Universidade Federal do Maranhão - UFMA
Centro de Ciências Exatas e Tecnológicas
Bacharelado em Ciência da Computação

Uma heurística baseada em GRASP aplicada ao problema de localização de hubs capacitados: um estudo de caso

Nathasha Araujo Pinto

São Luís - MA

2022

Nathasha Araujo Pinto

Uma heurística baseada em GRASP aplicada ao problema de localização de hubs capacitados: um estudo de caso

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal do Maranhão – UFMA

Centro de Ciências Exatas e Tecnológicas

Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Francisco Glaubos Nunes Clímaco

São Luís - MA

2022

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Pinto, Nathasha.
Uma heurística baseada em GRASP aplicada ao problema de alocação de hubs capacitados: um caso de estudo / Nathasha Pinto. - 2022.
38 f.

Orientador(a): Francisco Glaubos Nunes Clímaco.
Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luis - MA, 2022.

1. GRASP. 2. Heurística. 3. Loggi. 4. Problema de localização de hubs capacitados. I. Nunes Clímaco, Francisco Glaubos. II. Título.

Nathasha Araujo Pinto

Uma heurística baseada em GRASP aplicada ao problema de localização de hubs capacitados: um estudo de caso

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dr. Francisco Glaubos Nunes Clímaco
Universidade Federal do Maranhão
Orientador

Carlos Eduardo Portela Serra de Castro
Universidade Federal do Maranhão
Convidado

Tiago Bonini Borchartt
Universidade Federal do Maranhão
Convidado

São Luís - MA
2022

Agradecimentos

Agradeço a mim por não ter desistido no meio do caminho apesar de todas as frustrações que tive.

Sou grata aos meus pais por terem me criado com foco em crescer e ser alguém de quem se orgulhassem, e mesmo sabendo que teriam que me mandar para outro estado para isso nunca me desmotivaram, obrigada por terem acreditado em mim.

Agradeço aos meus bons amigos: Ana Paula, Mariana, Eduarda, Arthur e Luige, que sempre me motivaram e me apoioaram em inúmeros momentos da minha graduação. Sou muito grata ao meu parceiro José por ter suportado minhas reclamações sobre esse trabalho por meses e meses, você é um anjo.

Aos meus gatos: Amy, Ada e Gordofredo, que em momentos de extremo estresse eles estavam no ladinho fazendo a maior bagunça mas com uma carinha fofa irresistível, obrigada por existirem.

E por ultimo e não menos importante, agradeço ao meu orientador Francisco Glaubos que sempre me apoiou e tirou minhas dúvidas por mais simples que fossem, e ajudou (a ainda ajuda) a ser a profissional que sou hoje.

Obrigada a todos!

"Agora entendo que um dos motivos importantes para ir à universidade e obter uma educação é aprender que as coisas em que você acreditou a vida toda não são verdade, e nada é o que parece ser."

(KEYES, DANIEL; FLORES PARA ALGERNON)

Resumo

Em setores que exigem uma logística exemplar, como por exemplo em uma empresa de entregas, problemas de otimização são comuns. O fluxo de entrada e saída de objetos é muito grande e por isso é necessário se utilizar de métodos que minimizem os custos de transporte. O presente trabalho descreve um estudo de caso utilizando uma heurística baseada em GRASP no problema de localização de hubs capacitados, comparando seu desempenho ao de um algoritmo puramente guloso, buscando assim minimizar o custo total de deslocamento e verificar qual a melhor abordagem de solução. O problema foi adaptado pensando em seu uso real, no qual as instâncias utilizadas para os testes computacionais foram geradas a partir de uma base de dados real.

Palavras-chave: GRASP, heurística, Problema de localização de hubs capacitados, Loggi.

Abstract

In sectors that demands an exemplary logistics, such as a delivery company, optimization problems are common. The input and output flow of objects is very large and therefore it is necessary to use methods that minimize transport costs. This paper describes a study case using a GRASP based heuristic in the capacitated hub location problem, comparing its performance to that of a purely greedy algorithm, thus seeking to minimize the total cost of displacement and verify the best solution approach. The problem was adapted thinking about its real use, in which the instances used for the computational tests were generated from a real database.

Keywords: GRASP, heuristic, Capable hubs location problem, Loggi.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

GRASP	*Greedy Randomized Adaptive Search Procedure*
ILS	*Iterated Local Search*
LC	*Lista de Candidatos*
LCR	*Lista de Candidatos Restrita*
OC	*Otimização Combinatória*
PpH	*Problema de Alocação de Hubs*
PpHC	*Problema de Alocação de p-Hubs Capacitados*
PR	*Path-Relinking*
RVND	*Random Variable Neighborhood Search*
TS	*Tabu Search*
VNS	*Variable Neighborhood Search*

Sumário

1 Introdução

O aumento da eficiência das tecnologias computacionais modernas aliado ao aprimoramento de algoritmos de otimização que resolvem problemas de tomada de decisão, têm ampliado a possibilidade de modelagem matemática de uma maneira ainda não explorada. Nesse contexto, as técnicas de pesquisa operacional tem-se provado um grande instrumento em vários campos da ciência e da tecnologia, com soluções inovadoras e sendo implementadas com sucesso em problemas reais em um tempo computacional aceitável. (MORAES; FERREIRA; SILVA, 2019)

Problemas de Otimização Combinatória (OC) consistem em achar a melhor combinação entre um conjunto de variáveis para maximizar ou minimizar uma função, geralmente chamada de função objetivo ou função custo (BECCENERI, 2008). Esses problemas de otimização no contexto de uma rede de entregas são frequentes, uma vez que o fluxo de entrada e saída de mercadorias é muito grande. Há diversos métodos exatos que em teoria podem ser aplicados para a resolução de tais problemas, mas na prática, ao serem expostos a instâncias de grande porte se tornam complexos e inviáveis.

