseguem objetivos comuns, assumindo a ausência de uma localização central, controle central,

onisciência e a existência de uma relação confiável (ABBAS, 2004).

As organizações virtuais podem variar desde pequenos departamentos em uma mesma localização física a grandes organizações espalhadas pelo globo. Elas podem ser pequenas, grandes, estáticas ou dinâmicas. Uma entidade que pode ser compartilhada é um recurso. Num ambiente de *Grid* os recursos não possuem informação um do outro a priori. No ponto de vista de computação em *Grid*, um aglomerado de computadores (*cluster*) é um recurso compartilhado e um *Grid* pode ser considerado um aglomerado de aglomerados de computadores (*cluster of clusters*).

O sucesso da computação em *Grid* definitivamente será determinado pelos problemas que podem ser resolvidos com ela. Nem todas as aplicações podem tirar proveito da computação em *Grid*. A eficiência com que uma aplicação pode ser paralelizada determina se ela é ou não apropriada para computação em *Grid*. O paralelismo é uma propriedade inerente do aplicação, a qual pode ser divida em três categorias (ABBAS, 2004):

- Paralelismo perfeito, onde aplicação que pode ser dividida em conjunto de processos que requerem pouca ou nenhuma comunicação. Por exemplo, simulações de Monte Carlo.

- Paralelismo de dados, onde a mesma operação é aplicada em muitos elementos de dados simultaneamente. Por exemplo, múltiplos processos podem procurar em diferente partes do banco de dados para resolver uma busca específica.

- Paralelismo funcional, também chamado de paralelismo de controle. Múltiplas operações são resolvidas simultaneamente, sendo necessário comunicação de dados entre as operações. Por exemplo, numa usina hidrelétrica um processo pode simular o sistema de refrigeração, outro o gerador, etc.

A razão entre a quantidade de computação realizada por uma aplicação e a quantidade de comunicação é chamada de *granularidade*. A maneira como se refere a granularidade de uma tarefa é se ela possui um grão pequeno, médio ou grande. Em tarefas de grão pequeno, as operações são rápidas e fazem muita comunicação.

Já em tarefas de grão grande, pode-se executar uma aplicação toda antes de realizar comunicação ou sincronização. Por isto, normalmente quanto maior o grão de uma aplicação menor será o custo de sincronização e maior será o ganho na execução paralela (*speed-up*).

4.2 Algoritmos Estocásticos

4.2.1 Método da Próxima Reação

O método da próxima reação é uma otimização realizada no algoritmo de Gillespie proposta em (GIBSON; BRUCK, 2000). Neste método é montado um grafo de dependência das reações, que é usado para responder a pergunta: Se a reação r reagir qual serão as reações que terão suas densidades de probabilidade alteradas ?

A idéia central é somente atualizar as densidades de probabilidade (μ) quando necessário. Basicamente, são armazenadas as μ_i nas folhas da árvore completa e colocado em cada nó a soma dos seus filhos, da direita e da esquerda. Então, o nó raíz terá o valor das soma de todos os (μ_i). Quando é preciso atualizar, somente são atualizadas as μ_i que mudaram e seus ancestrais.

O algoritmo do método da próxima reação responde a seguinte questão: Para cada reação r, quando será a próxima vez que r ocorrerá ? Para implementar o método da próxima reação é utilizada uma estrutura de dados chamada de fila de prioridades para armazenar os μ de cada reação. Nesta estrutura de dados recupera-se a reação de menor μ com complexidade de tempo $O(1)$. A inserção e removeção de um elemento possuem complexidade de tempo $O(\log n)$.

Os algoritmos para gerar trajetórias estocásticas de um sistema de reações bioquímicas de Gillespie possuem complexidade $O(rE)$, onde r é o número de reações e E o número de eventos da trajetória. O método da próxima reação otimiza a geração das trajetórias para uma complexidade $O(r + E \log r)$ para reações fracamente acopladas como as encontradas na regulação gênica.

4.2.2 Método τ-Leap

O método τ-leap foi proposto em (GILLESPIE, 2001). Este método possui ganhos significativos no custo computacional com uma perda aceitável na precisão. Na versão original do algoritmo de Gillespie, a equação mestre é resolvida de maneira exata para produzir o comportamento temporal preciso dos sistemas pela geração do tempo exato de reagir cada reação. Entretanto, muitas vezes não é necessário obter tantos detalhes da simulação. Ao invés de achar qual reação acontece daqui a um certo passo de tempo, pode-se querer saber quantas vezes cada reação acontece em determinado intervalo de tempo. Se o intervalo de tempo é suficientemente grande para ocorrer em várias reações dentro dele pode-se esperar um ganho substancial no custo computacional.

O ganho na velocidade da computação é pago com a perda de precisão. Para manter a precisão do método entre um certo limite é necessário escolher um intervalo apropriado, τ, o qual pode ser suficientemente pequeno para que a mudança na função de densidade de probabilidade seja aceitável. Gillespie também apresentou diversos mecanismos de controle para escolher τ, incluindo o método explícito, método do ponto-do-meio-estimado e o método k-alpha. Infelizmente, o método τ-leap original sofre de com o problema de instabilidade. O erro da solução é sensível demais ao passo de tempo do algoritmo e instável sobre diferentes mecanismos de controle explícito.

A pouco tempo atrás em (GILLESPIE; PETZOLD, 2003) foi proposto uma melhoria no método τ-leap. Esta melhoria supre o problema de instabilidade numérica com sucesso, aperfeiçoando o procedimento de determinar o tamanho máximo do intervalo (*leap*) para um grau de precisão especificado. Entretanto, este novo método ainda não foi testado para sistemas de reações complexos. E segundo os próprios autores, o método ainda está na sua infância e não pode ser considerado como uma ferramenta robusta que automaticamente e confiavelmente trata qualquer simulação.

4.2.3 Stochsim

O algoritmo Stochsim foi proposto em (MORTON-FIRTH; BRAY, 1998) e trata os componentes biológicos como objetos individuais iterativos baseado na distribuição de probabilidade derivada de dados experimentais. Neste esquema, a cada rodada de computação, um par de moléculas é avaliada para a reação potencial. Pelo fato de as interações entre moléculas serem tratadas probabilisticamente, o Stochsim é capaz de reproduzir o comportamento estocástico dos fenômenos de sistemas biológicos (NOVèRE; SHIMIZU, 2001; MENG; SOMANI; DHAR, 2004).

Ao contrário do passo do tempo variável do algoritmo de Gillespie, o algoritmo do Stochsim usa um passo do tempo fixo e pode ser otimizado para a precisão desejada. Entretanto, a conveniência desta medida traz consigo um custo computacional a mais quando usado um passo de tempo vazio, por exemplo quando nenhum evento ocorre no período de um passo de tempo.

Uma limitação do algoritmo de Gillespie é a inviabilidade computacional do uso de moléculas multi-estados. Por exemplo, uma proteína com dez sítios de ligação terá um total de 2^{10} estados (ligado/desligado) e requererá a mesma quantidade de canais de reações para ser simulada. O Stochsim pode ser modificado para superar este problema associando estados para as moléculas sem aumentar o custo computacional significativamente.

4.2.4 Meta-Algoritmo

O Meta-Algoritmo foi proposto em (TAKAHASHI et al., 2004), ele provê um algoritmo modular com um escalonador de eventos discretos que pode incorporar qualquer tipo de algoritmo de simulação dirigido pelo tempo. Diferentes algoritmos possuem diferentes forças, e encaixam-se melhor em diferente escalas de espaço e tempo. No desenvolvimento de modelos que percorem múltiplas escalas pode-se escolher entre duas maneiras de tratar o problema: juntar a força dos algoritmos e produzir um algoritmo unificado de utilidade geral ou implementar os algoritmos como módulos mascarados por uma interface comum de um *framework* genérico de

avanço no tempo e comunicação entre os módulos. O Meta-Algoritmo implementa a segunda solução.

A implementação do Meta-Algoritmo é modular e orientada a objetos, baseada num escalonador de eventos discretos e na interpolação polinomial de Hermite. Ela pode lidar com diferentes algoritmos e diferentes escalas de tempo. Para demonstrar esta capacidade foi implementado o módulo estocástico do método da Próxima Reação e outro módulo para calcular equações diferenciais. O que seria semelhante ao método híbrido, Foi obtido um grande ganho de performance sem perda de precisão significativa.

4.3 Simuladores

Hoje em dia existem diversas ferramentas capazes de simular redes metabólicas de maneira estocástica. Na tabela 4.1 são comparadas algumas das principais ferramentas que possuem características comuns à arquitetura de simulação proposta neste trabalho, chamada de Metabolic Simulator na tabela. As ferramentas citadas na tabela estão em constante evolução. Nos próximos parágrafos será feita uma breve descrição do estado atual de cada uma delas.

A ferrameta E-CELL, proposta em (TOMITA et al., 1999), tem sido aplicada na tentativa de construir um modelo cinético para toda a célula. Na sua primeira versão, foram coletadas da literatura as taxas descrevendo as reações metabólicas envolvendo o produto de 127 genes do *Mycoplasma genitalium*, a célula com o menor genoma conhecido. Atualmente está sendo estudado como modelar no E-CELL o crescimento celular, replicação de DNA, segregação de cromossomo e divisão celular. A ferramenta ainda possui uma interface gráfica para modelar e acompanhar a simulação e a sua última versão é a 3.1.102.

O STOCKS, apresentado em (KIERZEK, 2002), é um software para simulação estocástica de processos bioquímicos que implementa um algoritmo derivado do algoritmo de Gillespie. Ele possui características dedicadas ao estudo dos processos celulares, como a possibilidade de estudar o processo num intervalo de várias

gerações da célula com a aplicação de um modelo de divisão de célula simples. O software está na versão 1.02.

O simulador Dynetica (YOU; HOONLOR; YIN, 2003) possui uma interface amigável para construir, visualizar e analisar a cinética dos modelos dos sistemas biológicos. Ele ainda possui facilidades para construir modelos onde muitas reações são expressão e interação entre produtos gênicos. O Dynetica pode realizar simulações tanto do modelo estocástico (Gillespie) quanto do determinístico (Runge-Kutta). O simulador está na versão 1.1.

O Cellware (DHAR et al., 2004b), é um ambiente multi-algoritmo para modelar e simular eventos estocásticos e determinísticos na célula. Atualmente implementa os algoritmos estocásticos de Gillespie e Gibson, e os seguintes algoritmos determinísticos Euler, Runge-Kutta e Dormand Prince. Ele ainda provê o método híbrido que é menos preciso que o estocástico mas necessita de menos poder computacional. Está na versão 2.0 que é capaz de utilizar *Grid* para computar as simulações. E ainda possui interfaces gráficas para modelar e controlar o simulador.

O Gepasi (`www.gepasi.org`) é um pacote de software para modelar sistemas bioquímicos, sua versão atual é a 3.30. Ele simula a cinética dos sistemas de reações bioquímicas de maneira determinística e provê várias ferramentas para ajustar o modelo aos dados experimentais, otimizar qualquer função do modelo, realizar análise do controle metabólico e análise da estabilidade linear. O Gepasi simplifica a tarefa de montar o modelo transformando as reações químicas em matrizes de equações diferenciais automaticamente. Ele ainda possui algoritmos numéricos sofisticados para garantir a precisão dos resultados. Outra facilidade do Gepasi é possuir uma interface gráfica para controlar a simulação e visualizar os resultados.

O Simulac (`http://gobi.lbl.gov/~aparkin/`), disponível na sua versão beta, é um simulador de propósito geral para simulação estocástica da genética de procariotos simples e sistemas bioquímicos. Ele foi desenvolvido para ser um dos núcleos do Biospice, que será descrito na próxima seção.

4.3.1 *Frameworks* de simulação para Sistemas Biológicos

O Biospice é uma colaboração entre estudiosos de várias áreas do conhecimento para desenvolver um conjunto de ferramentas e um *framework* de código aberto para modelar, representar e simular os processos celulares. Uma das áreas de pesquisas que envolvem o Biospice é a pesquisa de núcleos de simulação que representem a dinâmica da célula, ou seja, algoritmos de baixo custo computacional e precisos para simulação de redes metabólicas. O acesso a documentação e ferramentas do Biospice pode ser feito através do endereço eletrônico `www.biospice.org`.

O Systems Biology Workbench (SBW) tenta resolver um problema que muitos estudiosos de biologia computacional vêem enfrentando. Os modelos e resultados de simulações não podem ser comparados, compartilhados ou reutilizados diretamente, pois, as ferramentas desenvolvidas por diferentes pesquisadores não são compatíveis entre si. Infelizmente, nenhuma das ferramentas atuais responde todas as questões que a comunidade precisa. Um dos principais causadores deste problema é a impossibilidade de trocar modelos entre as ferramentas por falta de um formato comum. Para resolver este problema foi criada a Systems Biology Markup Language (SBML).

A parte de desenvolvimento do SBW é a elaboração de um *framework* que permita que diferentes ferrametas interajam entre si. Ele deve suportar ferramentas escritas em diferentes linguagens de programação e ser multi-plataforma. Desta forma, os desenvolvedores poderão se concentrar nas áreas que eles tem maior conhecimento e não precisarão reimplementar funcionalidades disponíveis em outras ferramentas. O *framework* e seus módulos podem ser encontrados em `www.sbw-sbml.org`.

TABELA 4.1 – Comparação entre algumas ferramentas para simulação de redes metabólicas

Nome	**Algoritmos de Simulação**	**Arquitetura de Software**	**Linguagem de Modelagem**	**GUI**	**Licença**	**Referência**
Metabolic Simulator	Métodos de Gillespie (Primeira Reação e Direto), Método Híbrido e ODE	*Stand-Alone*, *Grid* e *cluster*	XML próprio	Não	GPL	Este trabalho
Cellware	Métodos de Gillespie, Gibson, Stochsim, TauLeap e ODE	*Stand-Alone*, *Grid* e *cluster*	XML SBML	SIM	*Freeware*	(DHAR et al., 2004b), (DHAR et al., 2004a)
ECell	Métodos de Gillespie, Gibson, e ODE	*Stand-Alone*, *Grid* e *cluster*	XML SBML	SIM	GPL	(TOMITA et al., 1999)
STOCKS	Métodos de Gillespie	*Stand-Alone* e *cluster*	Própria	Não	GPL	(KIERZEK, 2002)
Dynetica	Métodos de Gillespie e ODE	*Stand-Alone*	Própria	Sim	Freeware	(YOU; HOONLOR; YIN, 2003)
Gepasi	ODE	*Stand-Alone*	XML SBML	Sim	Freeware	`www.gepasi.org`
Simulac	Métodos de Gillespie	*Stand-Alone*	Própria	Não	Código aberto	`http://gobi.lbl.gov/~aparkin/`

Capítulo 5

Simulador

O simulador proposto utiliza o algoritmo de Gillespie no modelo estocástico e no modelo determinístico o método de Runge-Kutta (PRESS et al., 2002) para evoluir um sistema de reações químicas acopladas. No modelo estocástico são simuladas trajetórias e calculada a média das trajetórias. A simulação de uma única trajetória usa pouco poder computacional, mas é necessário calcular muitas trajetórias. Então, a estratégia para estes métodos é distribuir a simulação das trajetórias em diversas máquinas. As seções 5.5.1 e 5.5.2 descrevem como o simulador realiza a distribuição das trajetórias.

Os métodos estocásticos implementados foram: método da Primeira Reação e método Direto. A cada execução do simulador podemos simular uma ou mais trajetórias. Quando simulamos mais de uma trajetória o resultado da simulação será a média das trajetórias simuladas usando números randômicos diferentes. As trajetórias são independentes, ou seja, podemos simulá-las separadamente.

A independência da simulação das trajetórias facilita a tarefa de paralelização e distribuição das simulações. Porém, devemos escolher sementes diferentes para os geradores de números randômicos. Elas não devem se repetir e nem serem muito próximas para evitar a geração de trajetórias iguais comprometendo o resultado das simulações.

