Relatório final

Projeto e Análise de Algoritmos

Alexander Decker de Sousa

2 de Julho de 2018

1 Introdução

O paradigma de Redes Definidas por *Software*, ou *SDN* (do inglês, *Software Defined Networks*), surgiu nos últimos anos e tem se mostrado bastante promissor. Dentre suas várias vantagens, destaca-se, por exemplo, seu potencial para a resolução da chamada *ossificação da Internet*, termo que se refere ao fato de novas tecnologias e protocolos de rede serem dificilmente inseríveis no mundo real, devido ao perigo de interrupção da rede e consequente dano a atividades às quais a *Internet* já se tornou ferramenta essencial [2].

O princípio fundamental que define as *SDN* consiste na definição da operação de elementos comutadores da rede, como roteadores e *switches*, através de comandos enviados em tempo de execução. As variações de *SDN* são diversas, porém todas mantém uma estrutura básica em camadas. A camada mais inferior consiste no plano de dados e é responsável pelo encaminhamento de pacotes propriamente dito. Por demandar alto desempenho, esta camada é mantida simples e genérica, sendo baseada na leitura de tabelas programáveis [7] ou na execução de microcódigos alteráveis em tempo de execução [5]. A camada intermediária consiste no plano de controle, constituído de dispositivos ou *softwares* controladores. Os controladores atuam efetivamente na reprogramação dos elementos de rede, oferecendo recursos para facilitar o controle da rede por parte das aplicações, como por exemplo uma visão logicamente centralizada da rede [2]. Finalmente, na terceira camada rodam as aplicações de rede, que atuam através dos controladores.

Todavia, a elaboração inteligente da topologia da rede, considerando as interligações entre os diversos elementos de *hardware* e o posicionamento dos mesmos, pode ser complexa em demasia. O posicionamento geográfico de controladores em redes *WAN* definidas por *software* é NP-difícil e se trata de uma aplicação do problema de localização de facilidades (*Facility Location Problem*). Os problemas que tratam do posicionamento ou da escolha dos enlaces que ligam os controladores aos switches ou mesmo entre si são chamados de Problemas de Localização de Controladores (*Controler Placement Problems*) [3].

2 Trabalhos Relacionados

A literatura relacionada aos Problemas de Localização de Controladores é normalmente agrupada de acordo com o objetivo da otimização. Um dos objetivos mais frequentes da literatura é a minimização da latência, que inclui a minimização do tempo de propagação das mensagens, do tempo em que as mensagens ficam armazenadas em filas e do tempo de processamento dos controladores. Esta abordagem também envolve a escolha dos enlaces de forma a fazer a distribuição balanceada de carga dentre os controladores. A maioria dos trabalhos aplica algoritmos populares de resolução do Problema das K-Medianas Mínimas [9], visto que esta versão do problema se assemelha bastante com o Problema de Localização de Facilidades.

Outro objetivo comum é o aumento da confiabilidade e da resiliência. A confiabilidade, no caso, diz respeito ao inverso da probabilidade de falha de um enlace e a resiliência é a capacidade da manutenção do funcionamento da rede mesmo na presença de nós ou enlaces errantes. Assim como a minimização da latência, este problema se assemelha bastante à localização de facilidades e, muitas vezes, os trabalhos na literatura também fazem uso de algoritmos aplicáveis ao Problema das K-Medianas Mínimas. Em [4], por exemplo, é utilizada a meta-heurística de Recozimento Simulado (*Simulated Annealing*) para uma solução não exata do problema.

Outros trabalhos objetivam a redução dos gastos de energia e de implantação da rede a partir da estimativa da quantidade de controladores e de enlaces que devem ser estabelecidos entre os mesmos e entre controladores e comutadores [8].

Há ainda uma abordagem pouco explorada [9], que consiste na otimização multi-objetivo. [10] é um dos poucos trabalhos a contemplar esta abordagem, buscando otimizar a confiabilidade da rede, balancear a carga entre os controladores e minimizar o pior caso da latência.

3 Definição do Problema

O problema aqui tratado segue uma abordagem diferente dos demais presentes na literatura. Ao invés de buscar a otimização de aspectos técnicos da rede, como latência, vazão ou balanceamento de carga, o objetivo é minimizar os custos financeiros de manutenção da rede, visto que esta é uma das principais preocupações sob uma visão empresarial. Sendo assim, o foco principal é a minimização dos custos com energia, visto que esta é uma das principais despesas em empreendimentos relacionados com processamento massivo de dados. Não obstante, restrições relacionadas à latência dentro da rede e divisão de carga são também incluídas, de forma a obter um sistema financeiramente compensativo e que garanta qualidade de serviço.

