Preenchimento de Cache de Video na Borda da Rede Baseado em Coloração de Grafos

Fábio Engel de Camargo[1,2], Elias P. Duarte Jr[2]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
R. Cristo Rei, 19 Toledo 85902-490 PR

[2]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19018 Curitiba 81531-990 PR

fabioe@utfpr.edu.br, elias@inf.ufpr.br

Abstract. *Video in streaming services are often based on clouds. As the number of mobile devices accessing this type of content increases, in order to ensure the required quality of service, it is important to use caches at the edge of the network. As caches bring content closer to the end user, they reduce traffic in the cloud and improve the end user experience. This work addresses one of the aspects inherent to the use of caches, the content placement considering that end users watch videos on mobile devices. Each base station has an associated cache. A strategy content placement is presented based on graph coloring, in which the content of the caches is modified after each request that does not match. The proposal was evaluated through simulation and compared with alternatives, presenting the highest hit-rate. We conclude that the number and the distribution of caches and end users has an impact on the efficiency of the different strategies.*

Resumo. *Serviços de* streaming *de vídeo são frequentemente baseados em nuvens computacionais. Tendo em vista o aumento dos dispositivos móveis acessando este tipo de conteúdo, para garantir a qualidade de serviço requerida, é importante fazer uso de caches na borda da rede. Na medida em que as caches aproximam o conteúdo do usuário final, provocam uma redução do tráfego na nuvem e uma melhoria da experiência do usuário final. Este trabalho aborda um dos aspectos inerentes à utilização de caches, o preenchimento de conteúdo tendo em vista usuários finais em dispositivos móveis. Cada estação base tem uma cache associada. Uma estratégia de preenchimento de cache baseada em coloração de grafos é apresentada, na qual o conteúdo das caches é modificado sempre que há uma solicitação não atendida. A proposta foi avaliada através de simulação e comparada com alternativas e sempre apresentou a melhor taxa de acertos. Concluímos que o número e a distribuição de caches e dos usuários finais tem impacto na eficiência das diferentes estratégias.*

1. Introdução

As vantagens e benefícios da computação em nuvem causaram, nos últimos vários anos, um forte movimento de migração de serviços de *streaming* de vídeo, que são baseados nestas plataformas [Koksal 2019, Li et al. 2018]. Entretanto, apesar de

todos os benefícios, é preciso notar que os serviços baseados em nuvem possuem, em menor ou maior nível, dependência da infraestrutura de rede para acesso aos serviços e dados disponibilizados remotamente. O aumento do uso da nuvem pode inclusive criar um gargalo sobre a infraestrutura da rede que fornece acesso à nuvem. A existência destes gargalos causam aumento do atraso e impactam negativamente a experiência do usuário ao utilizar um serviço remoto [Khakimov et al. 2018].

Para reduzir a quantidade de tráfego na rede é preciso compreender sua composição. Dois pontos importantes neste contexto são o aumento global do uso de dispositivos móveis e a demanda por vídeos na rede. O último Cisco Visual Networking Index (VNI) Complete Forecast [Cisco 2019], relatório que apresenta projeções globais para o tráfego de dados, evidencia estes dois pontos. De acordo com este relatório, o tráfego de dispositivos móveis apresentou aumento de 71% em 2017 sobre o ano anterior. Neste mesmo ano, 59% do tráfego de dispositivos móveis correspondia a transmissões de vídeos, estima-se que para 2022 este número aumente para 79%.

Uma vez que a largura de banda disponível é um recurso limitado e que depende de investimentos para sua melhoria, observa-se neste momento o surgimento de tecnologias que fazem o caminho inverso ao discutido anteriormente, isto é, trazem os serviços da nuvem para mais próximo do usuário final, normalmente a uma distância de um salto. Esta alternativa é conhecida como *edge computing* [Carnevale et al. 2018], [Giang et al. 2018], [Ren et al. 2018]. Como um exemplo proeminente da *edge computing* têm-se a *fog computing*, conceito introduzido pela Cisco [Khakimov et al. 2018], o qual implica, resumidamente, na implantação de servidores locais que consistem em versões mais simples dos servidores que estão na nuvem, porém, com capacidade de armazenamento e poder de processamento adequados aos serviços utilizados.

