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

BRKGA COMBINADO COM O MÉTODO Q-LEARNING PARA RESOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE COM AGRUPAMENTOS EM FAMÍLIAS

Bárbara Lessa Vianna
Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP
barbara.vianna@unifesp.br

Tiago Tiburcio da Silva
Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP
ttsilva@unifesp.br

Antônio Augusto Chaves
Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP
antonio.chaves@unifesp.br

RESUMO

Este trabalho aborda o Problema do Caixeiro Viajante com Agrupamentos em Famílias. Este problema é uma variante do Problema do Caixeiro Viajante, na qual os nós são agrupados em famílias e busca-se realizar um número específico de visitas dentro de cada família. O caixeiro não precisa necessariamente visitar todos os nós, mas apenas um número predefinido. Trata-se de um problema de otimização combinatória classificado como NP-Difícil. Para resolução do problema foi utilizada uma metaheurística híbrida que combina o *Biased Random-key Genetic Algorithm* (BRKGA) com o algoritmo de aprendizagem por reforço *Q-Learning* para configurar os parâmetros do BRKGA durante o processo evolutivo. Para validar o método foram realizados testes computacionais utilizando instâncias disponíveis na literatura. Os resultados obtidos pelos experimentos foram comparados com os melhores resultados da literatura e demonstraram que o método aplicado é robusto e eficiente.

PALAVRAS CHAVE. Problema do Caixeiro Viajante, Algoritmo Genético, Aprendizado por Reforço.

Tópicos (MH - Metaheurísticas)

ABSTRACT

This paper approach the Family Traveling Salesman Problem. This problem is a variant of the Traveling Salesman Problem, which qualifies nodes as grouped into families and seeks to carry out a specific number of visits to nodes within each family. In the problem studied, the traveling salesman doesn't necessarily have to visit all customers, but only a predefined number. This is a combinatorial optimization problem classified as NP-Hard. To solve this problem, we implement

a hybrid metaheuristic that combines the *Biased Random Key Genetic Algorithm* (BRKGA) with the reinforcement learning algorithm Q-Learning to configure the BRKGA parameters during the evolutionary process. To validate the method, computational tests were performed using instances available in the literature. The results obtained by the experiments were compared with the best results in the literature and show that our method is robust and efficient.

KEYWORDS. Traveling Salesman Problem. Genetic Algorithm. Reinforcement Learning.

Paper topics (Metaheuristic)

1. Introdução

Este trabalho aborda o Problema do Caixeiro Viajante com Agrupamentos em Famílias (FTSP, do inglês *Family Traveling Salesman Problem*), uma variante estudada recentemente na literatura do Problema do Caixeiro Viajante. O FTSP foi introduzido por [Morán-Mirabal et al., 2014] e consiste em uma generalização do problema clássico, no qual um conjunto de pontos é dividido em diferentes subconjuntos que são denominados famílias e objetiva-se determinar uma rota de custo mínimo que inicia e termina em um depósito e percorre um determinado número de pontos pertencentes a cada família. O FTSP é classificado como um problema NP-Difícil [Morán-Mirabal et al., 2014].

A motivação do FTSP está relacionada a um problema de otimização de estoque de produtos em um armazém. Neste problema o objetivo é localizar produtos do mesmo tipo que são armazenados separadamente, e realizar a separação dos mesmos conforme a demanda [Morán-Mirabal et al., 2014]. Neste sentido, os produtos são classificados como famílias e as demandas para cada tipo de produto são a quantidade de visitas necessárias para cada família.

[Morán-Mirabal et al., 2014] focaram em métodos heurísticos para resolução do FTSP. Eles propuseram duas metaheurísticas: um *Biased Random-key Genetic Algorithm* (BRKGA) e um *Greedy Random Adaptive Search Procedure* (GRASP). Por meio de experimentos computacionais, os autores concluíram que o método GRASP apresentou melhores resultados em relação ao método *BRKGA* em instâncias com dimensões maiores.

Diferentes modelos matemáticos para o FTSP foram propostos por [Bernardino e Paias, 2018] e através de experimentos computacionais instâncias de até 280 pontos foram resolvidas na otimalidade. Também foi proposto um algoritmo *Iterated Local Search* (ILS) capaz de melhorar os melhores limitantes superiores conhecidos na literatura.

Três métodos de solução foram propostos por [Bernardino e Paias, 2021]: um Algoritmo Genético (GA), um ILS adaptado de [Bernardino e Paias, 2018] e um algoritmo híbrido que combina um algoritmo *branch-and-cut* com procedimento de busca local. O algoritmo híbrido foi capaz de melhorar os limitantes superiores conhecidos para todas as instâncias *benchmark*. Além disso, foi desenvolvido um gerador de instâncias para criar instâncias com diferentes características. Este algoritmo híbrido é a heurística estado-da-arte para solução do FTSP.

Visto que o FTSP é NP-Difícil e por este ser um problema de otimização combinatória, à medida que a dimensão do problema aumenta, o tempo computacional necessário para resolução através de um método exato apresenta um crescimento exponencial. Diante de tal circunstância justifica-se a aplicação de algoritmos heurísticos na busca por soluções de boa qualidade. Contudo, para que uma metaheurística seja eficiente e capaz de alcançar resultados satisfatórios, é importante que o método seja configurado adequadamente considerando as características explícitas do problema abordado e os dados de entrada.