Dessa forma, o problema de otimização pode ser definido pela determinação de quais localizações dentre as pré-definidas devem ser escolhidas para possuir um controlador e quais enlaces dentre os pré-definidos devem ser escolhidos para serem enlaces de controle, de forma a minimizar o custo financeiro dos gastos de energia ao mesmo tempo em que se limita a soma de todas as latências nas conexões de controle e que se respeita a quantidade máxima de requisições que cada controlador posicionado pode receber. Podemos ainda definir o problema de decisão inerente ao de otimização, que consiste em determinar se há algum subconjunto de localidades e de enlaces de controle que sirva para a instalação dos controladores de forma que a demanda de cada *switch* seja satisfeita, a capacidade de todos os controladores seja respeitada, a soma das latências dos enlaces de controle utilizados não ultrapasse uma constante pré-definida e os custos totais com energia também sejam limitados superiormente por uma constante.

4 Modelagem do Problema

Seja a rede definida por $G = (V, A)$, em que os vértices V representem as localizações de comutadores e A os enlaces da rede dedicados a funções de controle. Os enlaces A são representados como uma matriz binária de adjacência. A energia demandada para enviar uma mensagem pelo enlace ij aparece em E_{ij} e o RTT esperado para um par de requisição e resposta entre os elementos ij aparece em T_{ij}. O vetor K_i contém o custo da energia na localidade i, o vetor ${}^sF_i$ contém a frequência média de requisições do comutador da localidade i, o vetor ${}^sE_i$ contém a quantidade de energia gasta por requisição no comutador da localidade i e o vetor ${}^sW_i$ contém a potência gasta pelo comutador da localidade i quando ocioso.

O número de controladores não é definido *a priori*, porém admite-se que todos sigam o mesmo modelo, suportando uma frequência de requisições de ${}^cF$, gastando uma quantidade de energia ${}^cE$ para processar cada mensagem que chega e gastando ${}^cW$ de potência quando ocioso. Deseja-se determinar quais comutadores terão controladores em suas localidades e quais enlaces de controle serão utilizados, de forma a minimizar o custo total, definido na equação

$$K_T = {}^cW \sum_{i=1}^{|V|} K_i P_i + \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} K_j C_{ij} {}^sF_i \cdot (E_{ji} + {}^cE) + \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} K_i C_{ij} {}^sF_i \cdot (E_{ij} + {}^sE_i) + \sum_{i=1}^{|V|} {}^sW_i K_i$$

P consiste no vetor binário de posicionamento dos controladores e indica se há um controlador na localidade i. C, por sua vez, consiste na matriz de conexões de controle. Cada posição de P se torna uma variável binária em um problema de programação inteira, assim como cada posição válida de C. Uma posição de C só é válida se a mesma posição estiver assinalada na matriz A. Podemos observar que, salvos casos degenerados, o vetor P é equivalente à diagonal principal de C, visto que um comutador será servido pelo controlador de sua própria localidade, se o mesmo existir.

Podemos notar que, como os comutadores são distribuídos de antemão, os únicos gastos influenciáveis pelas variáveis de decisão oriundos dos mesmos são os custos com a transmissão de requisições aos controladores. Sendo assim, os custos relacionados a comutadores ociosos e processamento de requisições representam constantes na função objetivo e, portanto, podem ser omitidos, como na equação

$$K_T = {}^cW \sum_{i=1}^{|V|} K_i P_i + \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} K_j C_{ij} {}^sF_i \cdot (E_{ji} + {}^cE) + \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} K_i C_{ij} {}^sF_i \cdot E_{ij}$$

O problema envolve um conjunto de restrições de ordem linear ou quadrática em relação a $|V|$, sumarizadas nos itens que se seguem.

