LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

Uma abordagem ILS-BanditVND para o Team Orienteering Problem

Jéssica Richards Nascimento
Universidade Federal do Rio de Janeiro - UFRJ
Rio de Janeiro/RJ - Brasil
nascj@cos.ufrj.br

Pedro Henrique González
Universidade Federal do Rio de Janeiro - UFRJ
Rio de Janeiro/RJ - Brasil
pegonzalez@cos.ufrj.br

RESUMO

Esse trabalho aborda o problema de orientação de times da classe problemas NP-difícil. Para solução do problema, foi utilizada a metaheurística *Iterated Local Search (ILS)* com *Variable Neighbourhood Descent (VND)*. No ILS-VND, a escolha da sequência de estruturas de vizinhanças a ser explorada impacta diretamente o resultado. Devido a isso, o *Random VND* e o *Multi-Armed Bandit* foram escolhidos para tomar decisões mais acertadas a respeito da ordem exploração. O objetivo do trabalho é verificar se essas variações com VND são boas alternativas. Dentre os resultados obtidos, ao escolher uma sequência inicial ruim tanto o Bandit quanto o RVND superam os resultados do VND. O Bandit em comparação ao RVND apresenta uma mediana superior em 63% dos casos testados e, em 56% dos casos, seus resultados piores foram maiores que os piores resultados obtidos com o RVND.

PALAVRAS CHAVE. Variable Neighborhood Descent, Multi-Armed Bandits, Team Orienteering Problem.

Tópicos: OC – Otimização Combinatória, IC – Inteligência Computacional

ABSTRACT

This work tackles the team orienteering problem, a NP-hard combinatory optimization problem. For solving the problem, we discuss a metaheuristics algorithm, the Iterated Local Search and Variable Neighbourhood Descent. In the ILS-VND the sequence order impact the final result. For that two approachs were chosen; the Random VND and the Multi-Armed Bandit for making a good choice for the sequence. We aim to verify if these VND variations are good alternatives for the problem encountered. In the results obtained, Bandit and RVND surpass a poorly chosen neighbourhood sequency. The median results obtained using Bandit are better than RVND in 63% of the cases tested, and in 56% of the cases, it's worst results have better values.

KEYWORDS. Variable Neighbourhood Descent, Multi-Armed Bandits, Team Orienteering Problem.

Topics: OC – Combinatorial Optimization, IC – Computational Intelligence

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

1. Introdução

O problema de orientação de times, em inglês *Team Orienteering Problem (TOP)*, formalizado no trabalho de Chao et al. [1996], dois anos após de ser introduzido inicialmente no trabalho de Butt e Cavalier [1994] com o nome de *Multiple Tour Maximum Collection Problem (MTMCP)*. é um problema derivado do problema de orientação proposto por Golden et al. [1987].

A orientação é um esporte em que o participante recebe uma búlsula e um mapa. O objetivo desse esporte é conseguir coletar o melhor conjunto de recompensas possível num período limitado de tempo. Neste esporte o indivíduo tem o ponto de largada e chegada fixos e outros pontos disponíveis que têm uma recompensa única e uma vez visitados não é mais possível coletar novamente sua recompensa. O problema de orientação é um problem NP-difícil, o que faz com que o TOP também seja NP-difícil [Butt e Cavalier [1994]].

O trabalho de Macedo e Senne [2023] explora uma nova maneira de resolver esse problema, usando o *Iterated Local Search (ILS)* [Lourenço et al. [2001]] com *Variable Neighborhood Descent(VND)* , cujo objetivo é gerar uma solução satisfatória para ser a solução inicial do algoritmo *Fix-and-Optimize (F&O)* [Gintner e Kliewer [2005]]. O ILS utilizado em Macedo e Senne [2023] só faz experimentos com o VND, o objetivo deste trabalho é analisar possíveis impactos no resultado final ao variar o VND por *Random-VND (RVND)* [Mladenović e Hansen [1997]], e utilizando o *Bandit* [Slivkins [2024]] para escolher automaticamente uma ordem de estrutura de vizinhança para o VND.

O trabalho de Lü et al. [2011] procura entender a influência de cada estrutura de vizinhança na busca local e propõe *Multiple Neighbourhood Search (U-VND)*. No U-VND uma única vizinhança é obtida como da união de múltiplas estruturas de vizinhança predefinidas. Wu et al. [2012] utiliza também essas múltiplas estruturas de vizinhança na busca TABU. Já em [Şevkli e Aydin [2006]], são analisadas variantes sequenciais de estruturas de vizinhança para resolver o problema do caixeiro viajante.

O restante deste artigo está organizado da seguinte maneira: a Seção 2 contém a formulação matemática do algoritmo, a Seção 3 apresenta a metodologia utilizada, na Seção 4 os resultados obtidos são analisados e a Seção 5 é a conclusão e trabalhos futuros.

