Uma Nova Representação do Espaço de Soluções para Arrefecimento Simulado em Problemas de Alocação de Circuitos Virtuais

Fernando D. M. Silva[1] **e Luís Henrique M. K. Costa**[1]

[1]GTA/Poli/COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brazil

{fernandodias,luish}@gta.ufrj.br

Abstract. *The optimization of virtual circuits in networks is often solved by heuristics, such as Simulated Annealing (SA), due to the complex topology and demand patterns. To reduce SA's execution time, other techniques have poor representation of the solution space or complex operations. We propose a solution space for SA that finds paths and optimizes the result simultaneously. The proposal represents the entire solution space, with a tradeoff of higher dimensionality. We compare with other heuristics and observe that the proposal finds viable solutions, although it requires more execution time. This representation can be used in scenarios where solutions based solely on κ-shortest paths are insufficient to achieve a near-optimal solution.*

Resumo. *A otimização de circuitos virtuais em redes é muitas vezes resolvidas com heurísticas, como o Arrefecimento Simulado (AS), devido a topologia e demanda complexas. Para agilizar a execução do AS, muitas técnicas utilizam representações insuficientes ou complexas. Esse trabalho define um espaço de soluções para AS que faz a descoberta de caminhos e alocação dos fluxos simultaneamente. A proposta representa todo o espaço de soluções, ao custo de uma dimensionalidade maior. Compara-se a proposta com outras heurísticas e observa-se que a proposta é viável, ao custo de mais tempo para execução. Essa representação tem potencial para cenários onde respostas somente com κ-menores caminhos não são suficientes para atingir resultados quase ótimos.*

1. Introdução

No planejamento de recursos de rede, a crescente demanda de clientes e necessidades de qualidade de serviço distintas faz com que provedores de acesso dependam de técnicas de engenharia de tráfego para a alocação eficiente de circuitos virtuais. Diversas tecnologias para encaminhamento de tráfego como *Software Defined Networking* (SDN) [Agarwal et al. 2013], *Multilayer Protocol Switching with Traffic Engineering* (MPLS-TE) e *Segmented Routing* (SR) [Šeremet e Čaušević 2020] são usados para encaminhamento de fluxo na rede através de caminhos predefinidos. Provedoras de acesso fazem uso dessa tecnologia para firmar contratos de comunicação entre dois pontos de sua rede, muitas vezes entre unidades corporativas, e necessitam que alocação prévia e a reserva de recursos tenha o menor impacto na rede. Problemas de alocação de recursos mais complexos ou não lineares, como no caso de atraso de

fila M/M/1 [Bertsekas e Gallager 1992], não podem ser resolvidos com o uso direto de programação linear. Para isso, são necessárias heurísticas para encontrar uma solução.

Diversos trabalhos otimizam métricas através de heurísticas próprias, que reduzem o espaço de busca ao fazer suposições sobre o problema, que podem não conter a solução ótima. Essas heurísticas utilizam representações que servem apenas a um conjunto limitado de funções objetivo em problemas de rede. O Arrefecimento Simulado (AS, ou *Simulated Annealing*) é uma meta-heurística que pode representar todo o espaço de soluções e pode ser utilizado na solução de diversas funções objetivo com uma mesma representação. Diversos trabalhos utilizam o AS para solução de problemas de otimização não-linear em redes, porém estes fazem o uso de representações menores que reduzem a dimensionalidade do problema e o tornam mais rápidos na execução, a custo de soluções piores.

Este artigo propõe uma definição completa do espaço de soluções do Arrefecimento Simulado para resolver problemas de alocação de fluxos em uma rede. Essa definição é capaz de realizar simultaneamente a descoberta de caminhos e a alocação de recursos entre caminhos descobertos, o que reduz a complexidade das operações. A proposta faz uso das definições lineares de conservação de fluxo e não define a função objetivo, o que permite que esse método possa ser utilizados em diversos problemas de otimização. O objetivo é avaliar a eficácia da solução proposta. Para isso, é feita a implementação de duas funções objetivo, minimização de custo e minimização de atraso de fila, e a proposta será comparada com o problema de programação linear, um algoritmo guloso para descoberta de respostas e uma segunda representação de espaços para Arrefecimento Simulado baseada em κ menores caminhos. Viu-se que a representação proposta encontra respostas viáveis para o problema e encontra resultados melhores em alguns dos casos avaliados, mas necessita de um maior número de iterações para alcançar melhores resultados.

Este trabalho está organizado da seguinte forma: A Seção 2 aborda os trabalhos relacionados. A seção 3 define o escopo do problema a ser resolvido. A Seção 4 apresenta a proposta de definição do espaço de soluções e sua aplicação prática. A Seção 5 apresenta os experimentos para validação da proposta. A Seção 6 discute os resultados obtidos e finalmente a Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

O arrefecimento simulado, assim como algoritmos genéticos, já foi utilizado na literatura para solução de problemas envolvendo engenharia de tráfego e projeto de redes. Ambos os tipos de problemas são relevantes, já que ambos necessitam de uma representação de espaço de soluções. Para trabalhos com algoritmos genéticos, a etapa de mutação pode ser utilizada como a pertubação do arrefecimento simulado, o que será destacado adiante.

