Solução eficiente de modelos estruturados a partir de descritores tensoriais combinados a estruturas arborescentes *

Ricardo M. Czekster, Paulo Fernandes, Afonso Sales, Thais Webber

[1]Pontifícia Universidade Católica do Rio Grande do Sul – Faculdade de Informática
Av. Ipiranga, 6681, Prédio 32 – 90619-900 – Porto Alegre – RS – Brasil

{ricardo.czekster, paulo.fernandes, afonso.sales, thais.webber}@pucrs.br

Abstract. *Structured Markovian formalisms have emerged to facilitate the representation of large state spaces and transitions. However, structuring introduces algorithmic challenges, such as detection of reachable states, as well as specialized numerical solutions. Recent researches have proposed the use of Multi-valued Decision Diagrams (MDD) for the compact storage of reachable states. This paper presents an efficient technique for mapping transitions between states represented by MDD, that can be adapted to the solution via iterative methods, or even direct methods. Preliminary results show significant gains in processing time and memory considering the ratio between state space size and amount of reachable states.*

Resumo. *Formalismos Markovianos estruturados surgiram para facilitar a representação de modelos com inúmeros estados e transições. Entretanto, esta estruturação introduz desafios algorítmicos, tais como detecção de estados válidos, bem como soluções numéricas especializadas. Pesquisas recentes propõem utilizar Diagramas de Decisão Multi-valorados (MDD) para armazenamento compacto deste estados. Neste artigo apresenta-se uma técnica eficiente de mapeamento das transições entre estados, permitindo adequar a solução via métodos iterativos, ou mesmo diretos. Resultados preliminares mostram ganhos significativos em tempo de processamento e memória em função da relação entre o conjunto de estados total e válidos do modelo.*

1. Introdução a Formalismos Estruturados

O uso de Cadeias de Markov é disseminado na modelagem de problemas complexos de diversas áreas, tais como: biologia, economia, linguística, computação no âmbito de avaliação de desempenho, entre outras aplicações. Entretanto este formalismo apresenta o problema conhecido como *explosão do espaço de estados* o que impossibilita a modelagem de sistemas cujo espaço de estados ultrapasse a capacidade de processamento e memória das máquinas disponíveis, logo este problema dificulta o tratamento numérico do modelo [Stewart 1994].

Neste contexto, formalismos com maior nível de abstração, também chamados de formalismos Markovianos estruturados, surgem como uma alternativa interessante para mitigar este problema de modelagem, já que possibilitam utilizar módulos

*Este trabalho é financiado pela Petrobras (Conv. 0050.0048664.09.9). Paulo Fernandes é também financiado pelo CNPq-Brasil (PQ 307284/2010-7) e Afonso Sales recebe financiamento da CAPES-Brasil (PNPD 02388/09-0). A ordem dos autores é meramente alfabética.

ou subsistemas para representar um grande sistema. Alguns formalismos, como é o caso das *Redes de Autômatos Estocásticos* (do inglês, *Stochastic Automata Networks* - SAN) [Plateau 1985] apresentam primitivas diferenciadas para representar comportamentos dependentes nas transições entre os estados. SAN tem como primitivas básicas a possibilidade de definir eventos que habilitam transições de forma local em um subsistema ou autômato, e eventos sincronizantes que habilitam transições simultâneas entre estados em autômatos diferentes. Além disto, taxas para os eventos podem determinar também dependências funcionais entre autômatos na rede, possibilitando a mudança de estado em pelo menos um autômato, ou em mais, dependendo do estado atual de outros [Brenner et al. 2005]. Modelos descritos em SAN podem ser mapeados para uma cadeia de Markov correspondente, que se caracteriza por representar todos os estados possíveis do modelo SAN, considerando as combinações dos estados locais de todos os autômatos da rede [Plateau 1985, Stewart 1994].

Outro fator relevante na solução de modelos estruturados como SAN é que nem sempre todos os estados combinados oriundos de um modelo são considerados possíveis ou válidos [Sales and Plateau 2009]. Em um modelo com vários subsistemas, chamamos de *espaço de estados total* a combinação de todos os estados de todos os seus subsistemas. De acordo com a natureza dos eventos e as transições que estes habilitam, determinados estados do espaço de estados podem ser *inatingíveis*, dando origem ao chamado *espaço de estados atingíveis*. A descoberta e o armazenamento destes estados válidos de uma maneira eficiente pode ser empregada através do uso de estruturas arborescentes chamadas *Diagramas de Decisão Multi-valorados* (do inglês, *Multi-valued Decision Diagrams* - MDD) [Miner and Ciardo 1999b, Sales and Plateau 2009]. Estas estruturas são extremamente compactas na representação de tais estados, entretanto, não apresentam informações sobre as transições do modelo.

Neste contexto, note que tanto as cadeias de Markov, quanto as SAN, podem ser solucionadas via métodos diretos ou iterativos através de multiplicações vetor-matriz, ou mesmo por simulação [Fernandes et al. 1998, Buchholz et al. 2000, Buchholz and Dayar 2004, Fernandes et al. 2008]. Entretanto, no caso dos formalismos estruturados, estes podem tirar proveito de álgebra tensorial para o armazenamento compacto de modelos, consequentemente, apresentando soluções numéricas eficientes, tanto em memória quanto em tempo de processamento.

SAN é um formalismo pioneiro no uso de tensores para representar modelos algebricamente, podendo armazená-lo de forma compacta apesar de aumentar a complexidade dos algoritmos para sua solução. Algoritmos que lidam com tensores oriundos de modelos estruturados são especializados para tratar com matrizes esparsas e operações de multiplicação vetor-matriz não convencionais, a exemplo da multiplicação de vetores por somas e produtos tensoriais [Fernandes et al. 1998]. Na literatura encontram-se maiores detalhes sobre os algoritmos chamados *Shuffle* e *Split*, especializados para solução de modelos em formato tensorial (os modelos também são chamados de descritores Markovianos) [Fernandes et al. 1998, Czekster et al. 2007, Czekster et al. 2010b].

