ALGORITMO DE ESCALONAMENTO UTILIZANDO MODELO DE ISING COM CAMPO EXTERNO

Nilton Guedes Duarte

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: José Ferreira de Rezende

Rio de Janeiro
Março de 2019

ALGORITMO DE ESCALONAMENTO UTILIZANDO MODELO DE ISING COM CAMPO EXTERNO

Nilton Guedes Duarte

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. José Ferreira de Rezende, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Diego Gimenez Passos, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2019

Duarte, Nilton Guedes

Algoritmo de escalonamento utilizando modelo de Ising com campo externo/Nilton Guedes Duarte. – Rio de Janeiro: UFRJ/COPPE, 2019.

XIII, 57 p.: il.; 29, 7cm.

Orientador: José Ferreira de Rezende

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 48 – 50.

1. redes de computadores. 2. redes sem fio. 3. modelo de ising. I. Rezende, José Ferreira de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À família.

Agradecimentos

Agradeço a minha família, que sempre me apoiou. Agradeço aos meus amigos pela ajuda sempre que solicitada. Agradeço à Lygia Marina, pelas incontáveis horas de conversas acadêmicas, com a paciência em me transmitir todo o seu conhecimento.

Agradeço ao meu orientador, José Rezende, pelo tempo investido na minha formação e no desenvolvimento deste projeto.

Obrigado à banca examinadora por ter aceito o convite e se dispor a participar da defesa desta dissertação

Agradeço à UFRJ pelo conhecimento transmitido.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMO DE ESCALONAMENTO UTILIZANDO MODELO DE ISING COM CAMPO EXTERNO

Nilton Guedes Duarte

Março/2019

Orientador: José Ferreira de Rezende

Programa: Engenharia de Sistemas e Computação

Algoritmos de acesso ao meio compartilhado para redes sem fio que levam em conta o tamanho da fila dos enlaces ganharam destaque recentemente. Neste artigo, propomos um algoritmo distribuído de escalonamento de enlaces baseado no modelo físico de Ising para o antiferromagnetismo. Esse modelo foi adaptado, por outro trabalho, para considerar o tamanho das filas e utilizar a dinâmica de Glauber para minimizar a energia capturada pelo modelo. Neste trabalho, propomos a inclusão do campo externo do modelo de Ising para evitar que o algoritmo fique preso em pontos de mínimos locais da função de energia. Além disso, é proposto um algoritmo adicional para transformar o resultado obtido pelo modelo em um escalonamento viável. Os resultados demonstram um bom desempenho no controle do tamanho das filas em comparação aos algoritmos existentes.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SCHEDULING ALGORITHM USING ISING MODEL WITH EXTERNAL FIELD

Nilton Guedes Duarte

March/2019

Advisor: José Ferreira de Rezende

Department: Systems Engineering and Computer Science

Shared medium access algorithms for wireless networks that take into account the link queue size have recently received a lot of attention. In this paper, we propose a distributed algorithm for link scheduling based on Ising's physical model for antiferromagnetism. This model was adapted, by another work, to consider the size of the queues and to use Glauber dynamics to minimize the energy captured by the model. In this work, we propose the inclusion of the external field of the Ising model to avoid that the algorithm get stuck in local minima of the energy function. In addition, a second algorithm is proposed to transform the result obtained by the model into a viable scheduling. The results demonstrate a good performance in controlling the queues sizes in comparison to the existing algorithms.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

CSMA	Acesso Múltiplo com Sensoriamento da Portadora, p. 1
EsMAI	Escalonador Maximal para Atribuição de Intervalo, p. 28
EsMa-S/F	EsMa Sem Fila, p. 41
EsMa	Escalonador Maximal, p. 27
GD	Dinâmica de Glauber, p. 30
IP	Protocolo de Internet, p. 1
MAC	Controle de Acesso ao Meio, p. 1
MCMC	Cadeias de Markov Monte Carlo, p. 36
MH	Metropolis-Hastings, p. 9
MWS	*Max-Weight-Scheduling*, p. 4
PGD-CSMA	Dinâmica de Glauber em Paralelo-CSMA, p. 10
TCP	Protocolo de Controle de Transmissão, p. 1
TDMA	Acesso Múltiplo por Divisão de Tempo, p. 1

Capítulo 1

Introdução

Redes sem fio estão presentes cada vez mais em nosso dia a dia. A empresa de telecomunicações Cisco [1] estima que, em 2022, o tráfego IP triplicará quando comparado com 2017, e o número de dispositivos conectados à rede IP será maior que 3 vezes a população global. Pela previsão da empresa, em 2022, 71% do tráfego IP será proveniente de dispositivos sem fio e móveis. Em 2017, tráfego IP proveniente de dispositivos sem fio e móveis foi de 52%.

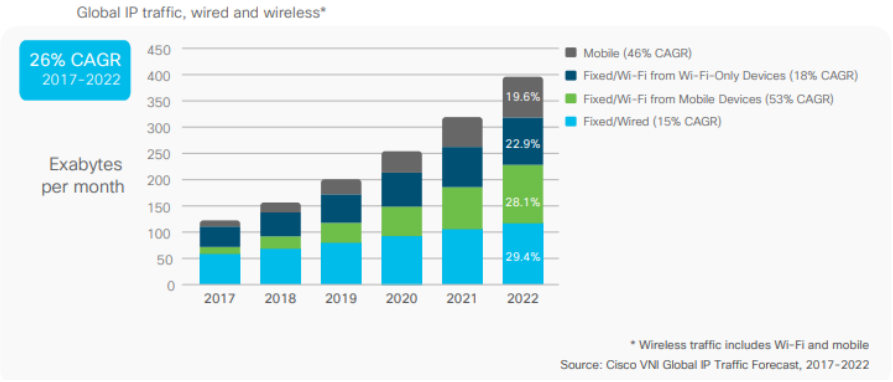


Figura 1.1: Tráfico IP Global 2017-2022. Imagem adaptada de [1].

A comunicação em redes de computadores acontece por meio de um conjunto de protocolos e mecanismos que permitem a troca de dados entre computadores. Por exemplo, entre os mais conhecidos estão o TCP, IP e IEEE 802.11 (Wi-Fi). Um protocolo é de especial interesse para este trabalho: o de escalonamento. Em uma rede é necessário decidir quando um computador transmitirá, essa escolha é feita pelos protocolos de escalonamento Controle de Acesso ao Meio (*Medium Access Control* - MAC), específicos para cada situação.

Dois métodos de acesso ao meio são importantes neste trabalho, o CSMA (Acesso múltiplo com Sensoriamento da Portadora) e o TDMA (Acesso Múltiplo por Divisão de Tempo). No primeiro, CSMA, o meio físico de transmissão é disputado pelos dispositivos, que fazem sensoriamento do meio para diminuir a probabilidade de um

tipo de falha na transmissão, as colisões. No método TDMA, o tempo de transmissão é dividido e cada dispositivo tem seu intervalo de tempo para transmitir. Os dois métodos estão ilustrados na Figura 1.2. No TDMA, cada usuário transmite na sua janela de tempo, já no CSMA, os usuários disputam o meio, e podem ocorrer falhas, como por exemplo, quando os usuários 1 e 2 transmitem simultaneamente. A falha ilustrada é uma colisão.

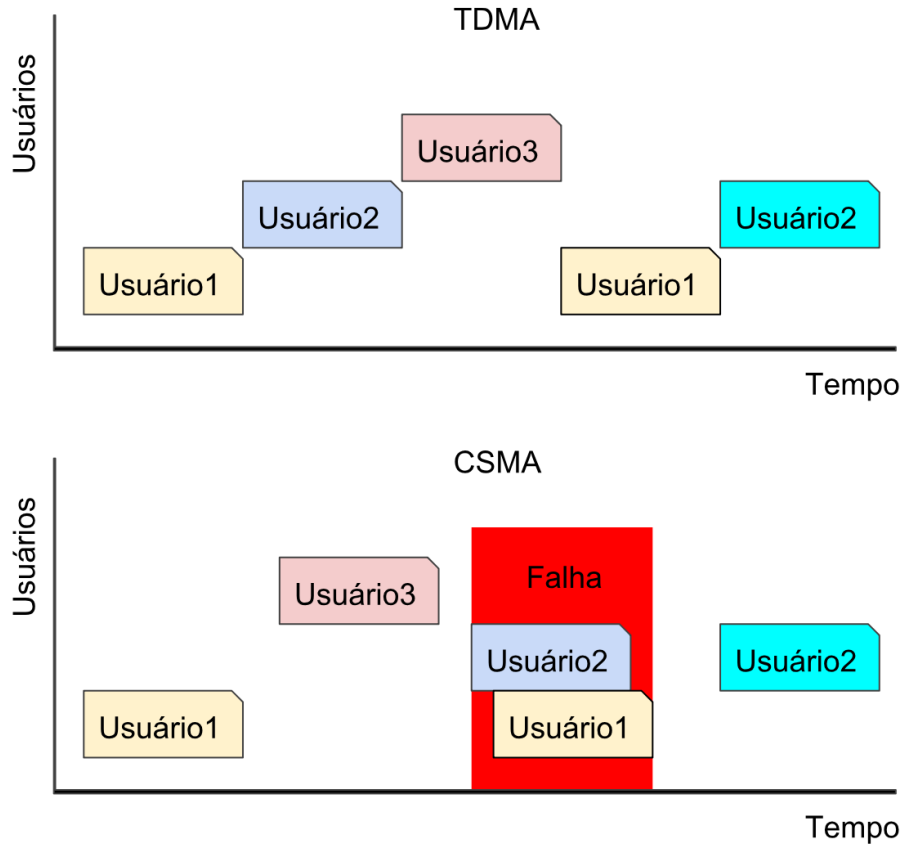


Figura 1.2: Métodos de Acesso ao Meio TDMA e CSMA

Quando dois dispositivos de rede sem fio desejam se comunicar, eles formam uma ligação lógica, um enlace. Quando dois enlaces têm seus dispositivos suficientemente próximos ou compartilham um mesmo dispositivo, eles se interferem. Esse é o modelo de interferência baseado em protocolo. Nessa modelo a relação é representada através de um grafo, o grafo de interferência, onde os nós representam os enlaces e as arestas representam a interferência mútua, isto é, quando dois enlaces se interferem, são considerados enlaces vizinhos. Para esse modelo de interferência não é considerado o sentido da transmissão. Se em algum sentido pode haver interferência na transmissão entre dois enlaces, então eles se interferem, mesmo que haja sentidos que não causem interferência. Essa simplificação se da pois é comum que transmissões sem fio tenham algum tipo de resposta, mesmo que seja um reconhecimento de recebimento.

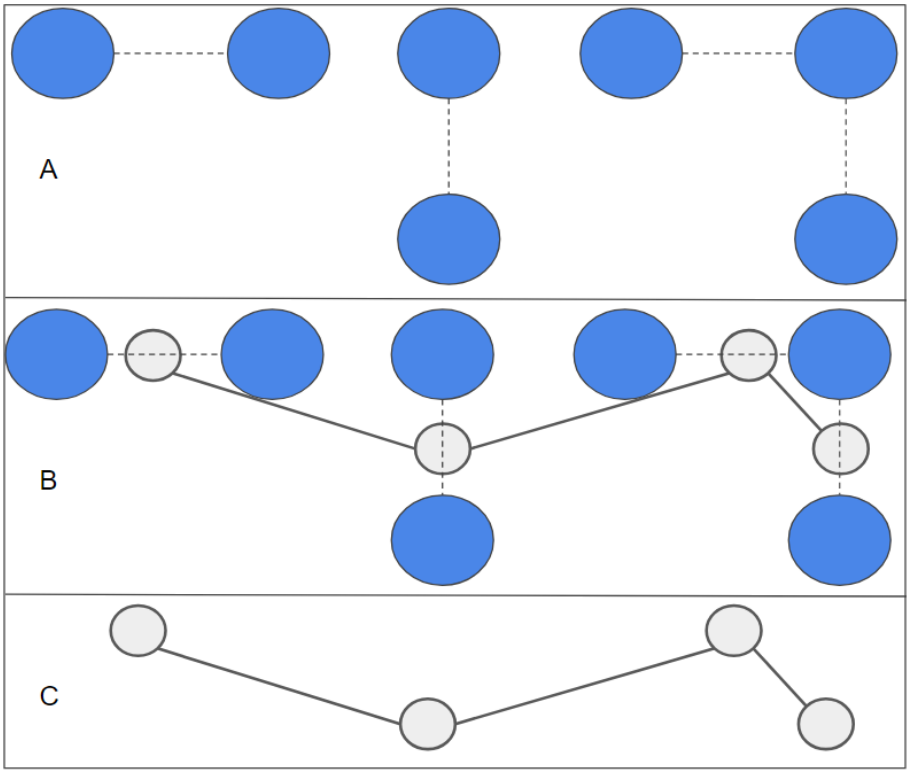


Figura 1.3: Construção do grafo de interferência de enlaces. (A) Grafo de dispositivos e enlaces. (B) Construção do grafo de interferência sobre o grafo de dispositivos. (C) Grafo de interferência.

A Figura 1.3 ilustra a criação de um grafo de interferência (C) a partir de um grafo de dispositivos e enlaces (A). Em (A), os dispositivos estão representados por nós em azul e o enlace pela aresta tracejada. Em (B), o grafo de interferência é construído sobreposto ao grafo de (A), onde cada enlace (aresta tracejada) passará a ser representado por um nó (Cinza) e a interferência entre os enlaces que estão suficientemente próximos ou compartilham um dispositivo é representada por uma aresta continua. O grafo em (C) é o grafo de interferência construído a parti de (A).

Para o escalonamento de enlaces, o grafo de interferência nos dá uma informação valiosa: caso dois enlaces (nós) vizinhos do grafo transmitam simultaneamente, ocorrerá uma falha na transmissão. Para que uma transmissão de um enlace (nó) seja bem sucedida, é preciso que todos os enlaces vizinhos no grafo não transmitam no mesmo instante. Neste trabalho, o foco estará no escalonamento dos enlaces.

O algoritmo para escalonar enlaces em redes sem fio deve buscar a eficiência da utilização do canal, proporcionando alta vazão com baixa latência. É também desejável que sua implementação seja simples e distribuída, exigindo apenas conhecimento local da rede. Pela lei de Little, a vazão (ou taxa de chegada) e a latência estão relacionadas com o tamanho da fila do enlace. Podemos, então, medir o tamanho das filas, sabendo que quanto menor o tamanho médio da fila, maior foi a vazão e menor foi a latência. Embora a vazão/taxa de chagada do sistema não seja possível

saber com exatidão, é possível estimá-la [2], o mesmo da latência. O tamanho da fila será de extrema importância para este trabalho. Escalonar um enlace para o qual não há dado na fila para ser enviado não traz ganho para aumentar a vazão ou diminuir a latência. Utilizaremos essa informação do tamanho da fila para tomar decisões sobre quais enlaces devem ser escalonados. Nos últimos anos, algoritmos onde a probabilidade de um enlace ser escalonado depende do tamanho de sua fila estão sendo estudados [2–7]. A alta vazão é alcançada quando o algoritmo consegue estabilizar as filas para todas as taxas de chegada dentro da região de capacidade da rede.

O algoritmo de escalonamento de peso máximo (*max-weight scheduling*-MWS) é *throughput-optimal* [8] quando o peso a ser otimizado é uma função do tamanho da fila dos enlaces. Porém, o algoritmo funciona resolvendo um problema combinatório NP-difícil, de encontrar um conjunto independente no grafo de interferência atendendo a certos critérios, sendo necessário conhecimento global da rede. Por outro lado, um algoritmo como o usado pelo IEEE 802.11 (Wi-Fi) é mais simples, não necessita de nenhuma informação da rede, porém não garante vazão máxima.

O problema de escalonamento de enlaces é que dois enlaces que são vizinhos não podem transmitir ao mesmo tempo, pois causariam interferência mútua levando a falha nas transmissões, resultando em desperdício de recurso. Por exemplo, o algoritmo CSMA/CA (utilizado pelo padrão IEEE 802.11 - Wi-Fi) tenta transmitir aguardando um tempo aleatório depois de verificar que o meio de transmissão está livre. Este algoritmo leva a problemas de colisão caso dois dispositivos vizinhos escolham aguardar o mesmo tempo, ambos tentarão transmitir simultaneamente e ocorrerá uma colisão, entre outros problemas.

Temos então um problema muito dinâmico para resolver. Os enlaces recebem pacotes para serem transmitidos em uma taxa que não é conhecida, criando uma demanda e aumentando a fila. Então é preciso encontrar um escalonamento para atender essa demanda e diminuir a fila.

A transmissão será dividida em duas fases: uma fase de controle e uma fase de transmissão. A fase de controle encontrará o conjunto de enlaces que não se interferem e na fase de transmissão ocorre a transmissão em si. A Figura 1.4 ilustra a divisão de fases e a dinâmica do problema. É importante ressaltar que nesta dinâmica as filas do sistema estão em constante mudança.

Então queremos resolver o problema de encontrar o conjunto independente de enlaces (um escalonamento viável) que maximize a soma das filas dos enlaces do conjunto. Garantir essa propriedade requer conhecimento global do sistema e resolver um problema difícil. Gostaríamos de uma solução aproximada, que não envolva um problema difícil e que necessite apenas de conhecimento local. Como a natureza do problema é dinâmica, essa solução aproximada precisará estar sempre mudando

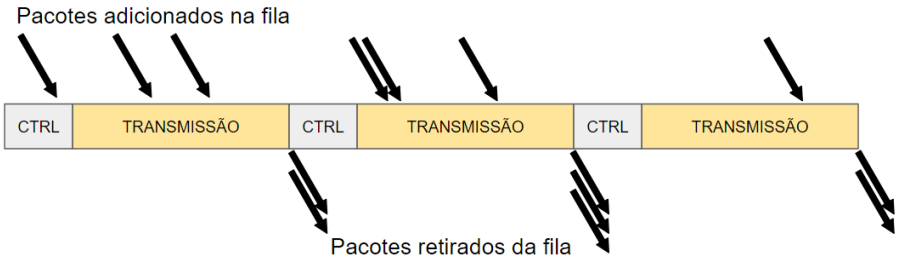


Figura 1.4: Dinâmica do problema. Fases de controle transmissão

para refletir as mudanças no sistema.

Antes de tentar aproximar a solução, precisamos encontrar um conjunto de enlaces que não sejam vizinhos para que possam transmitir simultaneamente, ou seja, o "desalinhamento"do momento de transmissão dos enlaces vizinhos. Um efeito de desalinhamento é encontrado em partículas com propriedades magnéticas. Podemos traçar um paralelo entre as partículas e o problema de escalonamento de enlaces. Um modelo conhecido e estudado sobre o ferromagnetismo é o modelo de Ising [9], esse modelo probabilístico, baseado na dinâmica de Glauber, se mostra interessante para o problema de escalonamento. Pode, por exemplo, ser utilizado praticamente sem alterações para encontrar uma aproximação de um escalonamento maximal de uma rede.