2. Formulação Matemática

A modelagem escolhida segue a definição proposta em Bianchessi et al. [2018], ou seja, este trabalho modela o problema como um grafo direcionado completo $G = (V, A)$, onde $A = \{(i, j)|i \neq n + 1; j \neq 0; i \neq j; i, j \in V\}$ é o conjunto de arcos e $V = N \cup \{0, n + 1\}$ é o conjunto de nós a serem percorridos (N) união com o nós de origem e destino $\{0, n + 1\}$.

Os n pontos possíveis para visitação estão contidos no conjunto N, onde $N = \{1, ..., n\}$. Cada nó $i \in V$, possui um respectivo prêmio p_i, com $p_0 = p_{n+1} = 0$. A cada arco $(i, j) \in A$ um tempo não negativo t_{ij} é associado. No TOP, um time com K integrantes tem como objetivo conseguir a maior pontuação possível. Cada caminho percorrido precisa começar no nó 0 e terminar no nó $n + 1$ respeitando um tempo pré-definido máximo, T_{max} e cada prêmio p_i só pode ser recolhido uma única vez.

Baseando-se no fluxo de entradas e saídas de dois índices, a formulção tem as variáveis binárias $x_{ij}, (i, j) \in A$, $y_i \in N$, onde $y_i = 1$, se o arco (i, j) é percorrido e o nó i é visitado. As variáveis contínuas $z_{ij} \in A\backslash\{0, n + 1\}$, representam o tempo de chegada no vértice j vindo do vértice i. Para cada $i \in N$, seja $\delta^+(i) = \{(i, j) \in A|j \neq i, j \in N\}$ e $\delta^-(i) = \{(j, i) \in A|i \neq j, j \in N\}$, o conjunto de arcos saindo e entrando no vértice *i*. A modelagem:

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

$$Max(-\sum_{i \in N} p_i y_i) \tag{1}$$

$$\sum_{j \in N} x_{0j} = K \tag{2}$$

$$\sum_{i \in N} x_{i,n+1} = K \tag{3}$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji} = y_i, \qquad \forall i \in N \tag{4}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} = y_i, \qquad \forall i \in N \tag{5}$$

$$z_{0j} = t_{0j} x_{0j}, \qquad \forall j \in N \tag{6}$$

$$\sum_{(i,j) \in \delta^+(i)} z_{ij} - \sum_{(j,i) \in \delta^-(i)} z_{ji} = \sum_{(i,j) \in \delta^+(i)} t_{ij} x_{ij}, \qquad \forall i \in N \tag{7}$$

$$z_{ij} \leq (T_{max} - t_{j,n+1}) x_{ij}, \qquad \forall (i,j) \in A \backslash \{(0, n+1)\} \tag{8}$$

$$z_{ij} \geq (t_{0i} + t_{ij}) x_{ij}, \qquad \forall (i,j) \in A \backslash \{(0, n+1)\} \tag{9}$$

$$y_i \in \{0,1\}, \qquad \forall i \in N \tag{10}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{11}$$

$$t_{0,0} = t_{n+1,n+1} = 0 \tag{12}$$

A função objetivo (1) maximiza a soma dos prêmios coletados. As Restrições (2) e (3) garante que existam exatamente K caminhos que saem do ponto inicial e K caminhos que chegam ao ponto final. As famílias de restrições (4) e (5) forçam que exista uma única entrada e uma única saída para todos os nós visitados. A família de restrições (6) limita o fluxo originado no ponto inicial, a família de restrições (7) representa a conservação do fluxo; a família de restrições (8) limita o tempo de cada caminho; já a família de restrições (9) impõe um limite inferior para os valores das variáveis z, com o objetivo de restringir o número de valores viáveis que essas variáveis poderiam assumir. E as famílias de restrições (10) e (11) se referem ao domínio das variáveis, e a Restrição (12) define o tempo de viagem no ponto de saída e no ponto de chegada.

3. Metodologia

O algoritmo utilizado é baseado no trabalho de Macedo e Senne [2023], que constrói a solução em três fases; a primeira é denominada construtivo enquanto que as duas seguintes são chamadas de otimização. Na primeira fase uma solução viável para o problema é construída. Na segunda e terceira fases o resultado obtido nas respectivas etapas anteriores é utilizado como ponto inicial, a fim de melhorar a solução encontrada. O algoritmo apresentado no trabalho deles, foi reproduzido da maneira mais fiel possível a fim de se verificar como diferentes meta-heurísticas poderiam influenciar no resultado final.

O restante desta seção é dividido nas seguintes subseções: a Subseção 3.1 descreve como a solução inicial é construída; a Subseção 3.2 cita as estruturas de vizinhanças utilizadas; e, por fim, a Subseção 3.3 descreve o algoritmo ILS + VND e suas variações.

3.1. Construção

Apesar do artigo de Macedo e Senne [2023] citar diferentes maneiras de construir a solução inicial para o problema, nos resultados finais analisam apenas a maneira mais vantajosa. Para fins de comparação, este trabalho adota o mesmo critério.

A fase de construção da solução começa pelo pré-processamento proposto por Chao et al. [1996]. Nele, uma elipse de eixo máximo T_{max} com pontos focais sendo os pontos de partida e chegada é construída. Todos os pontos fora desta elipse são eliminados. O objetivo desta restrição é descartar automaticamente candidatos que ultrapassariam o tempo máximo permitido para o percurso.

