LIII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2021

Um algoritmo genético com busca local para o problema do caixeiro viajante com veículo elétrico híbrido e janelas de tempo

Yure Rocha, Anand Subramanian
Universidade Federal da Paraíba - Centro de Informática
Rua dos Escoteiros, s/n, Mangabeira, 58055-000, João Pessoa/PB, Brasil
yuretalis@gmail.com, anand@ci.ufpb.br

RESUMO

Este trabalho aborda o problema do caixeiro viajante com veículo elétrico híbrido e janelas de tempo. O objetivo do problema é realizar um ciclo hamiltoniano de custo mínimo, com início e término no depósito, satisfazendo restrições de tempo e de carga da bateria. Para solucioná-lo, foi proposto um algoritmo genético combinado a um procedimento de busca local hierárquica. Foram implementados dois procedimentos para acelerar a etapa de busca local, cada um lidando com um tipo específico de restrição, além de um método de geração de descendentes baseado no *crossover OX*. Experimentos computacionais foram realizados em 216 instâncias disponíveis na literatura e 33 soluções melhores foram encontradas. De uma forma geral, o algoritmo proposto foi eficaz em encontrar soluções de qualidade igual ou superior às melhores existentes em um menor tempo computacional.

PALAVRAS CHAVE. Problema do caixeiro viajante. Veículo elétrico híbrido. Algoritmo genético.

Tópicos: L&T – Logística e Transportes, MH – Meta-heurísticas

ABSTRACT

This paper addresses the traveling salesman problem with hybrid electric vehicle and time windows. The objective of the problem is to perform a minimum cost Hamiltonian cycle, starting and ending at the depot, satisfying time and battery charging constraints. To solve it, we proposed a genetic algorithm combined with a hierarchical local search procedure. We implemented two procedures in order to speed-up the local search phase, each dealing with a specific type of constraint, as well as a crossover OX-based offspring generation procedure. Computational experiments were conducted on 216 instances available in the literature and 33 improved solutions were found. In general, the proposed algorithm was effective in finding solutions with quality equal or superior to the best existing ones using less computational time.

KEYWORDS. Traveling salesman problem. Hybrid electric vehicle. Genetic algorithm.

Paper topics: L&T – Logistics and Trasportation, MH – Metaheuristics

1. Introdução

As atividades de logística levam em conta cada vez mais os impactos sociais e ambientais causados pela utilização de veículos com motores de combustão interna. A preocupação não é somente no âmbito econômico dessas atividades, mas também com os seus efeitos a longo prazo no meio ambiente [Sbihi e Eglese, 2007] devido à emissão de gases de efeito estufa e também da poluição sonora causada [Dekker et al., 2012]. Diante disso, juntamente com a obrigação legal através de regulamentações, transportadoras têm considerado a utilização de veículos com fontes de propulsão alternativas para entregas finais [Schneider et al., 2014].

Veículos elétricos são uma alternativa mais amigável ao meio ambiente, visto a pequena quantidade de emissão de gases (se comparada à emissão dos veículos tradicionais), porém têm uma limitação na distância máxima percorrida [Dekker et al., 2012]. Os veículos elétricos híbridos (do inglês *hybrid electric vehicles* – HEVs) apresentam uma solução razoável para os problemas supracitados de emissão, de poluição sonora e de limite de distância, juntando um motor elétrico e um motor a combustão em um único veículo.

Diferentemente dos *plugin* HEVs (PHEVs), os quais proporcionam a possibilidade de recarga somente em estações próprias para isso, os HEVs permitem que a bateria seja recarregada enquanto opera-se no modo de combustão. Para isso, esse tipo de veículo normalmente apresenta quatro modos de operação distintos: combustão pura, recarregando a bateria enquanto trafega no modo de combustão, elétrico puro, e *boost* que é uma combinação dos dois tipos de motores. Um exemplo de veículo com essas configurações pode ser encontrado em [AG, 2021]. Na Figura 1, pode-se observar um exemplo de solução para o problema do TSP com um HEV com as configurações discutidas.

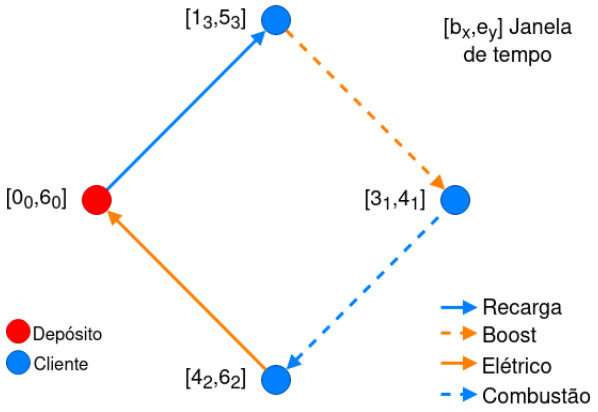


Figura 1: Exemplo de uma solução com relação às janelas de tempo e aos modos de operação do veículo.

Na Figura 1, a rota é composta por um depósito e três clientes. Os números entre colchetes indicam o início e término das janelas de tempo de cada cliente e do depósito. Os números subscritos servem para identificar os clientes. Por exemplo, o depósito tem um início de janela de tempo em 0 e término em 6 e é identificado com o número 0. O início da janela de tempo do depósito subtraído do seu término cria a restrição de duração da rota com o intuito de considerar a leis que regem a quantidade legal de horas de trabalho dos motoristas. Nesse exemplo, o veículo parte do cliente 0, i.e. depósito, para o cliente 3 no modo de recarga da bateria; do cliente 3, segue para o 1 no modo *boost*; do 1, segue para o 2 no modo de combustão; e por último, retorna ao depósito no modo elétrico.