O modelo de Ising representa utilizando a estrutura de grafo a interação entre *spins* atômicas, a temperatura do sistema e com o campo externo. O modelo será apresentado mais detalhadamente na Seção 2.1.1.

Em [3], os autores adaptam esse modelo para o problema do escalonamento de enlaces. O escalonamento dos enlaces da rede é associado a uma configuração do modelo de Ising e a energia da configuração é calculada. No entanto, essa adaptação apresenta dois problemas. Primeiro, ele apresenta um problema de convergência para a solução de escalonamento adequada. Em certas configurações, o algoritmo não consegue sair de pontos de mínimo da configuração de energia do sistema, fazendo com que o sistema tenha como resultado sempre o mesmo escalonamento, os enlaces que não fazem parte desse resultado sofrem de *starvation*.

O segundo problema é que o conjunto de enlaces E encontrado pelo algoritmo pode ser um subconjunto próprio de um outro escalonamento de enlaces S, em que S também é um escalonamento viável ($E \subsetneq S$), isto é, o resultado E poderia ser expandido para acrescentar outros nós e chegar até S e ainda assim se manter nas limitações do problema. No entanto, uma vantagem desse modelo em relação a outros modelos da literatura é o fato dele não se restringir a um subconjunto das configurações de escalonamentos viáveis da cadeia de Markov subjacente, permitindo assim que novos algoritmos sejam explorados.

Neste trabalho, propomos expandir o modelo para adaptar a ação do campo

externo do modelo de Ising para o problema de escalonamento. Essa modificação evita que o algoritmo fique preso em pontos de mínimos locais da função de energia. Além disso, propomos um algoritmo para transformar o resultado E obtido pelo modelo em um escalonamento viável maximal E', tal que para todo escalonamento viável S, temos que $S \neq E' \Rightarrow E' \not\subset S$.

Os resultados do algoritmo existente e dos algoritmos propostos serão avaliados através de simulação, medindo o tamanho médio das filas ao final do tempo de simulação

Esta dissertação está estruturada da seguinte forma. No Capítulo 2 estão os fundamentos e trabalhos relacionados mais relevantes para a compreensão da proposta. O modelo de Ising é apresentado na Seção 2.1.1, os principais fundamentos de Markov Chain Monte Carlo utilizados no modelo e no simulador são apresentados na Seção 2.1.2. O modelo de Ising adaptado por [3] para levar em consideração o tamanho da fila do enlace e de seus vizinho esta detalhado na Seção 2.2.1. A Seção 2.2.2 apresenta um algoritmo chamado DRAND, que é inspiração para os algoritmos de escalonamento propostos. O Capítulo 3 contém as propostas para solucionar os problemas mencionados nestra introdução. Na Seção 3.1 propomos o modelo MICE, com alteração para adaptar o campo externo atuando sobre o modelo. Na Seção 3.2, apresentamos dois algoritmos baseados em TDMA, EsMa e EsMAI, para encontrar um escalonamento viável a partir de uma configuração qualquer do modelo de Ising e na Seção 3.3 apresentamos e propomos algumas heurísticas para o algoritmo. No Capítulo 4 será apresentado o software desenvolvido e utilizado. Por fim, no Capítulo 5, apresentamos os resultados das simulações em vários cenários e concluímos no Capítulo 6.

Capítulo 2

Fundamentos e Trabalhos Relacionados

Neste capítulo, conectaremos o modelo físico de Ising com o problema de escalonamento de enlaces em redes sem fio, mas primeiro precisamos conectar a vazão e a latência com o tamanho das filas.

Pela lei de Little [10], em um sistema em equilíbrio, a taxa de chegada é igual à taxa de saída. A lei de Little se aplica para qualquer sistema, inclusive para sistemas dentro do sistema. Portanto, podemos utilizar a lei de Little para um único enlace, assim como para todo o conjunto de enlaces do sistema. Seja f o número de pacotes na fila, v a taxa de chegada de pacotes e a o tempo de serviço, por Little:

$$f = va \tag{2.1}$$

O tempo de serviço de um pacote é o tempo que ele demora para ser enviado pelo enlace, tempo de processamento mais o tempo de espera, portanto a é o atraso (*delay*). Um bom algoritmo irá escalonar mais vezes os enlaces com maior taxa de chegada de pacotes, diminuindo o tempo de serviço para os pacotes da rede.

Podemos então apresentar os fundamentos do modelo de Ising e brevemente o método de Cadeia de Markov Monte Carlo, importantes para este trabalho. Na sequência, são apresentados os trabalhos relacionados, com maior destaque para o I-CSMA e DRAND.

2.1 Fundamentos

2.1.1 Modelo de Ising

O modelo de Ising [9] [11] [12] foi desenvolvido pelo físico alemão Ernst Ising para o ferromagnetismo em 1925. O modelo representa os momentos de dipolo magnético

de *spins* atômicas, que podem estar em um de dois possíveis estados $\{-1, +1\}$.

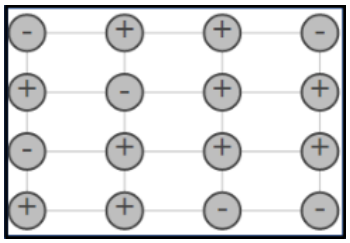


Figura 2.1: Representação dos *spins* no modelo de Ising

Para estudar o modelo, utilizaremos um grafo $K = (V, E)$ para representar as interações entre os sítios vizinhos. Os nós em V são os *spins* e as arestas em E representam as interações entre os *spins* vizinhos. Para um nó $v \in V$, a variável $\sigma_v \in \{-1, +1\}$ representa o estado do *spin* de v. Esta representação está ilustrada na figura 2.1.

O espaço de todas as possíveis configurações do sistema é $\Omega = \{-1, +1\}^{|V|}$. Uma configuração $\sigma \in \Omega$ desse modelo é representada por um vetor que associa um *spin* para cada nó $(\sigma_v)_{v \in V} = \sigma$. Para cada par $(i, j) \in E$ de nós vizinhos no grafo, existe uma interação $J_{i,j}$, e para cada nó $v \in V$ existe um campo magnético externo h_j e o momento magnético γ. A energia da configuração σ é dada pela função Hamiltoniana $\mathcal{H}(\sigma)$ (2.2):

$$\mathcal{H}(\sigma) = -\sum_{(i,j)\in E} J_{i,j}\sigma_i\sigma_j + \gamma \sum_{i \in V} h_i\sigma_i \tag{2.2}$$

O primeiro somatório de (2.2) é sobre todos os spins dos pares de nós vizinhos e o segundo sobre todos os spins do modelo. A probabilidade de uma configuração σ qualquer ocorrer no sistema é dada pela medida de Gibbs com o inverso da temperatura $\beta = (k_B T)^{-1} \geq 0$ (2.3), k_B é a constante de Boltzmann. A constante de normalização $Z(\beta)$ é chamada função de partição.

$$\mu(\sigma) = \frac{e^{-\beta\mathcal{H}(\sigma)}}{Z(\beta)} \qquad Z(\beta) = \sum_{\sigma \in \Omega} e^{-\beta\mathcal{H}(\sigma)} \tag{2.3}$$

Simplificaremos o modelo fazendo com que todas as interações entre os *spins* sejam iguais, $J_{ij} = J$. Podemos classificar o modelo de acordo com o sinal de J. Para $J > 0$, o modelo é ferromagnético, para $J < 0$ o modelo é antiferromagnético. No modelo ferromagnético, a interação entre os spins vizinhos faz com que eles desejem se alinhar (ter o mesmo sinal) e as configurações onde os vizinhos têm o mesmo sinal têm maior probabilidade de ocorrer. No modelo antiferromagnético, os vizinhos desejam se desalinhar (terem sinais opostos) e, consequentemente, configurações onde os vizinhos se encontram com sinais opostos têm maior probabilidade de ocorrência.

A temperatura tem papel importante para o modelo ferromagnético ($J > 0$). Quando $\beta = 0$ (temperatura infinita), o modelo se comporta de forma totalmente aleatória, de forma que todas as configurações têm a mesma probabilidade de ocorrer no sistema. Conforme a temperatura do sistema diminui e se aproxima de zero ($\beta \to \infty$), as probabilidades tendem a levar o sistema para o estado onde todos os *spins* estão alinhados.

A dinâmica de Glauber para o modelo de Ising é uma cadeia de Markov com o espaço de estados Ω e com μ como sua distribuição de estado estacionária. Para determinar a próxima configuração (estado), um nó v é escolhido aleatoriamente e será $\sigma_v = +1$ com probabilidade $q(+1; \sigma, v)$ (2.5) ou $\sigma_v = -1$ com probabilidade $q(-1; \sigma, v) \triangleq 1 - q(+1; \sigma, v)$, essas configurações serão representadas respectivamente por $\Psi^+_{\sigma,v}$, descrito em (2.4), e $\Psi^-_{\sigma,v}$. Então temos que $q(+1; \sigma, v)$ é dado por (2.5).

$$\Psi^+_{\sigma,v} = \{x_i | x_i = +1 \ : \ i = v, x_i = \sigma_i \ : \ i \in V \setminus \{v\}\} \tag{2.4}$$

$$q(1; \sigma, v) = \frac{\mu(\Psi^+_{\sigma,v})}{\mu(\Psi^+_{\sigma,v}) + \mu(\Psi^-_{\sigma,v})} \tag{2.5}$$

Na dinâmica de Glauber, apenas um nó é atualizado por vez.

O modelo antiferromagnético é interessante para o problema do escalonamento de enlaces, pois é necessário que dois enlaces que se interferem não transmitam simultaneamente, isto é, estejam *desalinhados*. A medida de Gibbs do modelo de Ising para o antiferromagnetismo tem maior probabilidade em configurações de menor energia, em que os vizinhos tendem a estar desalinhados. Um algoritmo conhecido utilizado no Modelo de Ising é o algoritmo de Metropolis-Hastings.

2.1.2 Método de Cadeia de Markov Monte Carlo (MCMC) e Metropolis-Hastings (MH)

O método MCMC constrói uma cadeia de Markov onde a distribuição estacionária de interesse (Equação 2.3) é preservada ao caminhar pela cadeia, de forma que o número de vezes que passamos por cada estado obedece a distribuição estacionária da cadeia. Esse método é utilizado por ser inviável calcular toda a cadeia.

O algoritmo MH é uma generalização da Dinâmica de Glauber (ou Amostragem de Gibbs). No algoritmo MH, é escolhido um estado do espaço de estados e aceitamos essa escolha como o novo estado do sistema com uma certa probabilidade, de modo a respeitar o estado estacionário da cadeia. No caso da Amostragem de Gibbs, escolhem-se as variáveis do vetor de estado em turnos, atualizando seus valores condicionando-os a todas as outras variáveis, utilizando o valor mais recente das demais variáveis como ilustrado na Equação 2.6 com o vetor $x^t = (x^t_1, x^t_2, x^t_3)$

contendo três variáveis. A probabilidade de aceitação na Dinâmica de Glauber é de 100%.

$$\begin{aligned} x^t &= (x_1^t, x_2^t, x_3^t) \\ x_1^{t+1} &= p(x_1 | x_2^t, x_3^t) \\ x_2^{t+1} &= p(x_2 | x_1^{t+1}, x_3^t) \\ x_3^{t+1} &= p(x_3 | x_1^{t+1}, x_2^{t+1}) \end{aligned} \tag{2.6}$$

Como no modelo de Ising a probabilidade de mudança de valor de uma variável (*spin*) do vetor de estado é dependente apenas dos seus vizinhos, podemos condicionar a sua atualização apenas a eles.

2.2 Trabalhos Relacionados

Muitos trabalhos buscam resolver o problema do escalonamento de enlaces utilizando o tamanho das filas. Os trabalhos apresentados a seguir não utilizam o modelo de Ising, mas utilizam a dinâmica de Glauber. O artigo que adapta o modelo de Ising é apresentado com mais detalhes em 2.2.1.

O algoritmo mais simples de dinâmica de Glauber para escalonamento de enlace é o PGD-CSMA (Dinâmica de Glauber em Paralelo-CSMA), os autores de [5] apresentaram o algoritmo para mostrar que CSMA baseado em dinâmica de Glauber tem algumas propriedades como crescimento polinomial do tamanho da fila e limitantes para o atraso sob certas circunstâncias. O algoritmo PGD-CSMA é o seguinte:

PGD-CSMA [5][1]

Inicialização:

1. Todos os enlaces em estado DESLIGADO

Para cada tempo $t \geq T$

2. Encontrar um conjunto de enlaces independentes $\mathcal{D}(t)$ aleatório
3. Para cada link $i \in \mathcal{D}(t)$:
4. Se os vizinhos de i em $t-1$ estão em estado DESLIGADO:
5. i =LIGADO com probabilidade $\frac{\lambda_i}{1+\lambda_i}$
6. i =DESLIGADO com probabilidade $\frac{1}{1+\lambda_i}$
7. Caso contrário: i = DESLIGADO
8. Todos os links que não pertencem a $\mathcal{D}(t)$ permanecem no mesmo estado

[1] Adaptado do artigo original do algoritmo PGD-CSMA [5].

9. Os enlaces em estado LIGADO transmitem

O algoritmo utiliza informações do tempo $t-1$, o parâmetro λ_i, que é chamado de fugacidade e que pode, por exemplo, ser uma função da fila de modo que o algoritmo seja *throughput-optimal*.

O problema de mínimos locais apontado no Capítulo 1 é conhecido e está relacionado à forte correlação temporal entre os estados da cadeia da Markov subjacente. Em [6], o autor busca solucionar esse problema. Ele apresenta o algoritmo Delayed-CSMA, que adiciona uma nova ordem de cadeia ao algoritmo, isto é, o algoritmo passa a ser uma cadeia de cadeias de Markov. A Figura 2.2 exemplifica esta diferença. Em (a) o algoritmo tradicional, uma sequência de estados de uma mesma cadeia, em (b) o algoritmo proposto, onde a sequência de estado alterna entre duas cadeias.

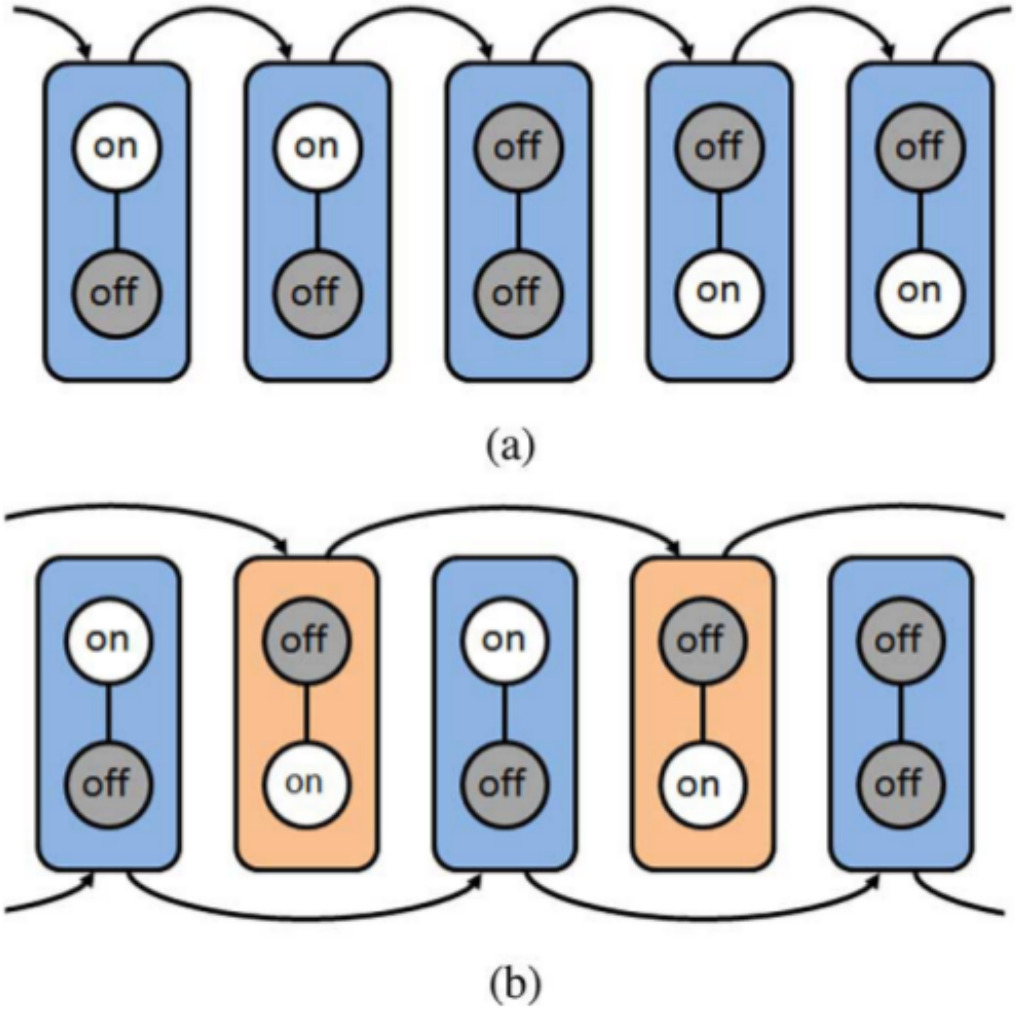


Figura 2.2: Comparação entre CSMA e Delayed CSMA. Figura retirada de [6].

O algoritmo a seguir é o Delayed-CSMA.

Delayed CSMA [6][2]

Inicialização:

1. Todos os enlaces em estado DESLIGADO

Para cada tempo $t \geq T$

2. Encontrar um conjunto de enlaces independentes $\mathcal{D}(t)$ aleatório
3. Para cada link $i \in \mathcal{D}(t)$:
4. Se os vizinhos de i em $t - T$ estão em estado DESLIGADO:
5. i =LIGADO com probabilidade $\frac{\lambda_i}{1+\lambda_i}$
6. i =DESLIGADO com probabilidade $\frac{1}{1+\lambda_i}$
7. Caso contrário: i = DESLIGADO
8. Todos os links que não pertencem a $\mathcal{D}(t)$ permanecem no mesmo estado
9. Os enlaces em estado LIGADO transmitem

Aqui, o algoritmo utiliza informações dos vizinhos no tempo $t-T$. O parâmetro T define quando cadeias serão alternadas e λ_i, a fugacidade, pode ser uma função da fila.

