Um algoritmo genético híbrido para o problema do caixeiro viajante com tempos de liberação

Gabriel Soares[1], **Teobaldo Bulhões**[2], **Bruno Bruck**[2]

[1]Programa de Pós-graduação em Informática – Universidade Federal da Paraíba (UFPB)
Rua dos Escoteiros s/n – 58055-000 – João Pessoa – PB – Brasil

[2]Departamento de Computação Científica – Universidade Federal da Paraíba (UFPB)
Rua dos Escoteiros s/n – 58055-000 – João Pessoa – PB – Brasil

soaresggm@gmail.com, tbulhoes@ci.ufpb.br, bruno.bruck@ci.ufpb.br

Abstract. *The focus of this research paper is on a modified version of the traditional traveling salesman problem, in which each customer is linked to a release date indicating when the requested product will be available at the depot. The problem involves using a single uncapacited vehicle to serve all customers by making multiple trips. However, the vehicle cannot start a route until all the products associated with the route's demands have been released, leading to possible waiting times before starting the next route. The objective is to minimize the completion time of the last route, which refers to the time taken by the vehicle to return to the depot after fulfilling all demands. To address this problem, this paper proposes a hybrid genetic algorithm that includes more advanced techniques for evaluating individuals and promoting population diversity. Additionally, a new dynamic programming splitting algorithm is introduced, which converts the giant-tour sequence of customer visits into the optimal set of routes that maintains the sequence. The algorithm was able to find the optimal solution for all 154 instances with known optima and found better upper bounds for 364 instances, in significatly less time when compared to the state-of-the-art algorithm.*

Resumo. *O foco deste artigo é uma variante do problema do caixeiro-viajante clássico, no qual cada cliente está associado a um tempo de liberação indicando quando o produto solicitado estará disponível no depósito. O problema envolve o uso de um único veículo sem capacidade para atender todos os clientes fazendo múltiplas viagens. No entanto, o veículo não pode iniciar uma rota até que todos os produtos associados às demandas da rota tenham sido liberados, levando a possíveis tempos de espera antes de iniciar a próxima rota. O objetivo é minimizar o tempo de término da última rota, que se refere ao tempo que o veículo leva para retornar ao depósito após atender todas as demandas. Para resolver este problema, este artigo propõe um algoritmo genético híbrido que inclui técnicas mais avançadas para avaliar indivíduos e promover diversidade na população. Além disso, é introduzido um novo algoritmo de split com programação dinâmica, que converte a sequência de visita do cliente em um conjunto ótimo de rotas que mantém a sequência. O algoritmo conseguiu encontrar a solução ótima para todas as 154 instâncias com ótimos conhecidos e encontrou melhores limites superiores para 364 instâncias, em significativamente menos tempo quando comparado ao algoritmo de estado-da-arte.*

1. Introdução

O Problema do Caixeiro Viajante com tempos de liberação (PCVTL), é uma variante do Problema do Caixeiro Viajante clássico no qual a cada pacote a ser entregue está associado um tempo de liberação, que é o tempo a partir do qual aquele pacote pode sair do depósito. O problema está definido sobre um grafo completo $G = (V, A)$. A cada arco $(i, j) \in A$ está associado um tempo de deslocamento denotado por $t(i, j)$. O conjunto de vértices $V = \{0\} \cup N$ é composto pelo vértice 0, que representa o depósito, e o conjunto $N = \{1, 2, ..., n\}$, que representa os clientes. O tempo de liberação para o cliente $i \in N$ é denotado por $r(i)$. Dessa forma, um veículo não capacitado deve realizar um conjunto de rotas para atender todos os clientes, podendo voltar ao depósito diversas vezes para pegar produtos que vão sendo liberados ao longo de um horizonte de tempo, podendo ou não ter tempo de espera entre as rotas. Este trabalho trata da variante do problema em que o objetivo é minimizar o tempo de término das rotas (PCVTL(tempo)), que é a soma dos tempos de deslocamento mais os tempos de espera no depósito, ou simplesmente o tempo que o veículo retorna ao depósito depois de ter atendido a todos os clientes.

Esse problema pode ser observado principalmente nos sistemas de entrega rápida, no qual um centro de distribuição recebe produtos ao decorrer de um dia, ou em sistemas de entrega no mesmo dia, em que os produtos já estão no depósito, mas pedidos são recebidos ao longo do dia. Os produtos necessitam ser entregues aos clientes em um mesmo dia, dessa forma o tempo de chegada dos produtos deve ser levado em consideração na etapa de planejamento das rotas.

