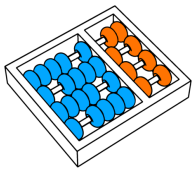
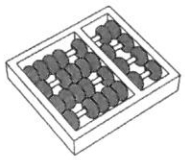


Igor Rosberg de Medeiros Silva

“Algoritmos para o problema do mapeamento de redes virtuais.”

CAMPINAS
2014



Universidade Estadual de Campinas
Instituto de Computação

Igor Rosberg de Medeiros Silva

"Algoritmos para o problema do mapeamento de redes virtuais."

Orientador(a): **Prof. Dr. Eduardo Cândido Xavier**

Co-Orientador(a): **Prof. Dr. Nelson Luis Saldanha da Fonseca**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Igor Rosberg de Medeiros Silva, sob orientação de Prof. Dr. Eduardo Cândido Xavier.

Assinatura do Orientador(a)

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

Si38a

Silva, Igor Rosberg de Medeiros, 1986-
Algoritmos para o problema do mapeamento de redes virtuais / Igor Rosberg de Medeiros Silva. – Campinas, SP : [s.n.], 2014.

Orientador: Eduardo Cândido Xavier.
Coorientador: Nelson Luis Saldanha da Fonseca.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Redes de computadores. 2. Virtualização de redes. 3. Algoritmos. I. Xavier, Eduardo Cândido,1979-. II. Fonseca, Nelson Luis Saldanha da,1961-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Algorithms for the virtual network embedding problem
Palavras-chave em inglês:
Computer networks
Network virtualization
Algorithms
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Eduardo Cândido Xavier [Orientador]
Daniel Macêdo Batista
Fábio Luiz Usberti
Data de defesa: 27-02-2014
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Defesa de Dissertação de Mestrado apresentada por **Igor Rosberg de Medeiros Silva** Defendida e Aprovada em 27 de fevereiro de 2014, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Daniel Macêdo Batista
IME-USP

Prof. Dr. Fabio Luiz Usberti
IC / UNICAMP

Prof. Dr. Eduardo Candido Xavier
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Algoritmos para o problema do mapeamento de redes virtuais.

Igor Rosberg de Medeiros Silva[1]

27 de fevereiro de 2014

Banca Examinadora:

- Prof. Dr. Eduardo Cândido Xavier (Supervisor/*Orientador*)
- Prof. Dr. Fábio Luiz Usberti
  Institute of Computing - UNICAMP
- Prof. Dr. Daniel Macêdo Batista
  IME-USP
-

[1]Financiamento: CAPES (03/2012–09/2012) e FAPESP (processo 2012/14845-0 - 10/2012–02/2014)

Abstract

In recent years, network virtualization has gained considerable attention from the scientific community, since it provides mechanisms to overcome the ossification problem of the current Internet architecture. Through separation of Internet Service Providers into Infrastructure Providers and Service Providers, network virtualization allows multiple heterogeneous virtual networks to share the same physical substrate. One of the main problems regarding network virtualization is the Network Embedding Problem, which is NP-Hard. Several algorithms and heuristics to find a set of good mappings that optimize the use of the bandwidth in substrate networks have been proposed. In this work, we present two new embedding heuristics based on the Tabu Search metaheuristic: the VNE-TS and VNE-TS-Clustering. We also propose a virtual network selection policy, the 2ks-VN-Selector, based on the Bidimensional Knapsack Problem, which aims to increase the profit of Infrastructure Providers. We compare the results obtained by using the VNE-TS and VNE-TS-Clustering heuristics, to those obtained by using the VNE-PSO, one of the best heuristics proposed in the literature for the Embedding Problem. We also compare the effects of the 2ks-VN-Selector with those obtained by using another well known selection policy: Most Prize First. Results show that both VNE-TS and VNE-TS-Clustering reject less virtual network requests than VNE-PSO and that the selection algorithm 2ks-VN-Selector is able to increase the profit of Infrastructure Providers when compared to the Most Prize First algorithm.

Resumo

Virtualização de Redes tem recebido recentemente atenção da comunidade científica, uma vez que ela provê mecanismos para lidar com o problema da ossificação da atual arquitetura da Internet. Através da decomposição de Provedores de Serviço de Internet em Provedores de Infraestrutura e Provedores de Serviço, a Virtualização de Redes permite que várias redes virtuais heterogêneas compartilhem o mesmo substrato físico. Um dos principais problemas relacionados à Virtualização de Redes é o Problema do Mapeamento de Redes Virtuais no substrato, que é NP-Difícil. Muitos algoritmos e heurísticas para encontrar bons mapeamentos, de modo a otimizar o uso da banda passante na rede física, têm sido propostos. Neste trabalho, apresentam-se dois novos algoritmos baseados na metaheurística Busca Tabu, o VNE-TS e o VNE-TS-*Clustering*. Propõe-se também um algoritmo de seleção de redes virtuais, o 2ks-VN-*Selector*, que se baseia no Problema da Mochila Bidimensional, cujo objetivo é aumentar o rendimento em Provedores de Infraestrutura. Os resultados obtidos pelos uso das heurísticas VNE-TS e VNE-TS-*Clustering*, são comparandos com os resultados obtidos pelo algoritmo VNE-PSO, uma das melhores heurísticas de mapeamento proposta na literatura para o Problema do Mapeamento de Redes Virtuais. São comparados, também, os resultados da política de seleção 2ks-VN-*Selector* com os obtidos pela política *Most Prize First*. Resultados mostram tanto VNE-TS quanto VNE-TS-*Clustering* rejeitam menos requisições do que o VNE-PSO e que o algoritmo de seleção 2ks-VN-*Selector* é capaz de aumentar o rendimento de Provedores de Infraestrutura em relação ao algoritmo *Most Prize First*.

Agradecimentos

Aos meus pais, por todo o apoio e educação que me trouxeram até aqui.

A todos os meus familiares que de algum modo contribuíram com a minha formação humana e acadêmica.

Aos Professores Eduardo Xavier e Nelson Fonseca, pela excelente orientação concedida durante o desenvolvimento deste trabalho.

A todos os professores que tive durante a minha formação acadêmica, sem os quais a conclusão deste trabalho não seria possível.

Finalmente agradeço a todos os meus colegas de classe, de trabalho e amigos que tornaram o caminho até aqui mais prazeroso.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Este Capítulo apresenta conceitos elementares relacionados à virtualização e mapeamento de redes. A Seção 1.1 apresenta o problema do mapeamento de redes virtuais, bem como apresenta conceitos gerais relacionados ao tema. A Seção 1.2 faz uma rápida introdução sobre as técnicas e algoritmos aplicados nas soluções propostas. Na Seção 1.3, apresenta-se a organização dos próximos capítulos.

1.1 Virtualização de Redes e o Problema do Mapeamento

Desde a sua criação, a Internet tem sido bem sucedida em prover formas de acesso e troca de informações. Ao passar de três décadas, a arquitetura da Internet tem provado seu valor tornando possível uma infinidade de aplicações distribuídas. Entretanto, a arquitetura inicial da Internet já não permite a implementação fácil de novas funcionalidades [1]. Adotar uma nova arquitetura ou modificar a atual requer consenso entre várias partes, cujos interesses muitas vezes divergem. Em virtude disso, a Internet tem passado por simples atualizações em detrimento de alterações radicais [1]. Em consequência dessa impossibilidade de mudança, o projeto e implantação de redes virtuais têm sido empregados para viabilizar a solução de muitos problemas encontrados na Internet.

A comunidade científica que estuda a Internet tem se dividido em dois grupos: puristas arquiteturais e pluralistas. O primeiro grupo acredita que a virtualização de redes é apenas um meio para avaliar novas arquiteturas, ao passo que o segundo considera a virtualização como um atributo fundamental da própria arquitetura da Internet [2]. De acordo com os pluralistas, a virtualização de redes pode atenuar os efeitos da ossificação da Internet e estimular inovação, possibilitando que diversas arquiteturas de rede coexistam em uma mesma Rede Física (*Substract Network* - SN).

Em várias áreas da computação, a separação de política e mecanismo tem sido empregada como um meio para se alcançar diversidade. No estudo de virtualização de redes, tal separação também tem sido proposta [3, 4]. Nesse contexto, o papel dos tradicionais provedores de serviço de Internet, conhecidos como *Internet Service Providers* - ISPs, é dividido em: provedores de infraestrutura e provedores de serviços.

Os provedores de infraestrutura, conhecidos como InPs (*Infrastructure Providers*), gerenciam a infraestrutura física, ao passo que provedores de serviço (*Service Providers* - SP) gerenciam redes virtuais (*Virtual Networks* - VNs) alocando recursos de um ou mais provedores de infraestrutura. Nesse cenário, são os provedores de serviço que oferecem serviços fim-a-fim aos usuários finais. Esse ambiente promove o desenvolvimento de múltiplas arquiteturas de redes distintas que podem coexistir, superando limites existentes na Internet [1]. Por exemplo, dois SPs, SP1 e SP2, podem propor duas arquiteturas de rede completamente distintas, com protocolos diferentes, usando a mesma infraestrutura física.

Frequentemente, o conceito de múltiplas redes lógicas coexistentes tem aparecido na literatura. Esse conceito pode ser categorizado em quatro principais classes: VLANs (*Virtual Local Area Networks*) , VPNs (*Virtual Private Networks*), Redes Programáveis (*Programmable Networks*) e Redes Sobrepostas (*Overlay Networks*) [5]. Este trabalho lida com Redes Sobrepostas e questões técnicas envolvidas em sua implantação. Mais precisamente, a dissertação tem como foco problemas NP-difíceis relacionados às redes virtuais sobrepostas, mais especificamente no problema relacionado ao mapeamento de elementos virtuais em elementos físicos.

Uma rede sobreposta consiste em uma rede virtual de computadores, que cria uma topologia virtual sobre uma topologia física. Os nós de uma mesma rede virtual são mapeados em nós distintos da rede física. Havendo recursos suficientes, os nós físicos podem acomodar vários nós de redes virtuais diferentes. Os enlaces da rede virtual também devem ser alocados na rede física. É comum que um enlace virtual consista de um caminho que passa por vários nós na rede física [1].

A Figura 1.1 mostra duas possíveis redes virtuais heterogêneas VN1 e VN2, criadas pelos provedores de serviços SP1 e SP2, respectivamente. As linhas tracejadas indicam o mapeamento dos nós virtuais nos nós físicos do provedor de infraestrutura InP1. Ainda na Figura 1.1, alguns enlaces virtuais são exibidos com cores distintas de preto. Essas cores determinam determinam o mapeamento de um enlace virtual para o caminhos de mesma cor no substrato.

A Figura 1.1 mostra que o SP1 e o SP2 criaram as redes virtuais VN1 e VN2 sobre os recursos físicos gerenciados por dois diferentes provedores de infraestrutura, InP1 e InP2. O SP1 provê serviços fim a fim para os grupos de usuários finais U1 e U4, enquanto o SP2 conecta os grupos de usuários finais U2 e U3. Os usuários finais pagam aos Provedores

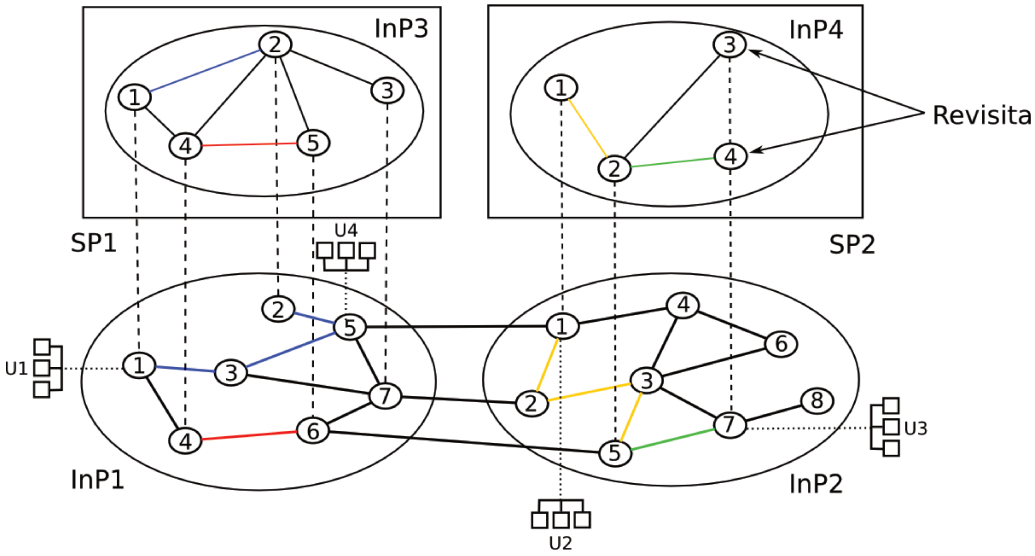


Figura 1.1: Arquitetura da virtualização de redes

de Serviços para utilizarem os recursos reservados no substrato. Um terceiro Provedor de Serços, SP3 por exemplo, poderia implantar uma rede virtual VN3 alocando recursos do provedor de infraestrutura InP3, que alocou anteriormente recursos do InP1. Nesse caso a rede VN3 seria criada como uma rede virtual filha do provedor de serviços SP1, num processo conhecido como recursão. Há ainda o conceito de revisita, onde dois ou mais nós de uma mesma rede virtual são mapeados para o mesmo nó físico. A aplicação desse conceito tem sido evitada na literatura devido às questões relacionadas à segurança e garantia de serviço. Isto é, caso três nós de uma rede virtual *vn* fossem mapeados para o nó *a* de uma rede física, em caso de falha no nó *a*, a rede virtual *vn* perderia três nós de uma única vez, dificultando procedimentos de recuperação. A revisita é ilustrada nos nós 3 e 4 da rede virtual VN2, pois ambos foram alocados para o nó 7 do InP2.

Redes sobrepostas têm sido usadas para implantar novas características na Internet. Recentemente, muitos projetos que lidam com virtualização de redes foram propostos. Eles tratam de diversas questões, tais como: garantia de desempenho [6] e disponibilidade [7] de roteamento[8], promovendo qualidade de serviço (QoS), proteção contra ataques de negação de serviço [9, 10], entre outras.

O problema de mapear os elementos virtuais em elementos físicos é conhecido como o Problema do Mapeamento de Redes Virtuais (*Virtual Network Embedding Problem* - VNEP). Nesse problema, tem-se uma rede física e várias requisições de redes virtuais, como ilustrado na Figura 1.1. Nesse problema, tenta-se mapear cada nó virtual em um nó físico, e cada enlace virtual em um caminho no substrato físico. A função objetivo no Problema do Mapeamento é minimizar o número de enlaces físicos alocados para os

enlaces virtuais, diminuindo o uso da banda passante. Os nós virtuais podem possuir requisitos de processamento, memória, entre outros. Para que haja um mapeamento de um nó virtual em um nó físico, é necessário que este possua os recursos requisitados. Por outro lado, os requisitos de enlaces virtuais, podem exigir largura de banda mínima, atraso máximo, entre outros. Tais requisitos devem ser atendidos em todos os enlaces físicos que compõem um enlace virtual em questão.

Caso existam recursos disponíveis na rede física para a alocação de todos os elementos de uma rede virtual, esta requisição será aceita, e consequentemente, deduz-se do substrato os recursos exigidos. Quando alocada, a rede virtual permanece por um tempo t, após o qual será desalocada e os recursos devolvidos para a rede física. Caso não existam recursos suficientes disponíveis para atender todos os elementos virtuais da requisição, esta é rejeitada.

Além do requisito de processamento, os nós podem possuir também requisito de localização. Em geral, nós físicos possuem localização física determinada por coordenadas (x_f, y_f). Os provedores de serviço que requisitam a criação de redes virtuais também podem definir regiões físicas preferenciais para alocação de cada nó virtual. Essas regiões preferenciais são determinadas por círculos com centro (x_v, y_v) e raio L. Assim, dado um nó virtual qualquer, os nós físicos que estiverem a uma distância menor ou igual a L do centro (x_v, y_v), que determina a área preferencial de alocação do nó virtual em questão, são considerados como candidatos para recebê-lo. A Figura 1.2 ilustra o requisito de localização.

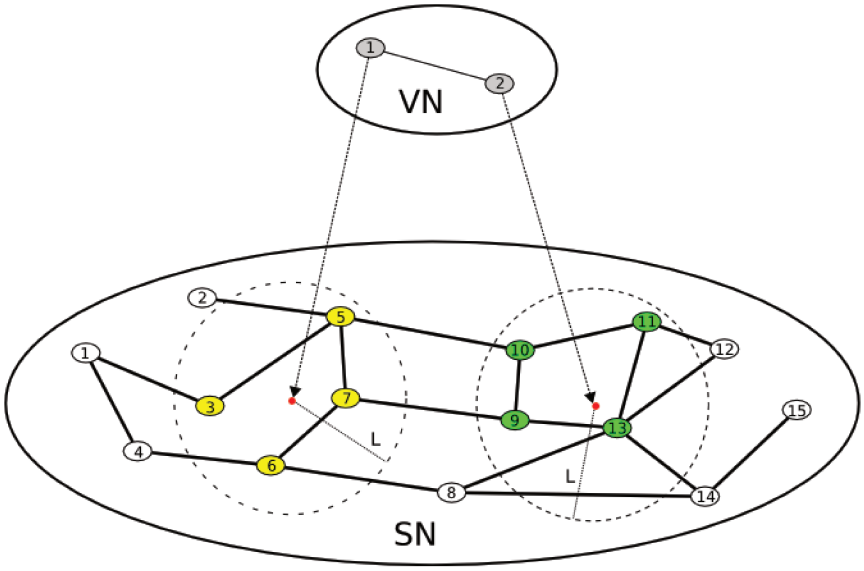


Figura 1.2: Restrição de localização: Candidatos a receber o nó virtual 1 em amarelo e candidatos a receber o nó virtual 2 em verde.

A determinação de regiões específicas para mapeamento de nós pode reduzir o atraso de entrega de pacotes entre dois ou mais nós virtuais. A existência de restrição de localização é determinante na escolha da estratégia de alocação de nós e tem grande impacto em métricas como taxa de bloqueio de requisições e custo do mapeamento. Se um provedor de serviço não determina restrição de localização, então todos os nós físicos são candidatos para todos os nós virtuais.

O Problema do Mapeamento de Redes Virtuais tem sido muito abordado na literatura por se tratar de um problema chave em Virtualização de Redes. Além disso, encontrar um mapeamento de custo mínimo é um problema NP-Difícil, mesmo quando os mapeamentos de nós virtuais são conhecidos previamente, ante a redução do problema NP-Difícil conhecido como *Unsplitable Flow Problem* para o Problema do Mapeamento de Redes Virtuais [11, 12]. A presente dissertação estuda o Problema do Mapeamento de Redes Virtuais. Entre as principais contribuições deste trabalho está a proposta de dois algoritmos para encontrar mapeamentos, baseados na metaheurística Busca Tabu, nomeados VNE-TS e VNE-TS-*Clustering*, apresentados no Capítulo 4.

Trabalhos anteriores, discutidos no Capítulo 2, mostraram que agrupar um conjunto de requisições de redes virtuais, durante uma fatia de tempo tolerável pelos SPs, é uma estratégia mais vantajosa do que a alocação imediata após a chegada de cada requisição. Essas requisições permanecem agrupadas durante uma janela de tempo w. Ao fim dessa janela, é aplicado um algoritmo de ordenação com base no lucro esperado pelo atendimento da requisição, com o objetivo de selecionar redes virtuais com maior custo-benefício. Assim, em adição à complexidade inerente ao problema de encontrar mapeamentos de custo mínimo para cada requisição de rede virtual, ainda se deve lidar com o problema de selecionar requisições lucrativas em caso de insuficiência de recursos para o atendimento de todas as redes virtuais agrupadas. A Figura 1.3 ilustra o uso de janelas para agrupamento de requisições.

A Figura 1.3 mostra que na primeira janela de tempo chegam quatro requisições de rede virtual. Ao fim da primeira janela de tempo, as quatro requisições são ordenadas de acordo com alguma métrica. Logo após, um algoritmo de alocação tenta mapear as requisições na ordem definida pelo algoritmo de ordenação. No exemplo da Figura 1.3, somente três requisições puderam ser atendidas. Uma das requisições foi rejeitada por insuficiência de recursos ou qualquer outro motivo conveniente para o provedor de infraestrutura. Outra importante contribuição deste trabalho é a proposta de um novo algoritmo de ordenação baseado no Problema da Mochila, nomeado 2ks-VN-*Selector*, apresentado em detalhes no Capítulo 4.

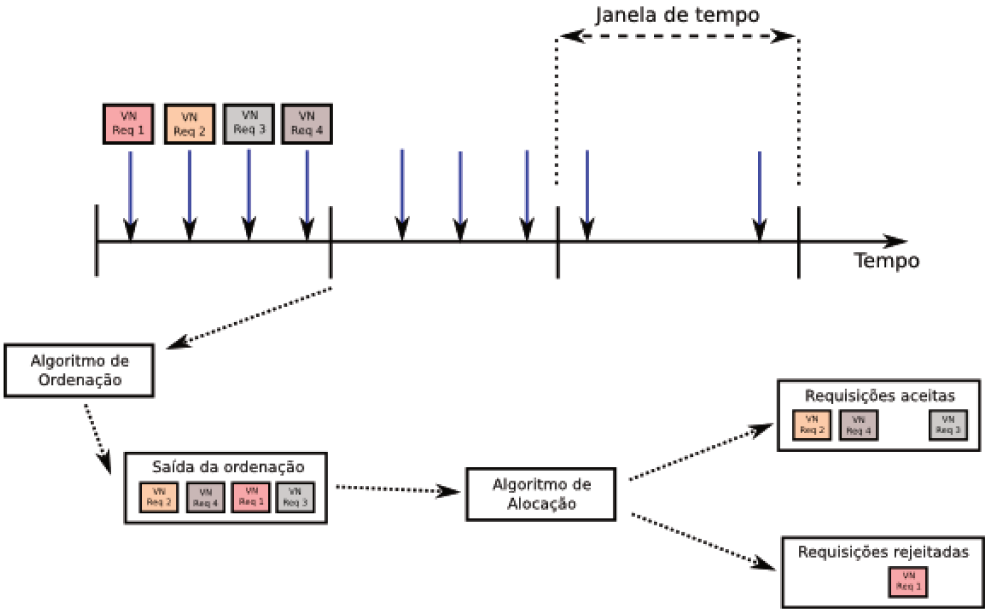


Figura 1.3: Processo de agrupamento, ordenação e alocação de redes virtuais.

1.2 Algoritmos

Esta seção oferece uma visão geral dos algoritmos e técnicas empregadas neste trabalho, a saber Busca Tabu, *Path Relinking*, *Kmeans* e Nuvem de Partículas.

