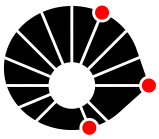

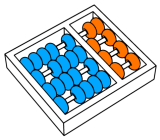

Problema de roteamento de veículos aplicado à entrega de compras

V. Coppo *F. K. Miyazawa*

Relatório Técnico - IC-PFG-22-28
Projeto Final de Graduação
2022 - Agosto

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Problema de roteamento de veículos aplicado à entrega de compras

Vitor Coppo Ferreira Flavio Keidi Miyazawa

Resumo

Este trabalho é relatório sobre o uso do problema de roteamento de veículos aplicado ao contexto de entrega de compras com dois tipos de veículos. A partir do trabalho desenvolvido, foi possível depreender que o uso de algoritmos exatos, por mais que seja possível para contextos pequenos e médios, se torna inviável com o crescimento do número de pontos de entrega.

1 Introdução

Nos dias de hoje, há acirrada competição entre empresas como resultado da facilitação do acesso à conhecimento e tecnologia, desencadeados por avanços em meios produtivos, de transporte e de comunicação. A fim de se destacar neste cenário, as empresas cada vez mais tem trabalhado para a condução de suas atividades de maneira mais eficiente, rápida e menos custosa, por meio de análises minuciosas de seus processos [1]. Neste contexto, destacam-se os esforços voltados à etapa de distribuição de mercadorias, que é tida como a mais cara na gestão da cadeia de suprimentos e logística [2], sendo realizados diversos estudos e investimentos nesta área como forma de manter os processos sob controle.[1]

A crescente globalização e o fornecimento offshore criaram um aumento considerável na complexidade das linhas de abastecimento e redes de transporte, o que levou as empresas a darem maior atenção à função de transporte e distribuição e suas decisões de design de longo prazo associadas. Para tanto, busca-se considerar o transporte como um processo completo para beneficiar de todos os modos de transporte. As decisões de transporte incluem a seleção do modo de transporte, tamanho da remessa, roteamento de veículos e programação, relacionando-os à localização dos pontos de transbordo, armazéns, clientes e fábricas [1].

Como uma das formas de se analisar um roteamento ótimo para uma frota de veículos, utiliza-se o conhecido Problema de Roteamento de Veículos (VRP), o qual considera que um conjunto de veículos, baseados em um depósito central, devem ser roteados de modo a

minimizar custos e fornecer aos clientes com demandas conhecidas sujeitas a restrições de capacidade do veículo [3]. Uma variante importante deste problema surge quando uma frota de veículos caracterizada por diferentes capacidades e custos está disponível para atividades de distribuição. O problema é conhecido como o VRP de Frota Mista ou como o VRP de Frota Heterogênea (HVRP) [3].

Nesta nova formulação, tem-se o intuito de determinar simultaneamente a composição e roteirização de uma frota heterogênea de veículos para atender a um conjunto pré-especificado de clientes com demandas de entrega conhecidas de um depósito central [1]. O HVRP consiste em projetar um conjunto de rotas de veículos, cada uma começando e terminando no depósito, e de forma que cada cliente seja visitado exatamente uma vez, e que a demanda total de uma rota não exceda a capacidade do veículo que lhe é atribuído, e o custo total é minimizado [1].

Neste projeto visamos estudar tecnicas de Programacao Linear Inteira (PLI) para solucionar de maneira ótima o VRP, considerando características voltadas a entrega de alimentos (como ocorre em entrega de supermercados ou entrega de comida pronta). Dentre as restrições consideradas estão a existência de mais de um tipo de veículo, cada um com suas respectivas capacidades de carregamento e produtos que podem ser transportados apenas por um tipo de veículo.

2 Objetivo

O objetivo deste trabalho é a elaboração de um programa para obter resultados exatos para o problema de roteamento de veículos com frota heterogênea para pequenas e médias instâncias.

3 Problema

O Problema de Roteamento de Veículos (Vehicle Routing Problem, ou VRP) foi originalmente proposto por Dantzig e Ramser em 1959 [4] como uma generalização do problema do caixeiro viajante, e desde então atraiu grande atenção da comunidade cientifica, com a primeira heurística para o problema sendo publicada em 1964 por Clarke e Wright[5]. Além de diversas variedades do problema sendo propostas ao longo dos anos, variando as resrições às quais o problema está sujeito, como por exemplo, ao colocarmos uma restrição sobre a capacidade de transporte dos veículos e uma frota homogênea, temos o que é conhecido como Problema do Roteamento de Veículos Capacitados (Capacitaded Vehicle Routing Problem, ou CVRP), podemos também diversificar a frota de veículos e associar um custo de travessia

a cada veículo, levando ao VRP Heterogêneo com Custos de Roteamento Dependente de Veículo (Heterogenous VRP with Vehicle Dependent Routing Costs, ou HVRPD).