No contexto das redes sem-fio uma possível solução para minimizar o impacto do tráfego gerado pelos vídeos, e também por outras categorias de conteúdo, ocorre através da utilização de *caches*. *Caches* são dispositivos que mantém cópias dos arquivos localmente, ou próximo ao usuário final, reduzindo a quantidade de acessos redundantes a servidores de conteúdo, consequentemente, reduzindo o tráfego gerado, além da latência e proporcionando uma boa experiência ao usuário (QoE - *Quality of Experience*). A utilização de *caches* para este fim possui uma série de aspectos que impactam sua eficiência, como por exemplo, localização, capacidade e estratégias de preenchimento de conteúdo [Yang et al. 2018], [Liu and Lau 2017], [Ayoub et al. 2018].

Neste trabalho apresentamos uma extensão da estratégia de preenchimento de cache baseada na proposta em coloração de grafos apresentada em [Javedankherad et al. 2018]. A estratégia original baseia-se na aplicação do algoritmo de coloração sobre o grafo que representa as estações base do sistema, que tem *caches* associadas. Deste modo, para cada cor atribuída, destina-se um conteúdo diferente a ser armazenado. Entretanto, neste trabalho mostramos que esta estratégia não é eficiente para um número pequeno de *caches*, além de depender do posicionamento dos usuários do sistema. Propomos então uma variação na qual o conteúdo das caches é modificado sempre que há uma solicitação não atendida. Comparamos

a estratégia proposta com a estratégia original e outra (*baseline*) em que todas as caches mantém o mesmo conjunto de arquivos mais populares. Concluímos que o número de estações base bem como o posicionamento dos usuários tem impacto na eficiências das diferentes estratégias.

O restante deste trabalho está organizado da seguinte maneira. A Seção 2 o modelo de sistema adotado. Na Seção 3 são apresentadas as estratégias de preenchimento de cache. Na Seção 4 a simulação implementada é descrita, bem como resultados obtidos. Por fim, a conclusão segue na Seção 5.

2. Modelo de Sistema

Este trabalho apresenta uma estratégia para posicionamento de vídeos em *cache* na borda da rede. O trabalho considera redes sem-fio de alta densidade, especificamente, com uma macro célula que cobre toda a rede. Ao seu centro, encontra-se uma estação base macro (MBS - *Macro Base Station*), responsável pela cobertura da rede. Dentro do alcance da MBS, encontram-se estações bases (BS - *Base Stations*) dotadas de uma interface de comunicação de menor raio de alcance. Todas as BSs possuem capacidade limitada para armazenamento (*cache*) de arquivos de vídeos.

As BS são distribuídas em um plano de duas dimensões em torno da MBS, seguindo um processo de Poisson homogêneo [Chen and Kountouris 2016]. Os usuários do sistema são distribuídos utilizando outro processo de Poisson homogêneo independente. Em particular, o trabalho considera que os usuários não se movimentam. A Figura 1 mostra um exemplo em que é possível visualizar a MBS ao centro, com raio de alcance de 350 metros. As BS, com raio de alcance de 80 metros cada, estão distribuídas ao redor da MBS. Note que todos os usuários estão dentro da área de cobertura da MBS, entretanto, podem estar ou não, na área de alcance de uma ou mais BS.

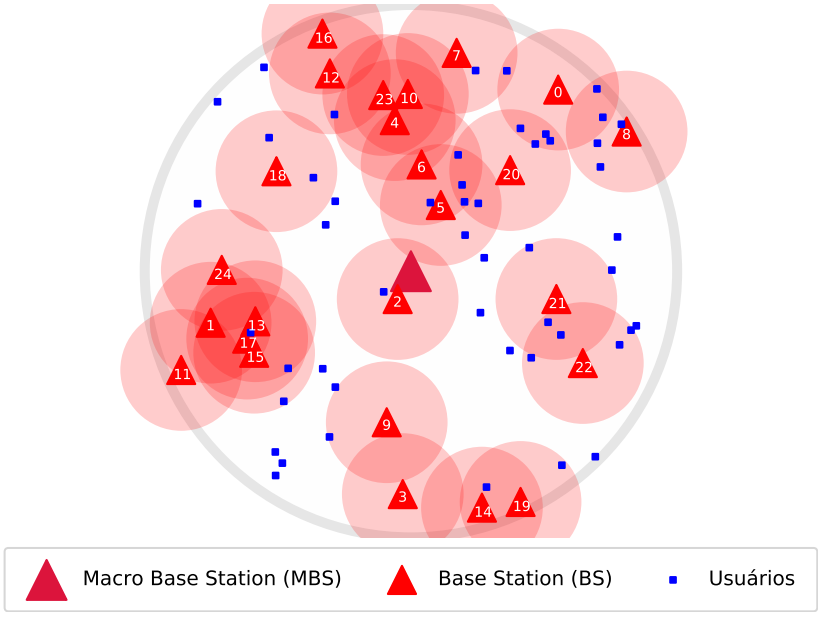


Figura 1. Exemplo de rede considerada.