O problema do caixeiro viajante com veículos elétricos híbridos e janelas de tempo (HEV *traveling salesman problem with time windows* – HEVTSPTW) proposto inicialmente por [Doppstadt et al., 2020], doravante denominado apenas HEVTSP, é uma variante do conhecido TSP com as restrições de janela de tempo e que faz uso de um HEV nas entregas.

Parte considerável dos problemas envolvendo veículos elétricos puros tem sua utilização em *vehicle routing problems* (VRPs) com estações de recarga, recarga completa ou parcial (ver [Keskin e Çatay, 2016]) da bateria e resolvidos tanto por meio de algoritmos exatos (ver [Bruglieri et al., 2015], [Lin et al., 2016] e [Desaulniers et al., 2016]), quanto por meio de heurísticas (ver [Erdoğan e Miller-Hooks, 2012], [Schneider et al., 2014] e [Montoya et al., 2017]). Uma outra parte dos problemas aborda os PHEVs, também para problemas de VRPs (ver [Vincent et al., 2017], [Bahrami et al., 2020] e [Li et al., 2020]). Ainda com relação a problemas de VRP, [Zhen et al., 2020] tem uma abordagem utilizando HEVs, mas com foco nas condições da estrada, i.g. tráfego congestionado ou livre, com modos de operação ligeiramente diferentes dos discutidos e considerando tanto estações (para recarga da bateria) quanto postos de combustível.

Com relação aos trabalhos solucionando problemas do tipo TSP, pode-se destacar [Chau et al., 2016] que também tenta otimizar a seleção dos modos de operação do motor, mas com um PHEV. Os trabalhos de [Doppstadt et al., 2016] e de [Doppstadt et al., 2020] abordam o mesmo problema discutido neste artigo, mas aplicando uma *variable neighborhood search* (VNS) em ambos os casos.

Este trabalho visa discutir acerca da aplicação da *hybrid genetic search* (HGS) [Vidal et al., 2014] adaptada para solucionar o HEVTSP, utilizando a *randomized variable neighborhood descent* (RVND) proposta por [Subramanian et al., 2010] na busca local. Os testes foram realizados em instâncias do HEVTSP disponibilizadas na literatura. Foram resolvidas 216 instâncias, das quais 33 soluções melhores do que as da literatura foram encontradas em tempo computacional médio menor do que os do algoritmo baseado em VNS proposto para resolver o mesmo problema.

O restante deste artigo é organizado da seguinte maneira. A Seção 2 descreve o problema do HEVTSP, apresentando suas restrições e algumas informações sobre mudanças na carga da bateria. A Seção 3 trata do algoritmo proposto, discutindo as especificidades de cada etapa. Os resultados computacionais, bem como as etapas seguidas para calibrar o algoritmo, são abordados na Seção 4. A conclusão é trabalhada na Seção 5.

2. Descrição do problema

Seja $G = G(V, A)$ um grafo direcionado, no qual $V = \{0, 1, 2, ..., n\}$ e A representam um conjunto de vértices e de arcos, respectivamente. Para cada vértice $i \in V$, existe um tempo de serviço s_i, representando o tempo necessário para que esse vértice, i.e. cliente, seja atendido; e também os tempos b_i e e_i, que indicam o início e o término da sua janela de tempo, respectivamente. Vale ressaltar que: (i) $\forall i \in V, e_i - b_i \geq s_i$; e (ii) $e_0 - b_0$ indica a duração máxima da rota, i.e. tempo máximo de trabalho do entregador. Não servir um cliente até o término da sua janela constitui em violação de *time-warp* [Vidal et al., 2013], já retornar ao depósito após o término da sua janela de tempo constitui uma violação de duração da rota.

A cada arco do conjunto A, associa-se um custo c e um tempo t para cada modo de operação do HEV: na operação com motor de combustão, o custo e o tempo são indicados por c_c e t_c, respectivamente; no modo recarga, são indicados por c_r e t_r; na operação com o motor elétrico, por c_e e t_e; já no modo *boost*, por c_b e t_b.

Com relação à bateria, tem-se uma carga máxima l_{max}, uma taxa de recarga l_r e uma taxa de descarga l_d. Cada modo de operação pode ter um impacto distinto no nível l da bateria, como pode-se observar na Tabela 1.

Tabela 1: Impacto do tipo de operação do veículo na carga da bateria.

Modo de operação	**Nível l da bateria =**
Combustão c	l
Recarga r	$l + min(l_{max} - l, t_r l_r)$
Elétrico e	$l - min(l, t_e l_d)$
Boost b	$l - min(l, t_b l_d)$

Da Tabela 1, é possível concluir que:

- na operação de combustão, o nível da bateria não é alterado;
- na de recarga, a operação $min(l_{max} - l, t_r l_r)$ garante que o nível da carga não exceda o limite l_{max};
- no modo elétrico, o nível não pode ser menor do que 0, o que é garantido pelo operador $min(l, t_e l_d)$;
- e em *boost*, algo análogo ao item anterior acontece, mas utilizando-se o tempo t_b correspondente a este modo de operação do motor.

O objetivo do problema é realizar um ciclo hamiltoniano de custo mínimo, com início e término no depósito, e sem violar as restrições de janela de tempo, de duração, nem de carga da bateria. Considerando-se a variável de decisão binária $x_{ij}^m = 1$, caso o arco (i, j) em um modo de operação $m \in \{c, r, e, b\}$ seja selecionado, e igual a 0, caso contrário:

$$min \sum_{(i,j) \in A} c_c x_{ij}^c + c_r x_{ij}^r + c_e x_{ij}^e + c_b x_{ij}^b.$$

Para mais informações sobre a formulação matemática, bem como mais detalhes sobre as restrições do HEVTSP, o leitor pode consultar [Doppstadt et al., 2020].

