Universidade Federal do Rio Grande do Norte
Grafos

Conjuntos dominantes em Redes sem fio

Discentes:
Irene Ginani Costa Pinheiro e João Emmanuel Izidio da Silva
Docente:
Elizabeth Ferreira Gouveia

Natal - RN
2017

Conteúdo

1 Introdução

Ao longo dos anos temos observado a quantidade de desastres naturais e guerras acontecendo por todo o planeta. Em situações tão adversas sabemos o quanto é necessário que a comunicação entre os pontos estratégicos, tanto para participar de uma ação de resgate quanto para participar de uma ação militar, sejam rápidos e eficientes. Porém em lugares de difícil acesso e onde temos uma infraestrutura precária para possam ser montadas bases de acesso e transmissão de informações se torna mais difícil garantir essa eficácia.

Dessa forma se faz necessário a utilização da comunicação sem fio através de redes do tipo ad hoc (1), que não precisam de uma administração centralizada e nem um suporte mais especializado, pois cada ponto de receptação e transmissão de informações funciona como um roteador para o próximo ponto, o que é denominado comunicação de multi-saltos e por isso esse tipo de topologia é a mais indicada para essas situações, por não necessitarem de um suporte mais específico.

Assim, com esse tipo de rede, podemos modelar os pontos de comunicação em um grafo de disco unitário, onde o disco de cada vértice representa a abrangência do sinal transmitido, e assim é possível aplicar processos os quais fazendo uso de conjuntos dominantes poderão realizar a comunicação de forma mais eficiente e rápida para essas localidades onde a infraestrutura é precária (2).

Para que possamos evidenciar melhor a solução do problema temos que nosso trabalho será dividido entre: problematização, onde teremos a especificação do problema, sua modelagem, onde teremos como nosso problema pode ser modelado em um grafo e por fim as formas de resolvê-lo, onde terremos os tópicos de solução do problema atravéz do uso das redes e o a solução do subproblema encontrado onde dado um grafo teríamos que achar o conjunto dominante mínimo presente nele.

2 Problematização

Atualmente com a globalização e evolução da tecnologia da informação faz-se necessário que as notícias e atualizações a respeito de algum fato sejam transmitidas com cada vez mais eficácia. Em lugares onde a infraestrutura é escassa ou precária é necessário usar de artifícios para que essa comunicação possa ser feita. Quando desastres naturais, acidentes de grande magnitude ocorrem ou temos um campo de concentração militar é preciso trocar uma certa quantidade de informações para que as pessoas que estejam fora dessa ambiente ainda possam ser informadas do que é preciso fazer para melhorar a situação dos que estão vivendo nesses ambientes, como sobreviventes ou soldados.

Para estabelecer uma comunicação em locais com esse tipo de acesso são utilizadas redes do tipo ad hoc que são redes que dispensam o uso de um ponto de acesso comum aos computadores conectados a ela, de modo a que todos os dispositivos da rede funcionam como se fossem um roteador, encaminhando comunitariamente informações que

vêm de dispositivos vizinhos. Assim podem ser montadas de maneira rápida e fácil além de permitirem que outros dispositivos se conectem também facilmente. Dessa forma é possível ter diferentes topologias de rede em momentos distintos dado a entrada, saída ou movimentação de algum dispositivo que faz parte da rede.

Para que a comunicação seja feita é necessário que cada componente conectado a rede seja configurado como um cluster (3), para que os computadores possam trabalhar em conjunto para processar as informações.

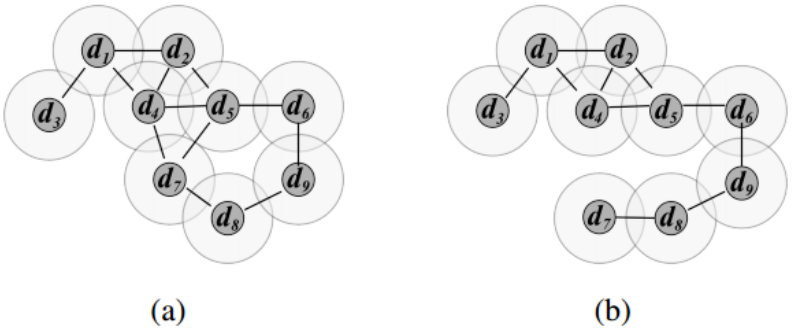

Figura 1: Momentos distintos da topologia de rede

A figura 1 mostra que mesmo com os clusters ainda teremos um problema na comunicação, pois no momento (a) o dispositivo d7 se comunica facilmente com o d4, porém no momento (b) temos a locomoção do dispositivo d7 e para se comunicar com o dispositivo d4 teríamos uma transmissão mais demorada já que a comunicação se dá através dos multi-saltos.

3 Modelando o problema

Ao observarmos novamente a figura 1, temos que a rede foi modelada em um grafo de disco, onde a área do disco represeta a abrangência do sinal daquele dispositivo, dessa forma se uma área intercepta a área de outro vértice implica que aqueles dois podem ter uma comunicação, ou seja, podem transmitir dados um para o outro. Dessa forma o grafo é modelado de acordo com a localização dos dispositivos dado um momento qualquer.

Devido o caráter de urgência das situações onde redes ad hoc são instaladas é necessário que a transmissão de dados entre os dispositivos seja feita da forma mais rápida possível, para isso foram usados métodos para melhorar clusterização (transformação dos dispositivos em clusters) (4), nos quais os dispositivos são divididos em subgrupos, sendo que cada subgrupo possui uma determinada estação responsável pela transmissão de mensagens entre o subgrupo e essas estações são denominadas cluster heads, mas tudo a nível de abstração (configurações de redes e softwares) já que esse tipo de rede não possui infraestrutura.

