LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

UM ALGORITMO BASEADO EM BRKGA E VND PARA O PROBLEMA DE ORIENTAÇÃO COM SELEÇÃO DE HOTÉIS

Jordi Henrique Marques da Silva
Universidade Federal de Viçosa
Km 7 – Zona Rural, MG-230, Rodoviário 38810-000 Rio Paranaíba/MG
jordih.silva@gmail.com

Christian Rodrigues Moura
Universidade Federal de Viçosa
Km 7 – Zona Rural, MG-230, Rodoviário 38810-000 Rio Paranaíba/MG
christianmoura.moura@hotmail.com

Matheus Nohra Haddad
Universidade Federal de Viçosa
Km 7 – Zona Rural, MG-230, Rodoviário 38810-000 Rio Paranaíba/MG
matheus.haddad@ufv.br

RESUMO

No Problema de Orientação com Seleção de Hotéis – POSH, existe um veículo disponível que deve realizar uma rota para atender um número de clientes, sem exceder um tempo limite de jornada diária. Levando em consideração este limite de jornada diário, torna-se necessária a seleção de um hotel mais próximo para realizar uma pausa, e assim, posteriormente retornar para uma nova jornada. Com base nessa restrição, uma rota de atendimento é composta inicialmente por um hotel preestabelecido, vários atendimentos a clientes, com os hotéis respectivos para descanso, e finalmente finalizando em um hotel também predefinido. O objetivo do POSH é encontrar uma rota cujo lucro obtido seja maximizado. Para solucionar o POSH é proposto um algoritmo heurístico baseado em Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA) e Descida em Vizinhança Variável (VND). O algoritmo foi aplicado nas instâncias da literatura e obteve resultados competitivos, atingindo 132 soluções ótimas.

PALAVRAS CHAVE. POSH, BRKGA, VND.

Tópicos: MH - Metaheurísticas

ABSTRACT

In the Orientation Problem with Hotel Selection – OPSH, there is a vehicle that must follow a route to serve a number of customers, without exceeding a daily journey time limit. Taking into account this daily journey limit, it is necessary to select a nearest hotel to take a break, and thus, later return for a new journey. Based on this restriction, a service route is initially composed of a pre-established hotel, several customer services, with the respective hotels for rest, and finally ending in a predefined hotel. The goal of OPSH is to find a route whose profit is maximized. To solve the OPSH, a heuristic algorithm based on Biased Random-Key Genetic Algorithm (BRKGA)

and Variable Neighborhood Descent (VND) is proposed. The algorithm was applied in the literature instances and obtained competitive results, reaching 132 optimal solutions.

KEYWORDS. OPHS, BRKGA, VND.

Paper Topics: MH - Metaheuristics

1. Introdução

Com o aumento da população no último século criaram-se novas necessidades, como o desenvolvimento de novos métodos para a fabricação de recursos, objetos e produção de alimentos. Visando este aumento na produção de recursos surge um amplo mercado do ramo da logística. A entrega destes recursos e alimentos, podem ter características únicas que variam de região a região. Estas características podem ser um limite de distância por jornada, e/ou um roteamento que dure dias.

Sendo assim, o planejamento de rotas se torna uma atividade de extrema importância, pois a ordem que cada cidade é visitada pode impactar em uma diminuição significativa nos custos, e este é um dos motivos que justifica o porquê dos problemas de roteamento serem amplamente estudados na área de otimização. Um dos problemas de roteamento amplamente conhecido é o Problema de Orientação (PO) que se consiste em um conjunto de localizações que possuem uma bonificação por visita, cada viagem começa em um ponto pre-estabelecido e termina em outro também pre-estabelecido, portanto o objetivo deste problema é maximizar a bonificação por visita e respeitar um limite de tempo por viagem. O PO foi introduzido por [Golden et al., 1987] como um problema que pertence à classe *NP-difícil* e hoje é amplamente explorado na literatura, com muitas variações.

Dentre as variações do PO encontra-se o Problema de Orientação com Seleção de Hotéis (POSH), proposto por [Divsalar et al., 2013] e seu objetivo é organizar a rota em um hotel inicial pre-estabelecido intercalado com várias jornadas que são compostas por um intervalo de um hotel a outro e vários clientes, o final da rota é definido por um hotel final que foi pre-estabelecido, sem que um limite de jornada seja desrespeitado. O POSH também se enquadra no conjuntos de problemas *NP-difícil*.

Habitualmente ao se trabalhar com problemas da classe *NP-difícil*, recorre-se à utilização de estratégias heurísticas que proporcionam encontrar soluções de boa qualidade em tempo computacional aceitável. Desta forma foi desenvolvido um algoritmo baseado na meta-heurística *Algoritmo Genético de Chaves Aleatórias Viciadas* (BRKGA) [Gonçalves e Resende, 2011] e que conta também com a heurística de refinamento *Descida em Vizinhança Variável* (VND)[Mladenović e Hansen, 1997] para solucionar o POSH. A primeira etapa do algoritmo é criar a população inicial contendo soluções aleatórias e soluções criadas por uma heurística construtiva parcialmente gulosa. Esta heurística possui o critério de seleção do cliente mais próximo ao último cliente já selecionado e que este induza a melhor bonificação. Posteriormente estas soluções são refinadas com a aplicação do VND. Em seguida, os indivíduos iniciam o ciclo evolutivo, onde cruzamentos e mutações são realizados, além de novas aplicações do VND. O algoritmo desenvolvido foi testado em instâncias de *benchmark* da literatura para validar a metodologia proposta.