Outro algoritmo importante é o Q-CSMA de [4], uma generalização da dinâmica de Glauber. O algoritmo utiliza subdivisões do intervalo de controle em mini-intervalos.

Q-CSMA [4][3]

Procedimento do enlace i no intervalo de tempo t:

1. Escolher um tempo de espera aleatório T_i uniformemente em $[0, W-1]$ e aguardar T_i mini-intervalos de controle
2. Se o enlace i ouvir uma mensagem de *INTENT* de um enlace vizinho antes de $T_1 + 1$ mini-intervalos de controle, ele não mais transmitirá sua mensagem de *INTENT* e permanecerá no mesmo estado
3. Se o enlace i não ouvir nenhuma mensagem de *INTENT* de seus vizinhos antes de $T_i + 1$ mini-intervalos de controle:
4. Se tiver uma colisão na transmissão do *INTENT*, o enlace permanecerá no mesmo estado
5. Se não houve colisão, o enlace i decidirá conforme a seguir:

[2]Adaptado do artigo original do algoritmo Delayed-CSMA [6].
[3]Adaptado do artigo original do algoritmo Q-CSMA [4].

6. Se nenhum vizinho de i estava ativo no tempo anterior:
7. i=LIGADO com probabilidade p_i
8. i=DESLIGADO com probabilidade $1 - p_i$
9. Caso contrário: i=DESLIGADO
10. O enlace transmitirá caso esteja no estado LIGADO

A probabilidade p_i depende do tamanho da fila do enlace i. Todos os algoritmo descritos apresentam uma restrição: o enlace só pode atualizar seu estado caso todos os seus vizinhos estejam em estado DESLIGADO. O algoritmo I-CSMA, apresentado a seguir, remove esta restrição.

2.2.1 I-CSMA e a adaptação do modelo para filas

O modelo de Ising foi adaptado por Yi Wang e Ye Xia em seus artigos [3] [13] [14] para considerar o tamanho das filas. Em seu artigo, os autores propõem um algoritmo de escalonamento para redes sem fio baseado na versão modificada do modelo de Ising, chamado de I-CSMA.

Segundo os autores, o I-CSMA é uma generalização de algoritmos anteriores *throughput-optimal* baseados na dinâmica de Glauber, como o Q-CSMA [4], em que o algoritmo se restringia a uma cadeia de Markov truncada apenas nos estados nos quais não havia interferência entre os enlaces. O artigo demonstra que a restrição não é necessária, a cadeia de Markov pode se mover livremente sobre todo o espaço de configurações e outros métodos de restringir a configuração para o subconjunto de configurações viáveis funcionam igualmente bem para alcançar *throughput-optimality*.[4]

Para o algoritmo do I-CSMA, a influência do campo externo do modelo de Ising é eliminada, $\gamma = 0$. A atualização dos *spins* do modelo é realizada em paralelo (dinâmica de Glauber paralela), um subconjunto ξ dos nós do grafo é escolhido aleatoriamente com a restrição de que dois nós vizinhos não podem fazer parte do mesmo conjunto. O conjunto é chamado de conjunto de atualização, então cada vértice de ξ é atualizado de acordo com as probabilidades $q(+1; \sigma, v)$ ou $q(-1; \sigma, v)$, independentemente de outros vértices em ξ.

Utilizaremos um grafo de interferência $G(V, E)$, em que V representa o conjunto dos enlaces e E indica a relação de interferência entre os enlaces de V. Os valores $\sigma_v = \{-1, +1\}$ que representavam os valores de *spin* no modelo de Ising, agora representarão o **desejo** de um enlace transmitir. Quando $\sigma_v = +1$, o enlace v deseja transmitir, mas não deseja quando $\sigma_v = -1$.

[4]Parágrafo adaptado do artigo original do algoritmo I-CSMA [3].

Dado $\sigma \in \Omega$, é calculado o vetor $F = (F_v)_{v \in V}$ em função da fila de cada enlace v, onde $F_v > 0$ é uma função crescente da fila de v. Por fim, é calculado a função $df_{\sigma,F}(v)$ (2.7) que depende do desejo e da fila de cada enlace v

$$df_{\sigma,F}(v) = \begin{cases} F_v, & \text{se } \sigma_v = +1 \\ -1, & \text{se } \sigma_v = -1 \end{cases} \tag{2.7}$$

Definimos então a energia antiferromagnética H da configuração σ sob o vetor F por (2.8)

$$H(\sigma, F) = \sum_{(v,w) \in E} df_{\sigma,F}(v) df_{\sigma,F}(w) \qquad J = -1 \tag{2.8}$$

As configurações de baixa energia tendem a ter poucos pares onde dois enlaces vizinhos desejam transmitir $(+1, +1)$ e muitos pares onde apenas um vizinho deseja transmitir $(+1, -1)$. Tem a mesma distribuição de probabilidade do modelo de Ising (2.3), agora β passa a ser interpretado como um parâmetro do sistema e $Z(\beta)$ apenas uma constante normalizadora. Para a probabilidade de atualização de um enlace $q(+1; \sigma, v)$ temos

$$q(1; \sigma, v) = \frac{\mu(\Psi^+_{\sigma,v})}{\mu(\Psi^+_{\sigma,v}) + \mu(\Psi^-_{\sigma,v})} = \frac{e^{-F_v \beta S(\sigma,v)}}{e^{-F_v \beta S(\sigma,v)} + e^{\beta S(\sigma,v)}} = \frac{1}{2}\left(1 - \tanh\left(\frac{F_v + 1}{2}\beta S(\sigma, v)\right)\right) \tag{2.9}$$

$$S(\sigma, v) \triangleq \sum_{w \,:\, (v,w) \in E} df_{\sigma,F}(w) \tag{2.10}$$

É importante perceber que a probabilidade depende apenas dos vizinhos de v. Essa propriedade faz com que apenas a informação local seja necessária.

O I-CSMA utiliza a função de fila (2.11), em que d^* é o maior grau do grafo e $Q_v(t)$ é o tamanho fila de v no tempo t.

$$F_v = 2(d^* - 1) + \log(Q_v(t) + 1) \tag{2.11}$$

A visão geral do algoritmo I-CSMA é a seguinte:

1. O algoritmo funciona com a dinâmica de Glauber paralela, em que os *spins* foram substituídos pelo vetor de fila F, que varia no tempo.

2. Quando um estado da dinâmica de Glauber no tempo t não for um escalonamento viável, o algoritmo converterá para um escalonamento válido.

Para a segunda etapa, seja $\sigma(t) \in \Omega$ um estado da dinâmica de Glauber. O algoritmo converte $\sigma(t)$ para um escalonamento viável $\sigma'(t)$. A conversão é definida por um mapeamento $\psi : \sigma(t) \mapsto \sigma'(t)$. Cada mapeamento ψ é uma nova versão do algoritmo.

A versão a seguir é apresentada pelos autores do I-CSMA. O intervalo de tempo é dividido em três intervalos: dois intervalos de controle e um intervalo de transmissão. Os intervalos de controle têm duração constante W e W' mini-intervalos. Na primeira fase de controle, será escolhido coletivamente o conjunto $\xi(t)$, o conjunto dos nós independentes que podem ser atualizados segundo a dinâmica de Glauber em paralelo, para definir o vetor $\sigma(t)$. Na segunda fase de controle, será encontrado o resultado $\sigma'(t)$ do mapeamento de ψ. Por fim, os nós em $\sigma'(t)$ transmitem.

I-CSMA - Algoritmo de escalonamento no enlace v

Inicialização:

1. No início do intervalo de tempo, o enlace v calcula $S(\sigma(t-1), v)$ baseado no estado dos vizinhos e do tamanho da fila que aprendeu durante o intervalo anterior. O enlace v calcula a probabilidade $q(+1; \sigma(t-1), v)$ de acordo com (2.9).

Fase de Controle 1 - W Mini-Intervalos: Estabelecer $\sigma_v(t)$

2. Enlace v seleciona um tempo aleatório T_1 distribuído uniformemente em $\{0, 1, ..., W-1\}$ e inicia um temporizador de T_1 mini-intervalos de controle.

3. Se o enlace v receber uma mensagem de *INTENT* de qualquer um de seus enlaces vizinhos antes do temporizador T_1 expirar, ele define $\sigma_v(t) = \sigma_v(t-1)$ e não vai transmitir uma mensagem de *INTENT* (v não será incluído em $\xi(t)$).

4. Caso contrário, quando T_1 expirar, o enlace v envia em *broadcast* uma mensagem de *INTENT* no inicio do mini-intervalo $(T_1 + 1)$.

   - Se a mensagem de *INTENT* de v colidiu, o enlace v define $\sigma_v(t) = \sigma_v(t-1)$ (v não será incluído em $\xi(t)$).
   - Caso contrário, o enlace v define $\sigma_v(t) = +1$ com probabilidade $q(+1; \sigma, v)$, ou $\sigma_v(t) = -1$ com probabilidade $q(-1; \sigma, v)$

Fase de Controle 2 - W' Mini-Intervalos: Estabelecer $\sigma'_v(t)$

5. Enlace v define $\sigma'_v(t) = 0$. Se $\sigma_v(t) = 1$, v executa o seguinte

   - Enlace v seleciona um tempo aleatório T_2 distribuído uniformemente em $\{0, ..., W'-1\}$ e inicia um temporizador de T_2 mini-intervalos de controle.
   - Quando o temporizador T_2 expira, v envia uma mensagem de *RESERVE* em *broadcast* com seu tamanho de fila atual

- Se o enlace v não recebeu nenhuma mensagem de *RESERVE* de seus enlaces vizinhos antes do fim do temporizador T_2 e sua mensagem de *RESERVE* não colidiu, o enlace v define $\sigma'_v(t) = 1$

Intervalo de Dado:

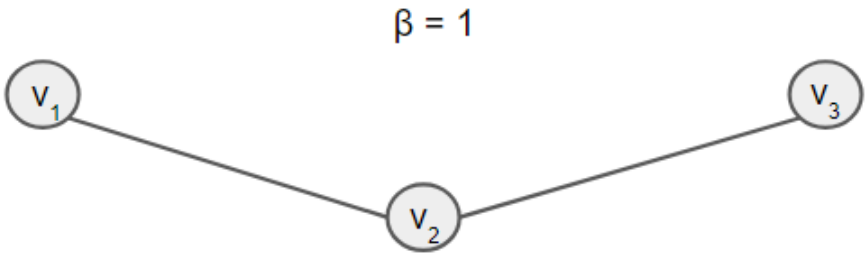
6. Se $\sigma'_v(t) = 1$, o enlace v transmite um pacote

$$\mathcal{IS}(l) = \{i : \text{enlace } i \text{ compartilha um dispositivo com } l, \text{ou } i_s \in \mathcal{N}(l_r), \text{ou } i_r \in \mathcal{N}(l_s)\}. \tag{2.12}$$

As informações para calcular $S(\sigma(t-1))$ são obtidas através da mensagem de *RESERVE* do intervalo anterior.

A seguir, as Figuras 2.3, 2.4, 2.5, 2.6 e 2.7 ilustram um grafo de 3 enlaces com alguns valores para a função de fila F_v, uma configuração para σ e as probabilidades $q(+1;\sigma,v)$ e $q(-1;\sigma,v)$ correspondentes de cada enlace para exemplificar o funcionamento do modelo.

O problema de mínimo local pode ser observado na Figura 2.3, onde o enlace v_2 com a maior fila tem probabilidade zero ($q(+1;\sigma,v_2) = 0$) de mudar para o estado $\sigma_2 = 1$. Os enlaces v_1 e v_3 também tem probabilidade muito baixa para mudar de estado. Ainda que algum dos enlaces v_1 ou v_3 mude para o estado $\sigma_i = -1$, como na Figura 2.4, a probabilidade de voltar para o estado $\sigma = \{+1, -1, +1\}$ da Figura 2.3 é muito alta.

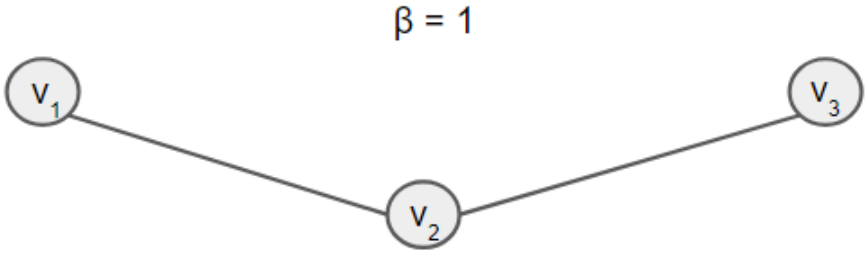


v_1	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_1)$	-1
$q(+1;\sigma,v_1)$	0.993
$q(-1;\sigma,v_1)$	0.007

v_2	
F_v	9
$df_{\sigma,F}$	-1
$S(\sigma,v_2)$	8
$q(+1;\sigma,v_2)$	0
$q(-1;\sigma,v_2)$	1

v_3	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_3)$	-1
$q(+1;\sigma,v_3)$	0.993
$q(-1;\sigma,v_3)$	0.007

Figura 2.3: Exemplo de probabilidades do I-CSMA. Configuração $\sigma = \{+1, -1, +1\}$.

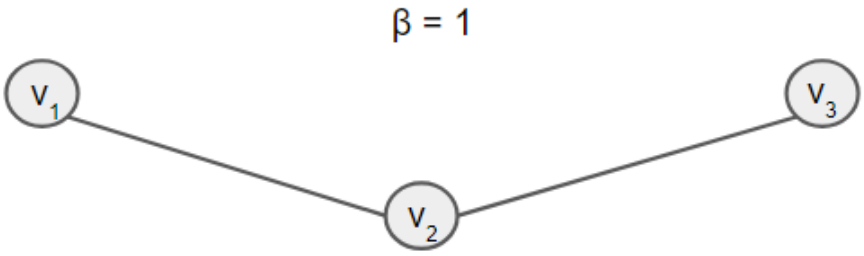


v_1	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_1)$	-1
$q(+1;\sigma,v_1)$	0.993
$q(-1;\sigma,v_1)$	0.007

v_2	
F_v	9
$df_{\sigma,F}$	-1
$S(\sigma,v_2)$	3
$q(+1;\sigma,v_2)$	~0
$q(-1;\sigma,v_2)$	~1

v_3	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_3)$	-1
$q(+1;\sigma,v_3)$	0.993
$q(-1;\sigma,v_3)$	0.007

Figura 2.4: Exemplo de probabilidades do I-CSMA. Configuração $\sigma = \{-1, -1, +1\}$.

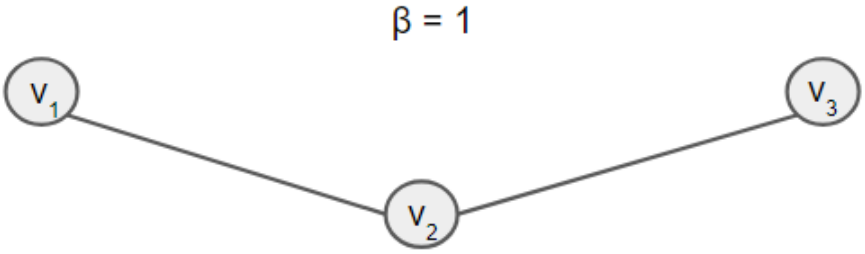


v_1	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_1)$	-1
$q(+1;\sigma,v_1)$	0.993
$q(-1;\sigma,v_1)$	0.007

v_2	
F_v	9
$df_{\sigma,F}$	-1
$S(\sigma,v_2)$	-2
$q(+1;\sigma,v_2)$	~1
$q(-1;\sigma,v_2)$	~0

v_3	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_3)$	-1
$q(+1;\sigma,v_3)$	0.993
$q(-1;\sigma,v_3)$	0.007

Figura 2.5: Exemplo de probabilidades do I-CSMA. Configuração $\sigma = \{-1, -1, -1\}$.

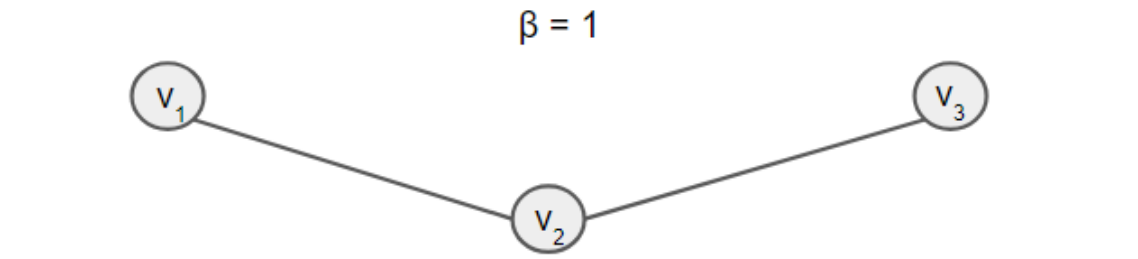


v_1	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_1)$	9
$q(+1;\sigma,v_1)$	0
$q(-1;\sigma,v_1)$	1

v_2	
F_v	9
$df_{\sigma,F}$	9
$S(\sigma,v_2)$	8
$q(+1;\sigma,v_2)$	0
$q(-1;\sigma,v_2)$	1

v_3	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_3)$	9
$q(+1;\sigma,v_3)$	0
$q(-1;\sigma,v_3)$	1

Figura 2.6: Exemplo de probabilidades do I-CSMA. Configuração $\sigma = \{+1, +1, +1\}$.



v_1	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_1)$	9
$q(+1;\sigma,v_1)$	0
$q(-1;\sigma,v_1)$	1

v_2	
F_v	9
$df_{\sigma,F}$	9
$S(\sigma,v_2)$	-2
$q(+1;\sigma,v_2)$	~1
$q(-1;\sigma,v_2)$	0

v_3	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_3)$	9
$q(+1;\sigma,v_3)$	0
$q(-1;\sigma,v_3)$	1

Figura 2.7: Exemplo de probabilidades do I-CSMA. Configuração $\sigma = \{-1, +1, -1\}$.

2.2.2 Problema de atribuição de intervalo em TDMA e o algoritmo DRAND

O problema de atribuição de intervalo em TDMA é formalmente definido em [15]. Dado um grafo de entrada G e definição de conflito, o objetivo é encontrar um intervalo de tempo para cada nó de modo que, se dois nós estiverem em conflito, eles não podem estar no mesmo intervalo de tempo.

Esse problema de atribuição é uma extensão do problema de coloração de vértices, em que o objetivo é colorir um grafo com o menor número de cores possíveis sem que dois nós vizinhos tenham a mesma cor. Vale notar que, dada uma solução do problema, cada conjunto de nós com a mesma cor equivale a um escalonamento TDMA válido. O autor de [15] também apresenta um algoritmo distribuído (DRAND) para solucionar o problema.