1.2.1 Metaheurísticas

A otimização estocástica é uma classe de algoritmos e técnicas que aplica algum grau de aleatoriedade para encontrar soluções tão boas quanto possível para problemas difíceis. Metaheurísticas são os tipos mais genéricos desses algoritmos, e são aplicadas a uma larga variedade de problemas [13], consistindo muitas vezes em técnicas de pesquisa iterativas inspiradas em fenômenos físicos e biológicos.

As metaheurísticas têm se mostrado úteis na resolução de problemas NP-Difíceis. Exemplos de sucesso são Algoritmos Genéticos [14], Simulated Annealing [15], Colônia de Formigas [16], Nuvem de Partículas [17], Busca Tabu [18], entre outras.

Busca Tabu (*Tabu Search*)

Busca Tabu [18] é um método de otimização que toma uma solução em potencial para algum problema e pesquisa sua vizinhança imediata, definida por um operador de busca local que aplica mudanças sutis à solução corrente. Essa metaheurística mantém um histórico atributos das soluções recém consideradas, conhecido como lista tabu, e proíbe o

retorno às soluções dessa lista, até que tenha se passado tempo suficiente desde a última visita [13].

A mais simples abordagem por Busca Tabu consiste em manter uma lista tabu L, com algum tamanho máximo l, de soluções candidatas visitadas. Ao se encontrar uma solução, ela é enviada para a lista tabu. Se a lista tabu estiver cheia, as soluções mais antigas são removidas e passam a não ser mais tabu. Outras versões da Busca Tabu não consideram tamanho para a lista tabu, permitindo que um número indefinido de soluções permaneçam proibidas durante certo tempo. Esta abordagem de lista tabu ilimitada é utilizada nos algoritmos propostos neste trabalho.

Luke [13] sugere o *template* para Busca Tabu exibido no Algoritmo 1.

Algoritmo 1 *Busca Tabu*

```
1: l := Tamanho máximo da lista tabu
2: n := Número de vizinhos explorados
3: S := Solução inicial randômica
4: Best := S
5: L := {}
6: Enfileire S em L
7: repita
8:    se Tamanho(L) > l então
9:       Remova o elemento mais antigo em L
10:   fim se
11:   R := Vizinho(S)
12:   para n − 1 vezes faça
13:      W := Vizinho(S)
14:      se W ∉ L e (Qualidade(W) > Qualidade(R) ou R ∈ L) então
15:         R := W
16:      fim se
17:   fim para
18:   se R ∉ L e Qualidade(R) > Qualidade(S) então
19:      S := R
20:      Enfileire R em L
21:   fim se
22:   se Qualidade(S) > Qualidade(Best) então
23:      Best := S
24:   fim se
25: até algum critério de parada
26: retorne Best
```

O template da metaheurística Busca Tabu apresentado no Algoritmo 1 enfrenta problemas quando o espaço de busca é muito grande, e particularmente se o problema tem

grandes dimensões. Nesses casos é comum que essa versão permaneça ao redor do mesmo vizinho, um ótimo local, mesmo quando o tamanho da lista tabu for grande. Uma abordagem alternativa é criar uma lista tabu não de soluções candidatas encontradas, mas de mudanças ocorridas em certas características de uma solução [13]. Esta última abordagem é utilizada neste trabalho.

Ao passar de quinze anos, surgiram centenas de trabalhos na literatura, apresentando aplicações para a Busca Tabu. Em muitos casos, tais métodos fornecem soluções muito próximas da otimalidade, encontrando-se entre os mais efetivos. Este sucesso tornou a Busca Tabu extremamente popular entre pesquisadores interessados em encontrar boas soluções para problemas combinatórios [19].

Na presente dissertação aplica-se a Busca Tabu ao Problema do Mapeamento de Redes Virtuais, em uma abordagem original. Propõe-se dois algoritmos de mapeamento, o algoritmo VNE-TS e o VNE-TS-*Clustering*, baseados nessa metaheurística. Esses algoritmos visam encontrar bons mapeamentos sem realizar pesquisa exaustiva no espaço de busca do problema e serão detalhados no Capítulo 4.

Nuvem de Partículas (*Particle Swarm - PSO*)

Nuvem de Partículas [17] é um algoritmo de otimização estocástica populacional que pode gerar boas soluções com menos tempo de computação, quando comparado a outros métodos [20]. O pequeno número de parâmetros ajustáveis o torna fácil de implementar.

Metaheurísticas populacionais diferem de outros métodos não populacionais por manter um conjunto de solução candidatas (população) ao invés de uma única solução. Em geral, as várias soluções diferem entre si e possuem qualidade distinta. Cada solução candidata afeta como as outras irão mudar ao longo da execução do algoritmo.

Em contraste a outros métodos baseados em população, PSO não gera novos indivíduos na população. Em PSO, o conjunto de indivíduos se mantém do mesmo tamanho. Os indivíduos sofrem mutações que os direcionam para novas regiões do espaço de busca [13].

Segundo Luke [13], cada partícula (indivíduo da população) consiste de duas partes:

- Um vetor localização da partícula, $\vec{x} = (x_1, x_2, ..., x_n)$, equivalente ao genótipo de indivíduos, em algoritmos evolucionários.
- Um vetor velocidade da partícula, $\vec{v} = (v_1, v_2, ..., v_n)$, que determina a direção e velocidade que a partícula viaja a cada iteração. Isto é, se $\vec{x}(t-1)$ e $\vec{x}(t)$ são posições de uma partícula no espaço de busca nos tempos $t-1$ e t, respectivamente, então no tempo t, $\vec{v} = \vec{x}(t) - \vec{x}(t-1)$.

Cada partícula representa uma solução para o problema a ser tratado. A qualidade da partícula é determinada por uma função objetivo, que é dependente do problema.

Em PSO, cada partícula começa com vetores posição e velocidade aleatórios. Em cada iteração do algoritmo, as seguintes operações são realizadas:

- Verifica-se a qualidade de cada partícula e mantém-se o valor da melhor partícula;
- Atualiza-se aleatoriamente o vetor velocidade de cada partícula com base em um conjunto de melhores soluções;
- Atualiza-se a posição de cada partícula com base nos novos vetores velocidade.

O Algoritmo 2 ilustra o template para a metaheurística Nuvem de Partículas, sugerido por Luke [13]. Observa-se que, não obstante o pequeno número de parâmetros, algoritmos baseados na metaheurística Nuvem de Partículas ainda são mais difíceis de implementar do que aqueles baseados em Busca Tabu, dada sua maior simplicidade e quantidade de parâmetros ainda menor.

Zhang et al. [21] propuseram um algoritmo baseado em Nuvem de Partículas para o problema da alocação de redes virtuais, o qual obteve um dos melhores resultados até então. Esse algoritmo foi implementado e os resultados obtidos por ele foram comparados com os obtidos pelos algoritmos propostos nesta dissertação. Os resultados são discutidos no Capítulo 5.

1.2.2 *Path Relinking*

No método *Path Relinking*, são geradas novas soluções a partir de duas soluções referência S^1 e S^2, com conjuntos de características $F^1 = \{f_1^1, f_2^1, ..., f_n^1\}$ e $F^2 = \{f_1^2, f_2^2, ..., f_n^2\}$. A partir da solução S^1, são geradas n soluções intermediárias, adicionando características de S^2 à S^1. Por exemplo, uma solução intermediária S' com o conjunto de características F' considerando F^1 e F^2 seria $F' = \{f_1^2, f_2^1, ..., f_n^1\}$.

Do ponto de vista espacial, o processo de gerar combinações de duas soluções referência, F^1 e F^2, pode ser caracterizado como a geração de caminhos entre soluções. Caminhos entre F^1 e F^2 contém soluções que compartilham um subconjunto significativo de atributos de ambas as soluçõs referência. A Figura 1.4 ilustra o processo do *Path Relinking*.

A Figura 1.4 ilustra o caminho percorrido por uma heurística, que inicia com uma solução S^1. A partir dessa solução, a cada iteração, a heurística salta para uma nova solução, geralmente através de busca local, seguindo o caminho definido pela linha sólida, até alcançar a solução final S^2. De posse das soluções S^1 e S^2, o *Path Relinking* geralmente segue um caminho mais curto, partindo da solução S^1 até chegar à solução S^2, saltando de uma solução para a próxima ao acrescentar em S^1 uma nova característica de S^2. Esse novo caminho é determinado pela linha tracejada na Figura 1.4. Se houver uma

Algoritmo 2 *Nuvem de Partículas*

1: $qtd\_particulas$:= Número desejado de partículas
2: α := Proporção da velocidade a ser mantida
3: β := Proporção da melhor posição alcançada pela partícula a ser mantida
4: γ := Proporção da melhor partícula informante a ser mantida
5: δ := Proporção da melhor partícula global a ser mantida
6: ε := Tamanho do salto de uma partícula
7: $P := \{\}$
8: **para** $qtd\_particulas$ vezes **faça**
9: $P := P \cup \{$nova partícula com posição e velocidades aleatórias$\}$
10: **fim para**
11: $Best$:= ♣ //(♣ é a melhor solução em P)
12: **repita**
13: **para cada** partícula $x \in P$ com velocidade v **faça**
14: **se** $Best$ = ♣ ou qualidade(x) > qualidade($Best$) **então**
15: $Best := x$
16: **fim se**
17: **fim para**
18: **para cada** partícula $x \in P$ com velocidade v **faça**
19: x^* := a melhor posição que x já obteve
20: x^+ := a melhor posição já obtida pelas partículas informantes de x
21: $x^!$:= a melhor posição já obtida dentre todas as partículas
22: **para cada** dimensão i **faça**
23: b := número randômico entre 0.0 e β inclusive
24: c := número randômico entre 0.0 e γ inclusive
25: d := número randômico entre 0.0 e δ inclusive
26: $v_i := \alpha v_i + b(x_i^* - x_i) + c(x_i^+ - x_i) + d(x_i^! - x_i)$
27: **fim para**
28: **fim para**
29: **para cada** partícula $x \in P$ com velocidade v **faça**
30: $x := x + \varepsilon v$
31: **fim para**
32: **até** até algum critério de parada
33: **retorne** $Best$

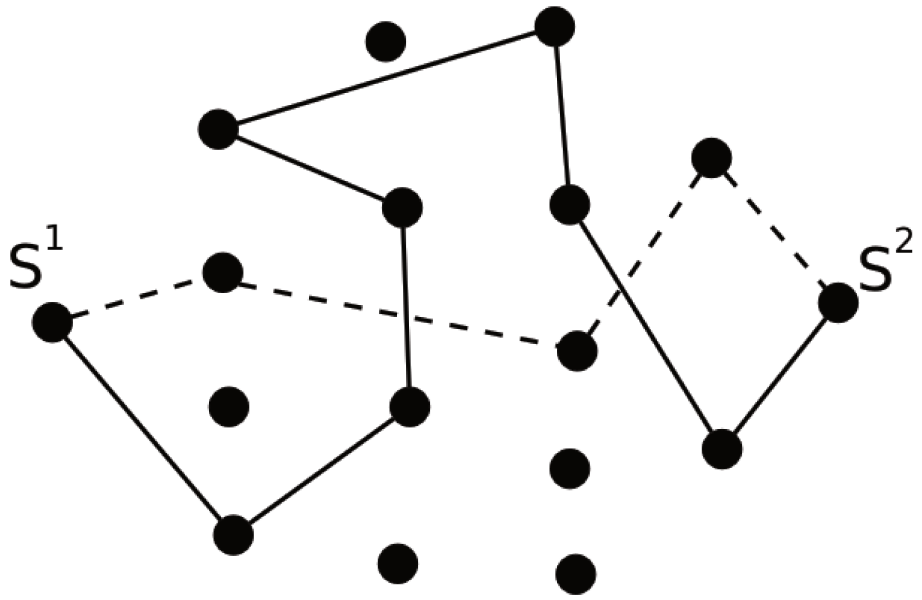


Figura 1.4: *Path Relinking*: Caminho original mostrado pela linha sólida e um possível caminho de ligação mostrado pela linha tracejada.

solução com melhor custo na linha tracejada, o *Path Relinking* seleciona essa solução. O Algoritmo 3 ilustra o template para o *Path Relinking*.

A estratégia *Path Relinking* é uma estratégia de intensificação e é utilizada como meio de melhorar as soluções obtidas em heurísticas propostas na presente dissertação. Sua aplicação é discutida no Capítulo 4.

Algoritmo 3 *Path Relinking*

Entrada: S_1: **Solução**, S_2: **Solução**
1: $Best$:= Melhor(S_1,S_2)
2: $S := S_1$
3: **para** i de 1 até $|S_2|$ **faça**
4: $S[i] := S_2[i]$
5: **se** S é melhor do que $Best$ **então**
6: $Best := S$
7: **fim se**
8: **fim para**
9: **retorne** $Best$

1.2.3 *Kmeans*

Técnicas de agrupamento (*clustering*) exploram semelhanças entre padrões de elementos, agrupando-os segundo critérios de similaridade. O agrupamento de elementos tem larga aplicação em análise e mineração de dados para ajudar na tomada de decisões de negócios.

O problema da clusterização é NP-Difícil. No entanto, existem algoritmos heurísticos eficientes que são comumente empregados e convergem rapidamente para um ótimo local. Um desses algoritmos é o *Kmeans*.

Kmeans é uma técnica de clusterização cujo objetivo é distribuir n elementos dentre k *clusters*, de modo que cada elemento pertença ao *cluster* mais próximo. A medida de proximidade é dependente do problema. Exemplos de medida de proximidade são a distância euclidiana e a distância em termos de saltos em roteadores de uma rede. Esse algoritmo começa escolhendo aleatoriamente k elementos para representar centróides, $c_1, ..., c_k$, dos *clusters* $C_1, ..., C_k$. Todos os n elementos são colocados em determinado *cluster* com base na distância entre o elemento e o seu centróide. Se a distância entre c_i e o elemento e_j é a menor dentre todos os outros centróides, então o elemento e_j é colocado no *cluster* C_i. Quando todos os registros tiverem sido colocados inicialmente em um *cluster*, o centróide de cada *cluster* é recalculado com base em seus respectivos elementos. Depois o processo se repete, examinando cada elemento novamente, colocando-o no *cluster* cujo centróide está mais próximo. Várias iterações podem ser necessárias para que o algoritmo convirja para um ótimo local [22]. Um ótimo local é uma solução com custo baixo, mas que nem sempre é a solução de menor custo global. O Algoritmo 4 exibe o template para o algoritmo de agrupamento *Kmeans*.

Algoritmo 4 *Kmeans*

1: escolha aleatoriamente k elementos como os centróides para os k *clusters*;
2: **repita**
3: atribua cada elemento e a um *cluster* de modo que a distância entre e e o centróide do *cluster* é a menor dentre os k *clusters*;
4: recalcule o centróide de cada *cluster* c, com base nos elementos atribuídos à c
5: **até** que nenhuma mudança ocorra
6: **retorne** os k clusters

Nesta dissertação, emprega-se o *Kmeans* no algoritmo VNE-TS-*Clustering* para definir regiões da rede física que possivelmente possam atender a todos os nós virtuais de uma requisição. Com isso, espera-se diminuir a distância entre os nós e, consequentemente, o número de enlaces utilizados no mapeamento. Essa abordagem é discutida com mais detalhes no Capítulo 4.

1.3 Organização da dissertação

As demais partes desta dissertação estão organizadas da seguinte maneira: No Capítulo 2 realiza-se um levantamento dos principais trabalhos que abordam o Problema do Mapeamento de Redes Virtuais em Redes Físicas. No Capítulo 3, descreve-se a formulação matemática do problema, além de mencionar métricas para avaliação dos algoritmos de mapeamento, propostas em trabalhos anteriores. Os algoritmos de mapeamento propostos, VNE-TS e VNE-TS-*Clustering*, são apresentados no Capítulo 4, bem como o algoritmo de seleção de requisições VN-*Selector*. No Capítulo 5 discute-se os resultados obtidos, e comparam-nos aos resultados do algoritmo VNE-PSO, um algoritmo de mapeamento baseado em *Particle Swarm*, proposto na literatura. Também compara-se a estratégia de seleção, 2ks-VN-*Selector*, com a estratégia *Most Prize First*, proposta em trabalhos anteriores. Por fim, apresentam-se as conclusões no Capítulo 6.

Capítulo 2

Trabalhos Relacionados

Este capítulo apresenta os trabalhos relevantes publicados na literatura, que lidam com o Problema do Mapeamento de Redes Virtuais.

Zhu e Ammar [23] estudaram o Problema do Mapeamento sob uma ótica de recursos ilimitados. Isto é, nós e enlaces físicos podem comportar qualquer número de nós ou enlaces virtuais, respectivamente. O objetivo era minimizar o *stress* máximo em elementos físicos. Os autores definiram o *stress* em um nó físico n_i^S como a quantidade de nós virtuais alocados em n_i^S. De maneira semelhante, o *stress* em um enlace físico e_{ij}^S é dado pelo número de enlaces virtuais que fazem uso de e_{ij}^S. Os autores propuseram várias heurísticas e as compararam entre si. Propuseram ainda um algoritmo de reconfiguração de alocações prévias, que visa atenuar o *stress* em regiões super utilizadas. Os resultados obtidos mostraram que a minimização do *stress* máximo leva a uma utilização uniforme dos recursos físicos, evitando gargalos em regiões pontuais do substrato. Mostraram ainda que o algoritmo de reconfiguração é capaz de realizar uma melhor distribuição dos recursos físicos, evitando gargalos. Apesar disso, o trabalho assume a hipótese irrealista da existência de recursos ilimitados em elementos físicos e, portanto, não analisa métricas importantes como a taxa de bloqueio às requisições de redes virtuais.

Yu et al. [24] abordaram o Problema do Mapeamento levando em consideração a limitação de recursos no substrato. Os autores consideraram a possibilidade de realizar ou não *path splitting* na rede física, isto é, um fluxo emergente de um nó físico pode ser dividido e seguir por vários enlaces físicos distintos. O algoritmo proposto pelos autores coleta um grupo de requisições em uma janela de tempo w e tenta então alocá-las no substrato, em ordem decrescente de rendimento. Os autores definiram o rendimento de uma rede virtual como o somatório dos recursos por ela requisitados. O mapeamento é realizado em duas etapas, primeiro os nós depois os enlaces virtuais. Há dois algoritmos de mapeamento de nós, a saber, um guloso que mapeia nós virtuais com maior demanda de processamento para nós físicos com maior oferta, e outro que considera a topologia

da rede virtual para realizar mapeamentos customizados mais eficientes. O mapeamento dos enlaces virtuais é então realizado através do algoritmo *k-shortest paths*, quando não se considera *path splitting*, ou resolvendo o *multicommodity flow problem*, no caso que considera *path splitting*. Os autores também consideraram o desbalanceamento no uso dos recursos físicos e propuseram um algoritmo de migração de caminhos. Os resultados mostraram que o algoritmo do *path splitting* e o algoritmo de migração são capazes de aumentar o rendimento e diminuir o custo de mapeamento. Contudo, apesar de *path splitting* prover balanceamento no uso dos recursos em enlaces virtuais, deve-se assumir que o substrato oferece suporte para este mecanismo.

Chowdhury et al. [25] propuseram um algoritmo de mapeamento que tenta coordenar o mapeamento de nós e enlaces virtuais. Os autores formularam o problema como um problema de programação linear inteira mista, aumentando o substrato através da inserção de meta-nós. As restrições inteiras são então relaxadas obtendo-se um problema de programação linear. O valor inteiro das variáveis envolvidas no problema é escolhido através de dois algoritmos D-ViNE e R-ViNE, que escolhem o valor de maneira determinística ou aleatória, respectivamente. Em um cenário que considera recursos limitados na rede física e restrições de localização nos nós virtuais, as simulações mostraram que o algoritmo proposto aumenta a taxa de aceitação e o rendimento, ao mesmo tempo em que reduz o custo no substrato. O rendimento e o custo são considerados como o total de recursos exigidos pela rede virtual e o total de recursos alocados no substrato, respectivamente. Os autores realizam o mapeamento dos enlaces virtuais através do *multicommodity flow problem*, assumindo que a rede física suporta *path splitting*.

Fajjari et al. [26] basearam-se na metaheurística Colônia de Formigas Max-Min [27] para construir um algoritmo de mapeamento de redes virtuais. A ideia principal por trás dessa metaheurística é a inspiração no comportamento coletivo de formigas ao procurar o melhor caminho entre o ninho e a fonte de comida. Os autores compararam os resultados obtidos com algoritmos propostos anteriormente. Tais resultados mostraram que o algoritmo proposto, o VNE-AC, supera os demais, obtendo melhor minimização da taxa de rejeição e maximização no rendimento. Contudo, o cenário considerado nas simulações faz uso de uma rede física com 100 nós e 0.5 de probabilidade de conexão entre pares de nós. Essa rede física possui em média n*(n-1)/4 enlaces físicos, onde n é o número de nós físicos. Assim, foi considerada uma rede física com 2475 enlaces, em média. Redes físicas densas como a considerada não se assemelham à redes reais. Limitações nos recursos em nós e enlaces físicos são consideradas, além de restrição de localização.

Masti e Raghavan [28] desenvolveram o algoritmo VNA, que realiza o mapeamento dos nós virtuais levando em conta os recursos residuais em enlaces físicos, reduzindo a chance de falha durante o mapeamento dos enlaces. Antes do mapeamento dos nós físicos, cria-se um grafo G'_S que consiste na rede física com todos os nós, mas com apenas enlaces físicos

com recursos suficientes para atender qualquer um dos enlaces virtuais. Como resultado, pode haver vários componentes desconexos em G'_S. O algoritmo proposto tenta então mapear todos os nós em um único componente de G'_S, para depois realizar o mapeamento dos enlaces virtuais. Os autores não restringiram o mapeamento dos enlaces virtuais a apenas um caminho na rede física, ao levar em consideração o mecanismo de *path splitting*. O mapeamento dos enlaces virtuais é realizado através do algoritmo de fluxo máximo. O fluxo máximo obtido identifica um conjunto de caminhos pouco utilizados, nos quais o enlace virtual pode ser mapeado, balanceando o uso dos recursos no substrato. Em um cenário com recursos limitados, os resultados apresentados mostraram que o VNA alcança maiores taxas de aceitação quando comparado a trabalhos anteriores. Contudo, o algoritmo proposto não funciona em cenários onde há restrição de localização em nós virtuais, visto que esta restrição pode impedir o mapeamento de todos os nós virtuais para um mesmo componente de G'_S.