Este artigo está estruturado da seguinte maneira, na seção 2 o problema tratado é especificado, na seção 3 é feito uma revisão bibliográfica destacando as principais inspirações para este artigo. Na seção 4 encontra-se a descrição da metodologia utilizada. Os resultados computacionais se encontram na seção 5. Por fim na seção 6 está a conclusão e o que pode ser feito para aperfeiçoamento deste trabalho.

2. Problema de Orientação com Seleção de Hotéis

O Problema de Orientação com Seleção de Hotéis (POSH) pode ser definido em um grafo $G = (V, A)$ em que $V = H \cup C$, sendo que $H = \{0, ..., f\}$ representa o conjunto de hotéis, no qual H_0 representa o hotel de partida e H_f representa o último hotel a ser visitado. $C = \{0, ..., n\}$ representa o conjunto dos possíveis clientes a serem atendidos. O conjunto de arestas $A = \{(i, j)|i, j \in V, i \neq j\}$, contém o tempo de viagem t_{ij} entre cliente ou hotel i e o cliente ou hotel j. Cada cliente $i \in C$ contém uma bonificação por visita, sendo representada por S_i, porém $S_i = 0, \forall i \in H$. Existe um veículo disponível que deve realizar D viagens. Cada viagem $d = \{1, ..., D\}$ possui um tempo limitado para sua finalização, representado por T_d. Deve-se garantir que se o cliente for visitado, será apenas uma vez. Nem todos os clientes precisam ser visitados.

Uma viagem é definida partindo de um hotel inicial, atendendo alguns clientes e finalizando em outro hotel. Uma rota é o conjunto de todas as d viagens percorridas para atender os clientes pré-estabelecidos, partindo do hotel H_0 e finalizando no hotel H_f. A rota possui um tempo máximo para ser concluída, representado por $T_{\max}$. O objetivo deste problema é encontrar uma rota que maximize a bonificação adquirida pelos atendimentos aos clientes, respeitando o tempo máximo da rota e os tempos limites por viagem.

2.1. Exemplo de Aplicação

Na Figura 1 é representada a solução de uma instância do POSH que possui 23 clientes e 4 hotéis. Os vértices representam os clientes e hotéis. Todos os clientes possuem uma numeração para identificação. Os hotéis também são numerados, porém sempre precedidos da letra 'H'.

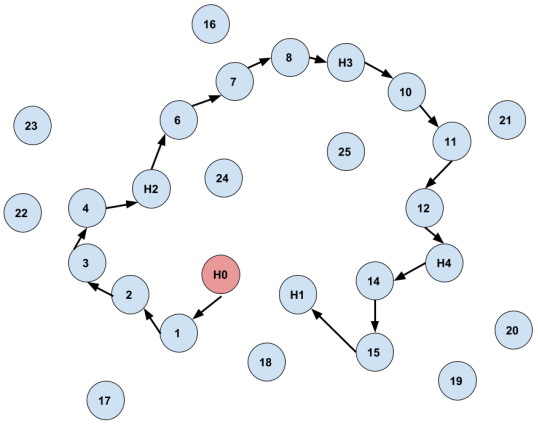

Figura 1: Exemplo de uma Solução para o POSH

Nesta solução a rota é constituída por 4 viagens. A rota tem início no hotel H0, sendo a sequência de viagens (D1, D2, D3, D4), em que $D1 = \{H0, 1, 2, 3, 4, H2\}$, $D2 = \{H2, 6, 7, 8, H3\}$, $D3 = \{H3, 10, 11, 12, H4\}$ e $D4 = \{H4, 14, 15, H1\}$. Nota-se que os clientes $(16, 17, 18, 19, 21, 22, 23, 24, 25)$ não foram visitados, pois no POSH há limitação de tempo para as viagens, o que pode impedir que todos os clientes sejam visitados.

2.2. Formulação Matemática

Fazendo uso da notação na primeira seção, o POSH pode ser formulado como um problema linear inteiro-misto. $X_{i,j,d} = 1$ se na viagem d, uma visita ao vértice i é seguida por uma visita ao vértice j. Recebe 0 caso seja o contrário. U_i = posição do vértice i na viagem. O modelo foi proposto por [Divsalar et al., 2013].

$$Max \sum_{d=1}^{D} \sum_{i=0}^{H_f+N} \sum_{j=0}^{H_f+N} S_i X_{i,j,d} \tag{1}$$

$$\sum_{l=1}^{H_f+N} X_{0,l,1} = 1 \tag{2}$$

$$\sum_{k=0}^{H_f+N} X_{k,1,D} = 1 \tag{3}$$

$$\sum_{h=0}^{H_f} \sum_{l=0}^{H_f+N} X_{h,l,d} = 1 \qquad d = 1, ..., D \tag{4}$$

$$\sum_{h=0}^{H_f} \sum_{k=0}^{H_f+N} X_{k,h,d} = 1 \qquad d = 1, ..., D \tag{5}$$

$$\sum_{k=0}^{H_f+N} X_{k,h,d} = \sum_{l=0}^{H_f+N} X_{h,l,d+1} \qquad d = 1, ..., D-1 \qquad h = 0, ..., H_f \tag{6}$$

$$\sum_{i=0}^{H_f+N} X_{i,k,d} = \sum_{j=0}^{H_f+N} X_{k,j,d} \qquad k = H+1, ..., H_f+N \qquad d = 1, ..., D \tag{7}$$

