FÁBIO TOZETO RAMOS

INFERÊNCIA PROBABILÍSTICA EM SISTEMAS COM RESTRIÇÕES DE TEMPO E MEMÓRIA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Mestre em Engenharia.

São Paulo

2003

FÁBIO TOZETO RAMOS

INFERÊNCIA PROBABILÍSTICA EM SISTEMAS COM RESTRIÇÕES DE TEMPO E MEMÓRIA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Mestre em Engenharia.

Área de Concentração:
Inteligência Artificial

Orientador:
Prof. Dr. Fabio Gagliardi Cozman

Universidade de São Paulo
Escola Politécnica

São Paulo

2003

À memória de meu avô,
Dorival Soares Ramos.
À memória de minha avó,
Heralda Silva.

AGRADECIMENTOS

Ao amigo e orientador Prof. Dr. Fabio Gagliardi Cozman por todo incentivo, orientação e valiosas discussões que enriqueceram este trabalho.

Aos professores Dr. Newton Maruyama e Dr. Gilberto Francisco Martha de Souza por suas sugestões no Exame de Qualificação.

À Marsha Duro da HP Labs, Estados Unidos, Edson Nery da HP Brasil e ao Instituto de Pesquisas Eldorado pelo indispensável apoio ao projeto.

Aos meus pais, Maria Regina Tozeto Ramos e Dorel Soares Ramos por toda dedicação e afeto dispensados à minha formação.

À minha avó e professora de português Elza da Silva Ramos por suas correções gramaticais.

Aos amigos do Laboratório de Tomada de Decisão do PMR-EPUSP pelo companheirismo e auxílio na publicação de artigos.

À minha noiva Karin por me acompanhar nos momentos mais difíceis, sempre com uma palavra de incentivo.

Errata

Página	Linha	Onde se lê	Deve-se ler
10	13	teorema	proposição
10	15	Teorema	Proposição
10	23	deste	desta
10	23	teorema	proposição
10	25	O teorema	A proposição
30	15	como mostra o Teorema 1	como mostra a Proposição 1
31	10	implementa o Teorema 1	implementa a Proposição 1
31	16	Teorema 2	Proposição 2
56	23	com teoremas simples	com proposições simples
57	33	o Teorema 2	a Proposição 2

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

RESUMO

Um dos maiores desafios que os sistemas de Inteligência Artificial têm de enfrentar atualmente é como possibilitar que grandes e complexos modelos de representação do conhecimento possam ser embarcados em dispositivos computacionais com recursos limitados. No presente trabalho este problema é tratado no contexto de modelos probabilísticos em Inteligência Artificial, mais precisamente, redes Bayesianas. Um novo algoritmo, capaz de produzir inferências sob várias restrições de tempo e espaço, é proposto e testado. Caracterizando-se por sua adaptabilidade e pela utilização de métodos de condicionamento, o algoritmo recebe o nome de condicionamento adaptativo. As diversas técnicas empregadas, assim como a possibilidade de produzir inferências associando vários algoritmos diferentes sob sua supervisão, tornam este algoritmo flexível e apto a ser utilizado em sistemas embarcados ou equipamentos com recursos limitados. Resultados experimentais com redes de grande porte são apresentados em gráficos de três dimensões (Qualidade da resposta×Memória×Tempo), indicando seu desempenho com redes reais.

ABSTRACT

One of the biggest challenges that artificial intelligence systems have to cope with today is how to enable large and complex knowledge representation to operate in embedded systems with bounded resources, such as real-time devices. This work is focused on probabilistic models in artificial intelligence, more precisely, Bayesian networks. A new algorithm, able to yield inferences under both time and memory constraints, is presented and tested. Displaying flexibility and adaptability as its fundamental features, the algorithm named Adaptive Conditioning uses conditioning methods mixed with search-based and cache methods to compute probabilistic inferences. Besides, it can also combine different algorithms under its supervision for the same inference. These features allow adaptive conditioning to be used in embedded systems or in devices with bounded resources in general. Experimental results with demanding networks are presented in three-dimensional graphs (Answer quality×Memory×Time), indicating the algorithm's performance in real applications.

1 INTRODUÇÃO

A utilização de redes Bayesianas como modelos gráficos para a representação de incerteza tem crescido rapidamente nos últimos vinte anos. Através dessas redes é possível representar de maneira simples e natural as diversas relações de causa-efeito entre as variáveis de um problema, além da incerteza associada a cada variável através de distribuições de probabilidade. Como modelos gráficos, podem ser facilmente criadas e visualizadas, além de permitir que vários tipos de algoritmos para inferência façam uso de suas propriedades para simplificar operações. Assim, as redes Bayesianas são também consideradas uma poderosa estrutura para inferência probabilística e a mais utilizada técnica para representação do conhecimento e raciocínio diante de problemas que envolvam incerteza em inteligência artificial.

Dentre suas numerosas aplicações estão os sistemas especialistas para auxílio à decisão médica (BEINLICH, 1989; ANDREASSEN, 1991), suporte técnico para solucionadores de problemas *(troubleshooters)* (HORVITZ, 1998), sistemas de decisão para interpretar dados de telemetria em tempo real (HORVITZ; BARRY, 1995), genética (DWARKADAS, 1994), reconhecimento da fala (ZWEIG; RUSSELL, 1998), *tracking* (KWOK; FOX; MEILA, 2002; MAJUMDER; SCHEDING; DURRANT-WHYTE, 2001; MURPHY, 1998), compressão de dados (DAVIES, 2002) e sistemas de diagnóstico em plantas industriais (RAMOS; MIKAMI; COZMAN, 2000).

A complexidade computacional para uma inferência probabilística em uma rede Bayesiana genérica é NP-difícil (COOPER, 1990b), ou seja, esta classe de problemas não pode ser resolvida deterministicamente em tempo polinomial. Além disso, a complexidade computacional para inferências aproximadas é também NP-difícil (DAGUM; LUBY, 1993), o que torna o estudo de algoritmos eficientes para inferência probabilística essencial para a aplicação desses modelos nas complexas situações reais.

Nos últimos anos tem crescido também a utilização de pequenos sistemas computacionais dedicados para aplicações específicas e que têm, como uma das funções principais, a tomada de decisão, muitas vezes, sob severas restrições de tempo. Exemplos desses sistemas podem ser encontrados em aviônicos (sistemas de controle de vôo de aeronaves), sistemas de injeção eletrônica de automóveis, robôs para aplicações diver-

sas, conversores de sinal digital-analógico para televisores analógicos operarem com o sistema digital (*set-top boxes*), entre outros. Estes sistemas, por estarem dentro de outros sistemas mais complexos, são denominados sistemas embarcados e podem ter, além de restrições de tempo, também limitações de espaço[1], fazendo com que apenas algoritmos com baixo consumo de memória devam ser utilizados.

Considerando-se todos estes aspectos, percebe-se que um dos maiores desafios para a aplicação de redes Bayesianas em situações reais é: Como possibilitar que tais modelos possam ser utilizados em sistemas embarcados e/ou sistemas em tempo real onde recursos como tempo e memória são considerados escassos? Além disso, mesmo quando apenas um desses recursos é limitado, como tornar o processo de inferência o mais eficiente possível?

Neste trabalho tais questões serão abordadas detalhadamente e um novo e *flexível* algoritmo será proposto como uma possível solução. Por *flexível* entende-se um algoritmo capaz de adaptar-se a diferentes restrições de tempo e espaço, trabalhando não apenas com uma solução de compromisso tempo×espaço ou qualidade de resultado[2]×tempo, mas com todas juntas em uma solução de compromisso entre qualidade×tempo×espaço. Para lidar com estes problemas, serão apresentados métodos de condicionamento combinados com algoritmos de eliminação de variáveis, a fim de produzir um algoritmo de inferência que possa trabalhar em quaisquer restrições de tempo e espaço. Esta abordagem para o problema de inferência probabilística não é similar a nenhuma outra encontrada na literatura. Assim, este trabalho apresenta métodos e algoritmos que configuram contribuição efetiva ao estado da arte do assunto.

A idéia básica do algoritmo proposto, denominado condicionamento adaptativo, é a de utilizar técnicas de condicionamento, ou seja, assumir valores para determinadas variáveis, a fim de dividir o problema inicial em problemas menores, os quais podem ser resolvidos com os recursos disponíveis. Em outras palavras, condicionam-se certas variáveis da rede Bayesiana inicial para dividi-la em redes menores, as quais podem ser resolvidas dentro das restrições de espaço, com algoritmos comuns de inferência como, por exemplo, o método de eliminação de variáveis (DECHTER, 1996a; COZMAN, 2000). Os resultados de cada sub-rede podem então ser combinados para cada instanciação[3] das

[1] "Espaço" será usado neste texto como sinônimo de memória.

[2] "Qualidade do resultado" ou "qualidade da resposta" serão usados neste texto como o valor absoluto da diferença entre a resposta fornecida e o valor exato da probabilidade marginal de uma dada variável.

[3] O termo *instanciação* é usado para identificar a operação em que as variáveis condicionadas assumem

variáveis condicionadas, até que todas as instanciações sejam percorridas, ou até que o limite de tempo seja atingido (retornando, neste último caso, uma resposta aproximada).

O condicionamento adaptativo trabalha em duas fases distintas: fase de planejamento e fase de execução. Na fase de planejamento, o algoritmo decompõe a rede inicial em sub-redes de forma a impedir que nas próximas operações de inferência o limite de memória seja ultrapassado. Depois disso, com as variáveis a serem condicionadas já determinadas, o sistema reorganiza a ordem em que estas variáveis serão instanciadas, a fim de obter maiores massas de probabilidade nas etapas iniciais do processo, aproximando-se mais rapidamente da resposta exata. Adicionalmente, após realizada uma inferência em cada sub-rede, pode-se avaliar a máxima quantidade de memória de fato utilizada. Caso este valor seja inferior ao máximo permitido, pode-se alocar *caches* para armazenar valores de inferências já realizadas, de forma a evitar que essas sejam repetidas. Resumidamente, a fase de planejamento caracteriza-se pelas seguintes etapas:

1. Decomposição da rede;
2. Organização das variáveis condicionadas e ordem de instanciações;
3. Alocação de *caches*.

A próxima etapa, onde as inferências serão executadas e o resultado será obtido, recebe o nome de fase de execução. Nesta fase, serão feitas inferências para cada sub-rede criada na fase de planejamento, com cada combinação de valores assumidos pelas variáveis condicionadas. Os resultados são então multiplicados e somados para o cálculo das probabilidades marginais. Quanto mais inferências para o conjunto de instanciações forem feitas e somadas, mais a resposta se aproximará do valor exato. Portanto, esta fase caracteriza-se por ser um processo "qualquer tempo" (*anytime*), ou seja, a resposta melhorará com o passar do tempo, e em qualquer momento que a computação seja interrompida, sempre retornará um resultado.

A fase de execução continuará até que o valor exato seja obtido ou até que o limite de tempo seja atingido, fornecendo, neste caso, um valor aproximado. Ademais, o algoritmo também fornece intervalos de variação para os valores das probabilidades, quando retorna um resultado aproximado. Estes intervalos indicam valores máximos e mínimos onde os valores das probabilidades marginais exatas podem estar.

valores determinados. Essa operação é explanada em detalhes no Capítulo 3.

Outra possibilidade é a utilização de algoritmos diversos para cada sub-rede uma vez que, quando a rede inicial é dividida, todas as suas sub-redes podem ser processadas independentemente umas das outras. Assim, o condicionamento adaptativo combina diversas técnicas para inferência probabilística, a fim de obter o melhor desempenho possível dentro das limitações exigidas por sistemas embarcados e/ou sistemas em tempo real.

Este trabalho está organizado da seguinte forma: no Capítulo 2 estão apresentados algumas definições e conceitos essenciais para compreensão do texto, bem como trabalhos correlatos. No Capítulo 3 o algoritmo denominado condicionamento adaptativo é apresentado e seus resultados experimentais com redes reais estão no Capítulo 4. As análises desses resultados estão no Capítulo 5 e as conclusões no Capítulo 6.

2 DEFINIÇÕES E TRABALHOS CORRELATOS

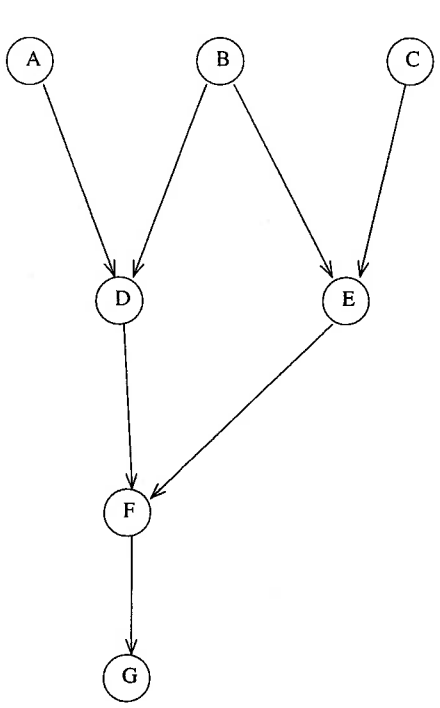
Este capítulo apresenta as principais definições da teoria dos grafos e redes Bayesianas. Posteriormente, é apresentada uma estrutura secundária utilizada por algoritmos exatos de inferência em redes Bayesianas, bem como técnicas para sua construção. Também são descritos os principais algoritmos para inferências exatas e aproximadas. O capítulo termina com algumas definições e conceitos relacionados à inteligência artificial com restrições de tempo e espaço.

2.1 Teoria dos Grafos e Redes Bayesianas

A maior vantagem em utilizar grafos na representação do conhecimento é a facilidade na visualização e construção de modelos. Nesta seção, serão apresentados conceitos básicos sobre grafos, além das definições de redes Bayesianas. As notações adotadas seguirão as encontradas em (COWELL, 1999).

Um *grafo* $\mathcal{G}$ é definido como um par $\mathcal{G} = (V, E)$ onde V é um conjunto finito de *vértices,* também conhecidos como *nós,* e E é um subconjunto do conjunto $V \times V$ de pares ordenados de vértices, denominados *arcos* de $\mathcal{G}$. Se $(\alpha, \beta) \in E$ mas $(\beta, \alpha) \notin E$, o arco (α, β) é chamado *direcionado* e α é um pai de β. Se todos os arcos de $\mathcal{G}$ forem direcionados, o grafo $\mathcal{G}$ é um *grafo direcionado*. Caso contrário o grafo é dito *não-direcionado*. Um *caminho* de comprimento n de α até β é uma seqüência $\alpha = \alpha_0, \ldots, \alpha_n = \beta$ de vértices distintos tal que $(\alpha_{i-1}, \alpha_i) \in E$ para todo $i = 1, \ldots, n$. Se um caminho tem o mesmo vértice como ponto inicial e final, ele é chamado *ciclo*. Um grafo é chamado *acíclico*, se não possui nenhum ciclo. Um *grafo moral* é um grafo não-direcionado gerado a partir de um grafo direcionado com a conexão dos pais de cada nó v_i e removendo as direções dos arcos. Um grafo moral é *triangularizado* se, e somente se todo ciclo de comprimento quatro ou maior que quatro, o grafo possuir um arco que conecta dois nós não-adjacentes.

Este trabalho está focado em raciocínio probabilístico conduzido por redes Bayesianas. Uma rede Bayesiana representa uma distribuição conjunta sobre variáveis $\{X_1, \ldots, X_n\}$



Densidades de probabilidade para a rede:
Pr(A), Pr(B), Pr(C), Pr(D|A,B),
Pr(E|B,C), Pr(F|D,E), Pr(G|F)

Figura 1: Exemplo de uma rede Bayesiana.

dispostas em um *grafo direcionado acíclico* (ou GDA) com n nós:

Definição 1 *Uma* rede Bayesiana *consiste de:*

- *Um conjunto de nós e um conjunto de arcos entre nós formando um grafo direcionado acíclico;*
- *Cada nó é associado a uma variável a qual apresenta uma distribuição de probabilidade dados seus pais e um conjunto finito e mutuamente exclusivo de estados.*

Dada esta definição, as palavras *nós* e *variáveis* serão usadas sem distinção a partir deste ponto. Cada variável X_i possui uma distribuição condicional $Pr(X_i|pa(X_i))$, onde $pa(X_i)$ é o conjunto de pais de X_i no grafo. A distribuição conjunta representada pela rede tem a forma $\prod_i Pr(X_i|pa(X_i))$. Conjuntos de variáveis serão representados por letras em negrito, assim, o conjunto de todas as variáveis X_i, com $i = 1, ..., n$, é representado por $\mathbf{X}$. A Fig.1 mostra um exemplo de rede e indica as densidades de probabilidade.

A partir da topologia de uma rede Bayesiana podem-se derivar duas expressões semânticas equivalentes que mostram as relações de independência entre os nós da rede:

1. Um nó é condicionalmente independente de seus não-descendentes não-pais, dados os seus pais;

2. Um nó é condicionalmente independente de todos os outros nós da rede, dados os seus pais, filhos, e os outros pais de seus filhos.

Estes fatos simplificam as operações de inferência probabilística uma vez que pode simplificar-se o modelo, descartando variáveis que não afetarão o resultado. Uma abordagem matemática para as relações de independência pode ser encontrada em (PEARL, 1988), com os métodos de d-separação, que podem ser implementados em tempo polinomial. Esses métodos serão utilizados no algoritmo proposto para descartar variáveis desnecessárias.

Definição 2 *Uma* inferência *é obtida quando se computa* $Pr(\mathbf{X}_Q|\mathbf{X}_E)$ *para as variáveis de interesse* $\mathbf{X}_Q$*, dadas as variáveis de evidência* $\mathbf{X}_E$*, com* $\mathbf{X}_Q \cap \mathbf{X}_E = \emptyset$*, de acordo com a equação abaixo:*

$$Pr(\mathbf{X}_Q|\mathbf{X}_E) = \frac{\sum_{X_i \notin \{\mathbf{X}_Q, \mathbf{X}_E\}} \prod_i Pr(X_i|pa(X_i))}{\sum_{X_i \notin \{\mathbf{X}_E\}} \prod_i Pr(X_i|pa(X_i))}. \tag{2.1}$$

A partir destas definições podem-se classificar grafos direcionados acíclicos em dois tipos principais definidos a seguir:

Definição 3 *Os GDAs onde existe apenas um caminho entre duas de suas variáveis são chamados singularmente conectados ou* polytrees.

Definição 4 *Os GDAs que possuem dois ou mais caminhos entre duas de suas variáveis são chamados multi-conectados.*

2.2 Árvores de Junção (*Junction Trees*)

Muitos algoritmos de inferência probabilística utilizam uma estrutura auxiliar para representar o conhecimento em uma etapa intermediária no processo de inferência. Esta estrutura, conhecida como *árvore de junção*, é adequada para trabalhar com algoritmos

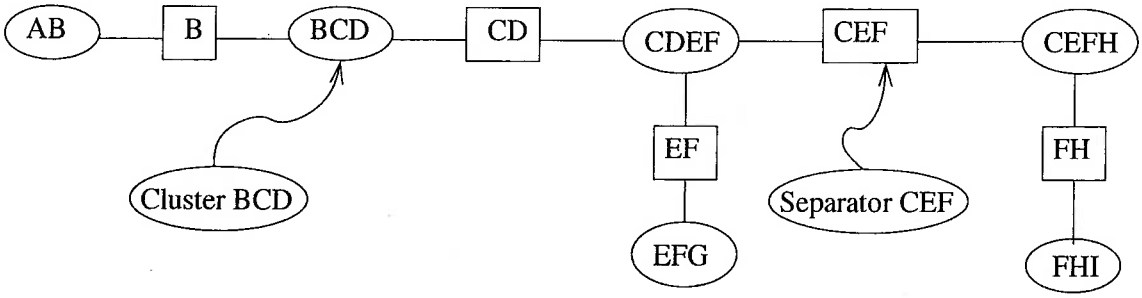


Figura 2: Um exemplo de árvore de junção.

de *clustering* ou eliminação. Neste trabalho, as árvores de junção serão utilizadas em dois momentos diferentes durante o processo de inferência. Inicialmente, essas árvores serão utilizadas no método heurístico de decomposição da rede inicial em redes menores. Depois, serão novamente utilizadas pelo algoritmo de eliminação de variáveis para gerar a ordem de eliminação em cada sub-rede.

As árvores de junção constituem uma representação gráfica alternativa para grafos em que, após passarem por um processo de *triangularização*, agrupam-se variáveis conectadas em sub-conjuntos. Esses sub-conjuntos, por sua vez, estão conectados a seus vizinhos por sub-conjuntos formados pela intersecção das variáveis do sub-conjunto com seus vizinhos respectivos. As árvores de junção obedecem uma propriedade fundamental que é descrita abaixo:

Definição 5 *(COWELL, 1999) Considere $\mathcal{C}$ como sendo uma coleção de sub-conjuntos de um conjunto finito V de variáveis de um GDA, e $\mathcal{T}$ uma árvore com $\mathcal{C}$ sendo seu conjunto de nós. Então, $\mathcal{T}$ é uma árvore de junção, se satisfizer a seguinte propriedade:*

- *Qualquer intersecção $C_1 \cap C_2$ de um par de sub-conjuntos C_1, C_2 em $\mathcal{C}$ tem de estar contida em todo nó pertencente ao caminho único entre C_1 e C_2. Equivalentemente, cada nó v em V terá sua família[1] incluída em, no mínimo, um sub-conjunto.*

Os nós C_i da árvore de junção são chamados *clusters* e as variáveis associadas com a intersecção $C_1 \cap C_2$ são chamados *separators*. A Fig.2 mostra uma árvore de junção e indica um *cluster* e um *separator*.

A construção de árvores de junção para um GDA geralmente envolve as seguintes etapas (DARWICHE; PROVAN, 1996):

[1] A família de um nó α é um conjunto formado pela união de α com seus pais.