Para realizar esse processo é utilizado o conceito de conjunto dominante (5), que é

um subconjunto de vértices de um grafo, de modo que cada vértice do grafo ou pertence ao conjunto ou é adjacente de pelo menos um vértice do conjunto. Como observamos na figura 2, onde os vértices de coloração preta formam um conjunto dominante para o grafo.

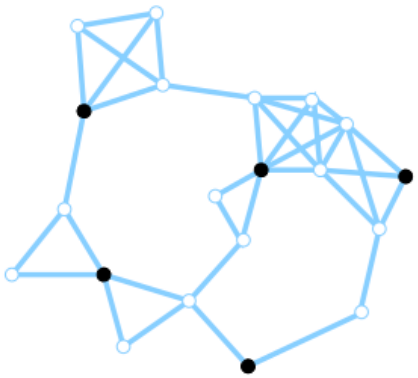

Figura 2: Exemplo de Conjunto Dominante

Dessa forma, depois que a rede é modelada como um grafo de disco, tentamos achar o conjunto dominante onde dado um grafo G = (V, E) modelando uma rede de comunicação ad hoc, onde os vértices são os dispositivos e as arestas são as conexões entre elas, existe um subconjunto S de V que forma um conjunto dominante deste grafo, no qual cada vértice de S representa um cluster head e a sua vizinhança, que são todos os dispositivos que se conectam a um outro, os clusters, que receberão as informações não mais através da comunicação de multi-saltos, mas agora diretamente do cluster head, o que torna a comunicação mais eficiente e rápida, solucionando o problema da figura 1, já que teremos as informações centralizadas e assim, o caminho da inforamção se torna mais curto, como evidênciado na figura 3.

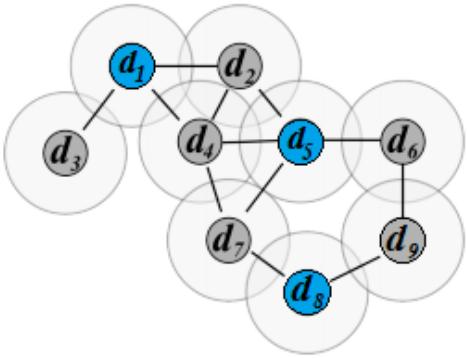

Figura 3: Exemplo de conjunto dominante em uma rede ad hoc, onde d_1, d_5, e d_8 são cluster heads e seus respectivos clusters são (d_2, d_3, d_4), (d_2, d_4, d_6, d_7) e (d_7, d_9)

4 Algoritmos para a construção de Conjuntos Dominantes

Como modelado, temos que o problema descrito tem como principal solução encontrar o menor conjunto dominante disponível em um grafo, dessa forma sabemos que esse é um problema NP-Completo provado por uma redução para o problema de cobertura de vértices que é um dos problemas de Karp onde podemos encontrar a prova em (6), assim não existe, para um grafo qualquer, um algoritmo em tempo polinomial que o resolva. Assim podemos usar algoritmos de aproximação que cheguem a um valor próximo ao conjunto dominante mínimo, ou teremos algoritmos exatos mas que não chegam em um resultado por possuir uma complexidade exponencial.

4.1 Algoritmos Exatos

Ao provarmos que achar o menor conjunto dominante de um grafo é NP-Completo temos que é efetuada uma redução para o problema de cobertura de vértices, temos assim os algoritmos estudados pelo grupo utilizam essa propriedade para encontrar o menor conjunto dominante. Também observamos resultados em grafos densos, grafos cordais e grafos circulares.

4.1.1 Algoritmo Exato de Johan M.M. van Rooij e Hans L. Bodlaender

O primeiro algoritmo estudado foi feito por Johan M.M. van Rooij e Hans L. Bodlaender, na entrada do algoritmo teremos o conjunto com o universo do grafo(vértices e arestas) e um conjunto com todas as coberturas de vértices disponíveis para aquele grafo. Assim no começo do algoritmo escolhemos a cobertura que possui a maior cardinalidade do conjunto de coberturas de entrada e a chamamos de S, então criamos dois novos conjuntos C_1 e C_2 onde, o primeiro conjunto é a união de S com a chamada recursiva da função sem o S nos parâmetros de entrada e o segundo conjunto é a chamada recursiva para o conjunto de coberturas sem a cobertura S e o universo total. Dessa forma a saída do algoritmo é a menor conjunto entre C_1 e C_2 achando assim o conjunto dominante mínimo, algumas verificações são feitas para que sua complexidade caia em algumas casas decimais, mas ao fim das reduções de complexidade com essas verficações temos que sua complexidade é $\mathcal{O}(1.4969^n)$(7)

É importante ressaltar que o algoritmo é modelado para instâncias de coberturas de vértices e ao operarmos com esses termos teremos o conjunto dominante mínimo. Observe que isso é possível devido a uma redução feita entre os dois problemas, notamos isso no algoritmo e também na sua prova para que seja um problema NP-Completo

4.1.2 Algoritmos exatos para algumas famílias de grafos

Nessa seção iremos mencionar algumas mudanças de complexidade e de estrutura no algoritmo quando observamos alguns grafos específicos. Como no algoritmo obtido anteriormente temos que usaremos a cobertura de vértices para achar algoritmos exatos para encontrar o menor conjunto dominante. Para esse problema o algoritmo em questão explora duas propriedades: vértices com graus eleados de forma que eles tenham muitos vizinhos e efetuar a árvore de decomposição. Assim teremos um algoritmo feito de maneira geral e para cada caso especial temos melhorias que fazem com que a complexidade do algoritmo diminua assim, para o algorito geral temos:

Algorithm DS-HighDeg-SmallTw(a graph $G = (V, E)$)
Input : A graph G fulfilling the conditions of Theorem 3.3.
Output: The domination number $\gamma(G)$ of G.