$$\sum_{d=1}^{D} \sum_{j=0}^{H_f+N} X_{i,j,d} \leq 1 \qquad i = H+1, ..., H_f+N \tag{8}$$

$$\sum_{i=1}^{H_f+N} \sum_{j=0}^{H_f+N} t_{i,j} X_{i,j,d} \leq T_d \qquad d = 1, ..., D \tag{9}$$

$$u_i - u_j + 1 \leq (N-1)(1 - \sum_{d=1}^{D} X_{i,j,d}) \qquad i = H_f+1, ..., H_f+N \qquad j = H_f+1, ..., H_f+N \tag{10}$$

$$u_i \in 1, ...N \qquad i = H_f+1, ..., H_f+N \tag{11}$$

$$X_{i,j,d} \in \{0,1\} \, \forall i, j = 1, ..., H_f+N \mid i \not\equiv j \qquad d = 1, ..., D \tag{12}$$

A equação (1) representa a função objetivo e é responsável por maximizar a pontuação total coletada. As restrições (2) garantem que as rotas comecem a partir do hotel inicial pre-estabelecido H_0. As restrições (3) garantem que as rotas terminem no hotel final H_f também pré-estabelecido. A garantia que cada viagem comece e termine em um dos hotéis disponíveis está representada pelas restrições (4) e (5). As restrições (6) garantem que uma jornada termina em um hotel e a próxima jornada começa no mesmo. A conectividade de arestas dentro de cada jornada está garantida pelas restrições (7). Cada vértice é visitado no máximo uma vez (restrições (8)). As restrições (9) limitam cada jornada com um orçamento de tempo. Se visitar um vértice, requer então um certo tempo de serviço. Modelado com a soma de todos os tempos de viagem para os vértices. (10) e (11) são restrições de eliminação de subciclo que são formuladas com base na Formulação de Miller – Tucker – Zemlin (MTZ) para o problema do caixeiro viajante [Miller et al., 1960].

3. Revisão Bibliográfica

O problema de orientação (PO) foi proposto por [Golden et al., 1987], em que existem dois hotéis preestabelecidos. Um vértice para representar o início e outro para o final da viagem, e vários vértices clientes. Sendo assim, o objetivo é traçar uma viagem cuja tenha a maior pontuação ao visitar clientes, respeitando um determinado limite de tempo de viagem.

O *survey* desenvolvido por [Gunawan et al., 2016] descreve o Problema de Orientação e suas demais variações, dentre elas, encontra-se o POSH. Este trabalho também apresenta diversos métodos baseados em heurísticas e meta-heurísticas para solução das variações do PO.

Um método de solução ótima para resolução do problema de orientação com seleção de hotéis foi proposto por [Divsalar et al., 2013]. Neste trabalho foi aplicado uma pesquisa de vizinhança variável inclinada que consiste em uma inicialização construtiva do procedimento e um procedimento de melhoria das novas soluções(construções). O algoritmo é baseado em um operador de pesquisa de vizinhança projetado especificamente para a parte de seleção de hotel deste problema, bem como alguns bairros típicos para o PO regular. Uma matriz contendo a pontuação potencial entre todos os pares de hotéis é criada. Em seguida, a lista de todas as combinações viáveis de hotéis é determinada. Depois disso, uma série de possíveis combinações de hotéis é selecionada para construir a solução inicial bem como para ser usado na fase de melhoria. Na fase de melhoria é aplicada uma estrutura de heurística gulosa que se baseia em quatro movimentos de pesquisa local diferentes: *Inserção, Substituição, Two-Opt e Move-best.* Um único movimento é aplicado desde que sejam encontradas melhorias. Esses movimentos são aplicados, em uma ordem fixa, desde que os movimentos possam melhorar a solução. Para avaliação, eles definem um conjunto de 229 instâncias, com 224 soluções ótimas conhecidas.

Outro método de resolução proposto por [Divsalar et al., 2014] foi com utilização de um algorítimo memético (MA). Um MA é um algoritmo evolutivo hibridizado com técnicas de pesquisa. Neste trabalho o algoritmo consiste em dois níveis: um componente genético concentra-se principalmente em encontrar uma boa sequência de hotéis intermediários focado na recombinação de diferentes hotéis, enquanto seis movimentos de busca local embutidos em uma vizinhança variável trabalham com a seleção e sequenciamento de vértices entre os hotéis. Assim com o termino de iterações, é definida a melhor solução para o problema. Neste trabalho também foi proposto 176 novas instancias com resultados ótimos.

Uma abordagem baseada em uma meta-heurística conhecida como *Greedy Randomized Adaptive Search Procedure* (GRASP). Um Procedimento de Pesquisa Adaptativa é apresentado por [Sohrabi et al., 2017] para resolver o problema. Um conjunto vazio é gerado e neste são inseridos valores até que seja uma solução viável. O GRASP se diferencia dos demais métodos pelo motivo de privilegiar a criação da solução inicial de qualidade e posteriormente utilizar busca local apenas para algumas melhorias. Foi introduzido uma nova abordagem de pesquisa local baseada em programação dinâmica. É aplicado um operador com o objetivo de encontrar a melhor sequência de hotéis de acordo com a ordem atual dos nós. Obtendo uma boa sequência de hotéis durante a construção de uma viagem, esta é a principal diferença entre a proposta e a literatura antecedente.