O trabalho de [Farrugia et al. 2023] resolve a maximização de fluxos com o uso de algoritmos genéticos. A representação de espaço consiste no cálculo prévio dos κ menores caminhos entre fonte e destino para cada par e a atribuição de um valor de fluxo para cada um desses caminhos, o que limita a exploração de soluções que precisem de caminhos ainda maiores. Em [Pióro e Medhi 2008] os autores apresentam a mesma técnica para resolução de diversos problemas de otimização em redes. Essa técnica será utilizada como comparação nesse trabalho.

O trabalho de [Riedl 2002] utiliza algoritmos genéticos com busca local para minimizar a maior utilização de enlaces da rede. Nesse trabalho, métricas do protocolo EIGRP da cisco são utilizadas como referência, e o resultado consiste na definição dessas métricas por enlace ao invés da definição individual dos caminhos por demanda. Essa representação é utilizada devido a facilidade de integração em redes cuja tecnologia já é utilizada, mas tem a desvantagem de não poder controlar o caminho individual de cada fluxo na rede.

Já [Yaghini et al. 2012] utiliza um híbrido de arrefecimento simulado e programação linear para fazer projeto de redes, onde o espaço de soluções apenas indica se um enlace é utilizado ou não. O calculo de custo é feito a partir da solução do problema de menor custo de múltiplos fluxos através de programação linear, com base nos enlaces selecionados pela resposta atual. Nesse caso, cada iteração do algoritmo necessita solucionar um problema de programação linear, o que pode contribuir para um tempo de execução elevado se comparado com soluções que envolvem apenas uma multiplicação de matrizes.

O trabalho de [La-Roque et al. 2020] resolve o problema de alocação de comprimento de ondas em redes óticas com algoritmos genéticos. Esse trabalho representa a solução com a lista de nós intermediários entre origem e destino, que é calculada com o uso de uma função de descoberta de caminhos. As recombinações são feitas com base nos nós comuns entre indivíduos e a mutação consiste no sorteio de um novo caminho a partir de um ponto intermediário do caminho original sorteado pelo algoritmo. Essa abordagem implementa uma realização completa do espaço de soluções, ao custo de etapas de cálculo com alto poder computacional. Além disso, o corte e sorteio de caminhos não garante vizinhança e pequenas pertubações, o que atrapalharia o processo de exploração no caso do arrefecimento simulado.

O trabalho de [Eren e Ersoy 2002] representa o espaço como uma lista que contém os nós intermediários entre uma origem e um destino. O caminho é formado pela união dos menores caminhos entre a fonte e o primeiro elemento da lista, o primeiro e o segundo, e assim por diante até o último elemento e o destino. Essa abordagem necessita a execução de um algoritmo de descoberta de menor caminho múltiplas vezes por iteração. Essa abordagem exige o uso de algoritmos com alto custo computacional.

Este trabalho apresenta uma novo espaço de soluções para uso com Arrefecimento Simulado em problemas de otimização de alocação de circuitos virtuais ou fluxos. Essa representação difere da literatura ao realizar a descoberta de caminhos e a otimização de forma simultânea, o que não restringe a solução a respostas que contenham caminhos não convencionais (como por exemplo os κ menores caminhos). A contrapartida é o aumento de dimensionalidade, que deve ser compensado com um maior tempo de processamento para respostas viáveis em comparação com as outras soluções.

3. Definição do Problema

Este trabalho restringe-se a problemas de planejamento de recursos de circuitos virtuais em redes, obtidos através de relações contratuais entre clientes e provedores de acesso. Estas relações contratuais mudam com frequência relativamente baixa, portanto são compatíveis com algoritmos que exigem um tempo de computação maior. Assim, as demandas que são definidas para serem otimizadas permanecem as mesmas durante alguns dias,

enquanto a topologia da rede se mantém a mesma durante o processo de alocação de novos recursos.

As variáveis e restrições podem ser modeladas como um problema de fluxos multi-produtos convencional, na qual diversos fluxos $\mathbf{f} = (s,t,d)$, na qual $f \in \mathcal{F}$, são definidos com uma demanda d entre uma fonte s e um destino t. Todos esses fluxos devem ser roteados através da rede, e todos os fluxos devem respeitar a capacidade c_{ij} de cada enlace. Define-se a rede como um grafo *direcionado* $\mathcal{G} = (\mathcal{V},\mathcal{E})$ definido por um grupo de vértices $\mathcal{V}$ e enlaces $\mathcal{E}$, na qual há apenas um enlace que define a conexão entre qualquer par de vértices i e j, ou seja, não existem dois enlaces (i,j) e (j,i) definidos no mesmo grafo. Pode-se definir o problema como:

$$\min_{x_{ij}^f} \quad J(x_{ij}^f) \tag{1a}$$

$$s.t. \quad \sum_{j\in(i,j)} x_{ij}^f - \sum_{j\in(i,j)} x_{ji}^f = \begin{cases} 0, \text{ se } i \ /\{s,t\} \\ +d, \text{ se } i = s \\ -d, \text{ se } i = t \end{cases} \quad \forall i \in \mathcal{V}, f \in \mathcal{F} \tag{1b}$$

$$\sum_f x_{ij}^f \leq c_{ij} \forall (i,j) \in \mathcal{E} \tag{1c}$$

$$x_{ij}^f \in \mathbb{Z} \tag{1d}$$

A variável x_{ij}^f é a quantidade de banda alocada ao enlace que liga o nó i ao nó j para definir o caminho do fluxo f, que é definida como um valor inteiro em (1d) para comportar configurações realísticas. Valores *negativos* de fluxo alocado representam fluxos no *sentido contrário* ao da direção do enlace. As restrições fundamentais, que sempre estão presentes independente do objetivo, do problema consistem na conservação de fluxos na rede, definido no termo (1b), e o limite de capacidade máxima de um enlace, definido no termo (1c) como o somatório de todos os fluxos que passam em um enlace ser menor ou igual a capacidade c_{ij} de cada enlace (i,j).