O DRAND é uma implementação distribuída do RAND [16], um algoritmo aleatório de atribuição de intervalo. O algoritmo, em linhas gerais, funciona com cada dispositivo da rede podendo estar em quatro estados: *IDLE*, *GRANT*, *REQUEST* ou *RELEASE*. Os dispositivos em estado *IDLE* com uma certa probabilidade tentarão transmitir. Para isso, trocam mensagens com seus vizinhos para negociar a permissão para transmitir em algum intervalo de tempo. As mensagens e os estados servem para garantir que não haverá colisão na transmissão. A Figura 2.8 apresenta o diagrama de estados do algoritmo DRAND. No inicio da seta está a condição para a transição e ao final da seta a ação a ser tomada.

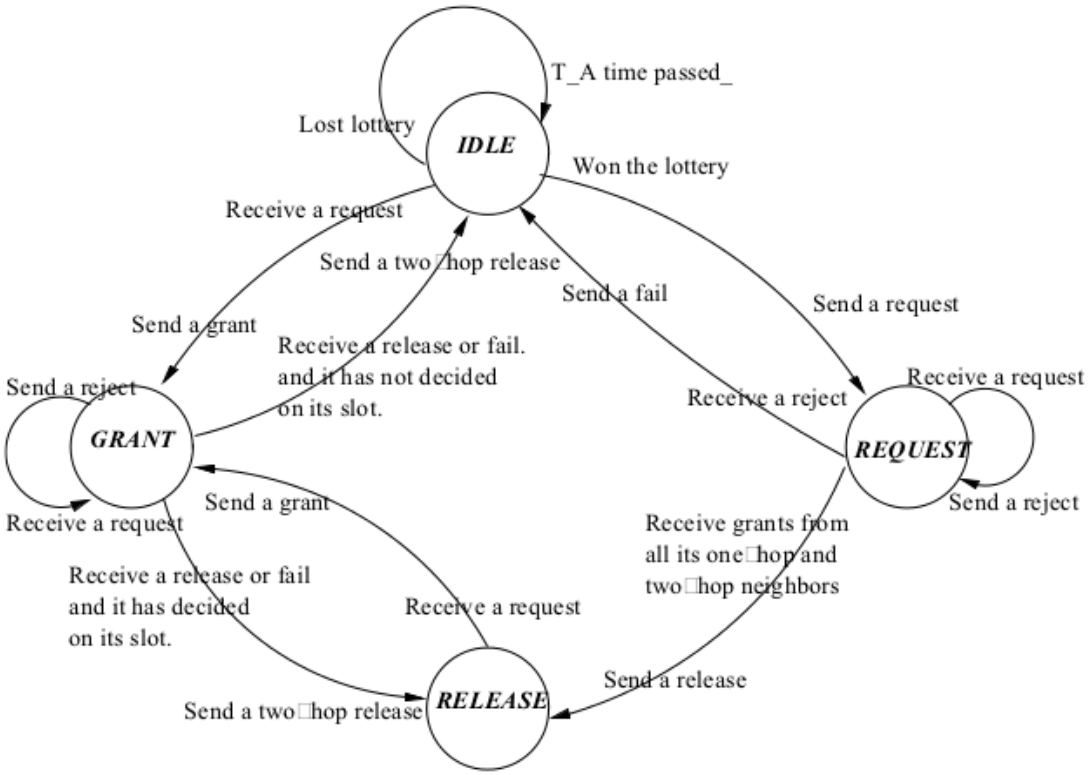


Figura 2.8: Diagrama de estados do algoritmo DRAND. Imagem retirada de [15]
.

Capítulo 3

Proposta

O trabalho de [3] de adaptação do modelo de Ising para o I-CSMA sem restrição dos estados pelos quais a cadeia de Markov pode se mover traz maior liberdade. Neste trabalho, propomos expandir o modelo para considerar a ação do campo externo do modelo de Ising para o problema de escalonamento de enlaces em rede sem fio.

A proposta é motivada pelo problema encontrado com o modelo utilizado pelo I-CSMA em que apenas a interação entre os vizinhos é utilizada. Quando apenas essa interação é considerada, a função de probabilidade tende a prender a cadeia em um estado no qual a energia do sistema é baixa quando comparada a energia dos estados que estão nas proximidades, o sistema fica *preso* nas proximidades de um ponto de mínimo local, fazendo com que o estado do sistema tenha dificuldade de se mover sobre a cadeia de Markov. Para o caso físico é o comportamento esperado, pois o sistema encontrou um ponto onde sair significaria aumentar a energia do sistema, mas para o problema de escalonamento de enlaces isso significa que apenas alguns enlaces serão escalonados em detrimento de outros, causando um problema conhecido como inanição (*starvation*).

Por exemplo, na Figura 2.3, o sistema dificilmente sairá desse estado, e quanto maior o tamanho da fila de v_2, menor a probabilidade de v_2 mudar de estado.

Uma forma de resolver esse problema seria diminuir o valor de β para que o sistema se comporte de forma mais aleatória e, com isso, não se prenda aos mínimos locais da função de energia. Porém, o que nos interessa são justamente os pontos de mínimo local da função de energia. Os estados de mínimo local são locais importantes para o problema. Para destacar esta importância, quando desconsideramos as filas nos enlaces o problema passa a ser encontrar os escalonamentos com os menores números de interferência entre enlaces. Este estados são justamente os estados que têm a menor energia em relação aos seus vizinhos, são os mínimos locais. Portanto, estes estados são a solução do nosso problema, o que é indesejável é ficar preso neles, em apenas uma das muitas possíveis soluções.

Buscamos agora impedir que o sistema fique preso em algum estados de mínimo,

mas permitir que ele encontre este estado. Para isso voltamos a origem do modelo de Ising para incluir o campo externo, que pode cumprir esta finalidade de impedir que o sistema fique preso.

3.1 MICE: Modelo de Ising com Campo Externo

Dado o problema mencionado acima, propomos a utilização do campo externo do modelo de Ising ($h_j, \gamma > 0$) de modo que o crescimento da fila de um enlace interagindo com o campo externo leve o enlace para o estado $\sigma_v = +1$, perturbando o sistema e fazendo com que se mova para outro estado de mínimo local.

Para provocar esse comportamento, a interação do campo externo será proporcional ao tamanho da fila

$$h_j = \alpha df_{\sigma,F}(v) \qquad \alpha > 0 \tag{3.1}$$

Assim, o somatório referente ao campo externo da equação 2.2 será:

$$\gamma \sum_{i \in V} h_i \sigma_i = \gamma\alpha \sum_{i \in V} df_{\sigma,F}(v)^2 \tag{3.2}$$

Para simplificar podemos incorporar α ao γ, $\gamma = \gamma\alpha$. Teremos, então,

$$H_\gamma(\sigma, F) = \sum_{(v,w) \in E} df_{\sigma,F}(v) df_{\sigma,F}(w) - \gamma \sum_{v \in V} df_{\sigma,F}(v)^2 \tag{3.3}$$

A distribuição de probabilidade é a mesma do modelo de Ising, agora considerando o campo externo γ.

$$\mu(\sigma) = \frac{e^{-\beta H_\gamma(\sigma,F)}}{Z(\beta,\gamma)} \qquad Z(\beta,\gamma) = \sum_{\sigma \in \Omega} e^{-\beta H_\gamma(\sigma,F)} \tag{3.4}$$

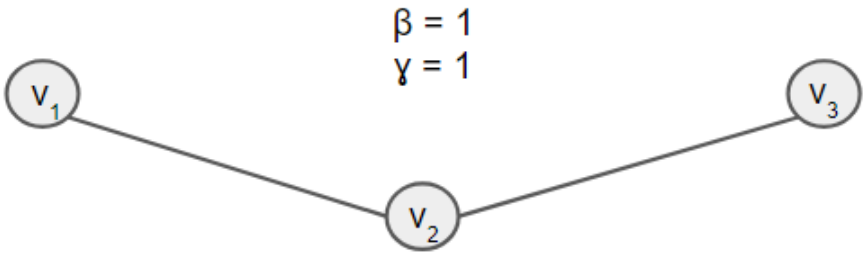
Precisamos agora encontrar as novas equações para $\mu(\Psi^+_{\sigma,v})$ (3.5) e $q(1;\sigma,v)$ (3.6).

$$\begin{aligned} Z(\beta,\gamma)\mu(\Psi^+_{\sigma,v}) = \exp(-\beta(F_v \sum_{w:(w,v)\in E} df(w) - \gamma F_v^2 \\ + \sum_{(k,w)\in E; k,w \neq v} df(k)df(w) - \gamma \sum_{w \in V\setminus\{v\}} df(w)^2)) \\ \mu(\Psi^+_{\sigma,v}) = \frac{e^{-F_v\beta S(v)} e^{+\beta\gamma F_v^2} e^{-\beta H(\sigma\setminus\{v\})}}{Z(\beta,\gamma)} \end{aligned} \tag{3.5}$$

$$
\begin{aligned}
q(1;\sigma,v) &= \frac{\mu(\Psi^{+}_{\sigma,v})}{\mu(\Psi^{+}_{\sigma,v}) + \mu(\Psi^{-}_{\sigma,v})} = \frac{e^{F_v\beta(F_v\gamma - S(\sigma,v))}}{e^{F_v\beta(F_v\gamma - S(\sigma,v))} + e^{\beta(S(\sigma,v)+\gamma)}} \\
&= \frac{1}{1 + e^{\beta((\gamma+S(\sigma,v)) - F_v(F_v\gamma - S(\sigma,v)))}}
\end{aligned}
\tag{3.6}
$$

Esta formulação foi chamada de Modelo de Ising com Campo Externo (MICE).

Voltando ao exemplo de três enlaces v_1, v_2 e v_3, as Figuras 3.1-3.5 ilustram a formulação do MICE nos mesmos cenários das Figuras 2.3-2.7. Vale destacar a diferença entre as Figuras 2.3 e 3.1, onde a probabilidade $q(+1;\sigma,v_2)$ passa de 0 para 0,5. Nas Figuras 2.4 e 3.2 probabilidade $q(+1;\sigma,v_2)$ era de ≈ 0 e agora é igual a 1.

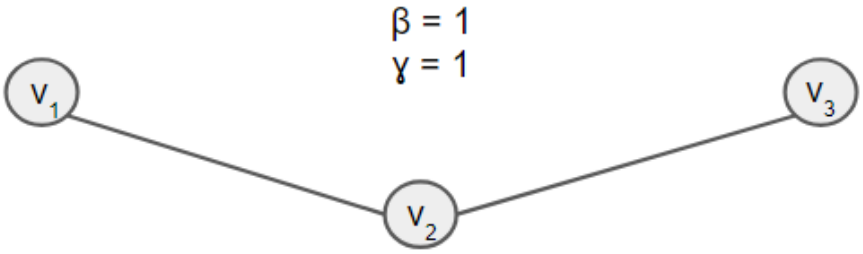


v_1	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_1)$	-1
$q(+1;\sigma,v_1)$	~1
$q(-1;\sigma,v_1)$	~0

v_2	
F_v	9
$df_{\sigma,F}$	-1
$S(\sigma,v_2)$	8
$q(+1;\sigma,v_2)$	0.5
$q(-1;\sigma,v_2)$	0.5

v_3	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_3)$	-1
$q(+1;\sigma,v_3)$	~1
$q(-1;\sigma,v_3)$	~0

Figura 3.1: Exemplo de probabilidades do MICE. Configuração $\sigma = \{+1, -1, +1\}$.

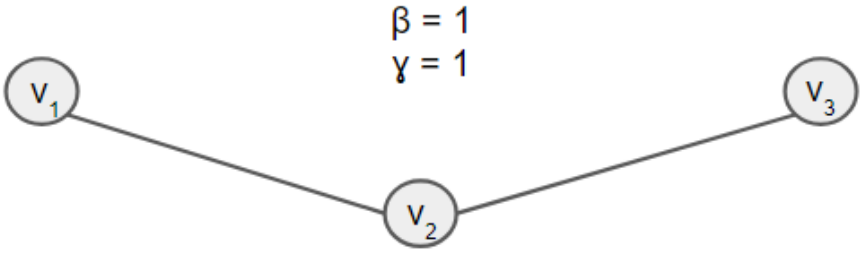


v_1	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_1)$	-1
$q(+1;\sigma,v_1)$	~1
$q(-1;\sigma,v_1)$	~0

v_2	
F_v	9
$df_{\sigma,F}$	-1
$S(\sigma,v_2)$	3
$q(+1;\sigma,v_2)$	1
$q(-1;\sigma,v_2)$	0

v_3	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_3)$	-1
$q(+1;\sigma,v_3)$	~1
$q(-1;\sigma,v_3)$	~0

Figura 3.2: Exemplo de probabilidades do MICE. Configuração $\sigma = \{-1, -1, +1\}$.

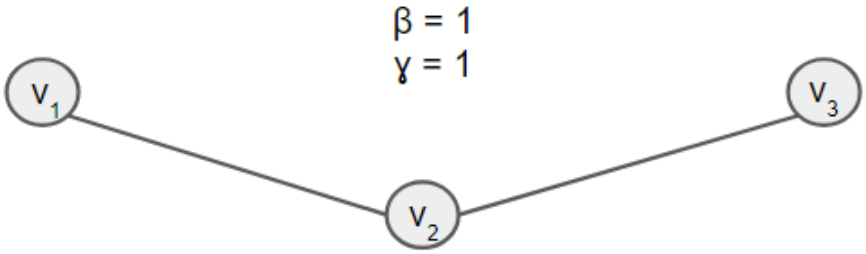


v_1	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_1)$	-1
$q(+1;\sigma,v_1)$	~1
$q(-1;\sigma,v_1)$	~0

v_2	
F_v	9
$df_{\sigma,F}$	-1
$S(\sigma,v_2)$	-2
$q(+1;\sigma,v_2)$	1
$q(-1;\sigma,v_2)$	0

v_3	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_3)$	-1
$q(+1;\sigma,v_3)$	~1
$q(-1;\sigma,v_3)$	~0

Figura 3.3: Exemplo de probabilidades do MICE. Configuração $\sigma = \{-1, -1, -1\}$.

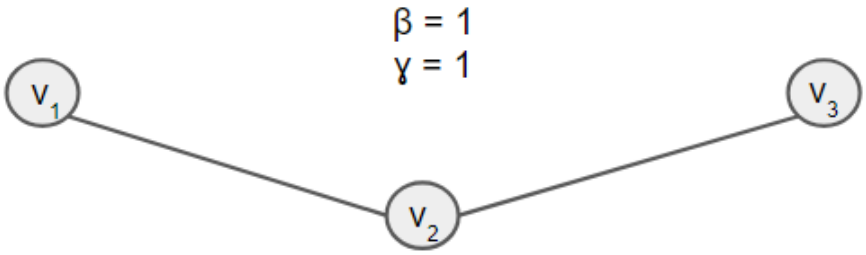


v_1	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_1)$	9
$q(+1;\sigma,v_1)$	~0
$q(-1;\sigma,v_1)$	~1

v_2	
F_v	9
$df_{\sigma,F}$	9
$S(\sigma,v_2)$	8
$q(+1;\sigma,v_2)$	0.5
$q(-1;\sigma,v_2)$	0.5

v_3	
F_v	4
$df_{\sigma,F}$	4
$S(\sigma,v_3)$	9
$q(+1;\sigma,v_3)$	~0
$q(-1;\sigma,v_3)$	~1

Figura 3.4: Exemplo de probabilidades do MICE. Configuração $\sigma = \{+1, +1, +1\}$.



v_1	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_1)$	9
$q(+1;\sigma,v_1)$	~0
$q(-1;\sigma,v_1)$	~1

v_2	
F_v	9
$df_{\sigma,F}$	9
$S(\sigma,v_2)$	-2
$q(+1;\sigma,v_2)$	1
$q(-1;\sigma,v_2)$	0

v_3	
F_v	4
$df_{\sigma,F}$	-1
$S(\sigma,v_3)$	9
$q(+1;\sigma,v_3)$	~0
$q(-1;\sigma,v_3)$	~1

Figura 3.5: Exemplo de probabilidades do MICE. Configuração $\sigma = \{-1, +1, -1\}$.

3.2 Algoritmos de escalonamento

Um problema do algoritmo I-CSMA é o número de colisões durante as fases de controle, que pode ser muito grande dependendo da topologia da rede e de β, deteriorando o desempenho do algoritmo. Para minimizar esse problema, desenvolvemos dois algoritmos distribuídos do tipo TDMA (sem colisões, baseado em [15]) para a segunda parte do problema, ou seja, o mapeamento de σ para um escalonamento viável. Os algoritmos foram baseados nas seguintes ideias:

- Cada nó deve transmitir apenas uma mensagem durante a fase de controle
- Apenas informação local pode ser utilizada
- O algoritmo pode demorar várias fases de controle para terminar
- Não é necessário que o algoritmo termine
  - Se interrompido deve entregar um resultado parcial viável
- Se o algoritmo terminar, o resultado é maximal
- Deve ser possível executar várias instâncias do algoritmo em paralelo

O resultado E é dito maximal quando não existe outro escalonamento viável S que contenha todos os enlaces de E e seja diferente de E.

Podemos desenvolver algoritmos simples para mapear σ em um escalonamento viável. O primeiro algoritmo é chamado de Escalonador Maximal (EsMa):

Escalonador Maximal - EsMa

Inicialização (Entrada inteiro k):

1. Cada nó v escolhe um número aleatório em $[0, M]$, anuncia para seus vizinhos e se inicia no estado **DISPUTA**

Atualização de cada nó v em estado DISPUTA

2. Se o número escolhido por v é o maior entre todos os nós em estado **DISPUTA** da rede
   - Estado de v passa para **GANHADOR**
   - Estado dos vizinhos de v passa para **PERDEDOR**

Finalização

3. Executar k-vezes o passo de atualização (passo 2)

4. Todos os nós em estado **GANHADOR** transmitem

O funcionamento é explicado utilizando o conhecimento global da rede para facilitar o entendimento, mas na prática é necessário apenas conhecimento local. Quando o número escolhido de um nó é o maior entre os seus vizinhos, ele é, ou eventualmente será, o maior da rede, pois nenhum dos seus vizinhos poderá passá-lo para o estado **PERDEDOR**. Assim, pode então tomar a decisão com a informação local dos vizinhos e passar para o estado **GANHADOR** e seus vizinhos para **PERDEDOR**. Para fazer a conexão com o MICE, cada nó v irá escolher um valor em $[0, N]$, caso $df_{\sigma,F}(v) = -1$, e um valor em $[N, M]$, $N < M$ caso contrário.