Um outro algoritmo baseado em metaheurística, o VNE-PSO, foi proposto por Zhang et al. em [21]. Em um cenário com recursos limitados e restrição de localização nos nós virtuais, os autores tentaram maximizar a taxa de aceitação e o rendimento, minimizando o custo em provedores de infraestrutura. Os autores utilizaram a metaheurística *Particle Swarm* [17] para realizar o mapeamento dos nós virtuais. Os autores propuseram duas maneiras de mapear os enlaces virtuais. Na primeira delas, o mapeamento é realizado através do algoritmo de menor caminho, quando *path splitting* não é suportado. No segundo caso, considerando-se suporte à *path splitting*, o mapeamento é feito através de um novo algoritmo guloso proposto, o *greedy k-shortest paths*. O VNE-PSO foi comparado com algoritmos existentes na literatura e os experimentos mostraram que ele obtém melhores resultados quanto ao rendimento médio a longo prazo, taxa de aceitação e custo de mapeamento. Os nossos algoritmos propostos, VNE-TS e VNE-TS-*Clustering*, são comparados contra os resultados obtidos pelo VNE-PSO.

Alkmim et al. [29] empregaram algoritmos baseados em programação linear inteira para resolver o Problema do Mapeamento, sem assumir recursos ilimitados nem topologias específicas. Os autores consideraram a presença de repositórios de imagens contendo o *software* e os protocolos necessários pela Rede Virtual. Essas imagens devem ser transferidas do repositório para o roteador real antes de a rede virtual entrar em operação. Os algoritmos propostos selecionam as imagens e o caminho que a transferência da imagem deve tomar, minimizando o total de banda alocada. Os autores compararam os resultados de seis algoritmos diferentes.

Um ponto negligenciado pela maioria dos trabalhos citados é a realização de estudos considerando a ordem de atendimento das requisições. Até onde se tem conhecimento, não foram realizados quaisquer estudos que analisem os efeitos de algoritmos de ordenação e seleção de requisições, em métricas como taxa de aceitação e rendimento. Vários trabalhos

anteriores lidam com o problema do mapeamento apenas na fase de alocação. Isto é, imediatamente após o recebimento de uma requisição, tenta-se alocá-la através do uso de algum algoritmo de mapeamento. Essa abordagem tende a mapear redes virtuais cuja relação de custo-benefício não é a mais vantajosa a longo prazo. Alguns dos trabalhos citados nesta seção agrupam várias requisições em um espaço de tempo w, e tentam alocá-las em ordem não crescente de rendimento, sem considerar o custo decorrente de sua alocação. Uma das principais contribuições desta dissertação é a proposta de um novo algoritmo de seleção/ordenação de redes virtuais agrupadas em uma janela de tempo w, que baseia-se no Problema da Mochila Bidimensional para realizar seleções que ofereçam melhor custo-benefício. Além do algoritmo de seleção, propõe-se também dois novos algoritmos de mapeamento baseados na metaheurística Busca Tabu, VNE-TS e VNE-TS-*Clustering*, apresentados no capítulo 4.

Em resumo, os trabalhos citados possuem pelo menos uma das seguintes deficiências:

- Consideração de cenários irreais, com recursos ilimitados ou redes físicas muito densas.
- Consideração do uso de *path splitting* durante o mapeamento de enlaces, que requer o emprego de algoritmos sofisticados de roteamento, o que aumenta o custo operacional [26].
- Negligência de restrições de localização.
- Consideração de somente o rendimento de redes virtuais e não o custo de mapeamento.
- Consideração de somente uma taxa de chegada, negligenciando-se os efeitos de frequências intensas de chegadas de requisições.

Capítulo 3

Modelo de Rede e Formulação Matemática do Problema

Neste capítulo, apresentam-se os modelos de representação da rede física, mantida pelo provedor de infraestrutura, e das redes virtuais requisitadas por provedores de serviço. Apresentam-se, ainda, a formulação do Problema do Mapeamento de Redes Virtuais e finaliza-se com a definição de algumas métricas relacionadas ao problema, comuns na literatura.

3.1 Rede Física

Assim como nos trabalhos citados no Capítulo 2, a rede física é modelada como um grafo não direcionado ponderado denotado por $G^S = (N^S, E^S)$, onde N^S e E^S representam os conjuntos de nós e enlaces físicos, respectivamente. Cada nó físico $n_i^S \in N^S$ possui capacidade de processamento $p(n_i^S)$ e localização física $loc(n_i^S)$. Cada enlace físico $e_{ij}^S = (n_i^S, n_j^S) \in E^S$, onde $n_i^S, n_j^S \in N^S$, com $i \neq j$, possui largura de banda $b(e_{ij}^S)$. A capacidade residual nos elementos físicos (nós e enlaces) é a quantidade de recursos disponíveis e não reservados por qualquer elemento virtual. O processamento residual em um nó físico n_i^S é denotado por $\mathcal{R}(n_i^S)$ e a largura de banda residual é denotada por $\mathcal{R}(e_{ij}^S)$. Seja P o conjunto de todos os caminhos em G^S. O conjunto $P_{n_i^S, n_j^S} \subseteq P$ contém todos os caminhos do substrato entre os nós físicos n_i^S e n_j^S em G^S. A capacidade de um caminho $p \in P_{n_i^S, n_j^S}$ é definida como a capacidade residual mínima ao longo do caminho: $\mathcal{R}(p) = min_{e_{ij}^S \in p} \mathcal{R}(e_{ij}^S)$

3.2 Rede Virtual

De maneira similar, uma rede virtual é definida por um grafo ponderado, não direcionado, denotado por $G^V = (N^V, E^V)$, onde N^V é o conjunto de nós virtuais e E^V define o conjunto de enlaces virtuais. Cada nó virtual $n_i^V \in N^V$ possui capacidade de processamento requisitada $p(n_i^V)$ e pode possuir ou não requisito de localização $loc(n_i^V)$, determinado pelo provedor de serviço. Cada enlace virtual $e_{ij}^V = (n_i^V, n_j^V) \in E^V$, onde $n_i^V, n_j^V \in N^V$ e $i \neq j$, possui quantidade de banda passante requisitada $b(e_{ij}^V)$. Dependendo do cenário, cada rede virtual G^V pode possuir ou não um requisito de distância máxima para mapeamento de nós virtuais, definido por $L(G^V)$. Se existir, esse requisito determina que um nó virtual $n_i^V \in N^V$ poderá ser mapeado apenas para nós físicos $n_i^S \in N^S$ que satisfaçam $dist(loc(n_i^S), loc(n_i^V)) \leq L(G^V)$, onde $dist$ determina a distância euclidiana entre dois pontos. O conjunto de nós físicos que satisfazem a restrições de localização e processamento para um nó virtual n_i^V é denotado por $CAND(n_i^V)$.

3.3 Descrição do Problema do Mapeamento de Redes Virtuais

Dado um conjunto de redes virtuais $\{G_1^V, G_2^V, ..., G_n^V\}$ e uma rede física G^S, o objetivo da solução no Problema do Mapeamento de Redes Virtuais é mapear os elementos de cada rede virtual G_i^V nos elementos físicos do substrato representado por G^S, alocando recursos para os nós e enlaces virtuais. Os recursos alocados são reservados durante todo o tempo de vida da rede virtual, não podendo ser cedidos a outra requisição. Os recursos reservados são devolvidos apenas quando o tempo de vida da VN acaba. Como os recursos no substrato são limitados, faz-se necessário realizar mapeamentos econômicos, de modo a maximizar ou minimizar alguma função objetivo.

O processo de mapeamento consiste de duas etapas:

- **Mapeamento de nós**: Cada nó virtual é mapeado para um nó físico que possua capacidade residual suficiente para atender os requisitos das solicitações de estabelecimento de VNs. Além disso, a função de mapeamento de nós deve ser injetiva, não permitindo revisita. Considere $f_n : N^V \rightarrow N^S$ uma função de mapeamento, então

  - $\forall n_i^V \in N^V, f_n(n_i^V) = n_i^S$, onde $n_i^S \in N^S \ \wedge \ p(n_i^V) \leq \mathcal{R}(n_i^S)$ $\wedge \ dist(loc(n_i^V), loc(n_i^S)) \leq L(G^V)$
  - $\forall n_i^V, n_j^V \in N^V, f_n(n_i^V) = f_n(n_j^V) \iff n_i^V = n_j^V$

- **Mapeamento de enlaces**: Durante o mapeamento dos enlaces, cada enlace virtual e_{ij}^V é mapeado em um caminho entre os nos físicos determinados por $f_n(n_i^V)$ e $f_n(n_j^V)$. A banda residual em cada enlace físico do caminho no substrato deve ser maior ou igual ao valor de banda requisitada pelo enlace virtual. Seja $f_l : E^V \rightarrow P$ uma função que mapeia um enlace virtual $e_{ij}^S = (n_i^V, n_j^V)$ em um caminho na rede física iniciado em $f_n(n_i^V)$ e terminado em $f_n(n_j^V)$. Então

  – $\forall e_{ij}^V \in E^V, f_l(e_{ij}^V) = p \ \wedge \ b(e_{ij}^V) \leq \mathcal{R}(p)$, onde $p \in P_{f_n(n_i^V)f_n(n_j^V)}$

A maioria dos algoritmos propostos nos trabalhos listados no Capítulo 2 realizam as etapas sequencialmente, isto é, primeiro mapeiam os nós e logo após realizam o mapeamento dos enlaces. Dentre os trabalhos mencionados, apenas Chowdhury et al. [25] propuseram um algoritmo que faz os mapeamentos de nós e enlaces concorrentemente.

3.4 Métricas no Problema do Mapeamento de Redes Virtuais

Para avaliar o desempenho dos nossos algoritmos, consideram-se algumas das métricas utilizadas em [21]. São elas

- **Taxa de bloqueio de requisições**: Esta métrica é definida como

$$\mathcal{B}(T) = 1 - \frac{|VNACC(T)|}{|VNREQ(T)|}$$

  onde *VNACC(T)* define o conjunto de requisições de VNs atendidas até o tempo T, e *VNREQ(T)* define o conjunto de requisições recebidas até o tempo T;

- **Ganho médio a longo prazo**: Do ponto de vista do provedor de infraestrutura, um algoritmo de mapeamento eficiente maximizaria o seu ganho, aceitando mais requisições de VN, a longo prazo. O rendimento ou ganho por aceitar uma única requisição de rede virtual é definido como

$$\mathcal{G}(G^V) = \sum_{n_i^V \in N^V} p(n_i^V) + \sum_{e_{ij}^V \in E^V} b(e_{ij}^V)$$

  Assim, o ganho médio a longo prazo como consequência da alocação de todas as redes virtuais em *VNACC(T)* é definido como

$$\frac{\sum_{G^V \in VNACC(T)} \mathcal{G}(G^V)}{T}$$

- **Custo médio a longo prazo**: O custo de aceitar uma requisição G^V é definido como o somatório de todos os recursos físicos alocados para G^V, isto é

$$\mathcal{C}(G^V) = \sum_{n_i^V \in N^V} p(n_i^V) + \sum_{e_{ij}^V \in E^V} |f_l(e_{ij}^V)| \cdot b(e_{ij}^V)$$

- **Taxa de ganho sobre custo a longo prazo**: A taxa de ganho sobre custo a longo prazo quantifica a eficiência no uso dos recursos do substrato em um tempo T, e é definida como

$$\frac{\sum_{G^V \in VNACC(T)} \mathcal{G}(G^V)}{\sum_{G^V \in VNACC(T)} \mathcal{C}(G^V)}$$

Capítulo 4

Novos Algoritmos propostos para o Problema do Mapeamento de Redes Virtuais: VNE-TS, VNE-TS-*Clustering* e 2ks-VN-*Selector*

Este capítulo descreve os algoritmos de mapeamento *Virtual Network Embedding Tabu Search* (VNE-TS) e *Virtual Network Embedding Tabu Search Clustering* (VNE-TS-*Clustering*), propostos nesta dissertação. Descreve-se também o algoritmo de seleção de requisições *Two dimensional Knapsack Virtual Network Selector* (2ks-VN-*Selector*).

Antes de apresentar os algoritmos, a forma de representação de um mapeamento (ou solução) e o conceito de mapeamento viável são apresentados. Esses conceitos são essenciais para a reprodução dos algoritmos, bem como dos resultados obtidos. A estratégia de seleção de nós físicos chamada L2S2 e a função objetivo largamente utilizada em trabalhos anteriores são também discutidas.

4.1 Representação de mapeamentos

Nos algoritmos propostos, o mapeamento de uma VN é representado por um vetor de m posições, onde $m = |N^V|$. Na i-ésima posição do vetor é armazenado o id do nó físico para o qual o i-ésimo nó virtual é mapeado, conforme mostra a Figura 4.1.

A Figura 4.1 mostra o mapeamento de duas redes virtuais, VN1 e VN2, no mesmo substrato. As setas direcionais representam os mapeamentos dos nós virtuais aos nós

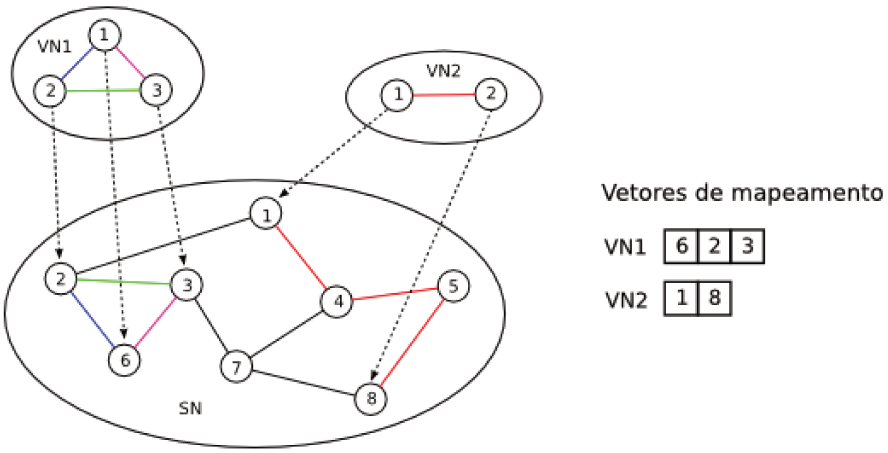


Figura 4.1: Representação de mapeamento.

físicos. As cores nos enlaces determinam os mapeamentos dos enlaces virtuais em caminhos na rede física. Por exemplo, o enlace virtual em vermelho na VN2 foi mapeado para o caminho em vermelho no substrato. O mapeamento da VN1 utiliza três enlaces físicos, assim como o mapeamento da VN2.

No caso específico da Figura 4.1, se o nó 2 da VN2 pudesse ser mapeado para o nó 4 ou 5 da rede física, então seria possível obter uma solução mais eficiente, pois o mapeamento da VN2 exibido, nessa hipótese, utilizaria mais enlaces físicos do que o necessário.

4.2 Viabilidade em mapeamentos

Conforme discutido na seção 3.3, o mapeamento de uma VN pode ser realizado em duas etapas: mapeamento de nós e mapeamento de enlaces. Os algoritmos de mapeamento propostos neste trabalho realizam mapeamento sequencial, isto é, primeiro mapeiam os nós e em seguida mapeiam os enlaces. Após a primeira etapa, tem-se um mapeamento de nós. Após a segunda etapa, tem-se um mapeamento de nós e enlaces.

Se não for possível obter um mapeamento de nós devido à falta de recursos nos nós físicos, então não se pode mapear os enlaces. Nesse caso, algoritmos que realizam mapeamento sequencial devem imediatamente rejeitar a requisição.

Caso seja possível um mapeamento de nós, então haverá ou não a possibilidade de gerar um mapeamento de enlaces. Um mapeamento de nós é dito viável se foi possível construir um caminho no substrato para todos os enlaces virtuais. Se a construção desses caminhos não for possível devido à falta de recursos no substrato, o mapeamento de nós é dito inviável. Nesse caso, algoritmos de mapeamento podem tentar encontrar um outro mapeamento de nós que possibilite o mapeamento dos enlaces.

Uma estratégia comum no mapeamento de um enlace virtual $e_{ij}^V = (n_i^V, n_j^V)$ é mapeá-

lo para o menor caminho, no substrato, entre os nós físicos $f_n(n_i^V)$ e $f_n(n_j^V)$, considerando apenas enlaces físicos com banda residual suficiente para atender os requisitos do enlace virtual e_{ij}^V. Nos algoritmos propostos, o mapeamento de enlaces é realizado desta maneira.

Não se considera *path splitting* durante o mapeamento de enlaces virtuais. A não utilização de *path splitting* torna o problema ainda mais difícil, visto que essa estratégia balanceia o uso da banda nos enlaces físicos, evitando gargalos [25]. Além do mais, ao usar *path splitting*, a banda residual não precisa atender toda a banda requisitada, uma vez que esta pode ser dividida entre vários enlaces.

4.3 Estratégia de seleção local L2S2

A estratégia de alocação de nós virtuais L2S2 (*large to large and small to small*) foi proposta em [21]. Este mecanismo evita sobrecargas pontuais no substrato, ao direcionar o mapeamento de um nó virtual n_i^V para nós físicos em $CAND(n_i^V)$ que possuam mais recursos disponíveis, sem completamente abrir mão da aleatoriedade.

Durante a fase de mapeamento de nós, a estratégia L2S2 leva em consideração a chance de se satisfazer as restrições de conectividade de nós virtuais na segunda etapa, mapeando nós virtuais para nós físicos com mais recursos de banda em enlaces adjacentes. Quanto mais largura de banda houver nos enlaces adjacentes a um nó físico, maior será a probabilidade de um nó virtual ser mapeado para ele. Esta probabilidade também aumenta proporcionalmente à quantidade residual de processamento no nó físico. A quantidade de recursos disponíveis em um nó físico n_i^S e em seus enlaces adjacentes é dada por

$$NR(n_i^S) = \mathcal{R}(n_i^S) \cdot \sum_{e_{ij}^S \in adj(n_i^S)} \mathcal{R}(e_{ij}^S)$$

A essência da estratégia L2S2 está na assertiva de que nós virtuais que demandam grande quantidade de recursos têm grande probabilidade de serem mapeados para nós físicos com grande quantidade de recursos disponíveis, aumentando a convergência de algoritmos que façam uso dessa estratégia. Além do mais, nós físicos com menores quantidades de recursos residuais se tornam menos prováveis de serem escolhidos, levando ao balanceamento no uso de recursos do substrato, evitando sobrecarga de uso em determinados pontos da rede física.

No L2S2, a probabilidade de alocação de um nó virtual n_i^V para um nó físico $n_i^S \in CAND(n_i^V)$, é dada por

$$\frac{NR(n_i^S)}{\sum_{u_i^S \in CAND_{(n_i^V)}} NR(u_i^S)}$$

4.4 Função objetivo dos algoritmos de mapeamento

A função objetivo dos algoritmos propostos neste trabalho é a mesma considerada em [21] e nos demais trabalhos relacionados listados no Capítulo 2. Ela é definida como

$$min \sum_{e_{ij}^V \in E^V} |f_l(e_{ij}^V)| \cdot b(e_{ij}^V)$$

Essa função objetivo não leva em consideração os recursos alocados para os nós virtuais porque, independentemente do mapeamento de nós escolhido, a quantidade de processamento reservado é sempre a mesma. Por outro lado, deseja-se minimizar a quantidade de banda passante alocada para a rede virtual, que pode variar dependendo do mapeamento dos nós e dos enlaces virtuais.

4.5 O algoritmo de mapeamento VNE-TS

O VNE-TS é um algoritmo de mapeamento que pode ser aplicado em ambientes com ou sem requisito de localização. O VNE-TS é apresentado no Algoritmo 5.

O algoritmo começa gerando um mapeamento aleatório de nós para a rede virtual G^V (Linha 1), isto é, gera um vetor cuja posição i é inicializada aleatoriamente com o identificador de algum nó físico $n_i^S \in CAND(n_i^V)$, através do procedimento *mapeamentoAleatórioComL2S2* (linha 1). Se não for possível construir um mapeamento inicial para os nós virtuais, então o algoritmo rejeita a requisição.

Se a função *mapeamentoAleatórioComL2S2* conseguir encontrar um mapeamento de nós, ela tenta ainda mapear os enlaces virtuais de G^V em caminhos físicos de G^S.

No mapeamento de um enlace virtual e_{ij}^V são considerados apenas enlaces físicos e_{ij}^S que satisfazem $b(e_{ij}^V) \leq \mathcal{R}(e_{ij}^S)$. Se não for possível mapear todos os enlaces virtuais, a solução retornada pela função *mapeamentoAleatórioComL2S2* será inviável. Caso contrário, será uma solução viável, isto é, nós e enlaces virtuais foram alocados com sucesso. Um enlace virtual $e_{ij}^V = (n_i^V, n_j^V)$ é sempre mapeado no caminho físico mais curto, em termos de *hops*, entre os nós físicos $f_n(n_i^V)$ e $f_n(n_j^V)$.

A solução S* representa a melhor solução encontrada pela Busca Tabu. Essa solução também é inicializada com o retorno da função *mapeamentoAleatórioComL2S2* (linha 2). Logo após, é criado o conjunto ELITE que guardará as melhores soluções encontradas pela Busca Tabu (linha 3). Esse conjunto possui tamanho limitado. Se ele estiver lotado e a Busca Tabu encontrar uma nova solução melhor do que a pior existente no conjunto, então esta é substituída por aquela. Ao término da Busca Tabu, é realizado o *path relinking* entre as melhores soluções armazenadas no conjunto ELITE (linhas 23 a 28).

Algoritmo 5 *VNE-TS*

```
1: S := mapeamentoAleatórioComL2S2()
2: S* := S
3: ELITE := {} #ELITE guarda no máximo 5 soluções
4: se viável(S) então
5:    ELITE := ELITE ∪{S}
6: fim se
7: enquanto critério de parada não é satisfeito faça
8:    vizinhos := gerarVizinhosComL2S2(S)
9:    repita
10:      vizinho* := melhorVizinho(vizinhos)
11:      vizinhos := vizinhos−{vizinho*}
12:      se viável(vizinho*) então
13:         S := vizinho*
14:         TABU(S) := tenure
15:         se |ELITE| < 5 então
16:            ELITE := ELITE ∪ {vizinho*}
17:         senão
18:            piorsolução := piorSolução(ELITE)
19:            se vizinho* é melhor que piorsolução então
20:               ELITE := ELITE - {piorsolução}
21:               ELITE := ELITE ∪ {vizinho*}
22:            fim se
23:         fim se
24:         break
25:      fim se
26:   até TABU(vizinho*) = 0 ou !viável(vizinho*)
27:   se qualidade(S) > qualidade(S*) então
28:      S* := S
29:   fim se
30:   atualizar(TABU)
31: fim enquanto
32: para i de 1 até |ELITE| − 1 faça
33:   S := pathRelinking(ELITE[i], ELITE[i + 1])
34:   se qualidade(S) > qualidade(S*) então
35:      S* := S
36:   fim se
37: fim para
38: se viável(S*) então
39:   retorne S*
40: senão
41:   retorne NULL
42: fim se
```

Após as inicializações, o VNE-TS realiza um número fixo de iterações (critério de parada), gerando soluções vizinhas, viáveis ou não, para o mapeamento corrente S (linha 8). A função *gerarVizinhosComL2S2* gera $|N^V|$ soluções vizinhas de S. A i-ésima solução vizinha difere da solução S na posição i do vetor solução, isto é, o i-ésimo nó é mapeado para outro nó físico através do procedimento L2S2. Após essa mudança, tenta-se mapear os enlaces virtuais afetados. Se não for possível reservar os caminhos físicos necessários, então a solução vizinha será inviável. A Figura 4.2 ilustra as soluções vizinhas para o mapeamento da VN1 mostrado na Figura 4.1. Note que nenhum dos vizinhos possui custo, em termos da função objetivo, melhor do que o mapeamento original.