1. Construção de um grafo moral a partir do GDA;
2. Adição de arcos ao grafo moral, a fim de deixá-lo triangularizado;
3. A partir do grafo triangularizado, identificar os *cliques*[2] e associá-los a *clusters*;
4. Construção da árvore de junção conectando-se os *clusters* de forma a obter, ao final do processo, uma árvore não-direcionada que satisfaça a propriedade fundamental das árvores de junção.

Informações adicionais sobre árvores de junção e sua construção podem ser encontradas em (JENSEN, 1988; JENSEN; JENSEN, 1994; KJAERULFF, 1990, 1992).

2.3 Inferência Exata para Redes Bayesianas

Basicamente os algoritmos para inferência em redes Bayesianas podem ser divididos em três classes: algoritmos de mensagens para *polytrees* (KIM; PEARL, 1983), algoritmos baseados em condicionamento (VEGAS, 1992; SHACHTER; ANDERSEN; SZOLOVITS, 1994; RAMOS; COZMAN; IDE, 2002; DARWICHE, 1995, 2001) e os algoritmos baseados em *clustering* e eliminação (COZMAN, 2000; DECHTER, 1996a; JENSEN; LAURITZEN; OLESEN, 1990).

2.3.1 Algoritmos de *polytree*

Os algoritmos de *polytree* utilizam a técnica de propagação de mensagens para computar probabilidades exatas em redes *polytrees*. Sua complexidade é polinomial no número de nós, porém eles apenas funcionam para redes singularmente conectadas ou *polytrees*. Este método foi proposto por Pearl (1988).

2.3.2 Método de Condicionamento

No presente trabalho, o método de condicionamento é utilizado para dividir uma grande rede em sub-redes menores e mais simples. Tal método foi apresentado inicialmente por Pearl (1988) e tem como idéia fundamental simplificar um modelo complexo,

[2] *Cliques* em um grafo triangularizado são conjuntos de variáveis onde cada variável do conjunto é conectada as outras.

assumindo valores para algumas de suas variáveis (condicionamento). A equação abaixo mostra o princípio de funcionamento:

$$Pr(X) = \sum_{c} Pr(X, C = c) \tag{2.2}$$

Nesta equação a probabilidade $Pr(X)$ pode ser representada pela soma de várias probabilidades $Pr(X, C = c)$ onde a variável C assume todos os seus valores. Para o caso de C possuir três estados c_1, c_2 e c_3, a probabilidade $Pr(X)$ é igual a $Pr(X, C = c_1) + Pr(X, C = c_2) + Pr(X, C = c_3)$. Assim, o problema inicial ficou dividido em três problemas mais simples.

Além de C, outras variáveis poderiam estar condicionadas, o que reduziria o problema inicial em problemas ainda menores. Essas variáveis condicionadas são denominadas variáveis de corte ou *cutset*.

No condicionamento adaptativo, este método é generalizado para considerar as várias sub-redes criadas quando uma rede é dividida. O teorema abaixo estende a Expressão (2.1) e indica as operações básicas usadas em condicionamento adaptativo:

Teorema 1 (Condicionamento Adaptativo) [3] *Considere* $\mathbf{C}$ *(variáveis de corte ou* cutset*) como um conjunto de variáveis da rede* $\mathcal{N}$ *que, quando condicionadas, tornam a rede desconectada em sub-redes* $\mathcal{N}_i$ *com* $i = 1, 2, ..., n$*, e considere* $\mathbf{Q}$ *um conjunto de variáveis em* $\mathcal{N}$ *onde estamos interessados em computar as probabilidades marginais. Então,*

$$Pr^{\mathcal{N}}(\mathbf{Q}) = \sum_{\mathbf{C}} \prod_{i} Pr^{\mathcal{N}_i}(\mathbf{Q}_i, \mathbf{C}_i)\,. \tag{2.3}$$

onde $Pr^{\mathcal{N}}(\mathbf{Q})$ *é a probabilidade de* $\mathbf{Q}$ *na rede* $\mathcal{N}$*,* $\mathbf{Q}_i$ *é um sub-conjunto de* $\mathbf{Q}$ *que está em* $\mathcal{N}_i$ *e* $\mathbf{C}_i$ *é um sub-conjunto de* $\mathbf{C}$ *que está em* $\mathcal{N}_i$*.*

A prova deste teorema vem do fato que $Pr^{\mathcal{N}}(\mathbf{Q}) = \sum_{\mathbf{C}} Pr^{\mathcal{N}}(\mathbf{Q}, \mathbf{C})$, e de que as variáveis de corte $\mathbf{C}$ produzem uma decomposição da forma $\prod_i Pr^{\mathcal{N}_i}(\mathbf{Q}_i, \mathbf{C}_i)$. O teorema é uma imediata generalização do teorema básico de *recursive conditioning* (DARWICHE, 2001); a diferença é que *recursive conditioning* sempre divide a rede em duas sub-redes por vez e, conseqüentemente, sempre obtém dois termos no somatório, enquanto o condicionamento adaptativo pode dividi-la em várias partes de uma só vez.

[3] São utilizadas as mesmas notações usadas por Darwiche (2001).

Os métodos de condicionamento usam a técnica de dividir um problema maior em problemas menores, fazendo-se suposições a respeito dos parâmetros de entrada. Então, os sub-problemas podem ser resolvidos para cada possível suposição, retornando uma resposta exata. Usualmente, esta estratégia é utilizada em conjunto com outra a qual seria intratável sem condicionamento. Assim, alguns algoritmos usam o algoritmo de *poly-tree* junto com condicionamento como por exemplo *local conditioning* (VEGAS, 1992), *global conditioning* (SHACHTER; ANDERSEN; SZOLOVITS, 1994) e *dynamic conditioning* (DARWICHE, 1995). Outros usam condicionamento com métodos de *clustering* como os apresentados por Dechter (1996b). Outra possibilidade é utilizar condicionamento para decompor uma rede em nós separados e computar as probabilidades recursivamente como em *recursive conditioning* (DARWICHE, 2001). Por serem de grande importância para o presente trabalho e também por apresentarem variações fornecendo resultados aproximados, os métodos de condicionamento serão discutidos mais adiante na Seção 2.5.

2.3.3 *Clustering* e Eliminação

A terceira classe de algoritmos para inferência exata é baseada em métodos de *clustering* e eliminação. Tais métodos utilizam basicamente a mesma técnica para transformar a rede inicial em uma segunda estrutura (uma árvore de junção) e realizar as computações necessárias nesta segunda estrutura. Em *clustering*, o princípio de inferências utiliza o método de passagem de mensagens para fazer uma árvore de junção consistente [4] e então calcular as probabilidades posteriores (LAURITZEN; SPIEGEHALTER, 1988). A complexidade é exponencial no tamanho do maior *cluster* da árvore de junção.

Para os algoritmos de eliminação, a idéia principal é eliminar variáveis a partir de uma ordem de eliminação, somando-as uma a uma. Neste trabalho será utilizado o algoritmo de eliminação de variáveis, inicialmente desenvolvido no algoritmo de *peeling* em genética (CANNINGS; THOMPSON; SKOLNICK, 1978). Este algoritmo será discutido em detalhe, a seguir, por ser fundamental para o entendimento deste trabalho.

O algoritmo de eliminação de variáveis é uma maneira simples de realizar inferências em redes Bayesianas. Sua simplicidade vem do fato de que este pode ser entendido como uma simples manipulação algébrica. Como foi visto na seção anterior, o obje-

[4] Uma árvore de junção é consistente quando todos os *cluster-separators* satisfazem a condição de consistência: para cada *cluster* $\mathbf{X}$ e vizinho *separator* $\mathbf{S}$, ocorre $\sum_{\mathbf{X}\setminus\mathbf{S}} Pr(\mathbf{X}) = Pr(\mathbf{S})$.

tivo de uma inferência em redes Bayesianas é calcular a Equação (2.1). Entretanto, por simplicidade pode-se calcular apenas:

$$Pr\left(\mathbf{X}_Q,\, \mathbf{X}_E\right) = \sum_{\mathbf{X}_R \setminus \{\mathbf{X}_Q, \mathbf{X}_E\}} \left(\prod_{X_i \in \mathbf{X}_R} Pr\left(X_i | pa\left(X_i\right)\right) \right), \tag{2.4}$$

onde $\mathbf{X}_R$ denota as variáveis necessárias para a computação das variáveis de interesse. Estas variáveis podem ser obtidas utilizando as regras de d-separação (PEARL, 1988). Uma vez que todas as variáveis necessárias foram obtidas, pode-se retirar destas as variáveis que estão em $\mathbf{X}_Q$ e em $\mathbf{X}_E$, a fim de se criar uma ordem de eliminação. Por exemplo, considere as variáveis necessárias sem os conjuntos $\mathbf{X}_Q$ e $\mathbf{X}_E$. Pode-se criar uma ordem de eliminação para as variáveis restantes que vão de 1 até N, $\{X_1, X_2, \ldots, X_N\}$. Então, as variáveis podem ser eliminadas multiplicando e somando as densidades de probabilidade onde cada variável a ser eliminada aparece. Para eliminar inicialmente a variável X_1, deve-se realizar o somatório onde esta aparece como indicado na expressão abaixo:

$$\sum_{X_1} \left(\prod_{X_j \in \{X_1, ch(X_1)\}} Pr\left(X_j | pa\left(X_j\right)\right) \right) \tag{2.5}$$

O processo continua até a eliminação de X_N e a obtenção da resposta. A questão principal deste método é que qualquer ordem arbitrária de eliminação pode ser utilizada; porém, diferentes ordens podem levar a diferentes esforços computacionais (a complexidade do método de eliminação de variáveis pode ser calculada pelo número de somas e multiplicações realizadas). O problema de encontrar uma solução ótima é conhecido por ser NP-difícil, assim heurísticas devem ser utilizadas. Outras implementações deste algoritmo podem ser encontradas em (DECHTER, 1996a; COZMAN, 2000).

2.4 Inferência Aproximada para Redes Bayesianas

Os métodos para inferência Bayesiana aproximada podem ser divididos em quatro classes: amostragem estocástica, simplificação do modelo, métodos baseados em busca e propagação de ciclos. Os métodos de amostragem estocástica ou algoritmos de Monte Carlo formam a maior classe e podem ser divididos em duas sub-classes: algoritmos baseados em *importance sampling* e métodos Monte Carlo Cadeias de Markov (MCMC).

Por serem algoritmos *anytime*, os métodos baseados em *amostragem estocástica* são largamente utilizados para inferência probabilística em grandes redes, quando os métodos exatos não são adequados, dada a complexidade do problema e os requerimentos de espaço. A idéia principal de *importance sampling* é a de simular a rede Bayesiana, e armazenar as freqüências em que cada variável aparece nos eventos relevantes. As amostras são geradas *a priori* e não dependem umas das outras (FISHMAN, 1995). Por outro lado, os algoritmos baseados em MCMC utilizam amostras que são dependentes, tais como *Gibbs sampling* (GEMAN; GEMAN, 1984). Estes algoritmos são úteis, quando o tempo e/ou a complexidade do modelo são importantes restrições e produzem bons resultados, quando as densidades de probabilidade não estão muito próximas de zero ou de um.

No *método de simplificação* de modelos, diversas técnicas são utilizadas para reduzir a rede original em uma rede mais simples. As inferências são então feitas nesta rede mais simples que, se mais tempo estiver disponível, podem ter sua complexidade aumentada até atingir a complexidade da rede original. Exemplos de métodos de simplificação de modelos incluem a remoção de dependências fracas e remoção de arcos (KJAERULFF, 1994), *mini-buckets* o qual representa simplificações das operações dentro de *buckets*[5] (DECHTER, 1997), *bounded conditioning*, o qual produz respostas utilizando apenas algumas das possíveis instanciações, a fim de reduzir o tempo de inferência (discutido na Seção 2.5), e o algoritmo de abstração de estado de espaços que reduz a cardinalidade das tabelas de probabilidade para simplificar o modelo probabilístico (WELLMAN; LIU, 1994).

Os *métodos baseados em busca* fazem uso da idéia de que uma maior densidade de probabilidade está concentrada em uma pequena fração do espaço da probabilidade conjunta. Estes métodos utilizam algoritmos de busca que dão prioridade para a computação das maiores massas de probabilidade, a fim de obterem melhores aproximações, dentro do tempo determinado. Exemplos destes métodos podem ser encontrados em *term computation* (D'AMBROSIO, 1993) e em métodos de busca de Henrion (1991).

A última classe de algoritmos de aproximação é conhecida como *Propagação de Ciclos* (*Loopy Belief Propagation*). Estes algoritmos fazem uso do algoritmo de Pearl (1988) para *polytrees* em uma rede com ciclos (Seção 2.3.1). Alguns resultados experimentais deste algoritmo mostram que este pode produzir boas aproximações para certas

[5] *Buckets* são sub-conjuntos de variáveis semelhantes aos *clusters* das árvores de junção.

redes em um relativo curto espaço de tempo, se comparado com os métodos estocásticos. Por outro lado, para outras redes, o método pode não convergir ou convergir para um resultado incoerente (MURPHY; WEISS; JORDAN, 1999; WEISS; FREEMAN, 1999); as razões deste problema ainda estão sendo investigadas na literatura.

2.5 Algoritmos de Condicionamento

Diversos algoritmos de condicionamento têm sido aplicados a inferência em redes Bayesianas. Em *cutset conditioning*, também conhecido como método de *loop-cutset* (PEARL, 1988; SHACHTER; ANDERSEN; SZOLOVITS, 1994), é usada a técnica de condicionamento para transformar uma rede multi-conectada em uma rede singularmente conectada. Isso é feito condicionando-se as variáveis dentro de *loops*[6]. Assim, a rede singularmente conectada pode ser processada em espaço linear usando o algoritmo de *polytree* (KIM; PEARL, 1983). Esta característica é bastante interessante, se comparada com o espaço de complexidade exponencial dos algoritmos baseados em *clustering* e eliminação. Entretanto, a grande desvantagem dos métodos de *loop-cutset* está em sua complexidade no tempo, a qual é exponencial no número de variáveis condicionadas. Alguns trabalhos tentam lidar com este problema, armazenando valores para evitar a repetição de cálculos (DARWICHE, 1995); outros tentam reorganizar as variáveis condicionadas utilizando o conceito de *knots* (VEGAS, 1992; PEOT; SHACHTER, 1991), mas ambos perdem a complexidade linear no espaço da versão original.

Por outro lado, *bounded conditioning* (HORVITZ; SUERMONDT; COOPER, 1989) trata o problema da complexidade no tempo retornando respostas aproximadas através de intervalos de variação. Em *bounded conditioning*, a técnica de condicionamento é utilizada para transformar uma rede multi-conectada em uma *polytree* e então utilizar o algoritmo de Pearl (1988) para *polytrees* em cada conjunto de instanciações. Desta forma, *bounded conditioning* pode produzir soluções de compromisso entre tempo e qualidade da resposta, fornecendo soluções de maior qualidade, quando mais inferências para o conjunto de instanciações forem feitas. Este algoritmo tem, portanto, um comportamento *anytime*, mas nenhuma melhora no desempenho será obtida se houver mais memória disponível.

[6] *Loops* são caminhos que partem de uma variável e retornam a ela independentemente do sentido dos arcos.

Existem vários algoritmos que utilizam condicionamento para produzir soluções de compromisso entre espaço e tempo e entre tempo e qualidade da resposta. Por exemplo, condicionamento é empregado dentro dos *super-buckets* e nos algoritmos de condicionamento mais eliminação por Dechter (1996b) para estabelecer soluções de compromisso entre tempo e espaço.

Os algoritmos mais relevantes para esse trabalho, incluem *recursive decomposition* (MONTI; COOPER, 1996) e *recursive conditioning* (DARWICHE, 2001). Nesses trabalhos, o método de condicionamento é usado recursivamente para decompor a rede Bayesiana em nós separados. Essas variáveis são separadas umas das outras através de um conjunto de variáveis condicionadas que recebe o nome de *cutset*. As variáveis são então organizadas em uma árvore binária para se iniciar a inferência. *Recursive decomposition* é particularmente interessante quando implementado de forma *anytime*, transformando-se no chamado *bounded recursive decomposition.* Este algoritmo possui uma fase de inicialização em que resultados intermediários são produzidos e armazenados em *caches.* Quando uma inferência é requisitada, o algoritmo usa os resultados armazenados nos *caches* para produzir intervalos para a resposta. Conforme o tempo, esses resultados são refinados com a adição de novos resultados intermediários.

Recursive conditioning utiliza esssas idéias para apresentar um comportamento *anyspace*. A solução de compromisso entre tempo e espaço é feita, alocando-se *caches* para evitar a repetição de cálculos já feitos. Entretanto, uma solução só será fornecida no final do processo, ou seja, *recursive conditioning* não apresenta um comportamento *anytime*. Em uma configuração, este algoritmo tem complexidade $O(n)$ no espaço e $O(n \exp(w \log n))$ no tempo, onde n é o tamanho da rede Bayesiana e w é o tamanho de uma ordem de eliminação. Em outra, tem complexidade $O(n \exp(w))$ no espaço e $O(n \exp(w))$ no tempo.

Em resumo, diversos são os algoritmos que podem lidar com restrições de memória ou restrições de tempo, porém nenhum deles consegue lidar com aplicações que possuem as duas restrições ao mesmo tempo.

2.6 Inteligência Artificial com Restrições de Tempo e Espaço

A despeito do comportamento em tempo real não ter sido uma preocupação primordial quando do início da inteligência artificial, percebe-se que na hodiernidade tal comportamento é extremamente desejável em muitas aplicações (GARVEY, 1996). A grande dificuldade, entretanto, é como lidar com complexos e pesados modelos sob severas restrições de tempo? Além disso, o que seguramente é um problema ainda maior, como lidar com problemas de inteligência artificial que tenham além de restrições de tempo, também restrições de espaço?

Aplicações de inteligência artificial com restrições de tempo e espaço envolvem robótica, processos de tomada de decisão, diagnóstico automatizado, etc. Nestas aplicações, tenta-se resolver problemas em situações reais onde a reação dentro do tempo especificado é o mais importante fator para o sucesso. Para lidar com estes problemas alguns trabalhos tentam simplificar a computação e retornar apenas uma solução satisfatória. Garvey (1996) analisa vários deste métodos e também apresenta uma definição de sistemas em tempo real que será usada neste trabalho:

Definição 6 *(GARVEY, 1996) Um sistema em tempo real é aquele que garante que uma tarefa será executada dentro das restrições de tempo mesmo no pior caso, quando estas forem as mais severas. Ou seja, um sistema em tempo real sempre retornará a melhor resposta possível (resultado de melhor qualidade) com os recursos disponíveis.*

Apesar desta definição, em se tratando de inteligência artificial, usualmente utiliza-se uma definição menos severa onde um sistema em tempo real sempre retornará uma resposta e irá estatisticamente atingir a qualidade requerida no tempo especificado, porém nenhuma garantia é feita sobre uma tarefa em particular. A fim de atingir esses requerimentos menos severos para um sistema em tempo real, alguns pesquisadores tentam produzir algoritmos que melhoram a qualidade da resposta com o tempo. Esta classe de algoritmos é chamada de *anytime*. Um algoritmo *anytime* foi inicialmente introduzido por Dean e Boddy (1988) no final dos anos 80 e hoje são aplicados a aproximação numérica, busca heurística, algoritmos de Monte Carlo e aprendizagem. Uma definição formal de algoritmos *anytime* é dada a seguir:

Definição 7 *Um algoritmo é* anytime *se este pode produzir uma solução em um dado tempo T, e a qualidade das soluções melhora com o tempo depois de T.*

Exemplos de algoritmos *anytime* são *bounded conditioning* (HORVITZ; SUERMONDT; COOPER, 1989), *term computation* (D'AMBROSIO, 1993) e *Gibbs sampling* (GEMAN; GEMAN, 1984).

Adicionalmente, problemas de inteligência artificial normalmente lidam com grandes conjuntos de dados para representar o conhecimento. Especificamente em modelos probabilísticos tais como redes Bayesianas, a modelagem envolve a construção de grandes tabelas para representar dependências entre variáveis. Entretanto, redes Bayesianas podem ser necessárias em sistemas que possuem recursos de memória bastante limitados, tais como PDAs (*Personal Digital Assistant*), controladores, robôs e sistemas embarcados. Além disso, algumas redes são tão grandes que, mesmo para um computador de mesa, o consumo de memória pode tornar-se um problema. Assim, não apenas o tempo mas também o consumo de memória deve ser considerado, quando se analisam problemas computacionais de inferência probabilística. Para lidar com este problema, alguns pesquisadores criaram uma classe de algoritmos que podem melhorar seu desempenho (retornam uma resposta mais rapidamente), se mais espaço é fornecido a eles. Estes algoritmos são chamados *anyspace*:

Definição 8 *(RAMOS; COZMAN; IDE, 2002) Um algoritmo é* anyspace *se for capaz de melhorar seu desempenho (seja retornando uma resposta mais apurada, seja levando menos tempo para resolver o mesmo problema), com o aumento de espaço de memória fornecido a ele, assumindo-se que tal espaço é maior que uma quantidade mínima.*

Nesta classe de algoritmos estão *recursive conditioning* (DARWICHE, 2001), *dynamic conditioning* (DARWICHE, 1995), *global conditioning* (SHACHTER; ANDERSEN; SZOLOVITS, 1994), *local conditioning* (VEGAS, 1992) e *adaptive variable elimination* (RAMOS; COZMAN; IDE, 2002). Dechter (1996b) também descreve uma série de métodos que estabelecem soluções de compromisso entre espaço e tempo. Ademais, uma boa fonte de informações sobre algoritmos de inferência em redes Bayesianas para sistemas em tempo real pode ser encontrada em (GUO; HSU, 2002).

3 CONDICIONAMENTO ADAPTATIVO

3.1 Conceitos Básicos

O algoritmo descrito nesta seção, denominado condicionamento adaptativo, foi concebido de forma a possibilitar o mais alto nível de controle, considerando-se restrições de tempo e espaço. Como mencionado anteriormente, o objetivo é garantir a execução do processo de inferência em redes Bayesianas dadas restrições de memória e, além disso, possuir também um comportamento *anytime*. Assim, o algoritmo projetado pode produzir soluções de compromisso tempo×espaço em diversas dimensões, essencialmente combinando técnicas de condicionamento com métodos normais de inferência.