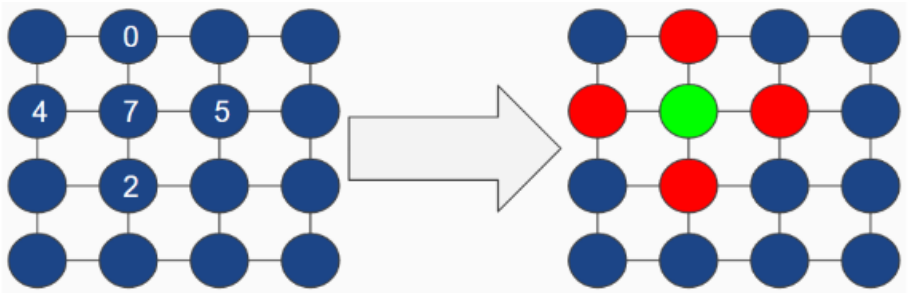


Figura 3.6: Exemplo do algoritmo EsMa. Os nós em azul estão em estado de DISPUTA, o nó em verde GANHADOR e os nós em vermelho PERDEDOR.

A Figura 3.6 ilustra o funcionamento do algoritmo na região numerada, onde o nó com o número 7 ganha a disputa com seus vizinhos, passa para o estado GANHADOR(verde), seus vizinhos para o estado PERDEDOR(vermelho). O nó com número 7 transmitirá depois do algoritmo terminar de ser executado.

O segundo algoritmo busca resolver o problema de atribuição de intervalo, definido em [15] como o problema de encontrar um intervalo de tempo para cada nó ser escalonado sem que nenhum de seus vizinhos seja escalonado no mesmo intervalo. O algoritmo é chamado de Escalonador Maximal para Atribuição de Intervalo (EsMAI), baseado no algoritmo anterior e no algoritmo proposto por [15].

Escalonador Maximal para Atribuição de Intervalo - EsMAI

Inicialização (Entrada inteiro k, inteiro c):

1. Cada nó v escolhe um número aleatório em $[0, M]$, anuncia para seus vizinhos e se inicia no estado **DISPUTA**

Atualização de cada nó v em estado DISPUTA

2. Se o número escolhido por v é o maior entre todos os nós em estado **DISPUTA**

   - Estado de v passa para **GANHADOR**
   - Estado dos vizinhos de v passa para **PERDEDOR**

Escalonamento parcial

3. Executar k-vezes o passo de atualização (passo 2)

4. Todos os nós em estado **GANHADOR** transmitem

Finalização - Atribuição de Intervalo

5. Todos os nós voltam ao estado **DISPUTA**

   - Nós que já foram escalonados escolhem um número em $[0, N], N < M$
   - Nós que ainda não foram escalonados escolhem um número em $[N, M]$
   - Executar o passo de atualização (passo 2)

6. Executar c-vezes o passo de atribuição de intervalo (passo 5)

Assim como no algoritmo EsMa, apesar da explicação utilizar conhecimento global, a implementação precisa de conhecimento apenas local. A conexão com o MICE é feita como no algoritmo anterior, com um adicional de que todo nó v com $df_{\sigma,F}(v) = -1$ começa o algoritmo como se já tivesse sido escalonado, isto é, escolhem um número em $[0, N]$. Para acelerar o término do algoritmo, a condição de ser o maior entre os vizinhos é relaxada e é necessário que seja pelo menos o segundo maior entre os vizinhos, com a condição de que um nó no intervalo $[0, N]$ não pode nunca ganhar de um nó no intervalo $[N, M]$.

Para que em cada intervalo de tempo tenhamos um escalonamento, é necessário executar k instâncias do algoritmo em paralelo, assim em cada intervalo de tempo uma das instâncias terminou o algoritmo. Desta forma o tempo gasto no controle do algoritmo é menor em troca de adicionar um atraso entre o momento em que a configuração do modelo é obtida e em que é realizado o escalonamento relacionado à configuração.

Na Figura 3.7 é ilustrada a execução em paralelo do algoritmo EsMAI. A primeira instância (A1) do algoritmo começa a ser executada no tempo t=0 e tem seu primeiro resultado (resultado11) em t=2. A segunda (A2) instancia começa a ser executa em t=1 e tem o primeiro resultado em t=3. Assim, excetuando t=0 e t=1, para as rodadas subsequentes, como estão sendo executadas várias instâncias do algoritmo em paralelo, sempre há um resultado de alguma instância do algoritmo.

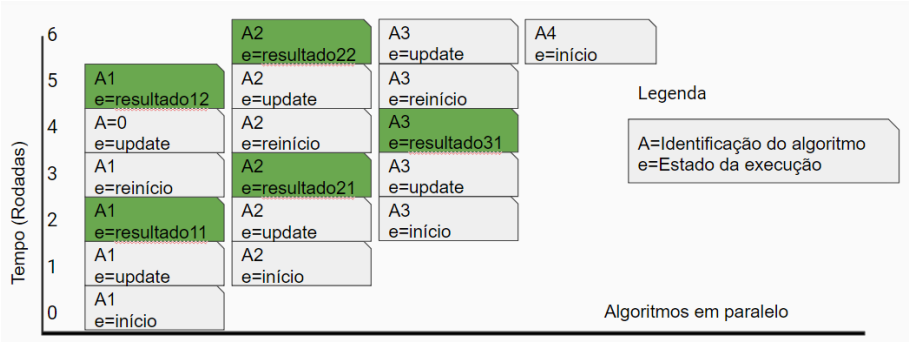


Figura 3.7: Exemplo da execução em paralelo do algoritmo EsMAI.

3.3 Heurísticas para o modelo

O modelo de Ising utiliza a dinâmica de Glauber sobre a cadeia dos estados, o que traz mais complexidade ao algoritmo. Duas heurísticas são propostas por [3]: uma permite que todos os nós atualizem seus estados ao mesmo tempo, sem a necessidade de se formar um conjunto ξ independente. A outra heurística utiliza a função de fila $F_v^H = log(Q_v(t) + 1)$, não utiliza $d*$ como a equação (2.11).

Outra heurística proposta e avaliada neste trabalho é relacionada à função $S(\sigma, v)$, a soma sobre todos os vizinhos. A função pode ter uma variação muito grande de valor dependendo da quantidade de vizinhos do nó, o que faz com que a interação de S com γ seja dependente do número de vizinhos. Para minimizar essa dependência relacionada à topologia do grafo, a função $S(\sigma, v)$ passa a ser a *média* entre todos os vizinhos: $S^H(\sigma, v) = \frac{S(\sigma,v)}{d_v}$, onde d_v é o número de vizinhos de v.

Com isso, podemos nomear os algoritmos de acordo com suas fases e heurísticas: com a formulação do modelo de fila (MICE, I-CSMA ou nenhum), com a utilização do conjunto ξ da dinâmica de Glauber em paralelo (GD) ou não e com o mapeamento para o escalonamento viável (ICSMA, EsMa e EsMAI). Temos, então, os algoritmos da tabela 3.1.

Tabela 3.1: Algoritmos

Algoritmo	Modelo de fila	GD	Mapeamento
I-CSMA	I-CSMA	Sim	ICSMA
MICE-ICSMA	MICE	Sim	ICSMA
MICE-EsMa	MICE	Não	EsMa
MICE-EsMAI	MICE	Não	EsMAI
MICE-GD-EsMa	MICE	Sim	EsMa
MICE-GD-EsMAI	MICE	Sim	EsMAI
EsMa-S/F	Nenhum	Não	EsMa

Na tabela 3.1 os algoritmos EsMa e EsMAI utilizam as funções S^H e F_v^H. Para os algoritmos MICE-GD-EsMa e MICE-GD-EsMAI, o conjunto ξ do nós independentes

é encontrado utilizando o algoritmo EsMa sem levar em consideração o tamanho das filas.

Capítulo 4

Softwares utilizados

Para o projeto, foi desenvolvido um conjunto de programas para obter os resultados dos algoritmos apresentados e propostos através de simulação.

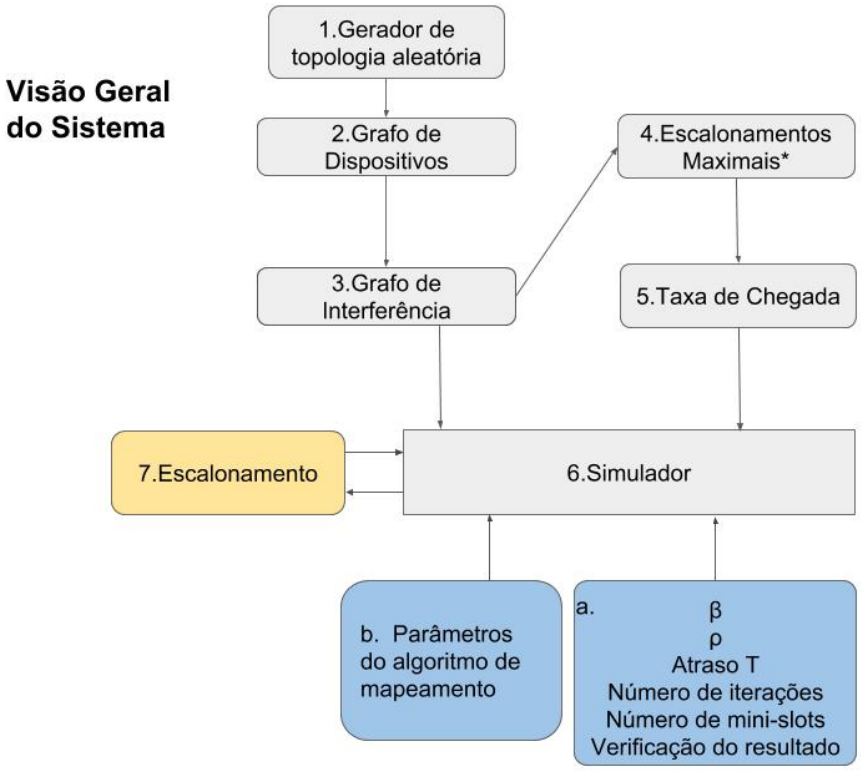


Figura 4.1: Visão geral do simulador implementado

A linguagem escolhida para o desenvolvimento de todos os softwares relacionados ao projeto foi o Python 2.7, principalmente por experiência em programação com a linguagem e pela simplicidade da linguagem que diminui o tempo de desenvolvimento. Pensando também na simplicidade, como o simulador foi desenvolvido para ser executado em um *cluster*, para utilizar bibliotecas adicionais seria necessário instalá-las no *cluster*, então apenas a biblioteca padrão da linguagem foi utilizada.

A visão geral do sistema será dividida em sete passos, como ilustrado na Figura 4.1.

O primeiro passo é gerar uma topologia. Três políticas de aleatoriedade para gerar as topologias foram utilizadas:

- (1) Aleatório com reposição

  Todos os nós são posicionados aleatoriamente dentro de uma área. Caso aconteça de no final algum dos nós não estiver a uma distância d de nenhum outro nó, esse nó é reposicionado novamente de forma aleatória e a verificação é refeita.

- (2) Próximo ao nó anterior

  O primeiro nó é posicionado no centro. Então, os próximos nós são posicionados, um a um, aleatoriamente em um raio de distância de até d de algum dos nós já posicionados anteriormente

- (3) Próximo ao nó anterior com máximo de escolhas

  Igual à política anterior, com a restrição do número de vezes que um nó pode ser escolhido para o posicionamento do próximo.

O segundo passo é construir o grafo de dispositivos. Para cada nó do passo anterior, dois dispositivos são posicionados na mesma coordenada (x, y), de modo que suas coordenadas no eixo z estejam a uma distância fixa, ficando um acima do outro para formar o enlace.

Em seguida, na terceira etapa, é calculado o grafo de interferência. O grafo é construído a partir do grafo de dispositivos. Se algum dispositivo de um enlace estiver na área de interferência de algum outro dispositivo de outro enlace, então existe uma aresta entre esses enlaces. Vale notar que, nessa construção, cada dispositivo pertence a apenas um enlace.

Na Figura 4.2, temos um exemplo de um grafo de interferência gerado pela política (3) de geração da topologia dos dispositivos, com máximo de 3 escolhas. Note que o limitante no número de vezes que o nó é escolhido não limita o número de vizinhos do nó.

Para a quarta etapa, foi utilizado o software de [17] para calcular todos os escalonamentos maximais do grafo de interferência. Com os escalonamentos, foram calculadas as taxas de chegada dos enlaces no quinto passo.

Como tráfego é um elemento abstrato do simulador, podendo representar diversas medidas, por exemplo, pacotes, bytes e bits, a medida que o tráfego assume não é relevante. Consideraremos que o tráfego é gerado em pacotes, sendo enviado um pacote por intervalo de dado, para simplificar a explicação. Assim, as taxas de geração de pacotes são calculadas da seguinte forma:

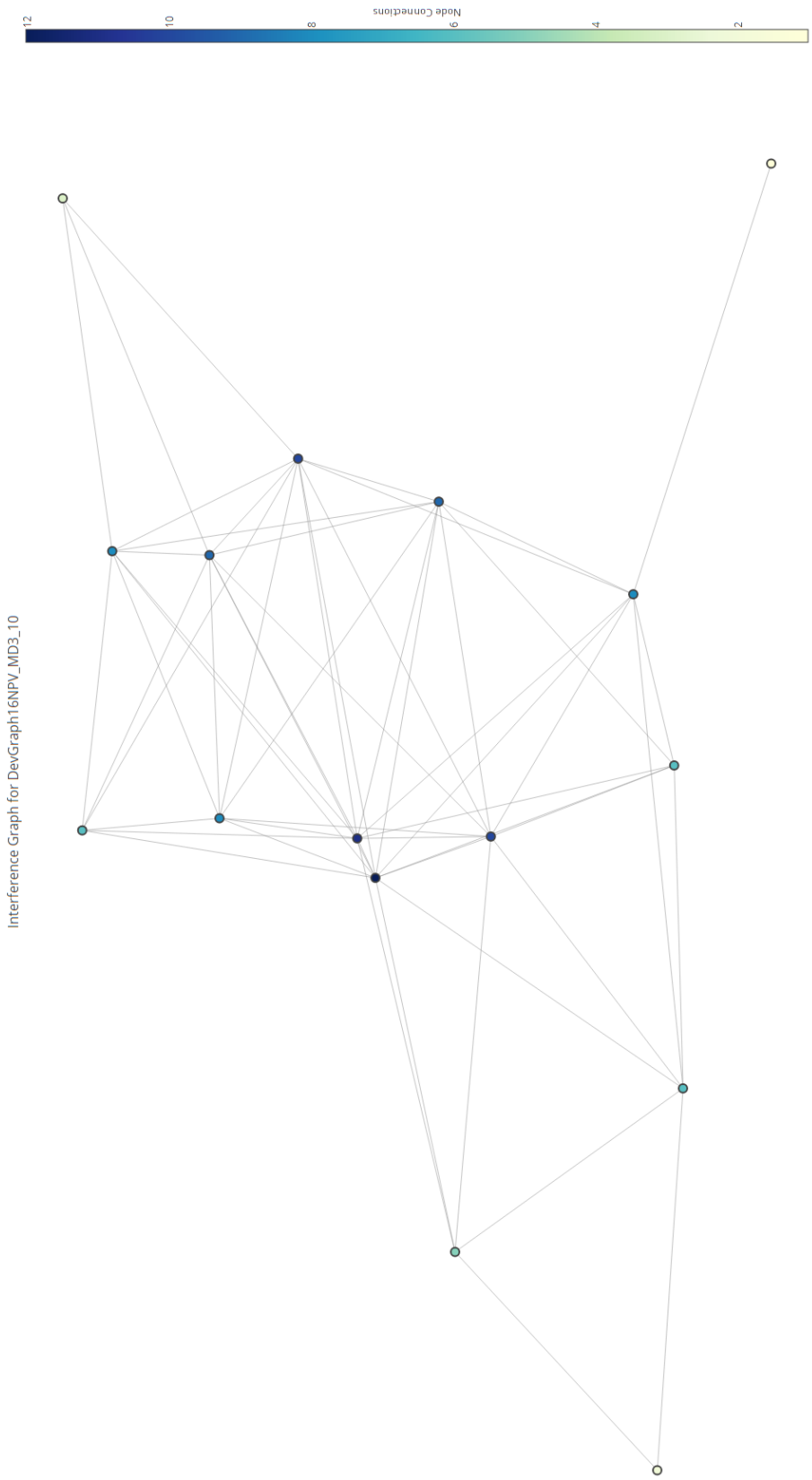


Figura 4.2: Exemplo de uma topologia gerada aleatoriamente. Topologia gerado a partir da política (3) com máximo de 3 escolhas.

- Todos os nós iniciam com peso zero
- Para cada escalonamento maximal do grafo, é atribuído um peso (ou importância)
- Soma-se ao peso de cada nó o peso dos escalonamentos aos quais ele pertence
- Os pesos dos nós são divididos pelo soma dos pesos dos escalonamentos.

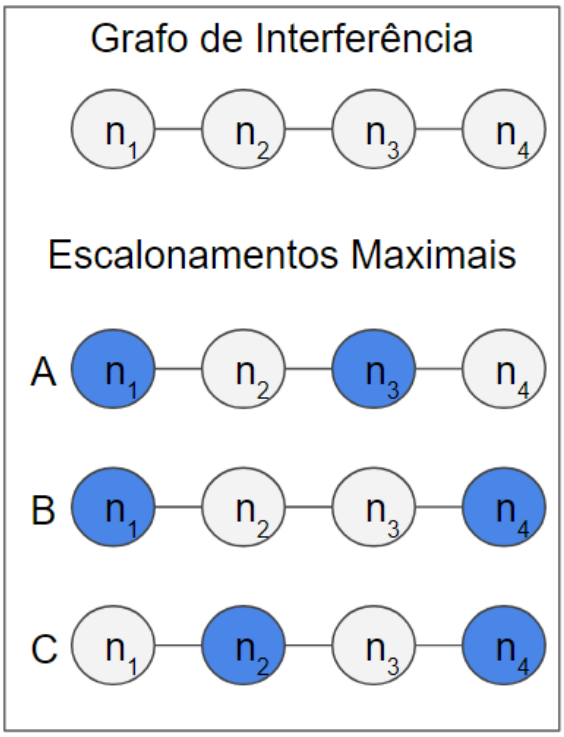


Figura 4.3: Exemplo de escalonamentos maximais.