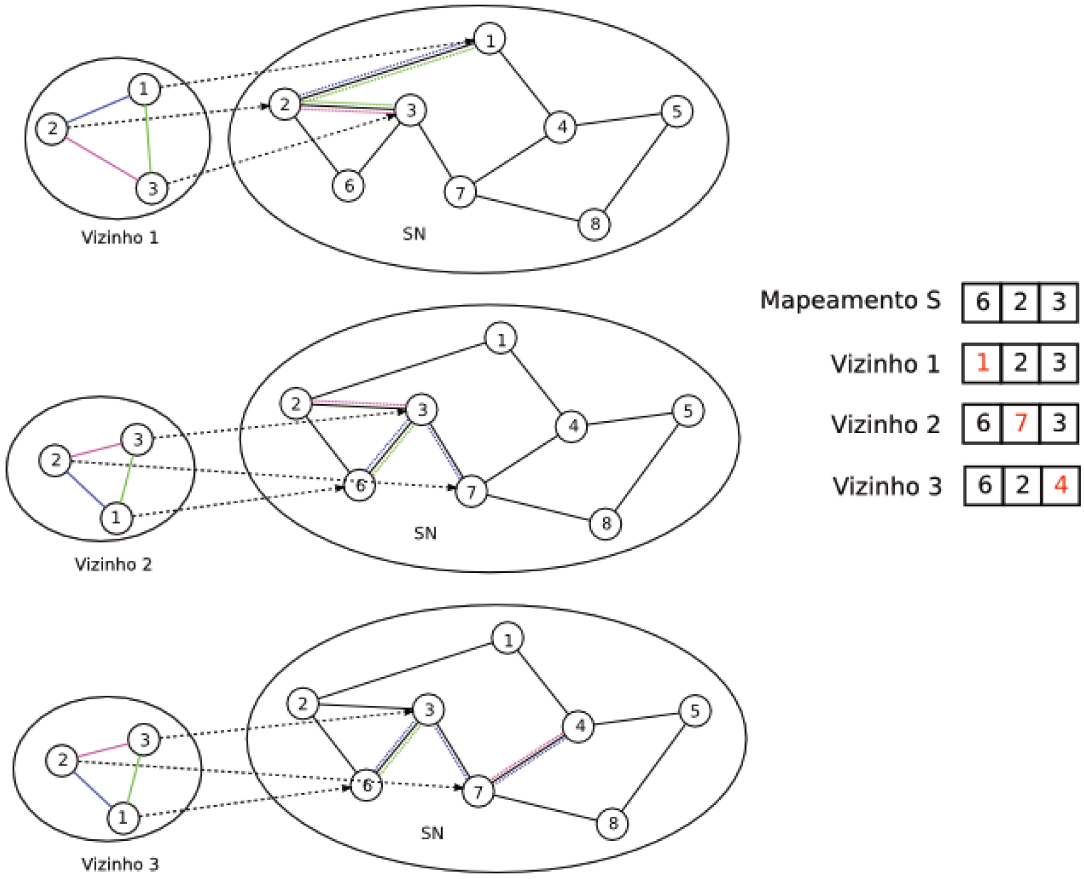


Figura 4.2: Um mapeamento S e possíveis vizinhos

Após gerar os vizinhos da solução corrente S, escolhe-se a melhor solução viável não TABU (linhas 10). No exemplo da Figura 4.2, os vizinhos 1 e 2 são melhores que o vizinho 3, pois usam menos enlaces. Contudo, duas soluções que usam a mesma quantidade de enlaces não possuem, necessariamente, o mesmo custo. Os enlaces podem ter diferentes requisitos de banda. Portanto, pode ser preferível alocar menos enlaces físicos para o enlace virtual com maior requisito. Ainda na Figura 4.2, todos os vizinhos são viáveis.

Mas isso nem sempre acontece. Devido à falta de recursos nos enlaces físicos, pode não ser possível alocar alguns ou mesmo todos os enlaces virtuais. Caso isso ocorra, o vizinho deve ser rejeitado.

Na Busca Tabu, sempre que se passa de uma solução corrente S para um vizinho S_i, esse movimento torna-se TABU, isto é, proibido, por um certo número de iterações. O objetivo dessa proibição é evitar ciclagens e contornar ótimos locais. Desta forma, para que os vizinhos 1 e 2 da Figura 4.2 sejam considerados melhores do que o terceiro, as atribuições S[1] := 1 e S[2] := 7 não podem estar no rol de atribuições proibidas da lista tabu.

Caso o melhor vizinho não TABU encontrado seja viável, a solução corrente é atualizada para ele e a posição na lista TABU, associada ao movimento de mudança de S para a solução *vizinho**, recebe o valor *tenure* (linhas 12, 13 e 14). Esse parâmetro corresponde ao número de iterações que o movimento permanecerá proibido. Se houver espaço no conjunto de soluções ELITE, a solução vizinho* é adicionada ao conjunto. Se o conjunto ELITE estiver cheio e vizinho* for melhor do que a pior solução em ELITE, então esta é substituída pelo mapeamento vizinho* (linha 15).

Se a nova solução S for melhor do que o melhor mapeamento S* encontrado até o momento, em termos da função objetivo, S* é atualizado para S (linhas 19 e 20). Após cada iteração, os valores das posições da lista tabu são decrementadas (linha 22). Ao fim da última iteração, realiza-se o *path relinking*, e finalmente o algoritmo retorna o melhor mapeamento viável S*, ou *NULL* se nenhum mapeamento viável foi encontrado.

O algoritmo VNE-TS superou o VNE-PSO[21] em todas as métricas consideradas. Os resultados são mostrados no Capítulo 5.

4.6 O algoritmo de mapeamento VNE-TS-*Clustering*

O algoritmo VNE-TS-*Clustering* é mais uma proposta deste trabalho. Ele difere do VNE-TS na construção da primeira solução S (linha 1), conforme mostra o Algoritmo 6.

A função *mapeamentoAleatórioComKmeans* tenta construir uma solução inicial S, de modo que todos os nós virtuais fiquem próximos. Quanto mais próximos eles estiverem, menos enlaces físicos serão usados na segunda fase do mapeamento. O Algoritmo 7 detalha a função *mapeamentoAleatórioComKmeans*.

A função *mapeamentoAleatórioComKmeans* começa definindo o número de *clusters*, que depende da razão entre o número de nós físicos e o número de nós virtuais (linha 1). Se essa razão for grande, então tem-se mais *clusters*.

Após a definição do número de *clusters*, escolhe-se aleatoriamente k nós físicos para serem os centróides dos *clusters* (linha 2). Então, o algoritmo atribui cada nó virtual ao *cluster* cujo centróide é o mais próximo, em termos de *hops* (linha 4). Após isso, para

Algoritmo 6 *VNE-TS-Clustering*

```
1: S := mapeamentoAleatórioComKmeans()
2: S* := S
3: ELITE := {}
4: se viável(S) então
5:    ELITE := ELITE ∪{S}
6: fim se
7: enquanto critério de parada não é satisfeito faça
8:    vizinhos := gerarVizinhosComL2S2(S)
9:    repita
10:      vizinho* := melhorVizinho(vizinhos)
11:      vizinhos := vizinhos−{vizinho*}
12:      se viável(vizinho*) então
13:         S := vizinho*
14:         TABU(S) := tenure
15:         ELITE := ELITE ∪ {vizinho*}
16:         break
17:      fim se
18:   até TABU(vizinho*) = 0 ou !viável(vizinho*)
19:   se qualidade(S) > qualidade(S*) então
20:      S* := S
21:   fim se
22:   atualizar(TABU)
23: fim enquanto
24: para i de 1 até |ELITE| − 1 faça
25:   S := pathRelinking(ELITE[i], ELITE[i + 1])
26:   se qualidade(S) > qualidade(S*) então
27:      S* := S
28:   fim se
29: fim para
30: se viável(S*) então
31:   retorne S*
32: senão
33:   retorne NULL
34: fim se
```

Algoritmo 7 *mapeamentoAleatórioComKmeans*

1: seja k o número de clusters, com $k = |N^S|/|N^V|$
2: escolha aleatoriamente k nós físicos para serem centróides dos k clusters
3: **repita**
4: atribua cada nó físico n_i a um *cluster* com centróide k_j, de modo que a distância entre n_i e k_j, em termos de *hops*, seja a menor dentre todos os centróides
5: **para cada** *cluster* c_j **faça**
6: calcule a soma das distâncias de cada nó de c_j para todos os demais nós de c_j. Escolha como centróide do cluster c_j o nó físico que possui a menor soma de distâncias
7: **fim para**
8: **até** que não haja mudança nos centróides
9: ordene os k clusters por maior soma de valores NR
10: tente alocar os nós virtuais para o *cluster* com maior soma NR e que possua pelos menos $|N^V|$ nós. Se não for possível, tente alocar os nós no próximo *cluster* da ordem.
11: se nenhum *cluster* puder receber todos os nós virtuais, rejeite a requisição
12: se a alocação dos nós virtuais foi realizada com sucesso em algum *cluster*, tente alocar os enlaces virtuais
13: **retorne** a alocação construída, seja viável ou não

cada nó de um *cluster* específico, o algoritmo calcula a soma da sua distância mínima para os demais nós do mesmo *cluster*, considerando apenas enlaces físicos cuja banda residual é maior ou igual do que a banda do enlace virtual com menor requisito (linhas 5, 6 e 7). Esse processo se repete até que não haja mudanças nos centróides dos *clusters*.

Ao término do *loop* (linhas 3 à 8), tem-se k *clusters*, com quantidades diferentes de nós, agrupados segundo a proximidade existente entre eles. É possível que um ou mais *clusters* possuam um número de nós físicos menor do que o número de nós virtuais requisitados. Logo, os nós virtuais não poderão ser mapeados para eles. Em contrapartida, poderá existir um ou mais *clusters* com uma quantidade de nós físicos maior do que o número de nós virtuais. Esses *clusters* são os candidatos a atenderem à requisição de mapeamento. Em síntese, todos os nós virtuais de uma requisição devem ser alocados no mesmo *cluster*.

Para balancear o uso dos recursos físicos, devem-se escolher *clusters* cujos elementos físicos possuem mais recursos disponíveis. Assim, os *clusters* candidatos são ordenados pela soma dos valores NR (definido na seção 4.3) de seus respectivos nós (linha 9). O algoritmo tenta então alocar os nós virtuais em um *cluster*, seguindo a ordem definida, até que um deles possa atender a todos os nós (linha 10).

Se nenhum *cluster* puder receber todos os nós virtuais, então a requisição deve ser rejeitada (linha 11). Caso contrário, tenta-se realizar o mapeamento dos enlaces (linha 12). O mapeamento construído, seja apenas de nós, ou de nós e enlaces é retornado pela

função (linha 13).

Devido à estratégia de construção da solução inicial, que tenta mapear todos os nós virtuais em uma mesma região do substrato, o VNE-TS-*Clustering* não se aplica à cenários com restrição de localização, porque essa restrição pode exigir que dois nós virtuais sejam mapeados para regiões totalmente diferentes e distantes.

4.7 O algoritmo de seleção 2ks-VN-*Selector*

Os algoritmos de mapeamento propostos em [23, 25, 26, 21, 28], tentam mapear as requisições imediatamente após a sua chegada. A longo prazo, esta abordagem leva à alocação de redes que possuem baixo custo-benefício.

Yu et al.[24] propuseram um sistema de seleção para requisições de redes virtuais, em oposição ao sistema de alocação imediata utilizado por outros trabalhos. O sistema proposto aguarda durante uma janela de tempo w, antes de tentar realizar alocações. Todas as requisições que chegam durante a janela de tempo w são agrupadas. Ao término de w, as requisições são ordenados em ordem não crescente de ganho e um algoritmo de mapeamento tenta alocá-las na ordem definida.

Propõe-se um novo algoritmo de ordenação-seleção de requisições, que baseia-se no Problema da Mochila Bidimensional 0-1 [30]. Esse problema é conhecido também como *bi-knapsack* e sua forma geral é definida pelo seguinte Programa Linear Inteiro (PLI):

$$\begin{bmatrix} max & cx \\ s.a. & A_1x \leq b_1 \\ & A_2x \leq b_2 \\ & x \in B^n \end{bmatrix}$$

A ideia geral do algoritmo 2ks-VN-*Selector* é considerar as requisições agrupadas durante a janela de tempo w, como os itens do problema da mochila. A rede física é considerada como a mochila, na qual se quer inserir o maior número de itens possíveis. É comum que todos os itens não caibam simultaneamente na mochila. Assim, escolhe-se um subconjunto de itens que resultem no maior lucro.

O nosso algoritmo de seleção tem por objetivo selecionar redes virtuais com maior custo-benefício. Para um conjunto de redes $C = \{G_1^V, G_2^V, ..., G_n^V\}$, agrupadas em uma janela de tempo w, o algoritmo de seleção tenta maximizar o sistema linear *2ks-VN*, resolvendo-o através de Programação Linear Inteira, com o *solver CPLEX*.

$$\textit{2ks-VN} = \left[\begin{array}{ll} max & \sum_{i=1}^{n} \mathcal{G}(G_i^V) x_i \\ s.a. & (\sum_{n_j^V \in N_i^V} p(n_j^V)) x_i \leq \sum_{n_j^S \in N^S} \mathcal{R}(n_j^S) \\ & (\eta \sum_{e_{ij}^V \in E_i^V} b(e_{ij}^V)) x_i \leq \sum_{e_{ij}^S \in E^S} \mathcal{R}(e_{ij}^S) \\ & x \in B^n \end{array} \right.$$

Para cada rede virtual, tem-se uma variável x_i indicando se a rede virtual G_i^V deve ser alocada ou não.

Em resumo, ao maximizar o sistema *2ks-VN*, procura-se um conjunto de redes virtuais cujo somatório dos requisitos de processamento, em cada rede, seja menor ou igual ao somatório do processamento nos nós físicos (primeira restrição do modelo).

Almeja-se, também, que a rede física possua oferta de banda suficiente para atender aos enlaces virtuais de cada rede do conjunto procurado. Assim, o somatório das bandas exigidas por uma rede virtual, multiplicada pela média de enlaces físicos usados em alocações prévias (denotada por η) também deve ser menor do que o total de banda ofertado pelo substrato (segunda restrição do modelo). A multiplicação é realizada para compensar o fato de que enlaces virtuais, em geral, são alocados para mais de um enlace físico.

No sistema *2ks-VN*, $\mathcal{R}(n_j^S)$ e $\mathcal{R}(e_{ij}^S)$ definem o processamento e largura de banda residuais de nós e enlaces físicos, respectivamente. A constante η é definida com base na média dos tamanhos de todos os caminhos físicos alocados anteriormente, isto é, dado um conjunto de requisições de redes virtuais mapeadas anteriormente a um tempo T, $VNs = \{G_1^V, G_2^V, ..., G_n^V\}$, o valor de η é denotado por

$$\eta = \frac{\sum_{G^V \in VNACC(T)} \sum_{e_{ij}^V \in E^V} |f_l(e_{ij}^V)|}{\sum_{G^V \in VNACC(T)} |E^V|}$$

O algoritmo 2ks-VN-*Selector* resolve o sistema linear inteiro *2ks-VN*. Após a resolução tem-se dois conjuntos de redes virtuais, S e N, que contêm as redes virtuais selecionadas e as não selecionadas, respectivamente. S e N satisfazem $S \cap N = \emptyset$ e $S \cup N = VNs$. O algoritmo de seleção então ordena as redes virtuais de S e de N em ordem não crescente de ganho. A ordem nos conjuntos S e N definem a ordem de seleção. Contudo, não há garantia de que todas, ou mesmo alguma, das requisições em S serão alocadas. Isto ocorre porque em situações excepcionais pode haver algum enlace em alguma rede virtual em S, cujo requisito de banda é muito alto. Além disso, algumas redes em N podem ser alocadas, visto que η é uma estimativa. O 2ks-VN-*Selector* é exibido no Algoritmo 8.

Algoritmo 8 *2ks-VN-Selector*

1: S, N := solução do PLI 2ks-VN(VNs)
2: ordene S
3: ordene N
4: **para cada** $vn \in S$ **faça**
5: tente mapear vn com algum algoritmo de mapeamento
6: **fim para**
7: **para cada** $vn \in N$ **faça**
8: tente mapear vn com algum algoritmo de mapeamento
9: **fim para**

Capítulo 5

Avaliação dos Algoritmos Propostos

Este capítulo apresenta os resultados obtidos com a execução dos algoritmos VNE-TS, VNE-TS-*Clustering* e VNE-PSO[21], combinados com as estratégias de seleção 2ks-VN-*Selector* e *Most-prize-first*[24], de acordo com o cenário mostrado na Figura 1.3. Os resultados foram obtidos através de simulação.

A Seção 5.1 descreve o ambiente de simulação considerado. A Seção 5.2 apresenta os experimentos realizados e seus objetivos. A Seção 5.3 apresenta os resultados obtidos em um cenário com restrição de localização. A Seção 5.4 discute os resultados obtidos em um ambiente sem restrição de localização.

Em todos os gráficos exibidos neste capítulo, os pontos representam a média de 30 execuções independentes e os intervalos de confiança são mostrados com 95% de confiança.

Nos gráficos apresentados, a sigla MPF significa *Most Prize First*, e que KNA significa a política de seleção 2ks-VN-*Selector*. A nomenclatura MPF-VNE-TS significa que os dados correspondem à combinação da políticia de seleção *Most Prize First*, com o algoritmo de mapeamento VNE-TS. As demais combinações seguem o mesmo padrão.

5.1 Ambiente de simulação

.

O ambiente de simulação é similar aos ambientes utilizados em trabalhos na literatura [23, 21]. A topologia física foi configurada para ter 100 nós e 316 enlaces em média, correspondendo a um ISP de tamanho médio. Os recursos de processamento e largura de banda nos nós e enlaces físicos são valores inteiros uniformemente distribuídos no intervalo [50,100].

Em cada rede virtual, o número de nós é determinado por uma distribuição uniforme no intervalo [2,10]. A probabilidade de conectividade entre nós das redes virtuais é 0.5, isto é, uma rede virtual com n nós possui em média $n \cdot (n-1)/4$ enlaces. Os recursos

de processamento e largura de banda requisitados por nós e enlaces virtuais são valores inteiros uniformemente distribuídos no intervalo [10,20]. As topologias de todas as redes, físicas e virtuais, foram geradas pela ferramenta *Georgia Tech Internet Topology Modeling tool* (GT-ITM)[31]. As coordenadas x e y de nós físicos e virtuais estão distribuídas uniformemente no intervalo [0,1000]. O requisito de localização para nós de uma rede virtual G^V, quando existente, é determinado por uma distribuição uniforme no intervalo [100,200].

O tempo de vida de redes virtuais segue uma distribuição exponencial com média $\mu = 1000$ unidades de tempo. A frequência de chegada de requisições de redes virtuais segue uma distribuição de Poisson com média λ. Foram realizados experimentos com várias taxas de chegada, a saber $\lambda = 5, 10, 15, 20$ e 25 requisições por 100 unidades de tempo. A duração de cada experimento foi de 40000 unidades de tempo. Com o objetivo de eliminar efeitos de transiência no experimento, a coleta de dados é iniciada após 20000 unidades de tempo.

Os parâmetros da metaheurística *Particle Swarm* no algoritmo VNE-PSO foram definidos conforme descrito em [21]: tamanho da população igual a 5 partícilas, $\alpha = 0.1$ e $\delta = 0.7$. O algoritmo de nuvem de partículas utilizado para realizar o mapeamento das redes virtuais e apresentado em [21], não utiliza o parâmetro γ, definido na linha 4 do Algoritmo 2. Isto é, não utiliza conjunto de partículas informantes para cada elemento da população.

Os parâmetros da Busca Tabu foram escolhidos através de simulações com um subconjunto de instâncias. Foram realizados experimentos com valores do parâmetro *tenure* iguais a 5, 10 e 15. Em ambos os ambientes de simulação, com e sem restrição de localização, o melhor resultado foi obtido com *tenure* = 10. Após a escolha do parâmetro *tenure*, foram realizados experimentos para definir o tamanho máximo do conjunto ELITE, usado pelo *path relinking*. Os melhores resultados foram obtidos com |ELITE| = 5.

O critério de parada em todos os algoritmos de mapeamento foi definido como o término de 100 iterações.

5.2 Experimentos realizados

Esta seção descreve os experimentos realizados e seus objetivos. Em cada experimento foram examinados dois cenários. No primeiro deles, considera-se a restrição de localização para nós e são aplicados os algoritmos VNE-TS e VNE-PSO. No segundo cenário, não foi levado em conta a restrição de localização. Assim, neste último caso, além dos algoritmos VNE-TS e VNE-PSO, empregou-se o VNE-TS-*Clustering*.

5.2.1 Análise de evolução do custo

O objetivo da Análise de evolução do custo é analisar como o custo do mapeamento de uma única requisição varia ao longo de cem iterações realizadas por cada algoritmo de mapeamento. Foram executadas trinta repetições. Em cada repetição foram consideradas uma rede física distinta, com recursos suficientes para atender qualquer requisição, e uma rede virtual com topologias aleatórias e número de nós fixo.

Através da análise de evolução do custo espera-se comparar quão rápido os algoritmos de mapeamento alcançam mapeamentos com custo baixo. Essa análise pode determinar se o número de iterações dadas para a realização do mapeamento é pequeno ou grande demais. Além disso, pode-se verificar quais algoritmos convergem mais rapidamente para uma solução com menor custo, evitando o desperdício de recursos utilizados por algoritmos que demoram muito para convergir.

5.2.2 *Performance Profiles*

Performance Profiles é uma ferramenta para avaliar e comparar o desempenho de algoritmos de otimização. Essa ferramenta considera uma função de distribuição cumulativa para mensuração do desempenho dos algoritmos [32]. Como métrica de desempenho, foi utilizada a razão "custo obtido" dividido pelo "melhor custo obtido" dentre todos os algoritmos.