3. Algoritmo Proposto

O algoritmo proposto faz uso do *framework* HGS, com algumas modificações. O método consiste em fazer buscas locais aleatórias, por meio da RVND, em descendentes gerados de subpopulações de indivíduos, i.e. soluções, por meio de um procedimento de *crossover*. O RVND foi baseado no procedimento *variable neighborhood descent* proposta por [Mladenović e Hansen, 1997]. O procedimento HGS pode ser observado no Algoritmo 1.

Inicialmente, duas subpopulações são criadas: uma de soluções viáveis e outra de inviáveis (linha 2). Na linha 3, o operador $m()$ retorna a melhor solução entre as duas subpopulações. Vale ressaltar que uma solução viável é sempre preferível a uma inviável, mesmo tendo um custo menor. Em um laço de repetição que terá no máximo *nMaxIt* iterações sem melhoras na melhor solução encontrada (linhas 5–17), dois torneios binários são realizados a cada iteração. Em cada torneio (linhas 7 e 8), um indivíduo de cada subpopulação é selecionado aleatoriamente e o que tiver melhor aptidão, de acordo com critérios que serão discutidos posteriormente, é considerado o campeão. Os dois indivíduos vitoriosos geram, então, um descendente por meio de um método de *crossover* (linha 9). Esse descendente passa por uma busca local hierárquica (linha 10) e é inserido na sua devida subpopulação, de acordo com sua viabilidade, que é testada por meio da operação $v()$ da condição da linha 11, e caso seja melhor do que a melhor solução encontrada (linha 13), passa a ser a solução global (linha 14). Quando uma nova solução global é encontrada, o laço das linhas 5–17 começa a ser executado desde o início, devido à atribuição da linha 15.

Algoritmo 1 HGS

1: **procedimento** HGS($nMaxIt, \Omega, \Omega^{'}, \lambda$)
2: $\Omega, \Omega^{'} \leftarrow$ **CriaSubpopulações**()
3: $s \leftarrow m(\Omega, \Omega^{'})$
4: $it \leftarrow 0$
5: **enquanto** $it < nMaxIt$ **do**
6: $it \leftarrow it + 1$
7: $P_1 \leftarrow$ **TorneioBinario**$(\Omega, \Omega^{'})$
8: $P_2 \leftarrow$ **TorneioBinario**$(\Omega, \Omega^{'})$
9: $S \leftarrow$ **Crossover**$(P1, P2)$
10: $S \leftarrow$ **RVND**(S)
11: **se** v(S) **então**
12: $\Omega \leftarrow \Omega \cup \{S\}$
13: **se** $f(S) < f(s)$ **então**
14: $s \leftarrow S$
15: $it \leftarrow 0$
16: **senão**
17: $\Omega^{'} \leftarrow \Omega^{'} \cup \{S\}$
18: **retorna** s

Uma explicação mais detalhada do Algoritmo 1 é apresentada nas próximas subseções. A Subseção 3.1 discute a manipulação e comparação de soluções viáveis e inviáveis. O procedimento construtivo, bem como a inserção do indivíduo na população, é detalhado na Subseção 3.2. A Subseção 3.3 abrange o cálculo da aptidão de cada solução. O gerenciamento das subpopulações é abordado na Subseção 3.4. A Subseção 3.5 descreve o torneio binário e o procedimento de *crossover*. Por último, a Subseção 3.6 apresenta os métodos utilizados na busca local.

3.1. Espaço de busca

Soluções para o problema do HEVTSP podem ser inviáveis caso não respeitem as restrições de *time-warp*, de duração ou de carga da bateria. Para comparar os custos de soluções viáveis e inviáveis, utilizou-se um custo penalizado semelhante ao descrito em [Vidal et al., 2012]. O custo penalizado é definido como o custo da rota, somado:

1. a um coeficiente de penalização de duração ω^D multiplicado pela violação de duração da rota;
2. a um coeficiente de penalização de *time-warp* ω^{TW} multiplicado pela soma das violações de *time-warp* dos clientes;
3. e a um coeficiente de penalização de carga da bateria ω^C multiplicado pela soma das violações da carga da bateria.

As violações dos itens 1 e 2 são calculadas em tempo $O(1)$ amortizado, através do algoritmo de [Vidal et al., 2013]. A violação da duração da rota é o resultado da operação $max(0, d - tw_0^e)$, onde d é a duração da rota, considerando o tempo de serviço dos clientes e do depósito, e tw_0^e é o término janela de tempo do depósito. A violação de *time-warp* é o resultado de operações semelhantes, mas considerando o d igual ao tempo de chegada no cliente somado ao seu tempo de serviço e o tw^e do cliente.

Para a avaliação das violações do Item 3, implementou-se um algoritmo com programação dinâmica inspirado no de [Vidal et al., 2013], mas com as particularidades inerentes às definições da Tabela 1. A ideia é que esse algoritmo também seja executado em tempo O(1) amortizado. O termo custo pode ser usado para substituir custo penalizado de agora em diante.

3.2. Procedimento construtivo

O método construtivo é divido em duas etapas: primeiramente, cria-se uma solução; depois, caso não seja repetida, essa solução é inserida na sua devida subpopulação.

3.2.1. Construção de soluções

O procedimento de construção de soluções é baseado no proposto em [Bulhões et al., 2018]. Seja CL uma lista de candidatos inicialmente contendo todos os vértices de V, exceto o depósito, e seja RCL uma lista restrita dos melhores candidatos da CL. A RCL tem tamanho α multiplicado pelo tamanho da instância, sendo α um valor escolhido aleatoriamente do conjunto $\{0.3, 0.4, 0.5\}$.