Por exemplo, na Figura 4.3, suponha que todos os três escalonamentos, $A = B = C = 1$, tenham o mesmo peso. Como os nós n_1 e n_4 aparecem duas vezes cada e os nós n_2 e n_3 aparecem apenas uma vez cada, o tráfego será de $n_1 = \frac{2}{3}, \quad n_2 = \frac{1}{3}, \quad n_3 = \frac{1}{3}, \quad n_4 = \frac{2}{3}$ pacotes por iteração do algoritmo simulado. Note que quando os pesos dos escalonamentos são iguais para todos, o valor do peso é irrelevante. Para pesos diferentes, poderíamos ter pesos $A = 3, \quad B = 0, \quad C = 1$, o resultado da proporcionalidade do tráfego seria $n_1 = \frac{3}{4}, \quad n_2 = \frac{1}{4}, \quad n_3 = \frac{3}{4}, \quad n_4 = \frac{1}{4}$ pacotes por iteração do algoritmo simulado. É importante ressaltar que ambos os resultados estão na borda da capacidade do sistema, esse ponto será mais aprofundado na sessão 5.1.

Os parâmetros do simulador serão divididos em dois conjuntos, os parâmetros do cenário e os parâmetros de algoritmo.

Os parâmetros do cenário são os mesmos para qualquer algoritmo que seja simulado. São eles: o grafo de interferência; um dicionário (estrutura nativa do Python

que faz um mapeamento chave-valor) com a taxa de chegada de cada enlace; o valor de ρ; o número de iterações do algoritmo a ser executado pelo simulador; o valor máximo da média das filas, e fazer a verificação de que o escalonamento resultante do algoritmo é viável.

Os parâmetros específicos dependem de qual algoritmo e versão será simulada (Tabela 3.1), são eles: o valor de β; o valor do campo externo γ; o modelo de fila; o mapeamento; o total de mini-*slots* de controle; a utilização da heurística que substitui a Dinâmica de Glauber; a utilização das funções heurísticas S^H e/ou F_v^H; habilitar uma etapa adicional para a segunda fase de controle do algoritmo I-CSMA e o atraso T referente à implementação da ideia do algoritmo *delayed-CSMA* [6].

Alguns parâmetros do simulador existem por razões de desenvolvimento, corretude e testes feitos durante a pesquisa, portanto, alguns dos parâmetros utilizados foram sempre os mesmos. Ao parâmetro de valor máximo da média das filas é sempre atribuído o valor *False*, desabilitando esse limitante. A verificação de se o resultado é um escalonamento viável é sempre *True*, para garantir que nenhum do escalonamento resultante do algoritmo simulado seja inviável. A etapa adicional da fase de controle do I-CSMA é sempre *False* e o valor do atraso T é sempre 1 (algoritmo sem atraso).

Finalmente o algoritmo é iniciado. A sexta etapa é o simulador, que é essencialmente um simulador de eventos discretos [10] e executa um método MCMC (Cadeia de Markov Monte Carlo), mais precisamente, o algoritmo de Metropolis-Hastings (MH) [18]. Vale ressaltar que, para o caso específico do problema sendo simulado neste trabalho, o estado estacionário da cadeia muda a cada iteração do algoritmo de escalonamento. O simulador executa a seguinte sequência de ações:

- Calcular $q(1; \sigma, v)$

  Adaptado de Eq. 2.5 de acordo com o algoritmo/versão simulada

- Calcular $S(\sigma, v)$

  Adaptado de Eq. 2.10 de acordo com o algoritmo/versão simulada

- Gerar tráfego nas filas dos enlaces

  De acordo com a Sessão 5.1

- Resolver primeira fase de controle

  Modelo de fila e Dinâmica de Glauber(GD) da Tabela 3.1

- Resolver segunda fase de controle

  Mapeamento da Tabela 3.1

- Reduzir as filas dos enlaces escalonados

O simulador retornará o escalonamento da última iteração e tem guardado em suas estruturas outras informações. Algumas dessas informações serão apresentadas no próximo capítulo.

Capítulo 5

Simulações e Resultados

Nesta seção, apresentamos os resultados de desempenho, obtidos via simulação, do algoritmo I-CSMA e dos algoritmos propostos. Simulamos 30 grafos de conflito com 16 nós posicionados aleatoriamente para cada uma das políticas explicadas no Capítulo 4, totalizando 90 grafos de conflito, com cada nó do grafo tendo obrigatoriamente pelo menos um vizinho. A rede é de salto-único, isto é, o pacote é gerado, enviado uma única vez e é removido da rede.

Cada simulação foi executada 5 vezes por 10^5 intervalos de tempo em pelo menos 30 grafos, totalizando entre 150 e 450 simulações. Os resultados coletados foram as médias das filas dos 16 nós (enlaces) do grafo. Todos os resultados estão com intervalo de confiança de 90%. Para os algoritmos EsMa e EsMAI utilizamos $k = 4$ e $c = 4$.

5.1 Geração de tráfego

Os tráfegos da rede foram gerados através de uma distribuição de Pareto limitada. A função de distribuição cumulativa é dada por

$$F(x) = \frac{1 - (L/x)^{\alpha}}{1 - (L/H)^{\alpha}} \tag{5.1}$$

Como sugerido em [13], utilizamos o fator $\alpha = 1.5$, o limite superior $H = 1000$. Já o limite inferior L depende da média da taxa de chegada de cada grafo.

O vetor de chegada λ_G de cada grafo é calculado através dos seus respectivos conjuntos de escalonamentos maximais. Seja $M_G = \{E_1, E_2, ..., E_n\}$ o conjunto de todos os E_i escalonamentos maximais do grafo G, definimos o vetor 16-dimensional s_i para representar o escalonamento E_i. Dada uma ordenação dos nós de G, se o nó $v \in E_i$ então $s_{i,v} = 1$, caso contrário, $s_{i,v} = 0$. Por exemplo, para o escalonamento $E_k = \{1, 4, 7, 11, 16\}$, temos $s_k = (1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1)$. A região de

capacidade do sistema é definida pela combinação convexa do conjunto M_G [8]:

$$\lambda_G = \rho \sum_i w_i s_i \qquad \text{restrito a} \quad \sum_i w_i = 1 \quad \text{e} \quad w_i \geq 0, \ \forall i \tag{5.2}$$

Sendo ρ a carga do sistema e w_i o peso dado ao escalonamento E_i, quando $w_1 = w_2 = ... = w_n = 1/n$, todos os escalonamentos E_i têm a mesma importância. Para este caso, consideramos o tráfego como sendo uniforme. Quando $0 < \rho < 1$, temos que λ_G está dentro da região de capacidade do sistema.

Encontramos o valor de L para fazer com que a média da distribuição de Pareto seja igual à taxa de chegada do sistema. Para tráfegos não uniformes, foram utilizadas distribuições de pesos seguindo a progressão geométrica (5.3).

$$w_i = \frac{r^i}{\sum_{j=0}^{n} r^j} \tag{5.3}$$

5.2 Resultados

Comparamos os algoritmos propostos e o algoritmo I-CSMA. Os valores utilizados foram $\beta \in \{0.01, 0.1, 1, 1.5\}$ e $\gamma \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$, os tráfegos simulados foram o tráfego uniforme e tráfegos não uniformes (5.3) com $r \in \{0.2, 0.6\}$. Os valores da carga do sistema são $\rho \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, mas para algumas simulações, apenas os resultados com maior carga são apresentados.

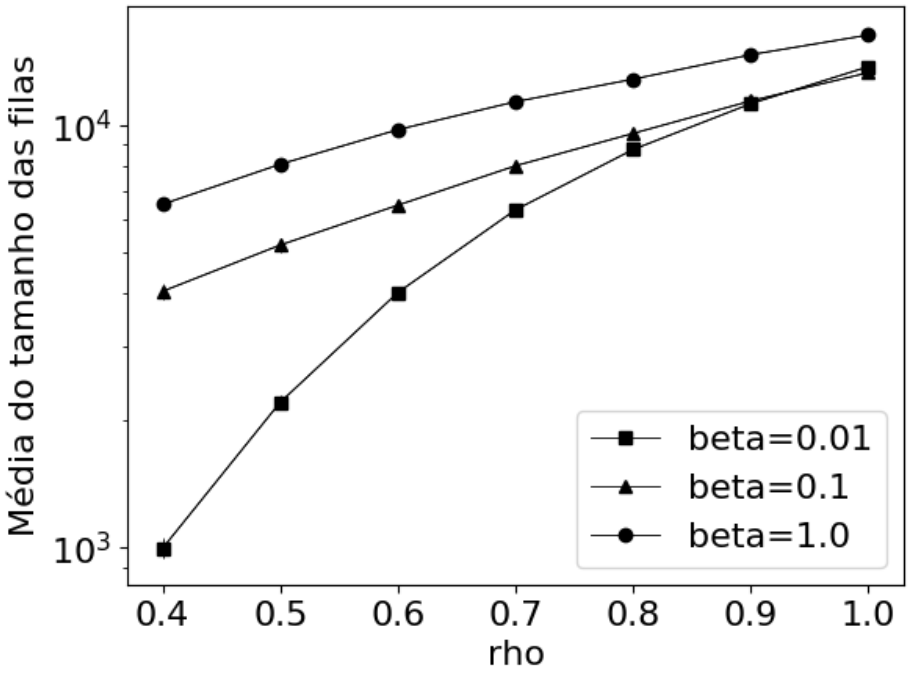
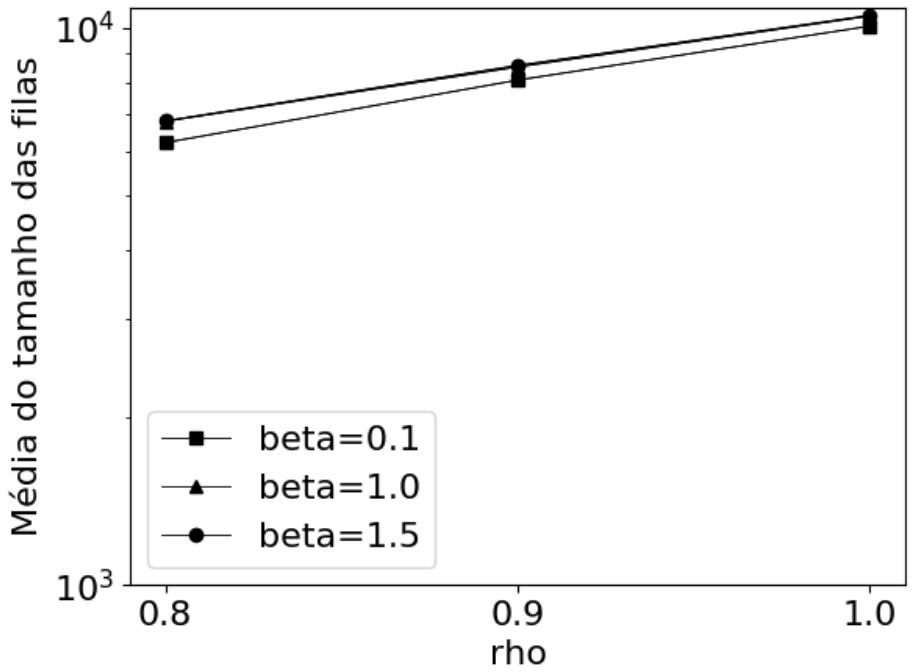
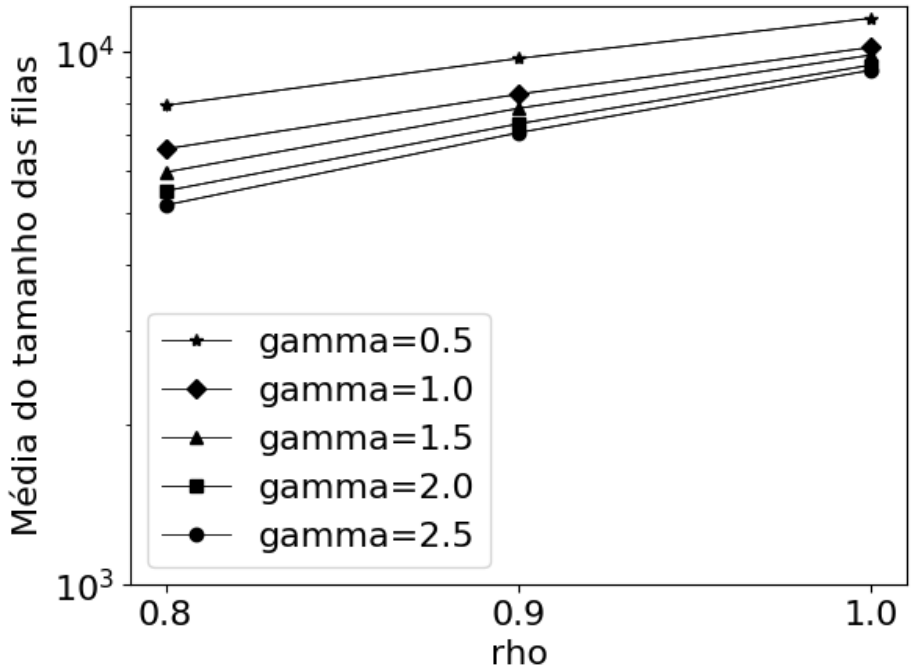


Figura 5.1: Algoritmo I-CSMA para diferentes valores de β



(a) Diferentes valores de β, $\gamma = 1.0$



(b) Diferentes valores de γ, $\beta = 0.1$

Figura 5.2: Algoritmo MICE-ICSMA para diferentes valores de β e γ.

Para o caso do tráfego uniforme com o I-CSMA (Figura 5.1), obtivemos resultados melhores quando diminuímos o valor de β. Já para o MICE-ICSMA (Figura 5.2), os diferentes valores de β têm resultados similares, enquanto aumentar o valor de γ melhora o desempenho do algoritmo. Tornar os algoritmos mais aleatórios melhorou suas performances, isso se deve à característica uniforme do tráfego gerado na rede, fazendo com que seja mais importante que o escalonamento seja maximal do que levar em consideração o tamanho das filas. Existe uma relação do valor de γ com o escalonamento maximal e o número de colisões na fase de controle. Quando aumentamos o valor de γ, estamos aumentando a probabilidade de um enlace ir para $\sigma(t) = 1$ e quando todos os enlaces estão com $\sigma(t) = 1$, o mapeamento do I-CSMA será maximal caso não ocorra colisões. Contudo, quando todos os enlaces estão em $\sigma(t) = 1$, aumentamos a chance de colisões durante a fase de controle.

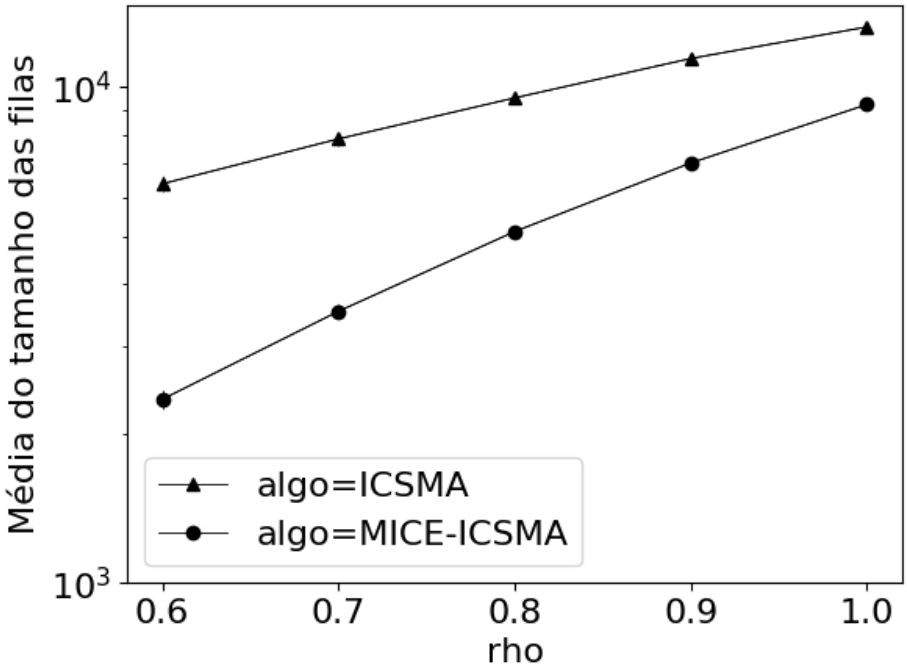


Figura 5.3: Comparação entre MICE-ICSMA e I-CSMA

Na Figura 5.3, os algoritmos foram simulados utilizando os melhores valores encontrados para β e γ de cada algoritmo e tráfego uniforme. A formulação proposta tem ganho de 63% para $\rho = 0.6$ e 30% para $\rho = 1.0$.

Para comparação, também foi simulado o algoritmo EsMa sem levar em consideração o tamanho da fila, EsMa-S/F (EsMa Sem Fila). A Figura 5.4 apresenta o resultado de todos os algoritmos propostos neste artigo com os melhores valores de β e γ encontrados mais o algoritmo EsMa-S/F para referência, todos com o tráfego uniforme. A diferença entre os algoritmos que utilizam o conjunto ξ e os que não utilizam é desprezível. O algoritmo MICE-ICSMA tem a média das filas mais de 400 vezes maior que o algoritmo EsMa-S/F para $\rho = 0.6$, demonstrando a importância

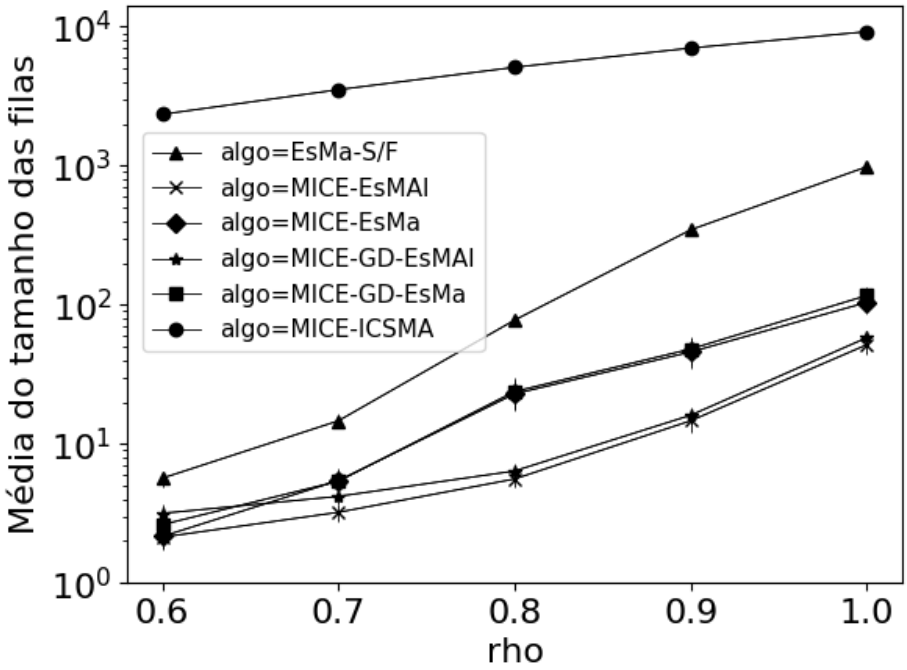


Figura 5.4: Comparação entre os algoritmos propostos

do resultado do mapeamento ser maximal e do algoritmo sem colisões para altas cargas no sistema. Os algoritmos MICE-EsMa e MICE-EsMAI, que consideram a fila para a tomada de decisão, têm performance 86% e 95% melhores, respectivamente, em relação ao algoritmo EsMa-S/F quando $\rho = 0.9$. Fica evidente a importância de considerar o tamanho das filas na tomada de decisão do algoritmo quando o resultado é garantidamente maximal e não há colisões.