No modelo determinístico a resolução do sistema de reações químicas é feita diretamente, resolvendo numericamente as equações diferenciais ordinárias que re-

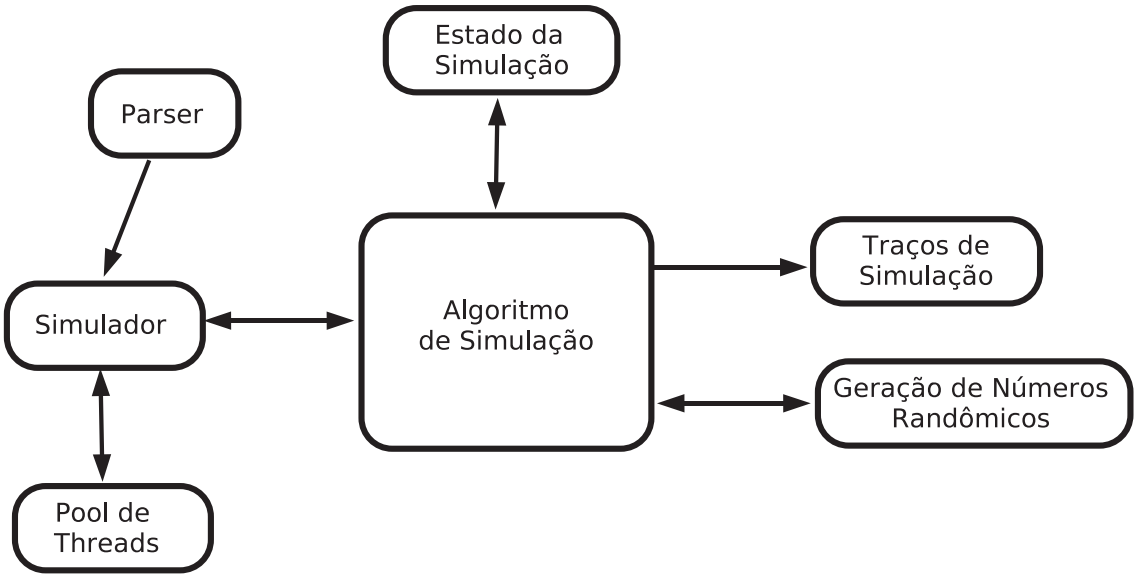


FIGURA 5.1 – Representação dos módulos de simulação e da interação entre eles.

presentam o sistema de reações químicas.

Na seção 5.1 são discutidos os componentes de software desenvolvidos para a arquitetura de software do simulador. A seção 5.2 mostra o formato do arquivo de configuração do simulador. As seções 5.3 e 5.4 apresentam as soluções para configurar as equações diferenciais automaticamente e as taxas variáveis das reações, em tempo de execução. A seção 5.5 descreve os ambientes nos quais a arquitetura de simulação foi projetada para executar, ambiente multi-processado (5.5.1) e ambiente de *grid* (5.5.2).

5.1 Arquitetura

A arquitetura do simulador visa a separação das principais tarefas do simulador: leitura e interpretação do arquivo de entrada de dados (*Parser*), computação do método de simulação (algoritmo de simulação), geração dos traços de simulação, gerenciamento de *threads*, geração de números randômicos e manutenção do estado da simulação. Estas tarefas são realizadas por módulos, na figura 5.1 estão representados os módulos e a interação entre eles.

O estado da simulação em determinado momento é representado por instâncias

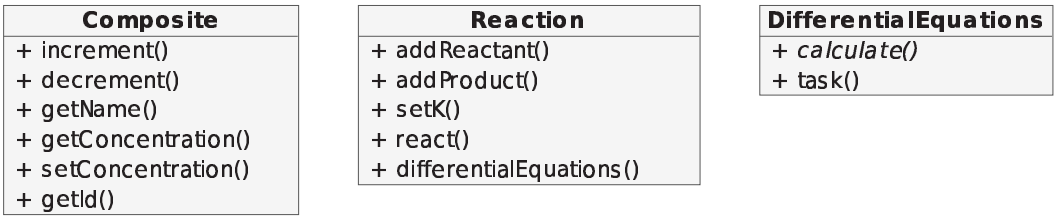


FIGURA 5.2 – Diagrama de classes da interface de manipulação dos objetos que guardam o estado da simulação.

da classe *Composite*, onde cada objeto *Composite* modela um composto bioquímico. Os atributos de um composto são: nome, identificador numérico único e concentração. As instâncias da classe *Reaction* alteram o estado da simulação, ou seja, modificam as concentrações dos compostos envolvidos na reação quando o método *react()* da mesma é chamado. Outra maneira de alterar o estado da simulação é através do método contínuo, o qual usa a representação das reações via equações diferenciais ordinárias. Um objeto do tipo *DifferentialEquations*, que será discutido com mais detalhes na seção 5.3, modifica o estado da simulação quando calcula as concentrações dos compostos através do método *calculate()*. As interfaces das classes *Composite*, *Reaction* e *DifferentialEquations* podem ser vistas no diagrama de classes da figura 5.2.

A classe *AbstractParser* disponibiliza a interface para a leitura e interpretação do arquivo de entrada de dados, como mostra o diagrama de classes da figura 5.3. O método *parse()* é chamado para ler e interpretar o arquivo de entrada e deve ser especializado pelas classes que estendem a classe *AbstractParser*. Os métodos *getComposites()* e *getReactions()* retornam os objetos que representam o estado inicial da simulação e as reações que afetam o mesmo, respectivamente. A classe *XmlParser* lê arquivos no formato XML (*Extensible Markup Language*) (HOLZNER, 2001) e usa XPath para localizar os parâmetros da simulação, sendo que o formato XML foi definido na seção 5.2. O simulador usa, por padrão, arquivo de configuração no formato XML, mas uma outra opção seria usar a classe *SimpleFormatParser*.

A geração dos traços de simulação foi implementada através do *design pattern*

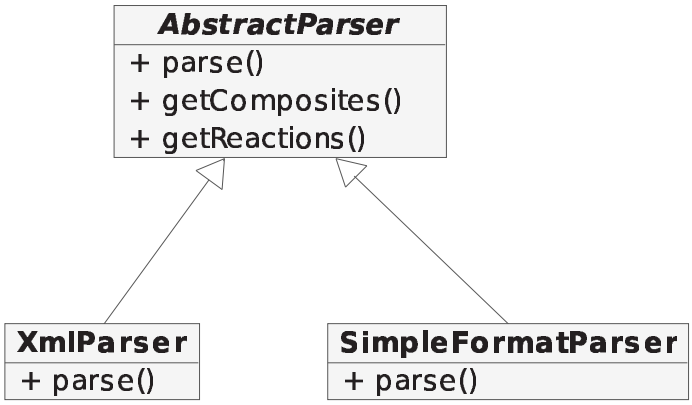


FIGURA 5.3 – Diagrama de classes da interface de leitura e interpretação do arquivo de entrada de dados.

Observer (GAMMA et al., 1995), onde a classe abstrata *TraceGeneratorObserver* provê a interface para implementação dos vários tipos de "observadores". Os objetos instânciados pelas classes *MeanTrace* e *SimpleTrace* estendem a *TraceGenerator-Observer* e por isso são observadores, como é mostrado no diagrama de classes da figura 5.4.

A classe *MeanTrace* precisa discretizar as concentrações dos compostos durante a simulação para computar a média delas ao final das simulações. A figura 5.5 mostra como é feito para determinar o valor das concentrações dos compostos em um determinado instante de tempo. A estratégia é sempre consolidar somente os estados anteriores ao evento, pois o tempo da simulação sempre aumenta, não podendo acontecer eventos com tempo inferiores aos que já aconteceram. Desta forma, se um evento acontece no tempo t os valores das concentrações dos passos anteriores a t não mudam mais e portanto são os valores atuais.

No exemplo da figura 5.5 apenas a reação que transforma o composto A em B acontece durante o tempo observado. No tempo $0,22$ reage pela primeira vez a reação e converte um A em um B, mas só é consolidado o estado dos passos $0,1$ e $0,2$. O resto dos passos não são consolidados e apenas guarda-se a informação sobre a reação que aconteceu neste tempo. Pois, pode acontecer mais reações no passo

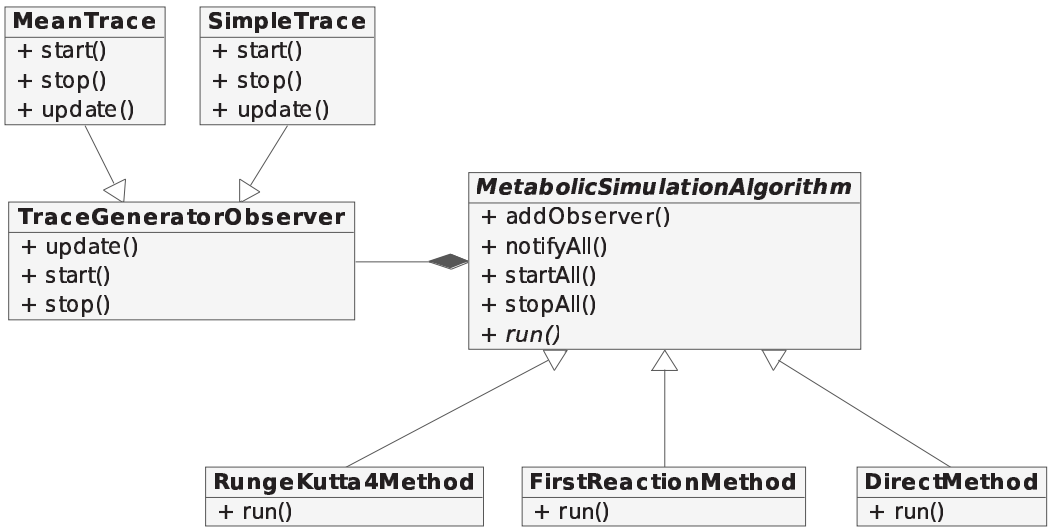


FIGURA 5.4 – Diagrama de classes da interface para geração de traços da simulação, *design pattern Observer*, e da interface para implementação de métodos de simulação.

entre $0,2$ e $0,3$. Quando a reação reage novamente no tempo $0,53$ consolida-se o estado dos passos $0,3$, $0,4$ e $0,5$. E finalmente quando no tempo $0,65$ acontece a reação novamente, consolida-se o estado do passo $0,6$.

Os diferentes métodos de simulação são implementados especializando a classe *MetabolicSimulationAlgorithm*. No diagrama de classes da figura 5.4 são mostrados alguns dos métodos implementados pelo simulador. A classe *MetabolicSimulationAlgorithm* ainda mantém a lista de observadores e disponibiliza as funções de comunicação com os mesmos. Os métodos *notifyAll()*, *startAll()* e *stopAll()* são usados para: avisar a ocorrência de uma mudança de estado a todos observadores, inicializar e finalizar todos os observadores.

O simulador é capaz de explorar o paralelismo intra-nó via *threads*, no processamento de tarefas. Para gerenciar o uso das *threads* é utilizado um repositório de *threads*. Uma das principais vantagens desta técnica é a diluição do custo de criar e destruir *threads* Mais precisamente,

$$c_{threads} = \frac{(n_{threads} \times (c_{criar} + c_{destruir}))}{n_{task}} + c_{gerenciameto} \times n_{threads} \qquad (5.1)$$

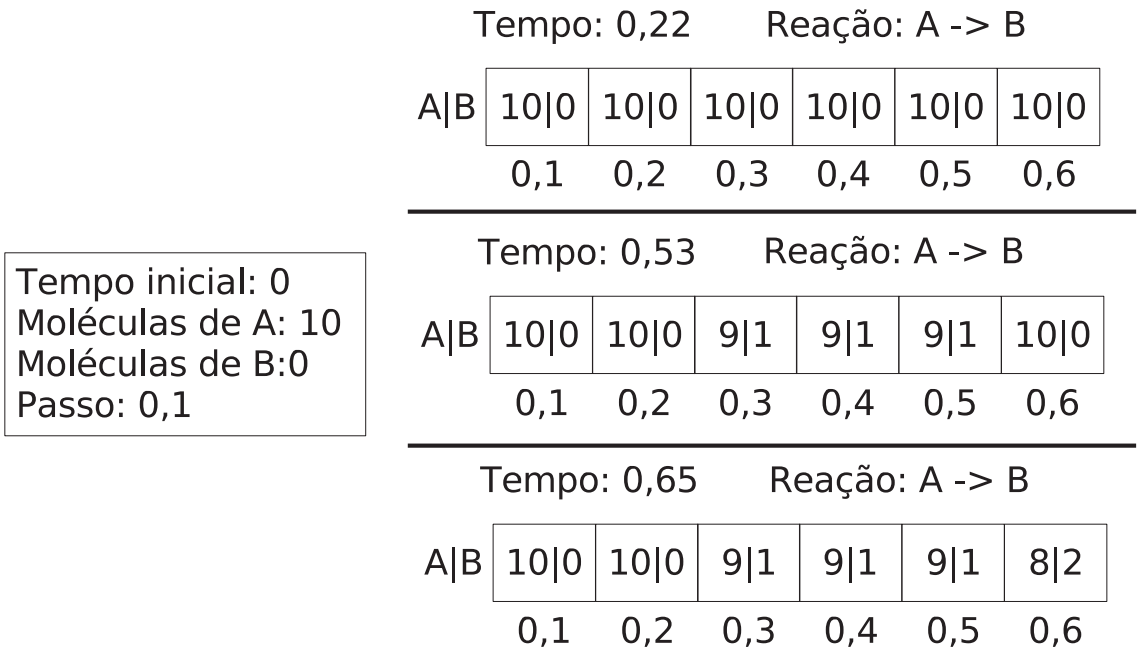


FIGURA 5.5 – Atualização do estado dos "observadores" quando um evento acontece.

onde temos que: $c_{threads}$, é o custo total envolvido no uso de *threads* para computar as tarefas; $n_{threads}$, é o número de *threads* no repositório; c_{criar}, é o custo de criação de uma *thread*; $c_{destruir}$, é o custo de destruição de uma *thread*; n_{task}, é o número de tarefas que foram processadas pelas *threads* e $c_{gerenciameto}$, é o custo de gerenciamento das *threads*, por exemplo achar uma *thread* livre para executar a tarefa, esperar por uma *thread* livre e sincronização entre as *threads*.

A classe *ThreadsPool* implementa o repositório e a sua interface pode ser vista no diagrama de classes da figura 5.6. Para a *ThreadsPool*, uma *thread* é qualquer classe que especialize a classe abstrata *ThreadInterface*, onde o único método abstrato é o *task()*. A *DifferentialEquations* é um tipo de *thread* e a sua tarefa é calcular as equações diferenciais ordinárias. Uma das classes que a utiliza é a *Runge-Kutta4Method*, a qual resolve um sistema de equações diferenciais ordinárias usando o método de Runge Kutta de quarta ordem (PRESS et al., 2002). As outras duas classes que estendem *ThreadInterface* são *MultiThreadDirectMethod* e *MultiThread-FirstReactionMethod*. As duas executam uma trajetória do algoritmo estocástico de Gillespie na tarefa, vide seção 3.2, a primeira com o Método Direto implementado

pela classe *DirectMethod* e a segunda com o Método da Primeira Reação implementado pela classe *FirstReactionMethod*.