Apesar destes algoritmos lidarem eficientemente com tais descritores, o espaço de estados dos modelos influencia diretamente na complexidade computacional de solução principalmente em termos de memória para armazenamento de vetores. A solução por cadeia de Markov pode demandar menos memória em relação a estes vetores utilizados,

pois a cadeia apenas contém os estados atingíveis do modelo SAN [Buchholz et al. 2000]. Apesar disto, existe uma dificuldade em resolver numericamente modelos em cadeias de Markov suficientemente grandes ao ponto de impossibilitar o armazenamento da matriz de transição, vetores e eventuais estruturas auxiliares. No caso de SAN, modelos com poucos estados válidos em relação ao espaço de estados total que pode ser gigantesco, também podem se tornar impraticáveis de resolver dado a necessidade de armazenar e acessar igualmente gigantescos vetores de probabilidade, apesar do modelo ser armazenado em poucos Kb usando tensores [Czekster 2010, Sales 2009, Webber 2009].

Neste artigo propõe-se uma técnica eficiente para o mapeamento das transições entre os estados representados por MDD, permitindo adequar a solução de modelos via métodos iterativos, ou mesmo diretos. O objetivo é manter a modularidade na modelagem, a estruturação através de álgebra tensorial e o uso de primitivas de modelagem, mas também possibilitar flexibilidade para uma solução eficiente que lide com o espaço de estados atingíveis armazenado de maneira compacta. Resultados preliminares para classes de modelos mostram ganhos significativos em tempo de processamento e memória em função da relação entre o conjunto de estados total e válidos do modelo.

As seções neste artigo estão divididas como segue. Seção 2 retoma conceitos de modelagem através de Redes de Autômatos Estocásticos e discute brevemente sua representação tensorial. Seção 3 ressalta a necessidade da detecção do espaço de estados atingíveis em modelos estruturados, bem como apresenta um exemplo de representação arborescente. Seção 4 apresenta uma visão geral dos algoritmos existentes para solução de descritores e seus problemas relacionados ao uso de memória e ao tempo de processamento. Seção 5 mostra experimentos para classes de modelos, especialmente aqueles com espaço atingível reduzido em relação ao espaço total de estados, onde os resultados numéricos obtidos são promissores, tornando válida a abordagem de solução proposta. Seção 6 aponta para novas direções no contexto de solução de modelos estruturados onde a priorização é manter o poder de modelagem aliado a uma solução otimizada.

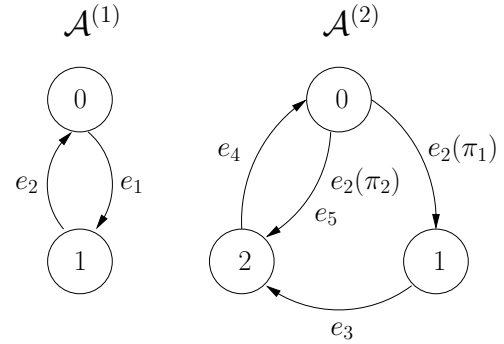
2. Modelagem com Redes de Autômatos Estocásticos e Álgebra Tensorial

Redes de Autômatos Estocásticos (SAN) é um formalismo baseado em Cadeias de Markov [Plateau 1985], que permite a definição de mais de um componente (ou entidade) na forma de autômatos estocásticos. Estas entidades podem interagir entre si através de primitivas de modelagem conhecidas como eventos sincronizantes, bem como podem ter um comportamento independente através de eventos locais. Além disto, eventos são determinados por suas taxas (ou probabilidades) de ocorrência, que no caso das SAN podem também dispor de taxas funcionais para o disparo dos eventos, ou seja, taxas que criam uma certa dependência com os estados correntes de autômatos da rede [Benoit et al. 2004].

A complexidade inerente em modelar sistemas com grande número de estados trouxe a necessidade de ampliação dos formalismos para permitirem uma estruturação mais complexa do que simplesmente uma matriz de transição entre estados, já que componentes podiam ser modelados separadamente mas precisavam ser avaliados em conjunto. Logo, de forma pioneira, as SAN foram mapeadas para os chamados descritores Markovianos [Plateau 1985], que são estruturas obtidas através da representação do modelo em um conjunto de matrizes que operam através de álgebra tensorial. Este avanço possibilitou

grande economia na memória para armazenamento do conjunto de eventos e transições que geram a cadeia de Markov implícita ao modelo. Cabe salientar que as matrizes que compõem um descritor normalmente são de baixa ordem devido à modularização aplicada na modelagem, e também esparsas, visto que os eventos podem ocorrer eventualmente em cada um dos autômatos em uma ou apenas algumas transições. SAN também opera matrizes com elementos funcionais oriundos das taxas de eventos que dependem dos estados ou condições de outras entidades para disparar, para isto introduziu-se também o uso de álgebra tensorial generalizada [Fernandes et al. 1998, Benoit et al. 2004].

A Figura 1 [Sales and Plateau 2009] mostra uma pequena rede de autômatos estocásticos para exemplificar dois autômatos, cinco eventos (4 locais e 1 sincronizante). O evento sincronizante e_2 é apresentado com probabilidades de rotação π_1 e π_2 em diferentes transições. O evento local e_1 é apresentado com uma função associada, esta descrita pela notação provida pelo formalismo através da ferramenta PEPS [Brenner et al. 2007, Czekster et al. 2009]. A interpretação de uma função pode ser vista como avaliação de uma expressão em linguagens de programação, *e.g.*, na linguagem C. Cada comparação é avaliada como *true* (valor 1), ou como *false* (valor 0). O disparo da transição do estado 0 para o estado 1 ocorre com taxa λ se o automato $\mathcal{A}^{(2)}$ esta no estado 0, ou taxa γ se o automato $\mathcal{A}^{(2)}$ esta no estado 2. Se o automato $\mathcal{A}^{(2)}$ esta no estado 1, a transição de 0 para 1 não ocorre, ou seja, taxa igual a zero.



tipo	evento	taxa
loc	e_1	f
sin	e_2	μ
loc	e_3	σ
loc	e_4	δ
loc	e_5	τ

$$f = \Big[\big(st\ \mathcal{A}^{(2)} == 0\big) \times \lambda\Big] + \Big[\big(st\ \mathcal{A}^{(2)} == 2\big) \times \gamma\Big]$$

Figura 1. Exemplo de modelo SAN [Sales and Plateau 2009]

A transcrição do modelo para o descritor Markoviano pode ser transparente para o usuário menos experiente, ou pode ser feita diretamente por modeladores mais experientes, o importante é que o modelo será dito estruturado e existem diferentes maneiras de tratar este modelo armazenado de maneira mais eficiente, em matrizes esparsas e operadores tensoriais [Stewart 1994, Czekster 2010].