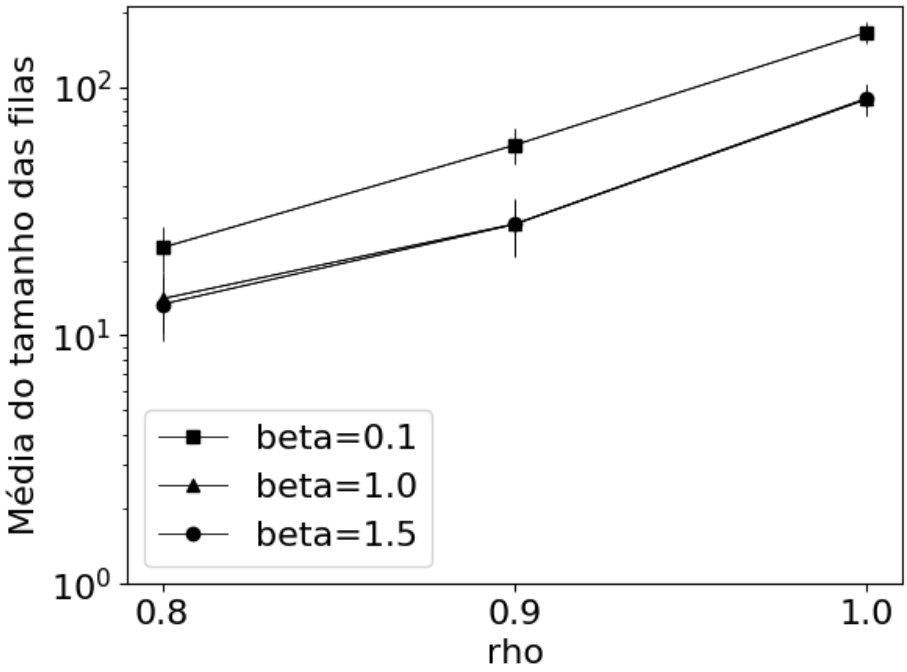


Figura 5.5: Algoritmo MICE-EsMa para diferentes valores de β.

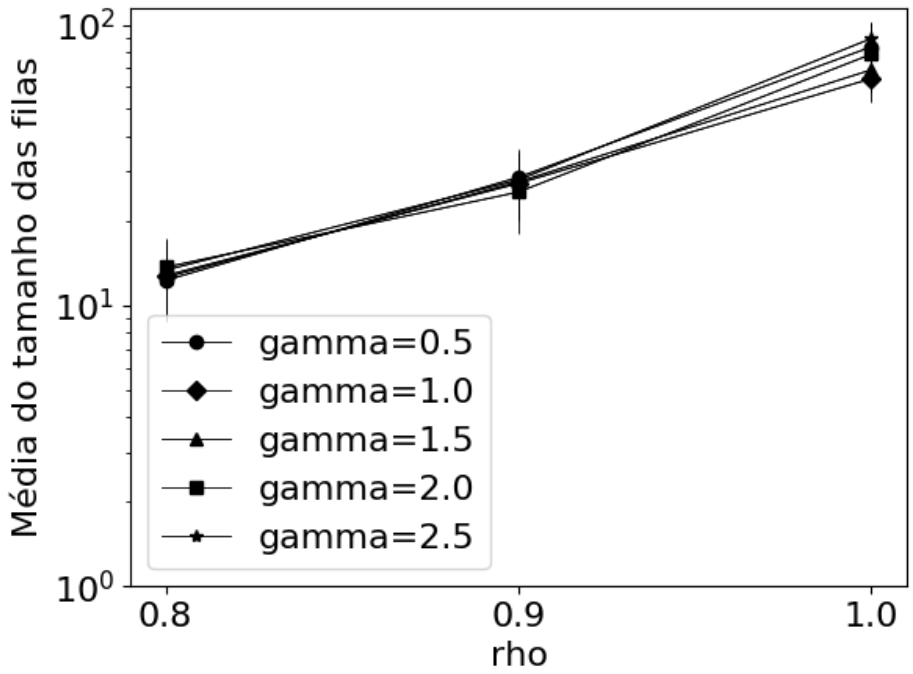


Figura 5.6: Algoritmo MICE-EsMa para diferentes valores de γ.

Na Figura 5.5 com o tráfego uniforme, quando aumentamos $\beta = 0.1$ para $\beta = 1.0$ a performance do algoritmo é melhor (51% para $\rho = 0.9$), isto é, diminuir a aleatoriedade do sistema trouxe resultados melhores. Já na Figura 5.6, também com tráfego uniforme, os diferentes valores de γ não apresentam mudança significativa para os valores simulados.

Também realizamos simulações com tráfegos não uniformes, na Figura 5.7 é apresentado o resultado de todos os algoritmos para $\beta = 1.0$ e $\gamma = 2.5$ quando $r = 0.2$. A diferença entre o MICE-ICSMA e EsMa-S/F é de apenas 7%, para tráfegos não uniformes considerar a fila é tão importante quanto ter um algoritmo sem colisões com resultado maximal. O algoritmo MICE-EsMa é 64% melhor que o EsMa-S/F ($\rho = 0.9$), reforçando a importância de considerar as filas.

O algoritmo MICE-EsMAI está com baixa performance para esse tipo de tráfego, ficando pior que o MICE-EsMa, provavelmente pela escolha de c para este cenário.

O algoritmo MICE-ICSMA apresenta comportamento diferente para $r = 0.2$ em relação ao parâmetro γ quando comparado com o tráfego uniforme, ou seja, a média das filas com $\gamma \in \{1.5, 1.0\}$ são menores quando comparadas com a filas quando $\gamma \in \{0.5, 2.0, 2.5\}$, como visto na Figura 5.8. O resultado difere do resultado obtido para o tráfego uniforme, no cenário não uniforme é preciso que a probabilidade $q(1; \sigma, v)$ se aproxime de 1 de forma mais moderada, e quanto maior o γ mais rapidamente a probabilidade se aproxima de 1.

Para o caso com $r = 0.6$, Figura 5.9, as médias da filas foram ligeiramente menores quando comparadas com $r = 2$, o valor de r mais próximo a 1 deixa o tráfego

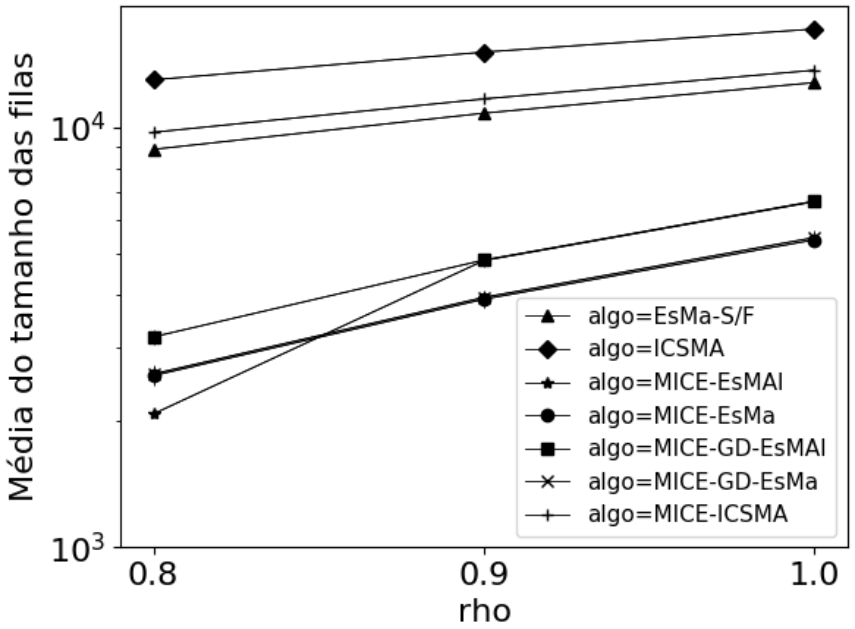


Figura 5.7: Comparação entre os algoritmos, $r = 0.2$

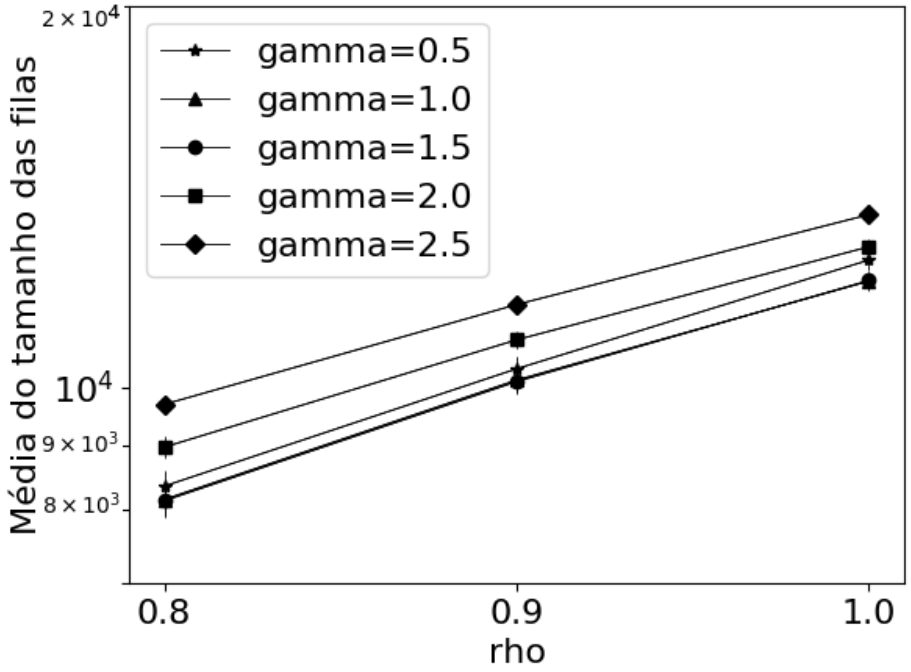


Figura 5.8: Algoritmo MICE-ICSMA para diferentes valores de γ, $\beta = 0.1$, $r = 0.2$

mais próximo do uniforme (quanto mais próximo r for de 1, "mais uniforme"). A ordem de performance foi a mesma da Figura 5.7.

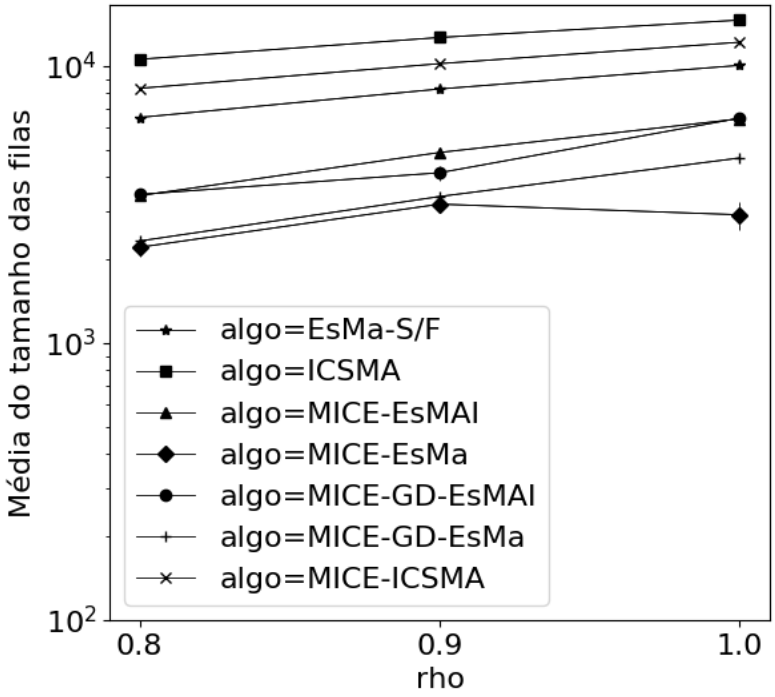


Figura 5.9: Comparação entre os algoritmos, $r = 0.6$

Capítulo 6

Conclusões e Trabalhos Futuros

A partir da contribuição do trabalho de [3], com maior liberdade de mover a cadeia de Markov sobre todo o espaço de configurações, propomos a inclusão do campo externo do modelo de Ising na adaptação para considerar o tamanho da fila dos enlaces. Como dito anteriormente, havia dois problemas na proposta original: mínimos locais e escalonamentos não-maximais.

A inclusão do campo externo busca solucionar o problema de mínimos locais, perturbando o sistema e evitando que a função de probabilidade prenda a cadeia em um único estado. Alteramos o modelo de fila para incluir campo externo e através de simulações obtivemos ganhos significativos, de até 68% para uma carga mais moderada.

Os algoritmos de mapeamento tem por finalidade resolver o segundo problema, o problema dos escalonamentos não-maximais. Propomos dois algoritmos de mapeamento para escalonamentos viáveis, EsMa e EsMAI, que são algoritmos distribuídos que tem como resultado escalonamentos maximais. Conseguimos através de simulações verificar a importância do resultado ser um conjunto maximal, com ganhos chegando a filas mais de 100 vezes menores entre o algoritmo MICE-GD-EsMa/MICE-GD-EsMAI quando comparado com o algoritmo MICE-ICSMA.

Outro resultado interessante é o impacto do conhecimento do tamanho da fila para a tomada de decisão do escalonamento. Para $\rho = 0.6$ o algoritmo MICE-EsMa tem performance 86% melhor que o algoritmo EsMa-S/F, que não leva em consideração o tamanho da fila. Fica evidente, portanto, a contribuição para a melhora da performance a partir do conhecimento da fila, mesmo que apenas conhecimento local.

Os algoritmos EsMa e EsMAI, apesar de terem um atraso entre o tempo em que é calculado e o tempo em que é realizado, apresentaram bons resultados. O algoritmo EsMAI tem performance relativamente melhor nos casos de carga uniforme, o que é esperado, visto que seu mecanismo promove maior igualdade entre os nós. Diminuir o valor de c para valores menores, como $c = 2$, provavelmente melhoraria

sua performance no caso não uniforme, mas poderia piorar o resultado no caso uniforme. Já o algoritmo EsMa, que não se preocupa com igualdade, tem performance relativamente melhor quando a carga do sistema é desbalanceada (não uniforme) quando comparado ao EsMAI.

Como trabalho futuro está provar se, com a inclusão do campo externo, o algoritmo é *throughput-optimal*. Avaliar mais valores de β e γ, que possam produzir resultados ainda melhores e ajudar a compreender melhor sua interação.

Outro ponto interessante que pode ser avaliado é a função do campo externo. Foi usada a função $h_j = \alpha df_{\sigma,F}(v)$, mas é necessário apenas que a função seja monótona crescente. Por exemplo, a função $h_j = \alpha \log(df_{\sigma,F}(v))$ tem o crescimento mais lento e poderia apresentar resultados melhores.

Também seria interessante avaliar a formulação do I-CSMA para os algoritmos EsMa e EsMAI.

Referências Bibliográficas

[1] CISCO. *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. Relatório técnico, Nov 2018.

[2] JIANG, L., WALRAND, J. C. "Approaching Throughput-optimality in Distributed CSMA Scheduling Algorithms with Collisions", *CoRR*, v. abs/1011.3594, 2010. Disponível em: <http://arxiv.org/abs/1011.3594>.

[3] WANG, Y., XIA, Y. "I-CSMA: A Link Scheduling Algorithm for Wireless Networks based on Ising Model", *IEEE Transactions on Control of Network Systems*, v. PP, n. 99, pp. 1–1, 2017. ISSN: 2325-5870. doi: 10.1109/TCNS.2017.2673539.

[4] NI, J., TAN, B., SRIKANT, R. "Q-CSMA: Queue-Length Based CSMA/CA Algorithms for Achieving Maximum Throughput and Low Delay in Wireless Networks". In: *2010 Proceedings IEEE INFOCOM*, pp. 1–5, March 2010. doi: 10.1109/INFCOM.2010.5462229.

[5] JIANG, L., LECONTE, M., NI, J., et al. "Fast Mixing of Parallel Glauber Dynamics and Low-Delay CSMA Scheduling", *IEEE Transactions on Information Theory*, v. 58, n. 10, pp. 6541–6555, Oct 2012. ISSN: 0018-9448. doi: 10.1109/TIT.2012.2204032.

[6] KWAK, J., LEE, C. H., EUN, D. Y. "A High-Order Markov-Chain-Based Scheduling Algorithm for Low Delay in CSMA Networks", *IEEE/ACM Transactions on Networking*, v. 24, n. 4, pp. 2278–2290, Aug 2016. ISSN: 1063-6692. doi: 10.1109/TNET.2015.2458703.

[7] XUE, D., EKICI, E., IBRAHIM, R., et al. "A novel queue-length-based CSMA algorithm with improved delay characteristics", *Computer Networks*, v. 122, n. Supplement C, pp. 56 – 69, 2017. ISSN: 1389-1286. doi: https://doi.org/10.1016/j.comnet.2017.04.036. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128617301676>.

[8] TASSIULAS, L., EPHREMIDES, A. "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks", *IEEE Transactions on Automatic Control*, v. 37, n. 12, pp. 1936–1948, Dec 1992. ISSN: 0018-9286. doi: 10.1109/9.182479.

[9] BISSACOT, R., CIOLETTI, L. "Phase Transition in Ferromagnetic Ising Models with Non-uniform External Magnetic Fields", *Journal of Statistical Physics*, v. 139, n. 5, pp. 769–778, Jun 2010. ISSN: 1572-9613. doi: 10.1007/s10955-010-9961-4. Disponível em: <https://doi.org/10.1007/s10955-010-9961-4>.

[10] STEWART, W. J. *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. Princeton, N.J, Princeton University Press, 2009. ISBN: 978-0-691-14062-9. OCLC: ocn255018592.