A idéia básica do condicionamento adaptativo é dividir a rede inicial em diversas sub-redes, a fim de assegurar que processo ocorrerá dentro das limitações de espaço. A decomposição é feita, condicionando-se variáveis que conectam dois conjuntos de variáveis. Por exemplo, a Fig.3 ilustra a divisão da rede $ABCD$ em duas redes, AB e CD através do condicionamento da variável B, instanciando-a como $B = \bar{b}$[1]. Deseja-se obter o valor para $Pr(D)$ que pode ser calculado com o algoritmo de eliminação de variáveis através da seguinte expressão: $Pr(D) = \sum_C Pr(D|C) \cdot \sum_B Pr(C|B) \cdot \sum_A Pr(B|A) \cdot Pr(A)$. A resposta para este cálculo é $Pr(D) = (0.4491, 0.5509)$. Com o condicionamento da variável B, e a conseqüente divisão da rede em duas sub-redes, a expressão para o cálculo de $Pr(D)$ pode ser obtida através do Teorema 1:

$$Pr(D) = \sum_B \sum_A Pr(B = b|A) \cdot P(A) \cdot \sum_C Pr(D|C) \cdot Pr(C|B = b).$$

Assim:

$Pr(D) = \sum_A$

$Pr(B\|A)$	b	$\bar{b}$
a	0.90	0.10
$\bar{a}$	0.20	0.80

×

$Pr(A)$	a	$\bar{a}$
	0.60	0.40

×

$\times \sum_C$

$Pr(D\|C)$	d	$\bar{d}$
c	0.90	0.10
$\bar{c}$	0.20	0.80

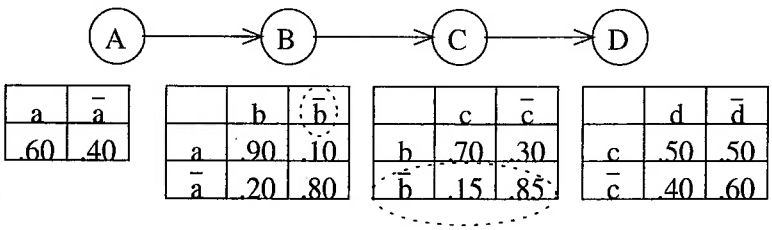
×

$Pr(C\|B = b)$	c	$\bar{c}$
	0.70	0.30

+

[1]Neste trabalho será usada a seguinte representação para os estados das variáveis binárias: $X = x$, quando X for verdadeiro e $X = \bar{x}$, quando X for falso.

Caso 1: Rede antes do condicionamento.



Caso 2: Rede dividida após condicionamento, B=$\bar{b}$.

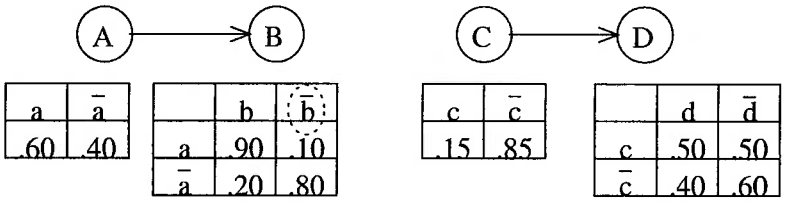


Figura 3: Condicionamento de variáveis para divisão da rede.

$$+\sum_A \begin{array}{|c|c|c|}\hline Pr(B|A) & b & \bar{b} \\ \hline a & 0.90 & 0.10 \\ \hline \bar{a} & 0.20 & 0.80 \\ \hline \end{array} \times \begin{array}{|c|c|c|}\hline Pr(A) & a & \bar{a} \\ \hline & 0.60 & 0.40 \\ \hline \end{array} \times$$

$$\times \sum_C \begin{array}{|c|c|c|}\hline Pr(D|C) & d & \bar{d} \\ \hline c & 0.90 & 0.10 \\ \hline \bar{c} & 0.20 & 0.80 \\ \hline \end{array} \times \begin{array}{|c|c|c|}\hline Pr(C|B=\bar{b}) & c & \bar{c} \\ \hline & 0.15 & 0.85 \\ \hline \end{array}.$$

Resolvendo as multiplicações entre as tabelas e somando A e C, tem-se:

$$\sum_A Pr(B|A) \cdot Pr(A) = Pr(B) \text{ e } \sum_C Pr(D|C) \cdot Pr(C|B=b) = Pr(D|B=b).$$

Assim,

$$Pr(D) = \boxed{Pr(B=b)=0.62} \times \begin{array}{|c|c|c|}\hline Pr(D|B=b) & d & \bar{d} \\ \hline & 0.47 & 0.53 \\ \hline \end{array} +$$

$$+ \boxed{Pr(B=\bar{b})=0.38} \times \begin{array}{|c|c|c|}\hline Pr(D|B=\bar{b}) & d & \bar{d} \\ \hline & 0.415 & 0.585 \\ \hline \end{array}.$$

$$\Longrightarrow Pr(D) = \begin{array}{|c|c|}\hline d & 0.62 \times 0.47 + 0.38 \times 0.415 \\ \hline \bar{d} & 0.62 \times 0.53 + 0.38 \times 0.585 \\ \hline \end{array} = \begin{array}{|c|}\hline 0.4491 \\ \hline 0.5509 \\ \hline \end{array}.$$

Condicionando-se variáveis e dividindo a rede, cada sub-rede pode ser processada

para cada conjunto de valores das variáveis condicionadas. Quanto mais tempo é fornecido, mais inferências para diferentes instanciações podem ser feitas e, conseqüentemente, de mais alta qualidade será a resposta.

Por tratar as sub-redes criadas de forma independente, o condicionamento adaptativo permite que diferentes algoritmos de inferência possam ser utilizados em diferentes sub-redes. Neste caso, o problema global aumenta em complexidade, pois deve ser estudado também qual o melhor algoritmo que deve ser alocado para cada uma das sub-redes criadas. Neste trabalho, as sub-redes serão processadas com o algoritmo de eliminação de variáveis pois ele:

1. Tem complexidade exponencial (tempo e espaço) no tamanho do maior *separator*, configurando o estado-da-arte para algoritmos exatos de inferência em redes Bayesianas;
2. Pode ser facilmente compreendido e implementado;
3. Por ser um algoritmo exato, permite a construção de intervalos de variação para a resposta global, conforme será visto na Seção 3.3;
4. Apresenta alta previsibilidade na utilização da memória após a construção da árvore de junção. Isso ocorre, pois a análise dos *separators* de árvores de junção permite uma fina previsão da quantidade total de memória necessária para inferências.

Apesar disso, outros algoritmos exatos também poderiam ser utilizados em conjunto com o de eliminação de variáveis.

Ademais, para ter um desempenho ainda melhor, o algoritmo de condicionamento adaptativo agrega técnicas adicionais:

- ordenar variáveis condicionadas, a fim classificá-las em ordem de importância - "importância" também será definida posteriormente - de forma que as mais importantes variem mais rapidamente em suas instanciações que as outras;
- gerar instanciações com maiores massas de probabilidade antes de outras instanciações, de forma a aumentar a velocidade do processo de convergência (similar a *term computation* (D'AMBROSIO, 1993));

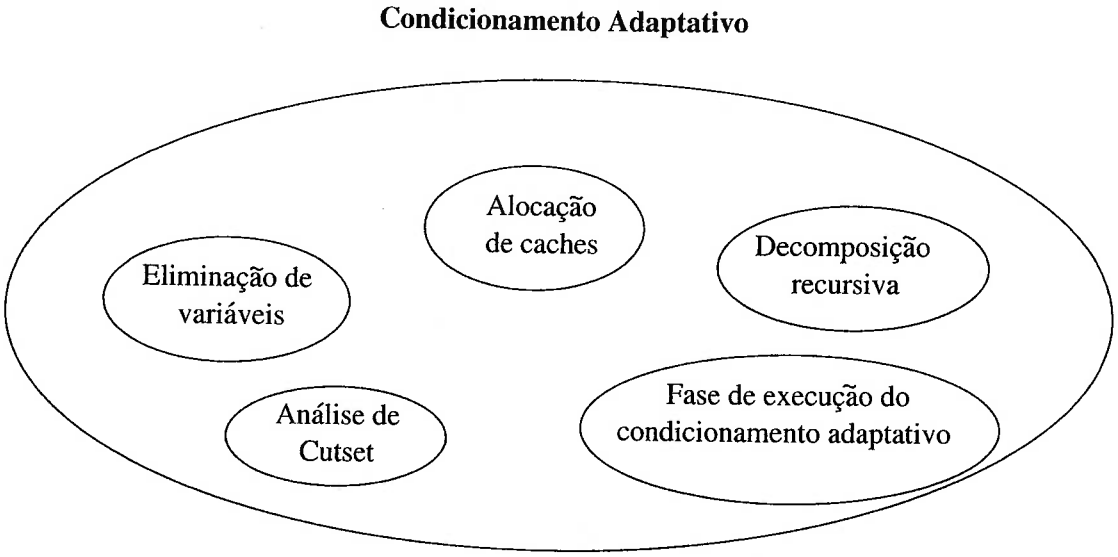


Figura 4: Condicionamento adaptativo.

- alocar *caches* para evitar cálculos repetidos.

A Fig.4 mostra os vários algoritmos que compõem o sistema de condicionamento adaptativo.

O algoritmo de condicionamento adaptativo recebe uma requisição para inferência e uma descrição dos recursos disponíveis, ou seja: a máxima quantidade de memória e o máximo tempo possível para produzir as respostas (quanto mais tempo, melhor será a qualidade da resposta). Para fins de simplificação, a quantidade de memória deve ser fornecida em termos do máximo *separator* permitido. A operação do algoritmo ocorre em duas fases. Na primeira, é feito o planejamento das operações a serem executadas, além das diversas ordenações para melhorar o desempenho (fase de planejamento); na segunda fase são realizadas as operações de inferência (fase de execução). Essas fases são apresentadas nas próximas seções.

3.2 Fase de Planejamento

3.2.1 Decomposição da Rede

A fim de assegurar que o sistema irá operar sob as restrições de memória, será discutido nesta seção como decompor a rede inicial em redes menores e mais simples. O objetivo aqui é, dada uma rede e uma limitação de memória fornecida por um tamanho

máximo de *separator*, encontrar uma decomposição que garanta que nenhuma sub-rede terá um *separator* maior que o limite, quando executar o algoritmo de eliminação de variáveis.

O problema de encontrar a melhor decomposição da rede, ou seja, que forneça sub-redes que tenham o melhor desempenho, é conhecido por ser NP-difícil (WEN, 1991). Entretanto, diversos métodos fornecem heurísticas eficientes para lidar com este problema (KJAERULFF, 1992; COOPER, 1990a; WEN, 1991).

No método implementado, são usadas decomposições recursivas para dividir uma rede nos pontos onde os *separators* da árvore de junção da rede são maiores que o permitido. Assim, inicialmente aplica-se o critério de d-separação (PEARL, 1988) para descartar variáveis que não serão usadas na inferência. Com as variáveis restantes, constrói-se a árvore de junção da rede. Depois disso, são analisados os *separators*, à procura de *separators* maiores que o tamanho máximo permitido. Quando um desses é encontrado, a rede é dividida, condicionando-se as variáveis pertencentes a ele. Então em cada sub-rede o mesmo processo é repetido e assim recursivamente até que nenhum *separator* maior que o permitido exista em qualquer rede. Desta forma, após a realização de todo processo, existirão várias sub-redes, todas com a seguinte propriedade: suas árvores de junção não terão nenhum *separator* maior que o valor limite fornecido.

Na Fig.5 existe uma árvore de junção dividida em várias partes, admitindo-se que o máximo valor para o tamanho do *separator* seja 4 (com todas as variáveis sendo binárias).

Também é necessário determinar quais variáveis fazem parte de quais redes após a escolha das variáveis de condicionamento. Este problema pode ser visualizado como a decomposição de um grafo, onde se retiram os arcos que ligam variáveis condicionadas a seus filhos. Ou seja, imaginando que as variáveis estão organizadas em um vetor, como separar este vetor em duas ou mais partes, as quais representam grafos produzidos pela decomposição do grafo original? Esta tarefa é feita através de uma modificação do algoritmo de busca por ciclos em grafos, inicialmente proposta por Pearl (1988). A idéia básica deste algoritmo é a de utilizar um vetor onde acrescentam-se variáveis junto com seus pais e filhos, até que todas as variáveis interligadas estejam no vetor. Cria-se então um novo vetor para as variáveis que não foram adicionadas até então, e assim sucessivamente. O algoritmo está representado na Fig.6.

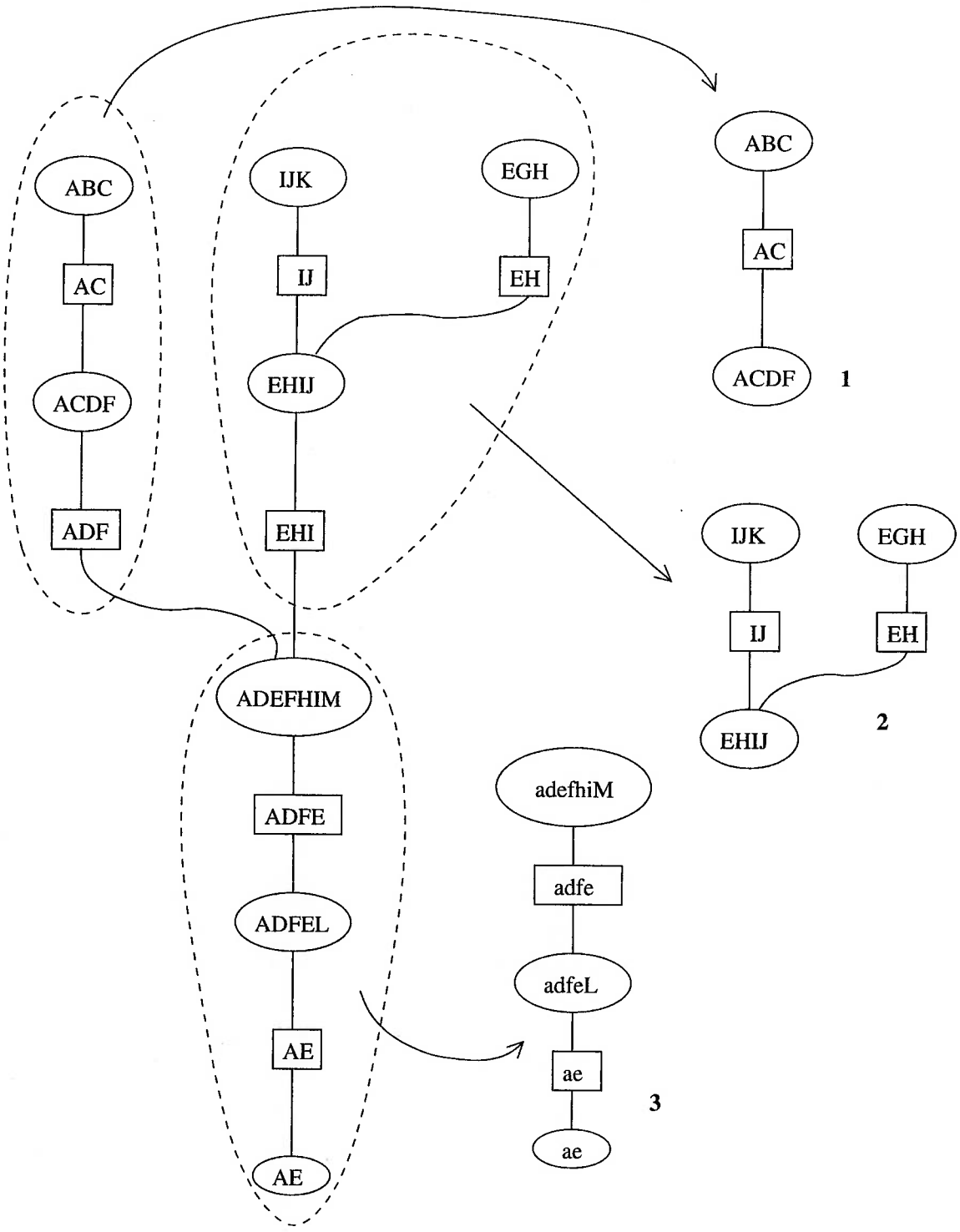


Figura 5: Uma árvore de junção dividida em três conjuntos de variáveis. As variáveis representadas com letras minúsculas foram condicionadas.

Algoritmo de Separação

SEPARAÇÃO(X)
Entrada: $\mathbf{X} \rightarrow$ conjunto de variáveis da rede após condicionamento.
Saída: $\mathbf{V} \rightarrow$ conjunto de vetores de variáveis interligadas.
01. A partir de $\mathbf{X}$, crie um vetor B de elementos booleanos representando as variáveis visitadas, com estado inicial *falso* para todos os elementos;
02. Crie um inteiro i com valor inicial zero;
03. Enquanto houver elementos com estado *falso* em B, faça:
04. Crie um vetor V_i e adicione a ele primeira variável não-visitada de $\mathbf{X}$.
05. Crie dois ponteiros I (inicial) e F (final) com valores iniciais *zero*;
06. Enquanto o valor apontado por I for diferente do apontado por F, faça:
07. Com a variável indicada pelo ponteiro I, acrescente seus pais e filhos no vetor V_i se estes estiverem com estado *falso* em B, a partir da posição indicada pelo ponteiro F.
08. Para cada variável adicionada, modifique seu estado no vetor B para *verdadeiro*.
09. Faça a posição indicada por I ser acrescida de 1 e F apontar para a última variável adicionada;
10. Adicione V_i a $\mathbf{V}$ e acrescente i de um;
11. Retorne ($\mathbf{V}$).

Figura 6: Pseudocódigo para o algoritmo de separação.

Para descrever o processo de decomposição, considere os *clusters* organizados em um vetor. O processo de decomposição começa analisando o primeiro *cluster* no vetor e seu respectivo *separator*. Tomando como exemplo a árvore da Fig.5, o primeiro *cluster* é composto por *A*,*B* e *C*. Como *A* e *C* são variáveis binárias, o tamanho do *separator* é 2 x 2 = 4, portanto a rede não precisa ser dividida neste ponto. O próximo *cluster* é *ACDF*; como seu *separator* tem tamanho 8, portanto maior que o limite, as variáveis de seu *separator* são condicionadas a fim de dividir a árvore. Como resultado têm-se duas sub-árvores sendo uma com os *clusters ABC* e *ACDF* e a outra com os *clusters* restantes. O processo continua recursivamente nas sub-árvores criadas até que não reste nenhum *separator* maior que 4. Como indicado na Fig.5, três sub-árvores foram criadas após o processo.

Após a criação das sub-redes, são adicionados alguns elementos a elas que facilitarão o processo de inferência que ocorrerá na fase de execução. O primeiro elemento adicionado é a lista das variáveis do *cutset* presentes na sub-rede em questão. Estas variáveis, denominadas *localcutset,* são úteis para informar quais variáveis devem ser adicionadas às variáveis de interesse desta sub-rede no cálculo das probabilidades marginais. Ele representa as variáveis em $\mathbf{C}_i$ indicadas na Equação (2.3) e sua definição

segue abaixo:

Definição 9 *O* localcutset $\mathbf{C}_i$ *de uma sub-rede* $\mathcal{N}_i$ *é definido como o conjunto de variáveis de* $\mathcal{N}_i$ *que fazem parte do* cutset $\mathbf{C}$ *de* $\mathcal{N}$.

Além do *localcutset*, também são adicionadas cópias das variáveis do *cutset* que não fazem parte da sub-rede, mas que possuem filhos nesta sub-rede. Essas variáveis estarão sempre observadas no processo de inferência por serem variáveis condicionadas e, portanto, sempre instanciadas com um valor determinado. Para posterior referência, essas variáveis são denominadas *addedcutset*.

Definição 10 *O* addedcutset $\mathbf{A}_i$ *de uma sub-rede* $\mathcal{N}_i$ *é definido como o conjunto de cópias das variáveis do* cutset $\mathbf{C}$ *de* $\mathcal{N}$ *que têm filhos na sub-rede* $\mathcal{N}_i$ *mas que não estão em* $\mathcal{N}_i$.

Na Fig.8, as variáveis adicionadas pertencentes ao *addedcutset* são representadas com linhas traço-ponto. Esta decomposição foi obtida a partir da rede da Fig.7.

O algoritmo de decomposição não é um processo *anytime;* ou seja, a qualidade da resposta não melhora com o tempo. Felizmente, de acordo com alguns resultados experimentais para redes reais, esta fase consome apenas 0.5% do tempo total de processamento; assim, não é tão relevante para o desempenho total do sistema.

3.2.2 Análise de Cutsets

No processo de inferência, deve-se rodar o algoritmo de eliminação de variáveis em cada sub-rede criada, para cada valor das variáveis do *cutset*. Esta seção tem por objetivo explicar alguns métodos heurísticos que podem melhorar o desempenho geral das inferências. A primeira heurística a ser tratada tem como finalidade a organização das variáveis do *cutset*. A idéia é ordenar as variáveis do *cutset* de forma que as variáveis mais próximas às variáveis de interesse variem mais rapidamente que as outras no processo de instanciação. Além disso, também os estados que cada variável do *cutset* percorrerá também são organizados segundo suas densidades de probabilidade. Desta forma, estados com densidade de probabilidade maiores serão instanciados preferencialmente, aumentando a velocidade de convergência para a resposta exata no início do processo.

A seguir, será apresentado o conceito da divergência de Kullback-Leibler utilizado para explicar o algoritmo de organização das variáveis do *cutset*. Depois disso, será apresentado o método denominado Mínima Distância Média e como funciona a ordenação dos estados das variáveis do *cutset* para o processo de instanciação.