No problema observado, o termo $J(x_{ij}^f)$ da função objetivo (1a) é o custo em função do fluxo resultante dos enlaces. Para o Arrefecimento Simulado, não há restrições quanto à definição dessa função, que pode ser não linear ou até mesmo não analítica, contanto que resulte em um valor numérico que deve ser minimizado. Duas funções serão avaliadas nesse trabalho: Minimização de custo e a minimização de atraso de fila. A minimização de custo consiste no problema clássico que consiste em associar um valor de custo y_{ij} $[1/\text{Mbps}]$ para cada unidade de banda alocada para um enlace e somar o custo para todos os fluxos da rede. A função de custo resultante é definida como:

$$J(x_{ij}) = \sum_f \sum_{(i,j)\in\mathcal{E}} y_{ij} |x_{ij}^f|, \tag{2}$$

onde x_{ij} é medido em (mas não limitado a) Mbps. O somatório $\sum_{(i,j)\in\mathcal{E}}$ representa o somatório para todos os enlaces (i,j), já $\sum_f$ é o somatório para todos os fluxos. Já a equação de minimização de atraso de fila, baseada no modelo de filas M/M/1, consiste no cálculo do peso dos enlaces em função da fluxo alocado (em pps) e no atraso por enlace resultante da equação

$$y_{ij} = \frac{1}{c_{ij} - \sum_f |x_{ij}^f|}, \tag{3}$$

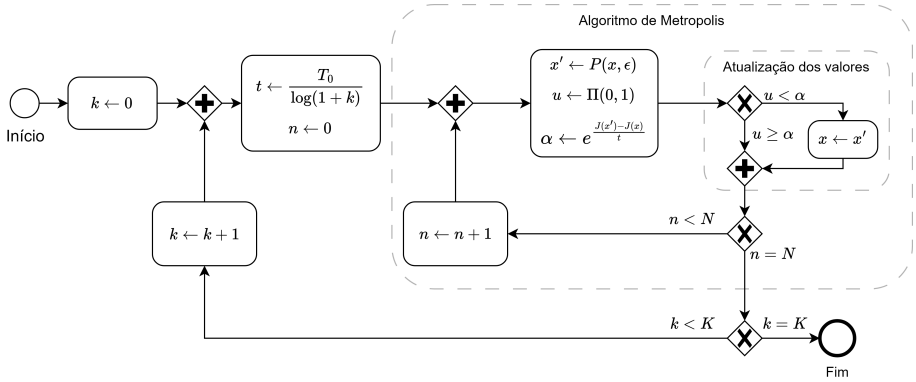


Figura 1. Diagrama de execução do algoritmo de arrefecimento simulado.

que após esse cálculo contabiliza o custo total da rede com a mesma Equação (2). A definição desse modelo é dada com mais detalhes em [Pióro e Medhi 2008].

3.1. Arrefecimento simulado

O arrefecimento simulado é uma técnica de otimização probabilística inspirada na natureza que se simula o processo de tempera de metais. Esse método é baseado no algoritmo de [Metropolis et al. 1953] e foi desenvolvido em [Kirkpatrick et al. 1983] para resolução de problemas de otimização.

Para execução do algoritmo, é necessário definir um espaço de solução $\mathcal{A}$, uma função de custo $J(\mathbf{x}) : \mathcal{A} \rightarrow \mathbb{R}^+$, que recebe uma ou mais entradas $\mathbf{x} \in \mathcal{A}$ e retornar um número real, e uma função de perturbação $P(\mathbf{x},\epsilon) = \mathbf{x}' : \mathcal{A} \rightarrow \mathcal{A}$ que retorna um elemento $\mathbf{x}'$ que está na vizinhança de $\mathbf{x}$, e cujo alcance dessa vizinhança pode ser ajustado em função de um parâmetro ϵ. Através de múltiplas execuções da função de pertubação, deve ser possível alcançar todos os elementos em $\mathcal{A}$.

A execução do algoritmo está descrita na Figura 1. Antes de iniciar, é necessário definir o número de execuções do algoritmo de Metropolis N, o número de etapas de redução de temperatura K, a temperatura inicial T_0 e o ajuste da perturbação ϵ. Para cada iteração de K, a temperatura atual $t(k)$ é calculada a partir da equação:

$$t(k) = \frac{T_0}{\ln(1+k)}, \tag{4}$$

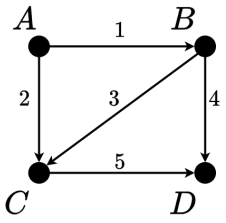
derivada em [Geman e Geman 1984]. Já para cada etapa de execução do algoritmo de Metropolis, a perturbação é aplicada ao estado atual $\mathbf{x}$, resultando em um novo estado potencial $\mathbf{x}'$. Calcula-se a probabilidade de aceitar o novo estado potencial com base no estado atual segundo a equação baseada na distribuição de Boltzmann-Gibbs:

$$e^{-\frac{J(\mathbf{x}')-J(\mathbf{x})}{t(k)}} > u \sim \Pi(0,1), \tag{5}$$

onde u é uma amostra de uma distribuição uniforme entre 0 e 1. Assim, se $J(\mathbf{x}') < J(\mathbf{x})$, o valor de $\mathbf{x}$ sempre é atualizado com o candidato. Caso contrário, ele dependerá do resultado da Equação (5). Esse processo se repete até que ocorram K reduções de temperatura, e cada redução ocorre após N iterações do algoritmo de Metropolis. Ao longo de todo o algoritmo, o resultado $\mathbf{x}^{[\min]}$ com o menor $J(\mathbf{x})$ é sempre salvo.

$$F = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \left[\begin{matrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{matrix}\right] \end{array} \begin{matrix} \\ A \\ B \\ C \\ D \end{matrix}$$

(a) Matriz F resultante.