- **Restrições de conectividade.** Cada comutador se conecta a exatamente um controlador. Essa restrição pode ser implementada fazendo $\sum_{j=1}^{|V|} C_{ij} = 1 \quad \forall i$.
- **Restrições contra sobrecarga.** Sendo ${}^cF$ a frequência máxima de requisições suportada por cada controlador, essa restrição pode ser implementada como $\sum_{i=1}^{|V|} C_{ij} {}^sF_i \leq {}^cF \quad \forall j$.
- **Restrição de latência.** Limita superiormente a soma dos RTTs de todas as conexões de controle utilizadas a um parâmetro $\hat{T}$. Pode ser implementada como $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} C_{ij} T_{ij} \leq \hat{T}$.
- **Restrições de acoplamento.** Responsável por impedir inconsistências entre P e C. Sendo assim, uma coluna em C pode apenas ter algum valor não nulo se a posição correspondente em P estiver assinalada. Esse conjunto de restrições pode ser implementado como $\sum_{i=1}^{|V|} C_{ij} - |V| P_j \leq 0 \quad \forall j$.

5 NP-Completude

O problema de decisão é facilmente provável como NP-Completo. Em primeiro lugar, o problema está em NP, visto que um certificado (P, C) pode ser verificado aplicando as inequações das restrições e em seguida verificando se $K_T \leq K_0$ para alguma constante pré-definida K_0, o que possui complexidade total limitada a $O(|V|^2)$.

Além disso, o problema é difícil, visto que o problema do *conjunto dominante de tamanho até* p é polinomialmente redutível ao mesmo, como demonstrado a seguir.

Lema 1. Seja $< G, p >$ uma instância qualquer do problema do conjunto dominante, onde G corresponde ao grafo e p corresponde ao número máximo de nós no conjunto dominante. Seja a instância do problema de localização de controladores $< G, {}^cW, K, E, {}^cE, {}^sF, {}^cF, T, \hat{T}, K_0 >$, onde ${}^cW_i = 1$, $K_i = 1 \forall i$, $E_{ij} = 0 \forall (i,j)$, ${}^cE = 0$, ${}^sF_i = 0 \forall i$, ${}^cF \geq 0$, $T_{ij} = 0 \forall (i,j)$, $\hat{T} \geq 0$ e $K_0 = p$. Existe um conjunto dominante de tamanho até p em G se e somente se existe uma configuração válida para o problema do localização de controladores dados os parâmetros especificados.

Prova. *Qualquer escolha de enlaces de controle é viável do ponto de vista da restrição de latência, assim como qualquer configuração de (C, P) é viável em relação às restrições contra sobrecarga de controladores, visto que todas as latências e demandas de comutadores são nulas. Sendo assim, sem entrar no mérito do custo de energia, o posicionamento dos controladores em qualquer conjunto dominante de G gera uma configuração viável da versão de otimização do problema do localização de controladores, visto que a única restrição não trivial passa a ser a de que cada comutador deve ser conectado a exatamente um controlador. Cada controlador instalado gera um custo constante unitário de energia, ou seja, o custo total de energia é numericamente igual ao número de controladores. Portanto, se existir um conjunto dominante de tamanho até p em G, haverá uma configuração viável para o problema do localização de controladores com até $p = K_0$ controladores, ou seja, com custo até K_0. Da mesma forma, se existir uma solução para o problema de localização de controladores com os parâmetros definidos no lema 1, as no máximo $K_0 = p$ localidades com controladores serão um conjunto dominante de G. Como as únicas operações necessárias para a elaboração da instância do problema de posicionamento de controladores a partir da instância do problema do conjunto dominante são definições de variáveis de até duas dimensões, a transformação pode ser dita polinomial.* ■

Como o problema está em NP e em NP-Difícil, concluímos que está em NP-Completo.

6 Algoritmo Exato

O processamento das instâncias de entrada para a elaboração dos parâmetros do problema de programação inteira foi implementado em *Python* e os problemas resultantes são resolvidos utilizando o algoritmo de *Branch and Bound* contido no pacote *GNU Linear Programming Kit* (*GLPK*) [1].

7 Heurísticas

Todas as três heurísticas aqui propostas são adaptações e extensões do algoritmo aproximativo de razão $ln(|V|)$ para o problema do conjunto dominante capacitado descrito em [6]. Tal algoritmo admite demandas inseparáveis dentre os nós e busca a minimização da soma dos pesos dos nós escolhidos para o conjunto dominante, sendo provado como $O(|V|^3)$ em tempo.