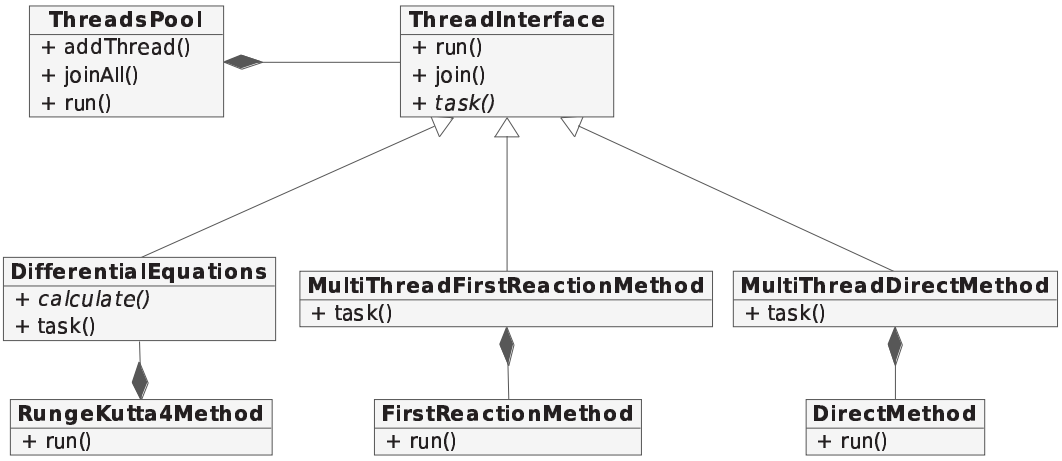


FIGURA 5.6 – Diagrama de classes: *Thread Pool*

5.2 Arquivo de Configuração

A passagem de parâmetros para o simulador é feita através de um arquivo texto no formato XML. A *Document Type Definition* (DTD) (HOLZNER, 2001) que especifica o formato do arquivo de configuração do simulador é mostrado no anexo XML DTD (A.1). Analisando a DTD nota-se que existem três partes essenciais de configuração. A parte de configuração dos parâmetros relativos ao método de simulação (*tag Parameters*), outra de declaração dos compostos bioquímicos (*tag Composites*) e a parte da especificação das reações que operam sobre os compostos (*tag Reactions*).

A primeira *tag* do arquivo é a *Simulation*, ela precisa conter como filhos as seguintes *tags* : *Parameters*, *Composites* e *Reactions*. A *tag Parameters* tem como filhos *Global* e *Methods*, onde o primeiro contém os parâmetros comuns a todos os métodos de simulação e o segundo possui os parâmetros específicos de cada método de simulação. Nos parâmetros globais pode-se opcionalmente selecionar quais compostos terão seus traços salvos ao final da simulação, pela *tag TracedComposites*.

Os compostos bioquímicos são declarados na *tag Composites* que deve conter um ou mais filhos do tipo *composite*. Cada *composite* tem como atributo um identificador único, um nome e a concentração inicial. Nas outras partes do XML sempre que for necessário referenciar um composto deve-se usar o atributo nome com o qual ele foi declarado.

As reações bioquímicas são listadas na *tag Reactions*, onde cada reação é representada pela *tag Reaction*. Cada *Reaction* possui os atributos k (taxa da reação) e *catalytic*. Uma reação ainda contém os reagentes e produtos, representados pelas *tags Reactants* e *Products* respectivamente. Os reagentes e produtos são descritos nas *tags reactant* e *product* respectivamente e possuem os seguintes atributos: *composite*, nome do composto; *stoichiometric*, coeficientes estequiométricos.

O anexo A.2 mostra uma configuração no formato XML. O modelo descrito por este exemplo contém quatro compostos *a*, *b*, *c* e *d*. O simulação evolui segundo três reações: uma molécula de *a* reage com uma de *c* na taxa 0.5 para formar uma molécula de *b*; uma molécula de *a* reage com uma de *b* na taxa 0.2 para formar uma molécula de *c*; uma molécula de *c* reage na taxa 0.6 para se transformar uma molécula de *d*. Durante a simulação serão salvos os traços dos compostos *a*, *b* e *c*. Este exemplo está configurado para ser computado em qualquer um dos seguintes métodos: *firstReaction*; *directMethod*; *rungeKutta4*; *hybridRk4*; *multiThreadHybridRk4*.

5.3 Equações Diferenciais Ordinárias

Para realizar simulações com o modelo determinístico é necessário transformar as reações bioquímicas em suas equações diferenciais ordinárias correspondentes. O aplicativo *reactions2diffeqs* automatiza este processo. Ele obtém as informações necessárias através do arquivo de configuração do simulador no formato XML, descrito na seção 5.2.

O programa segue o algoritmo de conversão das reações em equações diferen-

ciais descrito na seção 3.1. A equação

$$\frac{d[c_0]}{dt} = k \times ce_1 \times [c_1]^{ce_1} \times ce_2 \times [c_2]^{ce_2} \times ... \times ce_n \times [c_n]^{ce_n} \tag{5.2}$$

mostra a forma genérica das equações diferenciais produzidas pelo algoritmo. A implementação da resolução desta fórmula é codificada no arquivo *DifferentialEquations.cpp*, apresentado na seção 5.1. Ela possui duas partes distintas: uma de dados e outra com a implementação do algoritmo de conversão.

A parte de dados é composta por duas matrizes *pair* e *k*. A matriz *k* armazena o valor da taxa e quantos pares ela possui na matriz *pair* de cada equação. Cada par da matriz *pair* contém o coeficiente estequiométrico e o identificador do composto dele. O anexo A.3 é um exemplo gerado com o aplicativo *reactions2diffeqs*.

A implementação do algoritmo de conversão para este formato de dados é sempre igual e pode ser vista na função *calculate* no anexo A.3, onde, para cada composto para o qual irá ser calculada a equação diferencial, percorre-se a linha da matriz *k* correspondente multiplicando a taxa pelos pares apontados na matriz *pair*, como mostra a equação (5.2).

5.4 Reações com Taxa Variável

Modelos biológicos complexos normalmente envolvem reações com taxa variável. Para suportar este tipo de reação o simulador disponibiliza uma maneira de carregar a função do cálculo da taxa da reação em tempo de execução. As reações com taxa variável devem fornecer os atributos *k_library_name* e *k_function_name*, o nome da biblioteca dinâmica e o nome da função respectivamente, ao invés do atributo *k* na sua *tag Reaction* do arquivo de configuração.

A biblioteca dinâmica que implementa a computação da taxa variável deve conter uma função *init* que será chamada no momento em que o simulador criar o objeto que representa a reação com a correspondente taxa variável. No anexo A.4 é mostrado o código de uma biblioteca dinâmica para o cálculo da taxa variável de

uma reação.

5.5 Ambiente de Execução do Simulador

O simulador foi desenvolvido em C++ (SCHILDT, 2002) e suas dependências extras são as bibliotecas LibXML2 (VEILLARD; BRACK; BUCHCIK, 2004) para a leitura do arquivo XML e POSIX *thread* (BUTENHOF, 1997) para o repositório de *threads*. Para execução das simulações ainda é necessário ter o ambiente de desenvolvimento, *GNU Compiler Collection* (GCC, `http://gcc.gnu.org`), instalado em pelo menos uma máquina para a compilação das equações diferenciais ordinárias do modelo.

Existem duas situações onde a configuração do ambiente de simulação varia, quando a simulação será somente estocástica ou quando será determinística ou híbrida. O ambiente de simulação para simulações estocásticas é mais simples de configurar, somente sendo necessário o arquivo de configuração XML descrevendo o modelo e o executável do simulador.

O processo de configuração do ambiente para a computação da simulação híbrida ou determinística de um determinado modelo é feito em três etapas. Primeiramente, é necessário gerar as equações diferenciais ordinárias do modelo usando o arquivo XML de configuração, descrito na seção 5.2, e a ferramenta *reactions2diffeqs*, disponibilizada junto com o simulador, para obter o arquivo *DifferentialEquations.cpp*. Depois, compila-se o arquivo *DifferentialEquations.cpp* e obtém-se a biblioteca compartilhada com as equações diferenciais. Por último, configura-se o sistema para achar a biblioteca compartilhada com as equações diferenciais do modelo.

A necessidade de ter várias bibliotecas compartilhadas, uma para cada modelo, complica o processo de configuração do ambiente para realização das simulações. Mas por outro lado como as equações diferenciais ordinárias já estão compiladas não é preciso passá-las como parâmetros, melhorando o desempenho do simulador. O arquivo de configuração fica menor e não é necessário alocar memória dinamicamente para as equações diferenciais. A influência destes dois fatores é proporcional ao

número de reações e compostos, pois o número de equações diferenciais é equivalente ao número de compostos e o número de termos de cada equação diferencial depende de quão acopladas as reações são. Quanto mais compostos, maior teria que ser o arquivo de configuração e quanto mais acopladas as reações, mais memória dinâmica teria que ser alocada.

Para otimizar o processo de compilação o arquivo de equações diferenciais é compilado como biblioteca compartilhada, sendo apenas necessário compilá-lo e não todo o simulador para cada modelo. Depois de gerado e compilado o *DifferentialEquations.cpp* somente é preciso fazer sua geração e compilação novamente quando for feita alguma alteração na declaração das reações no arquivo XML de configuração.

5.5.1 *Threads*

O algoritmo de Gillespie é um algoritmo de Monte Carlo, e uma das características dos algoritmos de Monte Carlo é a fácil paralelização (NEWMAN; BARKEMA, 1999). O simulador usa o repositório de *threads*, descrito na seção 5.1, para computar concorrentemente mais de uma trajetória de simulação. Para computar uma trajetória é necessário configurar a semente do gerador de números aleatórios usado na escolha da reação que será executada e no cálculo do momento em que a reação irá acontecer. Quando usa-se *threads* para computar N trajetórias, são necessárias N sementes. Neste caso as sementes são tiradas de uma distribuição uniforme precisando apenas passar como parâmetro para o simulador apenas uma semente ao invés de N.

Os métodos que exploram o paralelismo intra-nó são *MultiThreadDirectMethod* e *MultiThreadFirstReactionMethod*, eles usam o repositório de *threads* para executar paralelamente a simulação das trajetórias. Um dos parâmetros destes métodos é o número de *threads*, ele deve ser menor ou igual ao número de trajetórias que se deseja simular. Na simulação utilizando estes métodos as trajetórias que serão simuladas são divididas entre as *threads*. Quando todas as *threads* terminarem de computar as

suas trajetórias é feita a média dos valores de todas as trajetórias simuladas e salvo o resultado no arquivo de traços de simulação.

5.5.2 *Grid*

Como já foi dito na seção 5.5.1, o algoritmo de Gillespie é um algoritmo de Monte Carlo. Por isto, se classifica na categoria de paralelismo perfeito (seção 4.1) sendo uma ótima aplicação para *cluster* e *grid*. As execuções de uma determinada simulação com sementes aleatórias diferentes podem ser realizadas de forma assíncrona, usando *threads* e *grid*. Os resultados de N simulações podem ser consolidados e considerados como se fossem o resultado de uma simulação que executou $N \times T$ trajetórias, eliminando a necessidade de guardar os N resultados. Para consolidar os resultados a arquitetura de simulação disponibiliza o aplicativo *merger*, o qual tem como parâmetros o nome do arquivo de saída e os vários arquivos de traço de simulação que serão consolidados, sendo calculada a média das concentrações a cada instante de tempo.

O resultado das simulações pode ser acompanhado consolidando periodicamente os resultados que foram gerados com os resultados obtidos nos períodos anteriores, fazendo a média do resultado da consolidação do período P com o resultado consolidado do período $P - 1$. E através dessa consolidação dos resultados parciais é possível utilizar algum critério para decidir se será necessário computar mais trajetórias ou não.

A figura 5.7 representa o fluxo de execução da simulação usando o simulador numa arquitetura de *grid*. O processo de simular uma rede metabólica segue os seguintes passos:

- Geração de lotes de simulações
- Computação dos lotes de simulações no *Grid*
- Consolidação dos resultados do repositório

O programa que gera lotes de simulação é encarregado de gerar as tarefas que

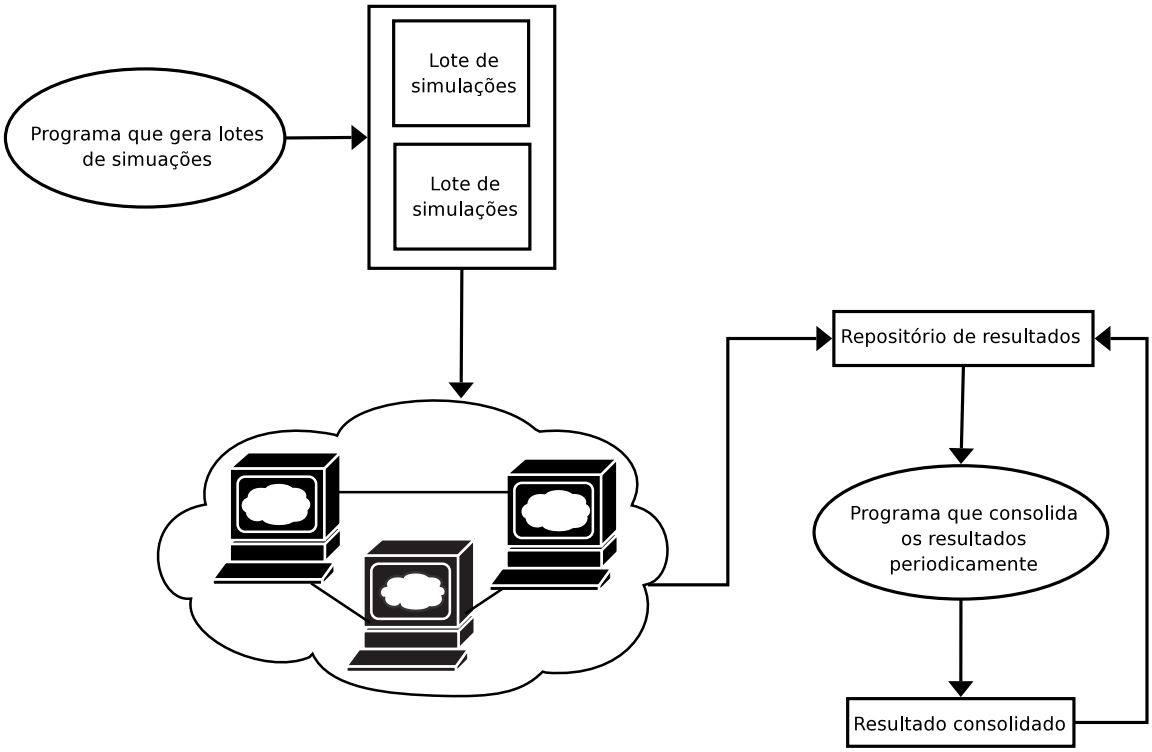


FIGURA 5.7 – Diagrama do fluxo de execução da simulação em um ambiente de *grid*

irão executar no *grid*. Para isto, ele usa um gerador de números aleatórios para obter as sementes aleatórias das tarefa de forma que elas não se repitam. Outra função do programa que gera lotes de simulação é verificar periodicamente se o resultado já estabilizou ou se já foi executado o número máximo de simulações configurado. Se alguma das condições foi atingida ele para de gerar lotes de simulações, caso contrário ele gera novos lotes de simulação.

O programa que consolida os resultados também roda periodicamente. A cada período, ele consolida os resultados disponíveis com os anteriores, gerando um arquivo de traço de simulação com o resultado parcial. O resultado parcial é usado pelo programa que gera lotes de simulação para decidir se deve ou não gerar mais lotes. Ou ainda para acompanhar o andamento da simulação através de gráficos feitos a partir do resultado parcial, por exemplo. Este modelo de execução não é afetado quando um computador do *grid* é desligado durante a computação de uma simulação ou por algum outro motivo qualquer a computação de uma trajetória for

interrompida antes de seu término. Quando isto ocorre, se necessário, o programa que gera lotes adiciona mais um lote de simulação para substituir o que foi perdido e o resultado da simulação não fica comprometido.