Por se tratar de uma generalização do problema do Caixeiro Viajante, um problema conhecidamente NP-Completo, temos então que o Problema de Roteamento de Veículos é um problema pelo menos NP-Difícl. Sendo assim, temos que, até hoje, nenhum algoritmo eficiente foi encontrado que obtenha resultados exatos para instâncias gigantescas, e, a não ser que se encontre que $P = NP$, não existe tal algoritmo.

Para este texto, estaremos considerando um VRP com uma frota heterogênea composta por dois tipos de veículos, com alguns clientes podendo ser atendidos apenas por um dos tipos de veículos.

3.1 Formulação

Para a resolução do problema, fez-se uma alteração à formulação elaborada por Golden et al. [6], sendo assim, temos as seguintes definições:

Seja $G = (V, E)$ um digrafo com o conjunto $V = \{0, 1, \ldots, n\}$ de vértices e $E = \{ij, \forall i, j \in V\}$ o conjunto de arcos, e o conjunto V' definido como $V' = V \backslash \{0\}$ onde o vértice 0 representa o depósito de onde sairão os veículos.

Seja também $T = \{1, 2, \ldots, k\}$ o conjunto de tipos de veículos disponíveis, e a_k a capacidade de transporte de cada tipo de veículo. Temos então que as constantes c_{ij} é o custo associado à travessia do arco ij.

Temos a variável triplamente indexada x_{ij}^k como uma variável binária que irá representar se um determinado arco ij pertence ou não à solução ótima para um determinado tipo de veículo k. Por fim, temos as variáveis de fluxo r_i representando a demanda total atendida por um veículo em um dado nó i, com a demanda de i inclusa.

Função objetivo:

$$z(F1) = min \sum_{k \in M} \sum_{\substack{i,j \in V \\ i \neq j}} c_{ij} x_{ij}^k \tag{1}$$

$$s.a. \sum_{k \in M} \sum_{i \in V} x_{ij}^k = 1, \quad \forall j \in V' \tag{2}$$

$$\sum_{i \in V} x_{ip}^k - \sum_{j \in V} x_{pj}^k = 0, \quad \forall p \in V', \forall k \in M \tag{3}$$

$$r_0 = 0 \tag{4}$$

$$r_j - r_i \geq (d_j + a_T) \sum_{k=1}^{T} x_{ij}^k - a_T, \quad \forall i \in V, \forall j \in V' \tag{5}$$

$$r_j \leq \sum_{k=1}^{T} \sum_{i=0}^{n} a_k x_{ij}^k \tag{6}$$

$$x_{i,j}^k \in \{0,1\}, \quad \forall i,j \in V, i \neq j, \forall k \in M \tag{7}$$

Nesta formulação, temos que a equação (2) é responsável por garantir que todo nó será visitado apenas uma vez, e a equação (3) garante que todo veículo que entra em um nó cliente também sai do mesmo.

As equações (4), (5) e (6) lidam com as variáveis de fluxo de produto pelas rotas, a primeira fixando o fluxo inicial no depósito como sendo 0, a segunda ditando que a diferença entre a quantidade de produtos que entra em um cliente e a quantidade que sai do mesmo é o valor de sua demanda. A segunda, por sua vez, garante que a demanda dos nós não excede a capacidade de transporte do veículo, além de garantir que, se uma aresta não está na solução, não haverá fluxo de produtos por ela.

Por fim, temos as equações (7) dita a variável de decisão x_{ij}^k como variáveis binárias.

3.2 Técnicas utilizadas

Detalhar a tecnica, motivações para uso, historico, falar sobre formulação utilizada, representação do problema, sobre as classes de instâncias

Neste projeto, visou-se estudar técnicas de Programação Linear Inteira (PLI) para solucionar de maneira ótima o CVRP, considerando características voltadas a entrega de alimentos (como ocorre em entrega de supermercados ou entrega de comida pronta). Dentre as restrições consideradas estão a existência de mais de um tipo de veículo, cada um com suas respectivas capacidades de carregamento e produtos que podem ser transportados apenas por um tipo de veículo.