A primeira heurística aqui descrita basicamente aplica o algoritmo aproximativo fazendo pequenas alterações de forma a melhor compreender o problema tratado. A principal alteração consiste em um método a partir do qual a solução garantidamente obedece a restrição de latência. Tal método é baseado no princípio de que a remoção de todas as arestas de latência maior que $\hat{T}/(|V|+1)$ garante que qualquer solução seja viável no que diz respeito à limitação da latência. Isso se deve ao fato de qualquer solução ter no máximo $|V|+1$ arestas de controle, o que ocorre no caso em que há apenas um controlador servindo toda a rede. Nesse caso, a latência total quando as arestas de latência superior a $\hat{T}/(|V|+1)$ são eliminadas é limitada superiormente a $(|V|+1)\cdot\hat{T}/(|V|+1) = \hat{T}$. Soluções com um número de arestas $|C| \leq |V|+1$ terão, portanto, latência total $|C|\cdot\hat{T}/(|V|+1) \leq \hat{T}$.

A segunda heurística, por sua vez, altera um pouco mais o algoritmo de obtenção do conjunto dominante, enquanto a terceira heurística modifica a segunda em relação ao método para a obtenção de uma solução que respeite a restrição da latência. As três heurísticas aparecem descritas nas subseções que se seguem.

7.1 Heurística 1

Seja $N(u)$ a lista de adjacência ainda não dominada do nó u em ordem decrescente de demandas, ou seja, de frequências de requisição. A eficiência ϵ de u é dada pela equação 1. Aqui, $w(u)$ é dado pelo custo da localidade u permanecer com um controlador e servir o próprio comutador, isto é, $w(u) = K_u({}^cW + {}^sF_u{}^cE)$. A função $x(u,i)$, por sua vez, retorna 1 se a capacidade residual de processamento de um eventual controlador em u suportar as demandas dos i primeiros elementos de $N(u)$ e a soma de tais demandas for maior que zero, retornando 0 caso contrário.

$$\epsilon(u) = \max_{1\leq i\leq |N(u)|} \frac{i}{w(u)\cdot x(u,i)} \tag{1}$$

A heurística para encontrar uma solução para o problema do localização de controladores com relaxação na restrição de latência é sumarizada no algoritmo 1. A variável $\hat{i}$, no caso, corresponde ao valor de i para o qual a razão $i/(w(u)\cdot x(u,i))$ foi maximizada para o nó de maior eficiência $\hat{u}$.

O algoritmo 1 garante que as localidades retornadas sejam um conjunto dominante e que as capacidades dos controladores sejam respeitadas, restando, portanto, apenas a restrição da latência sem garantias de satisfação. O algoritmo 2, por sua vez, garante que a solução encontrada seja viável em todos os aspectos. Ciente de que a remoção das arestas *problemáticas*, ou seja, de latência maior que $\hat{T}/(|V|+1)$, garante a satisfação da restrição da latência, o algoritmo remove uma aresta problemática qualquer por vez até encontrar uma solução viável, o que acontecerá no pior caso após todas as arestas problemáticas serem removidas.

Esse algoritmo necessariamente retornará uma solução viável desde que ${}^sF_i \leq {}^cF \forall i$, visto que nesse caso um conjunto dominante $P = V$ é trivialmente viável para posicionamento dos controladores. De fato, só existirão soluções viáveis caso essa condição seja satisfeita, visto que, caso contrário, nenhum controlador poderá servir o comutador com demanda superior a ${}^cF$.

A complexidade do *Posicionador de Controladores Relaxado* é $O(|V|^3)$ assim como o algoritmo aproximativo utilizado de base. A verificação da condição da latência, por sua vez, pode ser feita em $O(|A|)$. A *Heurística 1* chamará o *Posicionador de Controladores Relaxado* $O(|A|)$ vezes, o que gera uma complexidade total de $O(|A|(|V|^3+|A|))$. Como $|A| = O(|V|^2)$, podemos ignorar o custo assintótico da verificação da condição de latência, o que faz a complexidade ser equivalente a $O(|A||V|^3)$.

Algoritmo 1: Posicionador de controladores relaxado

Entrada: $< G, {}^cW, K, {}^cE, {}^sF, {}^cF >$

início
  $P = \emptyset$
  $C = \emptyset$
  $\zeta = \{{}^cF$ **para cada** u∈V}
  enquanto existirem nós não dominados
    $\hat{u} = \max_{u \in V} \epsilon(u)$
    se $\hat{u}$ não possuir um controlador
      $P = P \cup \hat{u}$
    fim
    para cada u em $N(\hat{u})_{1..\hat{i}}$
      Seja e a aresta entre u e $\hat{u}$, faça $C = C \cup e$
      considere u dominado
      $\zeta_{\hat{u}} = \zeta_{\hat{u}} - {}^sF_u$
    fim
    se $\hat{u}$ ainda não estiver dominado
      $\zeta_{\hat{u}} = \zeta_{\hat{u}} - {}^sF_{\hat{u}}$
      considere $\hat{u}$ dominado
      Seja e a aresta de *auto-loop* de $\hat{u}$, faça $C = C \cup e$
    fim
  fim
  retorne C,P
fim

