Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
DCC059 - Teoria dos Grafos Semestre 2013-3

Alocação de usuários em redes *clustered*-OFDM

Luã Silveira

Thiago Rizuti

Professor: Stênio Sã Rosário F. Soares

Relatório do trabalho da disciplina, parte integrante da avaliação da disciplina.

Juiz de Fora

Fevereiro de 2014

1 Introdução

Redes baseadas em modelos *clustered*-OFDM tem amplo uso em diversos meios de comunicação de dados, desde tecnologias com fio, como DSL (*Digital Subscriber Line*) e HFC (*Hybrid Fiber Coax*), à tecnologias sem fio, como WiMAX e 3G.

O problema tratado por esse trabalho é a alocação de usuários em sistemas *clustered*-OFDM, em específico, um sistema PLC (*Power Line Communications*), visando maximizar a média da Relação Sinal-Ruído (SNR) do Concentrador PLC. O Concentrador PLC é dividido em clusters, que são porções iguais da frequência de canal.

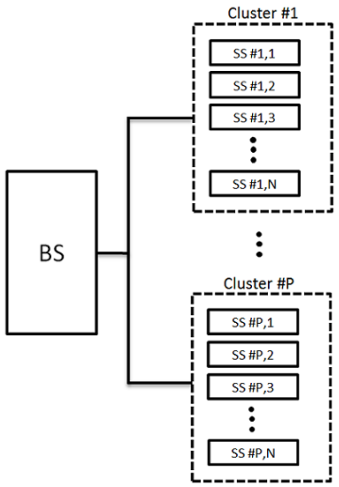

Figura 1: Arquitetura do concentrador PLC

Dado um sistema com n usuários e um número fíxo de clusters c (nesse caso, são cinco), cada usuário se comunica com todos clusters, estabelecendo assim um valor de sinal-ruído correspondente a cada um. Esse valor de SNR, no fim, é relativamente proporcional à taxa de transmissão desse usuário. O objetivo é maximizar a função objetivo dada pela soma da média da SNR's alocados em cada cluster.

Esse problema pode ser resolvido como um emparelhamento imperfeito máximo de um grafo bipartido G=(V,E) com as bipartições A, B $\subset$ V, |A| = n, |B| = c. O emparelhamento máximo é um conjunto P $\subset$ E que maximize a função objetivo, com a restrição de que todo elemento de B esteja ligado a apenas um elemento de A.

Algoritmos capazes de entragar a solução ótima para o problema, como o Algorítmo de *Kuhn*, não serão utilizados nesse trabalho, visto a complexidade de implementação. Usaremos, para isso, duas propostas de heurística para obter um resultado sub-ótimo.

2 Metodologia utilizada

Utilizaremos duas heurísticas para a alocação de usuários em clusters, sendo que cada uma delas terá uma variação gulosa, gulosa randomizada e randomizada reativa.

Como estrutura de dados para a representação do grafo, foi utilizada uma matriz de adjacência, pela facilidade em proceder processos como ordenação de linhas/colunas e busca diretamente pelo índice.

2.1 Heurística 1

A primeira abordagem heurística é dividida nas seguintes etapas, que podem ser vistas no pseudocódigo na seção A:

- Escolha do cluster: os clusters são ordenados pela média do valor de todos os vértices incidentes nele. É escolhido um cluster c_i, aquele com maior média;
- Escolha de usuário: usuários são ordenados pelo valor de seus vértices. O usuário n_i que tem a maior aresta incidente em c_i é escolhido. Nas versões randomizadas, serão sorteados os melhores usuários de acordo com a variável α;
- Todas as arestas de n_i que incidem em outros cluters são apagadas, garantindo que uma aresta de usuário incida apenas em um cluster;
- O procedimento é executado até que todos usuários estejam alocados em um cluster.

Essa heurística tenta alocar o maior número possíveis de usuários em seus melhores clusters, podendo aumentar a vasão em algum cluster específico, dependendo das instâncias envolvidas.

2.2 Heurística 2

A segunda heurística pode ser dividida nas seguintes etapas, que podem ser vistas no pseudocódigo na seção A:

- Escolha do usuário: usuários não alocados são ordenados pela diferença entre suas melhores e piores arestas. O usuário n_i com a maior diferença é escolhido para ser alocado em um buffer;
- Escolha do cluster: o usuário n_i é alocado em um cluster c_i em que tem o maior valor de aresta. Na abordagem randomizada o cluster é sorteado de acordo com a variável α;

- Todas as arestas de n_i que incidem em outros cluters são apagadas, garantindo que uma aresta de usuário incida apenas em um cluster;
- O procedimento é executado até que todos usuários estejam alocados em um cluster.