Considera-se que os usuários deste sistema estejam em busca apenas de ar-

quivos de vídeos. Além disso, considera-se que existem no sistema 1000 arquivos diferentes de vídeos. Este valor corresponde ao universo de opções oferecidas aos usuários do sistema e é representado pelo conjunto $\Gamma = \{f_1, \ldots, f_{1000}\}$.

Para cada elemento de Γ são associados dois atributos, tempo de duração e popularidade. O tempo de duração de cada arquivo é atribuído aleatoriamente seguindo uma distribuição normal. A popularidade é indicada por um número inteiro obtido a partir de uma distribuição do tipo Zipf-like, baseada na lei de Zipf. A lei de Zipf possibilita obter uma estimativa sobre o número de acessos a um objeto, baseado em sua popularidade. Esta lei foi proposta pelo linguista americano George Kingsley Zipf [Zipf 1949]. Originalmente, descrevia a frequência da utilização de diferentes palavras do vocabulário inglês. Observou-se que o artigo "the" ocorre em aproximadamente um décimo de um texto típico; a próxima palavra mais popular "of" ocorre aproximadamente um vigésimo.

Neste tipo de distribuição, a frequência diminui acentuadamente à medida que o número utilizado para classificação aumenta (quanto menor o número mais popular é o objeto), deste modo, um pequeno número de objetos aparecerá com maior frequência, enquanto que a ocorrência de um grande número de objetos será menor. Este tipo de distribuição mostrou-se apto a representar outros fenômenos estatísticos que seguem uma distribuição exponencial, tal como a frequência com que páginas Web são requisitadas. A lei de Zipf estabelece que a probabilidade relativa ao i-ésimo elemento mais popular corresponde a $1/i$. Autores como [Breslau et al. 1999] defendem uma distribuição alternativa, chamada Zipf-like (ou Zipf-equivalente), na qual a requisição do i-ésimo elemento mais popular corresponde a $1/i^{\alpha}$, com $0 < \alpha < 1$.

3. Coloração de Grafos

No cenário descrito na Seção 2, um mesmo usuário pode estar ao alcance de múltiplas estações bases. Na situação do usuário estar sob o alcance de uma única BS, a melhor política de preenchimento de conteúdo é atribuir os vídeos com maior popularidade a esta estação base. Ao possibilitar acesso a múltiplas BSs, convém diversificar os arquivos presentes na *cache* destas estações base que compartilham uma mesma área de alcance. A coloração de grafos é utilizada neste trabalho como uma ferramenta que possui como propósito atribuir qual conjunto de arquivos deverá ser armazenado em uma determinada BS.

Em teoria dos grafos, a coloração de grafos corresponde a um caso de rotulagem, no qual é atribuído um rótulo, tradicionalmente uma cor, a um elemento (vértice ou aresta) sujeito a algum tipo de restrição. Ao aplicar a coloração de vértices, busca-se colorir os vértices de um grafo de tal modo que não haja dois vértices adjacentes que compartilhem a mesma cor [Szwarcfiter 2018].

No modelo de sistema utilizado, as BSs são representadas por meio do grafo $G_{BS}(V, E)$. V é o conjunto não vazio de vértices que simbolizam as estações base, tal que $V = \{bs_0, \ldots, bs_{n-1}\}$, com n indicando o número total de BSs do sistema. Os elementos do conjunto E, denotados pelo par $\{v, w\}$, são as arestas do grafo. Somente existem arestas entre os vértices quando as distâncias entre as BSs são menores que um limiar pré-definido, nesta simulação é utilizado o mesmo valor de alcance de cada BS, isto é, se a distância entre bs_i e bs_j for menor que o limiar,

$(bs_i, bs_j) \in E$. A Figura 2 mostra o grafo G_{BS} gerado a partir do cenário ilustrado pela Figura 1.