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

Após esta eliminação de pontos os K caminhos são inicializados. A inicalização acontece escolhendo-se pontos de dentro da elipse. Estes pontos são organizados em uma lista, do mais distante ao menos distante do ponto inicial. Os K caminhos recebem o ponto inicial e o ponto final. Após isso os pontos de dentro da elipse são selecionados baseando-se na métrica proposta por Subramanian et al. [2013]:

$$z_1 = t_{i,\omega} + t_{\omega,j} - t_{i,j} \tag{13}$$

$$z_3 = z_1 - \gamma(t_{0,\omega} + t_{\omega,0}), \gamma \in [0,1] \tag{14}$$

A equação 13 representa a melhor economia de inserção do vértice ω entre dois pontos ao se considerar todas as rotas. Na equação 14, ao valor encontrado em 13 é subtraído a distância de ida e volta do ponto ω ao ponto inicial. Escolher a equação 14 permite com que pontos mais distantes do início possam ser incluídos na solução inicial, contribuindo para sua diversificação. Os pontos continuam a ser inseridos até que não se possa mais fazê-lo. O procedimento completo é descrito em pseudocódigo no Algoritmo 1.

Algorithm 1: Construtiva Melhor Economia

```
1  Data: G(V, A), γ, K, n, T_max, t, N
2  begin
3      path ← ∅;
4      time ← ∅ ;
5      for i in {1..K} do
6          path ← [V_0, V_{n+1}];
7          time ← t_{0,n+1} ;
8      end
9      for i in N do
10         z_3 ← T_max;
11         best ← ∅;
12         for j in {1...K} do
13             for m in {0...len(path[j])-1} do
14                 p_1 ← path[j][m];
15                 p_2 ← path[j][m + 1];
16                 t ← t_{p_1,i} + t_{i,p_2} − t_{p_1,p_2};
17                 z = t − γ(t_{0,i} + t_{i,0});
18                 if t + time[j] < T_max and z_3 > z then
19                     z_3 = z;
20                     best ← [j, m + 1, t];
21                 end
22             end
23         end
24         if best ≠ ∅ then
25             j, m, t ← best;
26             path[j].insert(m, i);
27             time[j] ← time[j] + t;
28         end
29     end
30     return path, time
31 end
```

3.2. Estruturas de Vizinhança

As estruturas de vizinhança utilizadas neste trabalho foram: Inserção, Permutação, 2-OPT e Substituição. O critério de aceite para inserir e substituir é a melhora da função objetivo. Já o

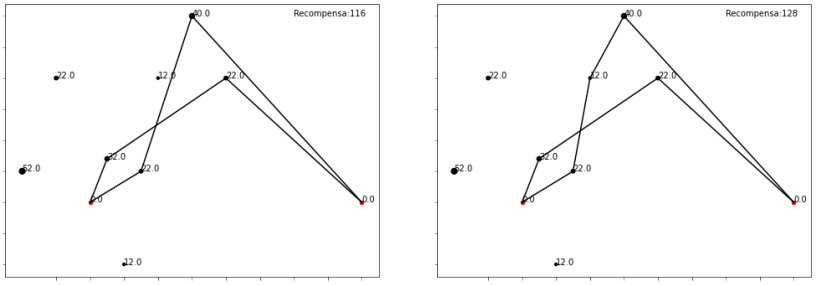
DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

critério para as funções de permutação e 2-opt [Croes [1958]] são baseados na melhora do tempo do caminho, se um tempo menor for encontrado a solução é atualizada.

As Subseções 3.2.1, 3.2.2, 3.2.3 e 3.2.3 explicam como foram feitas as estrutura de vizinhança da inserção, substituição, permutação e 2opt respectivamente.

3.2.1. Inserção

Na busca local de inserção verifica-se dentre todos os nós não visitados e todas as posições possíveis de inserção o valor de z3, calculado pela equação 14 . O vértice escolhido é o que possui o melhor custo-benefício, ou seja, o menor valor de z3 dentre todas as opções [Macedo e Senne [2023]]. A Figura 1 mostra uma inserção do vértice com recompensa 12 no caminho original, 1b, aumentando a recompensa para 128.



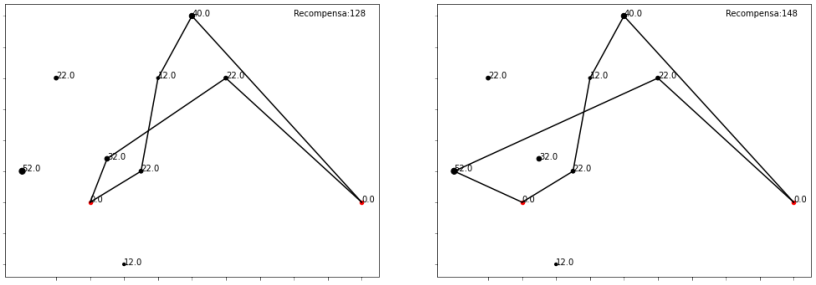
(a) Solução atual

(b) Solução após inserir o vértice com recompensa 12

Figure 1: Inserção

3.2.2. Substituição

A substituição é calculada de uma maneira similar a inserção. Nela são verificadas dentre as posições ocupadas pelos vértices presentes nos caminhos, qual o par posição+vértice-não-visitado é mais vantajoso, segundo o melhor valor de $z3$. Em todos os casos o novo vértice só será aceito caso o tempo máximo seja respeitado. A Figura 2b mostra o vértice visitado de valor 32 em um dos caminhos sendo trocado pelo vértice não visitado de valor 52, aumentando a recompensa do time.