Hoje em dia existem várias ferramentas para facilitar o uso de *grid*, cada uma delas com suas vantagens e desvantagens. Para viabilizar a utilização da arquitetura de simulação descrita nos parágrafos anteriores é necessário que a ferramenta de *grid* forneça pelo menos as seguintes características: capacidade de distribuir tarefas e seus respectivos arquivos de configuração pelo *grid* e um repositório onde os arquivos de traços das simulação gerados pelas tarefas fiquem acessíveis para o programa que consolida os resultados. Estes requisitos são mínimos e praticamente todas as ferramentas de *grid* os implementam.

Para realização dos experimentos deste trabalho foram criados vários *scripts* feitos em Python e Bash. Os quais aproveitam do ambiente de *grid* utilizado, que contém o serviço OpenSSH (YLONEN, 1995) em todas as máquinas. As transmissões dos dados de entrada e de saída dos experimentos são realizadas utilizando o serviço OpenSSH. Outra facilidade que os *scrpits* proporcionam é o balanceamento de carga entre as máquinas baseado na velocidade dos processadores. Por exemplo, uma máquina que possui a velocidade v vai computar n_{traj} trajetórias.

$$n_{traj} = \frac{t_{total}}{\sum v_i} \times v, \tag{5.3}$$

onde t_{total} é o número total de trajetórias que será computado, e $\sum v_i$ é a soma da velocidade de todas as máquinas.

Capítulo 6

Resultados

Este capítulo pode ser dividido em duas partes. Na primeira parte, apresenta-se a validação dos resultados obtidos com o simulador deste trabalho com dados encontrados na literatura, seção 6.1. Na segunda parte, são realizados diversos experimentos para avaliar o desempenho do simulador, seção 6.2.

6.1 Validação

6.1.1 Reações Bioquímicas

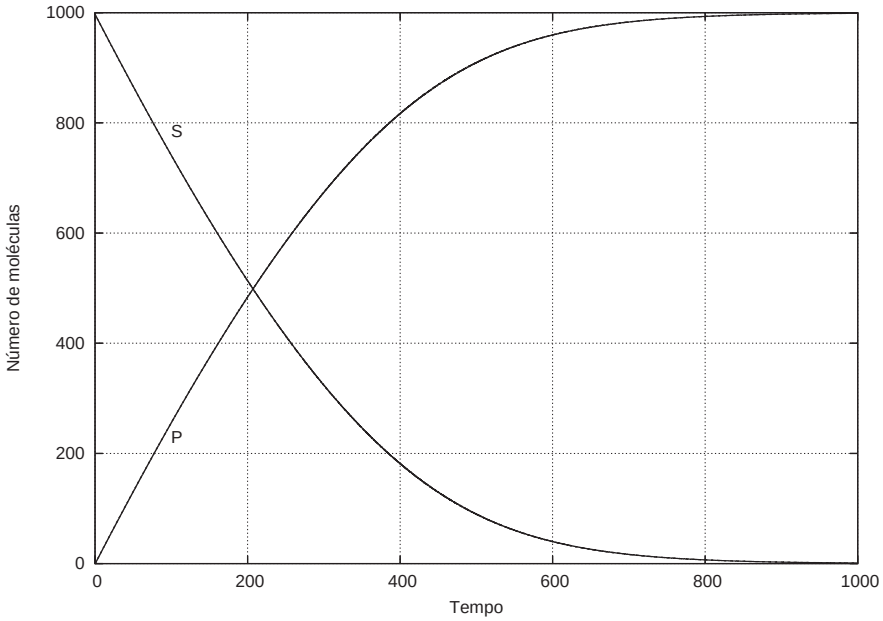
Nesta seção, serão comparados os resultados dos modelo determinístico e estocástico para os modelos de reações bioquímicas apresentados na seção 2.3: reação enzimática básica, cinética do substrato suicida e fenômeno cooperativo. Nos gráficos são apresentados os valores do modelo determinístico, calculados usando as equações diferenciais iguais às da literatura, e da média de 10.000 trajetórias do modelo estocástico, simuladas com o método direto de Gillespie.

Nos gráficos da figura 6.1 compara-se os resultados dos modelos determinístico e estocástico utilizando o modelo da reação enzimática básica, apresentado na seção 2.3.1. O estado inicial do sistema possui 1.000 moléculas de substrato, 5 moléculas de enzima e nenhuma de proteína. A tabela 6.1 contém as taxas utilizadas nas reações. No gráfico 6.1(a), mostra-se o número de moléculas de proteínas e de substrato em

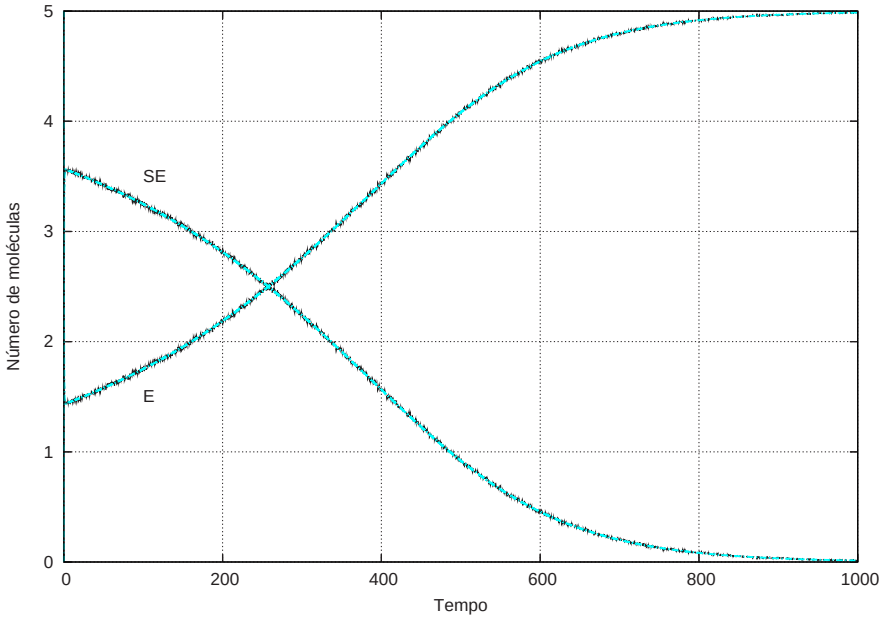
função do tempo. E no gráfico 6.1(b), mostra-se o número de moléculas de enzima e do complexo substrato-enzima em função do tempo.

TABELA 6.1 – Parâmetros Cinéticos do Modelo: Reação Enzimática Básica

Parâmetro	**Valor**
k_1	$0,0025$
k_{-1}	$0,25$
k_2	$0,75$



(a) Substrato e Proteína



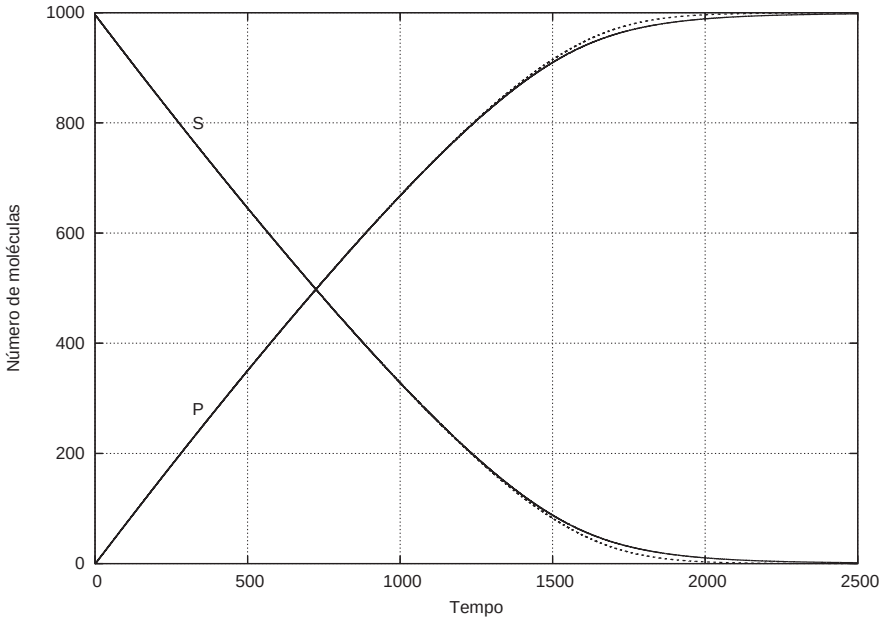
(b) Enzima e complexo Substrato-Enzima

FIGURA 6.1 – Comparação da variação do número de moléculas durante o tempo utilizando o modelo da reação enzimática básica. Na figura (a) as linhas pontilhadas são calculadas com o modelo determinístico e as linhas sólidas com o modelo de Gillespie. Na figura (b) as linhas mais claras são calculadas com o modelo determinístico e a linhas sólidas pretas com o modelo de Gillespie.

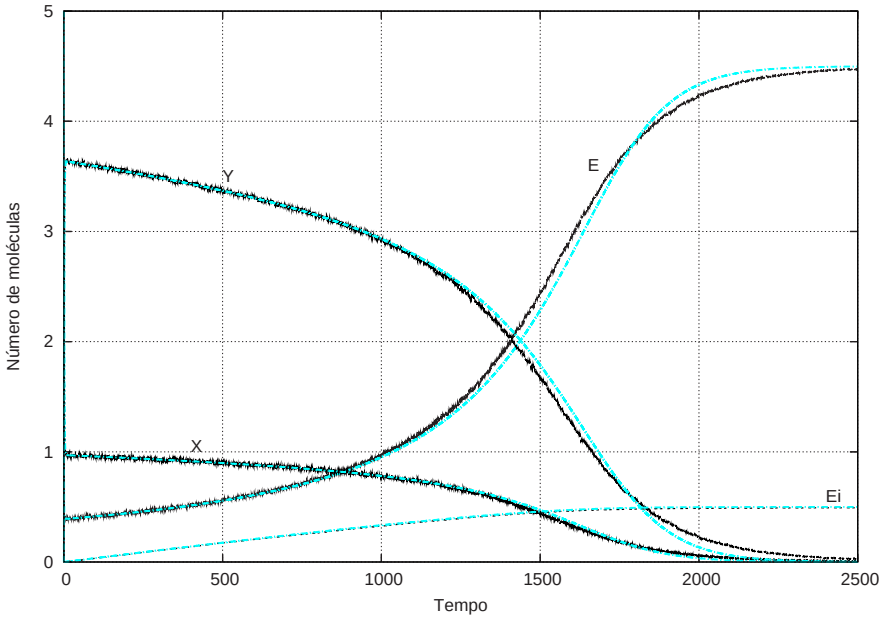
Nos gráficos da figura 6.2 comparam-se os resultados dos modelos determinístico e estocástico utilizando o modelo da cinética do substrato suicida, apresentado na seção 2.3.2. O estado inicial do sistema possui 1.000 moléculas de substrato, 5 moléculas de enzima e nenhuma de proteína, X, Y e enzima inativa. A tabela 6.2 contém as taxas utilizadas nas reações. No gráfico 6.2(a), mostra-se o número de moléculas de proteínas e de substrato em função do tempo. E no gráfico 6.2(b), mostra-se o número de moléculas de enzima, X, Y e enzima inativa em função do tempo.

TABELA 6.2 – Parâmetros Cinéticos do Modelo: Cinética do Substrato Suicida

Parâmetro	**Valor**
k_1	$0,0025$
k_{-1}	$0,25$
k_2	$0,75$
k_3	$0,2$
k_4	$0,0001$



(a) Substrato e Proteína



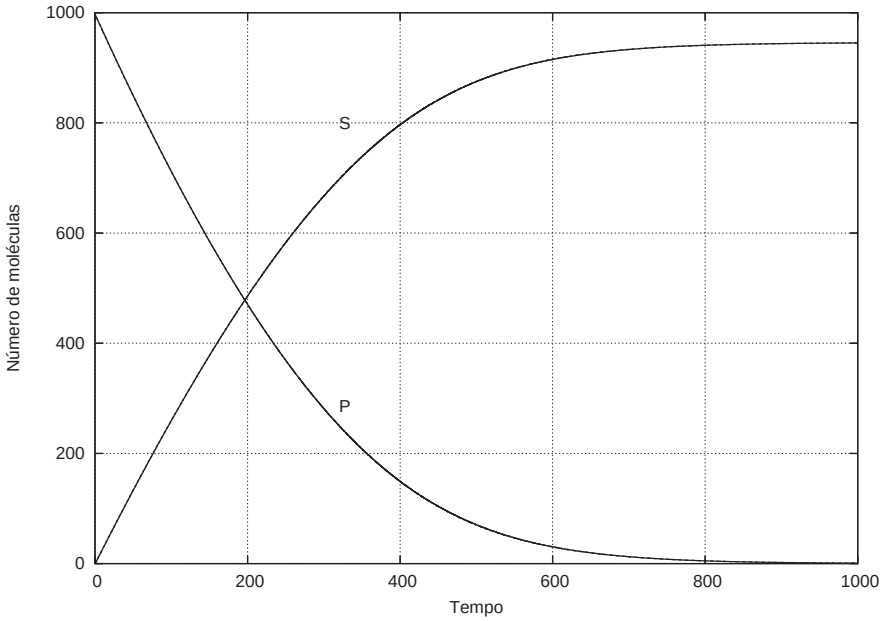
(b) Enzima, X, Y e Enzima inativa

FIGURA 6.2 – Comparação da variação do número de moléculas durante o tempo utilizando o modelo cinética do substrato suicida. Na figura (a) as linhas pontilhadas são calculadas com o modelo determinístico e as linhas sólidas com o modelo de Gillespie. Na figura (b) as linhas mais claras são calculadas com o modelo determinístico e a linhas sólidas pretas com o modelo de Gillespie.

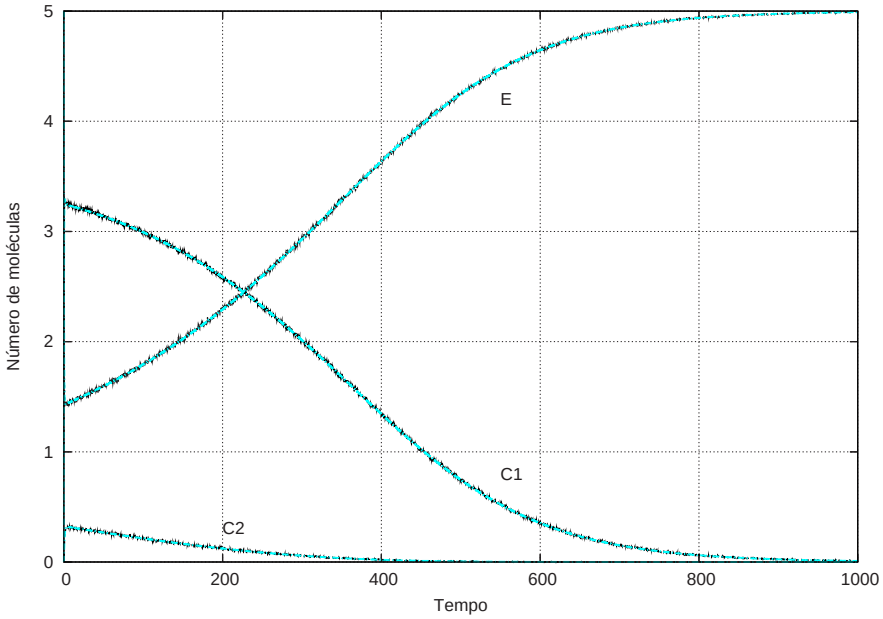
Nos gráficos da figura 6.3 compara-se os resultados dos modelos determinístico e estocástico utilizando o modelo do fenômeno cooperativo, apresentado na seção 2.3.3. O estado inicial do sistema possui 1.000 moléculas de substrato, 5 moléculas de enzima e nenhuma de proteína, $C1$ e $C2$. A tabela 6.3 contém as taxas utilizadas nas reações. No gráfico 6.3(a), mostra-se o número de moléculas de proteínas e de substrato em função do tempo. E no gráfico 6.3(b), mostra-se o número de moléculas de enzima, $C1$ e $C2$.