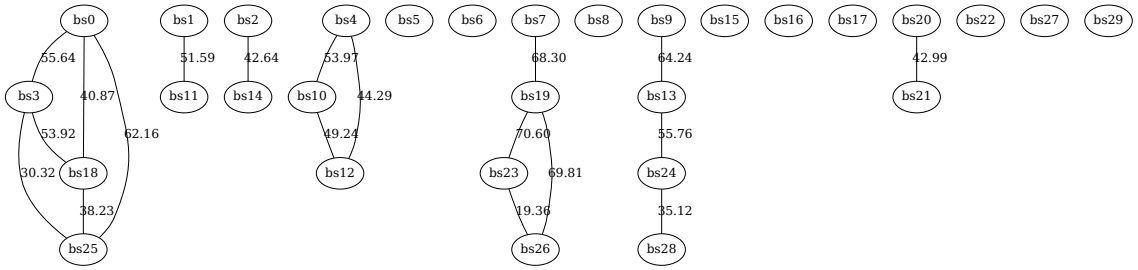


Figura 2. Exemplo de grafo gerado a partir do cenário utilizado na Figura 1.

Algoritmos de coloração de grafos regularmente buscam "colorir" o grafo utilizando o menor número de cores possíveis. A menor quantidade de cores necessárias para este feito é denominado "número cromático". Atualmente é desconhecida a existência de um bom algoritmo para determinação do número cromático, isto é, nega-se conhecer um algoritmo de coloração que seja executado em tempo polinomial. De fato, o problema de coloração é considerado um problema NP-completo, exceto para alguns tipos específicos de grafos bipartidos e árvores os quais não se aplicam no problema considerado neste trabalho [Ouerfelli and Bouziri 2011].

A alta complexidade computacional necessária para a execução de um algoritmo que resolva o problema de coloração conduz ao uso de métodos heurísticos que possuem boa aproximação e são executados em tempo polinomial. Assim como em [Ouerfelli and Bouziri 2011] e [Szwarcfiter 2018], este trabalho utiliza como alternativa algoritmos gulosos. Algoritmos gulosos são comumente utilizados em problema de otimização no qual atuam determinando a cada etapa a solução ótima local, na expectativa de encontrar o ótimo global.

Baseado em [Szwarcfiter 2018], utiliza-se neste trabalho o termo *Greedy Coloring* para representar o método de atribuição de cores descrito pelo Algoritmo 1. Este algoritmo recebe dois parâmetros de entrada, o grafo G e seu conjunto de vértices ordenados $K = \{v_1, \ldots, v_n\}$. A forma com que estes vértices são ordenados será discutida a seguir. A execução deste método atribuirá uma cor para cada vértice, tipicamente é utilizado um número inteiro para representar esta cor. O algoritmo *Greedy Coloring* nem sempre irá colorir o grafo utilizando o número cromático, entretanto, realiza tal tarefa com um número pequeno de cores.

Algoritmo 1: *GreedyColoring*

```
   Entrada: G, K
1  início
2  |  para cada v ∈ K faça
3  |  |  v ← menor cor possível, tendo em vista as cores dos vértices vizinhos;
4  |  fim
5  fim
6  retorna Grafo Colorido
```

Os vértices contidos em K podem ser ordenados de modo a produzir uma saída adequada ao problema. No trabalho original [Javedankherad et al. 2018],

opta-se por ordenar os vértices de acordo com o seu grau, do maior ao menor grau. O grau do vértice indica o número de vértices incidentes para com o próprio vértice. Para o grafo G_{BS}, vértices com maior grau indicam estações base que encontram-se em regiões sobrepostas em relação a outras estações base, portanto, supostamente regiões mais populosas. Utiliza-se para este fim o método denominado LF, acrônimo de *largest-first*, mostrado pelo Algoritmo 2 [Szwarcfiter 2018].