$\lambda \leftarrow \lambda(c, \alpha)$ `/* the value of λ is given in Theorem 3.3 */`
$X \leftarrow \{u \in V : d(u) \geq \lambda n/c\}$
if $|X| \geq \lambda n/c$ **then**
  use the algorithm of Theorem 3.1 and return the result
else
  use the algorithm of Theorem 2.5 and return the result

Figura 4: Algoritmo Geral para Conjunto Dominante

Onde λ(c,α) = $\frac{2}{\frac{1+d}{c}+d}$ e d = $\frac{1}{\log_4(\alpha)}$, de forma que as afirmações e teoremas são provados em (8). É importante ressaltar ainda os resultados dos teoremas citados no algoritmo onde o teorema 3.1 tem como resultado que se temos um algoritmo que calcula a cobertura de vértices para qualquer universo de um grafo, então teremos um algoritmo que calcula o conjunto dominante mínimo em $\mathcal{O}(\alpha^{2n-t(n)})$, onde n é o número de vértices e t(n): $\mathbb{N} \to \mathbb{R}$, também demonstrado em (8). Assim teremos melhorias para cada família já citada de forma que são feitos corolários e teoremas demonstrados em (8) para que possamos chegar a esses resultados, dessa forma temos que para cada família é obtida a complexidade:

Tabela 1: Melhorias nas Complexidades

Classe de Grafos	**Complexidade**
Grafos Densos	$\mathcal{O}(1.2273^{n(1+\sqrt{1-2c})})$
Grafos Cordais	$\mathcal{O}(1.4124^n)$
Grafos Circulares	$\mathcal{O}(1.4887^n)$

Como podemos observar uma complexidade relativamente melhor do que a apresentada no primeiro algoritmo. Por fim é importante definir que na família de grafos perfeitos em algumas subfamílias é possível achar uma resolução polinomial para o problema, além

disso temos que em algums famílias de árvores também podemos resolver esse problema em tempo polinomial. Observamos melhor essas provas e ainda provas sobre encontrar o conjunto dominante mínimo ser NP-Completo em algumas outras famílias de grafos em (9), além disso também podemos encontrar os algoritmos e suas provas de corretudes para essas famílias de grafos nesse mesmo artigo.

4.2 Algoritmos Aproximados

Para que possamos resolver encontrar o menor conjunto dominante de um grafo temos que é preciso então usar heurísticas de aproximação de forma que possamos chegar ao resultado mais aproximado da resposta ideal. Dessa forma estudamos também alguns algoritmos de aproximação usando heurísitcas gulosas para que fosse encontrado o conjunto dominante mínimo.

4.2.1 Algoritmo aproximado de David S. Johnson

O algoritmo guloso proposto por Johnson (10) é uma aproximação para o problema de Conjunto Cobertura (CC) e podemos utiliza-lo (com algumas mudanças) para o nosso caso, pois o problema de CC é intimamaente relacionado com a construção de um Conjunto Dominante Minimo (11). Dado um grafo G = (V, E), ele funciona da seguinte forma:

1. O vértice pertencente a V com maior grau é adicionado ao Conjunto Dominante e seus vizinhos são marcados como visitados.
2. A instrução 1 é repetida até não existir nó livre disponível.

A solução proposta de aproximação tem complexidade de $\mathcal{O}(1 + \ln \Delta)$ onde Δ é o grau máximo do gráfico.

5 Algoritmos para a construção de Conjuntos Dominantes em Redes Ad Hoc

Para se obter o melhor desempenho para transmissão de dados na rede temos que uma das saídas é encontrar o menor conjunto dominante disponível no grafo e para isso como o problema não pode ser solucionado com exatidão em tempo polinomial, utilizamos algoritmos de aproximação para que assim possamos ter o máximo de eficiência da rede. Dessa forma dentre os algoritmos analisados decidimos expor dois: o primeiro sendo resolvendo o problema de aproximação para conjunto dominante e o segundo levando em consideração aspectos mais práticas da rede como velocidade de transmissão e disponibilidade dos nós.

5.1 Conjunto Dominante Conectado

Um exemplo destes algoritmos é o proposto por Mallikarjun Avula , Seong-Moo Yoo and Seungjin Park em (12), que constrói um Conjunto Dominante Conectado (CDC) com tamanho mínimo na maioria das vezes. Um CDC é basicamente um Conjunto Dominante normal, a única diferença é que um elemento do CDC consegue alcançar qualquer outro nó do conjunto por um caminho que permanece dentro do CDC. O algoritmo tem 3 etapas e elas são: Broadcasting, Ordenação de nó e Formação do Conjunto Dominante Conectado.

Na primeira etapa, cada nó da rede não tem nenhuma informação sobre seu vizinho. O algoritmo começa com a transmissão de um pacote que contém o número de vizinhos (inicialmente zero para cada nó) e o ID do nó (endereço MAC). Após um certo período de tempo, que é decidido com base no intervalo de transmissão de cada nó, todos os nós terão informações sobre seus vizinhos e seus correspondentes IDs de MAC.