A Inteligência Artificial (IA) é o campo da ciência da computação que desenvolve *softwares* e máquinas capazes de pensar, raciocinar, aprender e resolver problemas altamente complexos. O Aprendizado de Máquinas (AM), um dos campos de estudo da IA, pode ser definido como um conjunto de algoritmos que são capazes de aprender assim como o cérebro humano, a partir de exemplos, dados e análises de problemas anteriores de tal modo a se tornar capacitado para tomar decisões, prever resultados, tendências e padrões. Dentre os métodos de AM têm-se o algoritmo *Q-Learning* [Watkins e Dayan, 1992], um algoritmo de aprendizagem por reforço que ao maximizar as recompensas, busca encontrar a melhor ação a ser tomada, dado um estado atual [Sutton e Barto, 1998].

O objetivo deste trabalho é resolver o FTSP através de um método híbrido combinando a metaheurística BRKGA com o método *Q-Learning*, chamado BRKGA-QL [Chaves e Lorena, 2021]. Desta forma, os parâmetros da metaheurística serão configurados durante o processo evolutivo do BRKGA baseando-se no aprendizado a partir das gerações com objetivo de maximizar os valores de recompensa. Esta abordagem visa minimizar os efeitos causados por uma configuração de parâmetros que não seja adequada para uma determinada instância.

Este trabalho está organizado em 5 seções, sendo esta a introdução. Na Seção 2 as notações e um modelo de programação matemática para o FTSP são apresentadas. Na Seção 3, o BRKGA-QL proposto aplicado ao FTSP é apresentado. Na Seção 4 são exibidos e analisados os resultados computacionais do BRKGA-QL. A Seção 5 apresenta as conclusões e trabalhos futuros.

2. O Problema do Caixeiro Viajante com Agrupamentos em Famílias

O FTSP é estruturado de modo que o conjunto de nós são agrupados em subconjuntos que foram denominados de famílias. Para tanto, o problema apresenta como parâmetros de entrada o número de nós (N), o número de famílias (L), o número de visitas necessárias para completar a rota do caixeiro (V), o número de visitas em cada família (v_l), a lista de nós pertencentes a cada família (F_l), e o custo (por exemplo, a distância) do nó i para o nó j (c_{ij}).

Desta forma, tem-se um grafo completo e direcionado $G = (Z, A)$, sendo que $Z = \{0, 1, ..., N\}$ e 0 representa o depósito. Considera-se, sem perda de generalidade, que os nós pertencentes à família 1 são os nós $1, 2, ..., n_1$, os nós que pertencem à família 2 são $n_1+1, ..., n_1+n_2$, e assim consecutivamente. Portanto, as famílias são partições disjuntas dos nós e união destas representa o conjunto Z. O número total de visitas que devem ser realizadas é expresso por:

$$V = \sum_{l=1}^{L} v_l. \tag{1}$$

Assume-se que $v_l \geq 1$ e uma família l é completa se houver v_l nós da família l na rota.

Um modelo matemático para o FTSP foi proposto por [Bernardino e Paias, 2018]. Para descrever este modelo, considere as variáveis x e y:

$$x_{ij} = \begin{cases} 1, & \text{se o arco de } i \text{ para } j \text{ estiver no caminho;} \\ 0, & \text{caso contrário;} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{se o nó } i \text{ for visitado na rota;} \\ 0, & \text{caso contrário;} \end{cases}$$

Assim sendo, o modelo de [Bernardino e Paias, 2018] é representado pelas equações (2)-(9):

$$min \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij} x_{ij} \tag{2}$$

Sujeito à:

$$\sum_{j=1}^{N} x_{0j} = 1 \tag{3}$$

$$\sum_{j=0}^{N} x_{ij} = y_i, \qquad i = 1, ..., N \tag{4}$$

$$\sum_{j=0}^{N} x_{ji} - \sum_{j=0}^{N} x_{ij} = 0, \qquad i = 0, ..., N \tag{5}$$

$$\sum_{i=1}^{F_l} y_i = v_l \qquad l = 1, ..., L \tag{6}$$

$\{(i, j) \in A : x_{ij} = 1\}$ não contém sub-rotas (7)

$$x_{ij} \in \{0, 1\} \qquad i = 0, ..., N, j = 0, ..., N \tag{8}$$

$$y_i \in \{0, 1\} \qquad i = 1, ..., N \tag{9}$$

A função objetivo (2) minimiza o custo total da rota. A restrição (3) garante que um e apenas um arco saia obrigatoriamente do depósito. O conjunto de restrições (4) conecta as variáveis x e y e afirmam que se o nó i for visitado, então deve haver um arco deixando i e caso contrário não haverá arco deixando i. As restrições (5) certificam em cada nó que o número de arcos que chegam e o número de arcos que saem são iguais, e, com a restrição (3) garantem que o circuito inicie e termine no depósito. As restrições (6) asseguram que sejam realizados os números de visitas necessárias para cada família. Finalmente, as restrições (8) e (9) definem o domínio das variáveis x e y, respectivamente. [Bernardino e Paias, 2018] definem várias formas de modelar o conjunto de restrições (7) para obter formulações válidas para o FTSP.

3. Métodos de Solução

Nesta seção são apresentados os métodos aplicados para solução do FTSP. Os conceitos do BRKGA são detalhados na subseção 3.1, abrangendo seus parâmetros. Na sequência, o algoritmo *Q-Learning* é especificado na subseção 3.2, e por fim tem-se a configuração geral do método BRKGA-QL na subseção 3.3.

3.1. *Biased Random-Key Genetic Algrithm*

Proposto por [Gonçalves e Resende, 2011], o *Biased Random-Key Genetic Algorithm* (BRKGA) é uma metaheurística que utiliza o conceito de chaves aleatórias como representação de suas soluções. As soluções são codificadas como vetores de *n* elementos de números reais gerados aleatoriamente no intervalo [0,1[. Uma solução viável é obtida por meio da decodificação de uma solução de chaves aleatórias pela função determinística *decoder* [Bean, 1994]. Esta é a única função dependente do problema que está sendo resolvido. Desta forma, é possível criar um *framework* capaz de solucionar problemas de otimização combinatória de diferentes naturezas [Toso e Resende, 2015].