1. A cada iteração do procedimento, é calculado o custo de inserir cada vértice da CL na solução parcial;
2. Os melhores candidatos, isto é, aqueles de menor custo penalizado, são inseridos na lista restrita;
3. Quando o custo de inserir cada vértice da CL é calculado, um é escolhido aleatoriamente da RCL e inserido na solução parcial;
4. O vértice escolhido no item anterior é removido da CL e o algoritmo volta à etapa 1.

3.2.2. Inserção da solução na população

A fim de inserir a solução em alguma subpopulação, primeiramente calcula-se sua viabilidade. Sabendo-se a qual conjunto o indivíduo pertence, é preciso então determinar se já há uma réplica do mesmo. Para isso, o custo da nova solução é comparado ao custo de todas as outras soluções da subpopulação. Caso não haja nenhuma outra com o mesmo custo, a solução é inserida no conjunto; caso contrário, é descartada.

3.3. Avaliação da solução

O cálculo da aptidão $AP(P)$ de um indivíduo P é feito de forma semelhante à de [Vidal et al., 2012]. Primeiramente, calcula-se a distância média do indivíduo para seus μ_{close} vizinhos.

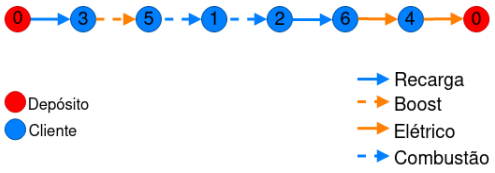


Figura 2: Exemplo de uma solução com relação aos clientes e aos modos de operação do veículo.

Para exemplificar o cálculo dessa distância, consideram-se as seguintes informações sobre a solução P_1 da Figura 2:

Arco i	(0, 3)	(3, 5)	(5, 1)	(1, 2)	(2, 6)	(6, 4)	(4, 0)
Cromossomo arco ϕ_i	(0, 3)	(3, 5)	(5, 1)	(1, 2)	(2, 6)	(6, 4)	(4, 0)
Cromossomo modo σ_j	r	b	c	c	r	e	e

Tabela 2: Cromossomos gerados dos arcos e de seus modos de operação.

A distância $\delta(P_1, P_2)$ entre P_1 e uma outra solução qualquer P_2 é calculada com base nas informações da Tabela 2, retirada da solução P_1 e com base em informações semelhantes obtidas da solução P_2. O cromossomo arco considera apenas os arcos da solução, independentemente dos modos de operação utilizados por eles. Já o cromossomo modo leva em conta apenas os modos, independentemente dos arcos.

$$\delta(P_1, P_2) = \frac{1}{2(n-1)} \sum_{i=1,2,\ldots,(n-1)} \left(\mathbf{V}(\phi_i(P_1), \phi_i(P_2)) + \mathbf{V}(\sigma_i(P_1), \sigma_i(P_2))\right),$$

onde *V(a, b)* é uma função que retorna 1, caso seus argumentos a e b sejam diferentes e 0, caso contrário.

A contribuição para a diversidade $C_{div}(P)$ de um indivíduo P, é calculada como:

$$C_{div}(P) = \frac{1}{\mu_{close}} \sum_{i=1,2,\ldots,\mu_{close}} \delta(P, P_i).$$

Existem dois *rankings*: um de custo penalizado e outro de contribuição para a diversidade C_{div}. No primeiro *ranking*, as soluções são ordenadas em ordem decrescente de custo penalizado; já no segundo, as soluções são ordenadas em ordem crescente de C_{div}. Cada solução P tem uma posição $rank_{quality}(P)$ com relação ao custo penalizado e outra posição $rank_{C_{div}}(P)$ com relação à contribuição C_{div}. A aptidão AP do indivíduo relaciona essas e outras informações:

$$AP(P) = rank_{quality}(P) + \left(1 - \frac{\mu_{elite}}{n}\right) rank_{C_{div}}(P),$$

onde μ_{elite} é o número de indivíduos de elite que deseja-se que sobrevivam para a próxima geração.

3.4. Gerenciamento das subpopulações

Cada subpopulação tem até $\mu + \lambda$ indivíduos. Quando esse limite é alcançado e tenta-se inserir mais uma, as piores λ soluções com relação à aptidão AP são excluídas, restando apenas as μ melhores. Inicialmente, 4μ indivíduos são criados [Vidal et al., 2012] e distribuídos de forma não necessariamente igual (a depender de suas viabilidades) entre as duas subpopulações.

3.5. Seleção dos pais e geração de descendentes

No torneio binário, dois indivíduos de cada subpopulação são selecionados aleatoriamente e o de menor AP vence, assim como no de [Vidal et al., 2012]. Para tirar proveito das subsequências inicial e final dos pais, de forma a não modificá-las bruscamente, fez-se uma adaptação no *crossover OX*. A Figura 3 ilustra a operação de *crossover* entre dois indivíduos P_1 e P_2.

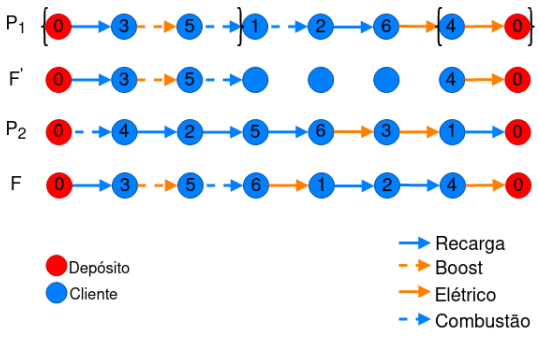


Figura 3: Operação de *crossover*.