TABELA 6.3 – Parâmetros Cinéticos do Modelo: Fenômeno Cooperativo

Parâmetro	**Valor**
k_1	$0,0025$
k_{-1}	$0,25$
k_2	$0,75$
k_3	$0,00013$
k_{-3}	$0,35$
k_4	$0,95$



(a) Substrato e Proteína



(b) Enzima, C1 e C2

FIGURA 6.3 – Comparação da variação do número de moléculas durante o tempo utilizando o modelo fenômeno cooperativo. Na figura (a) as linhas pontilhadas são calculadas com o modelo determinístico e as linhas sólidas com o modelo de Gillespie. Na figura (b) as linhas mais claras são calculadas com o modelo determinístico e a linhas sólidas pretas com o modelo de Gillespie.

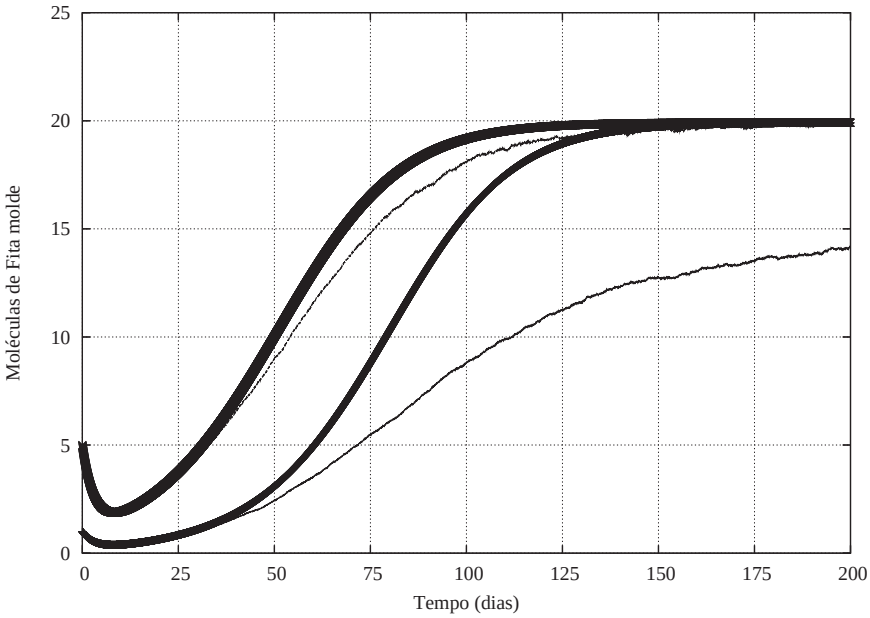
Nos gráficos apresentados acima, observa-se que a média do modelo estocástico oscila sobre o valor do modelo determinístico. Quanto menor for o número de trajetórias simuladas para calcular a média do modelo estocástico, maior será a amplitude da oscilação da média estocástica. No modelo determinístico, é pressuposto a existência de um grande número de moléculas de cada composto, o que não ocorre no caso das enzimas. Por isto, observam-se algumas diferenças entre o resultado dos dois modelos de simulação.

6.1.2 Método de Gillespie

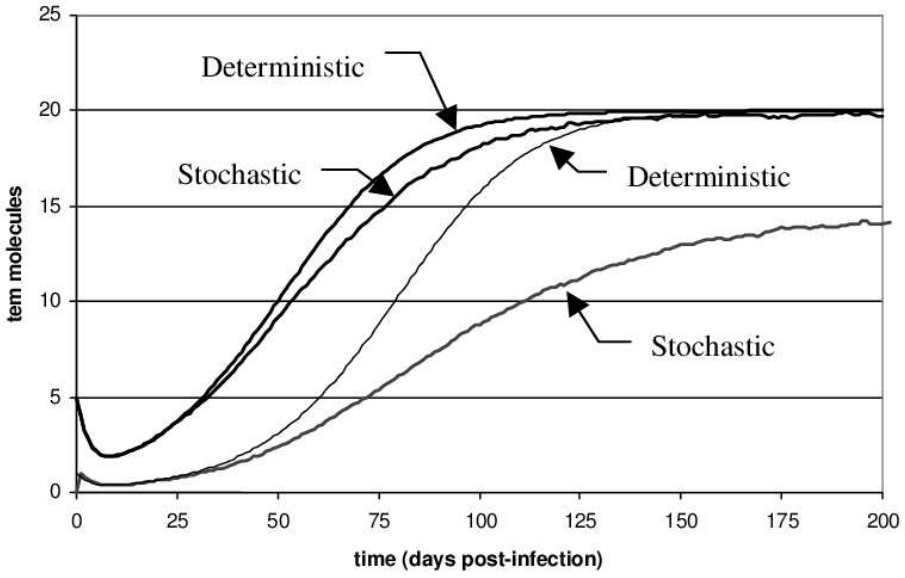
Os resultados da simulação determinística podem ser considerados como a média do comportamento da cinética viral (SRIVASTAVA; YOU; YIN, 2002). Então, o resultado determinístico pode ser comparado com a média de múltiplas trajetórias simulação estocástica. Na figura 6.4 a média de 3000 trajetórias estocásticas foi comparada com a simulação determinística, para o número de moléculas de fita molde. Foram simulados níveis de infecções baixo e alto, número inicial de fitas molde 1 e 5 respectivamente.

A cinética das simulações determinísticas e estocástica difere significativamente para um nível de infecção baixo. O modelo determinístico converge para o estado estacionário de 20 moléculas de fita molde, enquanto o modelo estocástico atinge um platô de 15 moléculas de fita molde. Já quando o nível de infecção é alto, o resultado do modelo estocástico fica semelhante ao do determinístico.

Na figura 6.4 pode-se observar que os gráficos, (a) Metabolic Simulator e (b) (SRIVASTAVA; YOU; YIN, 2002), são praticamente idênticos. Através da comparação do resultado destas simulações averiguou-se a corretude dos métodos de Gillespie do simulador deste trabalho.



(a) Metabolic Simulator



(b) (SRIVASTAVA; YOU; YIN, 2002)

FIGURA 6.4 – Comparação da cinética das moléculas de fita molde nos modelos determinístico e estocástico. No gráfico (a), as linhas claras são as médias estocásticas e as escuras o cálculo determinístico. Em ambos os gráficos, são simulados um baixo nível de infecção (1 fita molde inicial) e um alto nível de infecção (5 fitas molde iniciais). Os resultados estocásticos são a média de 3.000 trajetórias.

6.1.3 Método Híbrido

O modelo da cinética intracelular viral (seção 2.4) possui duas características interessantes (HASELTINE; RAWLINGS, 2002). A primeira, é as flutuações dos três componentes do modelo possuirem ordem de magnitude diferentes. Para a mesma escala de tempo, as proteínas estruturais flutuam de centenas para milhares de moléculas, enquanto as fitas molde e os genomas flutuam entre dezenas de moléculas. Além disso, a solução do modelo exibe uma distribuição bimodal. Mais precisamente, a célula pode exibir uma infecção "típica" na qual todos os compostos são produzidos, ou um infecção "abortada" na qual todos os compostos são eliminados da célula.

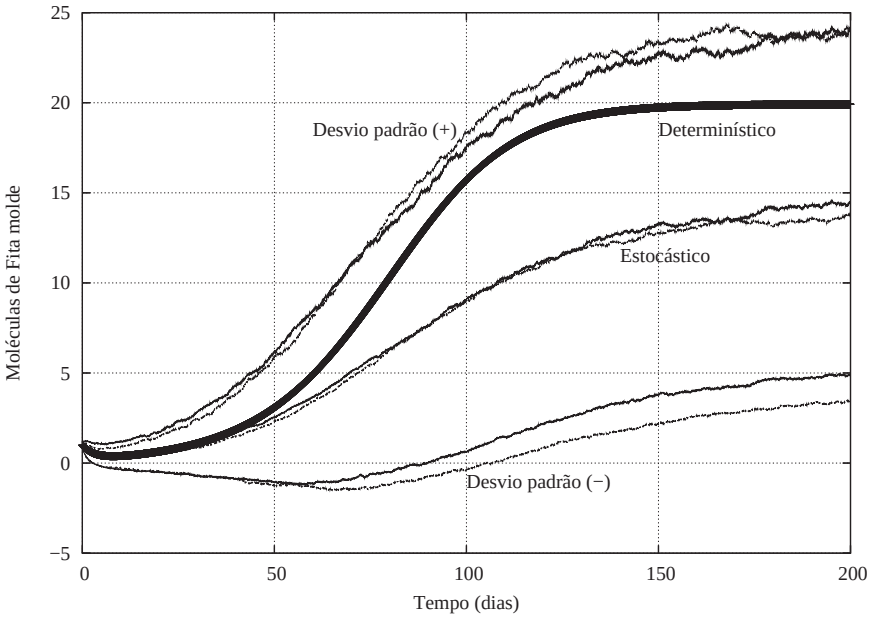
Quando o número de moléculas de fita molde e proteínas estruturais é maior que 0 e 100, respectivamente, as reações (2.27) e (2.28) ocorrem muito mais vezes do que em qualquer outras das reações restantes. Então quando o número de moléculas de fita molde é maior que 0 e proteínas estruturais é maior que 100, é feita a seguinte partição no sistema: (2.23), (2.24), (2.25) e (2.26) compõem o subconjunto de reações lentas; e (2.27) e (2.28) compõem o subconjunto de reações rápidas.

Durante a simulação do método híbrido, os resultados do modelo determinístico para o subconjunto de reações rápidas são arredondados, já que no modelo estocástico o número de moléculas é sempre um inteiro. Este arredondamento introduz um erro imperceptível no sistema (HASELTINE; RAWLINGS, 2002).

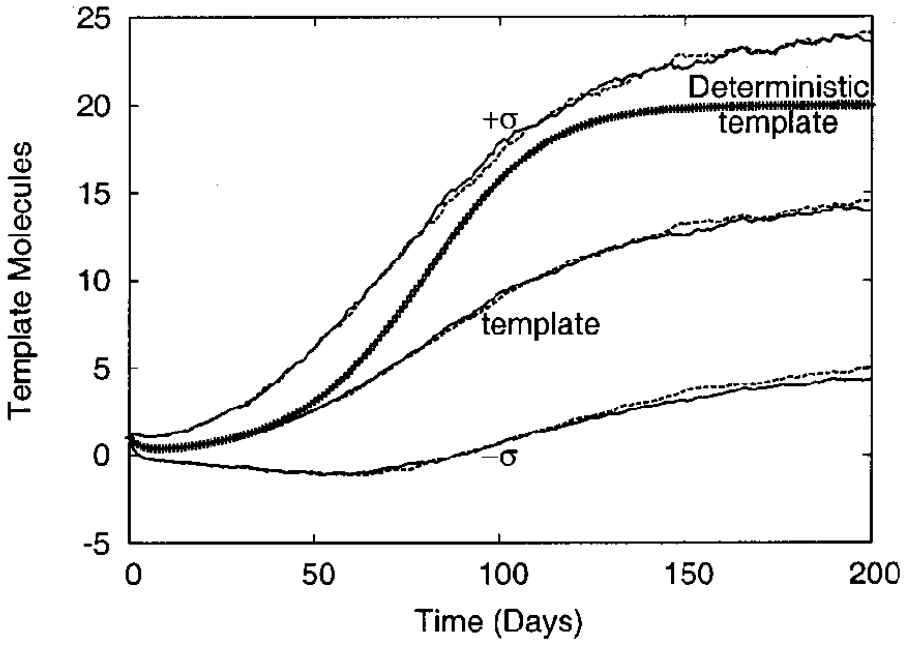
As figuras 6.5, 6.6 e 6.7 comparam a evolução da média e desvio padrão do número de moléculas de fita molde, genoma e proteínas estruturais, respectivamente. As médias e desvios padrões foram calculados a partir de 1000 trajetórias de simulação, tanto no método estocástico exato (Métodos de Gillespie) como no método determinístico-estocástico aproximado. Pode-se observar que o método Híbrido consegue reconstruir as curvas dos métodos de Gillespie com precisão.

Quando se compara os resultados dos gráficos gerados com o simulador deste trabalho e os gráficos apresentados em (HASELTINE; RAWLINGS, 2002), percebe-se que eles diferem. E os resultados do Metabolic Simulador não são tão acurados

quanto os apresentados em (HASELTINE; RAWLINGS, 2002). Os dois principais motivos para esta discordância é o uso de integradores de equações diferenciais diferentes, o Metabolic Simulator usa o método Runge-Kutta de quarta ordem com passo fixo e em (HASELTINE; RAWLINGS, 2002) usa-se o método de Euler-Maruyama. E segundo, a política de escolha do passo fixo que será utilizado pelo método de resolução das equações diferenciais conforme o tamanho do período que será calculado. Pois, os períodos podem variar muito de tamanho, dependendo da freqüência com que as reações são realizadas, e se o passo for maior que o período que será calculado obtém-se resultados incorretos.

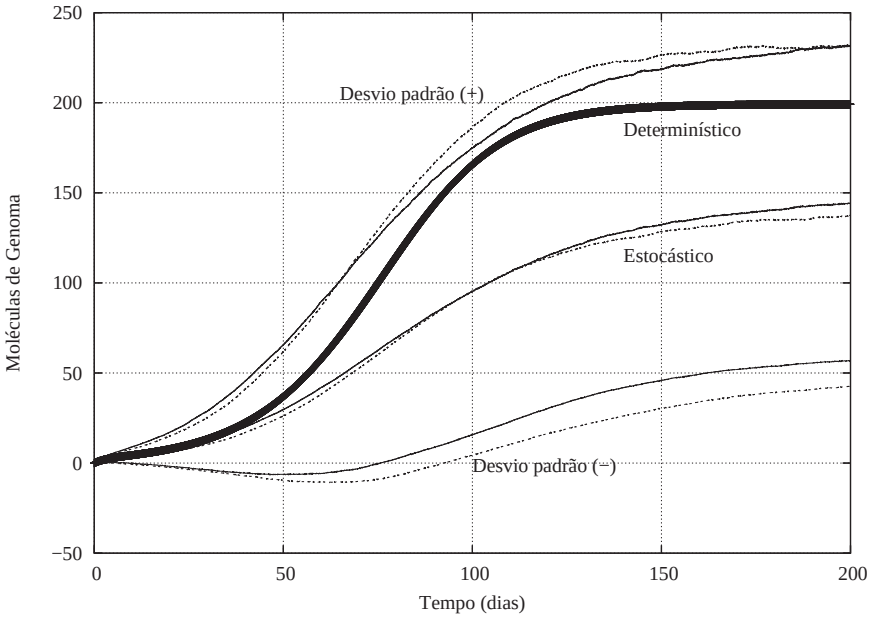


(a) Metabolic Simulator

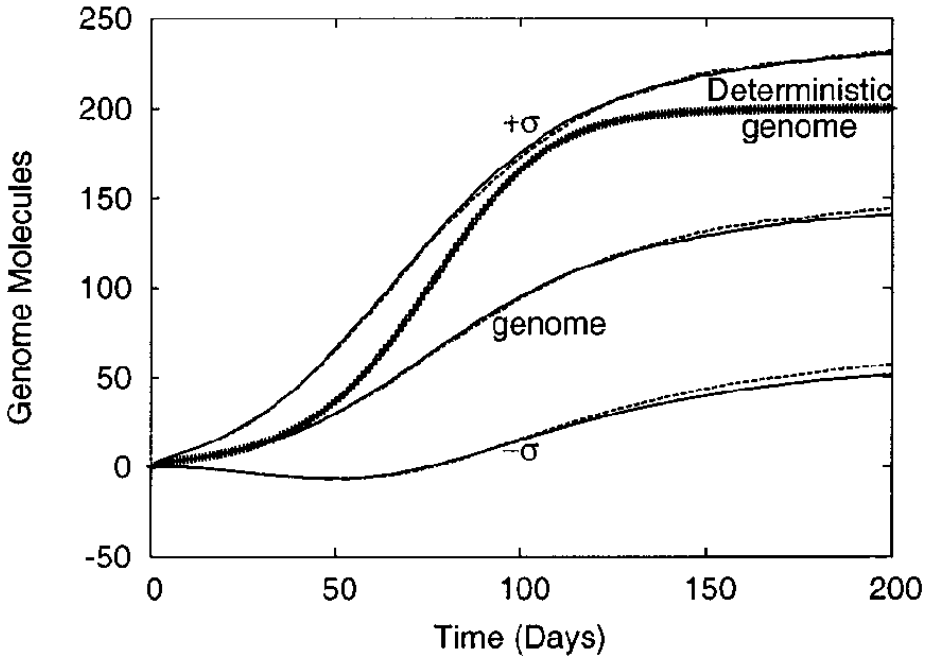


(b) (HASELTINE; RAWLINGS, 2002)

FIGURA 6.5 – Comparação entre a média e desvio padrão do número de moléculas fitas molde, para os modelos exato estocástico (linhas sólidas), determinístico-estocástico aproximado (linhas pontilhadas) e determinístico (pontos).