(a) Solução atual

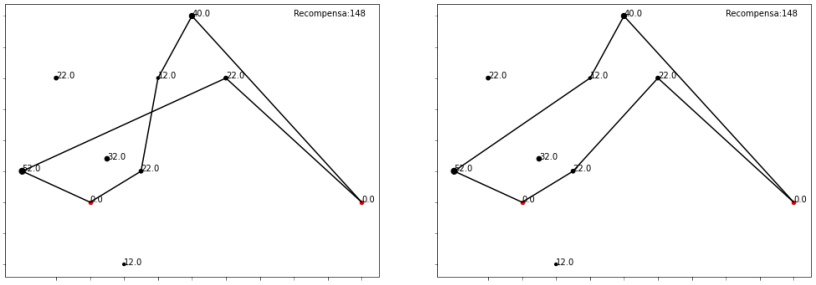
(b) Solução após a substituição do vértice 32 pelo 52

Figure 2: Substituição

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

3.2.3. Permutação

Na permutação o algoritmo procura trocar dois vértices de caminhos distintos. O objetivo é que ao trocar esses vértices o tempo total de cada caminho seja reduzido [Chao et al. [1996]]. Na Figura 3a, temos dois caminhos com as seguintes ordens de recompensa; {0, 22, 12, 40, 0}, {0, 52, 22, 0}, nesta permutação os vértices de recompensa 12 e 22 são escolhidos para serem trocados, resultando na seguintes sequências: {0, 22, 22, 40, 0}, {0,52,12,0}, representados na Figura 3b.



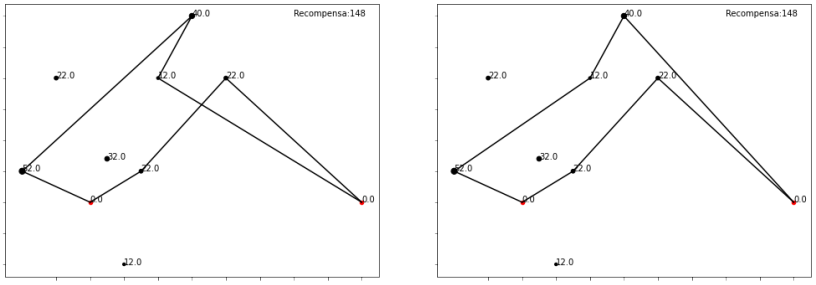
(a) Solução atual

(b) Solução após troca de vértices

Figure 3: Permutação

3.2.4. 2opt

Para tentar uma redução de tempo dentro de cada caminho o algoritmo 2-opt [Croes [1958]] é aplicado. O objetivo é que ao diminuir o tempo do caminho, na próxima iteração será possível adicionar novos pontos. Como mostra a Figura 4a, que antes de aplicar o algoritmo possui um tempo maior que após a aplicação do 2opt resulta na solução da Figura 4b.



(a) Solução atual

(b) Solução após 2opt

Figure 4: 2opt

3.3. Busca Local

Nesta seção serão apresentadas algumas variações de buscas locais utilizadas no trabalho. A Seção 3.3.1 apresenta o VND, a Seção 3.3.2, explica a variação do VND com sorteio de sequência de estruturas de vizinhança aleatória e a Seção 3.3.3 explica como o algoritmo de *Multi-Armed Bandit* foi utilizado para escolher a ordem de vinhança.

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

3.3.1. Variable Neighbourhood Descent

O VND é uma heurística para resolver problemas de otimização combinatória e otimização global. Esta heurística utiliza trocas determinísticas de vizinhança na busca local para melhorar a solução [Hansen et al. [2010]].

Utilizando as estruturas de vizinhança apresentadas na Seção 3.2, uma sequência possível seria: Inserção-Permutação-Subtituição-2opt. Existe a possibilidade desta não ser a melhor ordem para uma determinada instância [Şevkli e Aydin [2006]]. Por esta razão este trabalho testa algumas variações do VND, com o objetivo de automatizar a escolha desta ordem.

A hipótese inicial do trabalho é que existem ordens de estruturas de vizinhança melhores que outras e, consequentemente, existem piores. Pode-se ter o azar de escolher uma ordem ruim para a busca local. Como o RVND sorteia aleatoriamente essas ordens, é esperado que sua performance seja melhor que as piores sequências, mas seja pior que as melhores. Para o Bandit-VND é esperado uma performance melhor que o RVND, já que parte das escolhas é feita com a melhor ação.