O PCVTL foi introduzido por [Archetti et al. 2015]. Os primeiros métodos para resolução desse problema foram apresentados por [Archetti et al. 2018], que introduziram uma heurística *ILS* e um modelo de programação matemática que conseguiu o resultado ótimo para instâncias com até 20 clientes. Um novo modelo foi apresentado por [Montero et al. 2021] que usa um procedimento de *branch-and-cut* para resolver o problema, e conseguiu o resultado ótimo para todas as instâncias com até 30 clientes, e para algumas com até 50 clientes.

Neste trabalho é proposto uma heurística baseada em Algoritmos Genéticos para resolver o problema, o qual usa uma representação das rotas em *giant-tour*. Introduzimos um procedimento exato e eficiente de *Split*, através de uma programação dinâmica, o qual transforma a representação em *giant-tour* das rotas no melhor conjunto de rotas possíveis que respeita a sequência de visitação definida pelo *giant-tour*.

2. Algoritmo genético híbrido

O procedimento proposto é baseado na meta-heurística *unified hybrid genetic search* (UHGS) de [Vidal et al. 2014], mas não se considera uma população de soluções inviáveis, visto que estas não existem para esse problema. O procedimento inicia gerando uma população inicial, e então iterativamente gera novos indivíduos filhos através do cruzamento de dois indivíduos pais que são selecionados. Esse filho é então refinado na etapa de educação, no qual são realizados procedimentos de busca local, e então é adicionado à população. Quando a população atinge o tamanho máximo, é acionado um procedimento de seleção de sobreviventes, de forma que alguns indivíduos são descartados e preservados apenas os melhores. Esses indivíduos são avaliados não só pelo seu custo, mas também pela contribuição que o indivíduo tem para a diversidade da população. Uma

vez que são feitas It_{div} iterações sem obter melhora na solução, aciona-se um procedimento de diversificação da população, onde as piores soluções em relação ao *biased fitness* são descartadas, e novas soluções aleatórias são inseridas na população, com o intuito de explorar novas regiões no espaço de soluções. Quando é atingido It_{NI} iterações sem melhora, o procedimento é finalizado, e é retornada a melhor solução encontrada até então.

Um indivíduo é avaliado de acordo com a métrica de *biased fitness* proposta por [Vidal et al. 2012], que leva em consideração o valor da solução na função objetivo, e a contribuição do indivíduo para a diversidade da população. Essa métrica usa as distâncias entre soluções para calcular a contribuição para a diversidade, sendo a função de distância usada para o PCVTL(tempo) calculada como a proporção dos arcos em comum entre as duas soluções, como mostrado na Equação (1).

$$\delta(P_1, P_2) = 1 - \frac{|arcos(P_1) \cap arcos(P_2)|}{|arcos(P_1) \cup arcos(P_2)|} \tag{1}$$

Na etapa de cruzamento os pais são selecionados através de um torneio binário, no qual para selecionar cada um dos pais, é selecionado primeiro dois indivíduos aleatoriamente da população, e escolhido aquele com melhor *biased fitness*. É aplicado então o operador clássico de cruzamento *Order Crossover*, apresentado em [Davis 1991], para gerar uma solução filha.

Cada indivíduo da população é representado pelo cromossomo *giant-tour*, que é montado a partir da sequência em que os clientes são visitados nas rotas, sem as visitas ao depósito. Um procedimento adicional, comumente chamado de *Split*, é necessário para transformar essa representação em uma solução viável do problema. O algoritmo de *Split* proposto é apresentado a seguir.

2.1. O procedimento *Split* para o PCVTL(tempo)

O novo algoritmo de *Split* apresentado nesta seção determina o conjunto ótimo de rotas respeitando a sequência de visitação de um cromossomo *giant-tour* $(c_1^P, c_2^P, \ldots, c_n^P)$ de um indivíduo P. O algoritmo funciona sobre um conjunto de rotas $\mathcal{R} = \{R_{i,j} : i, j \in \{1, \ldots, n\}, i \leq j\}$, em que $R_{i,j}$ representa a rota $R_{i,j} = (0, c_i^P, c_{i+1}^P, \ldots, c_j^P, 0)$ que visita $j - i + 1$ clientes.