Após essa etapa todos os elementos da rede sabem a quantidade de vizinhos que possuem e são ordenados no conjunto de vértices em ordem decrescente deste valor.

A ultima etapa é dividida em dois passos, o primeiro colore o grafo e a segundo finaliza a criação do CDC.

Passo 1:

1. Todos os nós são coloridos com a cor branca (Não pertence ao CDC e nem é vizinho de um elemento do CDC).
2. O primeiro nó em V (Nó com maior número de vizinhos) muda sua cor para a preta, envia uma notificação para seus vizinhos e estes trocam a cor branca pela cinza.
3. Verifica o próximo nó em V.
   (a) Se a cor for branca, o processo é repetido.
   (b) Se a cor for cinza, verifique se tem vizinhos brancos. Se sim, a cor do nó é alterada para preto e seus vizinhos brancos ficam cinza.
4. O processo acima continua até não existir nós brancos.
5. Se qualquer nó cinza for encontrado com pelo menos dois vizinhos negros (possível candidato a pertencer ao CDC) sua cor deve ser modificada para amarelo.

Passo 2:

1. Verifique se existem nós amarelos.
   (a) Se todos os vizinhos de um nó amarelo não forem vizinhos de algum nó preto, a cor do nó é alterada para preto. Se não, a cor é alterada para cinza.
2. Se um nó preto tiver pelo menos dois vizinhos pretos

(a) Se todos os vizinhos de um nó preto não forem vizinhos de algum outro nó preto, a cor do nó é alterada para cinza.

3. O Conjunto Dominante conectado é formado por todos os nós pretos.

Devido a constante movimentação dos nós na rede ad hoc é necessário que o algoritmo dê suporte a essa mobilidade nos três seguintes casos.

1. Quando um nó não-CDC deixa a rede:

   (a) A segunda etapa do algoritmo é refeita e se verifica a existência de algum nó preto que seja redundante (todos os seus vizinhos são vizinhos de outros nós pretos).

2. Quando um nó CDC deixa a rede:

   (a) Se a retirada do nó forma duas redes distintas e desconectadas, cada rede forma seu próprio CDC.

   (b) Se não, um nó cinza de sua vizinhança é selecionado, de modo que ele não tenha nó negro vizinho. A partir desse nó, um outro nó cinza vizinho deste será marcado como preto.

3. Quando um novo nó se une à rede:

   (a) Se o nó é vizinho de algum nó negro, mudará sua cor para cinza e não será necessário mudar o CDC.

   (b) Se o nó não é vizinho de um nó negro, ele precisa encontrar um nó vizinho que tenha nó negro e mudar sua cor para preto.

O tamanho do CDS gerado pelo algoritmo proposto é significativamente menor do que o gerado por vários algoritmos existentes, mas não é o mínimo em todos os casos. Entretanto este só se baseia em um parâmetro (grau dos nós) para decidir quem será o cluster-head, uma prática que não é a ideal.

5.2 Weighted Clustering Algorithm

O Weighted Clustering Algorithm (WCA) sugerido por Mainak Cahtterjee, Sajal K. Das and Damla Turgut (13) propõem o uso de 4 parâmetros, são eles: Grau, potência de transmissão, mobilidade, energia da bateria. Dependendo da aplicação são usados pesos diferentes nos parâmetros.

O processo de eleição de cluster-heads é dividido em 8 etapas:

1. Encontre os vizinhos de cada nó e defina seu grau d_v.

2. Calcule a diferença de graus, $G_v = |d_v - Q|$, sendo Q um número natural definido antes do calculo, para cada nodo v. Sua função é calcular a eficiência que esse nó vai ter como cluster-head, quanto menor, melhor. Cada cluster só poderá ter Q nós para garantir o um funcionamento eficiente.

3. Para cada nó, calcule a soma das distâncias, D_v, com todos os seus vizinhos. Determina o consumo de energia, quanto menor essa distância, menos energia é gasta e a comunicação entre os nós nesse cluster é melhor.

4. Calcule a média corrente da velocidade para cada nó até o tempo atual T. Isto dá uma medida de mobilidade e é denotado por M_v, como
   $M_v = 1/T * \sum_{t=11}^{T} \sqrt{(X_t - X_t - 1) + (Y_t - Y_t - 1)}$
   onde (X_t, Y_t) e ($X_t - 1$, $Y_t - 1$) são as coordenadas do nó v no tempo t e (t - 1), respectivamente. Um nó com menos mobilidade é sempre uma escolha melhor para um cluster-head.

5. Calcular o tempo acumulado, P_v, durante o qual um nodo v age como um cluster-head. Determina quanta energia é gasta pelo nó durante esse periodo.

6. Calcular o peso combinado W_v para cada nó v, onde $W_v = w_1 * G_v + w_2 * D_v + w_3 * M_v + w_4 * P_v$, onde w_1, w_2, w_3 e w_4 são os fatores de pesagem para os parâmetros.

7. Escolha esse nó com o menor W_v como o cluster-head. Todos os vizinhos do cluster-head escolhido não são mais autorizados a participar no processo eleitoral.

8. Repita os passos 2 a 7 para os nós restantes ainda não selecionados como cluster-head ou atribuídos a um cluster.

Devido as mudanças de posicionamento dos nós pode ser que algum nó fique sem conexão com algum cluster-head, isso significa que o algoritmo de clustering deve ser chamado novamente.