Duas subsequências são copiadas do pai P_1: uma que vai de 0 até o índice i_{end} e outra que vai do índice i_{beg} até o fim da rota. Em seguida, o restante da sequência do filho intermediário F' é preenchida de forma circular a partir do segundo pai P_2, completando a formação do descendente F. Vale ressaltar que sempre que um vértice é copiado, o modo de operação entre ele e o próximo cliente também o é, como foi o caso do vértice 5 de P_1 e dos vértices 6, 1 e 2 de P_2. Os índices i_{end} e i_{beg} são escolhidos aleatoriamente, e $i_{end} < i_{beg}$.

3.6. Busca local

A política da busca local foi de *best improvement*, com a utilização dos seguintes movimentos de vizinhança:

- *reinsertion* – um cliente é removido e inserido em outra posição da rota;
- *or-opt2* – dois clientes consecutivos são removidos e inseridos em outra posição da rota;
- *or-opt3* – três clientes consecutivos são removidos e inseridos em outra posição da rota;
- *2-opt* – dois arcos não adjacentes são removidos e outros dois são criados, formando uma nova rota;
- *exchange* – movimento de permutação entre dois clientes;
- *swap(2)* – dois clientes consecutivos são permutados com outros dois clientes consecutivos;
- *swap(3)* – três clientes consecutivos são permutados com outros três clientes consecutivos;
- *3-mode-change* [Doppstadt et al., 2016] – todos os subconjuntos possíveis de três arcos são avaliados. Em cada avaliação, são testadas novas combinações de modos de operação dos arcos, considerando somente os modos não utilizados no subconjunto sob análise.

Em todos os movimentos, com exceção do *3-mode-change*, são testadas todas as combinações de modos de operação nos novos arcos gerados considerando uma lista restrita de modos. Por exemplo, no *reinsertion*, três novos arcos são gerados: um para conectar os dois clientes adjacentes ao que foi removido e outros dois para reconectar o cliente removido em uma outra posição. Para cada teste de combinação de arcos, também são testadas as combinações de modos de operação. A lista restrita é composta pelos modos de combustão e de recarga, os quais observou-se empiricamente serem mais utilizados.

Devido ao custo computacional elevado do *3-mode-change*, decidiu-se implementar uma RVND hierárquica, na qual as demais vizinhanças são exaustivamente testadas em busca de melhoras na solução, e, ao final dessa busca exaustiva, aplica-se o *3-mode-change*. Caso haja melhora, a busca volta a ser realizada com as demais vizinhanças. Esse procedimento se repete até que não seja mais possível melhorar o custo penalizado da solução.

Após a aplicação da RVND, a solução é inserida na subpopulação correspondente e, caso seja viável e apresente um custo melhor do que o da melhor solução encontrada, passa a ser a solução global.

4. Resultados computacionais

O algoritmo proposto foi executado em um computador com processador Intel(R) Core(TM) i7-3770 CPU 3.40 GHz, 8 GB de memória RAM e sistema operacional (SO) Linux Ubuntu 16.04. Foram utilizadas instâncias da literatura para os experimentos ([Doppstadt et al., 2020]). Os resultados obtidos foram comparados com aqueles encontrados pelo VNS proposto por [Doppstadt et al., 2020] que, por sua vez, foi executado em um computador com processador Intel(R) Core(TM) i7-4790 CPU 3.6 GHz, 24 GB de memória RAM, rodando o SO Windows 10. De acordo com o *site* `https://www.cpubenchmark.net/compare/Intel-i7-3770-vs-Intel-i7-4790/896vs2226`, o processador utilizado por [Doppstadt et al., 2020] é aproximadamente 1,07 vezes mais rápido do que aquele utilizado pra executar o HGS.

Para calibrar os parâmetros, primeiramente testou-se o melhor conjunto de vizinhanças, juntamente com a melhor política de melhora (*best* ou *first improvement*). Combinações distintas, com movimentos de vizinhança com diferentes quantidades de clientes consecutivos, foram testadas com $nMaxIt = 1$. Em seguida, testaram-se dois tipos de busca local: uma sequencial como descrita em [Mecler et al., 2021] e outra aleatória com o RVND. Nesse ponto, $nMaxIt$ foi configurado como $2n$, sendo n o tamanho da instância. Na sequência, analisaram-se as quantidades de iterações sem melhora da solução global. Os valores testados foram de 100, 500 e de 1000, sendo $nMaxIt = 500$ o valor final escolhido. Em todos os testes, 10 execuções do algoritmo foram realizadas, com o *gap* geral médio como parâmetro de comparação de melhora.

Com relação aos parâmetros utilizados no gerenciamento das subpopulações e no cálculo da aptidão, utilizaram-se os valores propostos em [Mecler et al., 2021]: $\mu = 20$, $\lambda = 40$, $\mu_{elite} = 10$ e $\mu_{close} = 3$. Para as penalizações: $\omega^D = 1$, $\omega^{TW} = 50$ e $\omega^C = 1$. O valor da penalização da violação de *time-warp* é reflexo das curtas janelas de tempo de algumas instâncias. Esse valor direciona a heurística na busca de soluções viáveis.

Na Tabela 3, são apresentados os resultados obtidos após 10 execuções do algoritmo HGS, juntamente com os resultados de 5 execuções do algoritmo VNS descrito em [Doppstadt et al., 2020]. Os nomes das instâncias são no formato $HEVTSP_\alpha_\beta_\gamma_\lambda$, onde: α representa três distâncias distintas do depósito para a área de entrega, β a quantidade de clientes, γ indica variações nas localizações dos clientes e nos tempos de serviço e λ são variações nas configurações e na quantidade de janelas de tempo dos clientes, de acordo com [Doppstadt et al., 2020]. Com o intuito de reduzir a poluição visual causada pelo excesso de informações, a *substring* $HEVTSP_$ foi retirada dos nomes das instâncias.