O BRKGA inicializa o processo de busca com uma população de tamanho p, sendo gerada aleatoriamente. Na sequência, em cada geração, os indivíduos são classificados em dois grupos: o primeiro grupo refere-se à população elite ($|P_e| = p * p_e$), onde estão alocados os indivíduos com melhores valores de aptidão e no segundo grupo são destinados os indivíduos não-elite ($p - |P_e|$).

Realizada a classificação dos indivíduos, todos os indivíduos da elite são copiados sem qualquer tipo de modificação para a próxima geração. Em seguida, são introduzidos os indivíduos mutantes ($|P_m| = p * p_m$), os quais são gerados da mesmo forma que os indivíduos da população inicial. A nova população será composta por P_e indivíduos elite, P_m mutantes, e $p - |P_e| - |P_m|$ indivíduos, os quais são os filhos gerados através do cruzamento de pares de indivíduos elite e indivíduos não-elite. O cruzamento utilizado no BRKGA é o *parametrized uniform crossover* [Spears e Jong, 1991], e consiste em combinações entre os pais para geração dos filhos de tal modo que as posições do vetor de chaves aleatórias são individualmente comparadas com ambos os pais. Considera-se o parâmetro ρ_e que indica a probabilidade do filho herdar o gene (componente do vetor) do pai elite. O critério de parada do BRKGA é o número máximo de gerações (max_{gen}).

3.2. Algoritmo *Q-Learning*

Considerado uma das contribuições mais relevantes em Aprendizagem por Reforço, o algoritmo *Q-Learning* desenvolvido por [Watkins e Dayan, 1992], baseia-se no aprendizado do agente através da interação com o meio no qual está inserido. De forma autônoma, ele aprende a identificar as melhores ações a serem executadas, com base nas experiências de atuações anteriores, considerando uma sequência de tentativas que envolvem recompensas e penalidades.

Atribui-se uma função estado-ação, $Q(s, a)$, que expressa a estimativa do retorno total esperado e como consequência das atualizações dos pares estado-ação esta se aproxima da função estado-ação ótimo [De Lima et al., 2007].

Para obter o retorno total esperado são estabelecidas intensificações às visitas direcionadas às boas ações e são conjuntamente utilizadas técnicas de diversificação para explorar outras ações que levarão possivelmente ao mapeamento de novas políticas com melhores convergências. A política ϵ-*Greedy* consiste em uma boa estratégia para realizar um balanceamento entre intensificação e diversificação, executando a ação que possui o maior valor em Q com probabilidade $1 - \epsilon$ ou selecionando uma ação de forma aleatória, caso contrário. Para mais informações sobre *Q-learning*, consultar [Sutton e Barto, 1998].

A expressão matemática para atualizar Q é definida pela Equação 10:

$$Q(s_t, a_t) = Q(s_t, a_t) + lf[R + df * \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \tag{10}$$

na qual s_t é o estado atual, a_t é a ação realizada no estado atual, R é o sinal de reforço (retorno ou recompensa) recebido após executar a_t em s_t, s_{t+1} é o estado seguinte, df é o fator de desconto e lf é o coeficiente de aprendizagem. A função Q é o valor agregado ao par estado-ação (s, a) e representa quão boa é a escolha dessa ação na otimização do retorno. O intervalo de t é definido pelo número de estados no *Q-Learning*.

3.3. BRKGA-QL

Neste trabalho foi utilizado o método BRKGA-QL, proposto por [Chaves e Lorena, 2021], para solucionar o FTSP. O *Q-Learning* é responsável pela definição de quais parâmetros serão utilizados durante o processo evolutivo, mediante uma lista predefinida, através do aprimoramento das experiências recorrentes de suas ações. Conforme descrito na Seção 3.1, os parâmetros configurados são: tamanho da população (p), partição elite (p_e), partição mutante (p_m) e probabilidade de herdar o gene do pai elite (ρ_e).

A função Q assume um estado (s) para cada parâmetro, enquanto as ações (a) correspondem aos possíveis valores no intervalo recomendado para cada parâmetro. A função Q define os parâmetros que serão utilizados no BRKGA e no *Q-Learning*. Assim sendo, aplica-se a política ϵ-*Greedy* de modo a determinar o valor de cada parâmetro em cada geração do BRKGA.

A lista de ações dos parâmetros corresponde aos valores recomendados na literatura para o BRKGA [Prasetyo et al., 2015] e para o *Q-Learning* [Öztop et al., 2020], e são descritos na Tabela 1. O tamanho da população (p) é definido pela sequência de *Fibonacci*, evitando o trabalho manual de especificação. A sequência de *Fibonacci* é aplicada uma vez que apresenta um comportamento uniforme de crescimento da população a uma taxa que pode ser controlada [Evans et al., 2020].

Tabela 1: Lista de ações dos parâmetros

Parâmetro	Lista de ações
p	{233; 377; 610; 987; 1597; 2584}
p_e	{0,30; 0,25; 0,20; 0,15; 0,10}
p_m	{0,25; 0,20; 0,15; 0,10; 0,05}
ρ_e	{0,80; 0,75; 0,70; 0,65; 0,60; 0,55}
ϵ	{0,10; 0,15; 0,20; 0,25; 0,30}
lf	{0,2; 0,4; 0,6; 0,8; 1,0}
df	{0,2; 0,4; 0,6; 0,8; 1,0}

O Algoritmo BRKGA-QL apresenta sete passos, que podem ser representados através do pseudocódigo do Algoritmo 1. O passo 1 consiste na inicialização dos valores de Q zerados. Em sequência, no passo 2, é gerada a população inicial representada por p cromossomos (vetores de chaves aleatórias). Cada cromossomo é composto por $n + 1$ genes, sendo n definido conforme o problema. A última posição do vetor é utilizada para definir qual função *decoder* será utilizada. Cada gene assume um valor gerado aleatoriamente no intervalo [0,1[. No passo 3 avalia-se a qualidade dos cromossomos por meio da função determinística *decoder* (descrita na subseção 3.3.1) e ordena-se a população.