Para problemas de OC do tipo NP-difícil, nos quais soluções ótimas são computacionalmente difíceis de serem encontradas ou suficientemente grandes para considerar técnicas exatas, são empregadas meta-heurísticas, que são estruturas algorítmicas independentes que fornecem um conjunto de diretrizes ou estratégias para desenvolver algoritmos de otimização e que podem gerar soluções viáveis para problemas reais (GLOVER; KOCHENBERGER, 2006).

No contexto de redes, quando a transmissão de dados entre os pontos não se dá de forma direta, essa rede é chamada de *hub-and-spoke* (AYKIN, 1994), e os problemas existentes nessa área se dão justamente na comunicação entre os pontos não-hubs e os hubs, pois é necessário avaliar qual o melhor meio para que essa transmissão ocorra, considerando um custo mínimo. A partir disso surge o problema de alocação de hubs (PpH), que está inserido na classe NP-difícil (GAREY; JOHNSON, 1979), cujo o objetivo é encontrar a melhor localização dos hubs e alocar pontos não-hubs à eles, minimizando uma função objetivo que descreve o fluxo intercambiado e, consequentemente, seu custo.

1.1 Motivação

O mercado de logística está focado na diminuição do custo e no aumento do lucro sem perder a qualidade do produto ofertado, e isso também se aplica à indústria de entregas. Um fluxo desse tipo de mercado é intenso e na maioria das vezes custoso. Deste modo, a

utilização de Pesquisa Operacional se torna imprescindível, pois ela busca o máximo de desempenho e contabiliza também, o impacto financeiro (MORO et al., 2018).

No contexto de uma empresa de entregas, aplicar um sistema otimizado na sua logística de armazenamento, poderia melhorar o desempenho e garantir seu maior lucro. Para garantir uma boa logística dessas entregas, é necessário verificar quais os melhores pontos de armazenamento das mercadorias. Assim, pode-se fazer uma associação ao problema de p-hubs, no qual muitos autores (MORO et al., 2018; ALKAABNEH; DIABAT; ELHEDHLI, 2019; O'KELLY, 1986) trazem abordagens exatas em busca de uma solução ótima.

Sendo assim, neste trabalho será abordado um estudo de caso sobre o problema de alocação de p-hubs capacitado (PpHC). Esse problema consiste em uma variante do problema formulado por Morton E O'kelly (O'KELLY, 1986) e tem como parâmetros que diferem do principal problema: um número fixo de p hubs, a capacidade máxima dos hubs selecionados, a restrição dos pontos (o ponto só pode se ligar a um hub), e o custo de configuração associado.

No intuito de resolver um problema real, o presente estudo utiliza instâncias criadas a partir de uma base de dados fornecida pela LOGGI, empresa de entregas. E nessas instâncias será aplicada, como método de solução, uma heurística baseada em GRASP (*Greedy Randomized Adaptive Search Procedures*) (FEO; RESENDE, 1995). Com o GRASP será obtida uma solução que é constituída de duas fases; a primeira trata-se da aplicação de um algoritmo guloso para a construção de uma solução, e na segunda é aplicada uma busca local nessa solução, a fim de aprimorá-la.

1.2 Organização do Trabalho

O restante deste trabalho está estruturado da seguinte forma:

- O Capítulo 2 trata da fundamentação teórica das técnicas utilizadas. Nele é abordado o problema de localização de hubs capacitado. Trata também de alguns conceitos relacionados a heurística utilizada.
- O Capítulo 3 apresenta as etapas adotadas que compõem a metodologia proposta para este trabalho. Iniciando pela leitura da instância a qual o problema será aplicado, seguido pela heurística aplicada.
- O Capítulo 4 trata sobre os resultados obtidos e discussões em relação aos experimentos feitos. Nele é exibida uma tabela comparando os resultados da heurística baseada em GRASP, aos resultados da aplicação do mesmo caso a um algoritmo guloso.

- O Capítulo 5 apresenta as considerações finais sobre os resultados e trabalhos futuros.

2 Fundamentação Teórica

2.1 O Problema de Alocação de Hubs (PpH)

Em uma rede em que se necessita conectar inúmeros pontos á um centro e essa conexão pode ser um tanto custosa por conta do fluxo que há. No problema de alocação de hubs busca-se encontrar qual o melhor local de instalação desse "centro", para que o custo do fluxo da conexão hub-nó seja mínimo.

Há diversas variantes para o PpH, tais variantes que consideram inúmeros atributos e classificações como: Domínio de solução, Função objetivo, Determinado número de hubs, capacidade dos hubs, o custo de alocar cada hub, a alocação de não-hubs a hubs e o custo dessa alocação (FARAHANI et al., 2013). Nesse trabalho, o autor busca classificar os tipos de PpH e também analisa algumas das soluções dadas para algumas das classificações feitas.