(b) Rede de teste.

Figura 2. Exemplo de formação da matriz F.

4. Definição do Espaço de Soluções

A definição do espaço de solução é feita a partir das equações de conservação de fluxos definidas na Equação (1b). Considere que $\mathbf{x}^f$ é o vetor com valor de vazão resultante de cada enlace para uma única demanda f. A restrição de conservação de fluxos pode ser reescrita como o sistema linear:

$$F\mathbf{x}^f = \mathbf{d}^f, \tag{6}$$

no qual $F \in \{-1,0,1\}^{n \times m}$ é a matriz de conservação de fluxos e $\mathbf{d}^f \in \mathbb{Z}^n$ é o vetor de demanda. Cada elemento de $\mathbf{d}^f$ representa um nó no grafo e ele codifica a informação de demanda no nó fonte como $+d_f$ e no nó destino como $-d_f$, onde d_f é o valor de demanda solicitado. Assim, cada equação representa a conservação de fluxos resultante de um nó, e qualquer valor $\mathbf{x}^f$ proposto que seja uma resposta dessa equação respeitará a conservação de fluxos em toda a rede. Cada linha da matriz F representa um vértice da rede analisada e cada coluna da matriz F é um vetor que indica a origem e o destino de cada enlace. Podemos comparar a construção de uma matriz F com uma topologia de exemplo na Figura 2.

Para cada demanda, a solução $\mathbf{x}^f$ pode ser separada em uma solução particular $\mathbf{x}_p^f$ e uma solução $\mathbf{x}_0^f$ no espaço nulo de F, representado pela matriz $N_F \in \mathbb{Z}^{k \times m}$. Assim, podemos encontrar uma solução para $\mathbf{x}^f$ a partir da equação:

$$\mathbf{x}^f = \mathbf{x}_p^f + N_F\mathbf{y}, \tag{7}$$

onde $\mathbf{y} \in \mathbb{Z}^k$ são as variáveis livres que serão utilizadas no algoritmo. A matriz N_F depende unicamente da topologia da rede, portanto o cálculo de N_F é realizado apenas uma vez para todos os fluxos.

Existem diferentes técnicas para obtenção de $\mathbf{x}_p$ e N_F. A técnica utilizada nesse trabalho para a obtenção dos valores de $\mathbf{x}_p$ e N_F consiste na obtenção da forma escalonada reduzida de F, que separa as variáveis independentes de $\mathbf{x}$ (que se tornam as variáveis de $\mathbf{y}$) e as transformações necessárias para a transformação em um resultado $\mathbf{x}$, já com a adição dos termos independentes que fazem parte da solução particular $\mathbf{x}_p$. O processo de eliminação pode ser feito com a matriz aumentada $\left[F|I\right]$ para obter uma matriz elementar M resultante da decomposição. O uso dessa matriz elementar permite que as operações

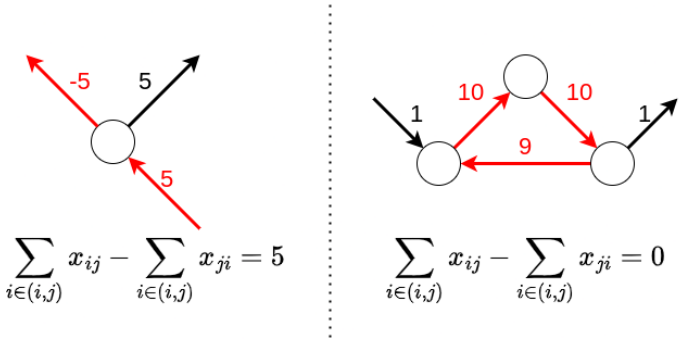


Figura 3. Exemplos de tráfego "fantasma": soluções que respeitam a conservação de fluxo mas contêm tráfegos impossíveis (em vermelho).

feitas em F possam ser replicadas em qualquer vetor de demanda, o que resulta nos termos independentes sem a necessidade de repetir o processo de decomposição. Assim, esse método tem a vantagem de só precisar recalcular N_F se houver mudanças da topologia da rede.

4.1. Soluções viáveis e tráfego fantasma

Segundo o Teorema 6.4 de [Lima 2020], pode-se concluir que a representação em (7) inclui todas as possíveis respostas que satisfaçam a Equação (6), o que inclui todas as possíveis combinações de caminhos que levem a demanda entre fonte e destino, incluindo o valor ótimo para qualquer problema de otimização solucionável. Essa representação é completa, porém possui um espaço mais amplo do que o necessário, com respostas que não fazem sentido. O exemplo da Figura 3 ilustra essa situação: Ciclos fechados que respeitam a conservação de fluxos são respostas válidas, mesmo que esse tráfego tenha origem ou destino incompatíveis com a demanda avaliada. Esse tipo de tráfego aumenta o valor de custo e pode fazer com que uma resposta pareça violar as restrições quando não está.