Os passos 4-7 definem as gerações do BRKGA. No passo 4 aplica-se a política ϵ-*Greedy* em Q, determinando os valores para cada parâmetro. O passo 5 consiste no processo evolutivo do BRKGA-QL conforme descrito na subseção 3.1. Cada novo cromossomo gerado é avaliado pelo *decoder* definido em sua posição $n + 1$, e em seguida a nova população é ordenada.

O passo 6 verifica se o algoritmo encontrou uma solução de melhor qualidade nesta geração a partir das configurações de parâmetros selecionadas. Caso verdadeiro, atualiza-se a melhor solução encontrada (s^*), define-se o valor de recompensa como 1 ($R = 1$) e atualiza-se Q, por meio da Equação (10), para todos os pares estado/ação selecionados no passo 4.

O último passo consiste na verificação do tempo máximo de execução do algoritmo, através do parâmetro run_{time}. Esse parâmetro é configurado conforme os recursos computacionais e objetivos específicos do problema, sendo este o único parâmetro que o usuário precisa configurar no BRKGA-QL.

3.3.1. *Decoder*

Neste trabalho foram desenvolvidos cinco *decoders* para o FTSP. Em todos os casos, a solução inicial obtida é gerada pela ordenação do vetor de chaves aleatórias em ordem crescente. A partir desta sequência, são aplicadas as funções determinísticas, as quais são heurísticas de refinamento e heurísticas construtivas.

Algoritmo 1 *Biased Random-Key Genetic Algorithm* com *Q-Learning*

procedimento BRKGA-QL(run_{time})
  Passo 1. Inicializar valores de Q
  Passo 2. Gerar população Pop com p cromossomos de $n + 1$ genes
  Passo 3. Aplicar *decoder* para avaliar a qualidade de cada cromossomo e ordenar Pop
  Passo 4. Escolher uma ação para cada parâmetro usando ϵ-*Greedy* em Q
  Passo 5. Processo evolutivo:
    a. Particionar Pop em Elite (P_e, com $p * p_e$ melhores cromossomos) e Não-Elite
    b. Inicializar a população da próxima geração $Pop^+ = P_e$
    c. Gerar um conjunto P_m com $p * p_m$ mutantes e adicionar P_m na próxima geração $Pop^+ = Pop^+ + P_m$
    d. Gerar $p - |P_e| - |P_m|$ filhos com *Parametric Uniform Crossover* e adicionar na próxima geração Pop^+
    e. Aplicar o *decoder* nos novos cromossomos, atualizar a população $Pop = Pop^+$ e ordenar Pop
  Passo 6. **se** $(fitness(Pop_{best}) < fitness(s^*))$ **então**
    $s^* = Pop_{best}$
    $R = 1$ e atualizar Q
  Passo 7. Se o tempo computacional é maior que run_{time}, retorne s^*. Caso contrário, volte para o *Passo 4*.
fim procedimento

No *FTSP* a rota inicia-se e termina no depósito, o qual é representado pelo nó 0 e não foi incluído no cromossomo. A partir do depósito, os nós são percorridos na ordem definida pelo *decoder*. Entretanto, o caixeiro viajante não visita todos os nós, visto que o objetivo é realizar apenas um número exato predefinido de visitas. Para tanto, uma condição de contagem do número de visitas para cada família é estabelecida. Sendo assim, para cada nó visitado é realizado uma verificação se o número total de visitas para a família correspondente já está satisfeito ou não, antes de inseri-lo na rota. Caso uma família já esteja completa, qualquer nó correspondente à mesma que venha na sequência é descartado da rota. Caso contrário, se qualquer das famílias ainda não estiverem completas, os nós pertencentes as mesmas são inseridos na rota, repetindo esse processo até completar todas as famílias. A função objetivo (*fitness*) realiza o cálculo da distância da rota obtida, incluindo a distância do depósito até o primeiro nó da rota, assim como a distância do último nó ao depósito.

As funções *decoder* implementadas são:

1. Básico: a rota configura-se na sequência obtida pela ordenação do vetor de chaves aleatórias, apenas respeitando o critério de visitas para cada família;

2. *2-Opt*: a partir da sequência obtida pelo *decoder* Básico, aplica-se uma iteração completa da heurística *2-Opt* usando a estratégia *first improvement*;

3. *Cheapest Insertion*: com a ordenação obtida pelo vetor de chaves aleatórias, os três primeiros nós são mantidos configurando uma subrota e a partir do quarto nó da sequência é calculada a posição mais barata para inserção de cada nó;

4. *k-Farthest Insertion*: com a ordenação obtida pelo vetor de chaves aleatórias, apenas o primeiro nó da sequência é mantido e a partir do segundo nó encontra-se o nó i, entre os k primeiros nós da sequência, que seja o mais distante da rota parcial. O nó selecionado é inserido na rota na posição com menor custo de inserção. Neste trabalho foi definido $k = 3$;

5. *k-Nearest Insertion*: o procedimento deste *decoder* apresenta a mesma configuração do *decoder* anterior, contudo diferencia-se pela busca do nó i mais próximo da rota parcial, entre os k primeiros nós da sequência.