Considere que $r(R_{i,j})$ e $t(R_{i,j})$ são, respectivamente, o tempo de liberação e a duração da rota $R_{i,j}$, que pode ser calculado pelas Equações (2) e (3).

$$r(R_{i,j}) = \max_{k=i}^{j} r(c_k^P) \tag{2}$$

$$t(R_{i,j}) = t(0, c_i^P) + \left(\sum_{k=i+1}^{j} t(c_{k-1}^P, c_k^P) \right) + t(c_j^P, 0) \tag{3}$$

Seja $\sigma(R_{i,j})$ o menor tempo viável de inicio da rota $R_{i,j}$, numa solução que respeite a sequência de visitação definida por P. Para calcular esse valor, deve-se considerar que todos os clientes anteriores $c_1^P, \ldots, c_{i-1}^P$ já foram atendidos (possivelmente por várias rotas) de forma ótima. Note que $\sigma(R_{1,j}) = r(R_{1,j})$ para todas rotas $R_{1,j}$. Seja $\phi(j)$, com

$j \in \{1, \ldots, n\}$, o menor tempo de término possível da solução parcial que serve os j primeiros clientes $c_1^P, \ldots, c_j^P$ de P. Dessa forma se tem que:

$$\phi(j) = \min_{i=1}^{j} \left\{ \sigma(R_{i,j}) + t(R_{i,j}) \right\} \tag{4}$$

$$\sigma(R_{i,j}) = \begin{cases} \max\{\phi(i-1), r(R_{i,j})\} & \text{se } i > 1 \\ r(R_{i,j}) & \text{se } i = 1 \end{cases} \tag{5}$$

Dessa forma o tempo mínimo de término de uma solução respeitando a sequência de visitação definida por P pode ser obtida calculando $\phi(n)$. Essa equação foi resolvida através de um algoritmo de programação dinâmica cuja complexidade de tempo é $O(n^2)$, que calcula também as rotas associadas à solução ótima obtida.

2.2. Educação

Na etapa de *Educação*, são realizados procedimentos de Busca Local em vizinhanças da solução. Essa etapa é dividida em três fases: aperfeiçoamento intra-rota, aperfeiçoamento inter-rota e aperfeiçoamento de depósitos. Essas três fases são aplicadas na solução em sequência até que não seja obtida melhora em nenhuma das fases.

Na fase de aperfeiçoamento intra-rota são aplicadas em ordem aleatória os movimentos de vizinhanças clássicos: *Exchange(1, 1)*, *Exchange(1, 2)*, *Exchange(2, 2)*, *Or-opt*, *Or-opt2*, *2-opt*. Um movimento intra-rota é avaliado pelo tempo de término da rota em que ele opera, dessa forma cada avaliação de movimento individual tem complexidade de tempo constante.

No aperfeiçoamento inter-rota são aplicados em ordem aleatória os movimentos clássicos *Swap* e *Relocation* e um movimento que foi criado para esse problema chamado *Divide&Swap*, que avalia dividir rotas em um certo ponto e trocar a ordem das duas rotas resultantes. Movimentos inter-rotas são avaliados de acordo com o tempo de término da rota que é realizada por último, entre as duas rotas envolvidas na operação. Dessa forma a complexidade de tempo de avaliação de solução é $O(|\mathcal{R}(s)|)$, no qual $\mathcal{R}(s)$ representa o conjunto de rotas de uma solução s.

O aperfeiçoamento de depósito busca melhorar a solução alterando apenas quando o veículo deve retornar ao depósito, sem alterar a ordem de visitação dos clientes. Dessa forma, o *Split* é usado como o movimento de aperfeiçoamento de depósito, visto que ele é capaz de encontrar a forma ótima de visitar o depósito e tem uma complexidade de tempo menor do que a avaliação de movimentos de depósito clássicos.

3. Resultados

A implementação do algoritmo foi feita na linguagem de programação C++. A execução foi feita em um computador com processador Intel Xeon E5-2650 v4 com 2.2 GHz e 128 GB de memória RAM e sistema operacional Ubuntu 16.04. Foram rodados testes em um total de 522 instâncias, apresentadas por [Archetti et al. 2015] e [Montero et al. 2021]. O algoritmo foi capaz de encontrar a solução ótima de todas as 154 instâncias que tem ótimo conhecido, e melhorou o resultado de 364 instâncias, em tempos computacionais significativamente inferiores quando comparado ao algoritmo estado-da-arte para o problema.

Referências

Archetti, C., Feillet, D., Mor, A., and Speranza, M. G. (2018). An iterated local search for the traveling salesman problem with release dates and completion time minimization. *Computer & Operations Research*, 98:24–37.

Archetti, C., Feillet, D., and Speranza, M. G. (2015). Complexity of routing problems with release dates. *European Journal of Operational Research*, 247(3):797–803.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

Montero, A., Méndez-Díaz, I., and Miranda-Bront, J. J. (2021). Solving the traveling salesman problem with release dates via branch-and-cut. Technical report, Universidad de Buenos Aires.

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