7.2 Heurística 2

A algoritmo da *Heurística 2* é essencialmente o mesmo do algoritmo da *Heurística 1*. Todavia, a função de eficiência passa considerar a soma das demandas a serem satisfeitas ao invés do número de nós a serem dominados, como mostra a equação 2. Além disso, o peso dos nós passa a ser o custo energético total a ser agregado tanto com o posicionamento do controlador na localidade quanto pelo estabelecimento dos enlaces de controle, como definido na equação 3.

$$\epsilon(u) = \max_{1 \leq i \leq |N(u)|} \frac{\sum_{j=1}^{i} {}^sF_{[N(u)_j]}}{w(u,i) \cdot x(u,i)} \tag{2}$$

$$w(u,i) = K_u {}^cW + \sum_{j=1}^{i} {}^sF_{[N(u)_j]} \left(K_u (E_{[u,N(u)_j]} + {}^cE) + K_{[N(u)_j]} E_{[N(u)_j,u]} \right) \tag{3}$$

Analogamente à *Heurística 1*, uma solução gerada pela *Heurística 2* possui garantia de viabilidade desde que ${}^sF_i \leq {}^cF \forall i$. A complexidade desse algoritmo também é a mesma da *Heurística 1*, ou seja, $O(|A||V|^3)$, visto que, mesmo tendo aspecto linear, o cálculo de $w(u,i)$ de forma constante a partir de $w(u, i-1)$.

7.3 Heurística 3

A *Heurística 3* se difere da *Heurística 2* pelo método utilizado para assegurar a viabilidade no que diz respeito à restrição da latência. Ao invés de escolher uma aresta problemática qualquer, como fazem as heurísticas 1 e 2, a *Heurística 3* escolhe sempre a aresta problemática de maior latência. Isso não agrega assintoticamente nenhum custo extra, o que faz com que esse algoritmo também rode em $O(|A||V|^3)$.

8 Experimentos

Para facilitar a inserção de parâmetros de novas redes, foi elaborado um *parser* de *XML* que transforma os parâmetros de controladores, comutadores, conexões de controle e outros mais nas

Algoritmo 2: HEURÍSTICA 1

Entrada: $< G, {}^cW, K, {}^cE, {}^sF, {}^cF, T, \hat{T} >$

início
    enquanto verdadeiro
        C,P = Posicionador de controladores relaxado
        $T' = 0$
        para cada aresta $e_{ij} \in C$
            $T' = T' + T_{i,j}$
        fim
        se $T' \leq \hat{T}$
            retorne C,P
        senão
            escolha aleatoriamente uma aresta problemática e
            $A = A - e$
        fim
    fim
fim

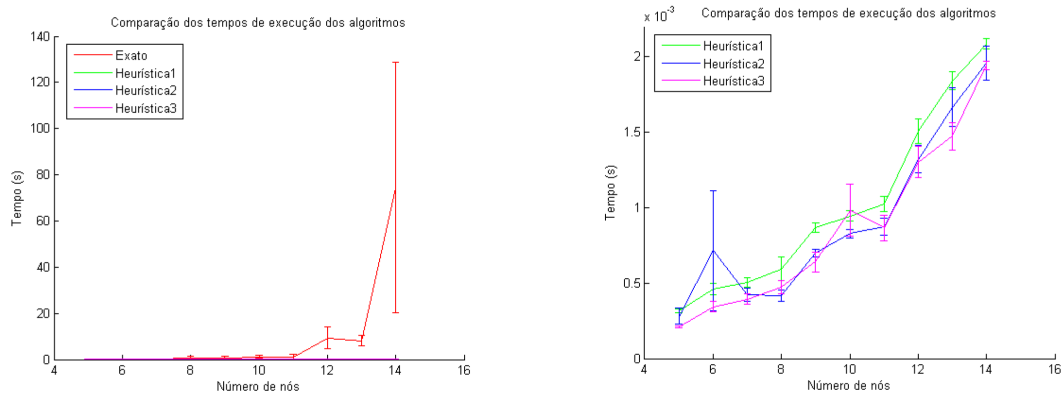
matrizes necessárias para a definição da instância do problema. Tal *parser* de *XML* também admite parâmetros coringa e, nesse caso, gera uma instância aleatória que garanta a existência de ao menos uma solução viável. Nesse caso, é necessário informar apenas os limites dentro dos quais cada parâmetro deverá se manter e os números de arestas e de vértices desejados. Tal recurso foi empregado para gerar instâncias aleatórias enquanto cada parâmetro de interesse era avaliado controladamente. A tabela 1 mostra os parâmetros utilizados nos experimentos. Os valores entre colchetes correspondem aos limites utilizados para as gerações de instâncias aleatórias.