O problema de alocação de hubs não é um problema recente, tendo sua primeira solução proposta por Morton E O'kelly (O'KELLY, 1987). O problema tem p hubs, sendo essa quantidade pré definida e nesse modelo os hubs não possuem custo para alocação ou uma capacidade máxima. O modelo resultante apresentado tem uma função objetivo quadrática não convexa e restrições lineares, o que apresenta um resultado aceitável para instâncias pequenas. A primeira formulação linear foi desenvolvida por James F Campbell (CAMPBELL, 1992), permitindo o uso de software de programação matemática padrão. Campbell introduziu a primeira formulação com características de alocações múltiplas de hubs.

Em trabalhos mais recentes como o de Lüer-Villagra et al. (LÜER-VILLAGRA; EISELT; MARIANOV, 2019), é sugerido o uso de uma combinação de algoritmo genético com outro modelo matemático aplicado, fazendo uso de até 200 pontos em seus testes. A conclusão obtida é que a meta-heurística aplicada permite a resolução de pequenas e médias instâncias de forma rápida e satisfatória, mas quando aplicada a grandes instâncias, se torna custosa.

Para a resolução do PpH, algumas pesquisas utilizam heurísticas híbridas, como mostra Sangsawang (SANGSAWANG; CHANTA, 2020), que aborda o problema de alocação única de hub capacitado. No estudo é feito o uso de VNS (**Variable Neighborhood Search**) juntamente com TS (**Tabu Search**), que foram aplicados a dados reais. O autor conclui que a combinação dos métodos melhora a performance do algoritmo e é capaz de prover boas soluções.

> Para resolver o modelo não linear resultante, uma abordagem Lagrangiana e um GRASP são propostos. Embora a função objetivo não seja convexa nem côncava, ambas as heurísticas forneceram soluções de alta qualidade dentro de um tempo computacional razoável (ALKAABNEH; DIABAT; ELHEDHLI, 2019).

2.2 O PpH Capacitado

Campbell (1994) propôs um problema no qual as capacidades dos hubs limita o fluxo da rede (CAMPBELL, 1994). O problema de alocação de hubs capacitados (PpHC) é uma variante que diz respeito a limitação de atendimento de cada hub.

No trabalho de Bütün et al. (2021) (BÜTÜN; PETROVIC; MUYLDERMANS, 2021) é usada uma junção de meta-heuristica com programação linear para dar uma solução para alguns problemas de OC sendo o maior foco em alocação de hubs capacitados. O problema foi formulado como um modelo de programação inteira mista com um custo de congestionamento não linear composto na função objetivo. É abordado uma aproximação linear por partes semicontínua para o custo de congestionamento para linearizar a função objetivo e projeta um algoritmo probabilístico de busca tabu. Os testes realizados se mostraram eficientes.

Um estudo realizado por Carvalho (2017) (CARVALHO, 2017) apresenta uma proposta de solução utilizando uma hibridização de meta-heurísticas para o problema que será analisado no atual trabalho, o PpHC. O autor traz um método nomeado ILS-RVND-PR que é um fusão das técnicas *Iterated Local Search* (LOURENÇO; MARTIN; STÜTZLE, 2010), *Random Variable Neighborhood Descent* (SOUZA et al., 2010) e *Path-Relinking* (REEVES, 1997), que mostra que a aplicação foi capaz de encontrar soluções rápidas em pouco tempo.

Como foi observado, muitos autores propõem abordagens distintas buscando encontrar a solução mais razoável para o problema. Deste modo, o presente trabalho propõe o uso da meta-heurística GRASP, para resolver o PpHC a partir de instâncias inéditas baseadas em dados reais, a fim de verificar a sua viabilidade de aplicação na indústria.

2.3 Heurísticas e Meta-Heurísticas

A palavra heurística (do grego heurisken), significa a arte de descobrir novas estratégias para resolver problemas. O prefixo meta, (também um termo grego), passou a significar um maior nível de abstração, no idioma inglês. O termo *metaheuristic* foi introduzido por Henderson et al. (2003) (HENDERSON et al., 2003) e denota uma estratégia de resolver um problema usando níveis mais altos de abstrações para guiar uma busca heurística no espaço de soluções (KAVEH, 2017).

Sendo assim, a meta-heurística traz uma melhora nessas estratégias de solução de problemas de OC, contudo vale ressaltar que meta-heurísticas não são universais. Logo não resolvem todos os problemas de otimização e sua aplicação requer conhecimentos específicos e fortes sobre o assunto onde será aplicada.

> Meta-heurísticas são aplicadas para encontrar respostas a problemas sobre os quais há poucas informações: não se sabe como é a aparência de uma solução ótima, há pouca informação heurística disponível e força-bruta é desconsiderada devido ao espaço de solução ser muito grande. Porém, dada uma solução candidata ao problema, esta pode ser testada e sua qualidade, averiguada (FRANCESQUINI, 2009).

2.4 Meta-Heurística GRASP

GRASP é uma meta-heurística para encontrar soluções aproximadas para problemas de otimização combinatória formulados como:

$$\min f(x) \text{ sujeito a } x \in X \tag{2.1}$$

Onde f (x) é uma função objetivo a ser minimizada e X é uma função discreta no conjunto de soluções viáveis. Desde a introdução do GRASP como meta-heurística, ele têm sido desenvolvido e aplicado em problemáticas de inúmeras áreas (FEO; RESENDE, 1995).

A meta-heuristica GRASP é uma combinação do algoritmo heurístico construtivo do tipo guloso, nomeado no trabalho como fase construtiva, e a heurística de busca através de vizinhança, nomeada como busca local.