Em 2019 foi proposto uma abordagem com Hiper-Heurística Populacional. Uma hiper-heurística é uma estrategia que explora o espaço de busca de outras heurísticas, diferente das heurísticas convencionais onde o foco é explorar diretamente o espaço do problema que esta sendo resolvido. Segundo [Toledo et al., 2019] a Hiper-Heurística proposta é de baixo nível, e utiliza heurísticas construtivas, de pertubação e agitação. No primeiro estágio é formada uma população inicial utilizando uma heurística construtiva focada em criar uma rota para POSH de forma recursiva, partindo do hotel final pre-estabelecido H_f cria-se sub-viagens com critério guloso até que o numero de viagens seja alcançado e termine no hotel inicial pre-estabelecido h_0. Nas próximas etapas são aplicadas estrategias de permutação: two-opt, troca, inserção, substituição de hotéis em sub-viagens, e substituição dupla de hotéis em mais de uma sub-viagem. Esta abordagem encontrou 217 valores ótimos dentre as 395 instancias.

4. Metodologia

4.1. Representação da Solução

Uma solução do POSH é representada por um arranjo que contém os hotéis que fazem parte da rota e todos os clientes. A Figura 2 representa uma solução para o problema com $D = 2$ viagens.

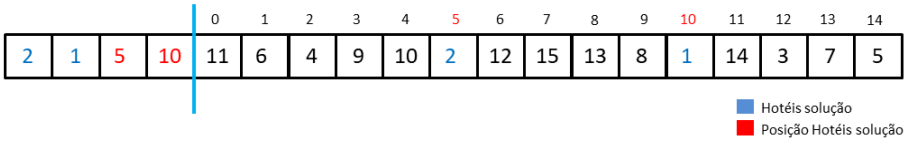

Figura 2: Representação Solução com $D = 2$.

As primeiras D casas do arranjo retratam quais hotéis serão visitados ao final de cada viagem, na ordem em que aparecem, neste exemplo os hotéis (2 e 1). As D casas sub-sequentes, definem as posições relativas de cada um destes hotéis na rota, no exemplo as posições (5 e 10). As casas restantes indicam a ordem de visitação dos clientes. Os clientes que aparecem após o último hotel visitado não fazem parte da solução. No exemplo, os clientes (14, 3, 7, 5) não serão visitados. Representar na solução os clientes que não serão visitados auxilia na realização das buscas locais.

4.2. BRKGA

Neste tópico será descrito como será realizada cada etapa do algoritmo BRKGA. A primeira etapa é gerar um conjunto de soluções com chaves randômicas que definirão a população inicial. Logo em seguida deve-se aplicar nas chaves randômicas o decodificador para que possa-se realizar a avaliação da aptidão de cada indivíduo. Logo após, a população é divida em três conjuntos: elite, não elite e mutante. Na etapa seguinte, inicia-se o ciclo evolutivo, onde por várias gerações serão aplicadas as etapas de seleção para cruzamento, cruzamento, aplicação do VND e mutação. Após a execução das etapas anteriores restará avaliar o *fitness* da nova população, e definir quais indivíduos irão para próxima geração. Este modelo pode ser implementado pelo Algoritmo 1.

Algoritmo 1 Pseudocódigo BRKGA

```
1: Gerar População Inicial
2: Decodificar as Chaves
3: Avaliar o Fitness da População
4: Separar a População (elite, não elite e mutante)
5: enquanto critério de parada não atendido faça
6:     Cruzamento
7:     VND
8:     Mutação
9:     Avaliar o Fitness da População
10:    Seleção Para Sobrevivência
11: fim enquanto
12:    Retornar Melhor Indivíduo
```

4.2.1. Gerar População Inicial

Na primeira etapa do algoritmo é realizada a geração da população inicial. Parte da população, mais precisamente a quantidade de indivíduos da população elitista, é gerada através de uma heurística parcialmente gulosa (Seção 4.2.2). O restante é gerado de forma aleatória. Depois de geradas todas as soluções, aplica-se sucessivas buscas locais com o VND. A utilização do

VND é de grande importância devido a dificuldade de se obter soluções iniciais viáveis, portanto sua aplicação aumenta a possibilidade de se trabalhar com soluções de boa qualidade a partir da primeira geração do algoritmo. A seguir é descrita a heurística construtiva parcialmente gulosa utilizada.

4.2.2. Heurística Construtiva Parcialmente Gulosa

Esta heurística inicialmente cria um conjuntos de hotéis que pertencerão à solução. Este conjunto é criado a partir da redução de cada viagem do POSH em uma instância para o Problema de Orientação (PO), em que uma solução é composta de um hotel de inicial, vários atendimentos a clientes e o término em outro hotel, sem que um limite de viagem seja extrapolado. Desta forma são criadas diversas soluções combinando todos hotéis para início e fim de viagem. A inserção dos clientes é feita a partir de um critério guloso. Para que um cliente seja inserido é necessário ter o maior valor de bonificação/distância do último vértice inserido, sem que desrespeite o limite da viagem. Em seguida, é selecionado o melhor par de hotéis (início e fim), sempre começando a rota pelo hotel pre-estabelecido para início de rota do POSH (H_0), as próximas viagens devem começar sempre do hotel final do par anterior, desta maneira respeitando as restrições de rota do POSH, e garantindo que na última viagem, o hotel final seja o hotel pre-estabelecido para fim de rota da solução do POSH (H_f). Este modelo de construção por pares de hotéis foi proposto por [Divsalar et al., 2013]. Depois de gerado o conjunto de hotéis, esta heurística insere de forma aleatória todos os clientes do cenário entre os hotéis resultando em uma solução para o POSH.