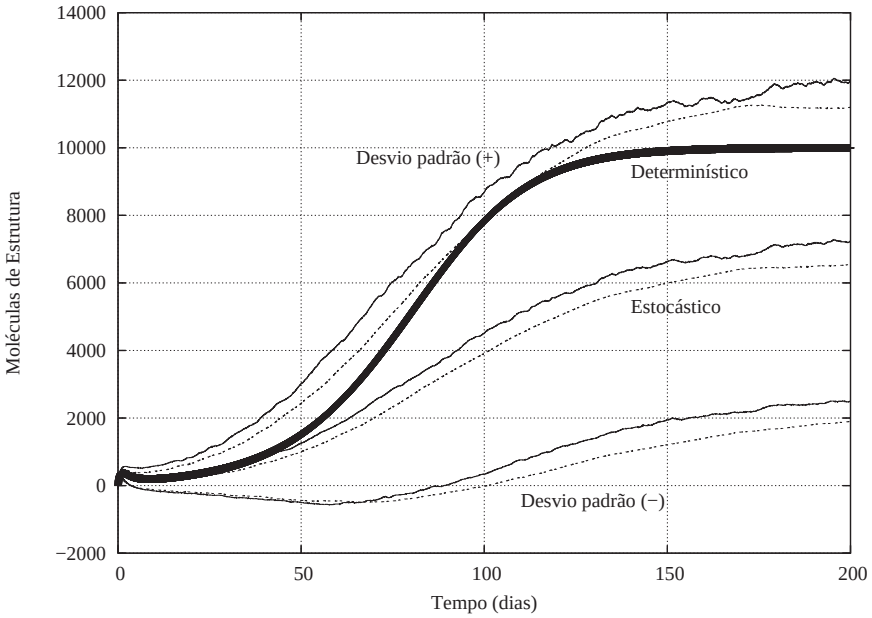


(a) Metabolic Simulator

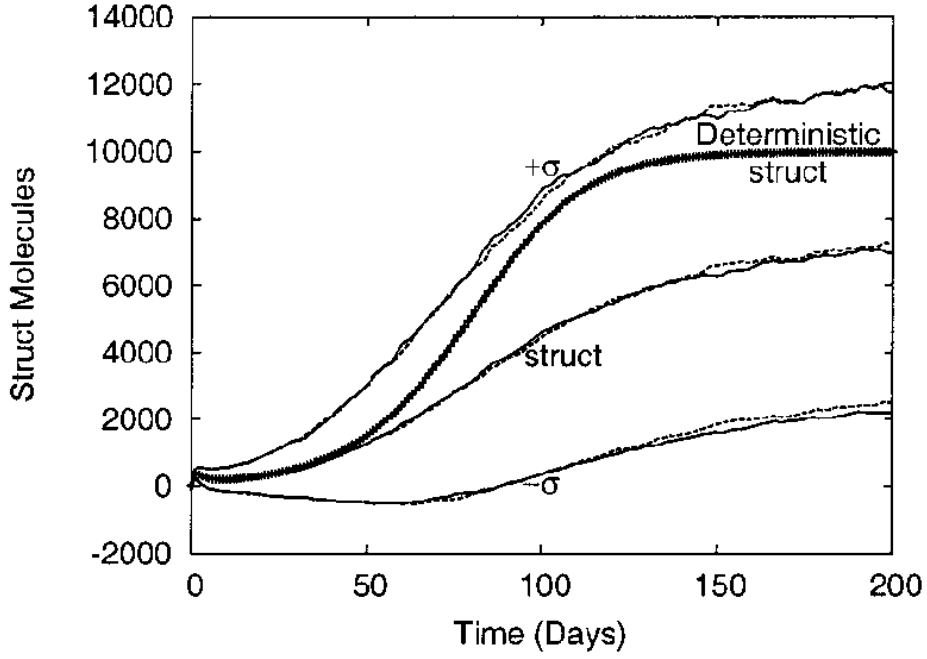


(b) (HASELTINE; RAWLINGS, 2002)

FIGURA 6.6 – Comparação entre a média e desvio padrão do número de moléculas genoma, para os modelos exato estocástico (linhas sólidas), determinístico-estocástico aproximado (linhas pontilhadas) e determinístico (pontos).



(a) Metabolic Simulator



(b) (HASELTINE; RAWLINGS, 2002)

FIGURA 6.7 – Comparação entre a média e desvio padrão do número de moléculas estrutura, para os modelos exato estocástico (linhas sólidas), determinístico-estocástico aproximado (linhas pontilhadas) e determinístico (pontos).

6.2 Avaliação de Desempenho

6.2.1 Paralelismo intra-nó

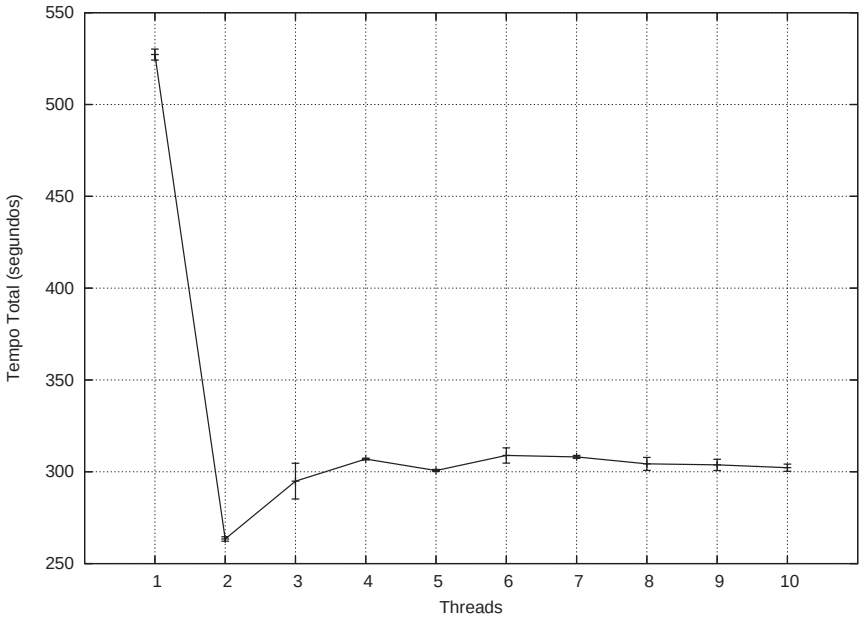
O primeiro teste de desempenho realizado buscou descobrir qual o número de *threads* calculadas paralelamente resulta no melhor custo-benefício para as máquinas bi-processadas com *hyperthreading* ativado. A máquina alvo possui dois processadores Xeon 2, 4 Ghz com *hyperthreading* ((MARR et al., 2002)) habilitado em ambos processadores, e usa o sistema operacional GNU Linux com *kernel* 2.6.9 e glibc 2.3.4.

As trajetórias computadas concorrentemente usam mais de uma *thread* no simulador, escolhe-se quantas *threads* irão ser utilizadas e o número total de trajetórias é divido igualmente entre as *threads*. Se o número total de trajetórias não tem uma divisão exata pelo número de *threads*, o resto da divisão também é distribuído entre as *threads*, fazendo com que elas calculem o número mais próximo possível de trajetórias. Por exemplo, para calcular 6 trajetórias com 3 *threads* cada uma processa 2 trajetórias, já com 4 *threads* 2 *threads* processam 2 trajetórias e as outras 2 processam 1 trajetória.

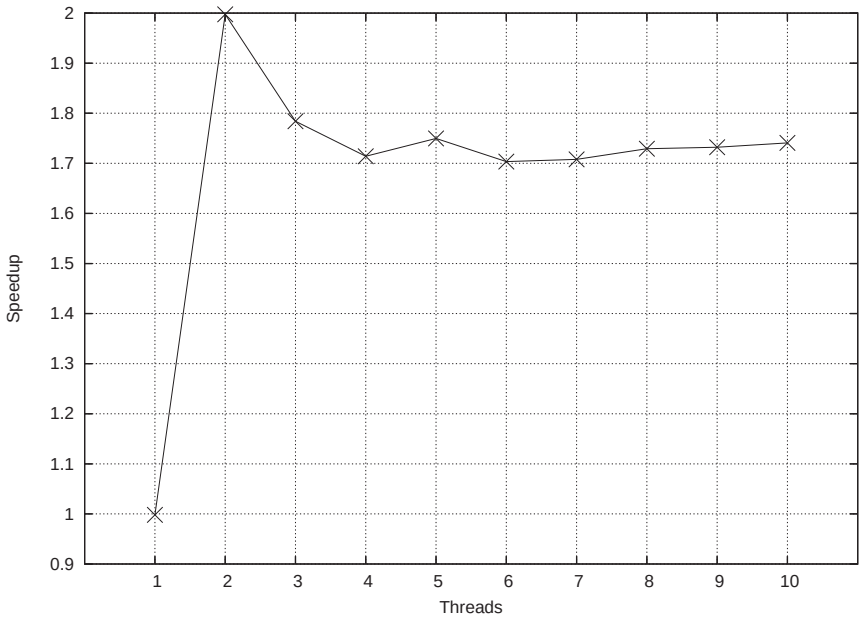
No gráfico da figura 6.8(a) é mostrado o resultado da variação do número de *threads* em função do tempo levado para calcular 50 trajetórias de simulação do modelo da cinética intracelular viral com multiplicidade de infecção 5 (número de fitas molde inicial igual a 5). Através do gráfico pode-se observar que o melhor desempenho é obtido utilizando duas *threads*. As máquinas com *hyperthreading* possuem para cada processador físico mais um processador lógico. Então, como as máquinas são bi-processadas para o sistema operacional existem 4 processadores (fluxo de execução) disponíveis. O processamento paralelo intra-nó, com *threads*, do simulador só realiza sincronização quando todas as *threads* terminam de computar, sendo possível para o simulador se beneficiar do paralelismo eficientemente.

O gráfico da 6.8(b) quantifica quanto foi o ganho com o aumento do número de *threads* em relação a uma *thread*, medida chamada de *speedup*. O maior *speedup*, aproximadamente 2, ocorre com a utilização de duas *threads*. Este é um dado interessante pois era esperado um ganho no *speedup* de até $2,6$. Dado que para as

aplicações domésticas e de servidores corporativos o próprio fabricante (Intel, `www.intel.com`) anuncia que o ganho com a utilização de *hyperthreading* pode chegar a 30%, ou seja, um *speedup* de $0,3$ por processador virtual. Mas a partir de mais de um fluxo de execução do simulador por processador obtém-se perda no *speedup*. Esta perda indica que aplicações de uso intensivo de processamento e que usam os mesmos recursos do processador, por exemplo operações em inteiros e pontos flutuantes usam recursos diferentes do processador, não conseguem tirar proveito da tecnologia de *hyperthreading*.



(a)



(b)

FIGURA 6.8 – Avaliação do tempo total para simular 50 trajetórias variando o número de threads em uma máquina com dois processadores e *hyperthreading* ativado nos dois, *speedup* máximo de aproximadamente 2 com duas *threads*.

6.2.2 Paralelismo entre-nós

O ambiente no qual as simulações foram computadas pode ser visto na figura 6.9. Ele é composto por dois *clusters*, um *cluster* de cinco máquinas com dois processadores Xeon 2, 4 Ghz com *hyperthreading* ((MARR et al., 2002)) habilitado em todos processadores. Elas usam o sistema operacional GNU Linux com *kernel* 2.6.9, glibc 2.3.4 e possuem servidor SSH. As máquinas estão conectadas por um *switch* Gigabit entre si. O outro *cluster* que compõe o *grid* possui seis máquinas com um processador Pentium 4 1, 8 Ghz em cada uma. Elas usam o sistema operacional GNU Linux com *kernel* 2.6.8, glibc 2.3.4 e possuem o servidor SSH. As máquinas estão conectadas por um *switch* 100 Mbits entre si.

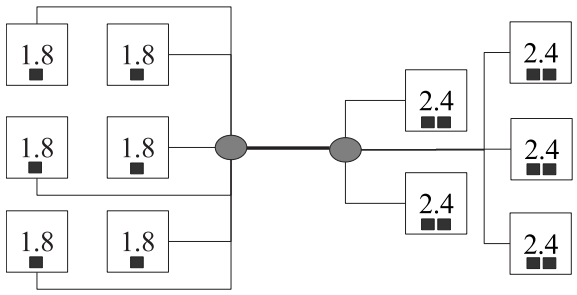


FIGURA 6.9 – Ambiente de grid formando por dois *clusters*, um com cinco máquinas bi-processadas de 2, 4 Ghz e outro com seis máquinas de 1, 8 Ghz.

No gráfico da figura 6.10 compara-se o tempo total em função do número de trajetórias simuladas no *grid*. Neste experimento foram simuladas trajetórias do modelo de infecção intracelular com o nível de infecção inicial igual a um. Pode-se observar que o tempo total de simulação cresce linearmente em função do número de trajetórias. Para simular 100 trajetórias seqüencialmente no processador Xeon 2, 4 Ghz leva em média 9, 8 minutos com desvio padrão de 0, 8 minutos, enquanto no *grid* leva 1, 3 minutos com desvio padrão de 0, 16 minutos. Neste experimento, o *grid* proporcionou um *speedup* de aproximadamente 7, 5.