Divergência de Kullback-Leibler e medidas relacionadas

A divergência de Kullback-Leibler, também conhecida como entropia cruzada ou distância entrópica entre duas funções de probabilidade f e g, é definida como:

$$C(f,g) = \begin{cases} \sum f \cdot \ln\left(\frac{f}{g}\right) & \text{se } f > 0 \text{ e } g > 0 \\ 0 & \text{se } f = 0 \text{ e } g = 0 \\ \text{não definida} & \text{caso contrário} \end{cases} \tag{3.1}$$

e expressa a divergência entre f e g, onde a expectativa é referente a f.

Já a influência de uma variável em outra pode ser medida usando a *informação mútua condicionada*. Considere uma função de probabilidade p sobre um conjunto de variáveis $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$. A informação mútua condicionada, escrita com $I(X_1, X_2 \mid \mathbf{X}_c)$, é definida como a entropia-cruzada $C(p,q)$ onde q é obtida removendo-se o arco $X_1 - X_2$ que faz X_1 independente de X_2 dado o conjunto de variáveis $\mathbf{X}_c = \mathbf{X} \setminus \{X_1, X_2\}$. Em termos de p, q é dado como:

$$q = \frac{p(X_1, \mathbf{X}_c)\, p(X_2, \mathbf{X}_c)}{p(\mathbf{X}_c)}. \tag{3.2}$$

Kjaerulff prova uma série de propriedades em (KJAERULFF, 1993, 1994):

1. A divergência de Kullback-Leibler pode ser computada localmente;
2. A divergência de Kullback-Leibler é aditiva;
3. O erro é atuado com o aumento da distância, ou seja, se houver uma simplificação do modelo, por exemplo pela remoção de um arco entre duas variáveis, o erro associado a essa simplificação é menor em variáveis mais distantes deste ponto.

A terceira propriedade indica que o erro devido à remoção de um arco é menor em variáveis distantes daquela onde o arco foi removido. Este raciocínio leva, intuitivamente, à seguinte afirmação: quanto maior a distância entre duas variáveis (medida pelo menor

caminho entre elas), menor será a influência que uma variável terá na outra. Esta conclusão é o fundamento do critério de mínima distância média explicado a seguir.

Mínima Distância Média

Conforme discutido anteriormente, a informação mútua condicionada fornece uma medida da influência que uma variável exerce sobre outra, não obstante seu cálculo seja bastante custoso computacionalmente. Para cada função, seria necessário somar todas as variáveis e também calcular seu logaritmo; depois disso, os resultados devem ser somados para cada sub-rede e todo processo deve ser repetido para cada variável de interesse. Para computar a informação mútua condicionada para cada sub-rede, tem-se complexidade $O(m \exp(n))$ no tempo, onde n é o número de variáveis da sub-rede e m o número de variáveis de interesse (*query*).

Nesta seção é apresentado um novo método que aproxima a informação mútua condicionada, retornando uma resposta qualitativa da influência entre variáveis. O princípio utilizado é o de que a influência entre duas variáveis decresce conforme a distância entre elas aumenta. Portanto, para aproximar esta análise basta verificar a distância entre as variáveis de interesse e as outras. Esta tarefa pode ser facilmente resolvida utilizando-se o algoritmo de menor-caminho (*shortest-path*) que conta o número de arcos entre duas variáveis pelo menor caminho possível entre elas. Depois disso, essas distâncias são somadas para cada variável de interesse através da seguinte expressão:

$$MDM\left(X'\right) = \sum_{i=0}^{n} \frac{d\left(X', X_i\right)}{n}, \tag{3.3}$$

onde n é o número de variáveis de interesse e $d\left(X', X_i\right)$ é a distância entre X' e X_i. Este método adota uma aproximação de que a informação mútua condicionada entre um nó pai e seus filhos é sempre a mesma para todas as variáveis.

A saída deste método é um valor de MDM para cada variável do *cutset*. Com isso, pode-se ordenar as variáveis dentro do *cutset*, sendo que valores de MDM maiores implicam em variáveis menos importantes para a rede. Desta forma, estas irão variar com freqüência menor no processo de instanciação.

Ordenação de Estados

Além da organização do *cutset*, pode-se também aumentar a velocidade de convergência do sistema, analisando-se os estados de cada variável do *cutset* e qual a melhor forma de variá-los. A idéia é computar primeiramente as instanciações dos estados que fornecem os maiores valores para a massa de probabilidade da sub-rede (método baseado em busca como o apresentado por Henrion em (HENRION, 1991)). Entretanto, essa busca pode ser bastante ineficiente computacionalmente, já que o número de possibilidades de instanciações é exponencial no número de variáveis condicionadas da sub-rede. Para lidar com este problema, podem-se usar heurísticas como por exemplo avaliar a *distribuição posterior* de cada variável condicionada que faz parte da sub-rede, deixando as variáveis condicionadas adicionadas à sub-rede (*addedcutsets*) como não-observadas. Ou seja, deve-se calcular $Pr^{\mathcal{N}_i}(C_j)$ para cada sub-rede i, para cada variável condicionada j. Com estas distribuições podem-se então enumerar os estados de variação de cada variável C_j, do maior para o de menor valor.

Nos testes realizados, o efeito deste método em algumas redes foi o de tornar a convergência muito mais rápida, ou seja, a resposta se aproxima mais rapidamente do valor exato. Em outras redes, não houve uma melhora significativa, porém em nenhuma das redes testadas houve piora na convergência.

3.2.3 Alocação de *Caches*

Nesta seção será apresentada outra técnica para aumentar o desempenho do sistema, quando pequenas porções de memória ainda estiverem disponíveis. Esta técnica representa uma solução de compromisso entre tempo e memória, porém, diferentemente do processo de decomposição, possui um comportamento suave, ou seja, um recurso não sofre alterações bruscas quando se aumenta gradativamente o outro. O princípio deste método é alocar *caches* (porções de memória) para que o sistema utilize ao máximo os recursos de espaço disponíveis. Esses *caches* podem armazenar resultados de inferências já realizadas, de forma que estas não precisem ser repetidas, economizando tempo de processamento.

Apesar da alocação de *caches* ocorrer durante a fase de execução, ela será discutida aqui por constituir um processo de planejamento para a efetiva computação da resposta. Apenas optou-se por implementá-la na fase de execução por ser neste momento em que

se tem o exato consumo de memória e portanto, o quanto pode ser alocado para os *caches*.

Durante o processo de decomposição, a rede inicial é decomposta em sub-redes menores para diminuir o consumo total de memória. Como resultado, têm-se diversas sub-redes cujo tamanho do *separator* é menor ou igual ao limite determinado. Entretanto, não há garantia de que elas utilizem o tamanho máximo disponível, ou seja, a decomposição pode originar sub-redes que usem menos memória que a quantidade total disponível. Com a utilização de *caches*, esta porção de memória que seria desperdiçada, pode ser alocada para armazenar resultados anteriores, e evitar que esses cálculos sejam feitos novamente na fase de execução. O problema desta técnica está na seguinte questão: como alocar *caches* entre as várias sub-redes e como atualizar seus valores de forma a maximizar o desempenho?

O problema será tratado aqui com heurísticas simples que forneceram resultados satisfatórios nos experimentos realizados. Trabalhos futuros poderão retornar ao problema de alocação de *caches*, investigando métodos ótimos para esssa tarefa. A heurística principal para a alocação de *caches* tem a seguinte descrição: aloque uma unidade de *cache*[2] para cada sub-rede em ordem decrescente de tamanho dessas sub-redes, até que toda a quantidade de memória disponível seja alocada. O espaço disponível para a alocação de caches é calculado como a diferença entre o maior *separator* permitido e o maior *separator* necessário para o processo de inferência, dentre as sub-redes formadas após a decomposição. Se em uma sub-rede já existir uma unidade de *cache* alocada, o processo continua alocando uma segunda unidade e assim sucessivamente. A atualização dos *caches* segue a lógica: atualize o valor do *cache* com o mais novo valor de inferência realizado em sua sub-rede.

3.3 Fase de Execução

A fase de execução inicia-se após a fase de decomposição e após a análise de *cutset*, porém antes do método de alocação de *caches*. Nesta fase, a qual é considerada um processo *anytime*, as sub-redes são processadas para cada instanciação das variáveis no *cutset*. Ao final do processo, limites superior e inferior são calculados, indicando onde a

[2]Uma unidade de *cache* possui a quantidade de memória necessária para armazenar a tabela de probabilidades marginais para as variáveis de interesse e *localcutset* de uma determinada sub-rede ($Pr^{\mathcal{N}}(\mathbf{Q}, \mathbf{C})$).

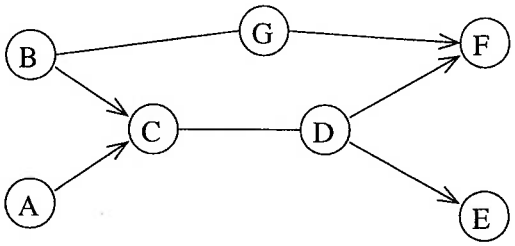


Figura 7: Uma rede Bayesiana genérica $\mathcal{N}$ que será decomposta.

resposta exata pode estar.

Após o primeiro processamento de cada sub-rede, utilizando o algoritmo de eliminação de variáveis, pode-se determinar qual foi o maior *separator* de fato utilizado. Com esta informação, é possível avaliar quanto espaço ainda está disponível no instante de máximo consumo de memória e assim verificar se existe espaço disponível para a alocação de *caches*. Neste caso, o algoritmo de alocação de *caches* pode ser utilizado, a fim de alocar a maior quantidade de *caches* possível até atingir o limite de espaço. Assim, no momento de máximo consumo, a memória utilizada pode ser calculada da seguinte forma:

$$Memória = Max\,Separator + \sum_i Caches(\mathcal{N}_i) \tag{3.4}$$

onde $Caches(\mathcal{N}_i)$ indica a quantidade de memória utilizada pelos *caches* da rede $\mathcal{N}_i$.

Feita a alocação de *caches*, a fase de execução continua computando $Pr^{\mathcal{N}_i}(\mathbf{Q}_i, \mathbf{C_i})$ para cada rede $\mathcal{N}_i$ e para cada instanciação diferente das variáveis em $\mathbf{C}$, como mostra o Teorema 1. Note que o número de inferências cresce exponencialmente com o número de variáveis em $\mathbf{C}$.

Como um exemplo, considere a rede $\mathcal{N}$ na Fig.7. Esta rede é decomposta em três sub-redes através do condicionamento de C e B. A Fig.8 mostra o resultado desta decomposição. Nesta figura os nós desenhados com traço-ponto fazem parte dos *addcutsets*. Os nós desenhados com espessura dupla constituem os *localcutsets*. Para fins de implementação, outra possibilidade é reduzir as tabelas de probabilidade dos filhos das variáveis condicionadas, considerando os valores assumidos pelo *cutset*, assim como foi indicado na Fig.3.

Neste exemplo, para se calcular as probabilidades marginais de E e F, deve-se com-

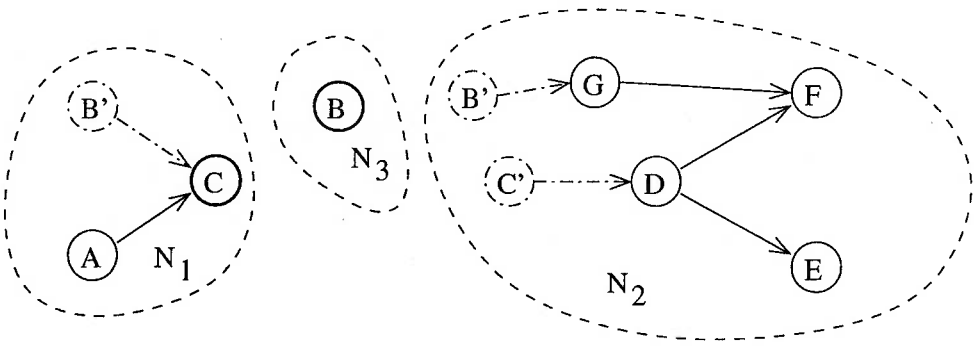


Figura 8: Rede $\mathcal{N}$ decomposta em três sub-redes.

putar as seguintes probabilidades: $Pr^{\mathcal{N}_1}(C \mid B' = b')$, $Pr^{\mathcal{N}_2}(E, F \mid C' = c', B' = b')$ e $Pr^{\mathcal{N}_3}(B)$. Supondo que todas as variáveis são binárias e considerando que B e C estão condicionadas, deve-se calcular $Pr^{\mathcal{N}_1}(C \mid B' = b')$ duas vezes, uma vez para cada instanciação de B'; deve-se calcular $Pr^{\mathcal{N}_2}(E, F \mid C' = c', B' = b')$ quatro vezes, pois são quatro combinações de instanciações possíveis para B e C; e computar $Pr^{\mathcal{N}_3}(B)$ apenas uma vez, já que não existe variável condicionada que seja pai de alguma variável desta rede.

A Fig.9 mostra todos os passos executados na fase de execução. O algoritmo implementa o Teorema 1 e ainda considera a existência de *caches* para armazenar valores intermediários. Como é possível perceber pelo pseudocódigo, pode-se atingir complexidade linear no espaço se não forem utilizados *caches*; apenas será necessário armazenar a rede. A complexidade computacional para este algoritmo é afetada pelo número de inferências que devem ser executados em cada sub-rede, o qual depende do número de variáveis do *cutset*:

Teorema 2 *Dada uma rede Bayesiana com n variáveis e um cutset de tamanho w_c que decompõe a rede em w_n sub-redes, o número de inferências executadas pelo condicionamento adaptativo tem complexidade $O(w_n \cdot exp(w_c))$.*

Os limites para as densidades de probabilidade são facilmente calculados no final do processo. Caso a fase de execução termine devido ao limite de tempo, o resultado retornado antes da normalização não somará um. Isto acontece porque as sub-redes não foram computadas para todas as instanciações, restando ainda valores de probabilidade para serem somados. Assim, os limites inferiores são exatamente os valores obtidos antes da normalização, já que, se o processo continuasse, densidades poderiam ser somadas mas nunca subtraídas do valor já obtido. Por outro lado, os limites superiores

Condicionamento adaptativo — Fase de Execução

AC(T)
Entrada: T → conjunto de sub-redes
Saída: $resultado$ → probabilidade marginal
01. $resultado \leftarrow 0$
02. para cada instanciação das variáveis do *cutset* faça:
03. se tempo não acabou então
04. $resultado \leftarrow resultado$ + MULTI(T)
05. senão
06. normalize($resultado$); BOUNDS($resultado$)
07. retorne $resultado$
08. normalize($resultado$)
09. retorne ($resultado$)

MULTI(T)
Entrada: T → conjunto de sub-redes
Saída: $resultado$ → probabilidade marginal
01. $resultado \leftarrow 1$
02. para cada sub-rede em T faça:
03. se cache[E] está vazio então
04. $resultado \leftarrow resultado \cdot inferência(E)$
05. cache[E] $\leftarrow inferência(E)$
06. senão
07. $resultado \leftarrow resultado$·cache[E]
08. retorne ($resultado$)

BOUNDS(Pr)
Entrada: Pr → probabilidade marginal
Saída: $inferior, superior$ → intervalo
01. $inferior[i] \leftarrow 0$; $superior[i] \leftarrow 0$; $soma \leftarrow 0$
02. para cada categoria de Pr
03. $inferior[i] \leftarrow$ Pr[i]
04. $soma \leftarrow soma + inferior[i]$
05. $Diff = 1 - soma$
06. para cada categoria de $inferior$ faça
07. $superior[i] \leftarrow inferior[i] + Diff$
08. retorne ($inferior, superior$)

Figura 9: Pseudocódigo para a fase de execução do condicionamento adaptativo.

$Pr(X_i$ =baixo)	$Pr(X_i$ =normal)	$Pr(X_i$ =alto)
0.12	0.56	0.17

Tabela 1: Exemplo de resultado obtido antes da normalização para uma inferência com restrição de tempo.

Limites	$Pr(X_i$ =baixo)	$Pr(X_i$ =normal)	$Pr(X_i$ =alto)
Inferior	0.12	0.56	0.17
Superior	0.27	0.71	0.32

Table 2: Limites inferiores e superiores para as probabilidades marginais de X_i .

podem ser calculados somando-se as probabilidades e subtraindo este resultado de um. Este valor pode então ser somado ao limite inferior para obter-se o limite superior, conforme mostra a expressão abaixo:

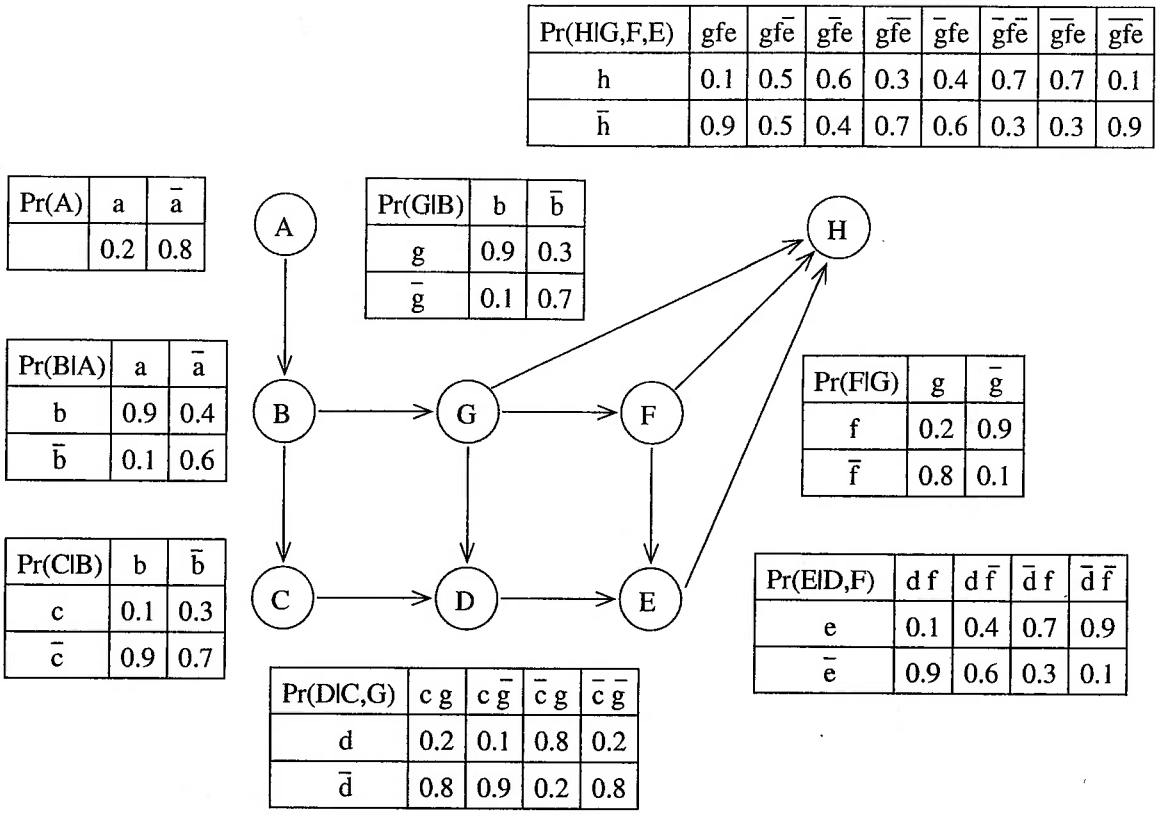
$$Pr(\mathbf{X_i} = a)_{superior} = Pr(\mathbf{X}_i = a)_{inferior} + 1 - \sum_{\phi \in \Psi \setminus \{a\}} Pr(\mathbf{X}_i = \phi)_{inferior} \qquad (3.5)$$

onde α é estado onde se deseja obter o limite superior e Ψ o conjunto de todos os estados da variável.

Como um exemplo, considere a Tab.1 como uma possível resposta para a probabilidade marginal de uma variável com três categorias (baixo, normal e alto), para uma inferência com restrições de tempo.

Os limites inferiores são exatamente os valores indicados na Tab.1. Para computar os limites superiores, deve-se primeiro calcular a diferença entre a soma de todas densidades já calculadas e um. Para o caso da Tab.1 este valor é: $1 - (0.12 + 0.56 + 0.17) = 0.15$. Os limites superiores são então iguais aos limites inferiores mais 0.15. A Tab.2 ilustra os limites encontrados para a inferência da variável X_i.

A resposta obtida na Tab.1 pode também ser normalizada para ter-se uma melhor aproximação. Como o resultado foi produzido tentando-se obter as maiores densidades de probabilidade no início do processo, graças à ordenação dos estados de variação das variáveis do *cutset*, a resposta depois da normalização se aproxima bastante da resposta exata. Nas próximas seções, estas afirmações serão comprovadas através de resultados experimentais obtidos com redes reais.



Pr(H|G,F,E)	gfe	gf$\bar{e}$	g$\bar{f}$e	g$\bar{f}\bar{e}$	$\bar{g}$fe	$\bar{g}$f$\bar{e}$	$\bar{g}\bar{f}$e	$\bar{g}\bar{f}\bar{e}$
h	0.1	0.5	0.6	0.3	0.4	0.7	0.7	0.1
$\bar{h}$	0.9	0.5	0.4	0.7	0.6	0.3	0.3	0.9

Pr(A)	a	$\bar{a}$
	0.2	0.8

Pr(G|B)	b	$\bar{b}$
g	0.9	0.3
$\bar{g}$	0.1	0.7

Pr(B|A)	a	$\bar{a}$
b	0.9	0.4
$\bar{b}$	0.1	0.6

Pr(F|G)	g	$\bar{g}$
f	0.2	0.9
$\bar{f}$	0.8	0.1

Pr(C|B)	b	$\bar{b}$
c	0.1	0.3
$\bar{c}$	0.9	0.7

Pr(E|D,F)	df	d$\bar{f}$	$\bar{d}$f	$\bar{d}\bar{f}$
e	0.1	0.4	0.7	0.9
$\bar{e}$	0.9	0.6	0.3	0.1

Pr(D|C,G)	cg	c$\bar{g}$	$\bar{c}$g	$\bar{c}\bar{g}$
d	0.2	0.1	0.8	0.2
$\bar{d}$	0.8	0.9	0.2	0.8

Figura 10: Rede Bayesiana didática e tabelas de probabilidade.