Dentre os exemplos de aplicações de SAN para modelagem de realidades, tem-se: avaliação de programas paralelos [Baldo et al. 2005], modelagem de sistemas tolerantes a falhas [Bertolini et al. 2004a], modelagem e geração de casos de teste estatísticos [Bertolini et al. 2004b], modelagem de desempenho de escalonamento e balanceamento de carga em sistemas operacionais [Chanin et al. 2006], avaliação de protocolos de redes sem fio [Dotti et al. 2005, Dotti et al. 2011], qualidade de serviço em redes [Czekster et al. 2011, Mokdad et al. 2001], modelagem de equipes em ambientes com desenvolvimento de software distribuído [Czekster et al. 2010a, Fernandes et al. 2011].

3. Detecção do Espaço de Estados Atingíveis em Modelos Estruturados

Uma característica dos formalismos estruturados é a existência de um *espaço de estados total* cujo número de estados é calculado pela combinação dos estados locais de todos os autômatos do modelo, e contido neste, um conjunto de estados válidos, formando o que chamamos de *espaço de estados atingíveis* [Fernandes et al. 1998]. Utilizando álgebra tensorial este fato pouco ou nada afeta o armazenamento das transições a partir de tensores. Entretanto, ao tentar-se resolver o modelo, vetores de probabilidade do tamanho do espaço total são necessários, pois os tensores representam implicitamente uma matriz esparsa da ordem deste espaço de estados total [Buchholz et al. 2000]. Logo, modelagens estruturadas podem introduzir desvantagens no uso de memória para a solução de modelos estocásticos principalmente quando se tratam de modelos com grande diferença entre os espaços de estados total e atingível.

É importante ressaltar que modelos estruturados apresentam uma cadeia de Markov implícita ao seu descritor, oriunda do mapeamento de todos os estados atingíveis do modelo [Fernandes et al. 1998]. Estes podem ser obtidos de forma trivial a partir do disparo de todos os eventos, a partir de todos os estados do modelo, e podem ser armazenados em uma matriz esparsa ou outra estrutura de dados mais complexa para evitar o consumo excessivo de memória. Neste âmbito, o primeiro passo é detectar quais são os estados válidos de um modelo estruturado com grande espaço de estados, onde torna-se difícil realizar manualmente, e armazená-los em alguma estrutura de dados eficiente.

Os estados atingíveis podem ser armazenados através de estruturas arborescentes chamadas de *diagramas de decisão multi-valorados* (do inglês, *Multi-valued Decision Diagrams* - MDD) [Miner and Ciardo 1999b], onde os estados são agrupados de maneira compacta e acessados de forma facilitada através de algoritmos de busca especializados. MDD é um grafo direcionado e acíclico, que apresenta nodos definidos em níveis, sendo estes nodos classificados em não-terminais (níveis mais altos) e terminais (Nível 0).

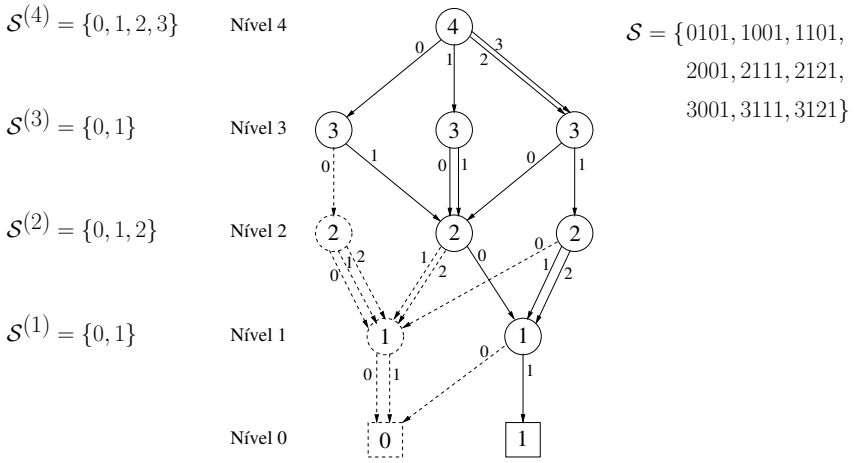


Figura 2. Espaço de estados atingíveis $\mathcal{S}$ representado por um MDD

A Figura 2 mostra a representação do espaço de estados atingíveis $\mathcal{S}$, onde o níveis mais altos representam os autômatos e as transições entre níveis os possíveis estados dos autômatos. O nível mais baixo apresenta dois *nodos terminais*, 0 e 1, representando respectivamente estados inatingíveis e atingíveis. Um elemento de $\mathcal{S}$ é um estado obtido

a partir dos rótulos dos arcos que compõem um caminho partindo do nodo raiz (*i.e.*, nodo do nível mais alto) em direção ao nodo terminal 1. Os nodos e arcos pontilhados formam os elementos que não são representados por $\mathcal{S}$, *i.e.*, são os caminhos que levam ao nodo terminal 0, os quais formam os estados que não são representados por $\mathcal{S}$. Este exemplo representa um modelo com quatro autômatos, sendo cada autômato composto por seus estados individuais $\mathcal{S}^{(i)}$.

Ao tratar-se o espaço atingível armazenando em MDD [Sales and Plateau 2009], torna-se possível a escolha de soluções numéricas adequadas para cada modelo estruturado, considerando características como: tamanho do modelo em termos de espaço total e atingível, bem como memória a ser utilizada pelas estruturas auxiliares.

4. Soluções Numéricas tradicionais para Modelos Estruturados Markovianos

Descritores Markovianos oriundos de modelos estruturados podem ser solucionados através de métodos iterativos [Stewart 1994, Saad 1995] (*e.g.*, método da Potência, Arnoldi, GMRES) que operam a multiplicação de vetores por matrizes na busca de um vetor de probabilidades contendo a solução estacionária (ou transiente) dado uma tolerância para precisão dos resultados [Stewart 1994]. Entretanto, a solução por métodos iterativos é combinada com algoritmos especializados para multiplicação vetor-descritor onde as matrizes operam entre si com álgebra tensorial. Estes algoritmos são implementados de forma a reduzir o número de multiplicações de ponto flutuante na medida que operam apenas sobre elementos não-nulos da estrutura tensorial [Fernandes et al. 1998, Buchholz et al. 2000, Czekster et al. 2007, Czekster et al. 2010b].