Algoritmo 2: *LF - Largest First*

```
   Entrada: G
1  início
2  |  K ← vértice de G ordenados em ordem decrescente em relação ao grau;
3  |  GreedyColoring(G, K);
4  fim
5  retorna Grafo Colorido
```

LF baseia-se na observação de que vértices com baixo grau possuem menores restrições em relação as cores utilizadas para sua coloração, sendo natural iniciar a coloração pelos vértices que possuem maiores restrições, isto é, vértices de maior grau. O método LS possui complexidade $O(|E|+|V|)$, onde $|E|$ representa o número arestas de G e $|V|$ representa o número de vértices de G. Ao aplicar o Algoritmo 2, cada vértice (estação base) adquire uma cor. Para exemplificar, será tomado como exemplo o grafo G_{BS} apresentado pela Figura 2. Os vértices são coloridos utilizando o conjunto de cores apresentados pela Figura 3.

Figura 3. Legenda de cores.

Assim, obtém-se o grafo G_{BS} com seus vértices coloridos, exibido pela Figura 4.

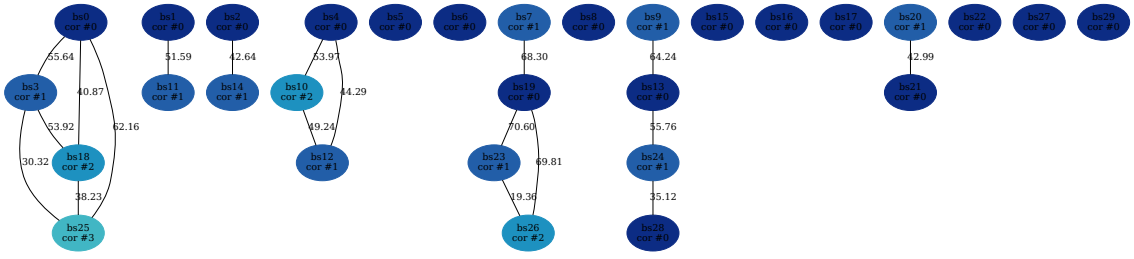


Figura 4. Cenário da Figura 2 tendo aplicado algoritmo de coloração.

Como é possível observar, o algoritmo de coloração permite a distinção das estações de base tendo como critério sua cor. Tal distinção possibilita a diversificação de conteúdo armazenado nas respectivos *caches* e aumenta a possibilidade do conteúdo requisitado pelo usuário estar em uma das estações base ao qual possui acesso.

Por fim, apresenta-se o algoritmo de preenchimento da *cache* das BSs, baseado na coloração de grafos mostrado pelo Algoritmo 3.

Algoritmo 3: PLACEMENT BY COLORING

```
   Entrada: G_BS, Γ
1  início
2  |  LF(G_BS);
3  |  δ ← arquivos de Γ ordenados de acordo com sua popularidade.;
4  |  para cada v ∈ V(G_BS) faça
5  |  |  para cada i ∈ {0, 1, 2, ..., γ} faça
6  |  |  |  σ ← cor atribuída para v;
7  |  |  |  v ← δ[(σ * γ) + i]
8  |  |  fim
9  |  fim
10 fim
```

Este algoritmo, nomeado *Placement by Coloring*, recebe como entrada o grafo que representa as BSs e o conjunto de arquivos do sistema. Inicialmente (linha 2) aplica-se o método LS a fim de colorir os vértices de G_{BS}. Em seguida, os arquivos de vídeo são ordenados de acordo com o número que indica sua popularidade, é utilizada a variável δ para representação do vetor de arquivos ordenados (linha 3). Para cada BS é atribuída um conjunto de arquivos com o total igual a γ, neste trabalho utiliza-se $\gamma = 50$, ou seja, atribui-se apenas 0.5% do total de arquivos do sistema para cada BS. De modo a ilustrar o método de preenchimento da *cache* utilizando coloração de grafos, foi criada a Figura 5.