4. Experimentos Computacionais

O BRKGA-QL foi codificado em C++ a partir do *framework* disponível em `https://github.com/antoniochaves19/BRKGA-QL` e os experimentos computacionais foram executados em uma *workstation* com dois processadores Intel Xeon Silver 4114, memória RAM de 96GB e sistema operacional CentOS Linux 8.2. O critério de parada do BRKGA-QL (run_{time}, em segundos) foi definido como o número de pontos (n) de cada instância.

Os experimentos foram realizados em 85 instâncias disponíveis no site dedicado ao FTSP (`http://familytsp.rd.ciencias.ulisboa.pt/`). As instâncias estudadas estão divididas em dois conjuntos. O primeiro conjunto corresponde a instâncias de *benchmark*, com 21 instâncias com número de nós variando de 14 e 1002. O segundo conjunto é representado por instâncias baseadas em instâncias simétricas da TSPLib, com 64 instâncias e uma variedade de nós entre 136 e 264. Estas instâncias foram criadas como um complemento às instâncias de *benchmark* por [Bernardino e Paias, 2021]. No conjunto de instâncias *benchmark* conhece-se o valor ótimo de 13 instâncias, enquanto que no conjunto de instâncias simétricas conhece-se o valor ótimo de 49 instâncias.

Para análise dos resultados, foram elaboradas quatro tabelas organizadas em: instâncias de *benchmark* com valor ótimo conhecido (Tabela 2), instâncias de *benchmark* com melhor limitante superior conhecido (Tabela 3), instâncias simétricas com valor ótimo conhecido (Tabela 4) e instâncias simétricas com melhor limitante superior conhecido (Tabela 5). As tabelas apresentam a comparação entre os resultados computacionais obtidos pelo BRKGA-QL e os métodos apresentados na literatura: GA [Morán-Mirabal et al., 2014], ILS [Bernardino e Paias, 2018] e Algoritmo Híbrido [Bernardino e Paias, 2021]. A coluna S^* apresenta as soluções ótimas e a coluna *UB* indica os melhores limitantes superiores obtidos por um algoritmo *branch-and-cut* descrito em [Bernardino e Paias, 2021] com limite de tempo de 10800 segundos. As colunas *S* descrevem as melhores soluções obtidas por cada método. A coluna *gap* calcula a diferença entre as melhores soluções obtidas pelos métodos e a melhor solução conhecida na literatura (*BKS*), definida por $100 \times (BKS - S)/BKS$, e a coluna *T* apresenta o tempo total, em segundos, que cada método gastou para encontrar a melhor solução.

A Tabela 2 apresenta a comparação dos resultados obtidos pelos testes computacionais com as instâncias de *benchmark* com valor ótimo conhecido na literatura. Para o BRKGA-QL foram encontradas soluções ótimas para 10 instâncias, que representa 76,98% de eficácia nesse conjunto e *gap* médio de -0,41%. Desta forma, observa-se um *gap* médio inferior aos apresentados pelo GA e pelo algoritmo híbrido.

Para o conjunto de instâncias de *benchmark* com melhor limitante superior conhecido (Tabela 3), o BRKGA-QL foi capaz de obter um resultado melhor que o limitante superior conhecido na literatura para a instância a280_2. Entretanto, o método não atingiu os resultados da literatura para as instâncias de dimensões superiores. Nesse conjunto de instâncias, o tempo computacional do BRKGA-QL para encontrar a melhor solução foi em média 283 segundos, e apresentou um *gap* médio de -2,54%, enquanto os *gaps* médios dos métodos GA, ILS e híbrido foram -4,71%, -1,01% e -0,49%, respectivamente.

Os resultados obtidos para o conjunto de instâncias simétricas com valor ótimo conhecido (Tabela 4) mostram uma boa convergência do método BRKGA-QL, visto que o *gap* médio apresentado foi de -0,69% e é melhor que os resultados obtidos pelo algoritmo híbrido (-1,05%). Não encontramos na literatura os resultados dos métodos GA e ILS para este conjunto de instâncias. O BRKGA-QL encontrou as soluções ótimas para 16 instâncias deste conjunto, enquanto o algoritmo híbrido encontrou 12 soluções ótimas.

Tabela 2: Instancias de *benchmark* com valor ótimo conhecido.

Instância	*S**	BRKGA-QL			GA			ILS			Algoritmo híbrido		
		S	*gap*	*T*	*S*	*gap*	*T*	*S*	*gap*	*T*	*S*	*gap*	*T*
burma14_1	13,93	**13,93**	0,00	0,01	**13,93**	0,00	0,00	**13,93**	0,00	0,00	**13,93**	0,00	0,00
burma14_2	25,66	**25,66**	0,00	0,02	**25,66**	0,00	0,00	**25,66**	0,00	0,00	**25,66**	0,00	0,00
burma14_3	11,89	**11,89**	0,00	0,03	**11,89**	0,00	0,00	**11,89**	0,00	0,00	**11,89**	0,00	0,00
bayg29_1	5345,89	**5345,86**	0,00	0,07	**5345,89**	0,00	3,40	**5345,86**	0,00	0,00	**5345,89**	0,00	0,00
bayg29_2	5791,01	**5791,01**	0,00	0,04	**5791,01**	0,00	1,40	**5791,01**	0,00	0,00	**5791,01**	0,00	0,00
bayg29_3	5544,33	**5544,33**	0,00	0,22	**5544,33**	0,00	1,60	**5544,33**	0,00	0,00	**5544,33**	0,00	0,00
att48_1	23686,00	**23686,02**	0,00	16,37	**23686,02**	0,00	143,40	**23686,00**	0,00	0,00	**23686,02**	0,00	0,00
att48_2	20609,10	**20609,09**	0,00	0,80	**20609,09**	0,00	62,80	**20609,10**	0,00	0,00	**20609,08**	0,00	1,00
att48_3	9024,58	**9024,58**	0,00	1,68	**9024,58**	0,00	1,80	**9024,58**	0,00	0,00	**9024,58**	0,00	1,00
bier127_1	33709,70	**33709,75**	0,00	47,95	36913,74	-9,50	498,00	**33709,70**	0,00	4,00	34100,3	-1,16	8,00
bier127_2	88736,40	89519,04	-0,88	47,94	98216,10	-10,68	1413,80	**88736,40**	0,00	4,00	89147,93	-0,46	7,00
bier127_3	47726,30	48293,59	-1,19	60,63	50513,10	-5,84	1048,60	**47726,30**	0,00	3,00	48374,13	-1,36	6,00
a280_3	1408,14	1453,69	-3,23	129,02	1720,23	-22,16	14758,40	**1408,14**	0,00	36,00	1454,07	-3,26	50,00
			-0,41			-3,71			0,00			-0,48	