4.2.3. População Elite

A população elite é um conjunto de indivíduos que são fortes candidatos a soluções de boa qualidade. No BRKGA a população elite tem o papel de guiar as demais soluções da população a alcançar melhorias em seus alelos e consequentemente a criação de novos indivíduos de boa qualidade. No conjunto de instâncias que foram utilizadas por [Divsalar et al., 2013], existe uma grande similaridade nos valores de bonificação (S_i) ao visitar cada nó e nas distâncias entre os nós, o que resulta em dificuldade para conseguir melhorias nas soluções. Em contrapartida foi implementado um critério de diversidade para auxiliar na filtragem de indivíduos que compõem a população elite. Para adentrar ao conjunto elite é necessário garantir dois critérios que são ter o valor de fitness maior que do pior individuo elite e ter no mínimo uma porcentagem (Ce) dos alelos diferentes de todos indivíduos presentes na população elite.

4.2.4. Decodificador

Entende-se que cada chave aleatória atribuída a um gene do cromossomo representa um possível cliente ou hotel. Um conjunto de chaves representa um cromossomo (indivíduo).

O decodificador é um operador que transmuta as chaves aleatórias em uma solução para o problema POSH. A formação da solução é dividida em três etapas, na primeira é decodificado quais hotéis fazem parte da solução e quais as posições que pertencem dentro da rota. A segunda etapa é definido qual a ordem que cada cliente está na rota. E por fim são inseridos os hotéis definindo as sub-viagens.

Sabemos que por instância temos uma quantidade de hotéis e clientes definidos. A quantidade de hotéis é utilizada para criar uma representação real para cada hotel. Esta representação é definida em um intervalo chamado de $intervaloHoteis$. O cálculo deste intervalo é definido por uma operação de divisão, em que ao dividir 1 por $H + 2$, temos os intervalos que representa cada hotel a partir de H_0. A quantidade de clientes mais o número de viagens é utilizada para definir as possíveis posições que os hotéis podem fazer parte na rota, logo é criado um intervalo chamado de $intervaloPosicoesHoteis$, que é definido dividindo 1 por $(N - 2) + D$, assim cada posição da solução tem uma representação real. A Figura 3 demonstra uma solução codificada e decodificada.

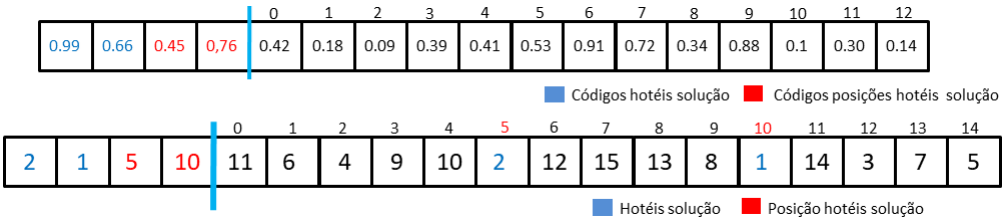

Figura 3: Solução codificada e decodificada com $D = 2$.

O primeiro passo do decodificador é percorrer as primeiras *D* casas, e identificar a qual seguimento do $intervaloHoteis$ cada chave pertence, desta forma é identificado qual hotel a chave representa. As próximas *D* chaves determinam quais são as posições que os hotéis estão na rota, então ao percorrer é identificado a qual seguimento as chaves fazem parte no $intervaloPosicoesHoteis$, assim definindo qual a posição que cada hotel está na rota, respectivamente. Na segunda etapa é realizada uma ordenação crescente dos valores aleatórios das próximas (N-2) casas, conservando qual índice cada valor pertence. A nova ordem das chaves resultam na sequência que cada cliente é visitado, a partir do índice que ela pertencia anteriormente. Por fim é inserido cada hotel em sua respectiva posição, assim resultando em uma solução do POSH.

4.2.5. Avaliação da Solução(Fitness)

A avaliação deste problema visa maximizar o Lucro(S_i) obtido ao visitar cada cliente, desde que respeite a restrição de tempo para cada jornada. Para computar a aptidão do indivíduo então é necessário realizar o somatório do S_j da rota e penalizar o tempo excedido em cada jornada. A constante α deve receber um valor alto, para que seja possível corrigir a avaliação dos indivíduos que não respeitarem as restrições de tempo de viagem. O valor de α é determinado por o software irace [López-Ibáñez et al., 2016].

$$\max \sum_{d=1}^{D} \sum_{i=0}^{N-2} \sum_{j=1}^{N-1} S_j - \alpha * \max(0, T_{pd} - T_d) \tag{13}$$

T_{pd} é o tempo percorrido na viagem d e T_d é o tempo total permitido na viagem d.

4.2.6. Cruzamento (*crossover*)

O método de cruzamento utilizado neste trabalho é o cruzamento uniformizado e parametrizado proposto por [Spears e De Jong, 1991]. Nesta etapa a população se divide em duas categorias, elite e não elite, e para cada filho que será formado é sorteado um individuo de cada categoria. Após isso é lançada uma moeda viciada por alelo, A moeda possui valor de 0 a 1. caso a moeda seja menor ou igual ao parâmetro Phe, o alelo escolhido será do pai elite, caso contrário é escolhido do pai não elite.