O tráfego fantasma é reduzido ao longo do algoritmo se a função de custo for construída de modo que esse tráfego $\mathbf{x}^\dagger$ seja aditivo às soluções válidas $\mathbf{x}$, ou seja, $J(\mathbf{x} + \mathbf{x}^\dagger) > J(\mathbf{x})$. Sob essa condição, espera-se que essas anomalias sejam minimizadas ao longo da execução do algoritmo, já que o AS prioriza respostas com menor $J(\mathbf{x})$. Porém, isso não impede que esses tráfegos ainda apareçam na resposta final, que mesmo assim pode ser válida. É preciso realizar um pós processamento para exclusão desses tráfegos.

4.2. Definições de restrição e perturbação

Para implementar o Arrefecimento Simulado no espaço proposto, é necessário definir a função de pertubação e a implementação de restrições. A primeira define a função $P(\mathbf{x},\epsilon)$ que leva a resposta atual a uma resposta na vizinhança e a segunda permite implementar restrições ao problema que ainda não foram implementadas, tal como a restrição de capacidade.

A função de perturbação consiste em sortear ϵ elementos da solução completa $Y \in \mathbb{Z}^{|\mathcal{E}|\times|\mathcal{F}|}$ e, com igual probabilidade, somar um ou subtrair um do valor atual desse elemento. Esse método não possui um viés e, com suscetivas repetições dessa perturbação, é possível explorar todo o espaço de soluções. A escolha da função de

perturbação é arbitrária, já que o efeito dessa função na execução do algoritmo não pode ser previsto antecipadamente.

A definição da restrição de capacidade é feita através do acréscimo de uma penalização por violação na função objetivo. A função objetivo efetiva é escrita como

$$J'(X) = (J(X))^p + M||\mathbf{r}||_2^2, \tag{8}$$

em que o termo $M||\mathbf{r}||_2^2$ acrescenta uma penalização M proporcional a magnitude do vetor $\mathbf{r} \in \mathbb{Z}^{|\mathcal{E}|}$, definido como o excedente de demanda além da capacidade em cada enlace, ou seja: $r_{ij} := \max(\sum_f |x_{ij}^f| - c_{ij}, 0)$. O termo M grande aumenta a magnitude do termo de penalização e faz com que a convergência seja para soluções viáveis que, uma vez encontradas, a probabilidade de retornar à região inviável segundo a Equação (5) é baixa. O termo $||\mathbf{r}||_2^2$ pode ser acompanhado ao longo da execução do algoritmo para determinar se a solução encontrada é viável ou não. Finalmente, a função objetivo é elevada a um fator p para aumentar ou reduzir a magnitude das variações causadas pela mudança nas respostas, o que impacta no grau de aceitação de novas respostas candidatas.

5. Experimentos

Os experimentos realizados observam o comportamento do arrefecimento simulado na solução de problemas com a representação de espaço proposta. Os dois problemas apresentados na Seção 3 serão analisados: Minimização de custo e o atraso de fila. Para cada problema, duas quantidades de demandas são utilizadas em três topologias distintas para avaliar quatro algoritmos: Programação Linear, um algoritmo guloso, o Arrefecimento Simulado com o uso de κ menores caminhos e o Arrefecimento Simulado com a solução proposta. A Programação Linear é utilizada somente no problema de minimização de custo, como forma de obter o valor ótimo e comparar com os resultados do algoritmo guloso e do Arrefecimento Simulado. Já no problema atraso de fila, são comparados os resultados apenas do algoritmo guloso e do Arrefecimento Simulado.

Os hiperparâmetros para o Arrefecimento Simulado foram obtidos através de otimização Bayesiana [Frazier 2018]. A otimização Bayesiana explora o espaço de hiperparâmetros de modo que os próximos pontos são escolhidos através das regiões de maior incerteza após a consideração dos valores dos pontos anteriores. O algoritmo aceita apenas valores reais como hiperparâmetros de ajuste, portanto a seguinte modificação foi necessária: Os parâmetros do AS são definidos como $T = 10^\tau$, $\epsilon = 10^e$, $K = \lceil 10^k \rceil$ e $N = \lceil C/10^k \rceil$. C é definido como um valor constante para que todas as execuções tenham aproximadamente o mesmo número de passos. Finalmente, temos que $C = 10^6$, $\tau \in [0,3]$, $e \in [0,3]$, e $k \in [1,3]$. O valor de C foi escolhido de modo que o problema mais lento de se executar demorasse em média 30 minutos.

Para avaliar os resultados dos algoritmos, compara-se o valor encontrado por cada uma das soluções através do método de otimização mencionado. Os menores valores encontrados durante a otimização Bayesiana são utilizados para comparar os resultados com o algoritmo guloso e o de programação linear.

Devido à natureza probabilística do Arrefecimento Simulado, uma resposta viável não é garantida em todas as execuções. Portanto, para avaliar a eficácia em obter soluções viáveis, executa-se repetidas vezes o algoritmo com os parâmetros encontrados e verifica-se a razão de execuções para encontro de soluções viáveis, o tempo médio para encontrar

uma solução viável e o tempo médio de melhora da solução. O tempo para a solução viável é medido a partir do início da execução do algoritmo até o momento que $||\mathbf{r}||_2^2 = 0$, como visto na Equação (8). O tempo médio de melhoria mede, a partir do momento que $||r||_2^2 = 0$, o tempo médio para que a solução atual seja substituída por uma solução melhor. Para isso, conta-se quantas melhorias foram obtidas entre a descoberta da primeira solução viável até o fim da execução do algoritmo.