Tabela 3: Instâncias de *benchmark* com melhor limitante superior conhecido.

Instância	*UB*	BRKGA-QL			GA			ILS			Algoritmo híbrido		
		S	*gap*	*T*	*S*	*gap*	*T*	*S*	*gap*	*T*	*S*	*gap*	*T*
a280_1	1692,92	1750,50	-3,40	94,28	1824,04	-7,75	10,00	1769,33	-4,51	45,00	**1739,30**	-2,74	63,00
a280_2	1568,56	**1549,53**	1,21*	112,69	1642,87	-4,74	12,00	1562,14	0,41	45,00	1556,08	0,80	58,00
gr666_1	1531,93	1569,44	-2,45	277,88	1579,93	-3,13	84,00	1554,86	-1,50	721,00	**1531,93**	0,00	906,00
gr666_2	1236,21	1300,86	-5,23	155,71	1310,33	-6,00	86,00	**1236,21**	0,00	700,00	1237,03	-0,07	914,00
gr666_3	1225,61	1263,02	-3,05	165,93	1325,11	-8,12	88,00	1235,48	-0,81	752,00	**1225,61**	0,00	968,00
pr1002_1	132821,40	136566,87	-2,82	406,62	136796,00	-2,99	237,00	**132821,40**	0,00	2037,00	134625,52	-1,36	3008,00
pr1002_2	144543,77	150667,32	-4,24	518,81	151682,00	-4,94	257,00	**144543,77**	0,00	2013,00	145326,69	-0,54	2537,00
pr1002_3	126282,93	126720,40	-0,35	535,99	126307,00	-0,02	230,00	128367,78	-1,65	2037,00	**126282,93**	0,00	3085,00
			-2,54			-4,71			-1,01			-0,49	

Tabela 4: Instâncias Simétricas com valor ótimo conhecido.

Instância	*S**	BRKGA-QL			Algoritmo híbrido			*Instância*	*S**	BRKGA-QL			Algoritmo híbrido		
		S	*gap*	*T*	*S*	*gap*	*T*			*S*	*gap*	*T*	*S*	*gap*	*T*
pr136_1	61448	**61942**	-0,80	50,746	61948	-0,81	7	rat195_2	912	**918**	-0,66	35,312	921	-0,99	21
pr136_2	43522	**43522**	0,00	23,432	44209	-1,58	7	d198_1	10945	**10974**	-0,26	59,727	10988	-0,39	27
pr136_3	81481	81713	-0,28	90,956	**81532**	-0,06	12	d198_2	10212	10245	-0,32	32,429	**10237**	-0,24	35
pr136_4	63246	**63321**	-0,12	54,503	**63321**	-0,12	6	d198_3	13843	**13868**	-0,18	63,668	13967	-0,90	17
gr137_1	44263	**44263**	0,00	82,966	44391	-0,29	11	d198_4	12418	**12426**	-0,06	85,481	12429	-0,09	46
gr137_2	36435	**36435**	0,00	55,651	**36435**	0,00	7	kroA200_2	12416	**12427**	-0,09	85,51	12973	-4,49	22
gr137_3	55919	**55919**	0,00	29,569	56265	-0,62	6	kroA200_4	16518	**16518**	0,00	57,93	16712	-1,17	16
gr137_4	46620	**46620**	0,00	38,241	**46620**	0,00	7	kroB200_1	17527	**17618**	-0,52	86,097	17755	-1,30	23
pr144_1	46376	**46376**	0,00	22,427	**46376**	0,00	7	kroB200_2	12881	**13037**	-1,21	87,784	13221	-2,64	22
pr144_2	36518	**36518**	0,00	87,652	**36518**	0,00	8	kroB200_4	17583	17661	-0,44	66,779	**17605**	-0,13	15
pr144_3	54635	**54635**	0,00	22,476	54869	-0,43	5	gr202_1	23394	23635	-1,03	72,979	**23604**	-0,90	64
kroA150_1	14307	**14394**	-0,61	22,511	14471	-1,15	15	gr202_2	14957	15076	-0,80	82,906	**15018**	-0,41	66
kroA150_2	9481	10170	-7,27	46,513	**9496**	-0,16	12	gr202_3	34547	**34562**	-0,04	83,045	34746	-0,58	230
kroA150_4	13404	13412	-0,06	34,062	**13404**	0,00	10	pr226_1	52109	**52109**	0,00	43,01	52286	-0,34	77
kroB150_1	14522	14873	-2,42	74,767	**14661**	-0,96	16	pr226_2	47585	**47585**	0,00	58,337	47628	-0,09	29
kroB150_2	9555	10073	-5,42	33,985	**10015**	-4,81	12	pr226_3	66812	**66812**	0,00	46,761	67286	-0,71	20
kroB150_3	19925	**19925**	0,00	59,646	19995	-0,35	11	pr226_4	51905	51944	-0,08	32,864	**51905**	0,00	18
kroB150_4	12532	**12532**	0,00	26,361	12587	-0,44	9	gr229_2	31653	31922	-0,85	98,519	**31789**	-0,43	40
pr152_1	51806	**51860**	-0,10	56,043	51925	-0,23	10	gr229_3	102841	**103595**	-0,73	62,679	105463	-2,55	37
pr152_2	45810	46677	-1,89	43,722	**45902**	-0,20	19	gr229_4	46231	46919	-1,49	74,225	**46522**	-0,63	26
pr152_4	57337	**58114**	-1,36	65,066	58147	-1,41	30	gil262_2	1069	1096	-2,53	115,644	**1077**	-0,75	75
u159_1	29821	**29821**	0,00	52,379	30035	-0,72	19	pr264_1	33904	34145	-0,71	109,986	**34039**	-0,40	192
u159_2	23404	**23559**	-0,66	55,451	26760	-14,34	19	pr264_3	40705	**40737**	-0,08	105,037	41049	-0,85	30
u159_3	36399	**36399**	0,00	26,704	36597	-0,54	10	pr264_4	35153	35383	-0,65	113,044	**35325**	-0,49	201
u159_4	30845	**30933**	-0,29	41,14	31057	-0,69	37	Média			-0,69			-1,05	