Para que dois clusters se comuniquem entre si, assumimos que as cluster-head's são capazes de operar no modo de alimentação dupla. Baixa potencia na comunicação dentro do cluster e alta potencia (tem mais alcance) para se comunicar com os cluster-head's vizinhos. Após a conexão dos clusters, a rede está preparada para o roteamento de mensagens, de modo que se um nó A quiser se comunicar com um nó B ele envia uma mensagem de solicitação de "descoberta de rota"contendo o id de B ao seu cluster-head. Se B não está presente no mesmo cluster que A, então o cluster-head de A propaga a mensagem de solicitação para seu cluster-head vizinho. Ao receber a solicitação, as cabeças de cluster podem verificar se B pertence a sua lista de membros. Essa consulta é feita em paralelo. Se B for encontrado, então uma confirmação positiva é enviada de B que atinge A através das cabeças de cluster e o procedimento de descoberta de rota é terminado. Se B não for encontrado, então a mensagem de solicitação é propagada para os vizinhos de dois saltos de cluster-head de A e o processo continua até B ser encontrado.

É importante ressaltar que os clusterheads formam um conjunto dominante para o grafo assim temos que ao achar os clusterheads levando em consideração os parâmetros dados estamos achando o melhor conjunto dominante para o grafo em questão.

6 Algoritmo proposto

6.1 Pseudocódigo

Algorithm 1 WCA modificado

function ALGORITMO WCA($M_{nxn}, \phi, f, v1, v2, v3, v4$)

for $i = 0$ to n **do**

for $j = 0$ to n **do**

if $M[i][j] > 0$ and $M[i][j] < f$ **then**

$g \leftarrow g + 1$ ▷ g = Grau do vértice

$d \leftarrow d + M[i][j]$ ▷ d = Somatório das distancias das arestas

end if

end for

$dif \leftarrow |grau - \phi|$ ▷ ϕ = Número ideal de vizinhos

Gera um valor para m ▷ m = Mobilidade do nó

Gera um valor para p ▷ p = Potência do nó

$W \leftarrow d * v1 + dif * v2 + m * v3 + p * v4$ ▷ W = Média ponderada

$V[i] \leftarrow v(W)$ ▷ V = Vetor de vertices

end for

$V \leftarrow sort(V)$

while $V \neq \emptyset$ **do**

$v \leftarrow V[0]$

Adiciona as arestas que envolvem o v em MF_{nxn}

Remove o v e seus vertices vizinhos do vetor V

end while

return MF_{nxn} ▷ Nova matriz de adjacencia

end function

6.2 Corretude

Suponha que o conjunto gerado não é um Conjunto Dominante. Então existe um vértice que não está no conjunto dominante e não tem nenhum vértice vizinho neste conjunto. Assim o algoritmo teria desconsiderado algum vértice da lista de vértices, o

que é um absurdo, pois o processo de escolha dos cluster-heads só termina quando essa lista estiver vazia.

6.3 Complexidade

O algoritmo tem dois laços encadeados, devido a manipulação com a matriz de adjacência, que são responsáveis pelo cálculo dos parâmetros de cada vértice, isto implica em uma complexidade de $O(n^2)$. A ordenação do vetor de vértices custa O(n log n), uma vez que a função utiliza é o método quicksort. Já na seleção dos cluster-heads é preciso ter $O(n^2)$ no pior caso, pois os vértices serão removidos um de cada vez (n vezes) do vetor e em cada uma dessas vezes seria verificado se existe algum vizinho (n vezes), este seria um grafo onde nenhum vértice tem conexão com outro e assim todos seriam denominados cluster-heads. Assim, temos: $O(n^2)$ + O(n log n) + $O(n^2)$. Portanto a complexidade final do algoritmo é $O(n^2)$.

7 Testes

Para que púdessemos realizar os testes do algoritmo realizamos a implementação de três algoritmos um que gera grafos, o segundo pega os vértices e arestas a partir de um arquivo de texto e por fim o terceiro faz com que de acordo com uma faixa de transmissão saibamos qual vértice é vizinho de qual. Dessa forma o algoritmo de geração aleatória irá ter como saída um grafo com a quantidade de vértices e arestas desejadas. Enquanto o segundo é necessário que no arquivo esteja descrito as arestas entre os vértices. Dessa forma podemos gerar um grafo com N vértices e A arestas.

7.1 Algoritmo I

Para o último algoritmo é necessário que o usuário informe apenas a quantidade de vértices que deseja ter no grafo e então a partir do tamanho da faixa de transmissão identificamos qual vértice está adjacente a outro vértice e assim instaceamos a matriz de adjacência. Após isso calculamos o conjunto dominante do grafo. É interessante notar que de acordo com o aumento da faixa de transmissão é possível ter um grafo com mais arestas ou menos arestas. Assim além de testarmos o algoritmo com instâncias de grafos com alguns milhares de vértices, fizemos testes com diferentes faixas de transmissão para observar a mudança no conjunto dominante de acordo com esse parâmetro. Em todos os testes também foi obtido sucesso ao encontrar o conjunto dominante. Assim observe as imagens para que possamos notar os diferentes conjuntos domintes gerados.