4 Experimentos Computacionais

4.1 Ambiente de desenvolvimento

Para o desenvolvimento da solução estabelecida, fez-se uso de uma biblioteca fornecida pelo professor Flávio Keidi Miazawa, disponível em [7]. Esta biblioteca oferece um formato de arquivo de texto para grafos e digrafos, além de funções de leitura para eles, e funções, definições e estruturas para lidar com grafos. A biblioteca disponibilizada funciona como middleware para a biblioteca de visualização de grafos chamada lemon, disponível em [8],

a qual visa fornecer estruturas de dados e algoritmos para problemas de otimização, com foco em grafos e redes.
Para o processo de otimização, foi utilizado o otimizador Gurobi, disponibilizado sob uma licensa acadêmica pela empresa Gurobi Optimization, LLC, em [9].
Uma vez desenvolvido o modelo, os experimentos foram executados em um computador portátil Acer Aspire F5-573G, com um processador Intel Core i7-7500U com velocidade de 2.70GHz, com 2 Núcleos e 4 Processadores e 16 GB de memória RAM. O sistema operacional da máquina é Manjaro XFCE 21.2.5, uma distribuição de Linux baseado em ARCH.

4.2 Funcionamento do programa

O programa desenvolvido para a realização destes experimentos, disponível em [], é responsável pela interpretação das instâncias de entrada, pela montagem do problema e do ambiente a ser utilizado pelo otimizador.

O programa toma como entrada toma como entrada arquivos de texto organizados com a terminação *.dig*, lendo-os com a função *ReadDigraph()* definida na biblioteca *mygraphlib*, a qual popula e retorna estruturas de dados definidas pela biblioteca *lemon*, para serem utilizadas pelo resto do programa. É criado então um mapa associando cada nó à sua demanda, e designa valores aleatórios para a demanda de cada nó, de tal forma que a soma de todas as demandas seja menor que a capacidade do maior veículo, para garantir a viabilidade do modelo.

Uma vez feito este pré-processamento, itera-se sobre os tipos de veículo, criando um mapa relacionando cada nó do digrafo a uma variável do otimizador, representando as variáveis de fluxo r_i do modelo, e um mapa relacionando cada uma das arestas a uma variável de decisão x_{ij}^k. Nesta mesma iteração sobre os tipos de veículos, montamos a restrição (3), além de algumas restrições auxiliares, para garantir que os veículos saiam do depósito, e que a quantidade de veículos que retorna ao depósito nunca é maior do que a quantidade de veículos disponível daquele tipo. Por fim, copiamos os mapas gerados para estruturas auxiliares para podermos relacionar diferentes tipos de veículos.

Passamos então pelas restrições que relacionam diferentes tipos de veículos em uma mesma equação, passando pelas restrições (2), (5) e (6), cada uma em se próprio laço, fazendo uso dos mapas relativos a cada um dos tipos de veículos.

As restrições com laços no número de nós e de arcos de entrada, no pior dos casos, tem que o número de arcos de entrada é $|V| - 1$, introduzindo uma complexidade de $O(|V| * (|V| - 1))$. No pior caso, em um digrafo completo, temos que o número de arestas é $|E| = |V| * (|V| - 1)$, e, portanto, a complexidade introduzida no algoritmo pelo pré-processamento é $O(|V|^3)$.

4.3 Resultados

O programa descrito na seção anterior foi executado para 11 diferentes instâncias de variados tamanhos, sendo as 5 primeiras classificadas como de pequeno porte, compostas por no máximo 8 vértices. As 6 subsequentes foram classificadas como de médio porte, tendo em sua composição entre 10 e 15 vértices. Todas as instâncias foram geradas como digrafos euclidianos simétricos, a fim de melhor se assemelhar a problemas no mundo real. O programa, as instâncias e seus resultados estão disponíveis em [10]. Os resultados apresentados a seguir foram compilados executando o programa 5 vezes para cada instância com 12 veículos do tipo 0 e 10 veículos do tipo 1.