4.2.7. Mutação

A etapa de mutação é a responsável por garantir a diversidade genética dentro da população. Nesta etapa é aplicado uma simples permutação onde é sorteado dois genes de cada solução, um gene que represente um hotel e um gene represente um cliente e sorteamos uma nova chave aleatória para cada gene. Por se tratar de um operador não tradicional do BRKGA, se torna necessário a sortear indivíduos da população não elite até que se tenha (Pm) indivíduos mutados respeitando o parâmetro definido no inicio do algoritmo .

4.2.8. Seleção para Sobrevivência

A seleção para sobrevivência no BRKGA combina as seguintes populações: elite, não elite e mutante. No final de cada geração a população elite vigente é copiada para a próxima geração, novas soluções criadas na etapa de mutação vão compor a proporção da população mutante e a quantidade de indivíduos restantes é composta por cruzamentos entre membros da população elite e não elite.

4.2.9. Estruturas de Vizinhança

Para a definição de uma estrutura de vizinhança é necessário definir um movimento sobre uma solução, o resultado deste movimento será uma solução vizinha a inicial. Neste trabalho foram definidas três estruturas de vizinhança, que são representadas por N_1,N_2, N_3 e N_4.

- N_1 **- Troca:** Um simples e eficiente método de aplicar uma permutação em uma solução é utilizar a troca de valores (posições) da solução. Dado um vetor de tamanho n, tem se um valor em s_i e um em s_j, então estes valores são invertidos. Ou seja, a posição **i** da solução recebe o valor de **j** e a posição **j** da solução recebe o valor de **i**.

- N_2 **- Inserção:** Para realizar a inserção é preciso selecionar uma posição **i**, remover o valor corrente, em seguida realizar a concatenação dos valores restantes e por fim inserir o valor de **i** removido na posição de **j**.

- N_3 **- 2-opt:** O movimento 2-opt é muito eficaz para evitar que a rota cruze a si mesma e realize uma reordenação para que ela não gere estes ciclos.

  Para explorar essa estrutura é necessário selecionar um nó **i** e um nó **j** desde que não sejam adjacentes, e inverter o sub-vetor resultante entre **i** e **j**. Esta inversão gera a troca das arestas eliminando possíveis ciclos (cruzamentos) naquele intervalo.

- N_4 **- Substituição de Hotéis:** O movimento de substituição de hotéis é crucial para garantir o uso do melhor hotel para cada sub-rota do POSH. A exploração desta vizinhança é dada ao substituir um hotel **i** que faz parte da solução por outro hotel **j** que não faz parte.

4.2.10. Descida em Vizinhança Variável

Após a implementação dos mecanismos de busca local, tornou-se mais prático a definição do VND. Dado que as vizinhanças: N_1, N_2, N_3, N_4 já estavam determinadas, seguimos para a definição da ordem em que tais estruturas vão ser exploradas e a estratégia de melhoria. A vizinhança que será aplicada primeiramente é a N_2 em primeira melhora (*first improvement*), a próxima vizinhança a ser explorada N_1 também em primeira melhora (*first improvement*). Sequencialmente são aplicadas as vizinhanças N_3 e N_4 ambas em melhor melhora (*best improvement*). Sempre que houver melhoras ao aplicar outras vizinhanças, a busca será redirecionada para vizinhança N_2, desta forma é garantido que o método terá fim apenas quando não for possível mais extrair melhoras de todas as vizinhanças. A aplicação do VND é realizada em todos indivíduos resultantes do cruzamento, visando explorar diferentes ótimos locais.

5. Experimentos e Resultados Computacionais

A implementação deste algoritmo é totalmente autoral, não foi utilizado o framework disponibilizada por [Gonçalves e Resende, 2011]. Para execução da metodologia foi utilizado instâncias benchmark presentes na literatura, onde o algoritmo foi executado 10 vezes para cada uma das instâncias, visando observar a variabilidade do algoritmo proposto. Sobre a implementação do algoritmo foi utilizado a linguagem C++ na versão 11.0, o compilador é o g++ na versão 7.4. A execução destes experimentos ocorreram em um notebook *Acer AN515-51-50U2* com 8 GB de memória RAM e processador Intel Core i5 versão 7300HQ com 2.5 GHz no sistema operacional Ubuntu 18.04 LTS.

5.1. Instâncias benchmark

Neste experimento foi utilizado um conjunto de 395 instâncias criadas e disponibilizadas por [Divsalar et al., 2014], todas elas possuem soluções ótimas conhecidas. Este conjunto é subdivido em grupos, onde varia a quantidade de hotéis extra, viagem e também o limite de jornada. Do primeiro ao último grupo são encontradas instâncias onde o número de viagens varia de 2 a 10, a quantidade de hotéis extra de 1 a 15 e o número de clientes de 30 a 100 clientes.

5.2. Parâmetros

O BRKGA possui parâmetros de entrada que guiam o algoritmo para diferentes regiões do espaço de busca, por isso a definição dos mesmos é um dos maiores desafios. A combinação dos parâmetros podem ocasionar em soluções de boa ou má qualidade. Neste trabalho foi utilizado o software irace [López-Ibáñez et al., 2016] na versão 3.4 para determinação dos valores de entrada. Os valores encontrados pelo irace foram: tamanho da população ($P = 68$), critério de diversidade entre indivíduos elitistas ($Ce = 0,29$), proporção da população elite ($Pe = 0,13$), proporção da população mutante ($Pm = 0,13$), probabilidade de herdar um alelo do pai elite ($Phe = 0,8$), número de gerações ($Ng = 83$) e por fim valor da constante para penalização ($\alpha = 836,01$).

