XXIV ONPCE: Oficina Nacional de Problemas de Corte e Empacotamento, Planejamento e Programação da Produção e Correlatos

Um *framework* para o problema dinâmico de roteamento de veículos com coleta, entrega e janelas de tempo

Arthur Henrique Souza Cruz[1]
Departamento de Ciência da Computação, UFLA
Mayron César O. Moreira[2]
Departamento de Ciência da Computação, UFLA
Franklina Maria Bragion Toledo[3]
Instituto de Ciências Matemáticas e de Computação, USP

O Problema Dinâmico de Roteamento de Veículos com Coleta, Entrega e Janelas de Tempo (PDRVCEJT) consiste no plajemento eficiente de rotas para o atendimeno de pedidos ao longo do dia, sem que tenha-se o conhecimento de uma distribuição de probabilidade que estime o recebimento de um pedido. Um pacote deve ser coletado em um ponto e entregue a um destino diferente. Por fim, a coleta ou entrega de cada ponto deve ser realizada em uma determinada janela de tempo. Este trabalho tem inspiração em um caso real de uma empresa de consultoria, que presta serviços para o segmento de logistíca. Nesse contexto, variações do PDRVCEJT são comuns, devido a restrições particulares que aparecem para cada cliente. Diante disso, busca-se uma forma de flexibilizar, a nível de código, a instanciação de classes e objetos que sejam adaptáveis às diferentes extensões do PDRVCEJT. As contribuições deste trabalho consistem em:

1. Proposição de um *framework* para a generalização de soluções para o PDRVCEJT e variantes, utilizando conceitos de herança e polimorfismo;

2. Implementação em Python de uma metodologia para a instanciação de objetos, a nível de código, por meio de *strings* advindas de arquivos de configuração do tipo ".json".

A generalização de elementos necessários para a escrita do *framework* foi feita a partir da criação de superclasses abstratas para: (i) o leitor dos dados de entrada; (ii) as restrições do problema; (iii) as funções objetivo utilizadas para guiar o método de solução; (iv) os pontos (ou vértices) que devem ser percorridos; (v) as rotas; (vi) os algoritmos e métodos de solução; (vii) a estrutura de armazenamento de solução; e $(viii)$ as operações de inserção e remoção de pedidos de uma rota. As classes filhas de cada uma dessas podem ser definidas de forma a atender as especificidades das variantes do PDRVCEJT. Para mais detalhes sobre a estrutura de classes proposta, recomenda-se a leitura de [1].

O trecho de código apresentado pela Figura 1 importa uma classe a partir de uma *string* que representa seu nome. Considera-se que a classe possa ser importada em um módulo chamado *src*, presente no diretório principal do programa. Essa definição é feita utilizando um arquivo *__init__.py*. A importação é possível a partir da função *get_attr*, que permite que um o tipo

[1]arthurhscruz@gmail.com
[2]mayron.moreira@ufla.br
[3]fran@icmc.usp.br

de objeto seja encontrado a partir de seu nome, permitindo a instancialização de um objeto. Por exemplo, o tipo *ModeloMatemático* pode ser buscado e armazenado utilizando o comando *class_type = get_attr(src, "ModeloMatemático")* (Linhas 3-6 da Figura 1). A inicialização do objeto pode ser feita a partir variável na qual o valor foi armazenado, como pode ser visto na Linha 7 da Figura 1. Os parâmetros de entrada para um algoritmo podem ser representados como dicionários no formato *"atributo":valor*, e são atribuídos de forma semelhante, como mostrado nas Linhas 8-10.

Figura 1: Código para a instanciação de objetos a partir de *strings* [2].

```
1  import src
2  def create_class_by_name(class_name, class_data):
3      class_type = getattr(
4          src,
5          class_name
6      )
7      class_object = class_type()
8      for attribute, value in class_data.items():
9          if (getattr(class_object, attribute) is None):
10             class_object.set_attribute(attribute, value)
11     return class_object
```

Com o trecho de código apresentado na Figura 1, é possível utilizar um arquivo de configuração ".json" que permite a escolha entre diferentes heurísticas, restrições e funções objetivos. Exemplos de configurações de diferentes variações do PDRVCEJT e seus respectivos algoritmos podem ser vistos em `https://github.com/thuzax/vrp-solver/tree/master/solver`.

A ferramenta desenvolvida foi testada através da implementação do algoritmo de [3], utilizado na atualização das rotas a cada intervalo de tempo que novos pedidos surgem, no contexto dinâmico. A implementação em Python trouxe flexibilidade na instanciação de novos métodos correlatos às heurísticas de [3], apesar da perda de eficiência comparada à implementação em C++ no artigo original. Como trabalhos futuros, considera-se a possibilidade de utilizar o padrão de projeto Fábrica Abstrata e Decoradores, visto que são boas ferramentas para auxiliar a na generalização e identificação do tipo das classes. Para aprimorar a eficiência, sugere-se a implementação dos métodos de solução utilizando pacotes como *ctypes* (`https://docs.python.org/3/library/ctypes.html`). Por fim, uma interface gráfica pode ser desenvolvida para que a configuração do arquivo ".json" seja feita de forma mais intuitiva para o usuário.

Referências

[1] A. H. S. Cruz. Um estudo sobre o problema dinâmico de roteamento de veículos. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Lavras, Lavras, 110 p., 2022.

[2] A. H. S. Cruz, M. C. O. Moreira, F. M. B. Toledo. A Framework for the Dynamic Pickup and Delivery Problem with Time Windows. Submetido para a revista Expert Systems With Applications (segunda rodada de revisão), 2023.

[3] C. S. Sartori, L. S. Buriol. A study on the pickup and delivery problem with time windows: Matheuristics and new instances. Computers & Operations Research, Elsevier, p. 105065, 2020.