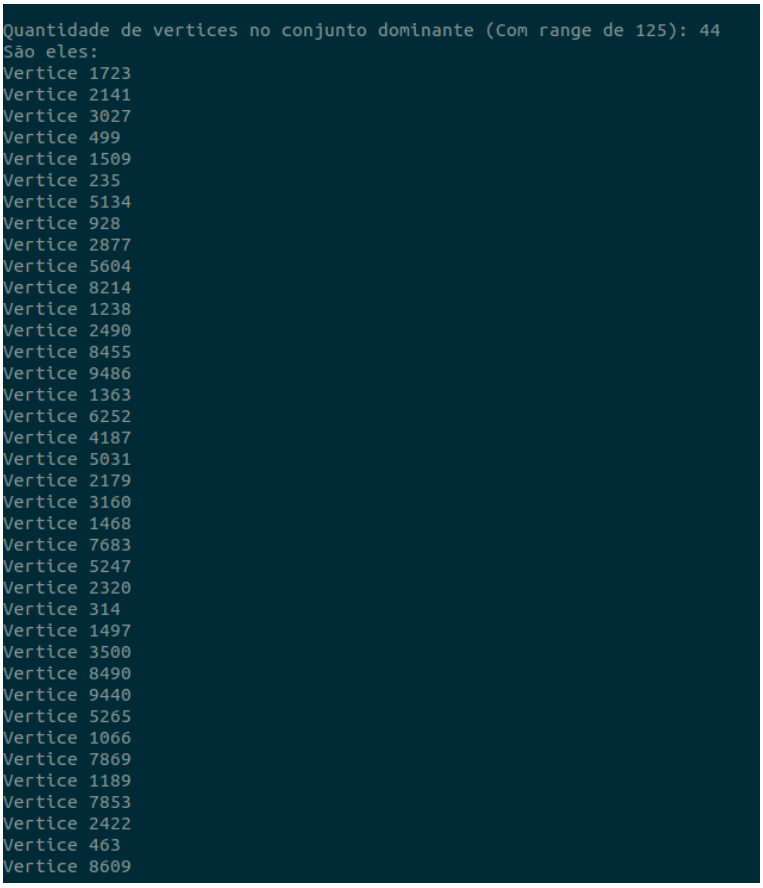

Figura 5: Conjunto Dominante gerado com range 125

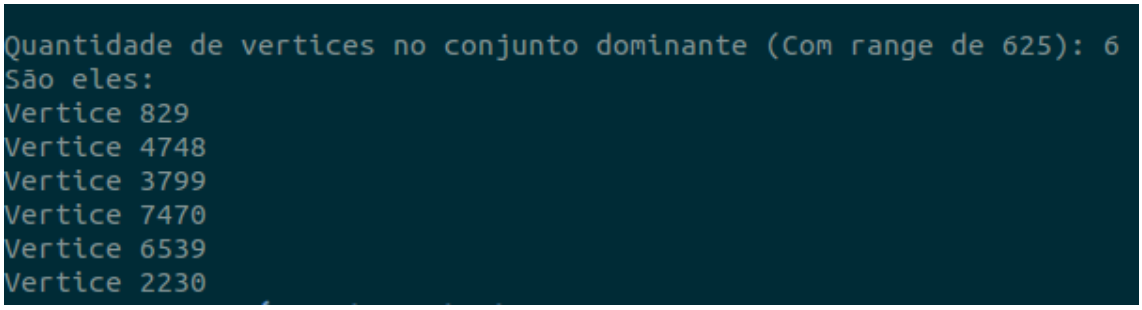

Figura 6: Conjunto Dominante gerado com range 625

7.2 Algoritmo II

Para o primeiro algoritmo é necessário que o usuário entre com a quantidade de vértices que deseja ter no grafo e a quantidade de arestas que cada vértice necessita ter, assim geramos um grafo e o colocamos para rodar no algoritmo proposto. Para esse algoritmo foram feito testes com até 200 vértices observando que ele sempre imprimia a saída desejada que são os vértices do conjunto dominante.

7.3 Algoritmo III

Para o segundo algoritmo, temos que é necessário que o usuário forneça um arquivo onde estarão presentes a quantidade de vértices que existe no grafo e as próximas linhas no arquivo serão as arestas, tendo os vértices que cada aresta incide. Para esse algoritmo geramos arquivos com poucas instâncias para que púdessemos observar a mudança no conjunto dominate a partir das alterações das arestas, além de utilizar arquivos que encontramos em (14) e modificamos os arquivos para que os testes pudessem ser feitos em grafos, com milhares de vértices e centenas de milhares, tendo um resultado positivo em todos os vértices. Porém para que possamos observar melhor os resultados, veja o caso que possuímos 10 vértices em instantes diferentes onde os vértices mudam de posição e mudam seus vizinhos, para que possamos ver com clareza a eficiência do algoritmo ao calcularmos o conjunto dominante aproximado do mínimo.