3.3.2. Random-VND

No *Random-VND* a cada execução da busca local, a ordem das estruturas de vizinhança é embaralhada [Oliveira [2021]]. Nas estruturas de vizinhança Inserção e Substituição a busca local escolhe o primeiro aprimorante, enquanto na de Permutação e 2opt é a primeira melhora.

3.3.3. Bandit-VND

O *Bandit-VND* utiliza o *Multi-Armed Bandits* para escolher a ordem de vizinhanças da busca local. O algoritmo *Multi-Armed Bandits* possui K possíveis ações, que são as armas, e T iterações. Em cada iteração o algoritmo escolhe uma ação e coleta uma recompensa. A recompensa é obtida a partir de uma distribuição Identicamente Independente, fixada, que depende somente da escolha da ação, mas é desconhecida para o algoritmo [Slivkins et al. [2019]].

Como o algoritmo só tem informação depois da escolha de cada ação, ele precisa de duas fases; uma de exploração, escolhendo ações e coletando as respectivas recompensas, e outra de extrapolação que escolhe a ação baseando-se nas informações coletadas anteriormente. Isso faz com que exista um *tradeoff* no algoritmo, o quanto de T deve-se gastar na exploração para que uma boa escolha possa ser feita na extrapolação.

Para este trabalho a recompensa escolhida é 1, caso a sequência obtenha sucesso, e 0, caso contrário. No *stochastic bandit*, utilizado no trabalho, a fase de exploração utiliza uma porcentagem de T para coletar recompensas. Na fase extrapolação a escolha da ação é baseada na ação que obteve a melhor recompensa média na fase de exploração, com desempate arbitrário.

3.4. Iterated Local Search - VND

A segunda fase do algoritmo proposto por Macedo e Senne [2023] é composta pelo ILS-BuscaLocal, *Algorithm* 2. Neste trabalho, a Busca Local são as variações do VND. Aqui o algoritmo começa na solução viável obtida com o construtivo. O ILS se baseia em uma melhora e em uma perturbação. Na etapa da perturbação o algoritmo sorteia aleatoriamente um vértice visitado e tenta retirá-lo. Retirar um vértice pode causar uma inviabilidade na solução, por isso uma operação de 2-opt é realizada logo em seguida para reduzir o tempo do caminho. Após essa etapa é verificado se a solução é viável e, caso não seja, um novo vértice é sorteado. Os vértices continuam a ser sorteados até que se encontre um que não cause inviabilidade na solução, ou que todos tenham sido sorteados.

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

Algorithm 2: ILS-BuscaLocal

```
1  begin
2      s* ← Solução Inicial
3      while Max_Iter do
4          s' ← perturbação de s*
5          s" ← BuscaLocal(s')
6          if f(s") > f(s*) then
7              s* ← s"
8          end
9      end
10     return s*
11 end
```

4. Experimentos Computacionais

Esta seção apresenta os resultados experimentais obtidos. Ela está organizada em duas subseções. A Seção 4.1 apresenta detalhes dos hiper-parâmetros e as instâncias utilizadas para os experimentos, enquanto que a Seção 4.2 apresenta e analisa os resultados obtidos.

Os experimentos foram implementados na linguem de programação Python versão no Sistema Operacional Linux, usando a versão 3.8. Os experimentos foram executados dez vezes para cada instância em um Intel(R) Core(TM) i5-8265U de frequência de 1.60GHz com 4GB RAM.

4.1. Instâncias e Configurações dos Algoritmos

As instâncias utilizadas neste trabalho foram descritas por Dang et al. [2013] e disponibilizadas por Hammami et al. [2020]. Foram realizadas 10 repetições para cada instância. O número de iterações escolhido foi 10^4. O γ escolhido da Equação 14, foi fixado em 0.1. O percentual para a fase de exploração do Bandit foi fixado em 70%.

A fim de comparar como seriam os resultados caso uma escolha ruim de sequência de vizinhanças fosse feita. Para cada par $\{\text{instância}_j, \text{permutação}_i\}$ foram realizadas 5 repetições, com 10^2 iterações. A partir desses resultados foi obtida uma média relacionada ao par. Para cada instância os experimentos utilizaram a permutação que teve seu resultado médio pior.

4.2. Resultados Computacionais

Nesta seção os resultados obtidos a partir de cada algoritmo são comparados. Para fazer essa análise, foi utilizada a métrica de distância dada pela Equação (15), onde $xvnd \in \{$ILS+VND, ILS+RVND, ILS+BVND$\}$ e o $otimo$ é o melhor valor encontrado, que foi retirado da literatura. Como os dados não são normalmente distribuídos, o teste de hipótese utilizado foi o *Wilcoxon Signed Rank Test*.

$$m = 1 - \frac{xvnd}{otimo} \tag{15}$$

Os resultados obtidos no teste de hipóteses são apresentados na Tabela 1. A primeira coluna contém as hipóteses e a segunda coluna os seus p-valores. Na hipótese nula, H_0 é assumido que não há diferença entre os valores obtidos a partir de dois algoritmos diferentes, e a hipótese alternativa, H_1, é assumido que os valores da métrica em um dos algoritmos é maior no outro. Como está a métrica usada é a distância até o melhor valor, aceitar a hipótese H_1, significa que o algoritmo é pior que o outro, já que ele está mais distante do ótimo.