Os algoritmos mais utilizados na literatura para multiplicação vetor-descritor são os algoritmos *Shuffle* [Fernandes et al. 1998] e *Split* [Czekster et al. 2010b] que lidam com o descritor de formas distintas acessando os tensores, mas operam igualmente com vetores do tamanho do espaço de estados total, o que se torna um fator limitante nestas soluções dependendo do modelo, principalmente aqueles modelos com espaço atingível conhecido pois este poderia ser mapeado para uma cadeia de Markov bem reduzida em relação ao tamanho do modelo estruturado. Sabe-se que cadeias de Markov, dependendo do número de estados podem ser resolvidas por simples iterações vetor-matriz bem como com outras otimizações relacionadas às propriedades da matriz [Stewart 1994].

Existem avanços recentes realizados no armazenamento de estados válidos em MDD no contexto de Redes de Autômatos Estocásticos [Sales and Plateau 2009]. Entretanto, a solução tradicional ainda precisa ser adaptada para levar em consideração o espaço atingível ao alocar os vetores para a multiplicação vetor-descritor. Mais do que isto, um passo natural para a otimização dos métodos de solução de modelos estruturados pode ser justamente acoplar esta representação MDD para gerar eficientemente a cadeia de Markov adjacente ao modelo, para aqueles casos onde a estruturação favorece a modelagem porém causa gargalos no cálculo dos resultados [Buchholz et al. 2000, Czekster et al. 2007].

5. Solução através do Mapeamento de Transições entre Estados Representados por MDD

O objetivo desta proposta é acelerar o processamento e solução de modelos estruturados através da transcrição do modelo para um descritor Markoviano e respectivo MDD de estados válidos, o que comprovadamente reflete em uma economia de

memória [Sales and Plateau 2009]. Esta estruturação serve para gerar o *descritor de atingibilidade* (do inglês, *Reachability descriptor* - RD) construído a partir deste descritor Markoviano, que combinado ao MDD torna possível aplicar soluções otimizadas para vetor-descritor ou vetor-matriz. O descritor de atingibilidade é uma estrutura também tensorial, criada para facilitar o mapeamento de transições entre os estados na criação do MDD, podendo estabelecer uma interface entre o descritor e o espaço atingível.

Outros trabalhos também utilizam estruturas compactas, *e.g.*, *Diagramas de Matrizes* (do inglês, *Matrix Diagrams* - MxD) para geração e representação das transições do modelo [Miner and Ciardo 1999a, Miner 2002]. Entretanto, na representação por MxD, não são consideradas transições funcionais entre os estados (*i.e.*, transições que podem ou não ocorrer em função do estado de outros subsistemas). Transições funcionais são primitivas de modelagem muito eficientes na descrição de modelos. Sales e Plateau desenvolveram uma representação compacta utilizando MDDs para os estados atingíveis de um modelo SAN permitindo a descrição de transições funcionais [Sales and Plateau 2009].

Os algoritmos implementados neste trabalho priorizam o uso de estruturas de armazenamento esparso, chamadas de *matrizes HBF* (*Harwell-Boeing Format*) [Stewart 1994], onde apenas elementos não-nulos e seus respectivos índices de linha/coluna são considerados. Métodos de acesso e atualização destas estruturas foram implementados para então extrair, a partir da representação estruturada, a cadeia de Markov correspondente à SAN, sem perdas de desempenho quando comparados aos algoritmos que percorrem estado a estado da SAN. A ideia geral para a solução de SAN utilizando esta abordagem pode ser descrita nos seguintes passos:

> **Passo 1:** dado a descrição de um modelo SAN representado por um descritor Markoviano e um estado inicial, calcula-se o espaço de estados atingíveis do modelo representando-o em um MDD [Sales and Plateau 2009]. Para o cálculo destes estados, um descritor de atingibilidade[1] é gerado a partir do descritor Markoviano;
>
> **Passo 2:** a partir deste MDD e utilizando o descritor de atingibilidade, calcula-se para cada estado atingível quais são as transições que levam para os demais estados do modelo e armazenam-se estas transições em uma matriz HBF;
>
> **Passo 3:** utilizando esta matriz de transição em HBF, calcula-se através de um método iterativo (*e.g.*, método da Potência) o vetor de probabilidade resultante do modelo (neste caso, a solução no seu estado estacionário).

É importante ressaltar que o **Passo 1** já foi comprovadamente apresentado como extremamente eficiente em termos de memória e processamento [Sales and Plateau 2009]. Também cabe frisar que se o **Passo 2** fosse feito de maneira ingênua (*i.e.*, analisando quais são as possíveis transições para cada estado possível do modelo) certamente seria o gargalo de todo o processo em termos de tempo. Entretanto, graças ao descobrimento e armazenamento do estados atingíveis do modelo por um MDD, percorremos *apenas* este conjunto de estados para determinarmos as transições do modelo, tornando assim a implementação do algoritmo de obtenção da matriz de transição extremamente eficiente.

[1]Este descritor de atingibilidade é extremamente importante para aceleração na descoberta das transições do modelo. Maiores detalhes sobre o descritor de atingibilidade podem ser encontrados em [Sales and Plateau 2009].

Outro detalhe importante é que além de ser eficiente em memória, a matriz de transição descrita em HBF permite o cálculo de maneira extremamente eficiente da solução do modelo (**Passo 3**).

Finalmente, esta maneira de extrair a solução de modelos torna-se muito flexível, pois reúne eficientes técnicas de armazenamento e representação de modelos com funções, e pode ser usada em ferramentas numéricas para diferentes formalismos estruturados, bastando que os mesmos sejam convertidos para um formato tensorial.