Foram executadas trinta repetições. Cada repetição possui uma rede física distinta e conjuntos distintos de 500 instâncias de redes virtuais, com tamanhos e topologias aleatórias, de acordo com o perfil descrito na Seção 5.1.

Em resumo, cada ponto em um gráfico *performance profile* determina o percentual de redes virtuais cuja razão "custo obtido"/"melhor custo" obtido é menor ou igual à razão determinada pelo eixo x do gráfico.

5.2.3 Simulações de rede

Nas simulações de rede, são analisados os resultados obtidos durante a simulação de chegada de requisições durante 40.000 instantes de tempo. O resultado obtido nos 20.000 primeiros instantes não são exibidos, para eliminar efeitos de transiência na simulação. Apenas o resumo dos resultados obtidos com as requisições realizadas nos 20.000 instantes de tempo finais são exibidos nos gráficos. O desempenho dos algoritmos é analisado considerando as métricas relacionadas na Seção 3.4.

Espera-se tirar conclusões sobre o comportamento dos algoritmos de mapeamento e das políticas de seleção, em ambientes nos quais as requisições chegam aleatoriamente ao longo do tempo. Todos os experimentos de simulação de redes apresentados nessa dis-

sertação foram realizados em trabalhos anteriores na literatura [21]. Os gráficos mostram o resultado de trinta repetições, e $T = 40.000$.

5.3 Experimentos em cenário com restrição de localização

Conforme discutido no Capítulo 3, a restrição de localização determina que um nó virtual seja mapeado apenas para nós físicos cuja distância euclidiana não exceda um certo valor. Esta seção apresenta os resultados obtidos ao considerar essa restrição. O algoritmo VNE-TS-*Clustering* não se aplica a este cenário, devido aos motivos discutidos no final da seção 4.6.

5.3.1 Análise de evolução do custo

A Figura 5.1 mostra a evolução dos custos obtidos pelos algoritmos VNE-TS e VNE-PSO, ao longo de 100 iterações. Cada gráfico mostra o resultado ao considerar redes virtuais com tamanhos distintos, a saber com:

- 2 nós - Figura 5.1a
- 4 nós - Figura 5.1b
- 6 nós - Figura 5.1c
- 8 nós - Figura 5.1d
- 10 nós - Figura 5.1e

Os valores numéricos no eixo y dos gráficos da Figura 5.1 são valores escalares obtidos pela definição de custo dada na seção 3.4. Quanto maior o custo, pior o mapeamento. Em todos os gráficos da Figura 5.1, ocorre sobreposição dos intervalos de confiança. Apesar de o VNE-TS apresentar a melhor média em todos os gráficos, essa sobreposição indica a possibilidade de o VNE-PSO superar o VNE-TS em alguns mapeamentos.

A análise da evolução do custo mostra o quão rápido os dois algoritmos encontram soluções com custos menores, considerando os cinco tamanhos distintos de redes virtuais. Abaixo, os efeitos do tamanho da rede virtual a ser mapeada são discutidos, de acordo com os cinco gráficos exibidos na Figura 5.1.

A Figura 5.1a mostra que, para redes com dois nós virtuais, o VNE-TS começou com custo médio igual a 28, ao passo que o VNE-PSO iniciou com custo médio 25. Entretanto, após as primeiras 10 iterações, o VNE-TS passa a obter, em média, mapeamentos com

custo menor. Essa tendência se mantém até o término das 100 iterações. A Figura 5.1a mostra ainda uma convergência das curvas do VNE-TS e VNE-PSO, a partir da trigésima iteração. Isso indica que os algoritmos encontraram um mapeamento com um determinado custo, e nas setenta iterações restantes não conseguiram encontrar um mapeamento melhor. Ao final das cem iterações, o VNE-TS alcançou custo médio 18, contra custo médio 20 obtido pelo VNE-PSO, resultando em uma melhora de cerca de 10%, em média. A diferença absoluta entre o custo médio obtido pelos dois algoritmos de mapeamento é de 2 pontos. Conforme será visto abaixo, essa diferença absoluta tende a aumentar com o tamanho da rede virtual.

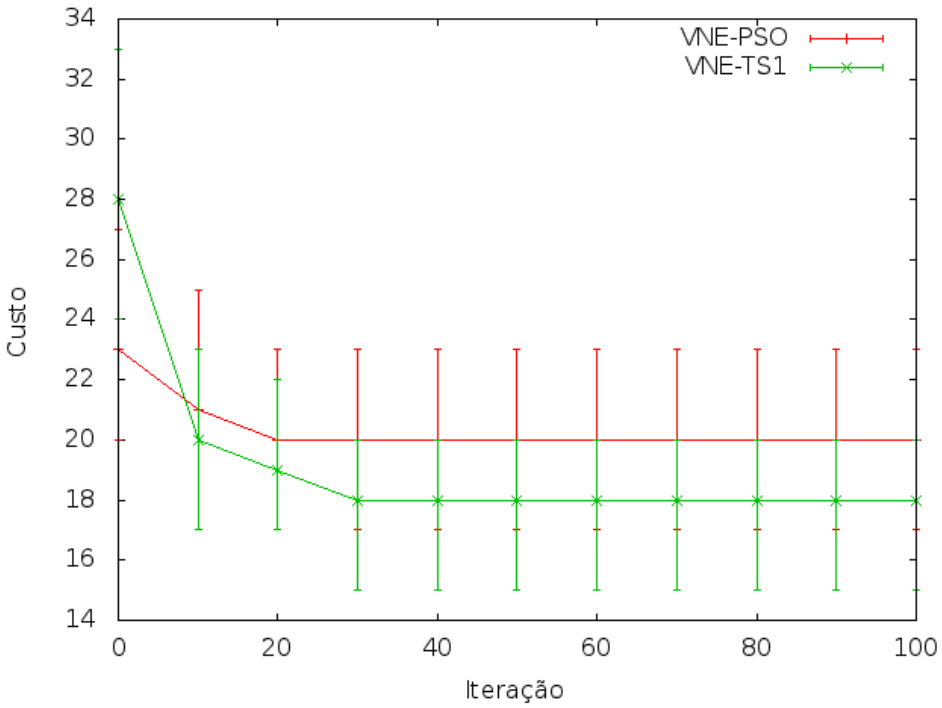
A Figura 5.1b mostra os resultados de 100 iterações, considerando o mapeamento de redes virtuais com 4 nós. Note que as faixas numéricas neste gráfico são maiores do que as observadas no gráfico 5.1a. Isso se deve ao fato de redes virtuais maiores naturalmente demandarem mais recursos e terem maior custo de mapeamento. No gráfico 5.1b, mais uma vez, o VNE-TS começou com uma solução pior do que a do VNE-PSO. Entretanto, após cerca de 20 iterações, o VNE-TS passa a encontrar soluções com menores custos. Ao final das 100 iterações, o VNE-TS alcançou custo médio 92, contra 101 obtido pelo VNE-PSO, o que indica que o VNE-TS obtém em média mapeamentos 10% melhores quando considera-se redes virtuais com 4 nós. Contudo, a sobreposição dos intervalos de confiança no gráfico 5.1b, indica que o VNE-PSO pode superar o VNE-TS em alguns mapeamentos, apesar de ser inferior na média de 30 execuções. O gráfico 5.1b mostra ainda uma diferença absoluta média de 9 pontos.

Na Figura 5.1c, ambos os algoritmos de mapeamento iniciaram com mapeamentos com custos similares. Entretanto, o VNE-TS passou a obter melhores mapeamentos após 10 iterações. Ao fim da centésima iteração, a diferença absoluta média entre os custos obtidos pelos dois algoritmos foi de 12 pontos. O VNE-TS terminou com uma solução com custo médio de 196, ao passo que o VNE-PSO terminou com uma solução de custo médio 208, resultando em uma diferença de cerca de 6% entre os dois algoritmos.

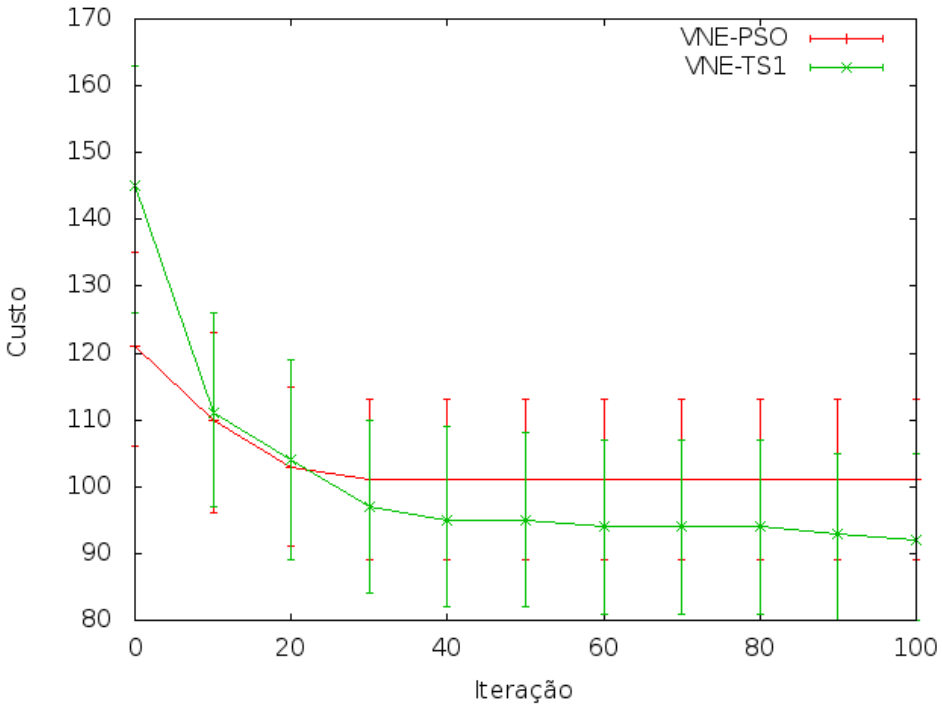
Na Figura 5.1d, os resultados do VNE-TS permanecem melhores do que os do VNE-PSO, em média, ao longo das 100 iterações. Mais uma vez a sobreposição dos intervalos de confiança indica que em algumas execuções, o VNE-PSO pode alcançar mapeamentos com menores custos em relação ao VNE-TS, apesar de ter obtido custo médio de mapeamento menor, nas 30 execuções. Nota-se, também, um aumento natural no custo dos mapeamentos, dado que redes com 8 nós terão em média mais enlaces virtuais do que redes com 6, 4 e 2 nós. Ao término de 100 iterações, a diferença absoluta foi de 23 pontos no custo médio. O VNE-TS superou o VNE-PSO em cerca de 6% em média.

A Figura 5.1e mostra os resultados obtidos ao considerar redes virtuais com 10 nós. Assim como ocorreu com redes virtuais com 8 nós, o VNE-TS superou o VNE-PSO ao longo das 100 iterações, ao se levar em consideração a média dos custos obtidos. Mais

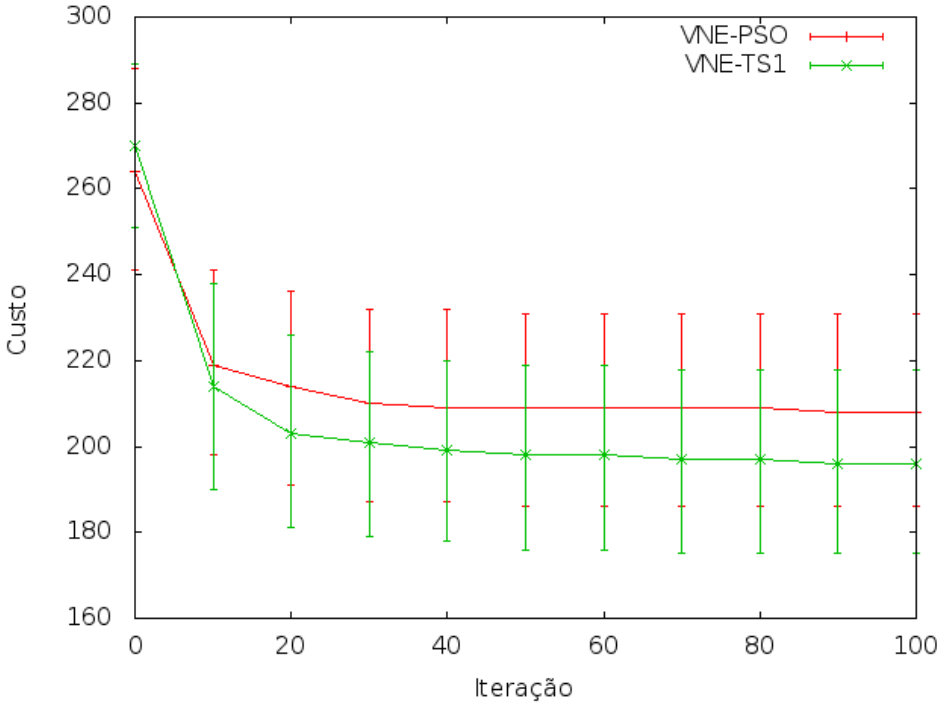
uma vez nota-se um aumento na diferença absoluta entre os custos obtidos ao término da execução. O VNE-TS obteve custo médio de mapeamento 684, ao passo que o VNE-PSO alcançou custo médio de mapeamento 605, resultando numa diferença absoluta de 79 pontos. Nesse último grupo de instâncias, o VNE-TS encontrou soluções em média 13% melhores do que as encontradas pelo VNE-PSO.


VNE-PSO
VNE-TS1
Custo
Iteração
34
32
30
28
26
24
22
20
18
16
14
0
20
40
60
80
100

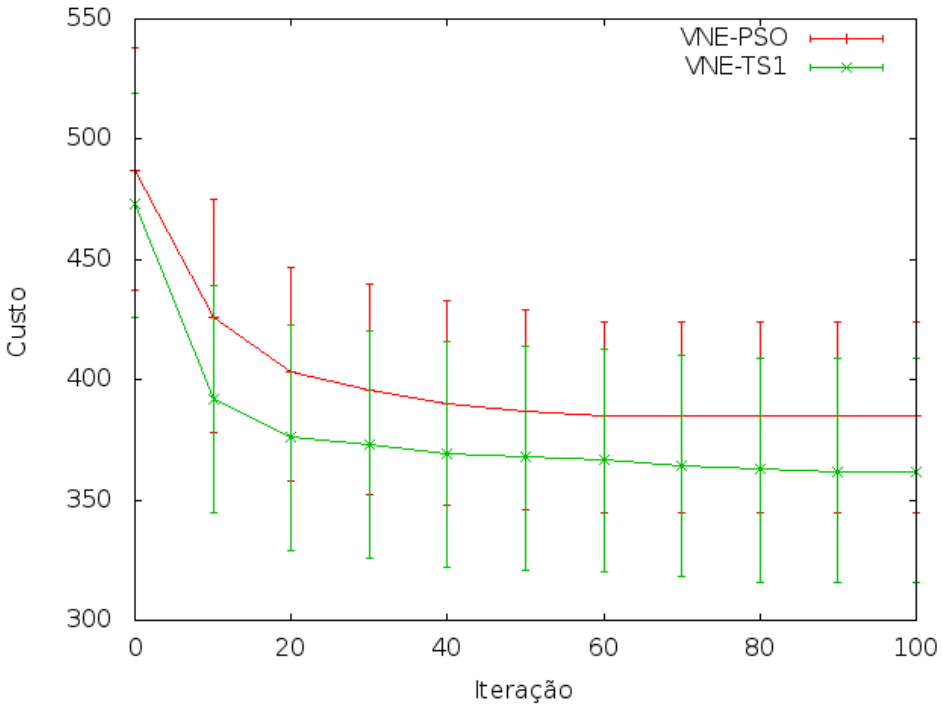
(a) 2 nós


VNE-PSO
VNE-TS1
Custo
Iteração
170
160
150
140
130
120
110
100
90
80
0
20
40
60
80
100

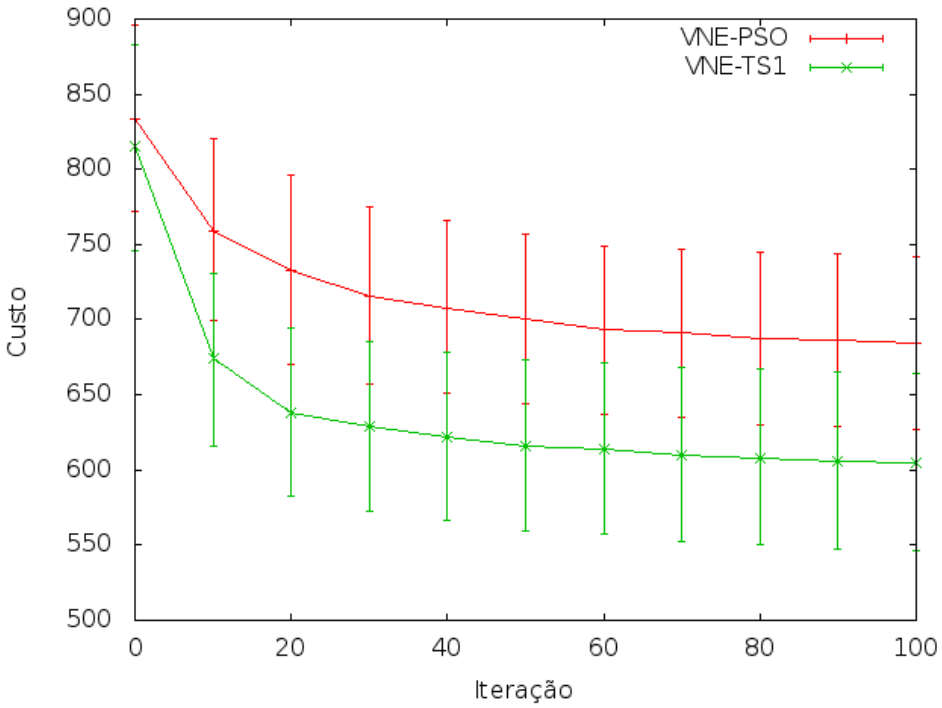
(b) 4 nós


VNE-PSO
VNE-TS1
Custo
Iteração
300
280
260
240
220
200
180
160
0
20
40
60
80
100

(c) 6 nós


VNE-PSO
VNE-TS1
Custo
Iteração
550
500
450
400
350
300
0
20
40
60
80
100

(d) 8 nós



(e) 10 nós

Figura 5.1: Desempenho dos algoritmos de mapeamento com redes virtuais de diferentes tamanhos (com restrição de localização)

5.3.2 *Performance profiles*

A Figura 5.2 mostra os resultados obtidos com o *performance profiles*, em cenário com localização de restrições. Esse experimento foi executado com 500 redes virtuais com tamanhos aleatórios no intervalo [2,10].

O ponto 1 do eixo x determina o percentual de melhores mapeamentos alcançados pelo algoritmo de mapeamento. Assim, das 500 redes virtuais aleatórias consideradas nesse experimento, o VNE-TS obteve custo menor em cerca de 78% delas. O VNE-PSO, por sua vez, obteve custo menor de mapeamento em cerca de 43% dos mapeamentos realizados. No caso em que os dois algoritmos obtém o mesmo custo de mapeamento para uma mesma instância, esse percentual é computado para ambos os algoritmos.

Mesmo em instâncias nas quais o VNE-PSO alcançou melhor mapeamento em comparação ao VNE-TS, os custos de mapeamento obtidos pelo VNE-TS não foram muito distantes dos obtidos pelo VNE-PSO. Isso é observado através da menor curvatura da linha VNE-TS.

Há uma grande probabilidade de a maioria dos empates haverem ocorrido em mapea-

mento de redes virtuais com menor quantidade de nós, visto que a diferença absoluta nos custos de mapeamento tende a ser menor, conforme discutido na Seção 5.3.1.

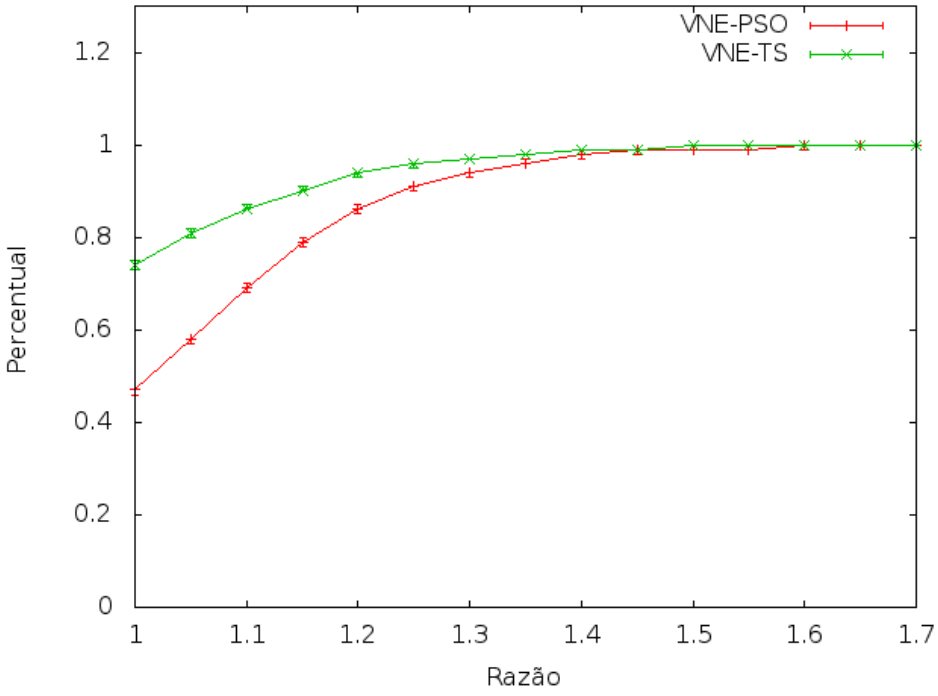


Figura 5.2: *Performance profiles* (com restrição de localização)

5.3.3 Simulações de rede

As simulações de rede têm como objetivo analisar o desempenho dos algoritmos segundo as métricas discutidas na seção 3.4. A simulação visa analisar o comportamento dos algoritmos ao longo do tempo. Neste trabalho, foram considerados 40.000 instantes de tempo, como explicado na Seção 5.1.

Taxa de bloqueio de requisições

O gráfico mostrado na Figura 5.3 mostra a taxa média de bloqueio de requisição a longo prazo, obtida após 40000 instantes de tempo de execução.

Com taxa de chegada igual a 5 requisições por 100 unidades de tempo, o VNE-TS e o VNE-PSO bloqueiam aproximadamente a mesma quantidade de requisições, independentemente do algoritmo de seleção utilizado (*Most Prize First* ou 2ks-VN-*Selector*), isto é aproximadamente 23% de rejeição.