Table 1: Parâmetros utilizados

Parâmetro	Intervalo de Valores	Unidade
Variação de longitude	10, 100	graus
Variação de latitude	10, 100	graus
Velocidade de Propagação	[150000,170000]	Km/s
Tempo de proc. por requisição (Controlador)	[0.001,0.003]	s
Tempo de proc. por requisição (Comutador)	[0.0005,0.001]	s
Energia gasta em transmissões	[0.00001,0.0001]	$J/(bit \cdot Km)$
Tamanho médio de uma mensagem	12000	bits
Frequência de requisições de um comutador	[416667,833333]	Hz
Custo	[0.0007,0.0017]	1/J
Potência do controlador quando ocioso	[400, 600]	W
Potência do comutador quando ocioso	[200, 300]	W
Energia gasta por mensagem (Controlador)	[0.005, 0.007]	J
Energia gasta por mensagem (Comutador)	[0.001, 0.003]	J

A figura 1(a) mostra comparativamente os tempos de processamento de cada algoritmo aqui descrito. Como esperado, o algoritmo exato explode exponencialmente enquanto as heurísticas seguem um padrão mais comportado, sendo mostradas em detalhe na figura figura 1(b). No caso, são avaliadas instâncias baseadas em grafos completos e cujos nós estão espalhados ao longo de uma área de variações de latitude e de longitude iguais a 10º cada. As latências de cada aresta foram compostas com base no tempo de propagação da luz em cada enlace e no tempo de processamento das mensagens de controle em ambos os dispositivos. As distâncias consideradas foram obtidas pela aplicação da fórmula de *Haversine* considerando, portanto, a curvatura terrestre.

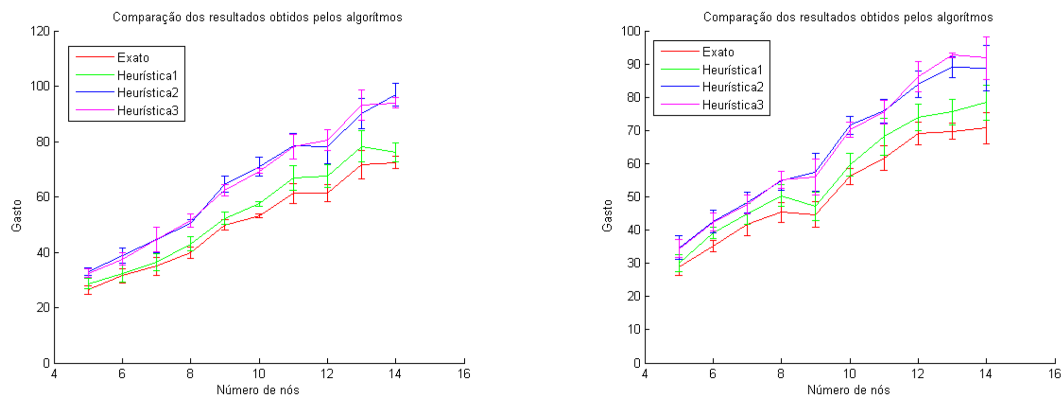
A figura 2, por sua vez, compara os algoritmos no que diz respeito à qualidade de suas respostas. Os gastos aqui representados incluem constantes como os gastos dos comutadores enquanto ociosos e os gastos dos comutadores com o processamento de mensagens de controle. Não é definida uma unidade de medida simplesmente pelo fato de nenhuma moeda em específico ter sido utilizada. A figura 2(a) mostra os resultados utilizando grafos completos, enquanto a figura 2(b) utiliza apenas metade das arestas, sem considerar os *auto-loops*. As topologias foram geradas aleatoriamente a



(a) (b)

Figure 1: Tempos de processamento para redes espalhadas ao longo de áreas de 10ºx10º e baseadas em grafos completos

parir de um número arestas desejado. É possível perceber que em ambos os casos avaliados a *Heurística 1* se mostrou superior às demais.