5.1. Experimentos e Análise de Resultados

Os experimentos desta seção foram realizados em uma máquina *Intel Pentium* 3.2 GHz, 4 Gb de RAM e 2 Mb de memória *cache*. O protótipo denominado YAMS (*Yet Another Markovian Solver*) utilizou-se da compilação e geração do descritor Markoviano descrito no ambiente da ferramenta PEPS [Brenner et al. 2007], bem como integra métodos de geração de MDD e do descritor de atingibilidade RD.

As novas rotinas de acesso a estas estruturas foram implementadas de forma a gerar uma matriz de transição HBF com estados válidos a partir de um modelo estruturado, e assim resolvê-la via métodos iterativos clássicos. YAMS foi codificado em C++ e para estes experimentos utilizou-se a versão $4.0.4$ do compilador g++ (GCC – *The GNU Compiler Collection*) com opção de otimização (-O3) e *dynamic linkage*.

Os modelos estruturados em SAN selecionados para os experimentos foram: *Redes ad hoc sem fio* (abreviado como Ad) [Dotti et al. 2005], Primeiro Servidor Disponível - do inglês *First Available Server* (FAS) [Benoit et al. 2004], *Kanban* (abreviado como Kn) [Sales and Plateau 2009], Jantar dos Filósofos - do inglês *Dining Philosophers* (DP) [Sales and Plateau 2009], Compartilhamento de Recursos - do inglês *Resource Sharing* (RS) [Sales and Plateau 2009].

(i) **Redes ad hoc sem fio** (Ad) – este modelo representa um conjunto de nodos móveis em uma rede sem fio sobre o padrão IEEE 802.11 para redes *Ad hoc*, onde um nodo *Fonte* gera pacotes na taxa permitida pelo padrão, com dois estados: *inativo* e *transmitindo*. Os pacotes são repassados na rede pelos nodos do tipo *Retransmissor* com três estados: *inativo*, *recebendo* e *transmitindo*, até o último nodo *Destino*. Este tem dois estados apenas: *inativo* e *recebendo*. O espaço de estados total é dado por $[2^2 \times 3^{N-2}]$ estados. Utiliza-se duas versões deste modelo, com e sem taxas funcionais para os eventos (logo tem-se descritores diferentes mas com a mesma semântica de modelo), variando o número N de nodos na rede.

(ii) **Primeiro Servidor Disponível** (FAS) – modelo que analisa a disponibilidade de servidores considerando uma rede contendo N servidores. Cada servidor tem dois estados: $I^{(i)}$ (inativo) e $B^{(i)}$ (ocupado). Neste modelo, os pacotes chegam e são roteados para os servidores na ordem em que estes estiverem configurados, em sequência, até rotear para um servidor inativo, ou seja, que não está ocupado. O espaço de estados é dado por 2^N estados. Este modelo foi estendido variando o número de servidores N.

(iii) **Kanban** (Kn) – modelo de um sistema de manufatura, com quatro estações (autômatos) semelhantes, onde N *kanbans* (estados que representam rótulos possíveis, do inglês *tags*) controlam o fluxo de peças. O processo apresenta peças que chegam primeiro na estação 1, depois cada peça é separada em duas partes

que entram nas estações 2 e 3, e então, essas duas partes são conectadas em uma única peça novamente que entra na estação 4. Neste ponto, a peça sai do sistema de manufatura. O espaço de estados total é dado por $[(N+1)^{16}]$ e $[(N+1)^{12}]$ estados para as versões com taxas constantes e funcionais respectivamente. Varia-se este modelo pelo número N de kanbans configurado em cada estação.

(iv) **Jantar dos Filósofos** (DP) – modelo do problema clássico de compartilhamento de recursos com exclusão mútua, onde K filósofos sentam em uma mesa circular para comer e pensar, tendo três estados cada: *L* (acessando garfo esquerdo), *R* (acessando garfo direito) ou *T* (pensando). Um filósofo apenas consegue comer se tiver os dois garfos reservados. O espaço de estados total é dado por $[3^K]$ estados. Varia-se este modelo pelo número K de filósofos na mesa.

(v) **Compartilhamento de Recursos** (RS) – modelo de compartilhamento de recursos com diferentes configurações de rede, sendo P o número de processos, com dois estados: *inativo* e *ocupado* - e R como número de recursos compartilhados. O espaço de estados é dado por $[2^P \times (R+1)]$ e $[2^P]$ estados para as versões com taxas constantes e funcionais respectivamente. Este modelo foi estendido para diferentes configurações de P e R.

A Tabela 1 mostra os resultados para os modelos apresentados com diferentes tamanhos de espaço de estados e versões de descritor. A nomenclatura dos modelos é padronizada por siglas, seguida do(s) valor(es) utilizado(s) como variável para estender o modelo, bem como das letras *c* ou *f* para representar um modelo com taxas *constantes* ou *funcional* respectivamente. São mostradas colunas contendo o espaço de estados total (coluna *PSS*) e o número de atingíveis (coluna *RSS*). A seguir, tem-se três colunas com o tamanho das estruturas auxiliares alocadas: memória para o descritor de atingibilidade (coluna *RD*), memória gasta pelo diagrama de decisão multi-valorado (coluna *MDD*) e memória para a matriz HBF (coluna *HBF*).

Analisando os dados numéricos da Tabela 1, nota-se que se as estruturas podem ser alocadas em memória principal, ou seja, se não existe chance de ocorrer *swapping* (na Tabela 2 indicados por "——"), consegue-se obter resultados em tempo satisfatório se comparado ao tempo de execução do PEPS. Por exemplo, para o modelo Ad_20c, com mais de 1,5 bilhão de estados totais e por volta de 12 mil atingíveis foi necessário menos de 840 Kb de memória para armazenar a matriz em HBF, e aproximadamente 75 Kb em estruturas auxiliares (RD e MDD) tanto para o caso constante quanto para o funcional.

A técnica para solução apresentada neste trabalho ressalta que apesar do espaço total (PSS) ser gigantesco, não necessariamente seu espaço de estados atingíveis (RSS) será igualmente massivo. Com a combinação de técnicas de armazenamento e geração automatizada usando estruturas sofisticadas como MDD, observam-se ganhos na solução de modelos estocásticos estruturados de diversas realidades que apresentem um RSS reduzido. Cabe salientar que muitas vezes o modelador não é experiente ao mapear realidades para um modelo. Logo, o espaço de estados total pode vir a ser gigantesco quando comparado com os estados efetivamente atingíveis do modelo.