O código que implementa o algoritmo do arrefecimento simulado foi escrito em `Python3`, com o uso das bibliotecas `Numpy` [Harris et al. 2020] para computação matricial e `Matplotlib` [Hunter 2007] para visualização dos gráficos. A biblioteca `Networkx` [Hagberg et al. 2008] define classes e diversos atributos para manipulação de grafos em código, e é utilizada para interpretar os grafos salvos em arquivos, calcular os κ menores caminhos e gerar as matrizes necessárias a partir das informações do grafo. O processo de decomposição descrito na Seção 4 foi feito com o uso da biblioteca `Sympy` [Meurer et al. 2017] que faz a decomposição com computação simbólica. Todo o código com os experimentos estão no repositório do Github[1].

5.1. Topologias de redes utilizadas

São utilizados 3 grafos nesse trabalho: A topologia da Rede Nacional de Ensino e Pesquisa (RNP)[2], da *Gigabit European Academic Network* (GEANT)[3] e uma terceira gerada aleatoriamente. Para cada uma das topologias apresentadas, são sorteados 100 e 500 fluxos distintos, com demanda $d = 1$, para serem roteados através da rede. Para as redes da RNP e da GÉANT, os pesos dos enlaces são proporcionais ao atraso de propagação, já para a terceira topologia é aleatório.

A rede aleatória é assim construída: 60 nós são inicialmente conectados usando 60 enlaces de modo a formar um único anel com todos os nós. Após isso, 40 pares de nós são sorteados aleatoriamente para adicionar enlaces que conectem entre si. Esse processo de formação garante que todos os nós são alcançáveis por todos os outros nós.

Para comparar as redes, é feita uma análise preliminar na Tabela 1. Nessa tabela pode-se comparar o número de vértices, enlaces, a razão entre eles, o tamanho médio dos caminhos, a conectividade média dos nós, a quantidade de variáveis livres encontradas segundo o espaço de soluções proposto, e as características do problema de menor custo, assim como o valor ótimo. A escolha da razão de nó e de enlaces na rede aleatória manteve a mesma proporção observada nas redes reais, o que também aproximou o tamanho médio dos menores caminhos. A magnitude das respostas ótimas evidencia a necessidade da compensação p vista na Equação (8) para que o resultado da Equação (5) não resulte em um *overflow*. Em cenários reais onde o valor ótimo é desconhecido, esse valor pode ser ajustado posteriormente, se forem observadas falhas nas execuções prévias do algoritmo.

O número de variáveis livres de $\mathbf{y}$ encontrado na Tabela 1 mostra um comportamento esperado de crescimento proporcional ao tamanho da rede. Esse é o número de variáveis necessárias para representar todos os possíveis caminhos que a demanda pode percorrer. Assim, aumentar o número dos κ caminhos para um valor além desse limite faz

[1] https://github.com/Fdms-3741/sa-mcfp

[2] Detalhes da rede disponíveis no site: https://www.rnp.br/sistema-rnp/infraestrutura-para-pesquisa/evolucao-da-rede-ipe/.

[3] Detalhes da rede GÉANT disponíveis no site: https://network.geant.org/gn4-3n/.

Tabela 1. Caracterização das redes de teste.

	Topologia de Rede		
Métrica	RNP	GÉANT	Aleatório
$\|\mathcal{V}\|$	27	42	60
$\|\mathcal{E}\|$	33	68	100
$\|\mathcal{V}\|/\|\mathcal{E}\|$	1,22	1,61	1,66
Tamanho médio menor caminho	3,80	3,29	3,54
Conectividade dos nós média	1,77	1,70	2,64
Variáveis livres ($\dim(\mathbf{y})$)	7	27	41
Faixa de custo	(141, 2567)	(55, 3614)	(1, 9)
Faixa de capacidade	100 fixo	100 fixo	(100, 900)
Resultado ótimo PL (100 contratos)	247536	211316	1614
Resultado ótimo PL (500 contratos)	1392561	1015078	7683

com que o número de operações por iteração seja maior do que o número com a solução proposta.

5.2. Heurísticas de comparação

Duas heurísticas são utilizadas nesse trabalho: Um algoritmo guloso e a representação de Arrefecimento Simulado com base nos κ menores caminhos.

A representação por κ-caminhos, como descrito em [Pióro e Medhi 2008], consiste no cálculo prévio de κ caminhos entre fonte e destino, por demanda. Para cada demanda, um vetor $\mathbf{u} = (u_1, u_2, u_3, \ldots, u_\kappa)$ representa a solução atual, que consiste na quantidade de fluxo que passa por cada caminho. Uma matriz auxiliar e binária δ_{edp} mapeia, para cada demanda d e para cada caminho p, os enlaces e contidos nos caminhos para obtenção dos valores de fluxo x_{ij}^f com base nos valores de u_i para cada caminho pré-calculado.

O algoritmo guloso executa da seguinte forma: O algoritmo inicializa com o vetor $\mathbf{u}$ zerado para todos os contratos. Para cada contrato, cuja ordem de preenchimento é aleatória, ele acrescenta uma unidade de demanda em cada caminho e avalia o aumento de custo. Por último, algoritmo escolhe o caminho que menor acresce ao valor de custo atual e não viola nenhuma restrição. O processo é repetido até que $\mathbf{u} = \mathbf{d}$ ou seja impossível de continuar devido a restrições. Já o arrefecimento simulado com κ menores caminhos simplesmente utiliza como entrada o vetor $\mathbf{u}$ de todos os contratos como entrada, e realiza como função de pertubação o mesmo processo descrito na Seção 4.2.