(a) Grafos completos (b) Grafos com metade das arestas

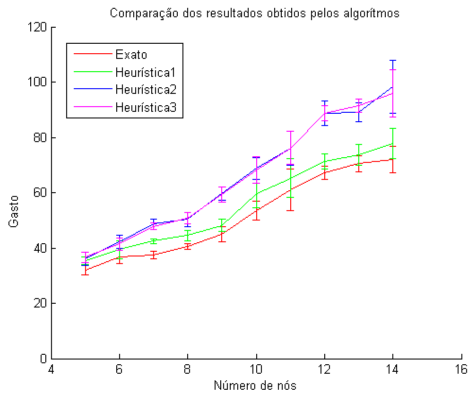
Figure 2: Gastos de energia para redes espalhadas ao longo de áreas de 10ºx10º

A figura 3 considerou um caso bastante extremo em termos de área, variando o posicionamento dos nós ao longo de uma área de variações de latitude e de longitude iguais a 100º cada. Todavia, não há alterações significativas no que diz respeito à superioridade da *Heurística 1* em relação às demais.

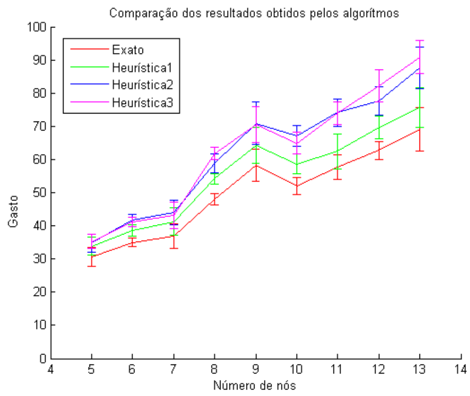
A figura 4 utiliza controladores de capacidades maiores que a soma das demandas de todos os comutadores de forma a avaliar o desempenho dos algoritmos quando a satisfação de demandas e capacidades deixa de ser problemática. Podemos observar que ainda assim a *Heurística 1* se mostra superior às demais e bastante acurada, tendo um erro inferior a 5% em muitas das instâncias consideradas. Nestes casos é possível perceber mais claramente que o custo mínimo cresce linearmente com o aumento do número de nós, apesar dos custos com o processamento e transmissão de mensagens de controle prevalecer (sendo cerca de 89-94% dos custos finais). A figura 5 compara as mesmas execuções no que diz respeito ao tempo de processamento. Podemos observar que nesse caso em particular a *Heurística 1* mostrou-se mais eficiente que as demais, além de continuar retornando a resposta de melhor qualidade.

9 Conclusões

Foi apresentada uma nova visão para o *Problema de Localização de Controladores*, visando otimizar parâmetros financeiros com garantias de qualidade de infra-estrutura de rede. A versão de decisão

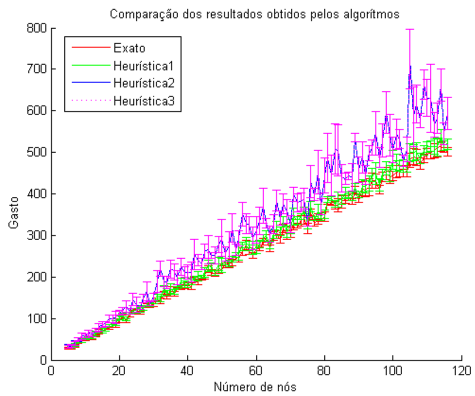


(a) Grafos completos

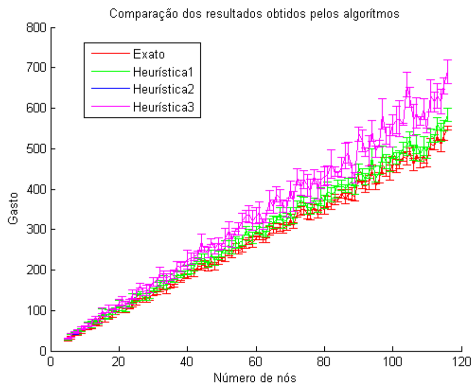


(b) Grafos com metade das arestas

Figure 3: Gastos de energia para redes espalhadas ao longo de áreas de 100ºx100º



(a) Variação de latitude/longitude de 100º e grafos completos



(b) Variação de latitude/longitude de 10º e grafos com metade das arestas

Figure 4: Comparação dos algoritmos sem as restrições contra sobrecarga

do problema foi provada como NP-Completa e, portanto, três heurísticas foram desenvolvidas a fim de superar a eventual natureza exponencial do problema.