Na fase de construção, elementos que podem compor uma solução são inseridos em uma lista de candidatos (LC), que então é ordenada de acordo com a contribuição do elemento na função objetivo. Em seguida, os melhores elementos de LC são escolhidos para compor uma lista de candidatos restrita (LCR). Por fim, escolhe-se, aleatoriamente, um elemento de LCR para compor a solução. Esse processo é repetido até que uma solução viável para o problema seja obtida.

De acordo com Feo e Resende (FEO; RESENDE, 1995), não há garantias que a solução construída seja localmente ótima. Portanto, a partir da solução construída, é realizado um procedimento de refinamento conhecido como Busca Local.

O Algoritmo 1 apresenta um pseudo-código do método GRASP, no qual *Seed* representa uma semente para as componentes aleatórias dentro do método, e *MaxIterations* é a quantidade de iterações que o GRASP irá executar. As listas LC e LCR serão geradas com a função *GreedyRandomizedConstruction*, e delas será gerada uma solução parcial. Já a busca local é feita na função *LocalSolution*, para refinar a solução.

Algorithm 1 Pseudoalgoritmo GRASP

```
1: function GRASP(MaxIterations, Seed)
2:     for k = 0, ..., MaxIterations do
3:         Solution ← GreedyRandomizedConstruction(Seed)
4:         Solution ← LocalSolution(Solution)
5:         UpdateSolution(Solution, BestSolution)
6:     end for
       return BestSolution
7: end function
```

As entradas para o algoritmo GRASP (linha 1), são: o máximo de iterações definidas pelo usuário e o "*Seed*"que se trata da instância de onde será retirada a solução. A primeira solução é adquirida de forma randômica (linha 3) e refinada com uma solução local (linha 4). Por fim é feita uma comparação para verificar se a solução local encontrado é melhor que a solução encontrada anteriormente.

2.4.1 Fase de Construção

A ideia central dessa fase é gerar uma solução de qualidade. Para isso primeiro cria-se uma lista de elementos candidatos utilizando conceitos de busca gulosa. No entanto um algoritmo guloso não garante uma variedade de soluções podendo fazer com que o algorítimo fique preso a "ótimos locais". Inserir aleatoriedade na geração de soluções pode auxiliar na diversidade de tais soluções.

A busca gulosa simples, consiste em ver o elemento que seja melhor para a solução, neste caso, não se leva em consideração o todo. O algoritmo 2 mostra um pseudo-código do que seria uma busca gulosa sem o fator de aleatoriedade.

Algorithm 2 Busca Gulosa Simples

```
1: Entrada: E = {conjunto discreto finito}
2: Saída: Solução S
3: S ← 0
4: while solução não construída do
5:     Para todo c ∈ C computar o valor da função gulosa g(c);
6:     Selecionar o melhor c* ∈ g(c);
7:     Adicionar c* à solução parcial: S ← S∪ c*;
8: end while
9: return S
```

A solução é contruida iterativamente (linhas 3 a 6). A cada iteração, na linha 5 o valor da função gulosa para cada um dos candidatos é calculado, determinando seu grau de adaptação. Na linha 6 ocorre a escolha do melhor candidato. Na linha 7 o candidato é adicionado à solução e, por fim, é retornada a solução completa.

Para garantir a diversidade de soluções é necessário adicionar um parâmetro $\alpha \in [0, 1]$, assim a probabilidade de aumento da qualidade de soluções é maior. Isso ocorre pois o elemento para ser incluído na solução, é escolhido aleatoriamente a partir da LCR, definida a partir do parâmetro α. O Algoritmo 3 mostra um pseudo-código do que seria uma busca gulosa com o fator de aleatoriedade.

Algorithm 3 Busca Gulosa Aleatoria (α)

1: Entrada: $E = \{conjunto\ discreto\ finito\}$
2: Saída: Solução S
3: $S \leftarrow 0$
4: $C \leftarrow 0$
5: **while** solucao não construída **do**
6: Para todo c ∈ C computar o valor da função gulosa $g(c)$;
7: Construir a lista restrita de candidatos LCR com base no parâmetro α;
8: Selecionar aleatoriamente $c* \in LCR(C)$;
9: Adicionar c* a soluçao parcial: $S \leftarrow S \cup c*$;
10: Seja C o conjunto de elementos que podem ser adicionados a S;
11: **end while**
12: **return** S

O que diferencia a busca gulosa aleatória da simples é o parâmetro α. É possível observar no Algoritmo 3 a utilização do α (linha 7), seu intuito é construir a LCR, de maneira a selecionar os melhores elementos a serem inseridos na solução. A partir da construção dessa lista, é escolhido aleatoriamente um dos elementos que a ela pertencem e esse é incorporado à solução.

2.4.2 Fase de Busca Local

A fase de construção não garante uma solução local ótima, portanto, a fim de melhorar a solução aplica-se a busca local para melhorá-la em cada iteração. Isso é feito através de sucessivas repetições entre a solução atual e novas soluções geradas que forem de maior qualidade, e o processo se encerra quando nenhuma das soluções na vizinhança for melhor que a corrente (FEITEIRA, 2021).

As estratégias para verificar a vizinhança são diversas, assim como o critério de parada. O sucesso dessa fase consiste em primeiro lugar, iniciar sobre soluções boas, ou seja, a fase de construção precisa gerar soluções de boa qualidade. O Algoritmo 4 mostra um pseudo-código da busca local.