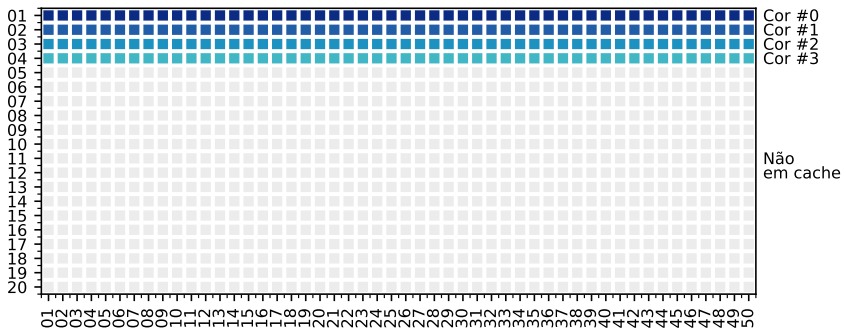


Figura 5. Representação de δ

A Figura 5 exibe o total de arquivos do sistema dispostos em uma matriz de acordo com sua popularidade, os que encontram-se ao topo possuem maior popularidade. As linhas de 4 a 9 do Algoritmo 3 garantem que BSs coloridas com a cor #0, armazenem os arquivos contidos na primeira linha da matriz. As coloridas com a "cor #1" recebam os próximos 50 mais populares, isto é, arquivos situados entre as posições 51 e 100 (linha 2) em termos de popularidade. O mesmo ocorre com as cores subsequentes.

4. Implementação e Resultados Experimentais

Para avaliar o Algoritmo 3 foi implementada uma simulação utilizando a linguagem de programação Python. A simulação possui como objetivo demonstrar o impacto da diversificação de conteúdo utilizando o algoritmo de coloração. Para este fim, foi considerado o cenário apresentado na seção 2, cujos parâmetros estão na Tabela 1.

Parâmetro	Valor
Raio de alcance da MBS	350m
Raio de alcance da BS	80m
Número de usuário do sistema	50
Número de opções de vídeo	1000
Número de vídeo armazenados em *cache* (γ)	50
Tempo de duração de cada vídeo	entre 1 e 10 minutos
Tempo de duração da simulação	1, 3, 6 e 12 horas
Número de cenários simulados para cada número de BSs	3000
Variável α da distribuição Zipf-Like	0.6

Tabela 1. Parâmetros utilizados na simulação.

A avaliação utilizou como métrica a taxa de acerto, ou *hit rate*. A taxa de acerto corresponde às solicitações de arquivos de vídeos por parte dos usuários que são prontamente atendidas por estarem armazenados na *cache* de alguma BS que está ao seu alcance. Para efeito comparativo, foram avaliados outros dois métodos de preenchimento de *cache* além do *Placement by Coloring*, são eles: *Trivial* e *Dynamic Coloring*.

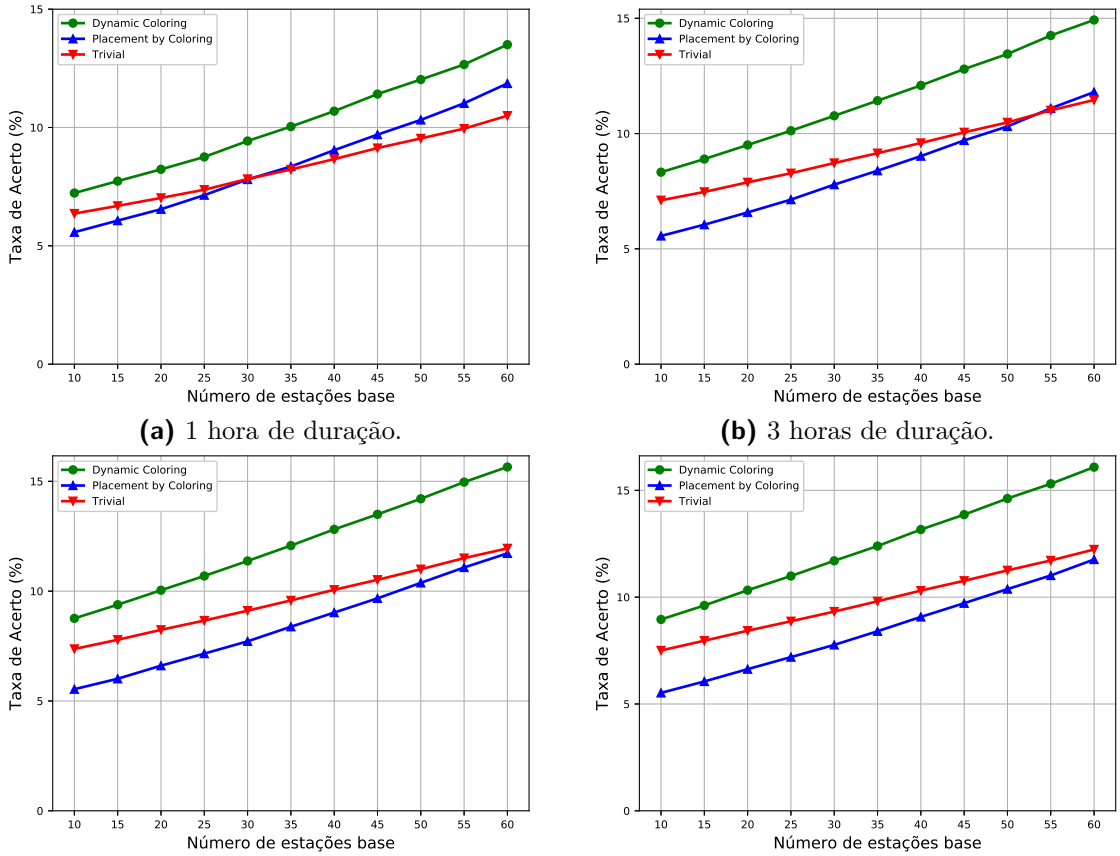
O método *Trivial* possui uma abordagem simples e de baixo custo computacional. Consiste apenas em atribuir o mesmo conjunto de arquivos considerados mais populares para todas as BSs. Este método torna homogêneas as *caches*, desconsiderando as possíveis vantagens em diversificar o conteúdo.