Dois conjuntos de soluções são reportados em [Doppstadt et al., 2020]: um denominado *runtime optimized* e o outro *quality optimized*. No primeiro, utilizaram-se parâmetros mais leves a fim de gerar soluções consideravelmente satisfatórias em um intervalo de tempo menor, e no segundo, a prioridade é a qualidade das soluções. Para tornar a comparação mais justa, decidiu-se confrontar os custos do HGS aos custos do conjunto *quality optimized*; já os tempos do HGS são comparados aos tempos do *runtime optimized*.

Os resultados ótimos foram obtidos para todas as instâncias nas quais foi possível determiná-los, isto é, nas de 8 e de 10 clientes. Ademais, o custo médio foi igual ao melhor e o tempo computacional médio de execução foi inferior a 0,2 segundos para todas elas. Como esses resultados não agregam na comparação, decidiu-se não reportá-los na Tabela 3.

Como pode-se notar pela Tabela 3, as diferenças de *gap* mais significativas se deram nas maiores instâncias (50 clientes), sendo o maior valor de *gap* igual a $-5{,}23\%$, registrado na instância $1_50_2_3$. Ao todo, foram encontradas novas soluções melhores para 33 instâncias. De uma forma

Tabela 3: Resultados computacionais.

	VNS			HGS					VNS				HGS			
Instância	**Melhor**	**Média**	**T (s)**	**Melhor**	**Média**	**Gap (%)**	**T (s)**	**Instância**	**Melhor**	**Média**	**T (s)**	**Melhor**	**Média**	**Gap (%)**	**T (s)**
1_20_1_0	2005,89	2005,89	4,1	2005,89	2005,89	**0,0**	0,61	1_50_1_0	2771,98	2819,37	182,7	2712,9	2715,83	**-2,13**	15,66
1_20_1_1	2005,89	2005,89	4,1	2005,89	2005,89	**0,0**	0,5	1_50_1_1	2712,9	2714,44	169,1	2712,9	2712,9	**0,0**	12,83
1_20_1_2	2005,89	2005,89	3,8	2005,89	2005,89	**0,0**	0,47	1_50_1_2	2716,54	2716,54	158,4	2716,54	2716,54	**0,0**	9,83
1_20_1_3	2005,89	2005,89	4,1	2005,89	2005,89	**0,0**	0,47	1_50_1_3	2761,61	2828,51	191,6	2712,9	2713,31	**-1,76**	16,48
1_20_1_4	2005,89	2005,89	4,0	2005,89	2005,89	**0,0**	0,58	1_50_1_4	2713,97	2714,53	198,7	2712,9	2713,05	**-0,04**	13,08
1_20_1_5	2005,89	2005,89	3,7	2005,89	2005,89	**0,0**	0,46	1_50_1_5	2716,54	2716,54	164,8	2716,54	2716,54	**0,0**	9,79
1_20_2_0	1969,78	1969,8	4,1	1969,78	1969,78	**0,0**	0,69	1_50_2_0	2574,63	2587,52	223,2	2489,24	2491,14	**-3,32**	17,01
1_20_2_1	1969,78	1969,78	4,3	1969,78	1969,78	**0,0**	0,66	1_50_2_1	2497,78	2508,73	224,3	2489,24	2489,24	**-0,34**	13,4
1_20_2_2	1969,78	1969,78	4,2	1969,78	1969,78	**0,0**	0,57	1_50_2_2	2497,78	2497,78	200,9	2497,78	2497,78	**0,0**	13,37
1_20_2_3	1969,78	1969,78	4,1	1969,78	1969,78	**0,0**	0,5	1_50_2_3	2626,64	2626,64	210,1	2489,24	2491,12	**-5,23**	17,18
1_20_2_4	1969,78	1969,78	4,1	1969,78	1969,78	**0,0**	0,59	1_50_2_4	2497,78	2502,45	224,4	2489,24	2489,24	**-0,34**	14,85
1_20_2_5	1969,78	1969,78	3,0	1969,78	1969,78	**0,0**	0,59	1_50_2_5	2497,78	2497,78	163,5	2497,78	2497,78	**0,0**	13,13
1_20_3_0	1606,57	1606,57	4,6	1606,42	1606,42	**-0,01**	0,51	1_50_3_0	2506,76	2506,76	198,5	2474,21	2475,55	**-1,3**	16,34
1_20_3_1	1606,42	1606,51	4,4	1606,42	1606,42	**0,0**	0,65	1_50_3_1	2499,59	2523,86	206,4	2474,21	2474,21	**-1,02**	14,18
1_20_3_2	1608,37	1608,37	3,8	1608,37	1608,37	**0,0**	0,36	1_50_3_2	2497,59	2497,63	194,1	2497,59	2497,59	**0,0**	10,42
1_20_3_3	1606,42	1606,51	4,2	1606,42	1606,42	**0,0**	0,45	1_50_3_3	2539,65	2656,19	196,2	2487,14	2489,24	**-2,07**	17,42
1_20_3_4	1606,42	1606,51	4,3	1606,42	1606,42	**0,0**	0,59	1_50_3_4	2531,96	2531,96	209,2	2487,14	2487,26	**-1,77**	18,53
1_20_3_5	1608,37	1608,37	3,5	1608,37	1608,42	**0,0**	0,45	1_50_3_5	2497,59	2497,86	136,6	2497,59	2497,59	**0,0**	10,64
2_20_1_0	7807,05	7807,25	3,5	7807,05	7807,05	**0,0**	0,68	2_50_1_0	8420,92	8421,19	102,8	8338,12	8343,52	**-0,98**	17,78
2_20_1_1	7807,05	7807,05	3,6	7807,05	7807,05	**0,0**	0,68	2_50_1_1	8363,71	8363,71	73,7	8363,71	8363,71	**0,0**	15,57
2_20_1_2	7807,05	7807,05	3,1	7807,05	7807,05	**0,0**	0,61	2_50_1_2	8363,71	8363,71	124,6	8363,71	8363,71	**0,0**	10,88