Um outro caso interessante é o modelo FAS com 23 servidores (FAS_23c e FAS_23f), onde todos os estados são atingíveis. Neste modelo, a matriz HBF possui 1,69 Gb estando dentro de um limite aceitável de uso da RAM, posto que o custo para armazenar as estruturas RD e MDD pouco mais de 50 Kb são necessários.

Tabela 1. Dados de alocação de memória para solução de modelos

modelo	*PSS*	*RSS*	*RD*	*MDD*	*HBF*
Ad_16c	19.131.876	1.766	22,48 Kb	27,32 Kb	105,71 Kb
Ad_16f	19.131.876	1.766	22,16 Kb	32,88 Kb	105,71 Kb
Ad_20c	1.549.681.956	12.102	32,88 Kb	35,18 Kb	839,59 Kb
Ad_20f	1.549.681.956	12.102	32,52 Kb	42,55 Kb	839,59 Kb
FAS_22c	4.194.304	4.194.304	37,07 Kb	15,31 Kb	832 Mb
FAS_22f	4.194.304	4.194.304	27,97 Kb	17,77 Kb	832 Mb
FAS_23c	8.388.608	8.388.608	40,29 Kb	16,36 Kb	1,69 Gb
FAS_23f	8.388.608	8.388.608	30,27 Kb	18,94 Kb	1,69 Gb
Kn_02c	43.046.721	4.600	19,81 Kb	14,34 Kb	511,27 Kb
Kn_02f	531.441	4.600	14,52 Kb	34,60 Kb	511,27 Kb
Kn_03c	4.294.967.296	58.400	20,13 Kb	18,40 Kb	7,70 Mb
Kn_03f	16.777.216	58.400	15,42 Kb	115,16 Kb	7,70 Mb
Kn_04c	152.587.890.625	454.475	20,44 Kb	23,01 Kb	67,66 Mb
Kn_04f	244.140.625	454.475	15,11 Kb	357,81 Kb	67,66 Mb
DP_16c	43.046.721	1.136.689	36,66 Kb	26,29 Kb	194,95 Mb
DP_16f	43.046.721	1.136.689	30,73 Kb	29,97 Kb	194,95 Mb
DP_18c	387.420.489	6.625.109	45,49 Kb	29,72 Kb	1,24 Gb
DP_18f	387.420.489	6.625.109	37,77 Kb	33,92 Kb	1,24 Gb
DP_20c	3.486.784.401	38.613.965	55,27 Kb	33,15 Kb	5,80 Gb
DP_20f	3.486.784.401	38.613.965	45,93 Kb	37,41 Kb	5,80 Gb
RS14_11c	196.608	16.278	34,45 Kb	78,39 Kb	3,71 Mb
RS14_11f	16.384	16.278	12,63 Kb	878,54 Kb	3,71 Mb
RS20_05c	6.291.456	21.700	59,55 Kb	90,58 Kb	3,40 Mb
RS20_05f	1.048.576	21.700	23,03 Kb	10,65 Mb	3,40 Mb

Diferente é a situação dos modelos DP_20c e DP_20f, onde as demandas de memória são grandes demais para que estes sejam resolvidos de maneira eficaz por ambas as ferramentas. Neste exemplo, a matriz HBF sozinha demanda 5.80 Gb, tornando a aplicação da técnica proposta impraticável para o YAMS, assim como já é para o PEPS.

O modelo Kanban (Kn) obteve resultados bastante impressionantes, pois apresenta um grande PSS em relação a um pequeno RSS (vide modelo Kn_04c). Ressalta-se que a matriz HBF ocupa em torno de 68 Mb tornando esta classe de modelo totalmente adequada para aplicação e uso de estruturas de armazenamento otimizadas aliadas a métodos vetor-matriz de solução. Note que para os modelos com PSS superior a 65 milhões de estados (tendo-se 4 Gb de RAM) a solução através da ferramenta PEPS dificilmente é viável, devido à necessidade de alocação de vetores probabilidades de tamanho PSS para realizar a multiplicação vetor-descritor.

A Tabela 2 mostra os tempos gastos na descoberta da solução estacionária das classes de modelo, utilizando o método da Potência, e executando iterações com tolerância igual a $1e$-10. A tabela apresenta os tempos coletados em cada parte do processo de solução, por exemplo, na coluna *RD* mostra-se os tempos para geração do descritor de atingibilidade e a coluna *MDD* mostra o tempo para criar a estrutura MDD em memória. Já a coluna *transições* mostra o tempo necessário para descoberta das transições do modelo (geração da matriz HBF), enquanto que a coluna *matriz* mostra o tempo gasto para salvar a matriz gerada em memória. Por fim, na coluna *YAMS*, temos o tempo para executar o método da Potência até a solução do modelo, e a coluna *PEPS* com o tempo de solução via a ferramenta PEPS para o mesmo número de iterações (coluna *# iterações*)

efetuados no YAMS, para fins comparativos. Todos os tempos estão apresentados em segundos (s), exceto os tempos relativos às execuções na ferramenta PEPS que podem estar apresentados em segundos (s), horas (h) ou dias (d) dependendo do modelo.