3.4 Exemplo Prático

Para ilustrar todos os métodos do condicionamento adaptivo, será apresentado um exemplo prático com uma rede criada especialmente para fins didáticos. Esta rede, juntamente com suas tabelas de probabilidade, pode ser visualizada na Fig.10. A rede é composta por 8 variáveis binárias com as relações entre elas indicadas na Fig.10. Neste exemplo, o objetivo é calcular a probabilidade marginal de H, $Pr(H)$, sem haver nenhuma variável observada na rede, e com restrição de memória de 24 bytes, ou seja, o tamanho máximo que o algoritmo pode utilizar para armazenar tabelas é de 24 bytes[3]. Considerando-se que cada valor de ponto flutuante tipo *double* ocupa 4 bytes, pode-se armazenar no máximo 6 valores com essa restrição. Neste exemplo, não existe restrição de tempo, e portanto a resposta exata será obtida ao final do processo.

[3]Este valor considera apenas a memória utilizada pelo algoritmo, sem a memória necessária para armazenamento da rede.

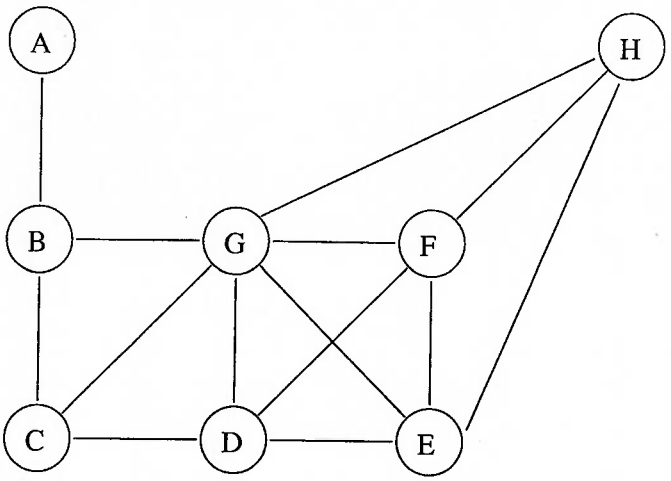


Figura 11: Rede triangularizada a partir de seu grafo moral.

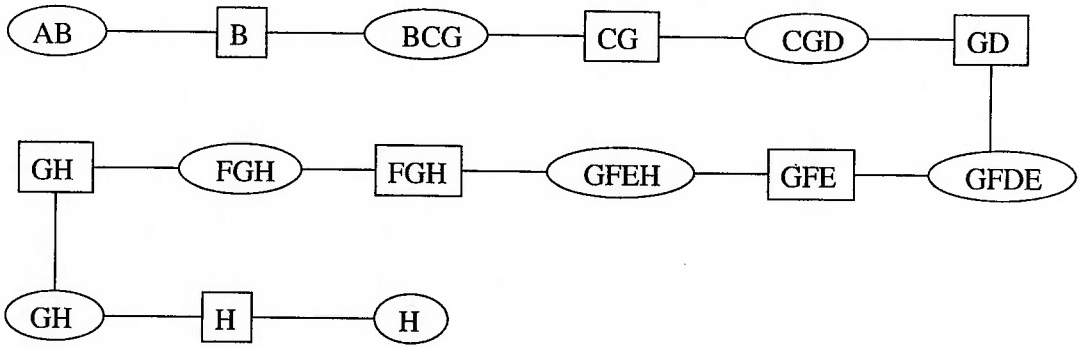


Figura 12: Árvore de junção para a rede do exemplo.

3.4.1 Fase de Planejamento

A primeira etapa do condicionamento adaptativo é a utilização dos métodos de d-separação para simplificar a rede. Neste caso, nenhuma variável é d-separada da rede, ou seja, todas as variáveis da rede devem ser incluídas no cálculo. A próxima etapa é a obtenção da árvore de junção da rede. Conforme descrito na Seção 2.2, a construção de uma árvore de junção inicia-se com a obtenção do grafo moral e posterior triangularização. Para a rede deste exemplo, a triangularização pode ser visualizada na Fig.11. Com esta triangularização pode-se agrupar as variáveis conectadas em *clusters* e *separators* para formar a árvore de junção. O resultado desta operação é apresentado na Fig.12.

A partir da árvore de junção pode-se analisar os requisitos de memória da rede para o algoritmo de eliminação de variáveis. Esta análise é feita a partir dos *separators* indicados pelos retângulos. Cada separator precisa de uma quantidade de memória de 2^n valores para armazenar variáveis binárias, sendo n o número de variáveis do *separator*. Observando-se os *separators* da Fig.12 percebe-se que a rede deve ser dividida no *se-*

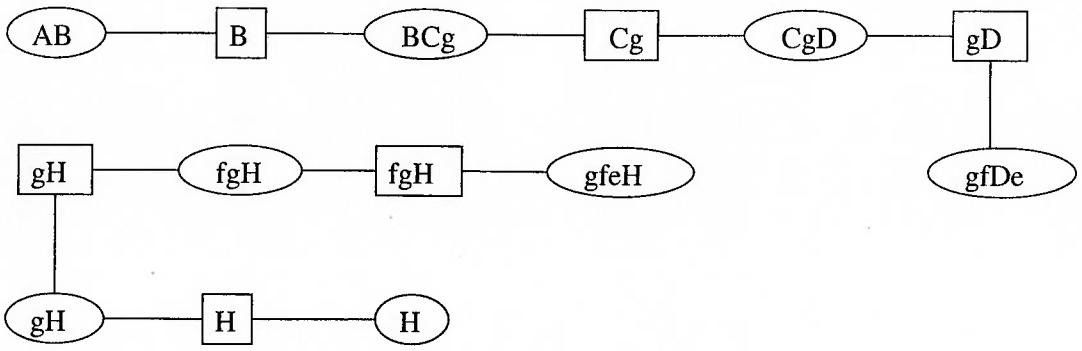


Figura 13: Árvore de junção dividida pelo condicionamento das variáveis GFE. Por estarem condicionadas, estas são representadas com letras minúsculas.

Rede	Prob. Marginal	Var. de Interesse	*Localcutset*	*Addedcutset*
N_1	$Pr(E, G\|F')$	E, G	E, G	F'
N_2	$Pr(F\|G')$	F	F	G'
N_3	$Pr(H\|G', F', E')$	H	$\emptyset$	G', F', E'

Tabela 3: Definições para as sub-redes.

parator GFE por possuir 2^3 valores. O resultado desta divisão na árvore de junção é mostrada na Fig.13, e na rede original, na Fig.14.

Conforme explicado na Seção 3.2.1, a decomposição da rede é recursiva. Portanto, após a primeira decomposição, deve-se analisar as sub-redes formadas e verificar se não há *separators* que excedem o limite. Observando-se as sub-redes da Fig.14, verifica-se que tanto a sub-rede N_2 como a N_3 contêm apenas uma variável real, isto é, uma variável não pertencente à *addedcutsets*. Assim, o processo de inferência nessas redes é extremamente simples, bastando buscar os valores na tabela de probabilidades. A análise recai então sobre a sub-rede N_1. Para esta sub-rede a árvore de junção é apresentada na Fig.15. Nota-se que nenhum *separator* contém mais do que 3 variáveis, portanto, ela não precisa ser dividida novamente, o que encerra o processo de decomposição recursiva.

Deste ponto em diante, pode-se analisar as sub-redes bem como o *cutset* de forma a organizá-lo para a posterior fase de execução. Com a divisão indicada na Fig.14, tem-se na Tab.3 as definições dos conjuntos para cada uma das sub-redes.

Com o *cutset* definido, pode-se iniciar sua organização com as técnicas de mínima distância média e ordenação de estados. Para as variáveis do *cutset,* G, F, E, tem-se as distâncias mínimas médias para a variável H indicadas na Tab.4. Como o valor é o mesmo para as três variáveis, não é necessário reorganizar as variáveis do *cutset* para a fase de execução.

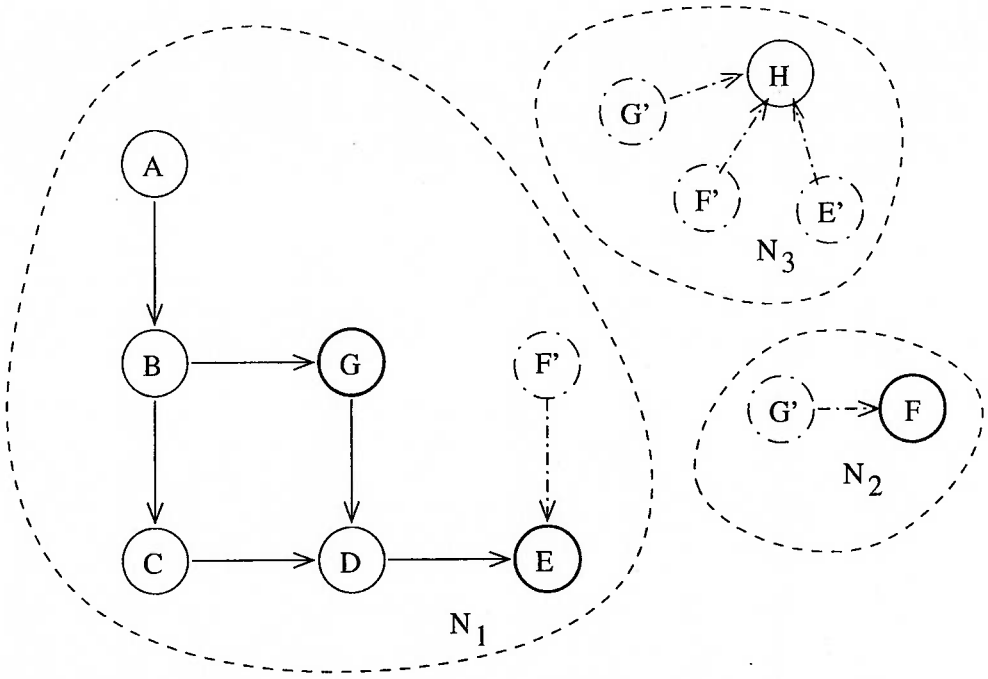


Figura 14: Rede dividida após condicionamento. Os nós desenhados com espessura dupla representam variáveis dos *localcutsets* e os nós com linha do tipo traço-ponto fazem parte dos *addedcutsets*.

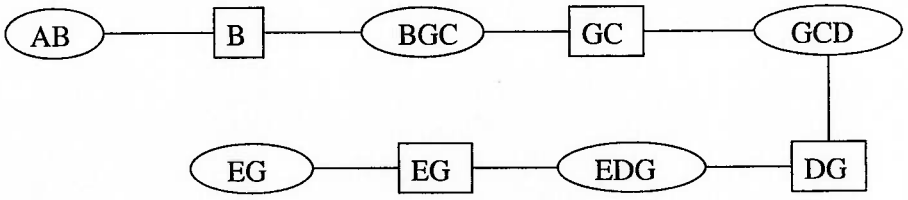


Figura 15: Árvore de junção da rede N_1.

Variável	MDM
E	1
F	1
G	1

Tabela 4: MDMs para as variáveis do *cutset*.

Rede	Variável	Verdadeiro	Falso
N_1	E	0.52775	0.47225
N_1	G	0.6	0.4
N_2	F	0.55	0.45

Tabela 5: *Posteriors* para as variáveis do *cutset*.

A próxima etapa é a organização dos estados de variação das variáveis do *cutset*, também para melhorar a convergência. Conforme explicado na Seção 3.2.2 pode-se utilizar uma heurística baseada no *posterior* das variáveis do *cutset*. Após a decomposição, calcula-se o *posterior* dessas variáveis para que, durante a fase de execução, os estados de variação dessas obedeçam a ordem decrescente das densidades de probabilidade dos *posteriors*. Pretende-se, com isso, somar ao resultado final as maiores massas de probabilidade logo no início do processo, a fim de melhorar a convergência. Para as três variáveis do *cutset*, tem-se os *posteriors* indicados na Tab.5. Esses valores indicam que para as três variáveis, segundo o método de organização de estados, deve-se instanciá-las primeiro com estado *verdadeiro* para depois instanciá-las com *falso*.

A próxima etapa é a alocação de *caches*. Como a restrição de memória era de 6 valores para o *separator* e conseguiu-se, após a decomposição, um *separator* de tamanho 4, pode-se utilizar os 2 valores que sobram para um *cache*. Este *cache* pode ser alocado tanto para a rede N_2 como para a N_3. No caso da rede N_1, o *cache* não pode ser utilizado uma vez que uma inferência nesta rede requer memória para armazenar no mínimo 4 valores. Assim, a decisão fica entre alocar o *cache* para a rede N_2 ou N_3. Como ambas têm basicamente a mesma complexidade, o *cache* será alocado para a rede N_2 pois esta apresenta apenas uma variável em seu *addedcutset*. Portanto, a variação de suas inferências será menor durante a fase de execução e, conseqüentemente, a técnica de *caching* será mais eficiente.

3.4.2 Fase de Execução

Nesta fase são feitas instanciações das variáveis do *cutset* e as sub-redes são processadas. Os resultados são então multiplicados e somados para se obter a resposta. Admitindo que as variáveis do cutset são instanciadas na ordem G, E, F, tem-se as Tabs. 6 e 7 para os respectivos valores de $Pr(H = h)$ e $Pr(H = \bar{h})$. Nessas tabelas, estão indicados o incremento ao resultado final de cada inferência dada, as instanciações de G, E e F, os limites superiores e inferiores para as probabilidades, os valores normalizados

G	E	F	Incremento	Lim. Sup.	Lim. Inf.	Normalizado	Erro Abs.
V	V	V	0.003288	0.970408	0.003288	0.1	0.380436
V	V	F	0.15696	0.865768	0.160248	0.544173	0.063737
V	F	V	0.04356	0.822208	0.203808	0.534088	0.053652
V	F	F	0.06552	0.669328	0.269328	0.44888	0.031556
F	V	V	0.085896	0.540484	0.355224	0.435997	0.044439
F	V	F	0.0227845	0.530719	0.378009	0.446139	0.034297
F	F	V	0.101682	0.487141	0.479691	0.483292	0.002856
F	F	F	7.45E-4	0.480436	0.480436	0.480436	0.0

Tabela 6: Resultados de inferências durante a fase de execução. Os valores indicados representam a probabilidade de $H = h$.

após a soma de cada conjunto de inferências e o erro absoluto. O incremento equivale à resposta dada pela multiplicação das probabilidades marginais nas redes N_1, N_2 e N_3, considerando as instanciações indicadas:

$$Pr^{N_1}(E, G) \times Pr^{N_2}(F) \times Pr^{N_3}(H)$$

A cada linha, um novo incremento é calculado e seu valor somado à resposta anterior. O limite inferior equivale à soma dos incrementos já obtidos e o limite superior é calculado conforme a Expressão 3.5. Por exemplo, para a primeira linhas das tabelas, tem-se os seguintes valores:

$Pr^{N_1}(E, G) =$

	e	$\bar{e}$
g	0.1644	0.4356
$\bar{g}$	0.2386	0.1614

, $Pr^{N_2}(F) =$

f	$\bar{f}$
0.2	0.8

e $Pr^{N_3}(H) =$

h	$\bar{h}$
0.1	0.9

.

Como as instanciações neste caso são: $G = g$, $E = e$ e $F = f$, o incremento é igual a 0.1644 × 0.2 × 0.1 = 0.003288 para $H = h$. No outro caso, para $H = \bar{h}$, tem-se incremento igual a 0.1644 × 0.2 × 0.9 = 0.029592.

Percebe-se que após todas as instanciações, a resposta obtida é a resposta exata, e os limites superior e inferior tornam-se iguais a ela. Pode-se verificar também que o intervalo de variação para as probabilidades, ou seja, a diferença entre os limites superior e inferior vai diminuindo conforme mais instanciações são realizadas e computadas. A Fig.16 mostra a variação dos limites com as instanciações no processo de convergência para a resposta exata.

G	E	F	Incremento	Lim. Sup.	Lim. Inf.	Normalizado	Erro Abs.
V	V	V	0.029592	0.996712	0.029592	0.9	0.380436
V	V	F	0.10464	0.839752	0.134232	0.455827	0.063737
V	F	V	0.04356	0.796192	0.177792	0.465912	0.053652
V	F	F	0.15288	0.730672	0.330672	0.55112	0.031556
F	V	V	0.128844	0.644776	0.459516	0.564003	0.044439
F	V	F	0.009765	0.621991	0.469281	0.553861	0.034297
F	F	V	0.043578	0.520309	0.512859	0.516708	0.002856
F	F	F	0.006705	0.519564	0.519564	0.519564	0.0

Tabela 7: Resultados de inferências durante a fase de execução. Os valores indicados representam a probabilidade de $H = \bar{h}$.

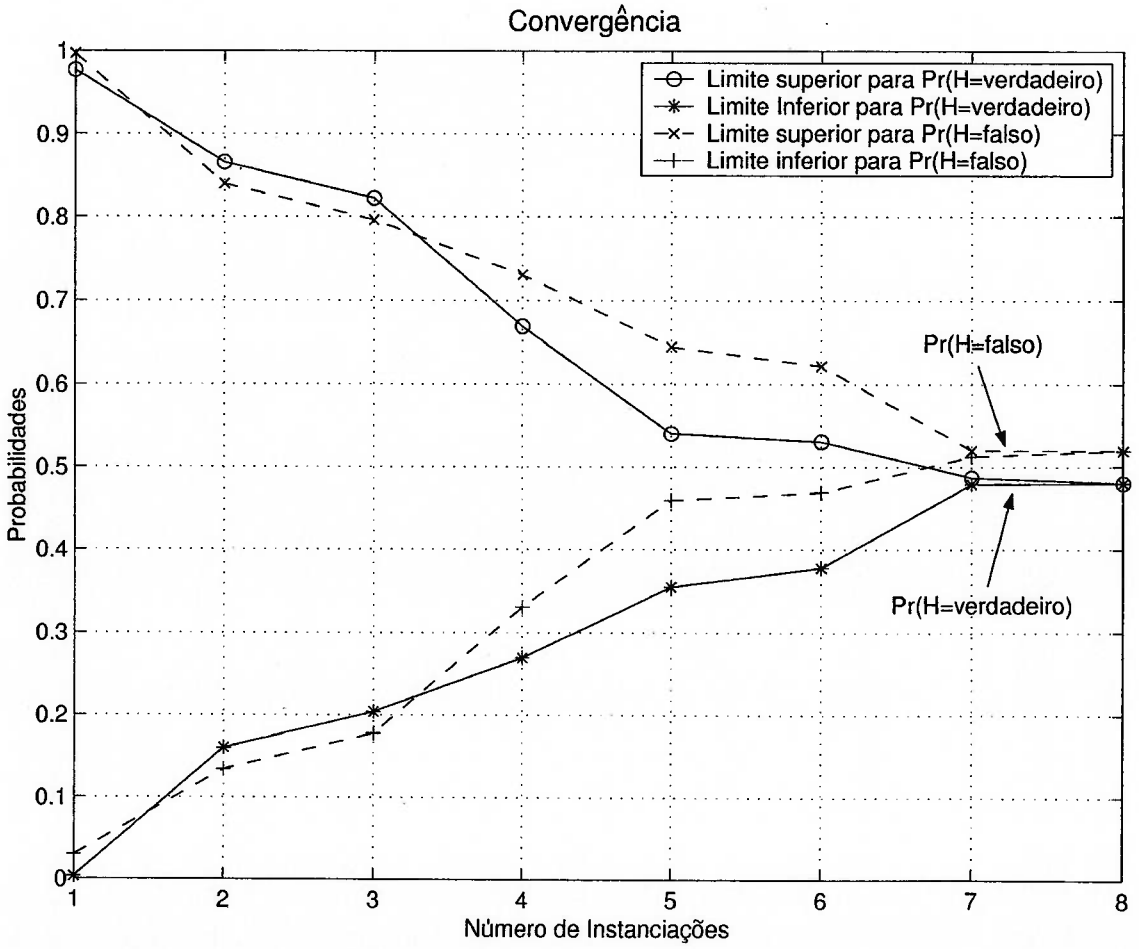


Figura 16: Convergência dos limites para as probabilidades $Pr(H = verdadeiro)$ e $Pr(H = falso)$.

4 RESULTADOS EXPERIMENTAIS

Nesta seção serão apresentados alguns resultados experimentais com redes Bayesianas reais. Os resultados foram obtidos não apenas nas condições de pior caso mas em várias configurações, indicando o desempenho do condicionamento adaptativo em várias situações. As redes foram testadas com diferentes configurações de memória e tempo e os resultados foram plotados em um gráfico de três dimensões. O microcomputador utilizado foi um Pentium IV, 1.7 GHz, RAM 1 Gb, rodando Linux kernel 2.4.7-10. A implementação dos algoritmos foi feita em Java e utilizou-se a máquina virtual da SUN, versão 1.3.1_01 para executar os testes.

4.1 Testes com Redes Bayesianas

A primeira análise foi feita usando a rede Alarm (BEINLICH, 1989) representada na Fig.17. Os testes foram realizados com restrições de memória variando de 3 até 24 para o tamanho do *separator*, e restrições de tempo variando de 1000 ms até 30000 ms. Para obter a quantidade de memória utilizada em *bytes*, basta multiplicar o tamanho do *separator* por 4, já que cada valor *double* ocupa 4 bytes na implementação em Java. Na Fig.18 foi plotado o gráfico 3D com os eixos representando Qualidade×Espaço×Tempo, sendo a qualidade apresentada através da largura dos intervalos entre os limites superiores e inferiores das respostas. As inferências foram realizadas para os valores das probabilidades marginais da variável "BP" pois esta é a variável que apresenta o menor número de variáveis d-separadas, o que torna a inferência mais complexa.