No Algoritmo 4, a variável x é inicializada com o valor da solução obtida na fase de construção, que é passada como parâmetro. Da linha 2 a 7 ocorre um *loop*, onde na linha 3 é realizada uma verificação com o intuito de descobrir se uma solução vizinha s melhora o valor da função objetivo, caso isso ocorra, a variável x recebe a solução vizinha s. A melhor solução é retornada ao final.

Algorithm 4 Busca Local (*Solucao*)

1: $x \leftarrow$Solucao
2: **while** criterio Parada Nao Atingido **do**
3: **if** existe $s \in N(x)$ tal que f(s) < f(x) **then**
4: $x \leftarrow$s
5: **end if**
6: **end while**
7: **return** x

3 Metodologia

Neste trabalho, propõe-se uma heurística baseada na meta-heurística GRASP proposta por Feo e Resende (FEO; RESENDE, 1995), que é um método iterativo constituído por duas fases: a de construção e a de busca local.

3.1 Leitura da instância

Para a realização dos experimentos computacionais, considerou-se um estudo de caso que se constitui de instâncias geradas a partir de uma base de dados real da empresa Loggi. Os arquivos foram fornecidos pela empresa em formato JSON (LOGGI, 2021). Buscando facilitar a leitura desses dados pelo algorítmo, eles foram estruturados de uma nova forma em arquivos .txt.

A base de dados utilizada contém localizações reais distribuídas em três estados: Pará, Distrito Federal e Rio de Janeiro. Foram extraídos dessa base o número de hubs que cada estado contém, as instâncias que indicavam a coordenada geográfica, e a demanda de cada ponto (cliente). As instâncias geradas variam de tamanho, tendo a menor 50 pontos e a maior 200, parâmetro definido com bases nos trabalhos anteriormente analisados (SANGSAWANG; CHANTA, 2020; CARVALHO, 2017; LÜER-VILLAGRA; EISELT; MARIANOV, 2019).

A partir dos arquivos obtidos foi necessário descobrir as distâncias entre todos os pontos, uma vez que o método de solução proposto necessita dessa informação. Para descobrir a localização real dos pontos e a distância entre eles, utilizou-se a Distance Matrix API fornecida pela Google (GOOGLE, 2021).

Essas distâncias juntamente com a demanda de cada nó, foram escritas em um arquivo `.txt`, organizados da forma como mostra o exemplo na Figura 1, contendo a quantidade de vértices, a demanda de cada um e por fim a matriz distância criada pela API.

Os hubs considerados no problema possuem uma capacidade de atendimento de demandas, porém tal capacidade não é fornecida pela base de dados da LOGGI, então usou-se uma fórmula matemática para atender com menor custo (sem desperdício de espaço), o tamanho de cada hub. A Fórmula 3.1 se dá pela soma da capacidade de todos os pontos (c) dividida pela multiplicação da quantidade de hubs (h) e de um valor x que representa a capacidade que será definida ao hub. O resultado da operação é a taxa de dificuldade (TX) que o algoritmo pode obter. Essa taxa de dificuldade é dada com valores entre 0 e 1, e quão mais próximo de 1 o valor obtido pela fórmula for, mais difícil

```
5 //Quantidade de nós//
9
2
8
8
1
1000.0          18.4          18.6          17.3          22.7
18.4            1000.0        0.2           7.7           40.7
18.6            0.2           1000.0        7.9           40.9
17.3            7.7           7.9           1000.0        33.5
23.3            41.2          41.4          34.1          1000.0
```

Figura 1 – Exemplo de instância com cinco pontos de entrega. Fonte: Própria do autor.

a resolução do problema será. Para o cálculo de todas as capacidades das instâncias utilizadas, assumiu-se $TX = 0,9$.

$$\frac{c}{TX * h} = x \tag{3.1}$$

As capacidades dos hubs foram adicionadas manualmente ao algoritmo, alterando-se de acordo com a variação dos tamanhos das instâncias: 50, 100, 150, 200. A Figura 2 ilustra uma solução para uma instância do PpHC, na qual são mostrados os hubs e os pontos ligados a eles juntamente com o custo total.

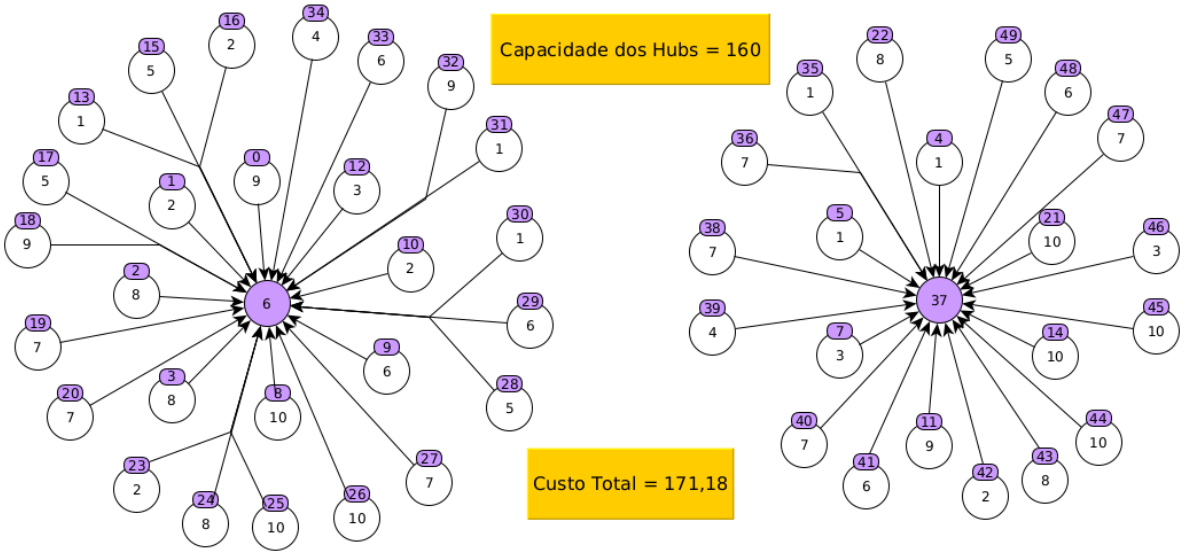

Figura 2 – Solução gerada a partir da instância de 50 pontos do estado do Pará. Fonte: Própria do autor