Tabela 2. Tempos em segundos (s) coletados para a solução dos modelos

modelo	*RD*	*MDD*	*transições*	*matriz*	*# iterações*	*YAMS*	*PEPS*
Ad_16c	≈0	≈0	0,02	0,02	75.603	2,72	2,87 d
Ad_16f	≈0	≈0	0,03	0,01	75.603	2,74	8,87 d
Ad_20c	≈0	0,01	0,24	0,11	108.918	28,92	—
Ad_20f	≈0	≈0	0,28	0,11	108.918	28,94	—
FAS_22c	≈0	≈0	146,60	75,95	457	181,87	694,80 s
FAS_22f	2,22	≈0	555,31	76,16	457	184,02	1,03 h
FAS_23c	≈0	≈0	220,35	159,08	478	411,92	1.565,40 s
FAS_23f	4,71	≈0	699,86	158,37	478	411,38	2,31 h
Kn_02c	0,49	≈0	0,06	0,04	797	0,13	2,57 h
Kn_02f	≈0	≈0	0,06	0,04	797	0,12	171,60 s
Kn_03c	22,80	≈0	0,95	0,65	1.096	4,31	—
Kn_03f	≈0	≈0	0,97	0,65	1.334	5,25	3,82 h
Kn_04c	438,52	0,01	9,83	5,87	1.448	50,35	—
Kn_04f	0,02	≈0	9,04	5,95	1.448	50,33	—
DP_16c	≈0	≈0	28,18	17,73	1.127	99,10	5,21 h
DP_16f	≈0	≈0	32,96	17,50	1.127	96,80	11,49 h
DP_18c	≈0	≈0	220,71	119,88	1.265	687,65	—
DP_18f	≈0	0,01	253,30	117,12	1.265	704,07	—
DP_20c	≈0	≈0	157,65	92,05	—	—	—
DP_20f	≈0	0,01	182,44	90,81	—	—	—
RS14_11c	≈0	0,01	0,91	0,30	125	0,19	12,37 s
RS14_11f	0,03	0,01	0,72	0,32	125	0,18	11,00 s
RS20_05c	≈0	0,01	3,53	0,29	46	0,07	262,08 s
RS20_05f	0,95	0,02	1,70	0,26	46	0,07	399,46 s

Tabela 2 mostra que, na maioria dos casos, a geração do RD e do MDD foi negligenciável. Dependendo do modelo, gastou-se um tempo considerável determinando as transições existentes, conforme mostra a Tabela 2. Os tempos obtidos são proporcionais ao tamanho do espaço de estados. Para o conjunto de modelos estudados, o tempo de solução mostrou-se bastante reduzido, inclusive para modelos com grande RSS (caso FAS_23c e FAS_23f). No entanto, quando o modelo resulta em um RSS muito grande (DP_20c e DP_20f), tanto a solução do YAMS quanto a do PEPS torna-se impraticável, sendo no YAMS devido ao tamanho do RSS armazenado na matriz HBF. Observa-se também na Tabela 2 que foi gasto um tempo considerável no cálculo das transições do modelo, podendo ser um novo alvo de otimizações em futuras versões do YAMS, onde pode-se empregar outras técnicas de acesso aos tensores e estruturas auxiliares no intuito de acelerar esta operação.

6. Conclusão

O grande desafio dos formalismos propostos para modelagem analítica é justamente proporcionar alto poder de modelagem aliado a um otimizado poder de solução. Uma das grandes vantagens do formalismo SAN por exemplo é prover primitivas de sincronização e dependências funcionais fáceis de serem utilizadas em modelos complexos, pois permitem maior liberdade na composição das transições entre estados de subsis-

temas modelados. Esta estruturação sem dúvida facilita a modelagem de realidades, ainda mais vindo acompanhada da transcrição automática para um descritor Markoviano que se utiliza de álgebra tensorial para armazenamento compacto dos eventos e transições.

A existência de tal mapeamento abriu inúmeras pesquisas no que tange a solução de descritores mantendo tal estrutura tensorial como forma de utilizar menos memória e, em alguns casos, obter bom desempenho em tempo de solução se comparado a uma alternativa tradicional via cadeias de Markov. Neste caso, modelos com grandes espaços de estados atingíveis podem justificar o uso da estruturação como forma de acelerar também a solução. Porém, o que acontece muitas vezes, é que modelos com grandes espaços de estados apresentam espaços atingíveis bem inferiores, o que dificulta o tratamento do modelo estruturado, dado que é preciso operar sobre o espaço total na busca da solução por métodos tradicionais de multiplicação vetor-descritor.

A multiplicação vetor-matriz é simplificada, porém no caso de modelos estruturados, a detecção automática do espaço atingível passa por diversas problemáticas como por exemplo, seu armazenamento e acesso otimizado através de estruturas de dados não-triviais. Ao avançar pesquisas na área, na medida em que foi possível armazenar os estados atingíveis em estruturas econômicas de memória, a solução pode também fazer uso desta vantagem e criar maneiras de gerar a cadeia de Markov implícita de forma eficiente. Pelos experimentos pode-se verificar que na maioria dos casos os custos de gerar estruturas auxiliares como o MDD e outras matrizes foi baixo em relação ao tempo gasto com o processamento tradicional do modelo estruturado na obtenção da solução por métodos iterativos. O tempo de execução reduziu drasticamente por iteração ao operar-se exatamente sobre o espaço atingível e com estruturas de dados otimizadas, comprovando a validade deste trabalho.

Dentre os trabalhos futuros tem-se a proposta de uma ferramenta com tomada de decisão para solucionar modelos estruturados, com amplo espectro de formalismos como entrada, onde seja possível extrair o descritor Markoviano, o MDD de estados atingíveis, o descritor de atingibilidade e o mapeamento para matrizes de transição entre estados atingíveis utilizados na solução. Os métodos de solução não estão restritos aos métodos iterativos, mas também métodos diretos dependendo do número de estados a tratar, ou mesmo simulação para aqueles casos onde a solução numérica não é mais possível via métodos iterativos, mantendo ou não a estruturação proposta inicialmente. Modelos com baixo número de estados atingíveis são o alvo principal deste tipo de solução econômica, pois ao mesmo tempo que modeladores ficam livres para utilizar os recursos do formalismo para diferentes níveis de abstração das realidades, tem-se a possibilidade de resolver modelos grandes em termos de espaço total de estados de forma eficiente e adequada com algoritmos de baixo custo em memória ou com menor impacto computacional.

Referências

Baldo, L., Brenner, L., Fernandes, L. G., Fernandes, P., and Sales, A. (2005). Performance Models for Master/Slave Parallel Programs. *Electronic Notes In Theoretical Computer Science (ENTCS)*, 128(4):101–121.

Benoit, A., Fernandes, P., Plateau, B., and Stewart, W. J. (2004). On the benefits of using functional transitions and Kronecker algebra. *Performance Evaluation*, 58(4):367–390.

Bertolini, C., Brenner, L., Fernandes, P., Sales, A., and Zorzo, A. F. (2004a). Structured Stochastic Modeling of Fault-Tolerant Systems. In *Proceedings of the 12th IEEE/ACM Int. Symposium on Modelling, Analysis and Simulation on Computer and Telecommunication Systems (MASCOTS'04)*, pages 139–146. IEEE Computer Society.