[11] BISSACOT, R., CASSANDRO, M., CIOLETTI, L., et al. "Phase Transitions in Ferromagnetic Ising Models with Spatially Dependent Magnetic Fields", *Communications in Mathematical Physics*, v. 337, n. 1, pp. 41–53, Jul 2015. ISSN: 1432-0916. doi: 10.1007/s00220-014-2268-6. Disponível em: <https://doi.org/10.1007/s00220-014-2268-6>.

[12] PADILHA, I. *Estudo das propriedades termodinâmicas no modelo de Ising aleatoriamente decorado com interações competitivas*. Tese de Mestrado, Universidade Federal do Amazonas, 2006.

[13] WANG, Y., XIA, Y. "Improving the queue size and delay performance with the I-CSMA link scheduling algorithm", *Computer Networks*, v. 122, n. Supplement C, pp. 105 – 119, 2017. ISSN: 1389-1286. doi: https://doi.org/10.1016/j.comnet.2017.04.011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128617301408>.

[14] AND Y. XIA. "A distributed CSMA algorithm for wireless networks based on Ising model". In: *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 170–176, Dec 2013. doi: 10.1109/GLOCOM.2013.6831066.

[15] RHEE, I., WARRIER, A., MIN, J., et al. "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks", *IEEE Transactions on Mobile Computing*, v. 8, n. 10, pp. 1384–1396, Oct 2009. ISSN: 1536-1233. doi: 10.1109/TMC.2009.59.

[16] RAMANATHAN, S. "A Unified Framework and Algorithm for (T/F/C)DMA Channel Assignment in Wireless Networks". In: *Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer*

and Communications Societies. Driving the Information Revolution, INFOCOM '97, pp. 900–, Washington, DC, USA, 1997. IEEE Computer Society. ISBN: 0-8186-7780-5. Disponível em: <http://dl.acm.org/citation.cfm?id=839292.842975>.

[17] IECKER, G. *Fractional Edge Coloring for Wireless Link Scheduling In the Physical Interference Model*. Tese de Mestrado, Universidade Federal do Rio de Janeiro, 2018.

[18] MURPHY, K. P. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. Cambridge, MA, MIT Press, 2012. ISBN: 978-0-262-01802-9.

Apêndice A

Implementabilidade

O Apêndice A trabalho de [13] apresenta um algoritmo baseado em dispositivos para a implementação do algoritmo I-CSMA. A seção A.1 a seguir é uma adaptação do apêndice A de [13] e apresenta o passo-a-passo de como transformar o algoritmo I-CSMA algoritmo baseado em enlaces para o algoritmo baseado em dispositivos. Essa transformação, apesar de ter o foco no algoritmo I-CSMA, é genérica. Na seção A.2 traremos as ideias de como implementar os algoritmos propostos.

A.1 Implementação do algoritmo I-CSMA

Em um enlace sem fio temos um dispositivo que atua como transmissor e outro como receptor, podendo fazer apenas um dos papeis por vez, não é possível fazer ambos os papeis ao mesmo tempo. Assumimos que um dispositivo pode ouvir mensagens de difusão (*broadcast*) de seus vizinhos quando não há colisão.

Um par de dispositivos é chamado de vizinho se eles podem ouvir a transmissão um do outro. Definimos $\mathcal{N}(v)$ como o conjunto de vizinhos do dispositivo v. Há uma simetria aqui: para dois dispositivos a e b, $a \in \mathcal{N}(b)$ se e somente se $b \in \mathcal{N}(a)$.

Para qualquer enlace l , seja l_s e l_r o dispositivo transmissor e o dispositivo receptor do enlace l, respectivamente. Cada pacote enviado pelo enlace l consiste em l_s transmitindo o pacote de dados para l_r e l_r respondendo com um pacote de reconhecimento ACK se o pacote de dado foi recebido com sucesso. Para que uma transmissão no enlace l seja bem sucedida, três requerimentos precisam ser satisfeitos no intervalo de tempo:

- Nenhum outro enlace que compartilhe um dispositivo com l pode transmitir.
- Nenhum outro dispositivo em $\mathcal{N}(l_r)$ pode transmitir pacotes de dados. Caso contrário o receptor l_r sofrerá interferência.

- Nenhum outro dispositivo em $\mathcal{N}(l_s)$ pode transmitir ACKs. Caso contrário o transmissor l_s sofrerá interferência no recebimento do seu respectivo ACK.

O conjunto de enlaces que interferem com l é denotado $\mathcal{I}S(l)$, como um resultado dos requerimentos, o conjunto é:

$$\mathcal{I}S(l) = \{i : \text{enlace } i \text{ compartilha um dispositivo com } l, \text{ou } i_s \in \mathcal{N}(l_r), \text{ou } i_r \in \mathcal{N}(l_s)\}. \tag{A.1}$$

Seja $S(v)$ o conjunto de enlaces que tenha o dispositivo v como transmissor e $R(v)$ o conjunto que tenha v como receptor. Podemos escrever o conjunto de interferência de l como:

$$\mathcal{I}S(l) = (S(l_s) \cup R(l_r) \setminus \{l\}) \cup \left(\bigcup_{j \in \mathcal{N}(l_r)} S(j) \right) \cup \left(\bigcup_{k \in \mathcal{N}(l_s)} R(k) \right) \tag{A.2}$$

Agora podemos apresentar a implementação baseado em dispositivos. Assumimos que cada dispositivo v ouve os pacotes de seus vizinhos para aprender sua vizinhança. Portanto, v conhece todos os enlaces em $S(v)$ e $R(v)$ como também o conjunto de interferência de cada enlace $e \in S(v) \cup R(v)$. O dispositivo v tem uma fila para cada enlace $l \in S(v)$, isto é, os enlace no qual ele é o transmissor.

Cada intervalo de tempo consiste em um intervalo de controle e um intervalo de dados. Por eficiência, o intervalo de controle deve ser muito menor que o intervalo de dados. O intervalo de controle ainda é dividido em $M + M'$ mini-intervalos de controle. Os primeiros M mini-intervalos formam a fase de controle I, os M' mini-intervalos formam a fase de controle II.

Durante a primeira fase de controle, os enlaces que podem alterar seus valores em $\sigma(t)$ precisam ser um conjunto independente do grafo de interferência, denotado $\xi(t)$. Isso é alcançado utilizando mensagens de *INTENT*/ACK-I. Se um transmissor l_s pretende incluir o enlace l em $\xi(t)$, ele enviará uma mensagem de *INTENT* para o receptor l_r. Se a mensagem de *INTENT* for recebida com sucesso, o receptor reponderá com uma mensagem de ACK-I. Se a resposta é recebida com sucesso pelo transmissor, o enlace l será incluído no conjunto $\xi(t)$. Depois das etapas 2 até 7, uma nova configuração $\sigma(t)$ é gerada.

Durante a segunda fase de controle, apenas os enlaces l tal que $\sigma_l(t) = 1$ participarão de outra rodada de competição pelo canal. As mensagens de *RESERVE*/ACK-R são utilizadas para resolver o conflitos destes enlaces. A transmissão bem sucedida de um par de mensagens de *RESERVE*/ACK-R permitirá o enlace ganhar o canal para o intervalo atual, enquanto silencia todos os seus enlaces

vizinhos n que também estão no estado $\sigma_n(t) = 1$, se eles existem. No final da segunda fase de controle, um vetor $\sigma'(t)$ livre de conflitos é gerado para a transmissão de pacotes de dados durante o intervalo de dados do tempo t.

Como a fila do enlace l é mantida no transmissor, é mais natural para o transmissor calcular a probabilidade $q(1; \sigma(t-1), l)$. Para isso, o receptor l_r precisa transmitir para o transmissor l_s as informações necessárias, em particular, os estados $\sigma_n(t)$ dos enlaces n vizinhos de l e o tamanho das filas do vizinhos no estado $\sigma_n(t) = 1$. O transmissor e o receptor aprenderam essa informação através das mensagens de *RESERVE*/ACK-R durante a segunda fase de controle no intervalo anterior. Então, o transmissor pode combinar a informação para calcular $q(1; \sigma(t-1), l)$.

No algoritmo baseado em dispositivos, cada mensagem de *RESERVE* e ACK-R do enlace l contém a informação do tamanho da fila do enlace correspondente para que os vizinhos do transmissor e do receptor possam ouvir a mensagem e receber a informação do tamanho da fila de l. Depois dos enlaces no estado $\sigma_v(t) = 1$ difundirem sua respectiva mensagem de *RESERVE*, o dispositivo receberá o estado $\sigma(t)$ e o tamanho da fila de alguns dos seus vizinhos. Informações de tempos anteriores também podem ser usadas caso não tenha informação mais recente.

No algoritmo, cada dispositivo usa duas variáveis para representar o estado do dispositivo nas fases I e II. Elas ajudam a determinar a função do dispositivo. As duas variáveis são:

1. Estado de atualização $US(t) \in \{0, 1, 2, -1, -2, -3\}$;
    - Valor 0: estado de atualização não decidido ou ocioso;
    - Valor 1: transmissor de um enlace em $\xi(t)$;
    - Valor 2: receptor de um enlace em $\xi(t)$;
    - Valor -1: não disponível como transmissor de um enlace em $\xi(t)$;
    - Valor -2: não disponível como receptor de um enlace em $\xi(t)$;
    - Valor -3: não disponível como transmissor ou receptor de um enlace em $\xi(t)$.
2. Estado de transmissão $TS(t) \in \{0, 1, 2, -1, -2, -3\}$;
    - Valor 0: estado de transmissão não decidido ou ocioso;
    - Valor 1: transmissor de um enlace no tempo t;
    - Valor 2: receptor de um enlace no tempo t;
    - Valor -1: não disponível como transmissor de um enlace no tempo t;
    - Valor -2: não disponível como receptor de um enlace no tempo t;

- Valor -3: não disponível como transmissor ou receptor de um enlace no tempo t.

Em cada fase de controle, o dispositivo inicia no estado 0. Se ouvir uma mensagem de ACK de algum de seus vizinhos, ele passará para o estado -2 e não será um transmissor. Se o dispositivo ouvir uma mensagem de *INTENT*/*RESERVE* não endereçada a ele, ele passará para o estado -1 e não será um receptor. Se ambos os eventos acontecem, o estado passará para -3 e o dispositivo não terá nenhuma função na fase de controle. Uma troca bem sucedida de um par de mensagens *INTENT*/ACK-I ou *RESERVE*/ACK-R passará o dispositivo para o estado 1 ou 2, isto é, como transmissor ou receptor de um enlace. Note que um dispositivo pode ter diferentes funções durante as duas fases de controle.

A seguir é apresentado o algoritmo em pseudocódigo, executado por cada dispositivo v no tempo t.

I-CSMA - Algoritmo baseado em dispositivo (no dispositivo v)

Inicialização:

1. No inicio do intervalo, o dispositivo v reinicia as duas variáveis de estado $US(t) = TS(t) = 0$.

Fase de Controle 1 - M Mini-Intervalos: Estabelecer $\sigma_l(t)$

2. v seleciona aleatoriamente um enlace l em $S(v)$.

3. Escolhe aleatoriamente um valor T_1 uniformemente em $\{0, 1, ..., M-1\}$ e inicia um temporizador de T_1 mini-intervalos de controle.

   - Todos os outros enlaces $k \in S(v)$ não serão incluídos no conjunto de atualização $\xi(t)$. Portanto $\sigma_k(t) = \sigma_k(t-1)$.

4. Se v escutar uma mensagem de ACK-I de um dispositivo em $\mathcal{N}(v)$ e se $US(t) = 0$ ou $US(t) = -1$, v define $US(t) = -2$ ou $US(t) = -3$, respectivamente, e define $\sigma_l(t) = \sigma_l(t-1)$.

   - Dispositivo v não transmitirá sua mensagem de *INTENT*.
   - l não será incluído em $\xi(t)$.

5. Se v escutar uma mensagem de *INTENT* de um dispositivo em $\mathcal{N}(v)$ que não seja para ele e se $US(t) = 0$ ou $US(t) = -2$, v define $US(t) = -1$ ou $US(t) = -3$ respectivamente.

- Dispositivo v não será receptor de nenhum enlace em $\xi(t)$.

6. Se v escutar uma mensagem de *INTENT* para ele e se $US(t) = 0$ ou $US(t) = -2$, ele envia um mensagem de ACK-I de volta para o transmissor e define $US(t) = 2$.

   - Dispositivo v será um receptor de um link do conjunto $\xi(t)$.
   - Também define $\sigma_l(t) = \sigma_l(t-1)$.
   - Dispositivo v não transmitirá sua mensagem de *INTENT*.
   - l não será incluído em $\xi(t)$.

7. Se o temporizador T_1 expirar e $US(t) = 0$ ou $US(t) = -1$, v transmite uma mensagem de *INTENT* para o dispositivo l_r no início do mini-intervalo (T_1+1)

   (a) Se v recebe um ACK-I de l_r com sucesso, ele define $US(t) = 1$ e inclui o enlace l em $\xi(t)$.

   (b) v atualiza o estado do enlace l: O dispositivo combina os estados σ e a informação das filas obtidas através da resposta ACK-I de l_r, como também através das mensagens de *RESERVE*/*ACK-R* no intervalo de tempo anterior. Com isso v pode calcular a probabilidade $q(1; \sigma(t-1), l)$;

   (c) Define $\sigma_l(t) = 1$ com probabilidade $q(1; \sigma(t-1), l)$ ou $\sigma_l(t) = -1$ com probabilidade $q(-1; \sigma(t-1), l)$

   (d) Se acontece uma colisão com ACK-I correspondente ou não há ACK-I de resposta de l_r, v define $US(t) = -2$ e continua escutando.

Fase de Controle 2 - M' Mini-Intervalos: Estabelecer $\sigma'_v(t)$

8. Dispositivo v escolhe aleatoriamente um enlace $l \in S(v)$ com $\sigma_l(t) = 1$. Ele então escolhe aleatoriamente um valor T_2 uniformemente em $\{0, 1, ..., M'-1\}$ e inicia um temporizador de T_2 mini-intervalos de controle. Se não existe enlace em $S(v)$, v define $TS(t) = -1$ e não inicia o temporizador. Para todos os outros enlaces $l' \in S(v)$ com $\sigma_{l'}(t) = 1$, v define $\sigma_{l'}(t) = 0$.

   - v ainda escuta

9. Se v escutar uma mensagem de ACK-R e se $TS(t) = 0$ ou $TS(t) = -1$, v define $TS(t) = -1$ ou $TS(t) = -3$, respectivamente.

   - v não transmitirá pacotes no intervalo de dados no tempo t.

10. Se v escutar uma mensagem de *RESERVE* não direcionada a ele e se $TS(t) = 0$ ou $TS(t) = -2$, v define $TS(t) = -1$ ou $TS(t) = -3$, respectivamente.

- v não será receptor no intervalo de dados no tempo t.

11. Se v escutar uma mensagem de *RESERVE* direcionada a ele, ele mande uma mensagem de ACK-R de volta para o transmissor. A mensagem de ACK-R repete a informação da fila transportada na mensagem de *RESERVE*. Se $TS(t) \in \{0, -2\}$, v define $TS(t) = 2$.

    - v será receptor de um enlace que transmitirá no intervalo de dados no tempo t.

12. Se o temporizador T_2 expirar e $TS(t) \in \{0, -1\}$, v envia uma mensagem de *RESERVE* para l_r contendo a fila atual do enlace l.

    - Se v escutar uma resposta ACK-R de l_r, v define $TS(t) = 1$ para o link l
        - Enlace l transmitirá um pacote de dados e o dispositivo v será o transmissor do enlace l no intervalo de dados no tempo t.
    - Se a mensagem de ACK-R correspondente não for recebida de l_r, o dispositivo v define $TS(t) = -2$ e continua escutando.

Intervalo de Dados:

13. Se $TS(t) = 1$, o enlace v transmite um pacote pelo enlace.

A.2 Adaptação para os algoritmos propostos

O algoritmo apresentado acima pode ser adaptado para ser utilizado para os algoritmos propostos. Para o algoritmo MICE-ICSMA basta utilizar a função $q(+1; \sigma, v)$ e $q(-1; \sigma, v)$ (Equação (3.6)) adequada.

Para o algoritmo baseado em TDMA EsMa, precisamos encontrar para as fases de controle I e II um conjunto estritamente ordenável $\mathcal{S}$ de escalonamentos de modo que todo enlace pertença pelo menos a um escalonamento e que dois enlaces em uma vizinhança de dois saltos (vizinho dos vizinhos) não pertençam ao mesmo escalonamento. Um conjunto trivial são dos escalonamentos onde apenas um enlace é escalonado, este conjunto funciona para todas as topologias. Podemos utilizar difusão de mensagens por toda a rede, visto que esse procedimento precisar ser realizado apenas na inicialização do sistema. Encontrar um conjunto de escalonamentos onde apenas um enlace é escalonado por vez é equivalente a encontrar uma boa ordenação estrita dos enlaces.

Para tal basta que cada enlace v escolha um número n_v aleatório dentro de um intervalo grande para evitar colisão na escolha, caso haja colisão é necessário executar o algoritmo novamente. Após escolher os número os enlaces irão difundir

suas escolhas pela rede até que todos os enlaces tenham conhecimento da escolha de todos os outros enlaces. Por fim cada enlace ordena os números $(n_v)_{v\in V}$, assim temos uma ordenação dos enlaces.

Teremos então um vetor ordenado de números aleatórios escolhidos pelos enlaces. Agora é necessário pseudo-aleatorizar o vetor para cada tempo t e para cada fase de controle, de modo que em qualquer fase de controle em qualquer tempo t todos os dispositivos tenham a mesma ordem de $\mathcal{S}$. Podemos utilizar sementes para geração ordenações aleatórios iguais para todos os dispositivos ou podemos fazer isso com permutações, isto é, funções bijetivas de $\mathcal{S}$ em $\mathcal{S}$ que sejam conhecidas por todos os dispositivos da rede.

Por fim, cada um dos escalonamentos de $\mathcal{S}$ será utilizado em um mini-intervalo de controle durante a fase de controle para encontrar $\xi(t)$ e e o mapeamento $\sigma'(t)$. O próprio algoritmo EsMa é utilizado para encontrar os conjuntos independentes $\xi(t)$, $\xi(t+1)$,..., $\xi(t+k')$ e para fazer os mapeamentos $\sigma'(t-1)$, $\sigma'(t-2)$,..., $\sigma'(t-k)$ simultaneamente, k e k' são os parâmetros de entrado do EsMa (k' pode ser grande para aumentar as chances ou garantir a finalização do algoritmo sem nenhuma perda). O algoritmo de mapeamento EsMAI é análogo ao algoritmo EsMa. Os algoritmo que não utilizam a dinâmica de Glauber não precisam encontram $\xi(t)$.