Tabela 5: Instâncias Simétricas com melhor limitante superior conhecido.

Instância	*UB*	BRKGA-QL			GA			ILS			Algoritmo híbrido		
		S	*gap*	*T*	*S*	*gap*	*T*	*S*	*gap*	*T*	*S*	*gap*	*T*
pr144_4	49379	**49379**	0,00	34,344	49763	-0,78	3	**49379**	0,00	6	**49379**	0,00	15
kroA150_3	20880	**20911**	-0,15	50,705	21748	-4,16	4	21459	-2,77	8	21038	-0,76	11
pr152_3	64425	**64607**	-0,28	63,027	65743	-2,05	3	65339	-1,42	7	64757	-0,52	8
rat195_1	1285	**1278***	0,54	69,352	1321	-2,80	4	1303	-1,40	14	1300	-1,17	146
rat195_3	1814	**1806***	0,44	96,384	1867	-2,92	4	1838	-1,32	10	1849	-1,93	14
rat195_4	1320	**1320**	0,00	67,037	1383	-4,77	4	1342	-1,67	11	1330	-0,76	124
kroA200_1	16441	**16432***	0,05	65,313	17176	-4,47	5	17134	-4,22	15	16767	-1,98	19
kroA200_3	24471	**24301***	0,69	59,38	24898	-1,74	2	25190	-2,94	11	24609	-0,56	1814
kroB200_3	24088	**24102**	-0,06	109,431	25250	-4,82	5	25225	-4,72	10	24542	-1,88	178
gr202_4	28039	**28122**	-0,30	38,543	28634	-2,12	4	28846	-2,88	13	28285	-0,88	39
gr229_1	70741	**70757**	-0,02	84,629	74768	-5,69	7	72288	-2,19	28	71612	-1,23	366
gil262_1	1526	1546	-1,31	55,907	1601	-4,91	6	1567	-2,69	26	**1526**	0,00	197
gil262_3	1988	**1978***	0,50	125,734	2085	-4,88	8	2067	-3,97	19	1988	0,00	1822
gil262_4	1671	1673	-0,12	75,115	1722	-3,05	7	1731	-3,59	23	**1671**	0,00	1827
pr264_2	28748	**28564***	0,64	128,471	29322	-2,00	7	28748	0,00	28	28778	-0,10	66
			0,04			-3,41			-2,38			-0,78	

Conforme observado na Tabela 5, o método BRKGA-QL apresentou melhor convergência se comparado aos resultados obtidos na literatura para o conjunto de instâncias simétricas em que a solução ótima não é conhecida. Para esse conjunto, o método encontrou as melhores soluções conhecidas da literatura em 8 instâncias. Enquanto o ILS encontrou a melhor solução para 2 instâncias e o algoritmo híbrido para 4 instâncias. Nota-se que, comparando com as soluções obtidas pelos três métodos heurísticos da literatura, o BRKGA-QL encontrou melhores soluções em 13 de 15 instâncias. Além disso, tem-se um *gap* médio de 0,04%, tendo encontrado novos limitantes superiores para 6 instâncias (marcadas com * na tabela).

Foram realizadas análises estatísticas para os resultados obtidos. O desvio percentual (RPD, do inglês *relative percentage deviation*) foi calculado considerando os resultados obtidos em 5 execuções para cada instância. Para tanto, foi calculado o desvio percentual de cada execução, definido por $100 \times (S - BKS)/BKS$ e o RPD correspondente a média dos 5 desvios.

A Figura 1 apresenta os *boxplots* relativos aos RPDs das soluções obtidas pelo BRKGA-QL nas instâncias do *benchmark* e nas instâncias simétricas. Pode-se observar que embora o BRKGA-QL não tenha encontrado as melhores soluções para as instâncias *benchmark*, em 75% dos resultados obtidos a solução média está a 6% da melhor solução conhecida. Além disso, é possível constatar que a mediana encontra-se em 2, o que significa que em 50% dos casos a solução média obtida está em 2% da melhor solução conhecida. As instâncias simétricas apresentaram menor variabilidade em seus resultados e valor médio próximo das melhores soluções conhecidas.

Também foram comparados os resultados obtidos pelo BRKGA e GRASP de [Morán-Mirabal et al., 2014] e dentre as 21 instâncias resolvidas, o BRKGA-QL foi capaz de encontrar melhores soluções que ambos os métodos.