Com 10 requisições por 100 unidades de tempo, o algoritmo VNE-TS bloqueia menos requisições que o VNE-PSO. A separação entre as linhas relativas ao VNE-PSO e VNE-TS são claras nesse ponto. Com taxa de chegada igual a 10, os algoritmos de seleção também não causam aumento ou diminuição significativas na taxa de bloqueio obtidas pelos algoritmos de mapeamento. Isto é verificado pela sobreposição das retas MPF-VNE-TS e KNA-VNE-TS, além da sobreposição das retas MPF-VNE-PSO e KNA-VNE-PSO.

A partir de 15 requisições por 100 unidades de tempo, a política de seleção 2ks-VN-*Selector* passa a afetar a taxa de bloqueio do VNE-PSO, reduzindo-a substancialmente em relação à política *Most Prize First*. Essa tendência se mantém para as taxas de chegada 20 e 25 requisições por 100 unidades de tempo, conforme mostram as retas MPF-VNE-PSO e KNA-VNE-PSO. Entre 20 e 25 requisições por 100 unidades de tempo, tem-se um ganho de aproximadamente 4% no VNE-PSO, apenas por trocar de política de seleção de requisições.

As retas MPF-VNE-TS e KNA-VNE-TS não se distanciam uma da outra nas várias taxas de chegada consideradas. Apesar disso, nota-se uma diminuição de 1% na taxa de bloqueio quando a taxa de chegada excede 20 requisições por 100 unidades de tempo.

O gráfico da Figura 5.3 mostra que a menor taxa de bloqueio foi obtida pela combinação do algoritmo de mapeamento VNE-TS com a política de seleção 2ks-VN-*Selector*. A combinação que causou maior taxa de bloqueio foi a do algoritmo VNE-PSO com a política *Most Prize First*.

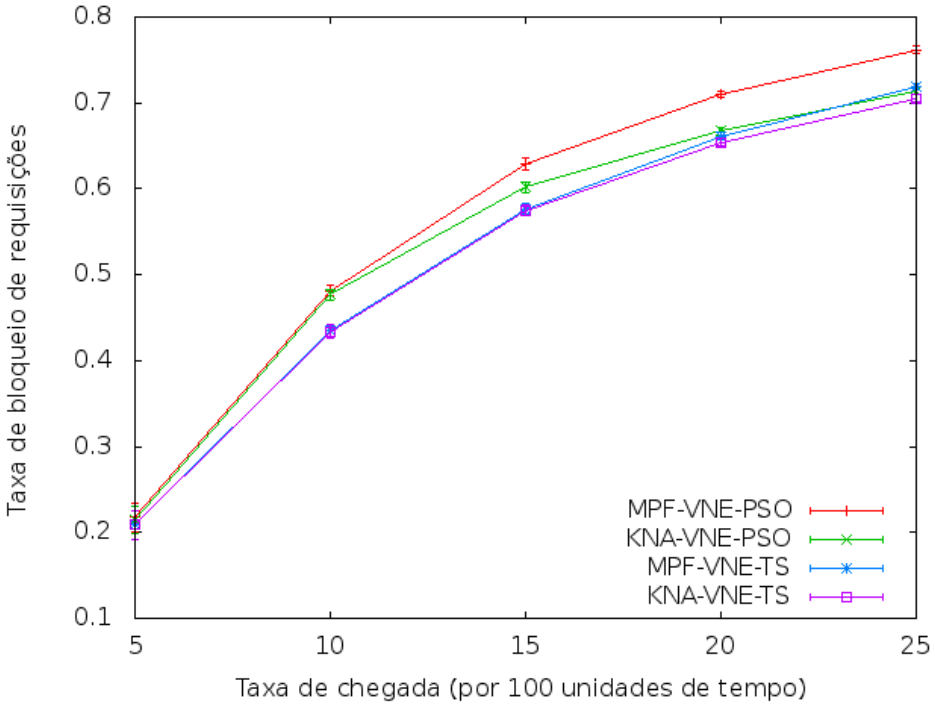


Figura 5.3: Taxa de bloqueio

Ganho médio a longo prazo

O gráfico da Figura 5.4 mostra o rendimento médio a longo prazo, obtido ao término de 40000 instantes de tempo de execução.

Com taxa de chegada igual a 5 requisições por 100 unidades de tempo, os algoritmos de mapeamento VNE-TS e VNE-PSO alcançaram aproximadamente o mesmo rendimento médio a longo prazo.

A partir de 10 requisições por 100 unidades de tempo, o algoritmo VNE-TS obtém maior rendimento que o VNE-PSO. As sobreposições da linha KNA-VNE-TS com a linha MPF-VNE-TS e da linha KNA-VNE-PSO com a linha MPF-VNE-PSO, demonstram que as políticas de seleção não causaram impacto no ganho médio a longo prazo, quando a taxa de chegada é 5 ou 10 requisições por 100 unidades de tempo.

Acima de 15 requisições por 100 unidades de tempo, o uso da política de seleção 2ks-VN-*Selector* passa a causar melhora substancial no ganho obtido pelo algoritmo VNE-PSO, em relação à sua combinação com a política de seleção *Most Prize First*. Isso acontece porque a política 2ks-VN-*Selector* seleciona redes virtuais com maior custo benefício.

A partir de taxas de chegada de 10 requisições por 100 unidades de tempo, o VNE-TS obteve maior rendimento a longo prazo, se comparado ao VNE-PSO. Contudo, a política

de seleção 2ks-VN-*Selector* não causou grande impacto no ganho médio a longo prazo nesse algoritmo, ao contrário do que ocorreu com o VNE-PSO. Apesar disso, quando a taxa de chegada é de 25 requisições por 100 unidades de tempo, a combinação da política 2ks-VN-*Selector* com o VNE-TS supera todas as demais combinações consideradas, em termos de aumento do ganho médio a longo prazo.

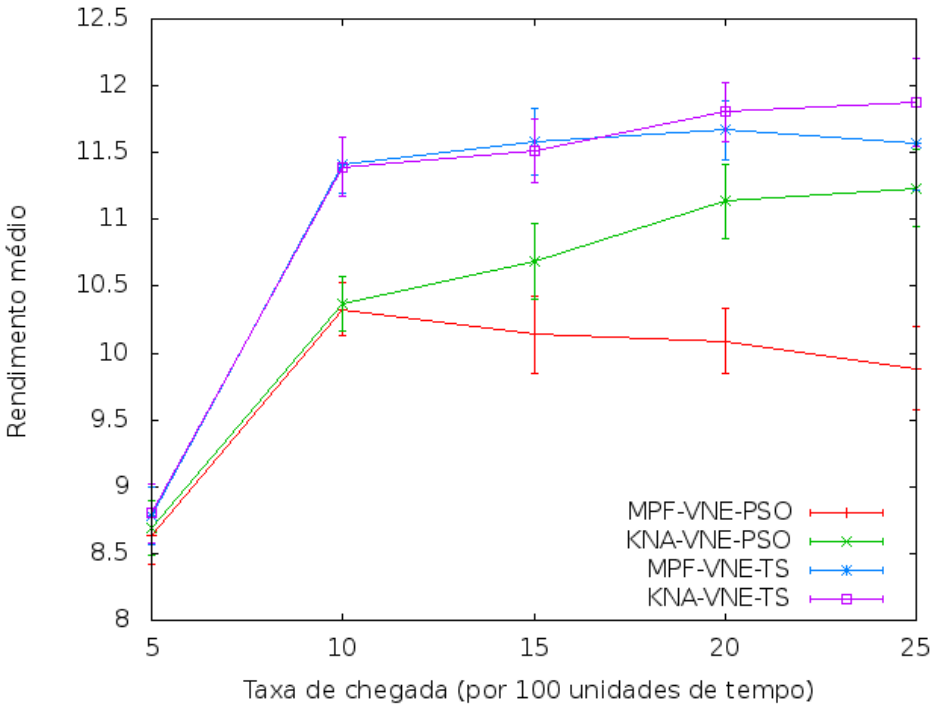


Figura 5.4: Ganho médio a longo prazo

Custo médio a longo prazo

O gráfico da Figura 5.5 mostra o custo médio a longo prazo, obtido ao término de 40000 instantes de tempo de execução. Percebe-se que as linhas relacionadas ao VNE-PSO estão praticamente sobrepostas. O mesmo acontece com as linhas relacionadas ao VNE-TS. Isso indica que os algoritmos de seleção não tiveram grande impacto no decréscimo do custo médio obtido por ambos os algoritmos.

Contudo, percebe-se no gráfico da Figura 5.5 que o algoritmo VNE-TS sempre alcançou custo de mapeamento menor do que o VNE-PSO em quase todas as taxas de chegada. Apenas quando a taxa de chegada é de 25 requisições por 100 unidades de tempo, a combinação KNA-VNE-PSO supera a combinação MPF-VNE-TS. Mais uma vez, a combinação da política 2ks-VN-*Selector* com o VNE-TS supera todas as demais combinações consideradas, em termos de redução do custo médio a longo prazo.

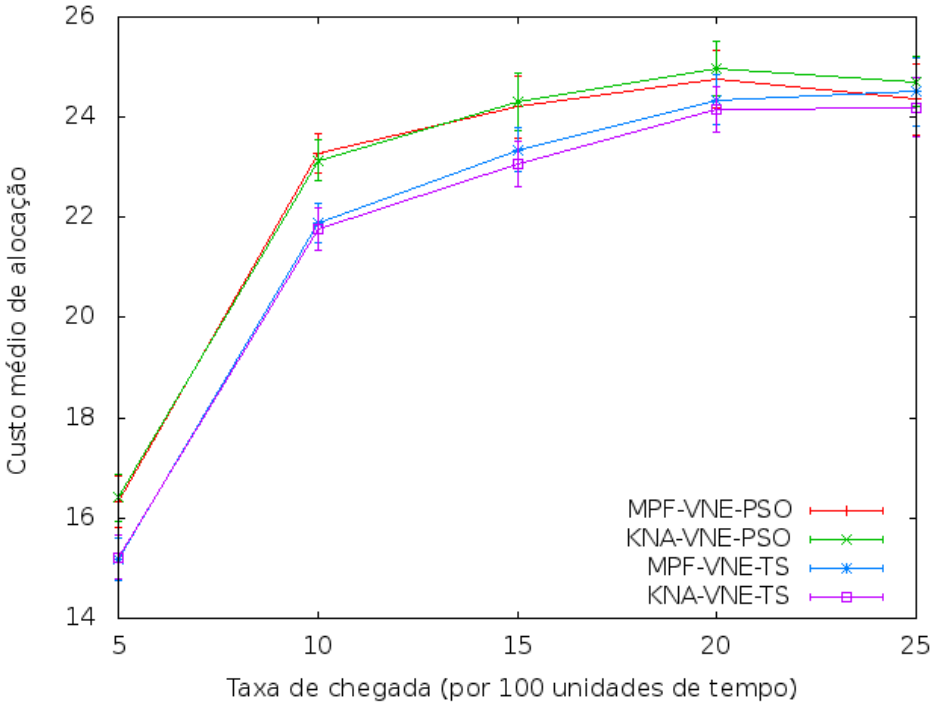


Figura 5.5: Custo médio a longo prazo

Taxa de ganho sobre custo a longo prazo

Dividir o ganho médio pelo custo médio nos dá a ideia da relação de custo benefício proveniente de diferentes políticas de mapeamento e seleção. A Figura 5.6 evidencia que o VNE-TS oferece uma melhor relação custo-benefício nos mapeamentos.

Quando a taxa de chegada é de 5 requisições por 100 unidades de tempo, nota-se que a política de seleção de redes virtuais não tem impacto sobre a relação de ganho sobre custo.

Pode-se perceber ainda que a partir de taxas de chegada maiores do que 10 requisições por 100 unidades de tempo, a combinação do algoritmo de seleção 2ks-VN-*Selector* com o algoritmo de mapeamento VNE-TS, supera todas as demais combinações consideradas.

Percebe-se também que a política 2ks-VN-*Selector* tem maior impacto sobre o algoritmo de mapeamento VNE-PSO, em taxas de chegada superiores a 10 requisições por 100 unidades de tempo.

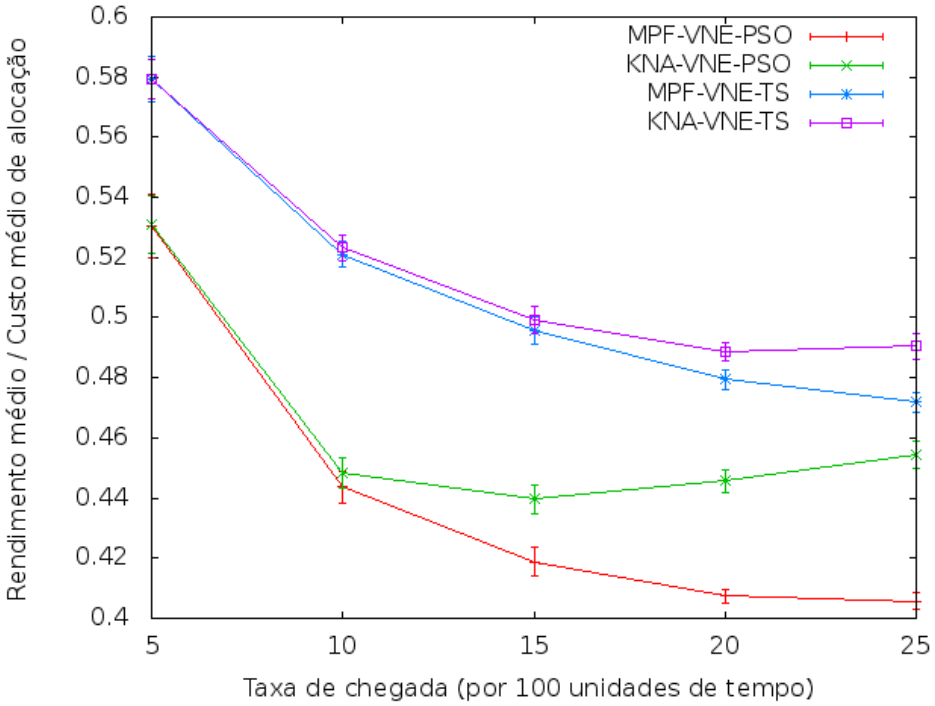


Figura 5.6: Taxa de ganho sobre custo a longo prazo

Análise do 2ks-VN-*Selector*

A Figura 5.7 exibe estatísticas sobre o algoritmo de seleção 2ks-VN-*Selector*, combinado aos algoritmos de mapeamento VNE-PSO e VNE-TS.

Conforme esperado, o total de requisições recebidas aumenta com o aumento da taxa de chegada. As barras vermelhas nos gráficos da Figura 5.7 mostram esse aumento, para ambos os algoritmos de mapeamento considerados.

O número de requisições selecionadas pelo 2ks-VN-*Selector* também aumenta com a taxa de chegada, seja quando combinado ao VNE-TS ou quando combinado ao VNE-PSO, conforme determina a barra verde dos dois gráficos.

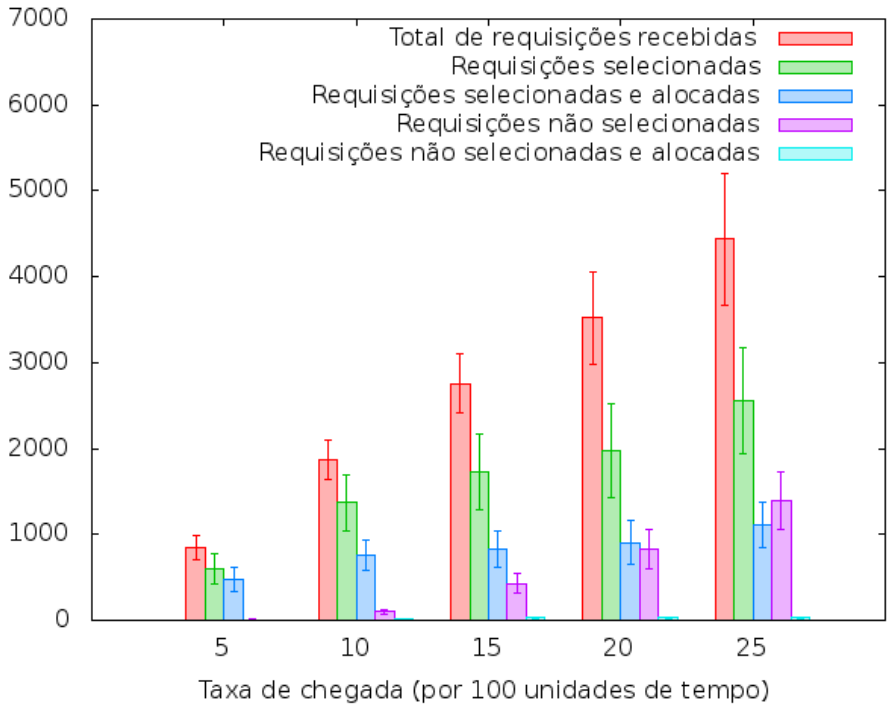
As barras azuis mostram a quantidade de requisições que foram selecionadas e efetivamente mapeadas. Percebe-se um pequeno aumento ao longo das várias taxas de chegada, com uma tendência de estabilização a partir de taxas de chegada maiores que 10 requisições por 100 unidades de tempo. Essa estabilização se justifica devido aos recursos serem limitados. Assim, o número de redes mapeadas não pode crescer sempre com a taxa de chegada.

Com o aumento da taxa de chegada, haverá mais requisições por janela de tempo w. Devido à escassez de recursos, muitas dessas requisições não serão selecionadas pelo 2ks-VN-*Selector*. Esse fato é mostrado pelo crescimento das barras roxas nos gráficos da

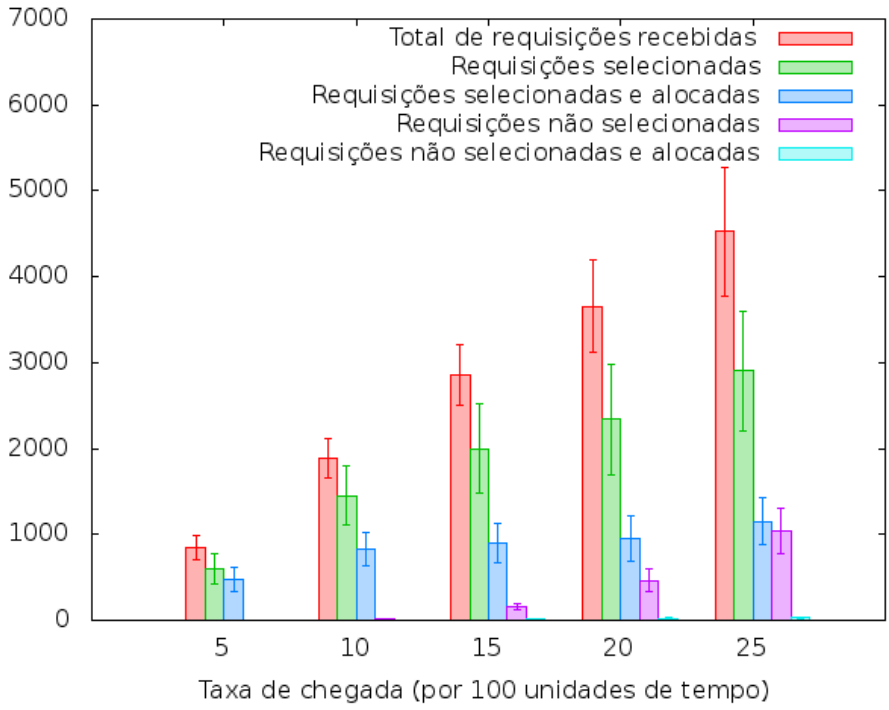
Figura 5.7.

Em cenários com taxa de chegada baixa, ambos os algoritmos de mapeamento alocam a grande maioria das requisições selecionadas pelo 2ks-VN-*Selector*. Em todas as taxas de chegada, os algoritmos VNE-TS e VNE-PSO alocaram poucas requisições não selecionadas pelo 2ks-VN-*Selector*, conforme mostram as barras ciano nos gráficos da Figura 5.7. Assim, conclui-se que o valor definido para o parâmetro η corresponde a uma boa estimativa.

Se o valor de η fosse muito grande, um grupo de redes virtuais não seria selecionado de maneira errônea. Redes virtuais desse mesmo grupo poderiam ser mapeadas depois, no laço da linha 7, no Algoritmo 8, que exibe o procedimento *2ks-VN-Selector*. Esse mapeamento secundário aumentaria o tamanho das barras ciano no gráfico da Figura 5.7, o que não é observado. Caso η fosse uma estimativa muito inferior à realidade, muitas redes virtuais seriam errôneamente selecionadas e não seriam alocadas pelos algoritmos de mapeamento, causando uma diminuição nas barras azuis e possível aumento nas barras ciano da Figura 5.7.



(a) VNE-PSO



(b) VNE-TS

Figura 5.7: Desempenho do 2ks-VN-*Selector* (com restrição de localização)

5.4 Experimentos em cenário sem restrição de localização

Esta seção apresenta os resultados obtidos ao desconsiderar a restrição de localização. Assim, nesta seção o algoritmo VNE-TS-*Clustering* é aplicado e comparado aos algoritmos VNE-TS e VNE-PSO.

5.4.1 Análise de evolução do custo

A Figura 5.8 mostra a evolução dos custos obtidos pelos algoritmos VNE-TS, VNE-TS-*Clustering* e VNE-PSO, ao longo de 100 iterações. Cada gráfico desta seção mostra o resultado ao considerar redes virtuais com tamanhos distintos, de 2, 4, 6, 8 e 10 nós.

Os valores numéricos no eixo y dos gráficos da Figura 5.8 são valores escalares obtidos pela definição de custo, introduzida na Seção 3.4. Quanto maior o custo, pior o mapeamento. A ocorrência de sobreposição de intervalos de confiança indica que determinado algoritmo pode superar os demais em alguns mapeamentos, mesmo que a média obtida em 30 execuções seja pior.

A Figura 5.8a mostra que, para redes com dois nós virtuais, o VNE-TS começou com custo médio de mapeamento próximo a 28. O algoritmo VNE-PSO iniciou com custo médio próximo a 21. O VNE-TS-*Clustering* iniciou com mapeamentos de menor custo médio (19). Isso ocorre devido ao uso do algoritmo *mapeamentoAleatórioComKmeans*, que tende a mapear todos os nós virtuais de uma mesma rede virtual para um *cluster* de nós físicos próximos. A partir da vigésima iteração, os três algoritmos de mapeamento considerados alcançaram um mapeamento com o mesmo custo médio, permanecendo com mesmo valor ao longo de 80 iterações. Assim, para redes virtuais com 2 nós, 100 iterações causa desperdício de recursos, segundo os nossos experimentos.