5.3. Resultados

A comprovação do método proposto é feita ao comparar seus resultados com os da literatura. A Tabela 1 resume os resultados do BRKGA para cada grupo, e o compara com quatro métodos que solucionam o POSH, MA [Divsalar et al., 2014], SVNS [Divsalar et al., 2013], GRASP [Sohrabi et al., 2017] e HH [Toledo et al., 2019]. Para possibilitar a comparação é utilizado o GAP em relação ao ótimo conhecido, que é calculado pela seguinte equação: $GAP(\%) = 100 \times \left(\frac{opt - resultadoobtido}{opt}\right)$. Em sua primeira coluna está o conjunto de instâncias, na segunda e terceira coluna mostram os GAPs relacionados aos melhores valores obtidos e as médias de todas execuções para o MA, na quarta coluna estão os GAPs dos melhores valores obtido pelo SVNS, na quinta e sexta colunas estão os GAPs das melhores soluções e das médias para o GRASP. Na sétima e oitava colunas são encontrados os GAPs das melhores soluções e das médias para o HH. Em seguida, na nona e décima colunas estão os GAPs das melhores soluções e as médias do BRKGA. Nas colunas seguintes estão a média do tempo computacional de cada algoritmo.

Observando a Tabela 1 é possível perceber que a estratégia se mostrou muito eficiente nas instâncias menores, em que a quantidade de hotéis extras varia entre 1 e 6 e vai de duas até quatro viagens. O algoritmo obteve bom desempenho no SET 1, adquirindo resultados melhores que a abordagem MA [Divsalar et al., 2014] na categoria ***best*** no *SET1-1-2*. Nos conjuntos 2-3 e 3-4 também do SET 1 e 5-3 e 6-4 do SET 2 superou os resultados do GRASP [Sohrabi et al., 2017] na categoria de melhor resultado. Ao analisar o desempenho do método proposto nos Set 1 e 2 é possível notar que a eficiência do algoritmo é maior ao se trabalhar com rotas onde o número de viagens e hotéis extras são próximos. Isto pode indicar um potencial de robustez do algoritmo proposto, pois o mesmo foi capaz de encontrar 99 valores ótimos de 175 instâncias e ficar em media a 2% do melhor resultado nas instâncias restantes. Em contra partida observa-se que a eficiência do BRKGA diminui com o incremento da diferença entre o número de viagens e o número de hotéis extras.

No grupo de instâncias com elevado número de hotéis extras e viagens, o BRKGA encontrou dificuldades para encontrar boas soluções. Devido à imensa quantidade de possíveis subviagens, se cria uma região muito grande a ser explorada o que exige maior tempo computacional para encontrar soluções de boa qualidade. Mesmo com estas dificuldades, o método proposto conseguiu alcançar **132** soluções ótimas, superando o SVNS [Divsalar et al., 2013] que atingiu **121**. Já

Tabela 1: Resultados por Conjunto de instâncias

	MA[1] Gap(%)		SVNS[2] Gap(%)	GRASP[3] Gap(%)		HH[4] Gap(%)		BRKGA[5] Gap(%)		MA[1] CPU(s)	SVNS[2] CPU(s)	GRASP[3] CPU(s)	HH[4] CPU(s)	BRKGA[5] CPU(s)
Set	best	avg	best	best	avg	best	avg	best	avg	avg	best	avg	avg	avg
SET1-1-2	2.82	3.31	1.21	0.25	0.25	1.81	1.90	2.46	5.36	1.84	0.12	2.57	1.90	30.38
SET1-2-3	0.42	0.88	0.92	1.56	1.56	0.34	0.36	1.55	3.55	1.43	0.14	3.71	1.19	25.37
SET1-3-4	0.45	0.60	0.92	4.60	5.19	0.31	0.32	1.68	3.29	1.09	0.41	4.78	0.87	46.51
SET2-5-3	0.40	0.80	0.92	1.56	1.56	0.34	0.34	1.51	3.77	1.41	0.21	4.93	1.19	44.31
SET2-6-4	0.39	0.64	1.21	3.81	4.89	0.31	0.33	1.96	4.41	1.15	0.88	6.11	0.90	41.32
SET3-10-4	0.95	1.71	2.61	0.30	0.42	0.39	0.61	11.66	14.83	6.59	5.12	37.38	7.11	232.12
SET3-12-5	1.43	1.98	3.57	1.06	1.85	0.65	0.84	13.15	16.90	5.44	4.21	41.25	5.37	100.31
SET5-10-5	1.30	1.98	4.15	0.78	1.38	0.67	0.83	9.76	13.76	5.43	4.26	38.88	5.18	495.75
SET5-10-6	1.23	2.30	6.26	0.73	1.24	0.56	0.75	10.84	14.90	4.66	3.83	38.51	4.22	216.66
SET5-12-4	1.24	1.77	2.73	0.35	0.59	0.72	0.85	9.21	12.37	6.50	5.01	41.51	7.10	446.69
SET5-12-6	1.62	2.27	5.63	1.61	2.78	0.71	0.99	11.38	14.84	3.49	3.84	41.17	4.50	195.35
SET5-15-4	1.32	1.92	2.66	0.30	0.43	0.61	0.77	10.41	13.33	6.63	4.95	46.03	7.43	343.26
SET5-15-5	1.39	2.21	4.42	0.95	1.54	0.88	1.02	10.69	14.25	5.41	4.20	45.47	5.87	529.24
SET5-15-6	1.38	2.54	6.85	0.89	1.65	0.92	1.02	10.57	14.37	4.78	4.29	45.41	5.08	216.66
SET5-15-8	2.95	3.66	-	4.58	5.92	1.87	2.38	14.43	19.27	5.16	-	64.74	5.43	237.41
SET5-15-10	3.77	5.03	-	7.55	8.97	1.82	3.63	14.71	17.61	5.04	-	86.96	5.28	668.75
Average	1.44	2.10	-	1.93	2.51	0.81	1.06	8.50	11.68	4.13	-	34.33	4.29	241.88