O método *Dynamic Coloring* preenche as *caches* assim como o *Placement by Coloring*, entretanto, o conteúdo da *cache* é modificado sempre que a requisição de um arquivo não é atendida, isto é, não se encontra em cache. Como exemplo, considere a situação ao qual um usuário está sob o alcance de duas BSs, bs_0 e bs_1, coloridas, respectivamente, com as "cores" cor #0 e cor #1. Ao solicitar um arquivo que não encontra-se na *caches* de bs_0 e bs_1, ocorre uma troca de arquivos na *cache* da estação base de "maior cor". Para isto, bs_1 remove o arquivo menos popular de seu *cache* e insere em seu lugar o arquivo recém requerido, para que futuras requisições sobre este arquivo sejam prontamente atendidas.

Os três métodos foram submetidos a simulações que consideraram o mesmo cenário em cada execução. Foram executados 3000 cenários distintos. Os cenários diferenciam-se por apresentarem diferentes posicionamentos dos usuários e BSs, e também diferentes arquivos de vídeos. No instante inicial de cada execução, todos usuários sob o alcance de uma BS solicitam um arquivo de vídeo, seguindo uma distribuição zipf-like. Ao terminar de assistir seu vídeo, imediatamente ocorre uma nova solicitação por parte do usuário, este comportamento segue até o final de cada execução.

Ao final de cada execução são contabilizados os acertos. A taxa de acerto de acordo com o número de BSs é apresentado pela Figura 6. Esta Figura divide-se em quatro outras figuras que apresentam o comportamento dos três métodos ao longo do tempo.

Diferente de [Javedankherad et al. 2018], no qual o *Placement by Coloring* é apresentado como superior ao método *Trivial*, observa-se que isto apenas ocorre



(a) 1 hora de duração.

(b) 3 horas de duração.

(c) 6 horas de duração.

(d) 12 horas de duração.

Figura 6. Taxa de acertos.

quando considera-se um sistema com muitas BSs e pequeno tempo de duração. Ao considerar um tempo maior de execução, o método *Trivial* apresenta melhor desempenho sobre o *Placement by Coloring*, isto ocorre devido ao seu princípio que consiste em conferir "menores cores" para os vértices de maior grau, ou seja, possui como prioridade atribuir conteúdo mais popular para a BS que encontra-se em uma região populosa em relação a outras BSs, não considerando a existência de usuários próximos. BSs que possuem "maior cores", e consequentemente conteúdos menos populares, podem fornecer acesso à um número maior de usuários do que BSs coloridas com "menores cores".

O método proposto por este trabalho, *Dynamic Coloring*, apresenta maior taxa de acerto independente do tempo de simulação e quantidade de estações base utilizadas. Ele combina a diversificação de conteúdo obtida por meio do *Placement by Coloring* em seu estado inicial, e considera as requisições dos usuários para tornar a cache dinâmica, isto é, ativa e com maior chances de acerto.