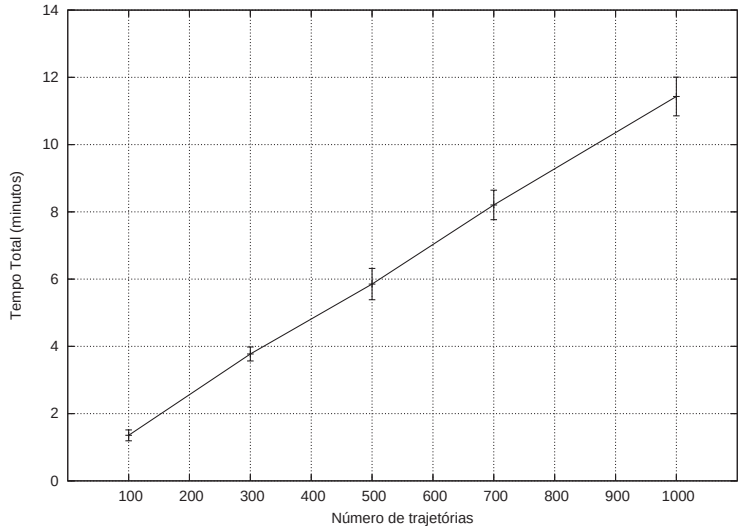
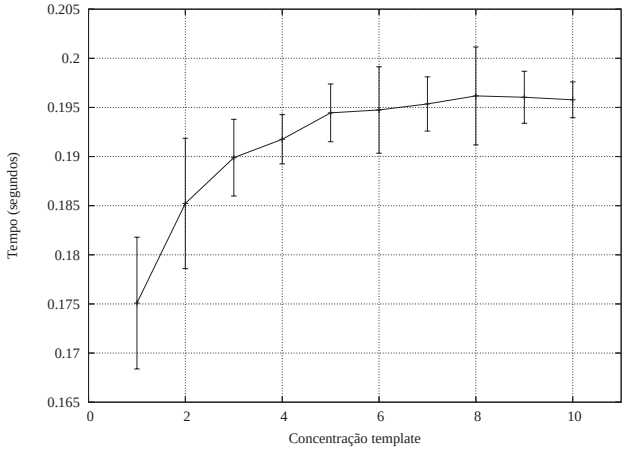


FIGURA 6.10 – Tempo total de simulação conforme o número de trajetórias. Experimentos executados em um *grid* com onze máquinas.

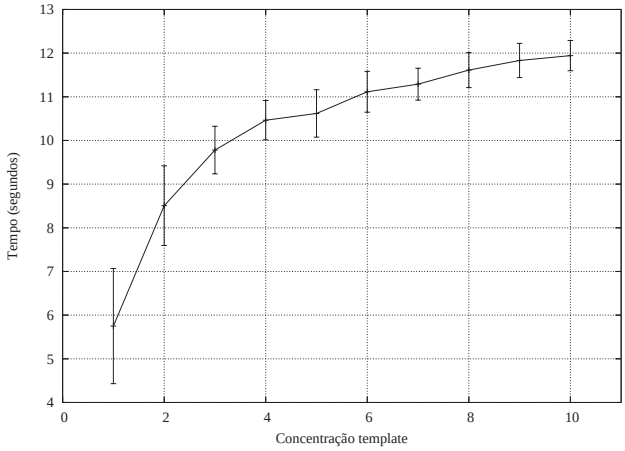
6.2.3 Método Híbrido

Nos gráficos da figura 6.11 é avaliado o desempenho do método híbrido em relação ao método da primeira reação para o modelo da cinética intracelular viral variando o nível de infecção. No gráfico 6.11(a) apresenta-se somente o resultado do método híbrido. No gráfico 6.11(b) mostra-se somente o resultado do método da primeira reação. E no gráfico 6.11(c) comparam-se os dois resultados juntos, com escala logarítmica no eixo vertical.

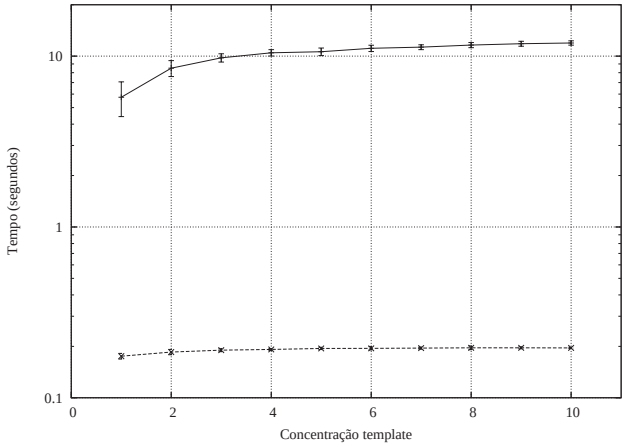
Nos dois métodos o tempo total de simulação aumenta em função do nível de infecção, sendo este aumento maior entre o níveis de infecção um e dois e dois e três. Pela escala do eixo do tempo já pode-se notar o grande ganho que se obteve com o método híbrido, um ganho na ordem de 100 vezes. O critério de partição utilizado foi o mesmo descrito na seção 6.1.3, que mostrou-se um bom critério de partição, pois, mantém o comportamento estocástico do modelo, reconstruindo as curvas das médias dos três principais compostos. E ainda, diminuiu consideravelmente o custo computacional da simulação.



(a) Híbrido



(b) Primeira Reação



(c) Híbrido e Primeira Reação

FIGURA 6.11 – Comparação entre o tempo total de simulação de 10 trajetórias do modelo da cinética intracelular viral conforme o nível de infecção varia.

6.2.4 Modelo do fago λ

Nesta seção serão apresentados alguns dos mecanismos envolvidos no modelo do fago λ. Para o experimento realizado utilizou-se um modelo similar ao apresentado por (ARKIN; ROSS; MCADAMS, 1998), o qual é chamado de modelo completo, pois, é complexo e rico em detalhes. Para maiores informações a respeito dos processos e componentes envolvidos no modelo consulte literatura.

O mecanismo de transcrição dos genes é representado pelas seguintes reações:

$$RNAP + DNA \xrightarrow{k_1} RNAP.DNA_n \tag{6.1}$$

$$RNAP.DNA_n \xrightarrow{k_2} RNAP.DNA_{n+1} \tag{6.2}$$

$$RNAP.DNA_{max} \xrightarrow{k_3} mRNA_{prot} \tag{6.3}$$

$$RNAP.DNA_t \xrightarrow{k_4} RNAP + DNA \tag{6.4}$$

onde, a RNA polimerase ($RNAP$) se liga ao DNA na taxa k_1. Ela começa a transcrever o gene a partir da posição n do DNA e percorre na taxa k_2 toda a extensão do gene. Quando ela passa pela última posição do gene ela libera o $mRNA_{prot}$ correspondente a este gene, e continua a percorrer o DNA até encontrar o último par de base do terminador ($RNAP.DNA_t$). Só pode existir uma $RNAP$ transcrevendo um gene em uma fita de DNA, mas genes diferentes podem ser transcritos simultaneamente. Assim como, fitas de DNA diferentes podem ser transcritas paralelamente.

O processo de tradução é semelhante à transcrição, como pode ser visto nas reações:

$$RIB + mRNA \xrightarrow{k_1} RIB.mRNA_n \tag{6.5}$$

$$RIB.mRNA_n \xrightarrow{k_2} RIB.mRNA_{n+1} \tag{6.6}$$

$$RIB.mRNA_{max} \xrightarrow{k_3} P + mRNA + RIB \tag{6.7}$$

onde, o ribossomo (RIB) se liga ao $mRNA$ com taxa k_1. Ele traduz todo o $mRNA$ na taxa k_2 e ao fim do $mRNA$ libera uma proteína (P), o $mRNA$ e o ribossomo. Um $mRNA$ pode ser traduzido diversas vezes antes de ser degradado.

Para representar todos os componentes do fago λ apresentados na seção 2.5 são necessárias aproximadamente 9.000 reações. Por causa dos processos de transcrição e tradução, o número de reações varia conforme o tamanho dos genes e mRNAs, coletados no Genbank (`www.ncbi.nlm.nih.gov`). Como escrever todas estas reações de forma não automática seria muito trabalhoso, foi desenvolvido um *script* em Python para geração do arquivo de configuração do simulador que contém todas as reações do modelo.

No gráfico da figura 6.12 foi variado o número inicial de moléculas de DNA viral e coletado o tempo total de simulação de uma trajetória em um processador Xeon $2,4$ Ghz para o método da primeira reação. A medição do tempo total não leva em consideração o tempo de inicialização da simulação. Pode-se observar que o tempo total de simulação cresce quase que linearmente em função do número inicial de moléculas de DNA viral.

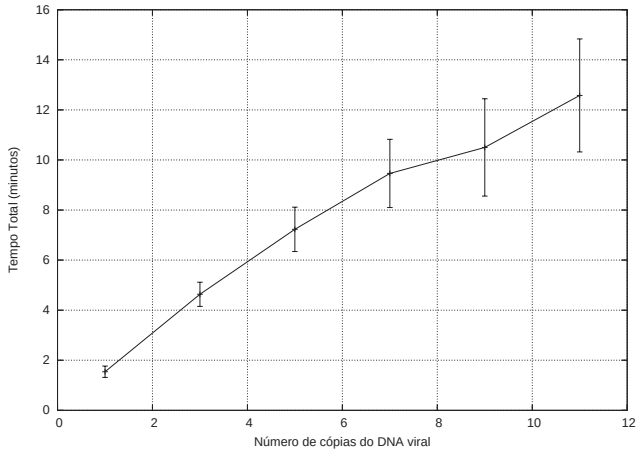


FIGURA 6.12 – Tempo total em função do número de moléculas de DNA viral para o modelo do fago λ com o método da primeira reação.

Para geração de um gráfico (ARKIN; ROSS; MCADAMS, 1998), simulou 1.000 trajetórias para cada número inicial de moléculas de DNA viral, em um supercomputador com 200 nós (Cray T3D). Se fossem simuladas 1.000 trajetórias somente para o nível de infecção igual 11 no mesmo ambiente deste experimento levaria cerca de 9 dias para processar a simulação. Neste caso, um ambiente de *grid*, como do experimento anterior, seria imprescindível.

Capítulo 7

Conclusões

Já faz alguns anos que a bioinformática deixou de ser uma ciência onde se estudava somente como armazenar, gerenciar e disponibilizar informações biológicas. A cada ano que passa, aumenta o conhecimento a respeito do funcionamento das células e organismos, e são dados mais alguns pequenos passos na direção de uma teoria que combine as descrições dos processos celulares com uma descrição computacional e teórica da dinâmica dos processos da vida.

Atualmente, ainda enfrenta-se o desafio de modelar e simular uma única célula de um organismo. O custo computacional de simular processos biológicos aumenta em função da complexidade dos sistemas modelados, onde apenas uma célula de um organismo simples já pode ser considerado um sistema complexo. Muito processos biológicos possuem um comportamento estocástico e a simulação computacional deste tipo de fenômeno é cara computacionalmente.

Para tentar diminuir o problema da falta de poder computacional, existem métodos de simulação de processos biológicos propícios para paralelização e distribuição do processamento. A utilização destes métodos em conjunto com a tecnologia de *grid* pode viabilizar o estudo de processos biológicos complexos num tempo aceitável.

Neste trabalho foi descrita a implementação de uma arquitetura computacional para simulação de redes metabólicas em larga escala. O ambiente alvo desta arquitetura são *grids* de computadores heterogêneos. A arquitetura utiliza o para-

lelismo intra-nó e entre-nós levando em conta as diferenças entre as máquinas. As simulações podem ser feitas usando tanto o modelo estocástico, baseado no algoritmo de Gillespie, quanto o modelo determinístico, utilizando o método de Runge-Kutta. Outra característica importante da arquitetura computacional é a possibilidade da utilização do método híbrido para diminuir o custo computacional do modelo estocástico.

Durante os últimos dois anos, período da confecção deste trabalho, muitos trabalhos que se relacionam a este surgiram, tanto com relação a novos métodos de simulação possíveis de serem implementados no simulador, como novos simuladores, ferramentas e ambientes de simulação. Arquiteturas de simulação que suportam múltiplos métodos de simulação e são capazes de utilizar *grids* para computar as simulações só surgiram recentemente. Para citar alguns exemplos (DHAR et al., 2004b), (KIEHL; MATTHEYSES; SIMMONS, 2004), (GILLESPIE; PETZOLD, 2003) e (YOU; HOONLOR; YIN, 2003).

Num cenário como este onde ocorrem mudanças muito rapidamente, criar uma ferramenta genérica que seja adaptável para teste de novos métodos e modelos, é um grande desafio. As primeiras tentativas de implementação desta ferramenta, visam criar um *framework* onde se possa encaixar ferramentas que implementem as novas funcionalidades. As duas soluções mais conhecidas são: Biospice e Systems Biology Workbench (SBW). Elas começaram os esforços para integração das diversas ferramentas existentes. No SBW foi proposto a Systems Biology Markup Language (SBML), que vem se tornando a linguagem canônica entre as diversas ferramentas.

A implementação do simulador com seus vários métodos de simulação foi validada através da comparação com resultados da literatura e com alguns dos modelos clássicos de Michaelis e Menten. Nos modelos de Michaelis e Menten, observa-se que a média do modelo estocástico oscila sobre o valor do modelo determinístico. Quanto menor for o número de trajetórias simuladas para calcular a média do modelo estocástico, maior será a amplitude da oscilação da média estocástica. No modelo determinístico, é assumida a existência de um grande número de moléculas,

o que não ocorre no caso das enzimas. Por isto, observa-se algumas diferenças entre o resultado do modelo determinístico e estocástico.

Os métodos de Gillespie e híbrido foram validados utilizando o modelo da cinética intracelular viral. Quando comparado os resultados da literatura com os obtidos pelo simulador, observa-se que para o algoritmo de Gillespie os resultados são reproduzidos com exatidão. Já para os resultados do método híbrido existe uma pequena diferença devido ao uso de diferentes métodos de resolução numérica para as equações diferenciais.

Nos experimentos de avaliação de desempenho, verificou-se que o melhor custo-benefício para as máquinas bi-processadas com *hyperthreading* ativado é utilizar duas *threads* para calcular as trajetórias. No ambiente de *grid* utilizado para os experimentos observou-se que o tempo total de simulação cresce linearmente em função do número de trajetórias. Para simular 100 trajetórias seqüencialmente no processador Xeon $2,4$ Ghz levou em média $9,8$ minutos com desvio padrão de $0,8$ minutos, enquanto no *grid* levou $1,3$ minutos com desvio padrão de $0,16$ minutos. Neste experimento, o *grid* proporcionou um *speedup* de aproximadamente $7,5$.

Na avaliação de desempenho do método híbrido em relação ao método da primeira reação para o modelo da cinética intracelular viral variando o nível de infecção. Nos dois métodos o tempo total de simulação aumenta em função do nível de infecção. O método híbrido levou um tempo cerca de 100 vezes menor que o método da primeira reação nos experimentos realizados.

Para o modelo do fago λ foi variado o número inicial de moléculas de DNA viral e coletado o tempo total de simulação de uma trajetória em um processador Xeon 2.4 Ghz para o método da primeira reação. Pode-se observar que o tempo total de simulação cresce quase que linearmente em função do número inicial de moléculas de DNA viral. Se fossem simuladas apenas as 1.000 trajetórias para o nível de infecção igual 11 no mesmo ambiente deste experimento levaria cerca de 9 dias para processar a simulação. Neste caso, pode-se notar claramente que para viabilizar o estudo de um modelo como este já é necessário realizar as simulações

num ambiente de *grid*.

Outra maneira de reduzir o custo computacional é o uso do método híbrido, como visto no experimento de avaliação de desempenho do mesmo. O problema desta solução é definir os critérios de partição para um modelo complexo como o do fago λ. Uma solução para este problema seria usar alguma política para definição automática da partição durante a simulação. Recentemente, foi proposto um algoritmo para particionar o sistema de reações durante a simulação baseado na densidade de probabilidade em (VASUDEVA; BHALLA, 2004).

Neste trabalho foi implementada uma arquitetura computacional para simulação de redes metabólicas em larga escala. Esta arquitetura foi validada com resultados da literatura e avaliado seu desempenho. Os resultados obtidos mostram a viabilidade do uso da arquitetura para simulação de redes metabólicas complexas em larga escala.

Como trabalhos futuros pretende-se implementar o suporte a SBML, e novos métodos de simulação: Método da Próxima Reação, Método τ-Leap, Meta-Algoritmo. Também está planejado facilitar o uso do método híbrido implementando um algoritmo de escolha automática entre o modelo determinístico e estocástico como o proposto em (VASUDEVA; BHALLA, 2004). Outra extensão computacional possível seria implementar o cálculo das taxas de ligação da RNA polimerase no DNA pelo método proposto por (SHEA; ACKERS, 1985) evitando a necessidade do usuário implementar sua própria biblioteca.