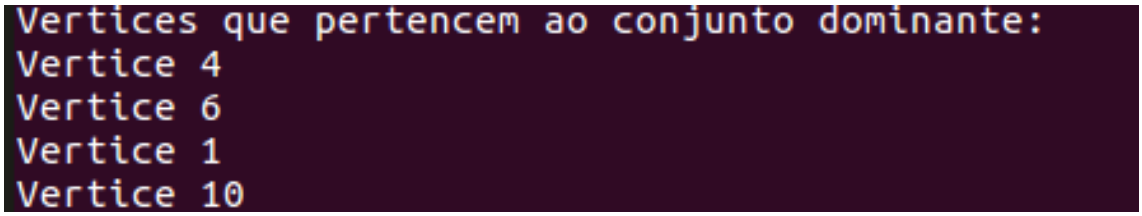

Figura 7: Conjunto Dominante no primeiro momento

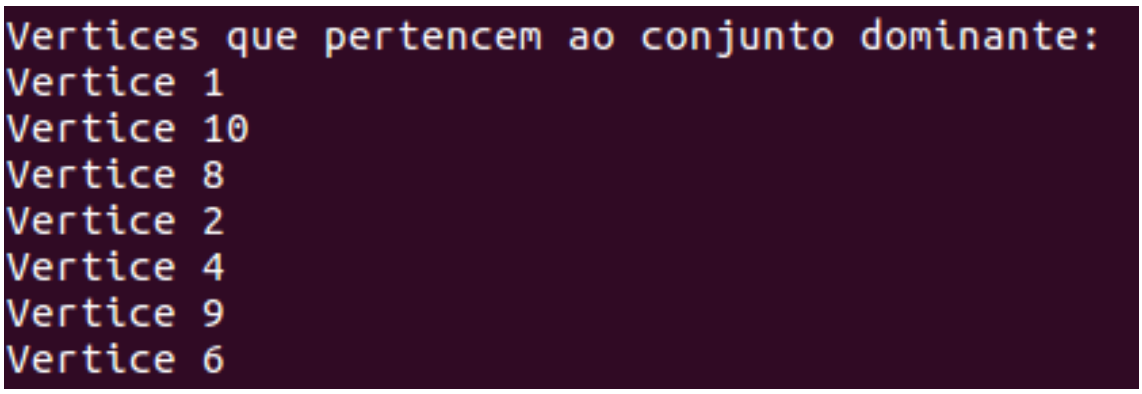

Figura 8: Conjunto Dominante no segundo momento

Assim observe que ao mudarmos as arestas do grafo indicando que os vértices estão se locomovendo, o conjunto dominante irá mudar e será atualizado de acordo com as novas arestas e novas posições. Podemos observar o grafo que foi usado para teste e tem como saída as descritas acima nos arquivos de texto que estão disponíveis na pasta do projeto.

Dessa forma observamos que a criação do conjunto dominante demente do tamanho da faixa de transmissão de cada vértice, pois quando a faixa de transmissão é maior menos vértices serão utilizados para o conjunto dominante.

```
Vertices que pertencem ao conjunto dominante:
Vertice 9
Vertice 1
Vertice 7
Vertice 10
Vertice 8
Vertice 2
Vertice 4
Vertice 6
```

Figura 9: Conjunto Dominante no terceiro momento

8 Resultado

Ao realizar os testes observamos que o algoritmo proposto pelo trabalho realizado tente a diminuir a quantidade de vértices no conjunto dominante de acordo com o aumento da faixa de transmissão e isso se fez presente em todos os testes realizados, como podemos notar na figura 10.

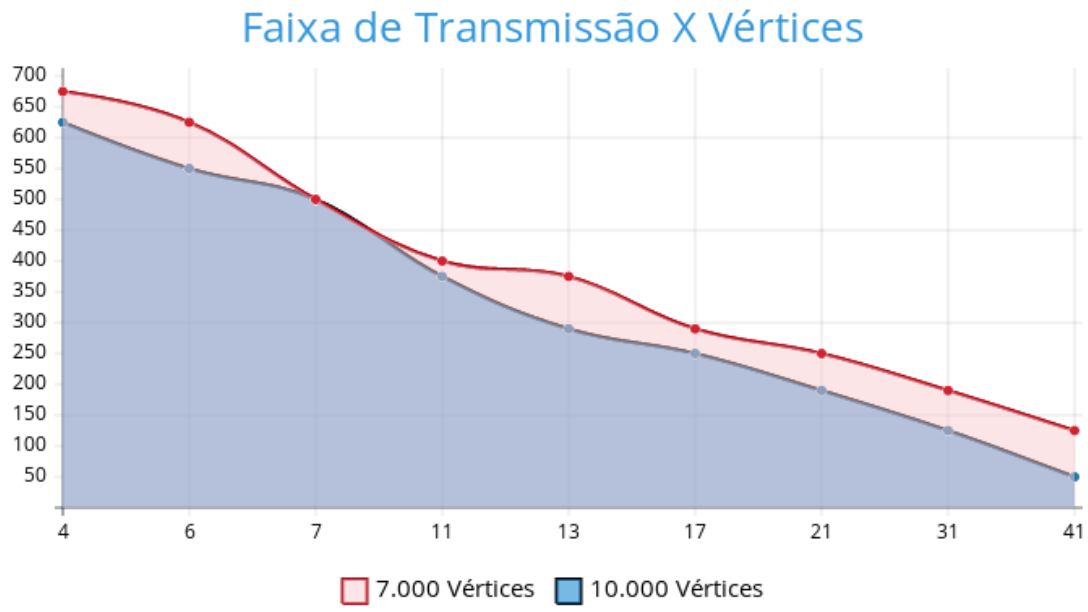

Figura 10: Gráfico da faixa de transmissão pelo número de vértices

Ainda observamos uma estabilidade no algoritmo, de forma que mesmo que aumentemos a quantidade de vértices no grafo a quantidade de vértices no conjunto dominante irá permanecer estável de forma que teremos poucas alterações nessa quantidade. Assim a heurística de aproximação para o conjunto dominante mínimo é considerada boa, por não se encaixar na categoria de heurístcas arbitrariamente ruins, como observamos logo abaixo.