A Fig.19 representa outro gráfico Qualidade×Espaço×Tempo porém a qualidade é calculada de forma diferente. Utilizou-se o valor dos limites inferiores para normalizá-los e então obter uma resposta aproximada. O valor indicado no gráfico representa o erro absoluto entre a resposta da inferência exata e a resposta fornecida, depois de normalizada.

A segunda rede testada com o algoritmo de condicionamento adaptativo foi a rede Link (JENSEN; KONG, 1996) representada na Fig.20. Esta rede possui 724 nós sendo quase todos binários e suas ligações representam relações de causa/conseqüência entre

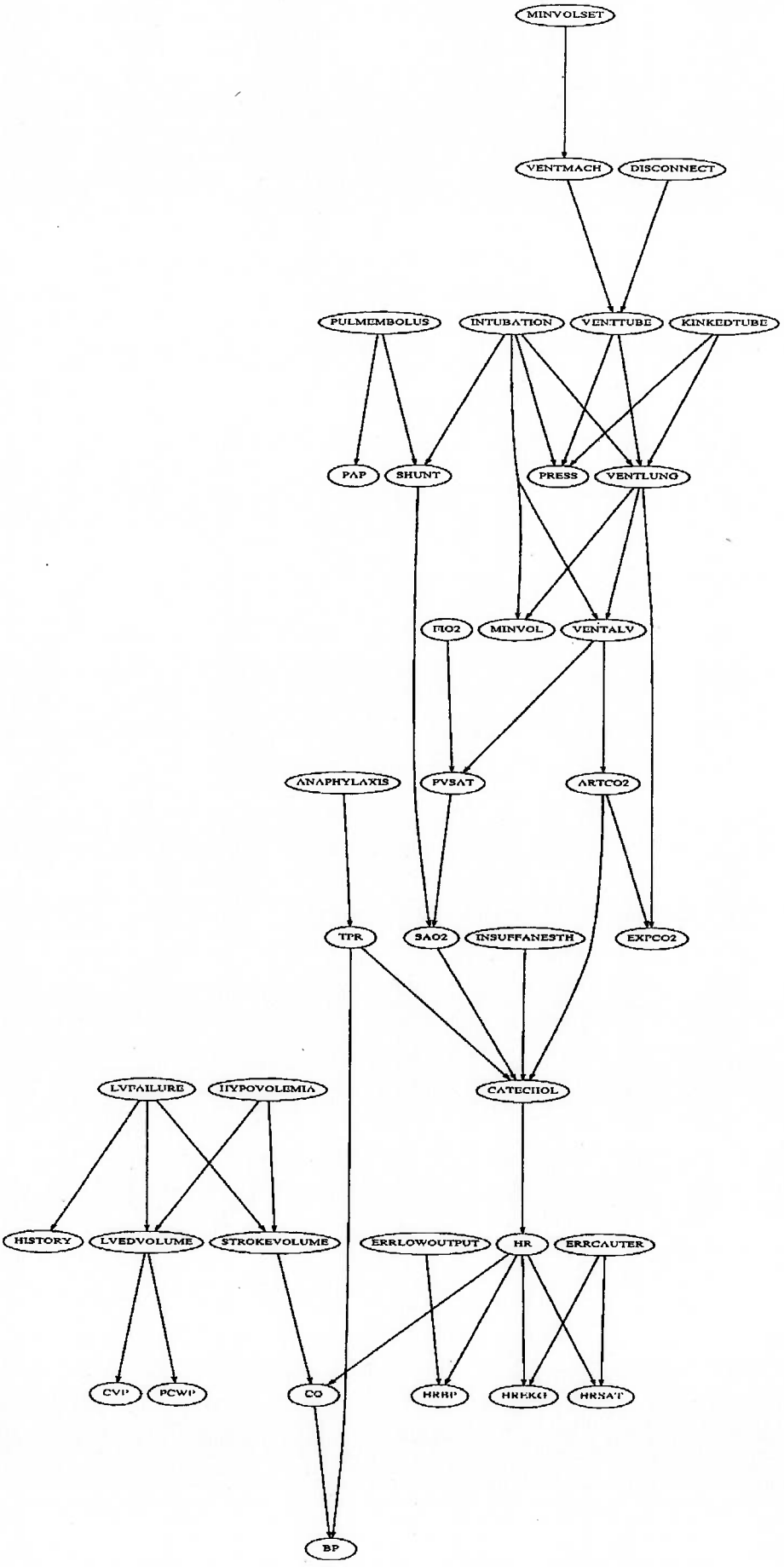


Figura 17: Rede Alarm.

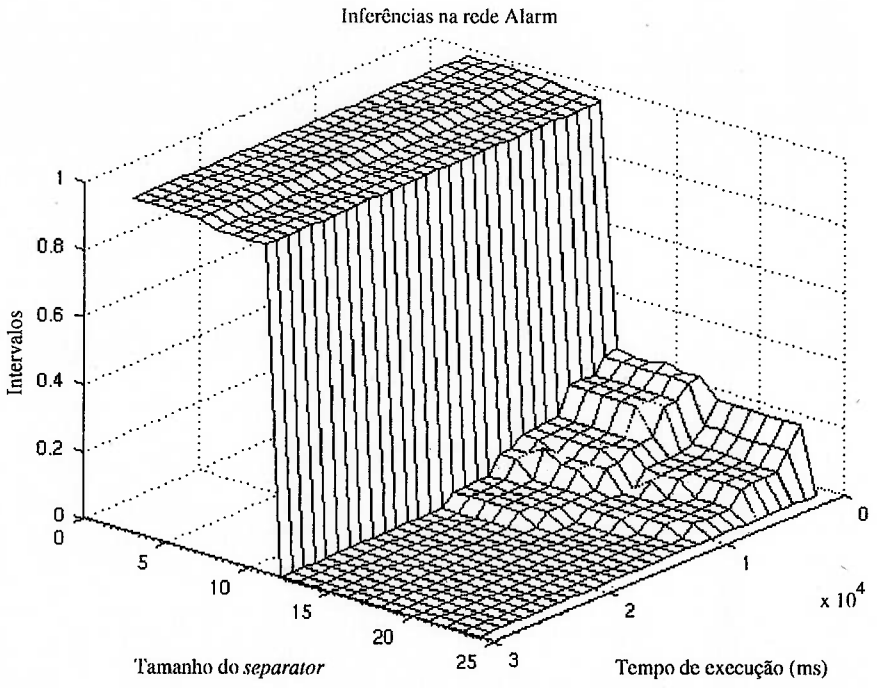


Figura 18: Intervalos de Variação para Inferências na rede Alarm.

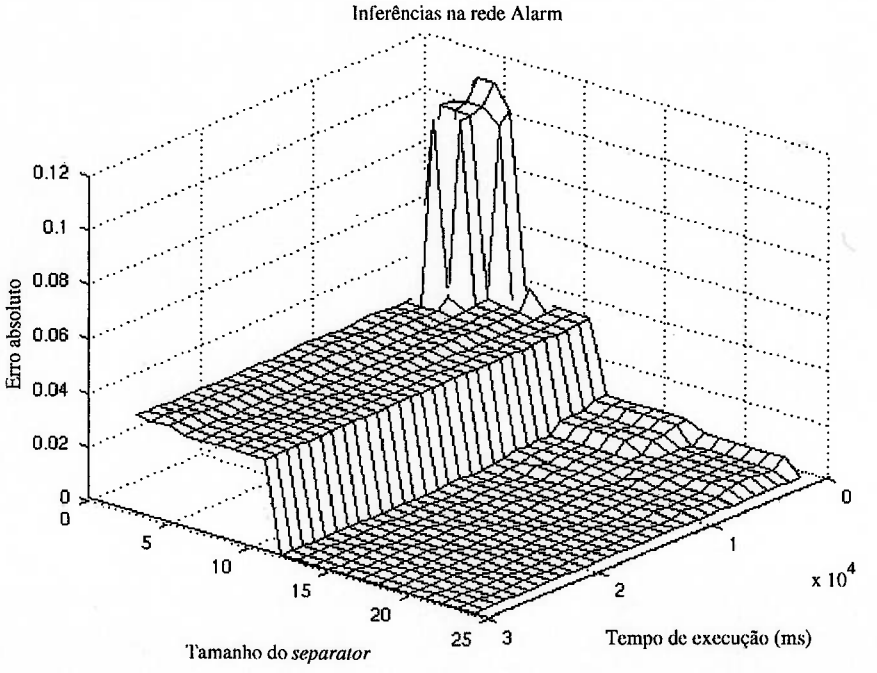


Figura 19: Erro absoluto para inferências na rede Alarm.

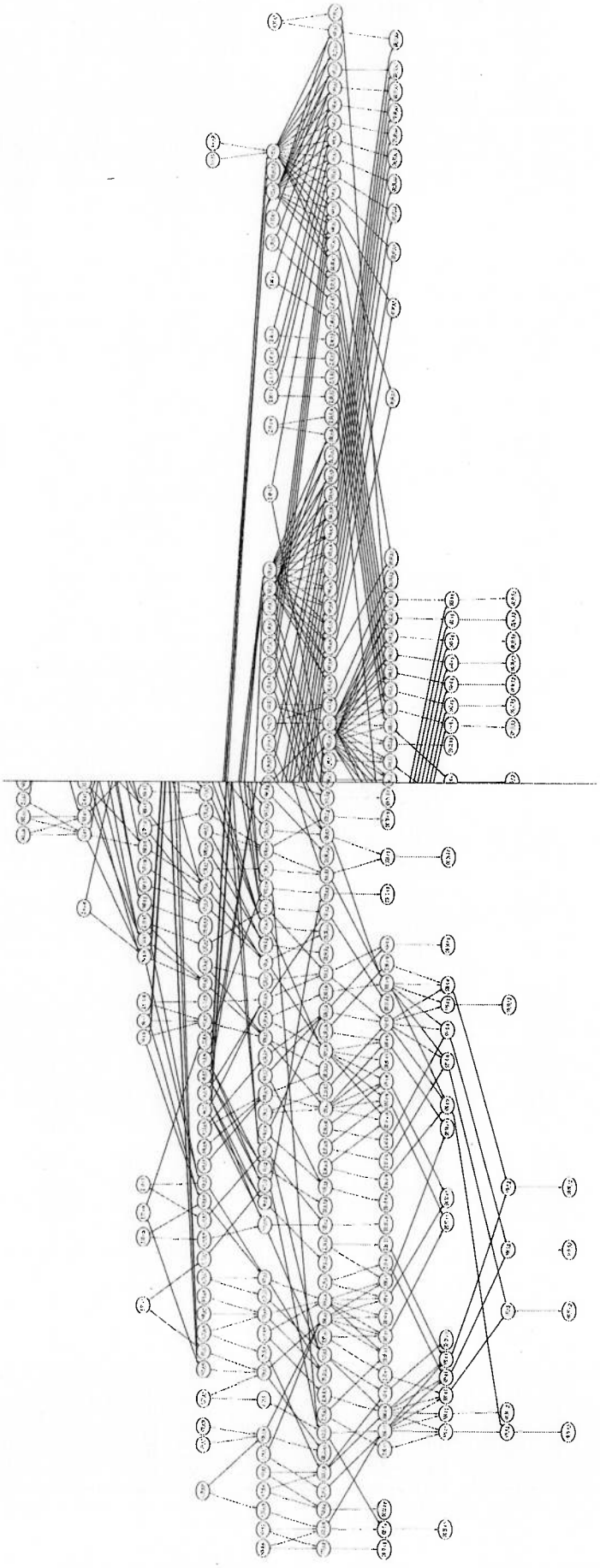


Figura 20: Partes da rede Link. Por motivos de espaço, estão representados dois cortes que mostram a estrutura fundamental da rede.

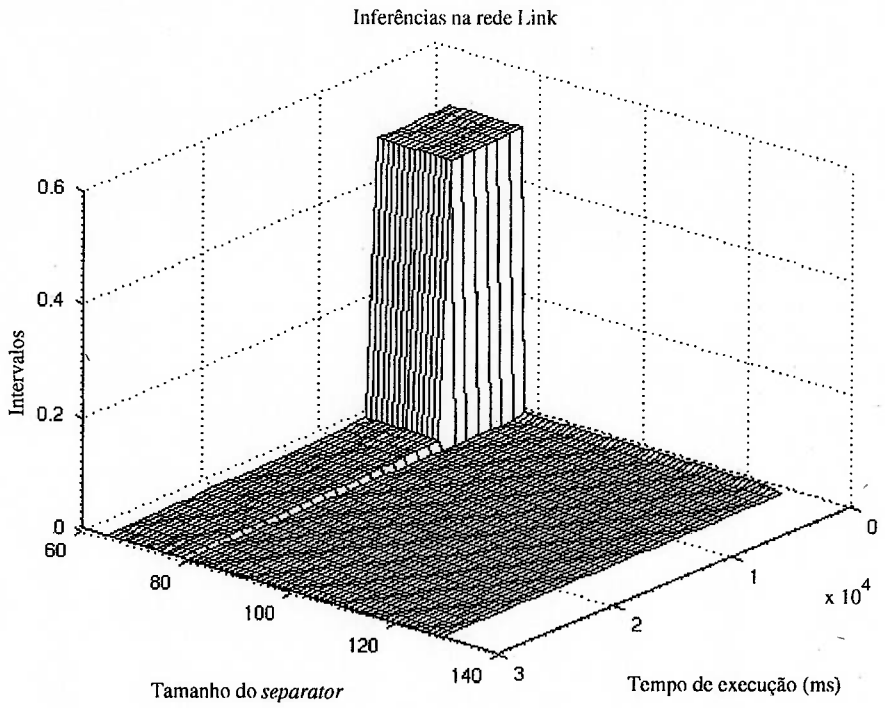


Figura 21: Intervalo de variação da resposta para a rede Link.

genes. Os testes consistiram de várias inferências feitas com diferentes limites para o valor máximo do *separator* e vários intervalos de tempo. A variação foi de 65 até 129 para os *separators* e 1000 até 30000 ms para o tempo. Assim como nos testes com a rede Alarm, a Fig.21 representa os intervalos de variação para avaliação da qualidade da resposta e a Fig.22 representa o erro absoluto como o padrão de medida da qualidade. A variável de interesse foi a variável "D0_56_d_p". Ela foi escolhida porque depende de várias outras variáveis da rede, fazendo com que o cálculo seja mais complexo.

4.2 Testes com Redes Bayesianas Dinâmicas (RBDs)

Para demonstrar as potencialidades do novo algoritmo, também foram realizados testes com redes Bayesianas dinâmicas. As redes Bayesianas dinâmicas são extensões das redes Bayesianas normais e caracterizam-se por representarem estados que se modificam com o tempo. São bastante utilizadas em modelos temporais que envolvem HMMs (*Hidden Markov Models*) para tarefas como reconhecimento de voz, navegação, reconhecimento de padrões em seqüências de ADN (ácido desoxirribonucléico), etc. Devido à sua simplicidade, as redes Bayesianas dinâmicas constituem uma importante ferramenta para representação de modelos probabilísticos temporais em inteligência artificial

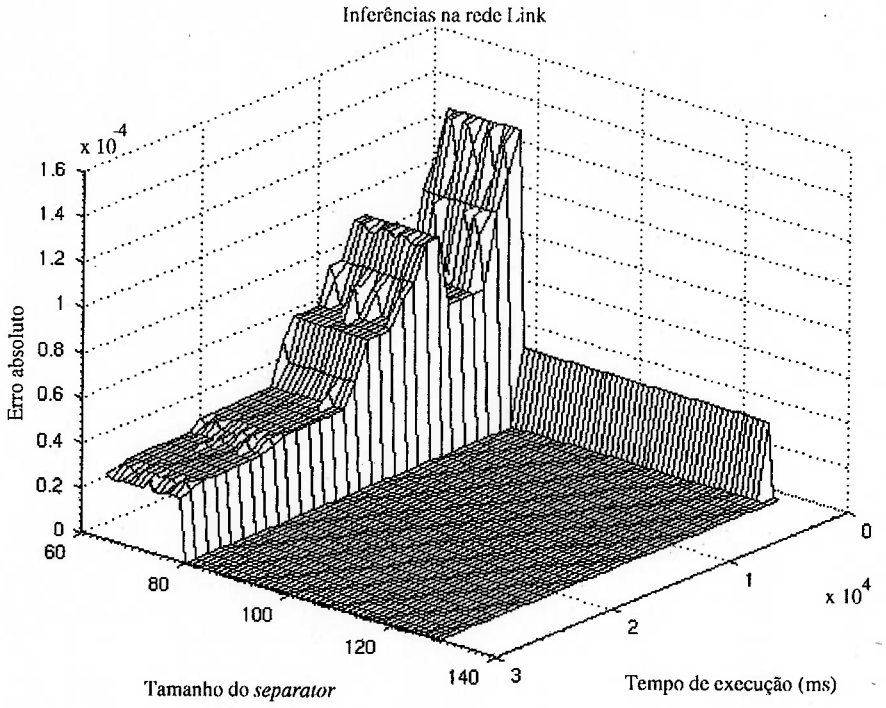


Figura 22: Erro absoluto para a rede Link.

(DEAN; KANAZAWA, 1989; MURPHY, 2002; RUSSELL; NORVIG, 2003).

Dados diversos conjuntos de variáveis aleatórias $\mathbf{Z}_1, \mathbf{Z}_2, \ldots$, uma rede Bayesiana dinâmica é definida como um par $(\mathcal{N}_1, \mathcal{N}_{\rightarrow})$, onde $\mathcal{N}_1$ representa o conhecimento *prior*, $Pr(\mathbf{Z}_1)$, e $\mathcal{N}_{\rightarrow}$ é a estrutura que se repete ao longo do tempo a qual define $Pr(\mathbf{Z}_t|\mathbf{Z}_{t-1})$. Considerando esta como um grafo direcionado acíclico:

$$Pr(\mathbf{Z}_t|\mathbf{Z}_{t-1}) = \prod_{i=1}^{N} Pr(\mathbf{Z}_t^i|pa(\mathbf{Z}_t^i)) \tag{4.1}$$

onde $\mathbf{Z}_t^i$ é o i-ésimo nó no tempo t e $pa(\mathbf{Z}_t^i)$ representam os pais de $\mathbf{Z}_t^i$ no grafo.

Inferências em RBDs podem ser realizadas *on-line,* ou seja, ao mesmo tempo em que novos estados são adicionados à estrutura (geralmente deseja-se obter probabilidades marginais das variáveis do estado-atual dado o passado), ou *off-line*, onde o objetivo é obter probabilidades marginais em qualquer período com a rede previamente definida e fixada. Assim, nos processos de inferência *off-line*, as RBDs podem ser consideradas como RBs convencionais.

Neste teste experimental, serão realizadas inferências com a rede Diabetes (ANDREASSEN, 1991), a qual pode ser visualizada na Fig.23 . Essa rede é formada por

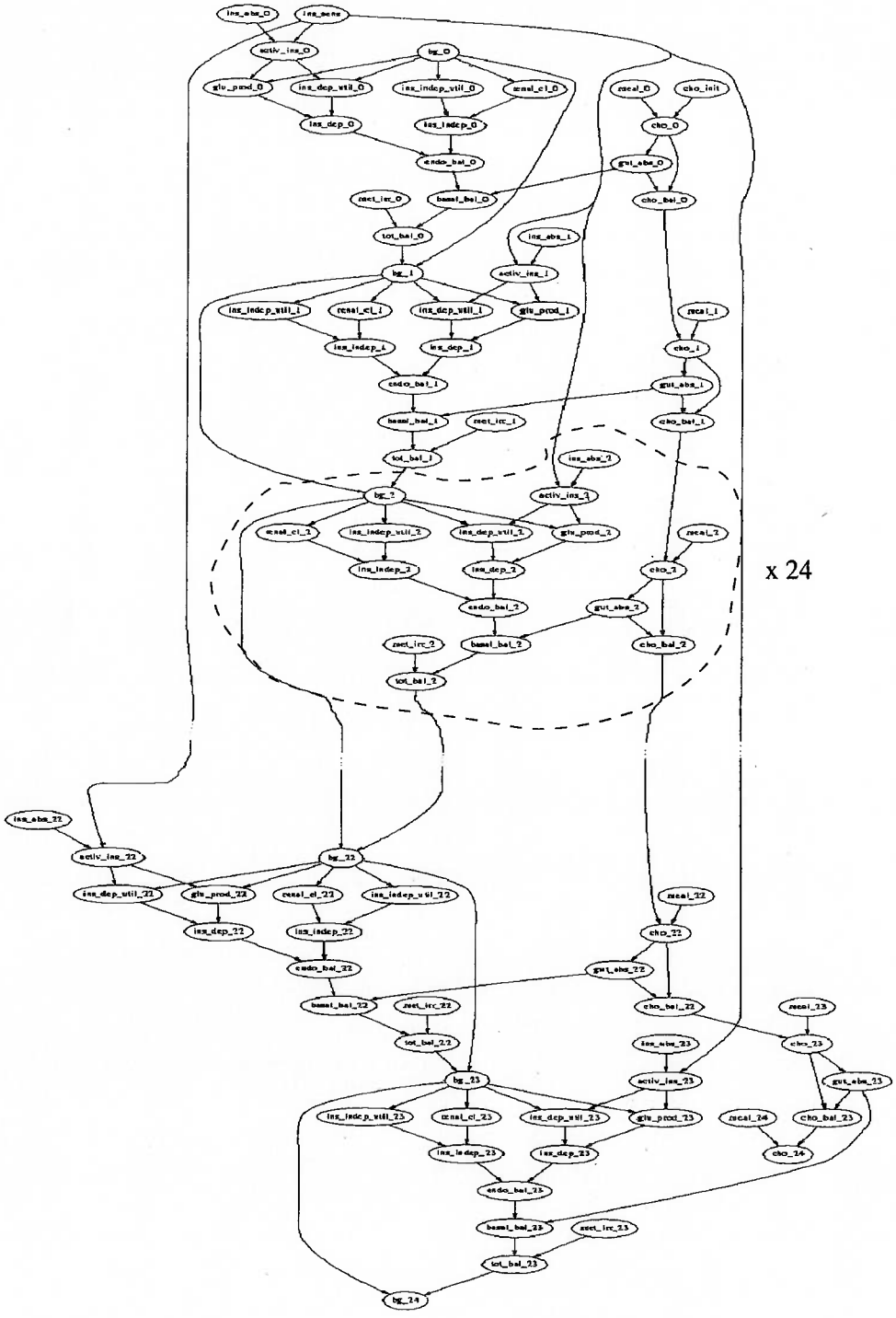


Figura 23: Rede Bayesiana Dinâmica Diabetes. Por motivo de espaço, apenas estão indicadas as etapas iniciais e finais da rede que tem 24 expansões no tempo. O conjunto de variáveis que se repete está indicado pela linha tracejada.