6. Resultados

Os valores de custo total, resultante da equação (2) com custos y_{ij} arbitrários, para cada algoritmo no problema de minimização de custo podem ser vistos na Tabela 2. Nessa tabela, a comparação percentual é feita em função do resultado obtido através de programação linear para uma rede e um conjunto de demandas, como apresentado na Tabela 1. A Tabela 2 mostra que o algoritmo guloso, mesmo com sua simplicidade, é capaz de obter respostas melhores do que ambas as alternativas de execução do Arrefecimento Simulado, porém falha em obter uma resposta viável em um dos casos. Observa-se também

Tabela 2. Resultados encontrados para o problema de minimização de custo. A variação percentual é comparada com o valor ótimo obtido por programação linear (PL). Os resultados com '-' não obtiveram soluções viáveis.

		Algoritmo		
Demanda	**Rede**	Guloso	AS κ-caminhos	AS proposto
100	RNP	249354(+0,7%)	271608(+9,7%)	272915 (+10,2%)
	GÉANT	217538(+2,94%)	46948 (+22,21%)	712665 (+237,25%)
	Aleatório	1668 (+3,34%)	1842 (+14,12%)	26630 (+1549,9%)
500	RNP	-	2170819 (+55,88%)	2048097 (+47,07%)
	GÉANT	1036857 (+2,14%)	1691061 (+66,59%)	-
	Aleatório	7889 (+2,68%)	11327 (+47,42%)	-

Tabela 3. Resultados encontrados para o problema de atraso de fila. Os valores são comparados com o algoritmo guloso. Os resultados com '-' não obtiveram soluções viáveis.

			Algoritmo	
Demanda	**Rede**	Guloso	AS κ-caminhos	AS proposto
100	GÉANT	259,71	305,44 (+17,6%)	153,93 (-40,73%)
500	GÉANT	1908,90	-	-
100	Aleatório	119,57	148,47 (+24,16%)	270,26 (+126,02%)
500	Aleatório	616,28	871,21 (+41,36%)	-
100	RNP	146,36	225,63 (+54,16%)	49,04 (-66,49%)
500	RNP	3575,05	-	187,52 (-94,7%)

que a variação percentual em relação ao valor ótimo varia fortemente entre redes avaliadas, com uma tendência de aumento proporcional a complexidade do problema. Isso pode ser associado a um tempo de execução insuficiente para a resolução de problemas mais complexos.

Já a Tabela 3 mostra os resultados para minimização do atraso de fila, que utiliza como custo dos enlaces a equação (3), onde os valores percentuais comparam os resultados com o algoritmo guloso. Nesse caso, o Arrefecimento Simulado com o método proposto foi capaz de encontrar respostas melhores do que o Algoritmo Guloso em três dos seis problemas avaliados. O uso de AS com κ-caminhos obteve vantagem em apenas dois dos cenários: Um onde obteve uma resposta melhor que o AS proposto mas pior que o algoritmo guloso e outro onde esse foi o único que encontrou respostas viáveis. Esse resultado apresentou uma vantagem do Arrefecimento Simulado para solução de problemas não-lineares, e o método proposto obteve vantagem na busca de novos resultados.

Na Tabela 4, vemos a razão do número de execuções que encontraram valores viáveis sobre o número total de execuções, representado na coluna de repetições. Essa tabela apresenta a razão entre todas as execuções que encontraram uma solução viável sobre todas as execuções realizadas para um determinado método e grafo, independente dos parâmetros escolhidos. Como esses problemas não representam nenhum congestionamento severo ou sobrecarga de rede, todas as execuções do AS para κ-caminhos encontraram respostas viáveis nas redes apresentadas. O método proposto não encontrou

Tabela 4. Taxa de encontro de soluções viáveis para o AS proposto.

Nº contratos	Rede	Taxa	Repetições
100	RNP	89,1%	1479
	GÉANT	52,3%	822
	Aleatório	44,1%	396
500	RNP	1,3%	501
	GÉANT	0%	183
	Aleatório	0%	92

Tabela 5. Tempo médio para encontrar a solução viável para o método proposto.

		Guloso	AS proposto	
Nº de contratos	Rede	Tempo para resposta (s)	Tempo para viável (s)	Tempo de melhoria (s)
100	RNP	$0{,}16 \pm 0{,}04$	$18{,}11 \pm 1{,}10$	$1{,}38 \pm 0{,}10$
	GÉANT	$0{,}24 \pm 0{,}09$	$107{,}83 \pm 11{,}44$	$2{,}04 \pm 0{,}24$
	Aleatório	$0{,}09 \pm 0{,}03$	$102{,}54 \pm 3{,}58$	$0{,}19 \pm 0{,}05$
500	RNP	$3{,}95 \pm 3{,}95$	$356{,}99 \pm 37{,}37$	$1{,}91 \pm 1{,}46$

soluções viáveis em todas as execuções, e a razão de respostas diminui a medida que o número de variáveis do problema aumenta. Essa condição novamente indica a necessidade de um aumento no número de iterações, além apontar a necessidade de uma busca por hiperparâmetros mais extensa.