A Figura 5.8b mostra os resultados de 100 iterações, considerando o mapeamento de redes virtuais com 4 nós. Note que as faixas numéricas no eixo y deste gráfico são maiores do que as observadas no gráfico 5.8a. Isso se deve ao fato de que redes virtuais maiores naturalmente demandam mais recursos e têm maior custo de mapeamento. No gráfico 5.8b, mais uma vez o VNE-TS-*Clustering* começou com uma solução melhor do que a do VNE-TS e VNE-PSO. Nota-se que o VNE-PSO obtém o pior custo médio ao longo das 100 iterações. Apesar disso, a sobreposição dos intervalos de confiança indicam que o VNE-PSO pode conseguir mapeamentos com menor custo em algumas instâncias. As linhas relativas ao VNE-TS e VNE-TS-*Clustering* permanecem entrelaçadas ao longo das 100 iterações. Isso indica que o VNE-TS-*Clustering* não traz grandes benefícios quando usado para mapear apenas uma rede virtual em uma rede física com recursos suficientes. O gráfico 5.8b mostra uma diferença absoluta média de 7 unidades de custo entre os

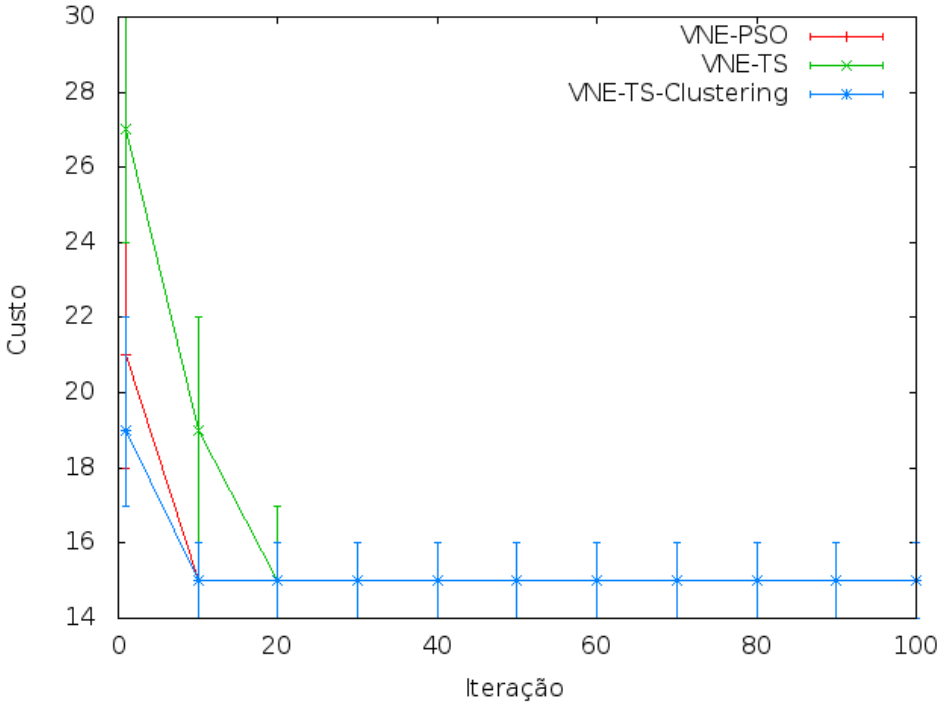
algoritmos baseados na Busca Tabu e o algoritmo VNE-PSO.

Na Figura 5.8c, mais uma vez o VNE-TS-*Clustering* iniciou com mapeamentos com custo médio menor. Nesse gráfico nota-se novamente o aumento na faixa de valores no eixo y, visto que quanto maior o tamanho da rede virtual, mais recursos serão demandados e maior será o custo de mapeamento. O VNE-PSO obteve a pior média de custo de mapeamento, não havendo sobreposição dos intervalos de confiança. Novamente, as linhas relativas aos algoritmos VNE-TS e VNE-TS-*Clustering* permanecem entrelaçadas ao longo das 100 iterações. A Figura 5.8c mostra uma diferença absoluta de 40 unidades no custo médio, ao final das 100 iterações, ao se comparar os algoritmos baseados na Busca Tabu e o VNE-PSO.

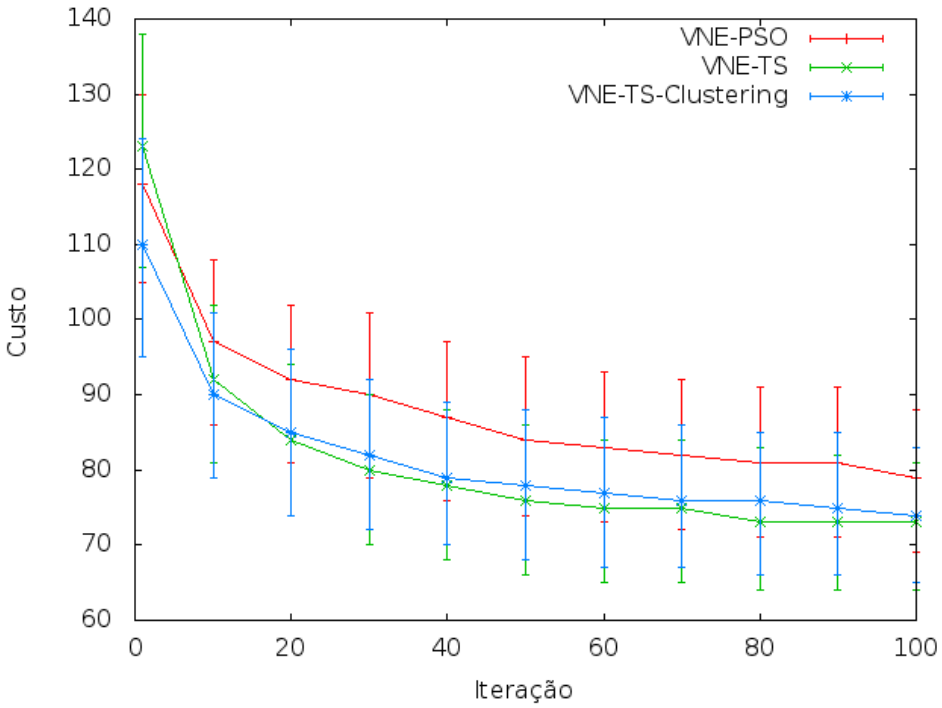
Na Figura 5.8d, o VNE-TS-*Clustering* não obteve diferença significativa no custo médio da solução inicial obtida por meio do procedimento *mapeamentoAleatórioComKmeans*. Os dois algoritmos baseados em Busca Tabu permanecem melhor que o VNE-PSO, em média, ao longo das 100 iterações. O VNE-TS e o VNE-TS-*Clustering* obtiveram custo médio de alocação semelhante, conforme ilustram suas respectivas linhas, que aparecem entrelaçadas no gráfico da Figura 5.8d. Mais uma vez nota-se um aumento natural no custo dos mapeamentos, dado que redes com 8 nós terão em média mais enlaces virtuais do que redes com 6, 4 e 2 nós. Ao término de 100 iterações, a diferença absoluta foi de 100 pontos no custo médio, so se comparar os algoritmos baseados em Busca Tabu com o VNE-PSO.

Por fim, a Figura 5.8e mostra os resultados obtidos ao se considerar redes virtuais com 10 nós. Assim como ocorreu com redes virtuais com 8 nós, os algoritmos VNE-TS e VNE-TS-*Clustering* superaram o VNE-PSO ao longo das 100 iterações, obtendo menores médias de custo de mapeamento. Mais uma vez, nota-se um aumento na diferença absoluta entre os custos obtidos ao término da execução. Os algoritmos baseados em Busca Tabu obtiveram custo médio de mapeamento igual a aproximadamente 300, ao passo que o VNE-PSO alcançou custo médio de mapeamento de aproximadamente 400, resultando numa diferença absoluta de aproximadamente 100 pontos.

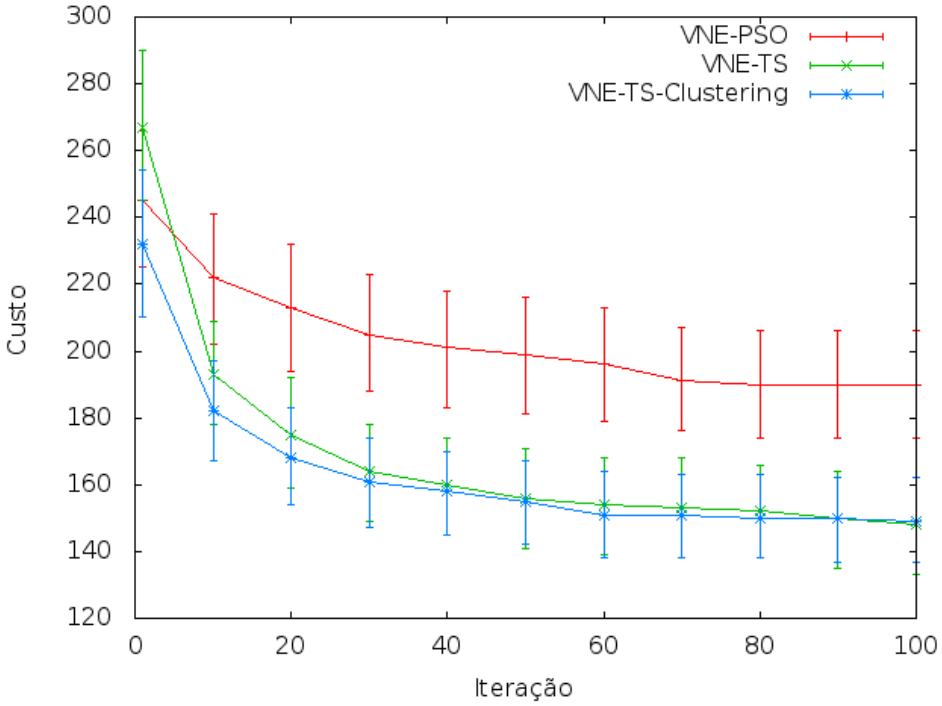
Ao se comparar os gráficos de evolução dos custos em cenário sem localização de restrição, com os mesmos gráficos no cenário que considera essa restrição, nota-se que há uma grande redução no custo médio de mapeamento. Isso pode ser observado através das faixas no eixo y de cada gráfico, nos dois cenários. Por exemplo, no cenário com restrição de localização, o custo médio ao final de 100 iterações obtido pelo VNE-TS no mapeamento de redes virtuais com 10 nós foi igual a 600. Já ao desconsiderar a restrição de localização e redes virtuais de mesmo tamanho, o VNE-TS obteve custo médio aproximado de 500. Isso se deve ao fato de que a restrição de localização reduz consideravelmente o espaço de soluções do problema, de tal forma que uma solução que antes era ótima para o problema sem localização, deixa de ser viável no problema com localização.


VNE-PSO
VNE-TS
VNE-TS-Clustering
Custo
Iteração
30
28
26
24
22
20
18
16
14
0
20
40
60
80
100

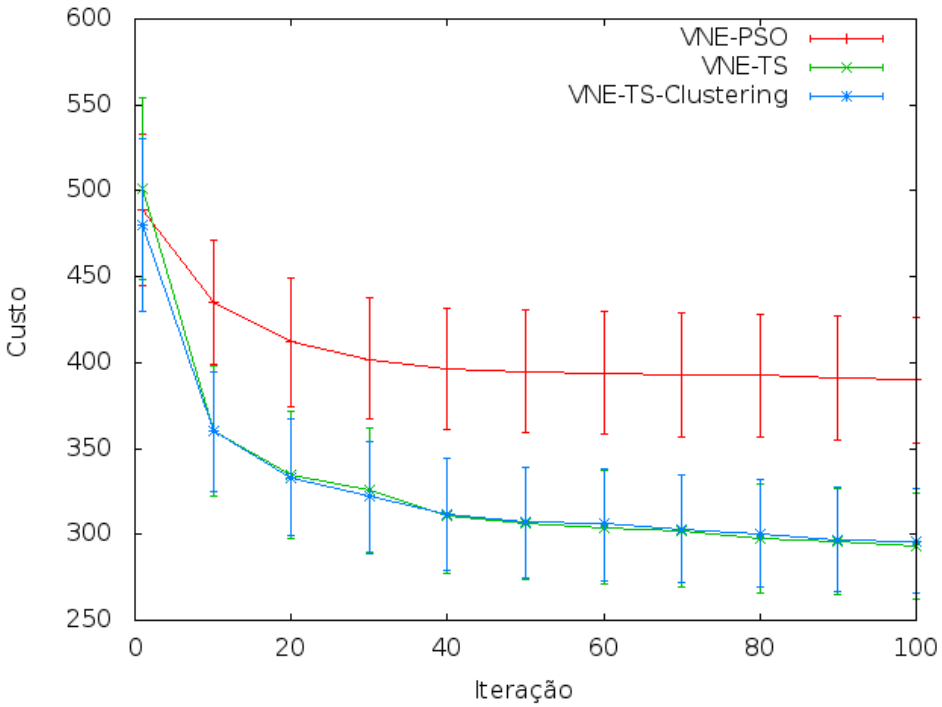
(a) 2 nós


VNE-PSO
VNE-TS
VNE-TS-Clustering
Custo
Iteração
140
130
120
110
100
90
80
70
60
0
20
40
60
80
100

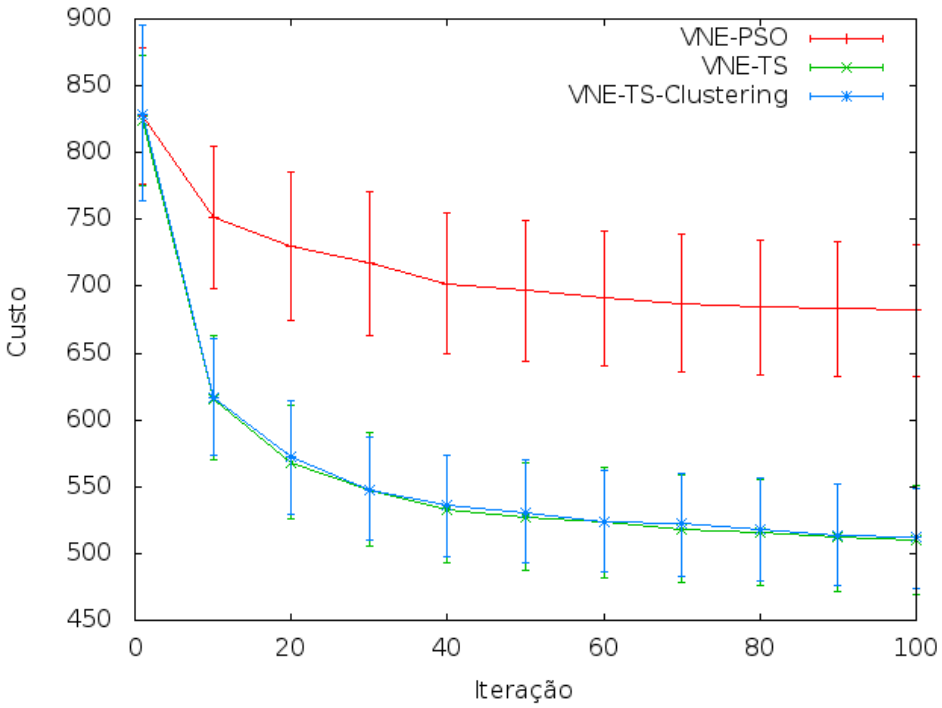
(b) 4 nós


VNE-PSO
VNE-TS
VNE-TS-Clustering
Custo
Iteração
300
280
260
240
220
200
180
160
140
120
0
20
40
60
80
100

(c) 6 nós


VNE-PSO
VNE-TS
VNE-TS-Clustering
Custo
Iteração
600
550
500
450
400
350
300
250
0
20
40
60
80
100

(d) 8 nós



(e) 10 nós

Figura 5.8: Desempenho dos algoritmos de mapeamento com redes virtuais de diferentes tamanhos (sem restrição de localização)

5.4.2 *Performance profiles*

Os resultados obtidos com o *performance profiles* mostram que em aproximadamente 62% das instâncias, ambos VNE-TS e VNE-TS-*Clustering* alcançaram o mapeamento de menor custo, incluindo instâncias com mapeamento de mesmo custo, como mostra a Figura 5.9. Apenas cerca de 22% das instâncias foram mapeadas com menor custo pelo VNE-PSO, incluindo empates. Nota-se que houve uma grande diferença de desempenho entre o VNE-PSO e os algoritmos baseados em Busca Tabu, no cenário sem restrição de localização. Apenas cerca de 22% das 500 redes virtuais mapeadas nesse experimento foram mapeadas com menor custo pelo VNE-PSO. No entanto, os algoritmos VNE-TS e VNE-TS-*Clustering* tiverem desempenhos equivalentes, visto que esses dois algoritmos obtém custo médio de mapeamento semelhantes, conforme observado na seção anterior.

Mais uma vez, mesmo em instâncias para as quais o VNE-PSO obteve melhor mapeamento em comparação ao VNE-TS e VNE-TS-*Clustering*, os custos obtidos pelos algoritmos baseados na Busca Tabu não foram muito piores, em termos do custo de mapeamento, do que os obtidos pelo algoritmo baseado em *Particle Swarm*, pois a inclinação

das curvas dos algoritmos baseados em Busca Tabu é pequena após o ponto 1.1 no eixo x, indicando que poucas instâncias foram solucionadas com razão maior que esse valor. Isto é, apenas em poucas instâncias os algoritmos baseados em Busca Tabu encontram soluções muito piores que o algoritmo de mapeamento baseado em Nuvem de Partículas.

Pode-se concluir mais uma vez, com base nos gráficos da Figura 5.8, que a maioria dos empates nos custos de mapeamento ocorreram em requisições de redes virtuais com menos nós. Esses empates ocorrem quando os algoritmos terminam as 100 iterações com um mapeamento de mesmo custo.

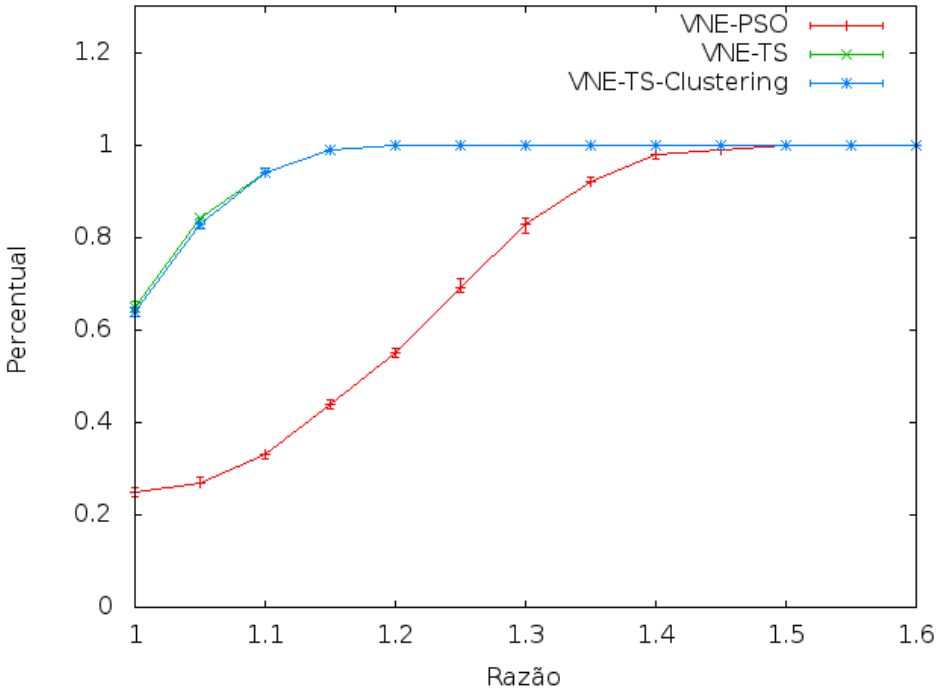


Figura 5.9: *Performance profiles* (sem restrição de localização)

5.4.3 Simulações de rede

Taxa de bloqueio de requisições

O gráfico mostrado na Figura 5.10 mostra que, com maiores taxas de chegada, o VNE-TS bloqueia menos requisições do que o VNE-PSO e do que o VNE-TS-*Clustering*, independentemente do algoritmo de seleção utilizado.

A Figura 5.10 mostra ainda que, principalmente para as taxas de chegada $\lambda = 15, 20$ e 25, o algoritmo de seleção 2ks-VN-*Selector* pode diminuir a taxa de bloqueio quando comparado ao *Most Prize First*, especialmente no caso do algoritmo VNE-PSO.

Nota-se na Figura 5.10 que, para taxas de chegada maiores ou iguais a 15 requisições por 100 unidades de tempo, o VNE-TS-*Clustering* obteve a maior taxa de bloqueio. O motivo desse maior bloqueio de requisições é que o algoritmo de mapeamento baseado no *kmeans* tenta construir uma solução inicial alocando todos os nós para uma pequena região do substrato físico. Essa estratégia aumenta a chance de que todos os nós da solução inicial não possam ser mapeados, causando mais rejeições. Contudo, apesar de causar aumento na taxa de bloqueio, o uso do VNE-TS-*Clustering* é justificável devido à grande redução no custo de mapeamento e no rendimento razoável obtido, conforme será reportado na análise das métricas seguintes.