Bibliografia

ABBAS, A. *Grid Computing: A pratical guide to technology and applications*. [S.l.]: Charles River Media, 2004. (Network Series). ISBN 1584502762.

ARKIN, A.; ROSS, J.; MCADAMS, H. H. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells. *Genetics*, v. 149, n. 4, p. 33–48, 1998.

BOWER, J. M.; BOLOURI, H. (Ed.). *Computational modeling of genetic and biochemical networks*. [S.l.]: MIT Press, 2001. (Computational molecular biology).

BUTENHOF, D. R. *Programming with POSIX threads*. [S.l.]: pub-AW, 1997. ISBN 0201633922.

DHAR, P. et al. Grid cellware: the first grid-enabled tool for modeling and simulating cellular processes. *Bioinformatics*, 2004.

DHAR, P. et al. Cellware - a multi-algorithmic software for computational system biology. *Bioinformatics*, v. 20, p. 1319–1321, 2004.

FEEL, D. A. System Properties of Metabolic Networks. *Internation conference on Complex Systems*, 1997.

FOSTER, C. K. *The Grid: Blueprint for a New Computing Infrastructure*. [S.l.]: Morgan-Kaufman, 1999.

GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. [S.l.]: Addison-Wesley, 1995. ISBN 0201633612.

GIBBS, W. W. Synthetic life. *Scientific American*, 2004.

GIBSON, M. A.; BRUCK, J. Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *Physical Chemistry*, v. 104, n. 9, p. 1876–1889, 2000.

GILLESPIE, D. T. Exact Stochastic Simulation of Coupled Chemical Reactions. *J. Phys. Chem.*, v. 81, p. 2340–2361, 1977.

GILLESPIE, D. T. Approximate accelerated stochastic simulation of chemically reacting systems. *Chemical Physics*, v. 115, n. 4, p. 1716–1733, 2001.

GILLESPIE, D. T.; PETZOLD, L. R. Improved leap-size selection for accelerated stochastic simulation. *Chemical Physics*, v. 119, n. 16, p. 8229–8234, 2003.

GOBLE, C. *Grids and Biology*. 2002. `www.bbsrc.ac.uk`. Workshop presentation.

HASELTINE, E. L.; RAWLINGS, J. B. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *Chemical Physics*, v. 117, n. 15, p. 6959–6969, 2002.

HOLZNER, S. *Inside XML*. [S.l.]: New Riders, 2001. ISBN 0735710201.

KIEHL, T. R.; MATTHEYSES, R. M.; SIMMONS, M. K. Hybrid simulation of cellular behavior. *Bioinformatics*, v. 20, n. 3, p. 316–322, 2004.

KIERZEK, A. M. STOCKS: STOcastic Kinetic Simulations of biochemical systems with Gillespie algorithm. *Bioinformatics*, v. 18, n. 3, p. 470–481, 2002.

KITANO, H. Computational system biology. *Nature*, v. 420, p. 206–210, 2002.

LEHNINGER, A. L.; NELSON, D. L.; COX, M. M. *Principles of Biochemistry*. 2nd. ed. [S.l.]: Worth Publishers, 1993.

MARR, D. T. et al. Hyper-threading thechnology. *Intel Thechnology Journal*, v. 06, 2002. ISSN 1535-766X.

MENG, T. C.; SOMANI, S.; DHAR, P. Modeling and simulation of biological systems with stochasticity. *In Silico Biology*, v. 4, n. 2, 2004.

MICHAELIS, L.; MENTEN, M. L. Die kinetik der invertinwirkung. *Biochem.*, v. 49, p. 333–369, 1913.

MORTON-FIRTH, C. J.; BRAY, D. Predicting temporal fluctuations in an intracellular signalling pathway. *Theorical Biology*, v. 192, p. 117–128, 1998.

MURRAY, J. D. *Mathematical Biology*. 3rd. ed. [S.l.]: Springer-Verlag, 2002. (Interdisciplinary Applied Mathematics).

NEWMAN, M. E. J.; BARKEMA, G. T. *Monte carlo methods in statistical physics*. [S.l.]: Oxford: Clarendon, 1999. ISBN 0-19-851797-1.

NOVèRE, N. L.; SHIMIZU, T. S. Stochsim: modelling of stochastic biomolecular processes. *Bioinformatics*, v. 17, p. 575–576, 2001.

PRESS, W. H. et al. (Ed.). *Numerical Recipes in C++ The Art of Scientific Computing*. 2nd. ed. Cambridge: Cambridge University Press, 2002.

SANTILLáN, M.; MACKEY, M. C. Why the lysogenic state o phage λ is so stable: A mathematical modeling approach. *Biophysical*, v. 86, p. 75–84, 2004.

SCHILDT, H. *C++: The Complete Reference*. [S.l.]: McGraw-Hill Osborne Media, 2002. ISBN 0072226803.

SHEA, M. A.; ACKERS, G. K. The o_r control system of bacteriophage lambda: A physical-chemical model for gene regulation. *Mol. Biol.*, v. 181, p. 211–230, 1985.

SRIVASTAVA, R.; YOU, L.; YIN, J. Stochastic vs. Deteministic Modeling of Intracellular Viral Kinetics. *Theory Biology*, v. 218, p. 309–321, 2002.

TAKAHASHI, K. et al. A multi-algorithm, multi-timescale method for cell simulation. *Bioinformatics*, v. 20, n. 4, p. 538–546, 2004.

TOMITA, M. et al. E-CELL: software enviroment for whole-cell simulation. *Bioinformatics*, v. 15, n. 1, p. 72–84, 1999.

VASUDEVA, K.; BHALLA, U. S. Adaptive stochastic-deterministic chemical kinetic simulations. *Bionformatics*, v. 20, p. 78–84, 2004.

VEILLARD, D.; BRACK, W.; BUCHCIK, K. *LibXML2*. 2004. `http://www.xmlsoft.org`.

WALSH, C.; WANG, E. Suicide substrates for alanine racemase of e. coli b. *Biochemistry*, v. 17, p. 1313–1321, 1978.

WATSON, J. D.; CRICK, F. H. C. A structure for deoxyribose nucleic acid. *Nature*, v. 171, p. 737, 1953.

YLONEN, T. *OpenSSH*. 1995. `http://www.openssh.com`.

YOU, L.; HOONLOR, A.; YIN, J. Modeling biological systems using dynetica - a simulator of dynamic networks. *Bioinformatics*, v. 19, p. 435–436, 2003.

Apêndice A

A.1 DTD do Arquivo de Configuração do Simulador

```
<!DOCTYPE Simulation [

<!ELEMENT Simulation (Parameters, Composites, Reactions)>
<!ELEMENT Parameters (Global,Methods)>

<!ELEMENT Global (TracedCompoistes)*>
<!ATTLIST Global  traceFilename CDATA #REQUIRED
                  simulationTime CDATA #REQUIRED>

<!ELEMENT composite EMPTY>
<!ELEMENT TracedCompoistes (composite)+>

<!ELEMENT Methods (method)+>
<!ELEMENT method EMPTY>
<!ATTLIST method  name CDATA #REQUIRED
                  seed CDATA #IMPLIED
                  step CDATA #IMPLIED
                  traceFilename CDATA #IMPLIED
                  propensityDiff CDATA #IMPLIED
                  threads CDATA #IMPLIED
                  interval CDATA #IMPLIED>

<!ELEMENT Composites (composite)+>
<!ATTLIST composite  id CDATA #IMPLIED
                     name CDATA #REQUIRED
                     concentration CDATA #IMPLIED>

<!ELEMENT Reactions (Reaction)+>
<!ATTLIST Reactions  comment CDATA #IMPLIED>

<!ELEMENT Reaction (Reactants, Products)>
<!ATTLIST Reaction  k CDATA #IMPLIED
                    k_library_name CDATA #IMPLIED
                    k_function_name CDATA #IMPLIED
                    catalytic CDATA #IMPLIED>

<!ELEMENT Reactants  (reactant)+>
<!ELEMENT reactant EMPTY>
<!ELEMENT Products  (product)+>
<!ELEMENT product EMPTY>
<!ATTLIST reactant  composite CDATA #REQUIRED
                    stoichiometric CDATA #REQUIRED>
<!ATTLIST product  composite CDATA #REQUIRED
```

```
                    stoichiometric CDATA #REQUIRED>
]>
```

A.2 Exemplo de Arquivo de Configuração do Simulador

```
<Simulation>
  <Parameters>
    <Global traceFilename="trace.out" simulationTime="10" >
      <tracedCompoistes>
        <composite name="a" />
        <composite name="b" />
        <composite name="c" />
      </tracedCompoistes>
    </Global>

    <Methods>
      <method name="firstReaction" interval="0.1" seed="28976341277"
              traceFilename="firstReaction.out" />
      <method name="directMethod" interval="0.1" seed="28976341277"
              traceFilename="directMethod.out" />
      <method name="rungeKutta4" step="0.01"
              traceFilename="rungeKutta4.out" />
      <method name="hybridRk4" interval="0.1" seed="28976341277"
              step="0.01" propensityDiff="100"
              traceFilename="hybridRk4.out" />
      <method name="multiThreadHybridRk4" propensityDiff="100"
              seed="28976341277" step="0.01" interval="0.1"
              threads="2" traceFilename="multiThreadHybridRk4.out" />
    </Methods>
  </Parameters>

  <Composites>
    <composite id="0" name="a" concentration="100" />
    <composite id="1" name="b" concentration="0" />
    <composite id="2" name="c" concentration="400" />
    <composite id="3" name="d" concentration="200" />
  </Composites>

  <Reactions comment="">
    <Reaction k="0.5">
      <Reactants>
        <reactant composite="a" stoichiometric="1" />
        <reactant composite="c" stoichiometric="1" />
      </Reactants>
      <Products>
        <product composite="b" stoichiometric="1" />
      </Products>
    </Reaction>
    <Reaction k="0.2">
      <Reactants>
        <reactant composite="a" stoichiometric="2" />
        <reactant composite="b" stoichiometric="1" />
      </Reactants>
      <Products>
        <product composite="c" stoichiometric="1" />
      </Products>
```

```
    </Reaction>
    <Reaction k="0.6">
      <Reactants>
        <reactant composite="c" stoichiometric="1" />
      </Reactants>
      <Products>
        <product composite="d" stoichiometric="2" />
      </Products>
    </Reaction>
  </Reactions>
</Simulation>
```

A.3 Exemplo de Código Gerado pelo *reactions2diffeqs*

```
#define PARS_SIZE 10
int pair[][10] = {
        {1, 0, 1, 2, 2, 0, 1, 1},
        {1, 0, 1, 2, 2, 0, 1, 1},
        {1, 0, 1, 2, 2, 0, 1, 1, 1, 2},
        {1, 2}
};
#define K_SIZE 6
float k[][6] = {
        {-0.5, 2, -0.2, 2},
        {0.5, 2, -0.2, 2},
        {-0.5, 2, 0.2, 2, -0.6, 1},
        {0.6, 1}
};
#include <math.h>

#include <MetabolicSimulator/DifferentialEquations.h>


void DifferentialEquations::calculate(std::vector<float> &y_,
                                      std::vector<float> &dydx_,
                                      std::vector<int> &composites_) {
  for(int i=0; i < composites_.size(); i++) {
    int comp = composites_[i];
    float f = 0.0f;
    int curp = 0;
    for(int j=0; j < K_SIZE; j= j+2) {
      float cur = k[comp][j];
      for(int l=0; l < ((int)k[comp][j+1]); l++) {
        if(pair[comp][curp] == 1)
          cur *= y_[pair[comp][curp+1]];
        else
          cur *= pair[comp][curp] *
                 powf(y_[pair[comp][curp+1]], pair[comp][curp]);

        curp +=2;
      }
      f += cur;
    }
    dydx_[comp] = f;
  }
```

```
}
```

A.4 Exemplo de Código de uma Biblioteca Dinâmica para Calculo da Taxa Variável de uma Reação

```
#include <iostream>
#include <vector>
using std::vector;
using std::cout;
using std::endl;

#include <MetabolicSimulator/Composite.h>
#include<cmath>

#define e               2.71828183          // e
#define RT              2577.34             // T=310K  R=8,314 J/(K mole)

float table_GS[] = {0.0,-10.9,-12.1,-11.7,-10.1,-12.5,-22.9,-20.9,-22.8,-23.7};
int   table_CRO2[]={0,  1,    1,    0,    0,    0,    2,    1,    1     0};
int   table_CI2[] ={0,  0,    0,    1,    1,    0,    0,    1,    1,    2};
int   table_RNAP[]={0,  0,    0,    0,    0,    1,    0,    0,    0,    0};
float table_k1[] = {0.0,0.0,  0.0,  0.0,  0.0,  0.011,0.0,  0.0,  0.0,  0.0};

vector<int>  comps;

// convert molecules to molar
float moleculesToNanomolar(float conc_, float volume_) {
  return (conc_ / (volume_ * 6e23) );
}

float z_PL(vector<Composite> &composites_, float volume_) {
  float cro2 =
    moleculesToNanomolar(composites_[comps[1]].getConcentration(), volume_);
  float ci2  =
    moleculesToNanomolar(composites_[comps[2]].getConcentration(), volume_);
  float rnap =
    moleculesToNanomolar(composites_[comps[0]].getConcentration(), volume_);

  float sum = 0.0f;
  for(int i=0; i < 10; i++) {
    float delta_gs = (-table_GS[i] * 4184)  / RT ;  // 1 kcal = 4184

    float cur = powf(e, delta_gs) *  powf(cro2, table_CRO2[i]) *
                powf(ci2, table_CI2[i]) * powf(rnap, table_RNAP[i]);

    sum += cur;
  }
  return sum;
}

int find(string name_, vector<Composite> &composites_) {
  for(int i=0; i < composites_.size(); i++) {
    if(composites_[i].getName().compare(name_) == 0) {
      return composites_[i].getId();
    }
```

```
51    }
52    return -1;
53  }
54
55  extern "C" void init(vector<Composite> &composites_) {
56    if(comps.size() > 0)
57      return;
58
59    int c = find("RNAP_", composites_);
60    if(c < 0) {
61      cout << "[ERROR] k_PL_PL: can't find RNAP composite." << endl;
62      exit(-5);
63    }
64    comps.push_back(c);
65
66    c = find("CRO_CRO", composites_);
67    if(c < 0) {
68      cout << "[ERROR] k_PL_PL: can't find CRO_CRO composite." << endl;
69      exit(-5);
70    }
71    comps.push_back(c);
72
73    c = find("CI_CI", composites_);
74    if(c < 0) {
75      cout << "[ERROR] k_PL_PL: can't find CI_CI composite." << endl;
76      exit(-5);
77    }
78    comps.push_back(c);
79  }
80
81  //
82  //  A Probabilistic Model of a Prokaryotic Gene and Its Regulation
83  //     Michael Andrew Gibson and Jehoshua Bruck
84  //
85  extern "C" float k1_PL(vector<Composite> &composites_, float volume_) {
86    int states = 10;
87    float pstates[states];
88
89    float z = z_PL(composites_, volume_);
90
91    float delta_gs = (-table_GS[5] * 4184)  / RT ;  // 1 kcal = 4184
92    float cro2 =
93      moleculesToNanomolar(composites_[comps[1]].getConcentration(), volume_);
94    float ci2  =
95      moleculesToNanomolar(composites_[comps[2]].getConcentration(), volume_);
96    float rnap =
97      moleculesToNanomolar(composites_[comps[0]].getConcentration(), volume_);
98    pstates[5] = powf(e, delta_gs) *  powf(cro2, table_CRO2[5]) *
99                 powf(ci2, table_CI2[5]) * powf(rnap, table_RNAP[5]);
100
101   pstates[5] /= z;
102
103   float k1_PL = table_k1[5] * pstates[5];
104   return k1_PL;
105 }
```