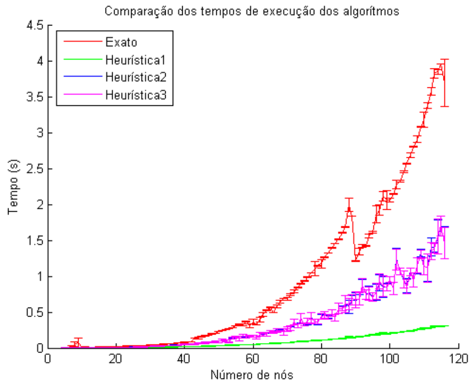
A *Heurística 1* foi a mais eficaz em todos os cenários avaliados, gerando entre 90% e 95% de acurácia. As três heurísticas se mostraram equivalentes em termos assintóticos, todas rodando em $O(|A||V|^3)$.

Apesar de claramente ser mais lenta do que as heurísticas, a solução exata pode ser empregada em casos com uma quantidade de nós condizente com muitas das redes que inspiraram a elaboração deste problema. Como espera-se que a execução do algoritmo seja necessária apenas no ato do projeto da rede, a utilização do algoritmo exato é altamente aconselhável, visto que haverá dinheiro envolvido.

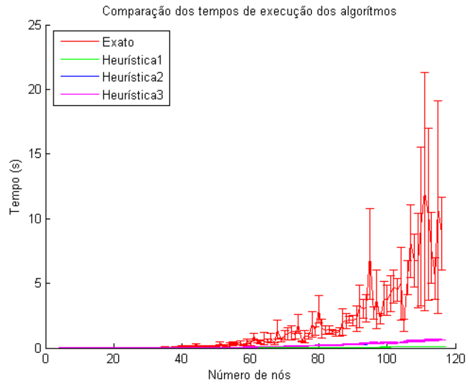
As heurísticas podem ser úteis no caso em que o número de controladores seja propositalmente superdimensionado de forma a adaptar a rede de forma dinâmica. Dessa forma, as heurísticas podem ser empregadas em tempo de execução para decidir quais enlaces e dispositivos devem permanecer ligados ou desligados a fim de gerar menos gastos financeiros ao mesmo tempo em que a rede permanece com certa qualidade de serviço.

References

[1] GLPKgnu linear programming kit. https://www.gnu.org/software/glpk/, 2012. [Online; accessed 23-May-2018].



(a) Variação de latitude/longitude de 100° e grafos completos



(b) Variação de latitude/longitude de 10° e grafos com metade das arestas

Figure 5: Comparação dos algoritmos sem as restrições contra sobrecarga

[2] Dorgival Guedes, L Vieira, M Vieira, Henrique Rodrigues, and Rogério Vinhal Nunes. Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, 30(4):160–210, 2012.

[3] Brandon Heller, Rob Sherwood, and Nick McKeown. The controller placement problem. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 7–12. ACM, 2012.

[4] Yannan Hu, Wendong Wang, Xiangyang Gong, Xirong Que, and Shiduan Cheng. On reliability-optimized controller placement for software-defined networks. *China Communications*, 11(2):38–54, 2014.

[5] Simon Jouet and Dimitrios P Pezaros. Bpfabric: Data plane programmability for software defined networks. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, pages 38–48. IEEE Press, 2017.

[6] Mong-Jen Kao, Han-Lin Chen, and Der-Tsai Lee. Capacitated domination: Problem complexity and approximation algorithms. *Algorithmica*, 72(1):1–43, 2015.

[7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[8] Alejandro Ruiz-Rivera, Kwan-Wu Chin, and Sieteng Soh. Greco: An energy aware controller association algorithm for software defined networks. *IEEE communications letters*, 19(4):541–544, 2015.

[9] Guodong Wang, Yanxiao Zhao, Jun Huang, and Wei Wang. The controller placement problem in software defined networking: a survey. *IEEE Network*, 31(5):21–27, 2017.

[10] Bang Zhang, Xingwei Wang, Lianbo Ma, and Min Huang. Optimal controller placement problem in internet-oriented software defined network. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2016 International Conference on*, pages 481–488. IEEE, 2016.