Essa heurística tem como objetivo recompensar a alocação de usuários muito ruins em certos cluster mas bons em outros. Ao alocar usuários com grandes diferenças entre valores de SNR em seus melhores canais, evita-se que eles sejam alocados em clusters ruins, diminuindo a vazão do sistema.

3 Experimentos computacionais

A análise de resultados foi executada em cima da eficiência das heurísticas para encontrar bons resultados. Para isso, em cada execução, foram rodadas 500 iterações de cada algoritmo guloso randomizado e randomizado reativo. Cada execução foi realizada com instâncias diferentes, criadas de forma randomizada. As instâncias de um problema real não foram utilizadas pela dificuldade de obter dados reais de medição em redes PLC, mas instâncias aleatórias representam um pior caso, onde os canais de usuários são imprevisíveis, seguindo modelos aleatórios de ruído.

As tabelas de resultados podem ser vistas no arquivo de planilhas que acompanha o relatório.

4 Conclusões

Os resultados mostram que as heurísticas gulosas foram as que obteram melhor resultado sub-ótimo. Na comparação direta entre as duas, a Heurística 1 mostrou resultados levemente superiores para instâncias menores que 100 usuários, com a Heurística 2 sendo melhor daí por diante. É possível observar o crescimento do desvio padrão a cada aumento de α, o que é esperado, devido à escolha de alocação randomizada fugir dos padrões orientados pela heurística.

Na análise de resultados da Heurística 2, a abordagem de sortear o cluster a ser alocado pode ter influenciado negativamente nos resultados, já que foge completamente da ideia de solução.

A Anexo I

Algorithm 1 Heurística 1

Require: Grafo g, numCluster $clusters$

1: $Grafosaida$
2: $numUsers \leftarrow g.vertices - clusters$
3: $matriz[clusters][numUsers] \leftarrow g$
4: $somaFinal \leftarrow 0$
5: $somaClusters \leftarrow 0$
6: **while** $numUsers$ **do**
7: $somaFinal \leftarrow 0$
8: **for** $i < clusters$ **do**
9: **for** $j < numUsers$ **do**
10: $somaClusters \leftarrow matriz[clusters][numUsers] + somaClusters$
11: **if** $somaClusters \geq somaFinal$ **then**
12: $somaClusters \leftarrow somaFinal$
13: $melhorCluster \leftarrow i$
14: **end if**
15: **end for**
16: $somaClusters \leftarrow 0$
17: **end for**
18: $sort(matriz[melhorCluster][:], indices[])$
19: $saida[melhorCluster][indices[0] + clusters] \leftarrow matriz[melhorCluster][indices[0]]$
20: $matriz[:][indices[0]] \leftarrow 0$
21: **end while**
22: **return** $saida$

Algorithm 2 Heurística 2

Require: Grafo g, numCluster $clusters$

1: $Grafosaida$
2: $numUsers \leftarrow g.vertices - clusters$
3: $matriz[clusters][numUsers] \leftarrow g$
4: **while** $numUsers$ **do**
5: $maiorDif \leftarrow 0$
6: **for** $i < numUsers$ **do**
7: $maiorSNR \leftarrow 0$
8: $menorSNR \leftarrow \infty$
9: $flag = 0$
10: **for** $j < clusters$ **do**
11: **if** $matriz[j][i] \geq maiorSNR$ **then**
12: $maiorSNR \leftarrow matriz[j][i]$
13: $aux \leftarrow j$
14: **end if**
15: **if** $matriz[j][i] \leq menorSNR$ && $matriz[j][i] \neq 0$ **then**
16: $menorSNR \leftarrow matriz[j][i]$
17: $flag \leftarrow 1$
18: **end if**
19: **end for**
20: **if** $flag = 0$ **then**
21: $menorSNR \leftarrow 0$
22: **end if**
23: **if** $maiorSNR - menorSNR \geq maiorDif$ **then**
24: $maiorDif \leftarrow maiorSNR - menorSNR$
25: $indice \leftarrow i$
26: $indCluster \leftarrow aux$
27: **end if**
28: **end for**
29: $quickSort(matriz[:][indice], indices[])$
30: $saida[indCluster][cluster + indice] \leftarrow matriz[indCluster][indice]$
31: $matriz[:][indices[0]] \leftarrow 0$
32: **end while**
33: **return** $saida$