A partir da Tabela 1 conclue-se que o Bandit e o RVND são preferíveis ao VND, já que no VND pode ser escolhido uma estrutura de vizinhanças ruim, o que não acontece para o Bandit ou RVND. Na Tabela 2, é possível observar que a mediana do Bandit supera em 60% a mediana do

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

RVND, que pode ser justificado devido a aleatoriedade do RVND, causando resultados melhores, mas também piores, como os obtidos em 56% das instâncias. Havendo um ajuste na porcentagem de iterações utilizada na fase de exploração do Bandit, os resultados podem ser melhorados.

A última tabela, a Tabela 3, apresenta as médias e desvio padrão da pontuação obtida em cada um dos algoritmos. O campo *Instância*, indica a instância utilizada no experimento. No campo *V Avg(Std)*, V é o VND com uma sequência de estrutura de vizinhança ruim, no campo *B Avg(Std)*, B é Bandit e no campo *R Avg(Std)*, R é o RVND. O *Avg(Std)*, é a respectiva média e desvio padrão da pontuação obtida e o Avg Time, o tempo médio de execução.

Table 1: Testes de Hipótese

Hipótese	**p-valor**
H_0: $\mu_{-vnd} = \mu_{rvnd}$ H_1: $\mu_{-vnd} > \mu_{rvnd}$	0.002
H_0: $\mu_{-vnd} = \mu_{bandit}$ H_1: $\mu_{-vnd} > \mu_{bandit}$	0.002
H_0: $\mu_{-vnd} = \mu_{+vnd}$ H_1: $\mu_{-vnd} > \mu_{+vnd}$	0.002

Table 2: Resultados

Algoritmo 1	**Algoritmo 2**	**Best Alg1 $\geq$ Alg2**	**Worst Alg1 $\geq$ Alg2**	**Median Alg1 $\geq$ Alg2**
ILS-Bandit	ILS-VND	0.86	0.98	0.93
IlS-Bandit	ILS-RVND	0.51	0.56	0.63
ILS-RVND	ILS-VND	0.84	0.97	0.91

5. Conclusão e trabalhos futuros

Este trabalho apresentou uma análise comparativa entre os algoritmos ILS com Busca Local. As buscas locais exploradas foram o VND, RVND e Bandit VND. Foi abordado também como cada um desses algoritmos é construído, o que inclui a heurística construtiva para gerar soluções iniciais e a perturbação utilizada para gerar soluções vizinhas.

Os experimentos computacionais mostraram que é vantajoso optar pelo uso do Bandit VND ou RVND, ao VND já que não é necessário escolher uma sequência de estruturas de vizinhanças. Em particular o Bandit VND possui a mediana dos resultados melhor em 63% das instâncias, se comparado ao RVND e em 56% das instâncias o pior resultado tem valor maior comparado ao pior resultado obtido com o RVND.

Para trabalhos futuros, no Bandit seria verificado o quanto a porcentagem utilizada na fase de exploração influencia os resultados, e propor um algoritmo de aprendizado mais eficiente, como o *Q-Learning*, para aprender qual seria uma boa ordem de estruturas de vizinhança dado uma instância qualquer.

Agradecimentos

Os autores agradecem o UFRJ (23079.227622/2023-70), CNPq (307663/2021-3) , FAPERJ (307663/2021-3), e CAPES(88887.948034/2024-00) pelo financiamento parcial desta pesquisa.