A Figura 2 mostra a ilustração de uma solução gerada a partir da instância de menor tamanho do estado do Pará. Essa instância possui dois hubs, de valores 6 e 37, posicionados no centro de cada rede. O id de cada hub está ilustrado na cor roxa logo acima do circulo de fundo branco que exibe a capacidade daquele hub em questão. A capacidade dos hubs e o custo total da solução também é exibido.

3.2 GRASP

Na fase de construção, elementos que podem compor uma solução são inseridos na LC, que então é ordenada de acordo com a contribuição do elemento na função objetivo. Em seguida, os melhores elementos de LC são escolhidos para compor a LCR. Por fim, escolhe-se, aleatoriamente, um elemento de LCR para compor a solução. Esse processo é repetido até que uma solução viável para o problema seja obtida.

De acordo com Feo Resende (1995)(FEO; RESENDE, 1995), não há garantias que a solução construída seja localmente ótima. Portanto, a partir da solução construída, é realizado um procedimento de refinamento conhecido como Busca Local.

3.2.1 Fase de Construção

A fase de construção desenvolvida neste trabalho, difere da construção tradicionalmente feita com o GRASP. Na nossa proposta, inicialmente são selecionados p pontos da rede para serem hubs, em seguida, são realizadas ligações de pontos não-hubs (V) a hubs (H), de forma que minimize o custo da solução, que se define pela soma total do custo de todas as ligações entre hubs e não-hubs. Por fim, essa fase retorna uma solução $S(H, V)$ contendo associações entre hubs e não-hubs. O Algoritmo 5 mostra com mais detalhes como são realizadas essas associações.

Algorithm 5 Algoritmo que liga os vértices aos hubs

1: A partir de V escolher aleatoriamente p hubs para compor H
2: $\bar{V} \leftarrow V$
3: **while** $\bar{V} \neq \emptyset$ **do**
4: escolher o $v \in \bar{V}$ mais próximo a um hub $h \in H$
5: $v.hub = h$ ▷ Associando v ao hub h, caso a capacidade de h seja respeitada
6: Atualizar a capacidade do hub h
7: $\bar{V} \leftarrow \bar{V} \setminus v$
8: **end while**
9: **return** S

O algoritmo inicia no primeiro ponto do conjunto e verifica qual o hub mais próximo a ele (linha 4). Na linha 5 é verificado se o hub escolhido tem capacidade para atender o vértice que será associado, caso atenda, o vértice é ligado e o hub tem sua capacidade atualizada.

Os parâmetros de decisão para escolha do hub são: proximidade e capacidade. Uma vez verificado qual o mais próximo, precisa-se conferir se a demanda do vértice é suportada pela capacidade do hub. Caso o hub atual não tenha capacidade para atender o vértice o algoritmo itera e faz uma tentativa de associá-lo a outro hub. Essa etapa gera uma LC que será usada na próxima fase em busca de um aprimoramento nas soluções obtidas.

3.2.2 Fase da busca local

O pseudo-código de busca local aplicado a partir de uma solução construída S é representado pelo Algoritmo 6. Para cada vértice não-hub é feita uma tentativa de troca de associação com um diferente hub, e caso essa troca resulte em uma solução de menor custo, essa nova associação é estabelecida, e as capacidades dos hubs envolvidos na troca são atualizadas.

Algorithm 6 Busca_Local(S)

```
1:  Hubs[]
2:  melhor_custo = custo(S)
3:  melhor_solucao = S
4:  S' = S
5:  for cada v ∈ V' do
6:      for cada h ∈ H' do
7:          if h = v.hub then
8:              Continue                    ▷ Nada é feito, e o próximo v é considerado
9:          end if
10:         v.hub = h                       ▷ associando vértice v a um outro hub
11:         Atualizar a solução S'          ▷ atualiza a solução caso a troca seja viável
12:         if custo(S') < custo(S) then
13:             melhor_custo = custo(S')
14:             melhor_solucao = S'
15:         else
16:             desfazer a associação
17:         end if
18:     end for
19: end for
20: return melhor_solucao
```

Para este trabalho a busca local ocorreu da seguinte forma, a lista de candidatos a possível solução foi gerada anteriormente e usada para verificar se trocando alguns valores com o vizinho mais próximo, alguma solução melhorava seu custo.

4 Resultados computacionais

Nesta seção serão apresentados os resultados obtidos com a aplicação do método proposto, e os seus resultados serão comparados com resultados obtidos a partir do algoritmo guloso 5. Neste trabalho, supomos que em situações reais de tomada de decisão acerca do PpHC, geralmente utiliza-se estratégias gulosas para a alocação de hubs.

Todos os códigos-fonte foram implementados na linguagem de programação Python3, em um computador pessoal Intel® Core™ i3 CPU@2.30GHz com 8GB RAM, utilizando o sistema operacional Linux.