Finalmente, os tempos para encontro da solução viável e de melhoria podem ser vistos na Tabela 5. Os resultados apresentados são apenas para o algoritmo guloso e o AS proposto, já que o AS por κ-caminhos encontra uma solução viável na primeira execução. Essa tabela mostra uma das desvantagens pela escolha do método: O tempo de execução é maior que o tempo para a execução das heurísticas. Também pode-se ver que o tempo aumenta consideravelmente com o aumento no número de variáveis, exceto entre os problemas da rede GÉANT e da rede aleatória. Porém, esse fenômeno não pode ser observado com a mesma intensidade no tempo médio para melhoria, que manteve todos os resultados próximos a 2 segundos. Esse tempo de melhoria rápido implica que, durante essa parte final da execução do algoritmo, o algoritmo ainda estava encontrando respostas melhores e ainda não atingiu um "congelamento", onde novos estados não são mais encontrados ou a resposta não melhora mais. Isso é mais um indício que o tempo de execução do problema escolhido é insuficiente.

7. Conclusões

Esse trabalho apresenta uma representação do espaço de soluções que pode ser utilizada no algoritmo de arrefecimento simulado para solução de alocação de circuitos virtuais em redes. A representação consiste no uso das variáveis livres da base do espaço nulo do sistema de conservação de fluxos para encontrar soluções para os fluxos de cada enlace. Essa representação faz a descoberta de caminhos e alocação de recursos de forma simultânea, de modo que todo o espaço de soluções possa ser explorado em um processo que requer poucas iterações.

Esse artigo mostra que a solução atual possui uma representação ampla do espaço de soluções, uma consequência da descoberta de caminho e da otimização conjunta, e como penalidade acaba exigindo um número de iterações de execução do algoritmo bem maior quando comparado ao algoritmo guloso e ao método de κ-menores caminhos. Os métodos de Arrefecimento Simulado obtiveram uma vantagem maior nos problemas não lineares, onde o algoritmo guloso foi insuficiente. Devido as características de execução, esse algoritmo está condicionado a problemas que não exigem respostas em tempos muito curtos e podem aproveitar do método para obter resultados melhores. Esse algoritmo possui a vantagem de poder explorar soluções que as representações mais enxutas não são capazes de explorar, e chega a encontrar resultados melhores que o κ caminhos em alguns dos cenários avaliados.

Como próximos passos, novas funções de pertubação e de custo podem ser exploradas para validar o método. Variações do arrefecimento simulado, como algoritmos genéticos e aprendizado por reforço também podem ser exploradas com o uso desse espaço para observar seu comportamento. Finalmente, utilizar esse método em conjunto com heurísticas em simulações de comportamento da rede podem ajudar a compreender os benefícios das soluções mais eficientes em implementações reais.

8. Agradecimentos

O presente trabalho foi realizado com o apoio do CNPq, da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES), código de financiamento 001, da FAPERJ (E-26/204.122/2024), da FAPESP (2023/00811-0 e 2023/00673-7) e da Fundação de Desenvolvimento da Pesquisa - Fundep - Rota 2030.

Ao professor Rodrigo de Souza Couto por providenciar o código com as topologias das redes GÉANT e RNP.

Referências

Agarwal, S., Kodialam, M., e Lakshman, T. V. (2013). Traffic engineering in software defined networks. Em *2013 Proceedings IEEE INFOCOM*, páginas 2211–2219.

Bertsekas, D. P. e Gallager, R. G. (1992). *Data Networks*. Prentice-Hall International Editions. Prentice-Hall International, Englewood Cliffs, NJ, 2. ed edition.

Eren, M. e Ersoy, C. (2002). Optimal Virtual Path Routing Using a Parallel Annealed Genetic Algorithm.

Farrugia, N., Briffa, J. A., e Buttigieg, V. (2023). Solving the multicommodity flow problem using an evolutionary routing algorithm in a computer network environment. *PLOS ONE*, 18(4):e0278317.

Frazier, P. I. (2018). A Tutorial on Bayesian Optimization. arXiv:1807.02811 [stat].

Geman, S. e Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

Hagberg, A. A., Schult, D. A., e Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. Em Varoquaux, G., Vaught, T., e Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, páginas 11 – 15, Pasadena, CA USA.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., e Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

La-Roque, E., Batista, C., e Araújo, J. (2020). A Parallel Strategy for a Genetic Algorithm in Routing Wavelength Assignment Problem Using GPU with CUDA. Em *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, páginas 740–751. SBC.

Lima, E. L. (2020). *Álgebra Linear*. Coleção Matemática Universitária. Associação Instituto Nacional de Matemática Pura e Aplicada, Rio de janeiro, RJ, 10 edition.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., e Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., e Scopatz, A. (2017). Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Pióro, M. e Medhi, D. (2008). *Routing, Flow, and Capacity Design in Communication and Computer Networks*. The Morgan Kaufmann Series in Networking. Elsevier/Morgan Kaufmann, Amsterdam, nachdr. edition.

Riedl, A. (2002). A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics. Em *IEEE Workshop on IP Operations and Management*, páginas 166–170.

Šeremet, I. e Čaušević, S. (2020). Advancing Multiprotocol Label Switching Traffic Engineering with Segment Routing in Software Defined Network environment. Em *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)*, páginas 1–6.

Yaghini, M., Momeni, M., e Sarmadi, M. (2012). A Simplex-based simulated annealing algorithm for node-arc capacitated multicommodity network design. *Applied Soft Computing*, 12(9):2997–3003.