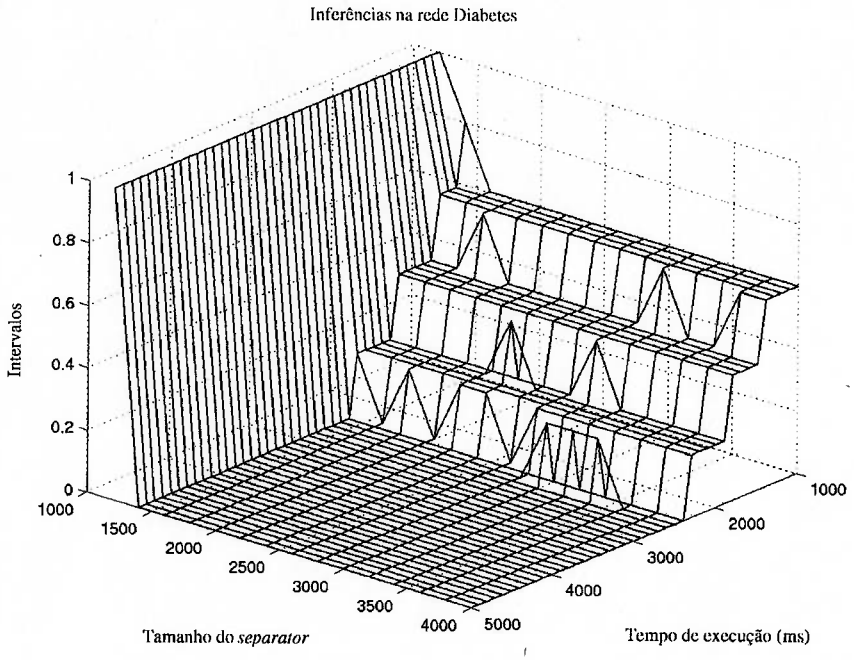


Figura 24: Intervalo de variação para inferências com a rede Diabetes.

uma expansão de 24 estruturas iguais à indicada pela linha tracejada, cada uma com 17 variáveis. Com esta configuração *off-line*, deseja-se obter a probabilidade marginal da variável mais abaixo da rede, a variável "bg_24" (esta variável é a única do estado 24 na Fig. 23). A grande dificuldade para se realizar inferências exatas nesta rede utilizando métodos convencionais como o de eliminação de variáveis, está no fato de que o espaço requerido para armazenar *separators* é muito grande. Na rede em questão, considerando que cada variável tem em média 6 estados, o método de eliminação de variáveis requer uma quantidade de memória suficiente para armazenar 6^{64} valores. Isso, evidentemente, inviabiliza a utilização de métodos de eliminação. Já com o condicionamento adaptativo, o espaço requerido pode ser configurado conforme a disponibilidade e inferências exatas podem ser computadas em menos de 3 segundos. Em testes, a quantidade de memória disponível variou de 1000 até 4000 valores (ou seja, de 4 até 16 Kbytes) e os tempos de inferência de 1000 até 5000ms. A decomposição encontrada eliminou uma dependência que fazia com que os *separators* fossem muito grandes. Condicionando-se a variável "ins_sens" (no alto da Fig. 23), os *separators* foram significativamente reduzidos, o que possibilitou realizar inferências com apenas 4 Kbytes de memória.

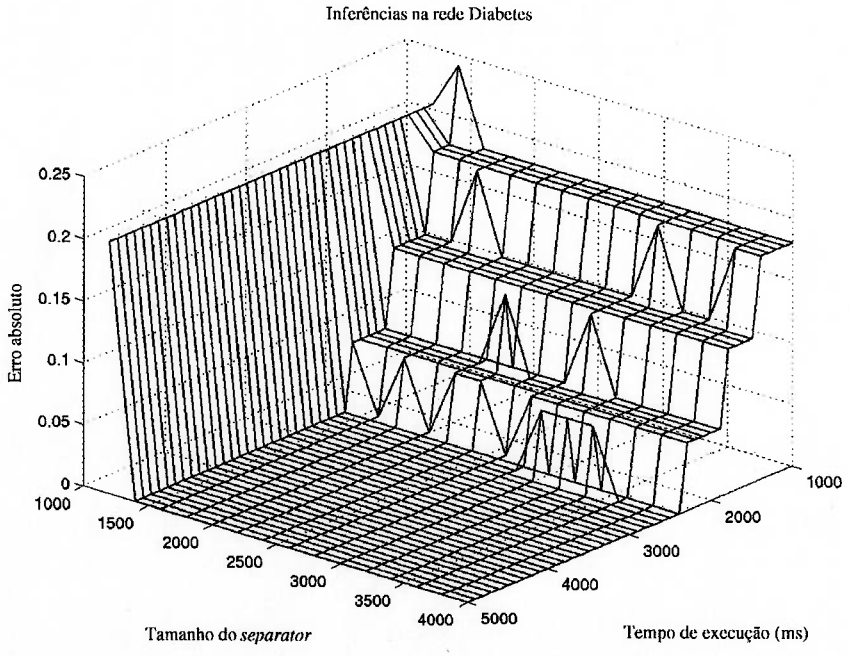


Figura 25: Erro absoluto para inferências na rede Diabetes.

5 DISCUSSÃO

Na seção anterior foram apresentados resultados experimentais para inferências feitas com o algoritmo de condicionamento adaptativo, utilizando redes reais. Nesta seção, estes resultados serão discutidos e comparados com resultados de outros algoritmos disponíveis.

5.1 Resultados Experimentais

Primeiramente, dado o grande número de possíveis configurações de memória e tempo, foram considerados na simulação não os piores casos, mas casos onde a memória disponível se aproxima de situações reais. Ou seja, seria inviável tentar processar uma rede do tamanho da rede Link (com 724 nós) com espaço suficiente para armazenar apenas a rede, já que o tempo de processamento seria muito longo para obter-se uma resposta razoável. Assim, em nossos testes com a rede Link, o tamanho mínimo do *separator* considerado foi de 65 e tempo mínimo de 1 segundo. Ademais, os limites superiores foram de 129 para o *separator* e 30 segundos para o tempo de processamento. Com relação à rede Alarm, uma rede bem menor que a Link, com apenas 37 nós, foram feitos processamentos com praticamente todas as possíveis configurações de tempo e memória. O menor *separator* para este caso foi de 3 e o menor tempo de 1 segundo. As restrições foram de 25 para o tamanho do *separator* e 30 segundos para o tempo. Para *separators* maiores do que 25, a rede não precisa ser decomposta e pode-se utilizar o algoritmo de eliminação de variáveis diretamente.

O primeiro aspecto interessante que pôde ser observado dos gráficos foi que, para alguns tamanhos de *separator*, o desempenho do algoritmo degrada abruptamente em comparação a valores maiores. Isto pode ser percebido nos gráficos como grandes degraus entre dois valores de *separator* próximos, resultado de duas divisões distintas, com tamanhos de *cutset* diferentes. Em outras palavras, na primeira situação em que o sistema ainda apresenta um desempenho satisfatório, o método de decomposição conseguiu encontrar a divisão da rede condicionando poucas variáveis. Entretanto, caso o limite seja diminuído, a próxima solução precisa do condicionamento de muito mais variáveis, o

que degrada o desempenho do sistema, já que complexidade no tempo é exponencial no número de variáveis condicionadas.

Nas Figs.19 e 21 observa-se regiões onde o erro representado pelos intervalos na Fig.21 e erro absoluto na Fig.19 aumentam drasticamente. Estas regiões indicam configurações sob severas restrições tanto de tempo como de espaço que resultam em qualidades de resposta pouco apuradas. Essas regiões devem ser evitadas em aplicações reais, pois um pequeno aumento no limite de memória ou no tempo de processamento poderia levar a um aumento significativo do desempenho.

Além disso, pode-se também observar o efeito dos *caches* no processo de inferência. Na Fig.18 por exemplo, para *separators* entre 3 e 12, existe uma região onde o desempenho aumenta com a memória, mantendo-se o mesmo tempo. Como a decomposição da rede permanece a mesma para *separators* entre 3 e 12, todo o aumento de desempenho deve-se à alocação de *caches* no processo. Já para a rede Link, o efeito dos *caches* não é tão pronunciado como na rede Alarm. Isto acontece porque a decomposição da rede Link resulta em muito mais sub-redes do que na rede Alarm. Assim, o restante de memória disponível é suficiente apenas para que se aloque *caches* para poucas sub-redes, não representando uma melhora substancial no processo como um todo.

Nos testes com a RBD Diabetes, pode-se observar como o algoritmo de condicionamento adaptativo pode ser útil em redes de grande porte. Comparando-se a enorme quantidade de memória que seria necessária para realizar inferências com algoritmos exatos convencionais, com os meros 1500 bytes do condicionamento adaptativo, dão uma idéia dos benefícios deste novo método. Observando os gráficos das Figs.24 e 25, percebe-se que modificações nos tamanhos do *separator* de 1500 a 4000 não afeta a qualidade significativamente. Entretanto, para tamanhos de *separators* menores do que 1500, a qualidade degrada rapidamente. Assim, a região ótima para utilização do algoritmo é a partir desta restrição de memória.

5.2 Comparação com Algoritmos *Anytime*

Condicionamento adaptativo oferece uma série de vantagens sobre os algoritmos *anytime* existentes. Em relação aos algoritmos estocásticos e *loop propagation*, condicionamento adaptativo tem a vantagem de trabalhar com intervalos convergentes para as aproximações, o que não acontece nesses outros dois algoritmos. Ademais, de acordo

com os resultados experimentais, a convergência é bastante rápida, se comparada com algoritmos estocásticos convencionais como *Gibbs sampling*. Este fato pode ser observado mesmo em redes de grande porte, com severas restrições de memória. Considerando os algoritmos de busca e *bounded conditioning*, condicionamento adaptativo também oferece várias vantagens. Ao invés de realizar buscas na rede inteira, os métodos de busca implementados operam nas sub-redes, com as variáveis condicionadas, em uma estrutura previamente organizada. Ao contrário de *bounded conditioning*, o algoritmo pode utilizar diferentes configurações de memória para melhorar a convergência e, conseqüentemente, aproveitar melhor os recursos disponíveis.

Outra possibilidade é utilizar condicionamento adaptativo como um puro algoritmo *anytime*. Neste caso, a fase de planejamento deverá operar de maneira diferente, tentando obter uma divisão que torne a convergência a mais rápida possível. Uma solução seria a divisão da rede no maior *separator* de maneira que a complexidade das redes resultantes ficasse menor que a rede original. Dessa forma, as sub-redes seriam computadas mais rapidamente do que a rede original e os resultados somados na fase de execução. Entretanto, deve-se considerar o intervalo entre cada nova resposta como um fator importante. Quanto mais complexas forem as sub-redes, maiores serão os intervalos, porém menos instanciações precisarão ser realizadas para se chegar à resposta exata. Por outro lado, quanto menos complexas forem as sub-redes, os intervalos entre as respostas será menor, porém mais instanciações deverão ser computadas para a resposta exata. Essa solução de compromisso deve ser considerada no processo de decomposição da rede inicial.

5.3 Comparação com Algoritmos *Anyspace*

Por ser o algoritmo de maior similaridade, *recursive conditioning* será discutido nesta seção e comparado com o condicionamento adaptativo. Os dois algoritmos utilizam a técnica de condicionamento para dividir o problema em problemas menores, porém tanto a profundidade desta divisão como a maneira de se organizar a computação são diferentes.

Ao contrário do condicionamento adaptativo que divide a rede apenas quando necessário, *recursive conditioning* caracteriza-se por dividir recursivamente a rede inicial, até transformá-la em um conjunto de redes de uma só variável cada (DARWICHE, 2001). Além disso, condicionamento adaptativo permite a alocação de algoritmos diferentes

para cada uma das sub-redes criadas.

Outra diferença é que *recursive conditioning* organiza as variáveis, depois de separadas, em uma árvore binária denominada *dtree* [1] e condicionamento adaptativo não faz uso desta ferramenta. Este fato pode parecer, à primeira vista, que condicionamento adaptativo poderia melhorar seu desempenho, se utilizasse uma estrutura como esta já que, quando uma *dtree* está balanceada, o número de chamadas nas sub-redes seria exponencial e log n e não em n, onde n é o número de sub-redes. Porém, condicionamento adaptativo também precisa apresentar um comportamento *anytime* e esta característica seria perdida se a computação fosse organizada em uma árvore. Isto acontece porque em uma árvore, as variáveis condicionadas que dividem uma rede em duas precisam ser instanciadas e computadas seguindo uma determinada ordem e, apenas após todas elas serem somadas, pode-se obter uma resposta. Isso não acontece se todas as variáveis condicionadas estiverem livres para serem instanciadas, pois desta forma em qualquer momento pode-se computar as sub-redes, multiplicar os resultados e obter uma resposta (mesmo que esta seja aproximada). Quando um novo valor intermediário estiver disponível, este pode ser somado aos anteriores, a fim de obter-se uma resposta de melhor qualidade. Esta é exatamente a característica inerente ao comportamento *anytime*, ou seja, a resposta melhora conforme o tempo for passando.

Além disso, os dois algoritmos também são diferentes considerando-se as estritas restrições de memória. Não apenas é desejável realizar uma inferência *anytime*, mas também garantir que o processo aconteça sob estritas restrições de memória; conseqüentemente, condicionamento adaptativo preocupa-se com o tamanho máximo dos *separators*. Enquanto *recursive conditioning* sempre tenta encontrar uma decomposição que produza árvores balanceadas que garantam no pior caso complexidade $O(n \exp(w \log))$ no tempo, condicionamento adaptativo somente divide a rede, quando necessário, encontrando no pior caso complexidade $O(n \exp(n))$ no tempo (onde n é o número de variáveis e w é o tamanho da ordem de eliminação). Percebe-se que condicionamento adaptativo é mais restrito considerando limitações de memória e degrada para processamento em força-bruta[2] no pior caso onde a rede inicial é dividida em n redes de uma variável cada. Também com relação às funcionalidades estes algoritmos diferem. En-

[1] Uma *dtree* é uma árvore binária cheia que representa a estrutura de divisão feita por *recursive conditioning*.

[2] No processamento em força-bruta, as probabilidades marginais são obtidas a partir da multiplicação e soma das probabilidades das variáveis, pelo produto cartesiano de suas tabelas.

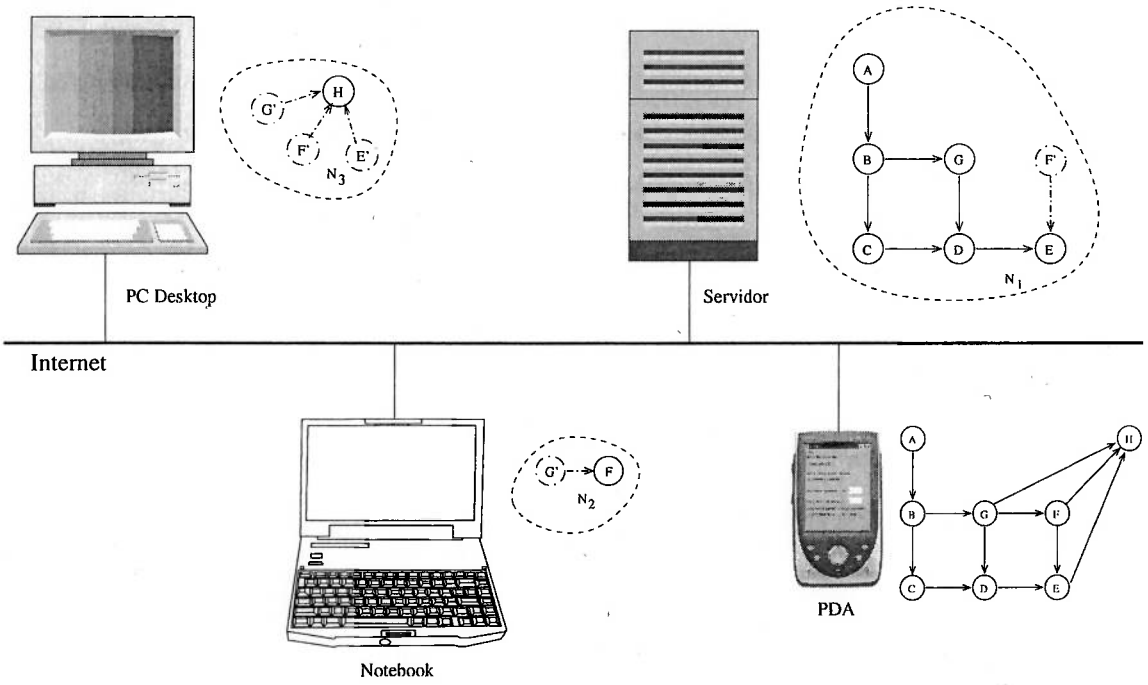


Figura 26: Esquema ilustrativo da paralelização do condicionamento adaptativo. Nesta figura o PDA mostra a interface gráfica do programa apresentado no Apêndice A.

quanto *recursive conditioning* faz inferências apenas para probabilidades de evidências, condicionamento adaptativo calcula também as probabilidades marginais para uma ou mais variáveis, o que aumenta o tempo de inferência dependendo do tamanho da tabela de probabilidades final.

5.4 Paralelização

Outra possibilidade é a implementação de condicionamento adaptativo visando sua paralelização. Processos de inferência paralelos têm sido tema de diversos trabalhos como em (KOZLOV; SINGH, 1996; PENNOCK, 1998; MADSEN; JENSEN, 1999). No condicionamento adaptativo, entretanto, pela própria concepção de dividir o problema inicial em sub-problemas mais simples, pode-se implementá-lo facilmente de forma paralela. Cada sub-rede pode ser enviada tanto para um outro processador de uma mesma máquina como para um outro computador conectado à mesma rede. Isto possibilita que um conjunto de sistemas computacionais possam formar uma comunidade, compartilhando e trocando recursos.

Na Fig. 26 está representado um esquema geral para implementação do condicionamento adaptativo em computadores conectados à Internet. Neste esquema, um PDA

deseja realizar uma inferência, porém não dispõe do poder computacional necessário para tal operação, no tempo designado. Este problema pode ser contornado utilizando uma implementação paralela do condicionamento adaptativo. Para a rede da Fig. 10, com a decomposição ilustrada na Fig. 14, pode-se organizar a computação enviando sub-redes para diferentes máquinas conectadas na mesma rede. As máquinas com poder computacional superior podem receber as sub-redes mais complexas. Assim, o servidor pode receber a maior sub-rede, N_1, seguido do computador de mesa com a sub-rede N_3 e do notebook com a N_2. Cada uma dessas máquinas realiza inferências nas respectivas sub-redes enviadas, cabendo ao PDA as tarefas de decompor e organizar a rede inicial, enviando as sub-redes para as outras máquinas que participarão do processo. Neste modelo, o PDA age como um mestre, sendo as outras máquinas suas escravas. Ele solicita as inferências nas sub-redes de cada máquina, especificando as instanciações a serem realizadas. Então, o PDA compõe os resultados multiplicando-os e somando-os, conforme descrito na fase de execução.

6 CONCLUSÃO

Neste trabalho foram apresentadas e discutidas diversas técnicas empregadas no processo de inferência probabilística com redes Bayesianas. Essas técnicas abordam diferentes maneiras de lidar com restrições de memória como também levam em consideração necessidades de tempo, sendo assim adequadas para aplicações em sistemas embarcados, robótica e inferências em redes de grande porte.

A principal contribuição deste trabalho é o algoritmo denominado condicionamento adaptativo. Este utiliza diversas técnicas para configurar seu comportamento com os recursos de memória e tempo disponíveis: decomposição da rede por condicionamento, classificação por importância através do método MMD, organização de estados de variação, alocação de *caches* e aproximação da resposta por limites. Além disso, condicionamento adaptativo pode também trabalhar com vários algoritmos de inferência ao mesmo tempo, processando redes diferentes. Esta característica torna-o ainda mais flexível e abre uma nova área de pesquisa em inferência Bayesiana, onde diversos algoritmos podem coexistir para processar a mesma rede. Assim, é possível agregar métodos de Monte Carlo com *clustering* e algoritmos de *polytrees* para resolver o mesmo problema, bastando que seja definido qual algoritmo é mais adequado para cada sub-problema.

De maneira geral, pode-se destacar as seguintes características do condicionamento adaptativo:

1. Possibilidade de trabalhar com restrições de tempo e memória, incorporando técnicas que permitem a máxima utilização desses recursos;
2. Combinação de técnicas de *clustering*, condicionamento e busca no mesmo processo de inferência;
3. Fácil de ser entendido e implementado, com teoremas simples baseados em condicionamento;
4. Explora diferentes soluções de compromisso entre tempo $\times$ memória $\times$ qualidade de resposta;
5. Implementação paralela simples dada sua estrutura de decomposição.