5. Conclusão

Com o avanço das nuvens computacionais, um número cada vez maior de organizações tem migrado seus serviços de *streaming* de vídeo para ambientes em nuvem. Concomitantemente, o número de usuários finais acessando este tipo de conteúdo vem aumentando de forma extraordinária. O uso de *caches* na borda da rede para

aproximar o conteúdo dos usuários é uma estratégia concreta para permitir que a qualidade de serviço seja atendida, reduzindo o tráfego na nuvem e melhorando a experiência do usuário final. Neste trabalho apresentamos uma estratégia de preenchimento de *caches* para usuários de dispositivos móveis, atendidos por estações base com *caches* associadas. A estratégia proposta é baseada em coloração de grafos. O conteúdo das *caches* é definido dinamicamente, sendo alterado à medida em que usuários fazem requisições. A proposta foi implementada utilizando simulação. Resultados comprovam a forte influência do número e posicionamento de *caches* e usuários no desempenho. Mostramos que a coloração pode ser utilizada de modo a obter um bom índice de acerto. A estratégia proposta apresentou a maior taxa de acerto independente do tempo de simulação e quantidade de estações base utilizadas. A estratégia combina a diversificação de conteúdo obtida por meio da coloração, mas considerando as requisições dos usuários para tornar a *cache* dinâmica, isto é, ativa e com maior chances de acerto.

Trabalhos futuros incluem considerar mobilidade, definindo em avaliando estratégias que levem em conta usuários móveis. Outro caminho relevante é a caracterização do usuário, identificar seu perfil e preferências para adotar uma política de preenchimento de *cache* mais efetiva.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. O projeto foi parcialmente financiado com o *grant* 311451/2016-0 do CNPq.

Referências

Ayoub, O., Musumeci, F., Addeo, C., Mussini, M., and Tornatore, M. (2018). Caching placement strategies for dynamic content delivery in metro area networks. In *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pages 1–6.

Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1999). Web caching and zipf-like distributions: evidence and implications. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, volume 1, pages 126–134 vol.1.

Carnevale, L., Celesti, A., Galletta, A., Dustdar, S., and Villari, M. (2018). From the cloud to edge and iot: a smart orchestration architecture for enabling osmotic computing. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 419–424.

Chen, Z. and Kountouris, M. (2016). D2d caching vs. small cell caching: Where to cache content in a wireless network? In *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–6.

Cisco (2019). Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022 white paper.

Giang, N. K., Lea, R., Blackstock, M., and Leung, V. C. M. (2018). Fog at the edge: Experiences building an edge computing platform. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 9–16.

Javedankherad, M., Zeinalpour-Yazdi, Z., and Ashtiani, F. (2018). Cache placement phase based on graph coloring. In *2018 9th International Symposium on Telecommunications (IST)*, pages 187–191.

Khakimov, A., Muthanna, A., and Muthanna, M. S. A. (2018). Study of fog computing structure. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 51–54.

Khakimov, A., Muthanna, A., and Muthanna, M. S. A. (2018). Study of fog computing structure. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 51–54.

Koksal, I. (2019). Video streaming platforms and the benefits of cloud. *Forbes Magazine*.

Li, X., Salehi, M. A., Bayoumi, M., Tzeng, N., and Buyya, R. (2018). Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services. *IEEE Trans. Parallel Distrib. Syst.*, 29(3):556–571.

Liu, A. and Lau, V. K. N. (2017). How much cache is needed to achieve linear capacity scaling in backhaul-limited dense wireless networks? *IEEE/ACM Transactions on Networking*, 25(1):179–188.

Ouerfelli, L. and Bouziri, H. (2011). Greedy algorithms for dynamic graph coloring. In *2011 International Conference on Communications, Computing and Control Applications (CCCA)*, pages 1–5.

Ren, P., Qiao, X., Chen, J., and Dustdar, S. (2018). Mobile edge computing – a booster for the practical provisioning approach of web-based augmented reality. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 349–350.

Szwarcfiter, J. L. (2018). *Teoria Computacional de Grafos: Os algoritmos*. Elsevier Editora, 1 edition.

Yang, P., Zhang, N., Zhang, S., Yu, L., Zhang, J., and Shen, X. S. (2018). Content popularity prediction towards location-aware mobile edge caching. *IEEE Transactions on Multimedia*, pages 1–1.

Zipf, G. K. (1949). *Human Behavior And The Principle Of Least Effort: An Introduction to Human Ecology*. Addison-Wesley Press.