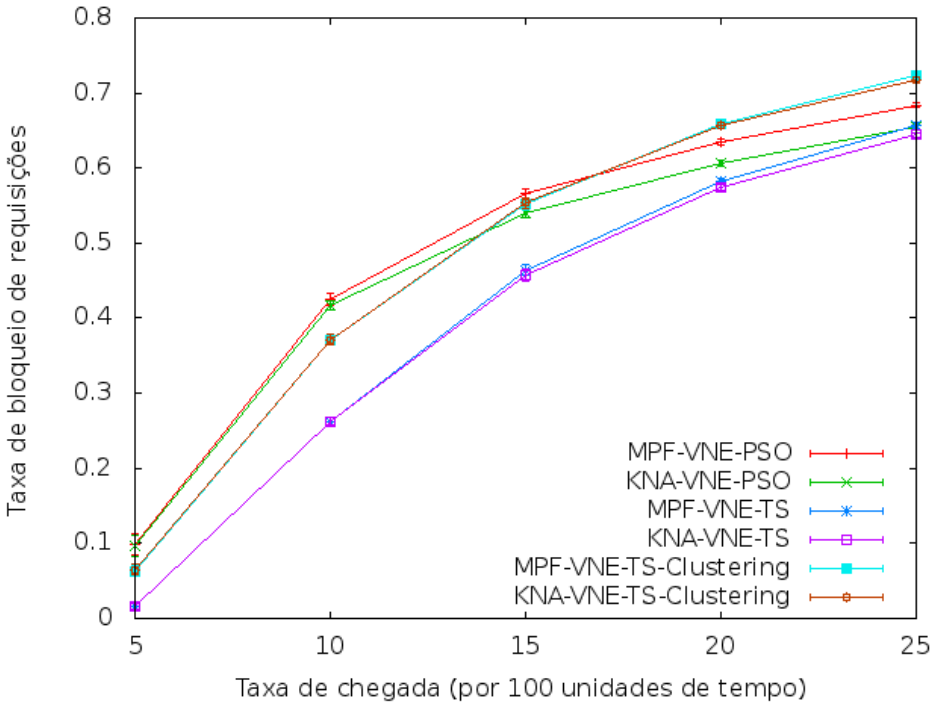


Figura 5.10: Taxa de bloqueio

Ganho médio a longo prazo

Como comentado anteriormente, a diminuição da taxa de bloqueio causa aumento do ganho. A Figura 5.11 mostra que o VNE-TS também superou o VNE-PSO, obtendo maior ganho médio a longo prazo. Para as taxas de chegada $\lambda = 15, 20$ e 25, a utilização do algoritmo de seleção 2ks-VN-*Selector* ocasionou aumento significativo de rendimento no algoritmo VNE-PSO. Mais uma vez, o VNE-TS superou o VNE-PSO, independentemente do algoritmo de seleção utilizado.

O ganho médio a longo prazo obtido pelo VNE-TS-*Clustering* ficou próximo ao obtido

pelo VNE-TS. A pequena diferença se deve ao maior bloqueio causado pelo VNE-TS-*Clustering*. Entretanto, mesmo com maior taxa de bloqueio, o VNE-TS-*Clustering* obteve melhor ganho em relação ao VNE-PSO. Isso acontece porque o VNE-TS-*Clustering*, combinado às duas estratégias de seleção, rejeita soluções com ganhos pequenos.

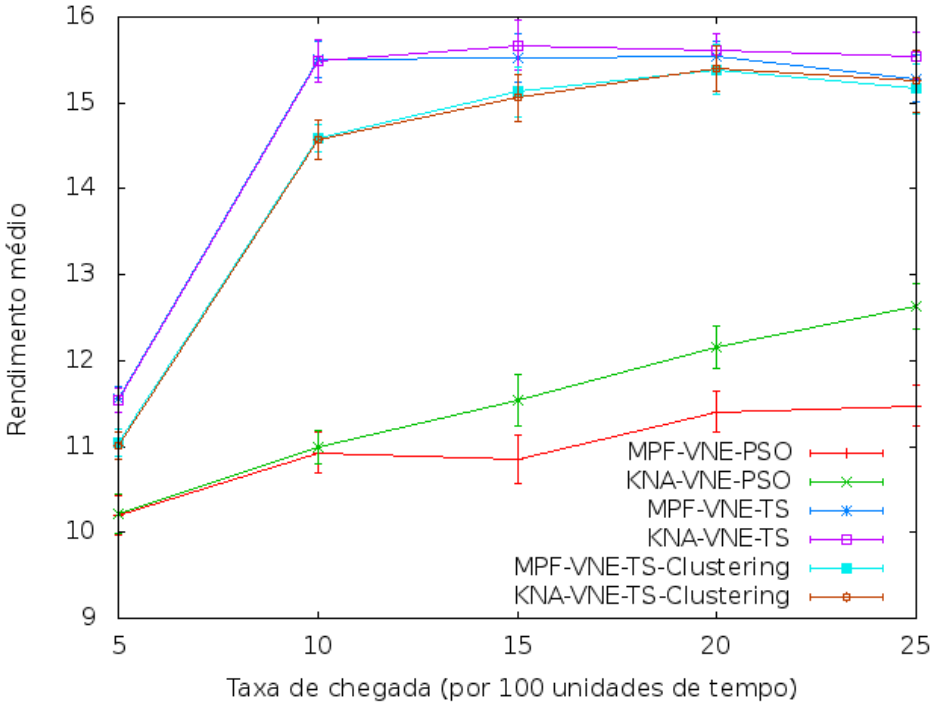


Figura 5.11: Ganho médio a longo prazo

Custo médio a longo prazo

A Figura 5.12 mostra que, no cenário sem localização, os custos médios a longo prazo obtidos pelos algoritmos VNE-TS e pelo VNE-PSO foram semelhantes sob taxas de chegada maiores.

O VNE-TS-*Clustering* obteve menor custo a longo prazo quando comparado ao VNE-TS e VNE-PSO. Esse resultado é esperado, visto que o VNE-TS-*Clustering* mapeia todos os nós virtuais para uma pequena região da rede física, diminuindo a distância entre os nós e, consequentemente, o custo de mapeamento, em troca de um pequeno aumento na taxa de bloqueio.

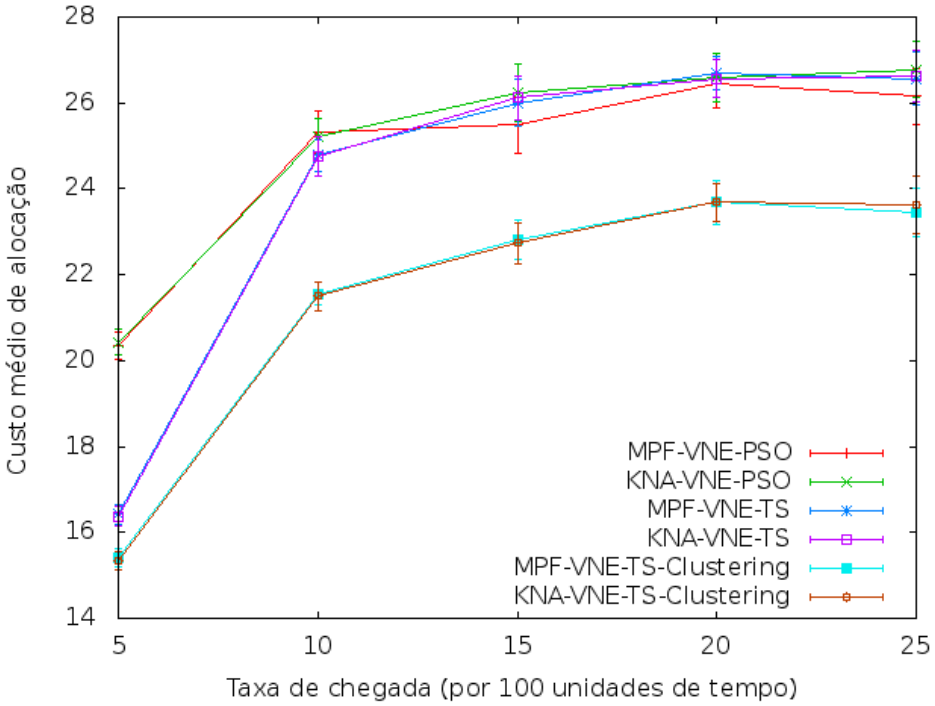


Figura 5.12: Custo médio a longo prazo

Taxa de ganho sobre custo a longo prazo

O VNE-TS-*Clustering* consegue diminuir substancialmente o custo médio de mapeamento, ao mesmo tempo em que obtém níveis razoáveis de rendimento. A combinação desses dois resultados torna o VNE-TS-*Clustering* o algoritmo de mapeamento com a melhor relação de custo-benefício, conforme mostra a Figura 5.13. Pode-se concluir ainda que a combinação *Most Prize First* e VNE-PSO acarretou no menor custo benefício. Além disso, o 2ks-VN-*Selector* causou melhora do custo benefício nos algoritmos VNE-TS e VNE-PSO.

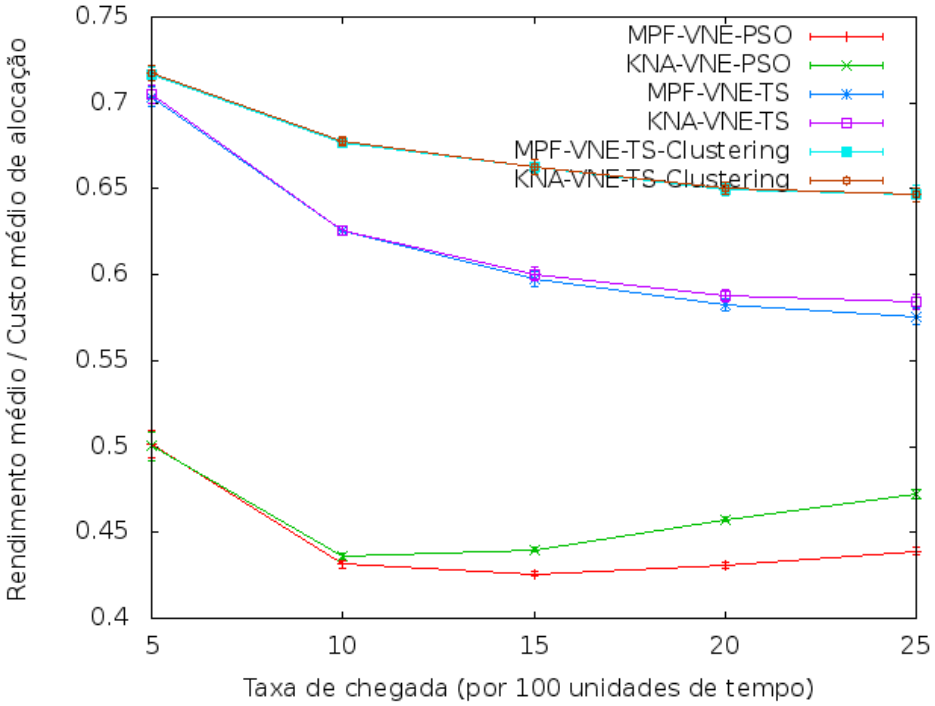


Figura 5.13: Taxa de ganho sobre custo a longo prazo

Análise do 2ks-VN-*Selector*

A Figura 5.14 exibe estatísticas sobre o algoritmo de seleção 2ks-VN-*Selector*, combinado aos algoritmos de mapeamento VNE-PSO, VNE-TS e VNE-TS-*Clustering*. Em cenários com taxa de chegada baixa, os três algoritmos de mapeamento alocam a grande maioria das requisições selecionadas. Em todas as taxas de chegada, os três algoritmos alocaram poucas requisições não selecionadas pelo 2ks-VN-*Selector*, o que levou a conclusão de que o valor definido para o parâmetro η corresponde a uma boa estimativa, pelo motivos discutidos na Seção 5.3.3, inclusive no cenário sem restrição de localidade.

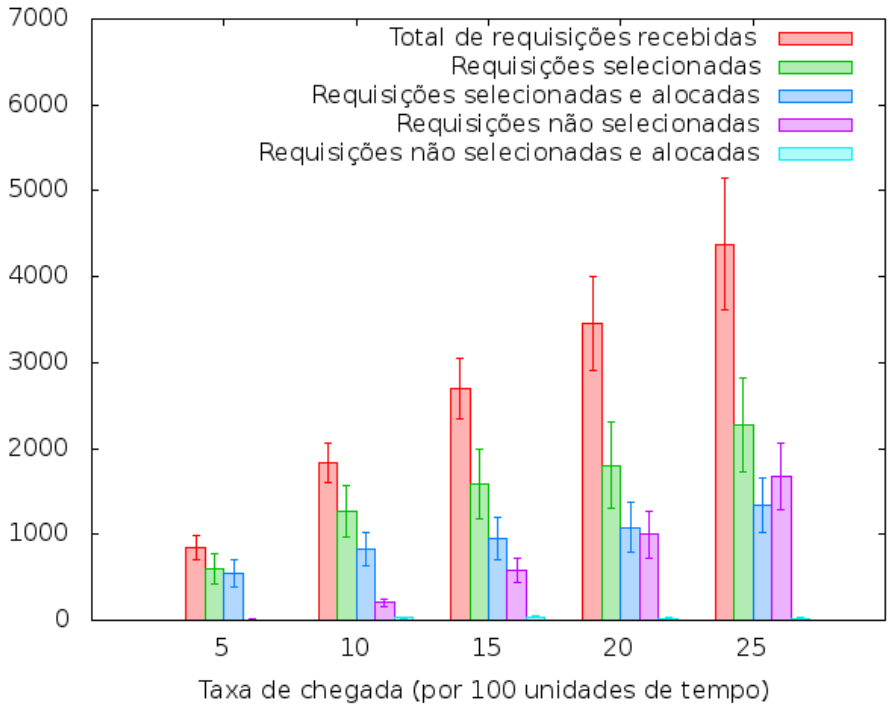
O total de requisições recebidas aumenta com o aumento da taxa de chegada. As barras vermelhas nos gráficos da Figura 5.14 corroboram essa observação, para os três algoritmos de mapeamento considerados no cenário sem restrição de localização.

O número de requisições selecionadas pelo 2ks-VN-*Selector* também aumenta com a taxa de chegada, qualquer que seja o algoritmo de mapeamento combinado, conforme determina a barra verde dos três gráficos da Figura 5.14.

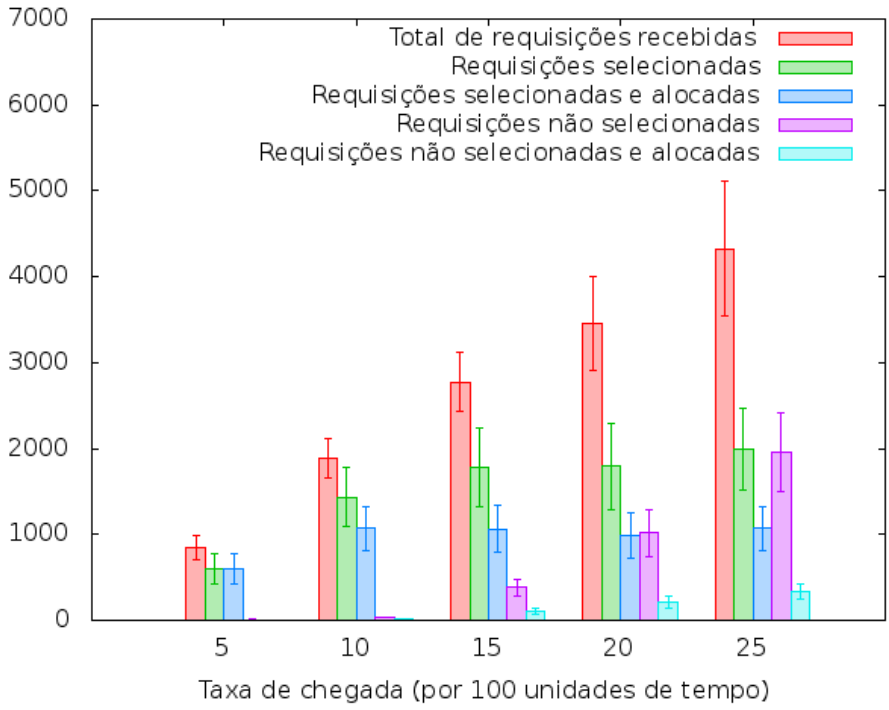
As barras azuis mostram a quantidade de requisições que foram selecionadas e efetivamente mapeadas. Percebe-se um pequeno aumento ao longo das várias taxas de chegada, com uma tendência de estabilização a partir de taxas de chegada maiores que 10 requisições por 100 unidades de tempo. Essa estabilização se justifica devido aos recursos

serem limitados. Assim, o número de redes mapeadas não pode crescer sempre com a taxa de chegada. Esse comportamento já foi observado no cenário com localização de restrição, discutido na Seção 5.3.3.

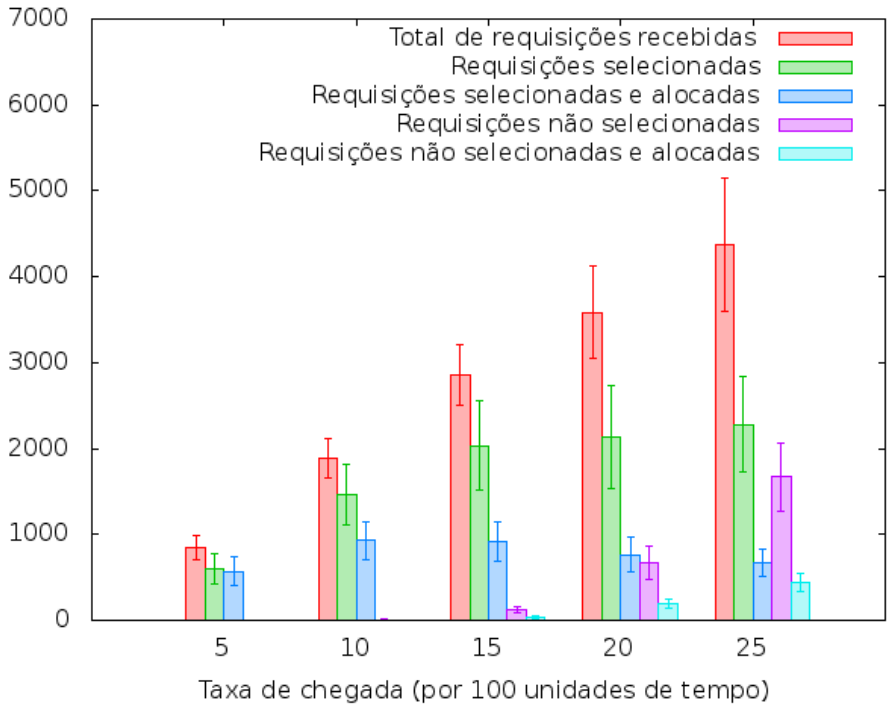
Com o aumento da taxa de chegada, haverá mais requisições por janela de tempo w. Devido à escassez de recursos, muitas dessas requisições não serão selecionadas pelo 2ks-VN-*Selector*. Esse fato é mostrado pelo crescimento das barras roxas nos gráficos da Figura 5.14, junto ao crescimento da taxa de chegada.



(a) VNE-PSO



(b) VNE-TS



(c) VNE-TS-*Clustering*

Figura 5.14: Desempenho do 2ks-VN-*Selector* (sem restrição de localização)

Capítulo 6

Conclusões

O Problema do Mapeamento de Redes Virtuais é um problema chave para superação das limitações arquiteturais relacionadas ao modelo corrente da Internet. Devido à sua complexidade, este problema tem sido abordado por vários pesquisadores nos últimos anos, através de diversas abordagens. O VNE-PSO é um algoritmo do estado-da-arte no Problema do Mapeamento de Redes Virtuais, que possui resultados expressivos quando comparado a algoritmos prévios.

Um ponto falho em alguns algoritmos projetados até então é a realização dos mapeamentos imediatamente após a chegada de uma requisição, dando margem a escolhas ruins, no que tange à relação de custo-benefício a longo prazo. Por outro lado, outros trabalhos agrupam uma certa quantidade de redes virtuais em detrimento da alocação imediata. As redes virtuais agrupadas em uma janela de tempo w são ordenadas em ordem não crescente de ganho. Apesar dessa política de seleção oferecer bons resultados para baixas taxas de chegada de requisição, seus resultados são comprometidos em cenários com frequência de chegada mais intensa. Isto foi comprovado pelos experimentos realizados com o nosso algoritmo de seleção 2ks-VN-*Selector*, baseado no Problema da Mochila Bidimensional 0-1, o qual obteve melhor desempenho quando comparado ao algoritmo de seleção *Most-prize-first*, principalmente em cenários com taxas de chegada altas.

Os resultados obtidos com os algoritmos de mapeamento VNE-TS e VNE-TS-*Clustering* são muito expressivos. Apesar de o VNE-PSO ser um algoritmo do estado da arte no Problema do Mapeamento de Redes Virtuais, os algoritmos VNE-TS e VNE-TS-*Clustering* o superaram em todas os experimentos e métricas analisadas, obtendo maior taxa de aceitação, maior ganho e melhor relação ganho/custo, em diversos cenários com taxas de chegada distintas.

Referências Bibliográficas

[1] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, pp. 862–876, Apr. 2010.

[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, pp. 34–41, Apr. 2005.

[3] B. E. Carpenter and D. D. Kandlur, "Diversifying internet delivery," *IEEE Spectr.*, vol. 36, pp. 57–61, Nov. 1999.

[4] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 61–64, Jan. 2007.

[5] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Comm. Mag.*, vol. 47, pp. 20–26, July 2009.

[6] S. Savage, T. A. A. Aggarawl, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: a case for informed internet routing and transport," *IEEE Micro*, vol. 19, pp. 50–59, 1999.

[7] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," 2001.

[8] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "Overqos: an overlay based architecture for enhancing internet qos," in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, (Berkeley, CA, USA), pp. 6–6, USENIX Association, 2004.

[9] A. D. Keromytis, V. Misra, and D. Rubenstein, "Sos: secure overlay services," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 61–72, Aug. 2002.

[10] D. G. Andersen, "Mayday: distributed filtering for internet services," in *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, USITS'03, (Berkeley, CA, USA), pp. 3–3, USENIX Association, 2003.

[11] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," 1996.

[12] S. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pp. 426–436, 1997.

[13] S. Luke, *Essentials of Metaheuristics*. Lulu, second ed., 2013. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[16] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948 vol.4, 1995.

[18] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, pp. 533–549, May 1986.

[19] E. K. Burke and G. Kendall, eds., *Search methodologies : introductory tutorials in optimization and decision support techniques*. New York: Springer, 2005.

[20] P. Bajpai and S. Singh, "Fuzzy adaptive particle swarm optimization for bidding strategy in uniform price spot market," *Power Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 2152–2160, 2007.

[21] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *International Journal of Communication Systems*, pp. n/a–n/a, 2012.

[22] R. Elmasri, S. Navathe, M. Pinheiro, C. Canhette, G. Melo, C. Amadeu, and R. de Oliveira Morais, *Sistemas de banco de dados*. Pearson Addison Wesley, 2005.

[23] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1 –12, april 2006.

[24] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 17–29, Mar. 2008.

[25] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM 2009, IEEE*, pp. 783–791, 2009.

[26] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1 –6, june 2011.

[27] T. Stützle and H. H. Hoos, "Max-min ant system," *Future Gener. Comput. Syst.*, vol. 16, pp. 889–914, June 2000.

[28] S. Masti and S. Raghavan, "Vna: An enhanced algorithm for virtual network embedding," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, 2012.

[29] G. Alkmim, D. Batista, and N. da Fonseca, "Mapping virtual networks onto substrate networks," *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 3, 2013.

[30] A. Freville and G. Plateau, "The 0-1 bidimensional knapsack problem: Toward an efficient high-level primitive tool," *Journal of Heuristics*, vol. 2, no. 2, pp. 147–167, 1996.

[31] "Gt-itm." http://www.cc.gatech.edu/projects/gtitm/.

[32] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.