2_20_1_3	7807,56	7807,56	3,6	7807,05	7807,05	**-0,01**	0,51	2_50_1_3	8414,32	8418,38	85,1	8338,12	8338,25	**-0,91**	16,32
2_20_1_4	7807,56	7807,56	3,7	7807,05	7807,05	**-0,01**	0,63	2_50_1_4	8386,99	8387,11	56,4	8386,99	8386,99	**0,0**	15,03
2_20_1_5	7807,05	7807,05	2,8	7807,05	7807,05	**0,0**	0,51	2_50_1_5	8404,0	8404,56	122,3	8394,93	8408,84	**-0,11**	20,25
2_20_2_0	7670,94	7670,94	3,7	7670,94	7671,31	**0,0**	0,86	2_50_2_0	8585,05	8586,54	241,1	8425,7	8430,55	**-1,86**	19,81
2_20_2_1	7670,94	7670,94	3,6	7670,94	7670,94	**0,0**	0,82	2_50_2_1	8427,26	8427,26	153,4	8425,9	8426,04	**-0,02**	19,39
2_20_2_2	7670,94	7670,94	3,2	7670,94	7670,94	**0,0**	0,72	2_50_2_2	8425,9	8425,9	152,6	8425,9	8425,9	**0,0**	13,41
2_20_2_3	7670,94	7670,94	3,6	7670,94	7671,19	**0,0**	0,76	2_50_2_3	8565,78	8576,23	174,9	8425,7	8425,98	**-1,64**	20,01
2_20_2_4	7670,94	7670,94	3,7	7670,94	7671,07	**0,0**	0,94	2_50_2_4	8427,26	8427,26	124,3	8425,9	8425,9	**-0,02**	17,64
2_20_2_5	7670,94	7670,94	3,1	7670,94	7670,94	**0,0**	0,6	2_50_2_5	8425,9	8425,9	145,6	8425,9	8425,9	**0,0**	11,52
2_20_3_0	7713,39	7713,39	3,4	7709,14	7710,42	**-0,06**	0,62	2_50_3_0	8508,2	8508,2	162,9	8422,32	8429,38	**-1,01**	20,53
2_20_3_1	7709,14	7714,31	3,3	7709,14	7709,14	**0,0**	0,74	2_50_3_1	8428,38	8431,13	90,7	8428,38	8428,38	**0,0**	17,03
2_20_3_2	7709,14	7709,14	3,2	7709,14	7709,14	**0,0**	0,54	2_50_3_2	8428,38	8428,38	168,6	8428,38	8428,38	**0,0**	10,79
2_20_3_3	7709,14	7711,08	3,5	7709,14	7709,14	**0,0**	0,65	2_50_3_3	8475,79	8497,2	159,7	8428,38	8431,62	**-0,56**	19,27
2_20_3_4	7709,14	7710,76	3,6	7709,14	7709,14	**0,0**	0,63	2_50_3_4	8428,38	8429,78	69,1	8428,38	8428,38	**0,0**	14,7
2_20_3_5	7709,14	7709,14	3,1	7709,14	7709,14	**0,0**	0,46	2_50_3_5	8428,38	8428,38	155,0	8428,38	8428,38	**0,0**	10,41
3_20_1_0	13329,82	13329,82	3,3	13329,8	13329,8	**0,0**	0,66	3_50_1_0	13881,72	13942,6	196,2	13847,0	13854,15	**-0,25**	20,56
3_20_1_1	13343,32	13343,34	3,5	13343,3	13344,81	**0,0**	0,91	3_50_1_1	13853,2	13891,55	175,7	13847,0	13884,0	**-0,04**	18,47
3_20_1_2	13343,32	13343,32	3,5	13343,3	13343,3	**0,0**	0,8	3_50_1_2	13846,98	13846,98	179,7	13847,0	13857,68	**0,0**	16,36
3_20_1_3	13329,82	13329,82	3,4	13329,8	13329,8	**0,0**	0,68	3_50_1_3	13847,37	13907,15	178,5	13847,0	13852,38	**-0,0**	19,78
3_20_1_4	13343,32	13343,32	3,7	13343,3	13350,93	**0,0**	0,86	3_50_1_4	13847,37	13875,43	212,1	13847,0	13893,94	**-0,0**	18,85
3_20_1_5	13343,32	13343,32	3,4	13343,3	13350,93	**0,0**	0,61	3_50_1_5	13846,98	13846,98	162,7	13847,0	13868,88	**0,0**	13,75
3_20_2_0	13287,61	13287,61	3,4	13287,6	13294,4	**0,0**	0,79	3_50_2_0	14043,83	14072,36	180,7	13935,6	13954,37	**-0,77**	21,14
3_20_2_1	13287,61	13288,14	3,6	13287,6	13287,6	**0,0**	0,79	3_50_2_1	13980,34	13980,34	173,3	13980,3	14006,27	**-0,0**	15,68
3_20_2_2	13287,61	13287,61	3,3	13287,6	13287,6	**0,0**	0,67	3_50_2_2	13985,22	13985,22	172,8	13985,2	14012,49	**-0,0**	16,71
3_20_2_3	13287,61	13287,61	3,6	13287,6	13287,6	**0,0**	0,7	3_50_2_3	14077,99	14085,2	148,2	13985,2	13987,36	**-0,66**	23,13
3_20_2_4	13287,61	13287,61	3,7	13287,6	13287,6	**0,0**	0,91	3_50_2_4	13985,22	13985,22	161,4	13985,2	14021,75	**-0,0**	18,46
3_20_2_5	13287,61	13287,61	3,2	13287,6	13287,6	**0,0**	0,84	3_50_2_5	13985,22	13985,22	162,2	13985,2	14005,14	**-0,0**	14,11
3_20_3_0	13247,33	13247,33	3,4	13247,3	13247,3	**0,0**	0,64	3_50_3_0	13866,67	13866,67	188,3	13805,5	13809,54	**-0,44**	15,75
3_20_3_1	13247,33	13247,33	3,2	13247,3	13247,3	**0,0**	0,71	3_50_3_1	13821,48	13821,48	206,2	13806,0	13906,58	**-0,11**	16,27
3_20_3_2	13247,33	13247,33	3,2	13247,3	13247,3	**0,0**	0,65	3_50_3_2	13820,18	13820,18	173,4	13819,3	13877,98	**-0,01**	13,76
3_20_3_3	13247,33	13247,33	3,4	13247,3	13247,3	**0,0**	0,66	3_50_3_3	13953,07	13981,12	183,3	13806,0	13806,01	**-1,05**	21,26
3_20_3_4	13247,33	13247,33	3,5	13247,3	13247,3	**0,0**	0,81	3_50_3_4	13821,48	13821,48	169,2	13806,0	13892,31	**-0,11**	18,07
3_20_3_5	13247,33	13247,33	2,9	13247,3	13247,3	**0,0**	0,61	3_50_3_5	13819,3	13819,83	177,8	13819,3	13916,96	**0,0**	15,54