Os resultados experimentais apresentados aqui demonstram o potencial do condicionamento adaptativo com três redes reais: Alarm, Link e Diabetes. Para as duas primeiras, pôde-se observar o desempenho das inferências feitas em várias configurações diferentes de tempo e memória, a partir da qualidade do resultado obtido. No conjunto, o sistema apresentou uma resposta bastante satisfatória para a maioria das configurações. Entretanto, para algumas configurações onde os recursos eram mais limitados, houve uma abrupta queda de desempenho. Também foi possível observar regiões no gráfico onde existem suaves variações de desempenho para certas configurações e outras com grandes variações, possibilitando que o usuário escolha qual o melhor ponto para o operação do sistema, dado seus recursos. Para a rede Bayesiana dinâmica Diabetes, o algoritmo mostrou suas potencialidades ao computar inferências para uma variável do 24º estágio. Essa rede, dado seus requisitos de memória, seria inviável de ser computada usando um algoritmo de inferência exata tradicional. Com o condicionamento adaptativo, a mesma rede pode ser processada em poucos segundos, com 6 kbytes de memória. Este fato demonstra as potencialidades do algoritmo para trabalhar com redes Bayesianas dinâmicas, as quais são largamente empregadas em problemas de robótica como o SLAM (*Simultaneous Localization and Mapping*), problemas de genética que envolvem análise e decodificação de genes e diagnósticos médicos.

A implementação paralela aparece como outra possibilidade para a utilização do algoritmo. Conforme discutido no Capítulo 5, pode-se distribuir tarefas do mesmo processo de inferência para várias máquinas conectadas em rede. Da mesma forma, pode-se trabalhar com o condicionamento adaptativo em um sistema multi-agentes onde sub-redes podem ser trocadas, ou seja, dividem-se tarefas entre os agentes do sistema com o mesmo objetivo final.

Contudo, deve-se observar que embora o condicionamento adaptativo apresente novas técnicas para a inferência Bayesiana que tornam o processo muito mais flexível, algumas melhorias podem ainda ser propostas. Pelos gráficos experimentais percebe-se que o desempenho do sistema é altamente influenciado pela divisão da rede encontrada. Quanto mais variáveis forem condicionadas, pior será o desempenho do sistema. No método de divisão implementado, a rede é dividida em sub-redes para que o limite de memória seja atingido. Contudo, outros aspectos não são considerados, como por exemplo o número de variáveis que têm de ser condicionadas para aquela decomposição, já que, conforme mostra o Teorema 2, o número de inferências é exponencial no tamanho

do *cutset* (ou seja, no número de variáveis condicionadas). Assim, poder-se-ia adicionar à heurística implementada considerações sobre o número de variáveis condicionadas de forma que o sistema buscasse sempre a solução com o número mínimo de variáveis condicionadas.

Outra possibilidade é investigar a alocação de algoritmos, ou seja, avaliar qual seria o melhor algoritmo para cada sub-rede, considerando-se o desempenho do sistema como um todo. Neste caso, critérios de avaliação de algoritmos precisam ser criados para que, em conjunto com a análise das complexidades no tempo e no espaço, forneçam uma decisão mais apurada.

Novas maneiras de alocação e atualização dos *caches* também podem melhorar o desempenho de condicionamento adaptativo. Métodos de alocação dinâmica podem evitar ainda mais a repetição de cálculos já realizados. Neste caso, os *caches* seriam alocados durante a fase de execução observando as próximas instanciações a serem computadas. Outra alternativa é a utilização de *caches* fracionais, ou seja, *caches* que armazenam porções das probabilidades marginais das sub-redes. Desta forma, a memória restante poderia ser utilizada até o seu último *byte*, apesar da alocação e atualização tornar-se um processo bem mais complexo.

Por fim, deve-se destacar a flexibilidade que caracteriza o algoritmo proposto. Não apenas pode-se alocar diferentes algoritmos e máquinas para um mesmo problema, após este ser decomposto em sub-problemas, como também pode-se associar diferentes técnicas para o processo de organização e cálculo do resultado final. Assim, abre-se um novo tópico de pesquisa em inferência probabilística caracterizado pela coexistência de diversas técnicas atuando com o mesmo objetivo.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDREASSEN, S. et al. A model-based approach to insulin adjustment. In: STEFANELLI, M. et al. (Ed.). *Proceedings of the Third Conference on Artificial Intelligence in Medicine, Lecture Notes in Medical Informatics 44*. Berlin, Alemanha: Springer-Verlag, 1991. p. 239–248.

BEINLICH, I. et al. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Second European Conference on Artificial Intelligence in Medicine*, p. 247–256, 1989.

CANNINGS, C.; THOMPSON, E. A.; SKOLNICK, M. H. Probability functions in complex pedigrees. *Advances in Applied Probability*, v. 10, p. 26–61, 1978.

COOPER, G. F. *Bayesian belief-network inference using recursive decomposition*. CA 94305, 1990. (Relatório Técnico KSL-90-05).

COOPER, G. F. The computacional complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, v. 42, p. 393–405, 1990.

COWELL, R. G. et al. *Probabilistic Networks and Expert Systems*. Nova Iorque, Estados Unidos: Springer-Verlag, 1999.

COZMAN, F. G. Generalizing variable elimination in Bayesian networks. In: *Workshop on Probabilistic Reasoning in Artificial Intelligence*. São Paulo: Tec Art, 2000. p. 27–32.

DAGUM, P.; LUBY, M. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, v. 60, p. 141–153, 1993.

D'AMBROSIO, B. Incremental probabilistic inference. In: *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*. Washington, Estados Unidos: Morgan Kaufmann, 1993. p. 301–308.

DARWICHE, A. Conditioning methods for exact and approximate inference in causal networks. In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. São Francisco, Estados Unidos: Morgan Kaufmann, 1995. p. 99–107.

DARWICHE, A. Recursive conditioning. *Artificial Intelligence*, v. 126(1-2), p. 5–41, Fevereiro 2001.

DARWICHE, A.; PROVAN, G. Query DAGs: A practical paradigm for implementing belief-network inference. In: *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*. São Francisco, Estados Unidos: Morgan Kaufmann, 1996. p. 203–210.

DAVIES, S. *Fast Factored Density Estimation and Compression with Bayesian Networks*. Tese (Doutorado) — School of Computer Science, Carnegie Mellon University, Maio 2002.

DEAN, T.; KANAZAWA, K. A model for reasoning about persistence and causation. *Artificial Intelligence*, v. 93(1-2), p. 1–27, 1989.

DEAN, T. L.; BODDY, M. An analysis of time-depedent planning. In: *Proceeding of Seventh National Conference on Artificial Intelligence*. Menlo Park, Estados Unidos: AAAI Press/The MIT Press, 1988. p. 49–54.

DECHTER, R. Bucket elimination: A unifying framework for probabilistic inference. In: *XII Uncertainty in Artificial Intelligence Conference*. São Francisco, Estados Unidos: Morgan Kaufmann, 1996. p. 211–219.

DECHTER, R. Topological parameters for time-space tradeoff. In: *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*. São Francisco, Estados Unidos: Morgan Kaufmann, 1996. p. 220–227.

DECHTER, R. Mini-buckets: A general scheme for generating approximations in automated reasoning in probabilistic inference. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. Nagoya, Japão: Morgan Kaufmann, 1997. p. 1297–1302.

DWARKADAS, S. et al. Parallelization of general linkage analysis problems. *Human Heredity*, v. 44, p. 127–141, 1994.

FISHMAN, G. S. *Monte Carlo: Concepts, Algorithms, and Applications*. Nova Iorque, Estados Unidos: Springer-Verlag, 1995.

GARVEY, A. J. *Design-to-time Real-time scheduling*. Tese (Doutorado) — University of Massachusetts Amherst, Department of Computer Science, Fevereiro 1996.

GEMAN, S.; GEMAN, D. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, n. 6(6), p. 721–741, Novembro 1984.

GUO, H.; HSU, W. A survey of algorithms for real-time Bayesian network inference. In: *AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. Edmonton, Canadá: AAAI Press, 2002. p. 1–12.

HENRION, M. Search-based methods to bound diagnostic probabilities in very large belief nets. In: *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*. Los Angeles, Estados Unidos: Morgan Kaufmann, 1991. p. 142–150.

HORVITZ, E.; BARRY, M. Display of information for time-critical decision making. In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Montreal, Canadá: Morgan Kaufmann, 1995. p. 286–305.

HORVITZ, E. et al. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Madison, Estados Unidos: Morgan Kaufmann: São Francisco, 1998. p. 256–265.

HORVITZ, E.; SUERMONDT, H. J.; COOPER, G. F. Bounded conditioning: Flexible inference for decisions under scarce resources. In: *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*. Windsor, Canadá: Morgan Kaufmann, 1989. p. 182–193.

JENSEN, C. S.; KONG, A. *Blocking Gibbs Sampling for Linkage Analysis in Large Pedigrees with Many Loops*. Fredrik Bajers Vej 7, DK-9220 Aalborg Ø, 1996. (Relatório Técnico R-96-2048).

JENSEN, F. *Juntion Trees and decomposable hypergraphs*. Dinamarca, 1988. (Relatório Técnico Judex Datasystemer A/S).

JENSEN, F. V.; JENSEN, F. Optimal juntion trees. In: *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Seattle, Estados Unidos: Morgan Kaufmann, 1994. p. 360–366.

JENSEN, F. V.; LAURITZEN, S.; OLESEN, K. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, n. 4, p. 269–282, 1990.

KIM, J. H.; PEARL, J. A computational model for causal and diagnostic reasoning in inference engines. In: *Proceedings of the 8th International Joint Conference on Artificial Intelligence*. Karlsruhe, Alemanha Ocidental: Morgan Kaufmann, 1983. p. 190–193.

KJAERULFF, U. *Triangulation of Graphs — Algorithms Giving Small Total State Space*. Dinamarca, Março 1990. (Relatório Técnico R-90-09).

KJAERULFF, U. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, n. 2, p. 7–17, 1992.

KJAERULFF, U. *Approximation of Bayesian networks through edge removals*. Dinamarca, 1993. (Relatório Técnico).

KJAERULFF, U. *Reduction of Computational Complexity in Bayesian Networks through Removal of Weak Dependencies*. Dinamarca, Fevereiro 1994. (Relatório Técnico R94-2009).

KOZLOV, A. V.; SINGH, J. P. Parallel implementations of probabilistic inference. *Computer*, v. 29, n. 12, p. 33–40, Dezembro 1996.

KWOK, C.; FOX, D.; MEILA, M. Real-time particle filters using mixtures of samples sets. In: *AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. Edmonton, Canadá: AAAI Press, 2002. p. 20–27.

LAURITZEN, S. L.; SPIEGEHALTER, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistics Society*, Series B, 50(2), p. 157–224, 1988.

MADSEN, A. L.; JENSEN, F. V. *Parallelization of Inference in Bayesian Networks*. Dinamarca, 1999. (Relatório Técnico DK-9220).

MAJUMDER, S.; SCHEDING, S.; DURRANT-WHYTE, H. Multi sensor data fusion for underwater navigation. *Robotics and Autonomous Systems*, v. 35, p. 97–108, 2001.

MONTI, S.; COOPER, G. F. Bounded recursive decomposition: a search-based method for belief network inference under limited resources. *International Journal of Approximate Reasoning*, v. 15(1), p. 49–75, 1996.

MURPHY, K. *Learning Switching Kalman Filter Models*. Cambridge, Estados Unidos, 1998. (Relatório Técnico 98-10).

MURPHY, K. P. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Tese (Doutorado) — Department of Computer Science, University of California, Berkeley, Setembro 2002.

MURPHY, K. P.; WEISS, Y.; JORDAN, M. I. Loopy belief propagation for approximate inference: An empirical study. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Stockholm, Suécia: Morgan Kaufmann, 1999. p. 467–475.

PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. São Mateo, Estados Unidos: Morgan Kaufmann, 1988.

PENNOCK, D. M. Logarithmic time parallel Bayesian inference. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Madison, Estados Unidos: Morgan Kaufmann, 1998. p. 431–438.

PEOT, M. A.; SHACHTER, R. D. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence*, v. 48(3), p. 299–318, 1991.

RAMOS, F. T.; COZMAN, F. G.; IDE, J. S. Embedded Bayesian networks: Anytime anyspace inference. In: *AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. Edmonton, Canadá: AAAI Press, 2002. p. 13–19.

RAMOS, F. T.; MIKAMI, F.; COZMAN, F. G. Implementação de redes Bayesianas em sistemas embarcados. In: *Proceedings of the IBERAMIA/SBIA 2000 Workshops (Workshop on Probabilistic Reasoning in Artificial Intelligence)*. Atibaia, Brasil: Editora Tec Art, 2000. p. 65–69.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Segunda ed. New Jersey, Estados Unidos: Prentice Hall, 2003.

SHACHTER, R. D.; ANDERSEN, S. K.; SZOLOVITS, P. Global conditioning for probabilistic inference in belief networks. In: *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Seattle, Estados Unidos: Morgan Kaufmann, 1994. p. 514–522.

VEGAS, F. J. D. *Local conditioning in Bayesian networks*. Los Angeles, Estados Unidos, 1992. (Relatório Técnico R-181).

WEISS, Y.; FREEMAN, W. T. *Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology*. Berkeley, Estados Unidos, 1999. (Relatório Técnico CSD-99-1046).

WELLMAN, M. P.; LIU, C. L. State-space abstration for anytime evaluation of probabilistic networks. In: *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Seattle, Estados Unidos: Morgan Kaufmann, 1994. p. 567–574.

WEN, W. X. Optimal decomposition of belief networks. In: *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*. Nova Iorque, Estados Unidos: Elsevier Science Publishing Company,Inc, 1991. p. 209–224.

ZWEIG, G.; RUSSELL, S. J. Speech recognition with dynamic Bayesian networks. In: *Proceedings of the 15th National Conference on Artificial IntelligenceAAAI/IAAI*. Madison, Estados Unidos: AAAI Press, 1998. p. 173–180. Disponível em: <citeseer.nj.nec.com/zweig98speech.html>.

APÊNDICE A - IMPLEMENTAÇÃO EM JAVA

Neste apêndice são apresentadas as principais telas da interface gráfica construída na implementação do algoritmo de condicionamento adaptativo. A implementação foi feita em Java e o sistema foi testado em um Pocket PC (HP Jornada 568), com a máquina virtual Personal Java para Windows CE. As telas foram construídas utilizando-se a biblioteca AWT pois esta biblioteca pode ser utilizada na maioria das máquinas virtuais. Ao contrário da biblioteca Swing, a AWT pode ser utilizada em máquinas virtuais para processadores StrongARM, SH3 e MIPS.

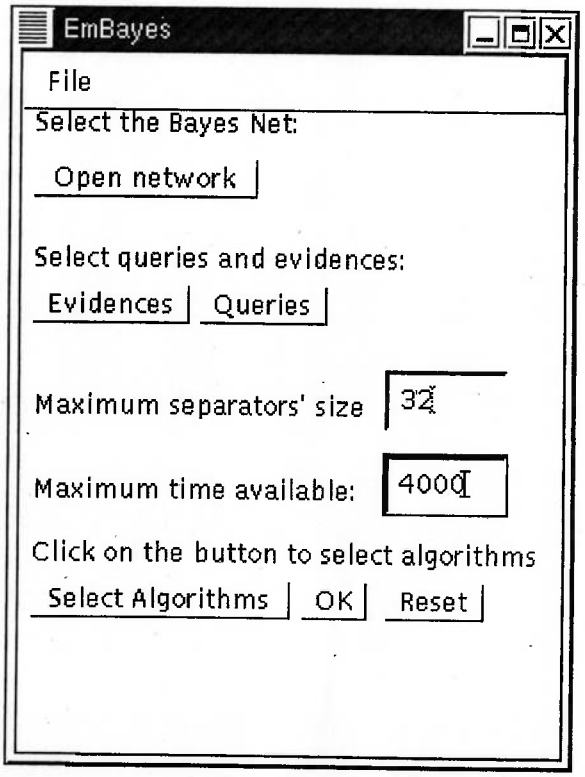


Figura 1: Tela inicial do condicionamento adaptativo.

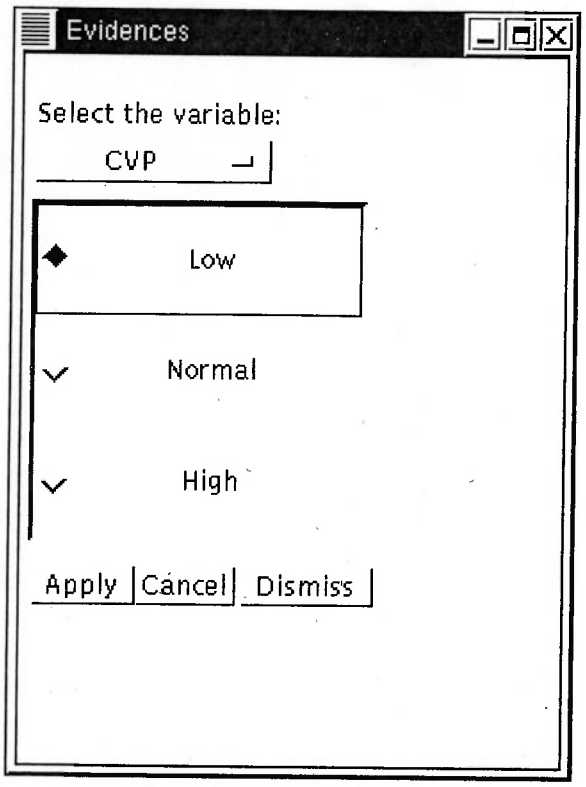


Figura 2: Tela para realizar observações na rede carregada.

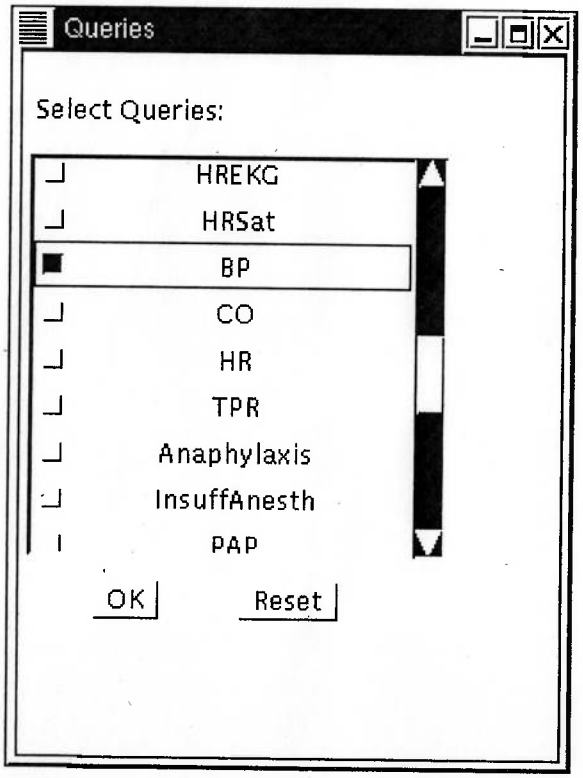


Figura 3: Tela para seleção de variáveis de interesse.

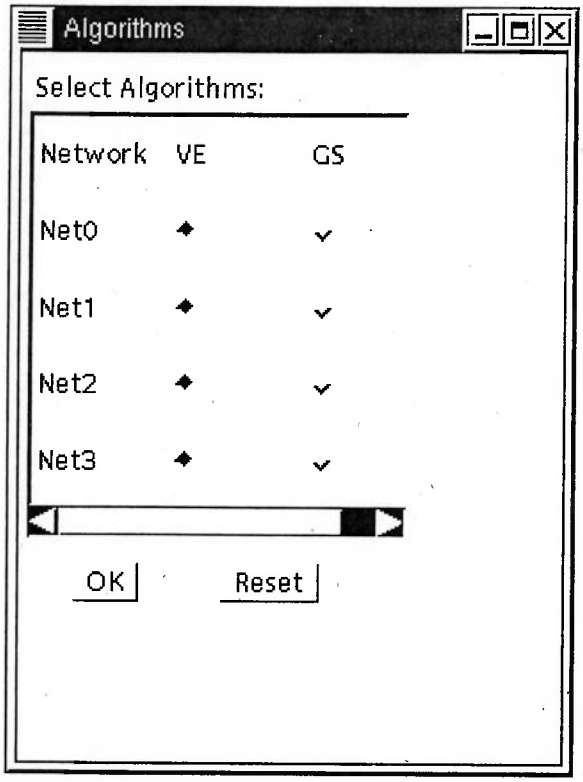


Figura 4: Tela para seleção de algoritmos para as sub-redes formadas. VE = eliminação de variáveis e GS = *Gibbs sampling*.

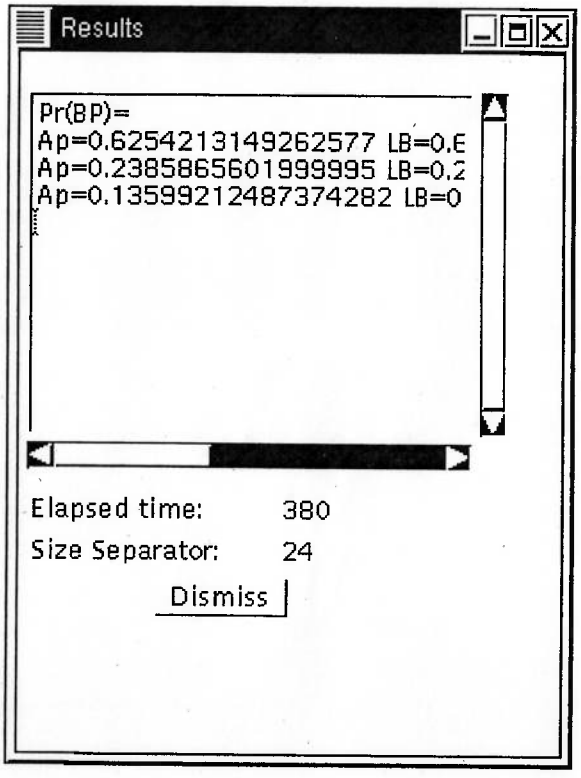


Figura 5: Tela para visualização dos resultados.