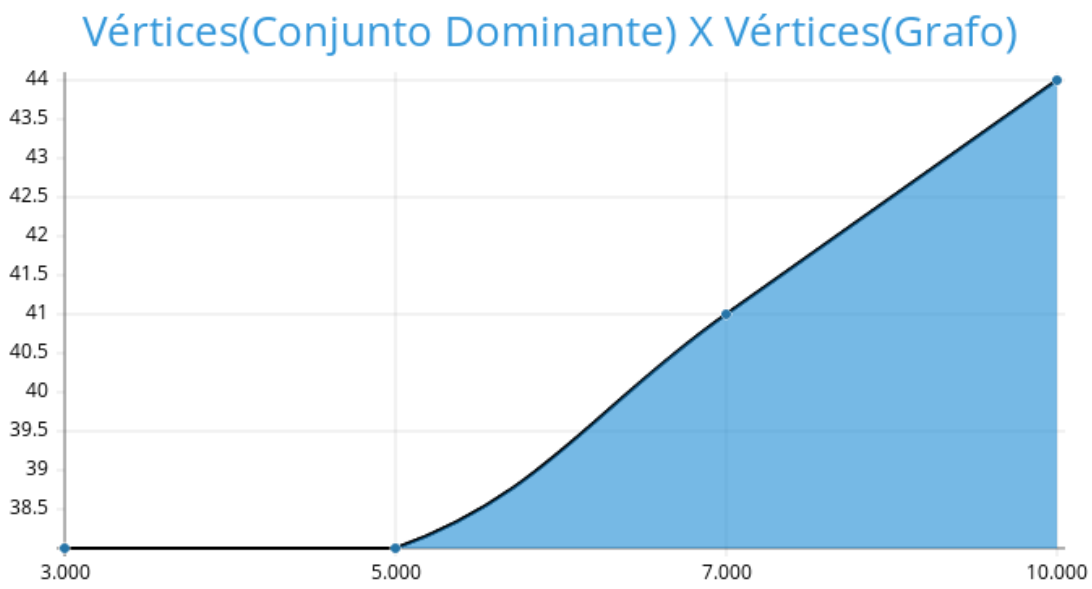

Figura 11: Gráfico da quantidade de vértices do conjunto dominante em relação a quantidade de vértices no grafo

Por fim, concluimos que ao comparar o algoritmo desenvolvido com o algoritmo no qual nos baseamos para ter a proposta(12), teremos um resultado semelhante, porém o algoritmo que propomos possui uma complexidade maior do que o anterior e a quantidade de vértices também é maior. Acreditamos que isso ocorre devido o acréscimo de mais um peso no algoritmo, que é o processamento do computador, o que para o grupo é um fator importante a ser considerado mas que o algoritmo no qual nos baseamos não considera.

9 Conclusão

Assim, utilizando a modelagem em grafos e o conceito de conjuntos dominantes, é possível tornar uma transmissão em locais de catástrofe e com recursos baixos, onde em geral a comunicação é dificultada em uma rede com transmissão mais eficiente, já que em locais assim a comunicação se faz tão precisa e necessária, mesmo com dificuldades para encontrar o conjunto dominante em alguns casos, essa tática funciona para esse tipo de situação devido a pequena quantidade de vértices presentes no grafo da rede, o que auxilia a encontrar com maiss facilidade o conjunto dominante, tornando a rede mais rápida.

Referências

[1] "As redes ad hoc em redes e segurança." https://www.maistecnologia.com/forum/redes-e-seguranca/16/as-redes-ad-hoc/16/0;wap2. Accessed: 2017-02-27.

[2] R. T. d. Oliveira *et al.*, "Sobre conjuntos dominantes eficientes em grafos," 2009.

[3] "Cluster: conceito e caractersticas." https://www.infowester.com/cluster.php. Accessed: 2017-02-27.

[4] Y. Chen, A. Liestman, and J. Liu, "Clustering algorithms for ad hoc wireless networks," *Ad Hoc and Sensor Networks*, vol. 28, p. 76, 2004.

[5] B. Balasundaram and S. Butenko, "Graph domination, coloring and cliques in telecommunications," in *Handbook of Optimization in Telecommunications*, pp. 865–890, Springer, 2006.

[6] S. Khuller, "Design and analysis of algorithms: Course notes," p. 60, 1996.

[7] J. M. Van Rooij and H. L. Bodlaender, "Exact algorithms for dominating set," *Discrete Applied Mathematics*, vol. 159, no. 17, pp. 2147–2164, 2011.

[8] S. Gaspers, D. Kratsch, M. Liedloff, and I. Todinca, "Exponential time algorithms for the minimum dominating set problem on some graph classes," *ACM Transactions on Algorithms (TALG)*, vol. 6, no. 1, p. 9, 2009.

[9] G. J. Chang, "Algorithmic aspects of domination in graphs," *Handbook of Combinatorial Optimization*, pp. 221–282, 2013.

[10] D. S. Johnson, "Approximation algorithms for combinatorial problems," *Journal of computer and system sciences*, vol. 9, no. 3, pp. 256–278, 1974.

[11] C. Shen and T. Li, "Multi-document summarization via the minimum dominating set," in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 984–992, Association for Computational Linguistics, 2010.

[12] M. Avula, S.-M. Yoo, and S. Park, "Constructing minimum connected dominating set in mobile ad hoc networks," *International Journal on Applications of Graph Theory in Wireless Ad Hoc Networks and Sensor Networks*, vol. 4, no. 2/3, p. 15, 2012.

[13] M. Chatterjee, S. K. Das, and D. Turgut, "Wca: A weighted clustering algorithm for mobile ad hoc networks," *Cluster computing*, vol. 5, no. 2, pp. 193–204, 2002.

[14] "Instâncias de testes para grafos." http://www.diag.uniroma1.it/challenge9/download.shtml. Accessed: 2017-06-3.