geral, o algoritmo HGS com RVND apresentou soluções com custos iguais ou melhores às do conjunto *quality optimized* em tempos computacionais médios inferiores aos do *runtime optimized*.

5. Conclusão e trabalhos futuros

A proposta deste trabalho foi aplicar um algoritmo genético híbrido com busca local hierárquica baseada no procedimento RVND ao problema do HEVTSP. A performance do algoritmo foi avaliada em instâncias da literatura. De um modo geral, o algoritmo proposto apresentou soluções competitivas em menor tempo computacional médio. Experimentos computacionais foram realizados em 216 instâncias disponíveis na literatura das quais 33 soluções melhores foram encontradas.

Como trabalhos futuros, pretende-se testar diferentes métodos de geração de descendentes, além de outras formas de gerar cromossomos a partir das soluções. Por último, faz-se necessário efetuar mais testes a fim de ratificar a eficiência computacional do algoritmo de checagem de violação da carga da bateria.

Referências

AG, A. (2021). Retrofit electric drive kit for diesel delivery vehicles. URL `https://www.aradex.de/en/system-solutions/aradex-project-report/light-vehicles-small-vans/retrofit-electric-drive-kit-for-diesel-delivery-vehicles/`.

Bahrami, S., Nourinejad, M., Amirjamshidi, G., e Roorda, M. J. (2020). The plugin hybrid electric vehicle routing problem: A power-management strategy model. *Transportation Research Part C: Emerging Technologies*, 111:318–333.

Bruglieri, M., Pezzella, F., Pisacane, O., e Suraci, S. (2015). A variable neighborhood search branching for the electric vehicle routing problem with time windows. *Electronic Notes in Discrete Mathematics*, 47:221–228.

Bulhões, T., Subramanian, A., Erdoğan, G., e Laporte, G. (2018). The static bike relocation problem with multiple vehicles and visits. *European Journal of Operational Research*, 264(2):508–523.

Chau, C.-K., Elbassioni, K., e Tseng, C.-M. (2016). Fuel minimization of plug-in hybrid electric vehicles by optimizing drive mode selection. In *Proceedings of the Seventh International Conference on Future Energy Systems*, p. 1–11.

Dekker, R., Bloemhof, J., e Mallidis, I. (2012). Operations research for green logistics–an overview of aspects, issues, contributions and challenges. *European journal of operational research*, 219 (3):671–679.

Desaulniers, G., Errico, F., Irnich, S., e Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405.

Doppstadt, C., Koberstein, A., e Vigo, D. (2016). The hybrid electric vehicle–traveling salesman problem. *European Journal of Operational Research*, 253(3):825–842.

Doppstadt, C., Koberstein, A., e Vigo, D. (2020). The hybrid electric vehicle—traveling salesman problem with time windows. *European Journal of Operational Research*, 284(2):675–692.

Erdoğan, S. e Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation research part E: logistics and transportation review*, 48(1):100–114.

Keskin, M. e Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127.

Li, X., Shi, X., Zhao, Y., Liang, H., e Dong, Y. (2020). Svnd enhanced metaheuristic for plug-in hybrid electric vehicle routing problem. *Applied Sciences*, 10(2):441.

Lin, J., Zhou, W., e Wolfson, O. (2016). Electric vehicle routing problem. *Transportation Research Procedia*, 12:508–521.

Mecler, J., Subramanian, A., e Vidal, T. (2021). A simple and effective hybrid genetic search for the job sequencing and tool switching problem. *Computers & Operations Research*, 127:105153.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100.

Montoya, A., Guéret, C., Mendoza, J. E., e Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110.

Sbihi, A. e Eglese, R. W. (2007). Combinatorial optimization and green logistics. *4OR*, 5(2): 99–116.

Schneider, M., Stenger, A., e Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, 48(4):500–520.

Subramanian, A., Drummond, L. M. d. A., Bentes, C., Ochi, L. S., e Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., e Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

Vidal, T., Crainic, T. G., Gendreau, M., e Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research*, 40(1):475–489.

Vidal, T., Crainic, T. G., Gendreau, M., e Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673.

Vincent, F. Y., Redi, A. P., Hidayat, Y. A., e Wibowo, O. J. (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing*, 53:119–132.

Zhen, L., Xu, Z., Ma, C., e Xiao, L. (2020). Hybrid electric vehicle routing problem with mode selection. *International Journal of Production Research*, 58(2):562–576.