[1]Testes realizados em um PC Intel Intel Core 2 com 3.00 GHz e 4 GB de RAM

[2]Testes realizados em um PC Intel Core 2 Duo com 2.4 GHz e 2 GB de RAM

[3]Testes realizados em um PC Intel Core i7-2600 com 3.50 GHz com e 16 GB de RAM

[4]Testes realizados em um PC Intel Core i7-2600 com 3.40 GHz com e 16 GB de RAM

[5]Testes realizados em um PC Intel Core i5-7300HQ com 2.50 GHz com e 8 GB de RAM

os algoritmos HH [Toledo et al., 2019], GRASP [Sohrabi et al., 2017] e MA [Divsalar et al., 2014] alcançaram **217**, **201** e **170** respectivamente.

Também pode-se observar pela Tabela 1 a discrepância de tempo entre as abordagens da literatura e o BRKGA, porém isto se deve a vários motivos, como a diferença de desempenho das máquinas que foram realizados os testes. Outro motivo determinante é a forma em que a metodologia proposta foi implementada, ao optar por desenvolver uma estratégia em que se combina uma heurística de refinamento e uma meta-heurística populacional, espera-se que tenha um alto custo computacional. É importante ressaltar ainda que o algoritmo não utiliza o ***framework*** de Gonçalves e Resende [2011] e que foi totalmente implementado pelos autores, por isto algumas otimizações podem ter passado despercebidas.

6. Conclusão e Trabalhos Futuros

Este trabalho apresentou um método de resolução para o POSH que utiliza uma nova abordagem de representação de solução para o problema junto a uma metodologia que estuda a junção entre a meta-heurística BRKGA e a heurística de busca local VND. O algoritmo proposto foi executado em instâncias de benchmark da literatura e demonstrou grandes indícios de ser competitivo com os algoritmos do estado da arte para o POSH devido aos resultados obtidos. Para os grupos de instâncias do SET1, o algoritmo obteve melhores resultados ao ser comparado ao MA[Divsalar et al., 2014], no grupo *SET1-1-2* em relação à categoria ***best*** e ainda superou o GRASP [Sohrabi et al., 2017] nos grupos *SET1-2-3* e *SET1-3-4* no mesmo quesito. Para os grupos de instancias do SET2, obteve melhores resultados em comparação ao GRASP [Sohrabi et al., 2017]. Porém, nos demais grupos de instâncias do SET3 e SET5, o algoritmo obteve dificuldades devido ao elevado número de hotéis e viagens gerando um enorme número de combinações possíveis.

Em relação ao tempo de processamento médio é notável um dispêndio de tempo maior pelo algoritmo, isso ocorre devido ao uso do VND em todos os indivíduos gerados pelos cruzamentos. A busca local foi utilizada desta maneira afim de explorar o comportamento em conjunto

com o BRKGA e melhorar as soluções geradas. Foi possível observar que o uso constante do VND melhora bem as soluções, mas interfere bastante na evolução do BRKGA. Ainda sim o método proposto conseguiu alcançar **132** soluções ótimas conhecidas. A relevância destes resultados obtidos comprovam a potencialidade da metodologia abordada, que se corretamente lapidada poderá atingir altos níveis de competitividade.

Finalmente, propõe-se, como trabalhos futuros realizar o estudo do uso da busca local VND, condicionando a execução conforme a dificuldade do BRKGA em melhorar as soluções elite buscando assim reduzir o seu tempo computacional e explorar melhor o potencial do BRKGA. Além disso, pretende-se analisar melhor a implementação do BRKGA e, possivelmente testar o ***framework*** desenvolvido por [Gonçalves e Resende, 2011].

Referências

Divsalar, A., Vansteenwegen, P., e Cattrysse, D. (2013). A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145(1):150 – 160. ISSN 0925-5273.

Divsalar, A., Vansteenwegen, P., Sörensen, K., e Cattrysse, D. (2014). A memetic algorithm for the orienteering problem with hotel selection. *European Journal of Operational Research*, 237(1): 29 – 49. ISSN 0377-2217.

Golden, B. L., Levy, L., e Vohra, R. (1987). The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318.

Gonçalves, J. e Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics*, 17:487–525.

Gunawan, A., Lau, H. C., e Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255 (2):315 – 332. ISSN 0377-2217.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43 – 58. ISSN 2214-7160.

Miller, C. E., Tucker, A. W., e Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100. ISSN 0305-0548.

Sohrabi, S., Ziarati, K., e Keshtkaran, M. (2017). A novel method for solving the orienteering problem with hotel selection. In *2017 International Symposium on Computer Science and Software Engineering Conference (CSSE)*, p. 7–11.

Spears, W. e De Jong, K. (1991). On the virtues of parametrized uniform crossover.

Toledo, A., Riff, M., e Neveu, B. (2019). A hyper-heuristic for the orienteering problem with hotel selection. *IEEE Access*, 8:1303–1313.