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

Instância	VND Av(Std)	Tmp Avg VND	RVND Avg(Std)	Tmp Avg RVND	B-VND Avg(Std)	Tmp Avg B-VND
eil101b_m3	770(49)	107.11	829(15)	33.57	829(10)	46.94
cmt101c_m3	1152(66)	159.84	1207(42)	36.27	1203(40)	55.39
eil101c_m3	1096(28)	219.12	1138(20)	38.94	1122(25)	58.18
gil262a_m4	2511(237)	232.1	2946(51)	103.09	2952(31)	148.51
bier127_gen2_m4	4526(296)	239.87	4799(57)	52.35	4830(62)	76.97
pr264_gen1_m4	94(6)	240.56	102(6)	78.06	103(1)	95.54
kroB200_gen3_m3	2293(162)	412.31	2722(80)	184.53	2688(86)	159.71
gil262_gen1_m4	68(3)	373.99	75(1)	209.03	75(1)	198.14
bier127_gen1_m3	92(6)	420.65	97(1)	60.03	98(1)	83.8
kroA150_gen2_m2	3533(198)	508.21	3882(58)	100.1	3784(108)	95.12
rat195_gen2_m2	4000(309)	886.6	4375(108)	147.59	4405(94)	150.03
eil101c_m2	1165(38)	648.07	1175(26)	46.15	1184(17)	63.93
cmt200b_m3	1655(80)	668.04	1756(51)	125.48	1769(19)	193.09
cmt151c_m3	1673(59)	693.23	1712(22)	81.19	1730(23)	112.24
kroB200_gen3_m2	3362(332)	1005.28	3692(317)	191.58	3792(167)	169.09
pr264_gen3_m3	2368(103)	993.43	2370(86)	279.88	2442(113)	274.9
pr264_gen2_m3	5347(471)	988.93	5800(198)	349.27	5820(249)	279.73
pr299_gen1_m4	69(5)	360.0	80(1)	210.16	80(1)	191.63
gil262b_m3	5891(289)	1093.9	6319(211)	202.86	6358(183)	286.76
bier127_gen3_m2	2316(206)	1080.4	2505(133)	1046.74	2489(136)	107.99
cmt151b_m3	1164(65)	310.24	1238(22)	68.9	1230(22)	98.99
pr264_gen2_m4	4733(262)	257.62	5223(250)	79.26	5202(253)	77.47
gr229_gen2_m4	10995(86)	2220.3	10872(160)	231.64	10922(99)	224.88
bier127_gen3_m4	1889(242)	209.95	2155(59)	56.21	2153(43)	77.59
gil262b_m2	6239(273)	2931.38	6600(150)	238.52	6636(255)	335.21
gr229_gen3_m4	7029(370)	2267.23	6980(191)	229.96	7057(181)	227.88
rat195_gen3_m3	2092(219)	351.79	2362(52)	132.39	2361(35)	131.04
gil262_gen2_m2	5835(500)	2467.26	6052(228)	346.59	5993(112)	320.89
pr299_gen1_m3	96(3)	893.63	103(1)	337.1	101(2)	297.45
rd400_gen3_m4	7758(282)	2018.42	8294(405)	810.09	8373(316)	814.98
bier127_gen3_m3	2039(129)	386.82	2330(140)	59.59	2257(114)	83.98
gil262_gen3_m4	1995(253)	342.63	2396(38)	258.96	2395(49)	205.06
rd400_gen1_m4	172(9)	2206.05	189(6)	812.95	188(5)	865.42
lin318_gen3_m2	5955(453)	4702.23	6167(430)	552.46	6295(262)	520.82
rd400_gen3_m3	8909(301)	3889.42	9587(362)	883.58	9007(486)	860.1
kroB200_gen2_m2	4537(477)	1162.7	5162(304)	194.45	5235(342)	178.02
gil262_gen1_m3	82(7)	608.8	93(2)	373.14	92(2)	242.58
gil262_gen3_m2	5115(370)	2184.13	5843(156)	312.89	5712(203)	300.5
bier127_gen2_m3	4750(235)	462.9	4909(60)	60.03	4904(41)	84.66
pr299_gen3_m4	1872(112)	350.49	2144(37)	192.74	2157(31)	196.31
kroB200_gen1_m2	82(8)	1067.32	98(4)	184.34	98(5)	173.46
rd400_gen2_m4	9354(380)	2267.95	9683(271)	931.06	9590(330)	819.63
kroB200_gen2_m4	4066(172)	286.46	4598(40)	150.49	4624(152)	152.82
cmt200c_m2	2480(50)	4935.52	2516(29)	189.42	2536(26)	248.26
bier127_gen1_m2	98(3)	1369.97	99(1)	77.84	99(1)	108.45
lin318_gen2_m3	5574(666)	1322.11	6990(260)	430.73	6784(198)	409.49
pr136_gen1_m2	49(5)	271.22	60(1)	77.51	59(1)	76.81
gr229_gen2_m3	11109(17)	4707.04	11086(19)	297.4	11110(15)	290.22
gr229_gen3_m3	7392(180)	5340.69	7378(77)	302.1	7378(60)	291.58
gil262_gen2_m3	4451(340)	642.45	4935(114)	255.19	4961(102)	255.88
gil262c_m3	8591(405)	2773.26	9076(270)	250.59	9170(327)	338.24
pr299_gen3_m3	2971(165)	844.12	3215(63)	313.01	3189(47)	310.12
kroA200_gen1_m4	66(5)	268.16	78(1)	150.32	78(1)	136.93
rd400_gen2_m3	9719(143)	4067.22	10122(205)	860.27	10332(260)	908.74
lin318_gen1_m3	111(11)	1331.09	136(7)	419.87	133(8)	413.82
pr299_gen3_m2	3999(481)	2313.44	4816(256)	378.36	4721(283)	369.41
cmt200c_m3	2370(73)	1658.06	2462(62)	149.53	2484(36)	203.71
gil262a_m2	3122(137)	590.76	3542(200)	124.99	3473(146)	177.06
pr299_gen1_m2	100(18)	2216.88	125(4)	478.89	122(6)	392.86
pr299_gen2_m4	3659(284)	391.77	4165(76)	241.81	4157(74)	195.08
cmt200b_m2	1740(60)	1886.17	1824(26)	145.29	1840(51)	200.26
gr229_gen1_m4	215(5)	2458.01	214(2)	230.07	214(1)	225.0
pr136_gen2_m2	2681(397)	277.36	3190(128)	76.21	3244(68)	77.02
cmt151c_m4	1591(53)	334.21	1696(30)	73.46	1693(28)	101.91
rd400_gen2_m2	9896(307)	7707.93	10245(190)	968.1	10230(199)	1019.66
gil262c_m4	8162(333)	1216.53	8772(139)	224.25	8854(295)	300.06
cmt200c_m4	2357(52)	787.63	2417(48)	131.19	2467(40)	178.29
ts225_gen2_m2	4483(552)	1196.99	5137(182)	197.72	5012(165)	186.86
kroA150_gen3_m3	2077(198)	200.09	2506(61)	85.08	2527(58)	84.08
bier127_gen2_m2	4907(192)	1268.95	5074(92)	374.09	5016(51)	109.28
pr299_gen2_m3	4901(279)	901.05	5258(87)	336.3	5320(79)	302.45
cmt200b_m4	1576(105)	394.26	1723(40)	116.61	1711(40)	164.69
rd400_gen1_m3	175(8)	3136.84	194(6)	886.98	197(4)	863.15
lin318_gen3_m4	2935(229)	768.91	3351(229)	371.6	3342(230)	357.53
gil262b_m4	5406(288)	627.79	5760(202)	181.32	5970(97)	249.45
pr264_gen2_m2	6308(305)	2114.66	6430(16)	434.7	6416(42)	360.19
lin318_gen1_m2	141(10)	3699.29	153(6)	526.44	154(7)	504.39
cmt151c_m2	1701(51)	1370.26	1768(41)	100.69	1735(24)	135.81
gil262c_m2	9014(270)	4450.37	9408(171)	322.49	9539(171)	447.43
lin318_gen2_m2	7079(470)	2608.0	7874(368)	630.36	7923(233)	527.98
rd400_gen3_m2	9436(572)	5667.35	9868(402)	952.44	9780(274)	989.63
rd400_gen1_m2	183(3)	5569.09	199(2)	970.97	200(2)	954.75