Bertolini, C., Farina, A. G., Fernandes, P., and Oliveira, F. M. (2004b). Test Case Generation using Stochastic Automata Networks: Quantitative Analysis. In *Proc. of the 2nd IEEE Int. Conf. on Software Engineering and Formal Methods*, pages 251–260.

Brenner, L., Fernandes, P., Plateau, B., and Sbeity, I. (2007). PEPS2007 - Stochastic Automata Networks Software Tool. In *Proceedings of the 4th IEEE International Conference on the Quantitative Evaluation of Systems (QEST 2007)*, pages 163–164.

Brenner, L., Fernandes, P., and Sales, A. (2005). The Need for and the Advantages of Generalized Tensor Algebra for Kronecker Structured Representations. *International Journal of Simulation: Systems, Science & Technology (IJSIM)*, 6(3-4):52–60.

Buchholz, P., Ciardo, G., Donatelli, S., and Kemper, P. (2000). Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS Journal on Computing*, 12(3):203–222.

Buchholz, P. and Dayar, T. (2004). Block SOR for Kronecker structured representations. *Linear Algebra and its Applications*, 386:83–109.

Chanin, R., Corrêa, M., Fernandes, P., Sales, A., Scheer, R., and Zorzo, A. F. (2006). Analytical Modeling for Operating System Schedulers on NUMA Systems. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 151(3):131–149.

Czekster, R. M. (2010). *Solução numérica de descritores Markovianos a partir de reestruturações de termos tensoriais*. PhD thesis, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Brasil.

Czekster, R. M., Fernandes, P., Sales, A., and Webber, T. (2010a). Analytical Modeling of Software Development Teams in Globally Distributed Projects. In *Proc. of the 5th IEEE Int. Conference on Global Software Engineering (ICGSE'10)*, pages 287–296.

Czekster, R. M., Fernandes, P., Sales, A., and Webber, T. (2010b). Restructuring tensor products to enhance the numerical solution of structured Markov chains. In *Proceedings of the 6th International Conference on the Numerical Solution of Markov Chains (NSMC '10)*, pages 36–39, Williamsburg, VA, USA.

Czekster, R. M., Fernandes, P., Sales, A., Webber, T., and Zorzo, A. (2011). Stochastic model for QoS assessment in multi-tier web services. In *International Workshop on Practical Applications of Stochastic Modelling (PASM '11)*, pages 93–109.

Czekster, R. M., Fernandes, P., Vincent, J. M., and Webber, T. (2007). Split: a flexible and efficient algorithm to vector-descriptor product. In *Proc. of the 2nd Int. Conf. on Performance Evaluation, Methodologies and Tools (ValueTools '07)*, Nantes, France.

Czekster, R. M., Fernandes, P., and Webber, T. (2009). GTAexpress: a Software Package to Handle Kronecker Descriptors. In *Proc. of the 6th IEEE Int. Conference on Quantitative Evaluation of Systems (QEST 2009)*, pages 281–282, Budapest, Hungary.

Dotti, F. L., Fernandes, P., and Nunes, C. M. (2011). Structured Markovian Models for Discrete Spatial Mobile Node Distribution. *Journal of the Brazilian Computer Society (JBCS)*, 17(1):31–52.

Dotti, F. L., Fernandes, P., Sales, A., and Santos, O. M. (2005). Modular Analytical Performance Models for Ad Hoc Wireless Networks. In *Proc. of the 3rd IEEE Int. Symp. on Modeling and Optim. in Mobile, Ad Hoc, and Wireless Nets*, pages 164–173.

Fernandes, P., Plateau, B., and Stewart, W. J. (1998). Efficient descriptor-vector multiplication in Stochastic Automata Networks. *Journal of the ACM*, 45(3):381–414.

Fernandes, P., Sales, A., Santos, A. R., and Webber, T. (2011). Performance Evaluation of Software Development Teams: A Practical Case Study. In *International Workshop on Practical Applications of Stochastic Modelling (PASM '11)*, pages 5–21.

Fernandes, P., Vincent, J. M., and Webber, T. (2008). *Perfect Simulation of Stochastic Automata Networks*, volume 5055 of *LNCS*, pages 249–263. Springer-Verlag.

Miner, A. S. (2002). Efficient state space generation of GSPNs using decision diagrams. In *International Conference on Dependable Systems and Networks (DSN 2002)*, pages 637–646, Washington, DC, USA.

Miner, A. S. and Ciardo, G. (1999a). A data structure for the efficient Kronecker solution of GSPNs. In *Proceedings of the 8th International Workshop on Petri Nets and Performance Models*, pages 22–31, Zaragoza, Spain.

Miner, A. S. and Ciardo, G. (1999b). Efficient Reachability Set Generation and Storage Using Decision Diagrams. In *Proc. of the 20th International Conference on Applications and Theory of Petri Nets*, volume 1639 of *LNCS*, pages 6–25. Springer-Verlag.

Mokdad, L., Ben-Othman, J., and Gueroui, A. (2001). Quality of Service of a Rerouting Algorithm Using Stochastic Automata Networks. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, pages 338–343, Hammamet, Tunisia.

Plateau, B. (1985). On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proc. of the 1985 ACM SIGMETRICS Conf. on Measurements and Modeling of Computer Systems*, pages 147–154, Austin, Texas, USA.

Saad, Y. (1995). *Iterative Methods for Sparse Linear Systems*. Boston: PWS Publishing Company.

Sales, A. (2009). *Réseaux d'Automates Stochastiques: Génération de l'espace d'états atteignables et Multiplication vecteur-descripteur pour une sémantique en temps discret*. PhD thesis, Institut National Polytechnique de Grenoble, France. http://tel.archives-ouvertes.fr/tel-00436020/en/.

Sales, A. and Plateau, B. (2009). Reachable State Space Generation for Structured Models which use Functional Transitions. In *Proc. of the 6th IEEE Int. Conference on the Quantitative Evaluation of Systems (QEST 2009)*, pages 269–278, Budapest, Hungary.

Stewart, W. J. (1994). *Introduction to the numerical solution of Markov chains*. Princeton University Press.

Webber, T. (2009). *Reducing the Impact of State Space Explosion in Stochastic Automata Networks*. PhD thesis, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Brasil.