4.1 Resultados numéricos

A Tabela 1 mostra os resultados obtidos dos testes realizados com GRASP e um algoritmo guloso, exceto informações de quais hubs foram escolhidos na execução e quais pontos estão ligados a ele. Na primeira coluna é exibido o tamanho da instância (quantidade de pontos), e nas colunas seguintes, a quantidade de hubs (p), a média de custos, a média de tempo e o melhor custo dentre dez execuções do algoritmo GRASP. Nas colunas seis e sete, são exibidos o custo e o tempo utilizando pelo algoritmo guloso. Cada estado verificado contém quatro tamanhos diferentes de instâncias que variam entre 50 e 200. O tempo de execução é dado em segundos, e o custo em quilômetros, que se trata da soma de todas as distâncias percorridas entre cada hub e seus pontos associados.

O algoritmo GRASP foi iterado 10 vezes e partir dessas iterações foram obtidas as médias de custo e tempo, e o melhor custo (melhor solução) dessas iterações.

Tam. Instância	Q. Hubs	GRASP			ALGORITMO GULOSO	
PA		Média Custos	Média Tempo(s)	Melhor Custo	Tempo	Custo
50	2	183,18	0,026	**177,1**	0,005	183,0
100		382,31	0,126	**365,2**	0,012	381,4
150		914,78	0,275	**886,6**	0,045	898,5
200		1872,54	0,458	**1752,9**	0,086	1820,2
DF						
50	3	106,18	0,04905	**40,5**	0,00811	467,2
100		302,53	0,11353	**119,8**	0,01103	796,3
150		570,25	0,24586	**360,5**	0,04421	1149,2
200		857,81	0,42185	**748,8**	0,05904	1338,7
RJ						
50	7	5,24	0,01755	**4,0**	0,01093	48,9
100		10,35	**0,06192**	**6,1**	0,09646	123,7
150		25,65	0,14467	**14,2**	0,03378	260,1
200		42,86	0,23688	**31,4**	0,04868	506,9

Tabela 1 – Tabela de resultados dos testes com GRASP e Algoritmo Guloso

Como verificado na Tabela 1, a quantidade de pontos (clientes) para atender influencia diretamente no custo total e na média de tempo gasto. Quanto mais clientes para atender, mais ligações são feitas entre pontos não-hubs e hubs.

Considerando os três estados, temos que o PA e DF apresentam os maiores custos, enquanto o RJ apresenta os menores custos. Como nas instâncias as distâncias entre dois pontos não costumam variar muito, atribuímos essa diferença de valor de solução à densidade da rede. As instâncias de RJ são mais densas que as demais, com mais rodovias e opções de ligação entre pontos não-hubs e hubs, o que pode causar também um aumento no tempo de processamento.

Analisando os resultados da meta-heurística e do algoritmo guloso pode-se notar que há grande diferença nos resultados.Ao comparar o tempo gasto da execução dos algoritmos pode-se observar que o algorítimo guloso é mais rápido, no entanto, o seu custo é sempre maior.

Em relação ao custo da solução obtida, o método GRASP foi superior em todos os casos, utilizando um tempo computacional similar ao do algoritmo guloso. Por exemplo, nos valores referentes ao Distrito Federal, ao compará-los na menor instância (50 pontos) há um decréscimo de mais de 90% no custo total ao utilizar o GRASP.

As 12 instâncias quando aplicadas ao algoritmo GRASP tem um custo menor, viabilizando assim o uso dessa heurística como forma de achar uma solução. O tempo obtido tem influência de vários fatores sendo um deles o hardware da máquina em que os testes foram executados, ao melhorar os componentes é esperado uma melhora no tempo gasto.

5 Conclusão

Problemas de otimização são cada vez mais comuns, ainda mais inseridos em uma rede de entregas, como a empresa Loggi. A entrada e saída de mercadorias tem um fluxo muito grande, assim há sempre a necessidade de melhorar a logística do negócio. Sabe-se que buscar sempre os menores custos é uma forma de obter lucro e crescer a companhia, por conta disso neste trabalho foi proposto o uso da meta-heurística GRASP adaptada ao problema de localização de hubs capacitados (PpHC), que visa minimizar o custo total do fluxo de mercadorias.

Utilizou-se uma base de dados da empresa com pontos reais, conferindo assim uma aplicação real do trabalho. Os resultados foram gerados a partir de testes em doze instâncias de tamanhos variados e apresentados em uma tabela mostrando o tempo médio, o custo médio, e o melhor custo de cada execução separados por estado. A capacidade de cada hub foi obtida por meio de formulações matemáticas, que garantem um maior esforço para resolução desse problema.

Grandes corporações geralmente tomam decisões gulosas sempre visando o melhor resultado no momento, seja este de maximizar os lucros ou diminuir os custos, sem visualizar esses ganhos e perdas a longo prazo. Por isso que o uso de uma heurística que disponibiliza essa visão, explorando inúmeras soluções, se torna extremamente importante.

O atual trabalho mostrou bons resultados para o problema de localização de hubs, conseguindo atingir soluções satisfatórias quando comparadas a um algoritmo puramente guloso, em um tempo computacional interessante.

Como trabalhos futuros pretende-se buscar na literatura formas de melhorar o algoritmo tanto em relação ao tempo quanto ao custo, e para isso, técnicas como Mineração de Dados e Reconexão de Caminhos devem ser analisadas (CLÍMACO et al., 2019). Além disso, pretende-se também desenvolver uma interface gráfica que facilite a utilização do método por meio de mapas interativos.