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

References

Bianchessi, N., Mansini, R., e Speranza, M. (2018). A branch-and-cut algorithm for the team orienteering problem. *International Transactions in Operational Research*, 25:627–.

Butt, S. E. e Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers Operations Research*, 21(1):101–111. ISSN 0305-0548. URL https://www.sciencedirect.com/science/article/pii/0305054894900655.

Chao, I.-M., Golden, B. L., e Wasil, E. A. (1996). The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474. URL https://EconPapers.repec.org/RePEc:eee:ejores:v:88:y:1996:i:3:p:464-474.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6): 791–812. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/167074.

Dang, D.-C., Guibadj, R., e Moukrim, A. (2013). An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229:332–344.

Gintner, V. e Kliewer, N. (2005). Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *Operations Research-Spektrum*, 27:507–523.

Golden, B., Levy, L., e Vohra, R. (1987). The orienteering problem. *Nav Res Logist*, 34:307–318.

Hammami, F., Rekik, M., e Coelho, L. C. (2020). A hybrid adaptive large neighborhood search heuristic for the team orienteering problem. *Computers Operations Research*, 123:105034. ISSN 0305-0548. URL https://www.sciencedirect.com/science/article/pii/S0305054820301519.

Hansen, P., Mladenovic, N., e Moreno-Pérez, J. (2010). Variable neighbourhood search: Methods and applications. *4OR*, 175:367–407.

Lourenço, H., Martin, O., e Stützle, T. (2001). A beginner's introduction to iterated local search. p. 1–11.

Lü, Z., Hao, J.-K., e Glover, F. (2011). Neighborhood analysis: A case study on curriculum-based course timetabling. *J. Heuristics*, 17:97–118.

Macedo, E. A. A. G. e Senne, E. L. F. (2023). Hybrid approach to solve the team orienteering problem. In *Anais do Simpósio Brasileiro de Pesquisa Operacional.*, Campinas. Galoá.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers Operations Research*, 24(11):1097–1100. ISSN 0305-0548. URL https://www.sciencedirect.com/science/article/pii/S0305054897000312.

Oliveira, J. P. F. (2021). Iterated local search aplicado ao problema de roteamento de veĺculos com coleta e entrega simultÂnea, janela de tempo e frota heterogÊnea. Master's thesis, Universidade Federal de Ouro Preto – UFOP.

Slivkins, A. (2024). Introduction to multi-armed bandits.

Slivkins, A. et al. (2019). Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en

Subramanian, A., Penna, P., Ochi, L., e Souza, M. (2013). *Um Algoritmo Heurístico Baseado em Iterated Local Search para Problemas de Roteamento de Veículos*, p. 165–180. ISBN 9788564619104.

Wu, Q., Hao, J.-K., e Glover, F. (2012). Multi-neighborhood tabu search for the maximum weight clique problem. *Annals of Operations Research*, 196.

Şevkli, M. e Aydin, M. (2006). Variable neighbourhood search for job shop scheduling problems. *JSW*, 1:34–39.

DOI: 10.59254/sbpo-2024-193342
https://proceedings.science/p/193342?lang=en