Em relação ao BRKGA proposto inicialmente por [Morán-Mirabal et al., 2014], nosso método encontrou melhores soluções em todas as instâncias *benchmark* testadas. O *gap* médio do BRKGA-QL para este conjunto foi -1,22% enquanto que o *gap* médio do BRKGA foi de -9,50%.

Por fim, foi realizada uma análise estatística para comparar o BRKGA-QL com o algoritmo híbrido de [Bernardino e Paias, 2021], que é a heurística estado-da-arte para o FTSP. Para tal foi aplicado o *Wilcoxon Signed-Rank test* (WSR) [Rey e Neuhauser, 2014] sobre os resultados para as instâncias *benchmark* e simétricas. O WSR indica que os resultados do BRKGA-QL não são estatisticamente significantes em relação aos resultados do algoritmo híbrido ($z = 888$ e *p*-

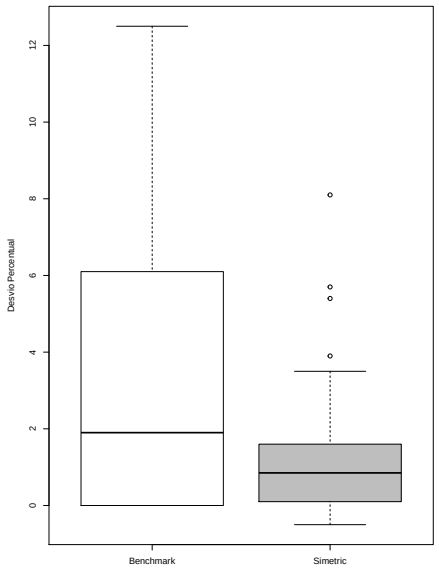


Figura 1: Desvio percentual (RPD) do BRKGA-QL considerando 5 execuções para cada instância.

value $= 0.025$). Estes resultados sugerem que ambos métodos produzem resultados similares para o FTSP em termos de qualidade de solução.

5. Conclusão

Neste artigo foi apresentada uma variante do problema do caixeiro viajante que considera famílias, ou agrupamento de pontos, e número predefinido de visitas a cada família, conhecida na literatura como *Family Traveling Salesman Problem* (FTSP). Uma metaheurística híbrida BRKGA-QL foi aplicada ao FTSP. Esta metaheurística é composta por um BRKGA e um algoritmo de aprendizagem por reforço *Q-Learning* utilizado para configurar os parâmetros do BRKGA em tempo de execução.

Foram realizados experimentos computacionais em 85 instâncias disponíveis na literatura. O método BRKGA-QL se mostrou eficiente tanto na qualidade das soluções quanto no tempo computacional, sendo capaz de melhorar os limitantes superiores conhecidos em 7 instâncias e encontrar a solução ótima conhecida para 33 instâncias. O BRKGA-QL mostrou-se melhor que à heurística estado-da-arte proposta por [Bernardino e Paias, 2021] para instâncias com até 300 pontos, mas para instâncias maiores os resultados foram piores que os da literatura. De toda forma, o BRKGA-QL apresentou *gap* igual ou melhor em 68% das instâncias.

Como trabalhos futuros pretende-se elaborar heurísticas de busca local específicas para o FTSP para intensificar a busca em regiões promissoras e aplicar o BRKGA-QL em outros problemas de otimização.

Agradecimentos

Essa pesquisa foi parcialmente financiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (processos 423694/2018-9, 303736/2018-6), Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (processos 2018/15417-8, 2020/03408-4), e pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

Bernardino, R. e Paias, A. (2018). Solving the family traveling salesman problem. *European Journal of Operational Research*, 267(2):453–466. ISSN 0377-2217.

Bernardino, R. e Paias, A. (2021). Heuristic approaches for the family traveling salesman problem. *International Transactions in Operational Research*, 28(1):262–295.

Chaves, A. A. e Lorena, L. H. N. (2021). An adaptive and near parameter-free brkga using q-learning method. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, p. 1–8. IEEE. URL `http://www.optimization-online.org/DB_HTML/2021/02/8260.html`.

De Lima, F. C., De Melo, J. D., e Neto, A. D. D. (2007). Using Q-learning algorithm for initialization of the GRASP metaheuristic and genetic algorithm. In *IEEE International Conference on Neural Networks - Conference Proceedings*. ISBN 142441380X.

Evans, B. P., Xue, B., e Zhang, M. (2020). An adaptive and near parameter-free evolutionary computation approach towards true automation in automl. *arXiv preprint arXiv:2001.10178*.

Gonçalves, J. F. e Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Morán-Mirabal, L., González-Velarde, J., e Resende, M. G. (2014). Randomized heuristics for the family traveling salesperson problem. *International Transactions in Operational Research*, 21 (1):41–57.

Öztop, H., Tasgetiren, M. F., Kandiller, L., e Pan, Q.-K. (2020). A novel general variable neighborhood search through q-learning for no-idle flowshop scheduling. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, p. 1–8.

Prasetyo, H., Fauza, G., Amer, Y., e Lee, S. (2015). Survey on applications of biased-random key genetic algorithms for solving optimization problems. In *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*, p. 863–870. IEEE.

Rey, D. e Neuhauser, M. (2014). Wilcoxon-signed-rank test. In Lovric, M., editor, *International Encyclopedia of Statistical Science*, p. 1658–1659. Springer Berlin Heidelberg.

Spears, W. M. e Jong, K. A. D. (1991). On the virtues of parameterized uniform crossover. *Proc. of the Fourth International Conference on Genetic Algorithms*, p. 230–236.

Sutton, R. e Barto, A. (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*. ISSN 1045-9227.

Toso, R. e Resende, M. (2015). A c++application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93.

Watkins, C. J. C. H. e Dayan, P. (1992). Q-learning. *Machine Learning*. ISSN 0885-6125.