Referências

ALKAABNEH, F.; DIABAT, A.; ELHEDHLI, S. A lagrangian heuristic and grasp for the hub-and-spoke network system with economies-of-scale and congestion. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 102, p. 249–273, 2019. Citado 2 vezes nas páginas 20 e 24.

AYKIN, T. Lagrangian relaxation based approaches to capacitated hub-and-spoke network design problem. *European Journal of Operational Research*, Elsevier, v. 79, n. 3, p. 501–523, 1994. Citado na página 19.

BECCENERI, J. C. Meta-heurísticas e otimização combinatória: Aplicações em problemas ambientais. *INPE, Sao José dos Campos*, 2008. Citado na página 19.

BÜTÜN, C.; PETROVIC, S.; MUYLDERMANS, L. The capacitated directed cycle hub location and routing problem under congestion. *European Journal of Operational Research*, Elsevier, v. 292, n. 2, p. 714–734, 2021. Citado na página 24.

CAMPBELL, J. A survey of network hub location. *Studies in Locational Analysis*, v. 6, p. 31–49, 01 1994. Citado na página 24.

CAMPBELL, J. F. Location and allocation for distribution systems with transshipments and transportion economies of scale. *Annals of operations research*, Springer, v. 40, n. 1, p. 77–99, 1992. Citado na página 23.

CARVALHO, R. de. Heurísticas paralelas aplicadas a problemas de alocação de concentradores. Universidade Federal de Minas Gerais, 2017. Citado 2 vezes nas páginas 24 e 29.

CLÍMACO, G. et al. Combining integer linear programming with a state-of-the-art heuristic for the 2-path network design problem. *International Transactions in Operational Research*, Wiley Online Library, v. 26, n. 2, p. 615–641, 2019. Citado na página 35.

FARAHANI, R. Z. et al. Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & industrial engineering*, Elsevier, v. 64, n. 4, p. 1096–1109, 2013. Citado na página 23.

FEITEIRA, I. d. F. Reconfiguração de sistemas de distribuição usando uma estratégia baseada na meta-heurística grasp. 2021. Citado na página 27.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization*, Springer, v. 6, n. 2, p. 109–133, 1995. Citado 4 vezes nas páginas 20, 25, 29 e 31.

FRANCESQUINI, E. de C. Escalonamento através de perfilamento em sistemas multi-core. 2009. Citado na página 25.

GAREY, M. R.; JOHNSON, D. S. *Computers and intractability*. [S.l.]: freeman San Francisco, 1979. v. 174. Citado na página 19.

GLOVER, F. W.; KOCHENBERGER, G. A. *Handbook of metaheuristics*. [S.l.]: Springer Science & Business Media, 2006. v. 57. Citado na página 19.

GOOGLE. *Distance Matrix API*. 2021. <https://console.cloud.google.com/marketplace/product/google/distance-matrix-backend.googleapis.com?q=search&referrer=search>. Acesso 07 de outubro de 2021. Citado na página 29.

HENDERSON, D. et al. Handbook of metaheuristics. *Chapter Theory Pract. Simulated Annealing*, v. 57, p. 287–319, 2003. Citado na página 24.

KAVEH, A. *Applications of metaheuristic optimization algorithms in civil engineering*. [S.l.]: Springer, 2017. Citado na página 24.

LOGGI. *Github LOGGI*. 2021. <https://github.com/loggi/loggibud>. Acesso 08 de outubro de 2021. Citado na página 29.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search: Framework and applications. In: *Handbook of Metaheuristics*. [S.l.]: Springer, 2010. p. 363–397. Citado na página 24.

LÜER-VILLAGRA, A.; EISELT, H. A.; MARIANOV, V. A single allocation p-hub median problem with general piecewise-linear costs in arcs. *Computers & Industrial Engineering*, Elsevier, v. 128, p. 477–491, 2019. Citado 2 vezes nas páginas 23 e 29.

MORAES, D. G. de; FERREIRA, C. V.; SILVA, A. M. da. Otimização da produção utilizando programação linear: estudo de caso em uma indústria de esquadrias de alumínio. *Refas-Revista Fatec Zona Sul*, v. 5, n. 4, p. 26–37, 2019. Citado na página 19.

MORO, M. F. et al. Técnicas de pesquisa operacional aplicadas na otimização de rotas de uma rede de lojas de materiais de construção. *Produção em Foco*, v. 8, n. 3, 2018. Citado na página 20.

O'KELLY, M. E. The location of interacting hub facilities. *Transportation science*, INFORMS, v. 20, n. 2, p. 92–106, 1986. Citado na página 20.

O'KELLY, M. E. A quadratic integer program for the location of interacting hub facilities. *European journal of operational research*, Elsevier, v. 32, n. 3, p. 393–404, 1987. Citado na página 23.

REEVES, C. Modern heuristics techniques for combinatorial problems. *Nikkan Kogyo Shimbun*, 1997. Citado na página 24.

SANGSAWANG, O.; CHANTA, S. Capacitated single-allocation hub location model for a flood relief distribution network. *Computational Intelligence*, Wiley Online Library, v. 36, n. 3, p. 1320–1347, 2020. Citado 2 vezes nas páginas 23 e 29.

SOUZA, M. J. et al. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, Elsevier, v. 207, n. 2, p. 1041–1051, 2010. Citado na página 24.