Table 1: Resultados obtidos para 11 instâncias, considerando uma frota de 12 veículos do tipo 0 e 10 do tipo 1

Quantidade de Vértices	Quantidade de Arestas	Tempo (ms)	Valor função objetivo	Relação nós tipo 0
5	10	4646	1327	0,400
5	10	1784	1327	0,400
5	10	1854	1327	0,400
5	10	1993	1327	0,400
5	10	2478	1327	0,400
5	10	1848	1327	0,400
7	21	3696	2484	0,286
7	21	3697	2484	0,286
7	21	9327	2484	0,286
7	21	3245	2484	0,286
7	21	10517	2484	0,286
7	21	3681	2484	0,286
8	28	15227	1487	0,375
8	28	5198	1487	0,375
8	28	4406	1487	0,375
8	28	4506	1487	0,375

Continued on next page

Table 1: Resultados obtidos para 11 instâncias, considerando uma frota de 12 veículos do tipo 0 e 10 do tipo 1 (Continued)

Quantidade de Vértices	Quantidade de Arestas	Tempo (ms)	Valor função objetivo	Relação nós tipo 0
8	28	4083	1487	0,375
8	28	4962	1487	0,375
8	28	175070	2153	0,375
8	28	14310	2153	0,375
8	28	38089	2153	0,375
8	28	15563	2153	0,375
8	28	25141	2153	0,375
8	28	14626	2153	0,375
5	10	1576	2108	0,400
5	10	1691	2108	0,400
5	10	1637	2108	0,400
5	10	1649	2108	0,400
5	10	1548	2108	0,400
5	10	1769	2108	0,400
6	15	2760	3295	0,333
6	15	2825	3295	0,333
6	15	2852	3295	0,333
6	15	2746	3295	0,333
6	15	3011	3295	0,333
6	15	6838	3295	0,333
10	45	3148	4110	0,500
10	45	3244	4110	0,500
10	45	6927	4110	0,500
10	45	3274	4110	0,500

Continued on next page

Table 1: Resultados obtidos para 11 instâncias, considerando uma frota de 12 veículos do tipo 0 e 10 do tipo 1 (Continued)

Quantidade de Vértices	Quantidade de Arestas	Tempo (ms)	Valor função objetivo	Relação nós tipo 0
10	45	3172	4110	0,500
10	45	3105	4110	0,500
13	78	35920	3759	0,538
13	78	38703	3759	0,538
13	78	36155	3759	0,538
13	78	39054	3759	0,538
13	78	43312	3759	0,538
13	78	38805	3759	0,538
12	66	11213	3148	0,500
12	66	13690	3148	0,500
12	66	13749	3148	0,500
12	66	12388	3148	0,500
12	66	12578	3148	0,500
12	66	12040	3148	0,500
11	55	15965	3177	0,545
11	55	17729	3177	0,545
11	55	20127	3177	0,545
11	55	17473	3177	0,545
11	55	17000	3177	0,545
11	55	27352	3177	0,545
15	105	103414	2775	0,467
15	105	97721	2775	0,467
15	105	119427	2775	0,467
15	105	93135	2775	0,467

Continued on next page

Table 1: Resultados obtidos para 11 instâncias, considerando uma frota de 12 veículos do tipo 0 e 10 do tipo 1 (Continued)

Quantidade de Vértices	Quantidade de Arestas	Tempo (ms)	Valor função objetivo	Relação nós tipo 0
15	105	92445	2775	0,467
15	105	93267	2775	0,467

Table 2: Tempo médio de processamento obtido para 11 instâncias, considerando uma frota de 12 veículos do tipo 0 e 10 do tipo 1

Quantidade de Vértices	Quantidade de Arestas	Tempo médio de processamento (ms)
5	10	2434
7	21	5694
8	28	6397
8	28	47133
5	10	1645
6	15	3505
10	45	3812
13	78	38658
12	66	12610
11	55	19274
15	105	99902

Analisando a tabela 1, podemos observar uma variação considerável no tempo de execução para uma mesma instância, um motivo para tal é a existência de diversos processos ocupando a CPU, uma vez que o computador utilizado para executar tais experimentos é um computador de uso pessoal e não uma máquina dedicada para experimentação.

Ordenando as instâncias por número de nós, podemos traçar os gráficos de tempo médio para cada grupo, apresentados nas imagens 1 e 2. Podemos observar em ambos casos que os gráficos começam a apresentar o que se assemelha a um crescimento exponencial do tempo

de processamento em função do número de nós.

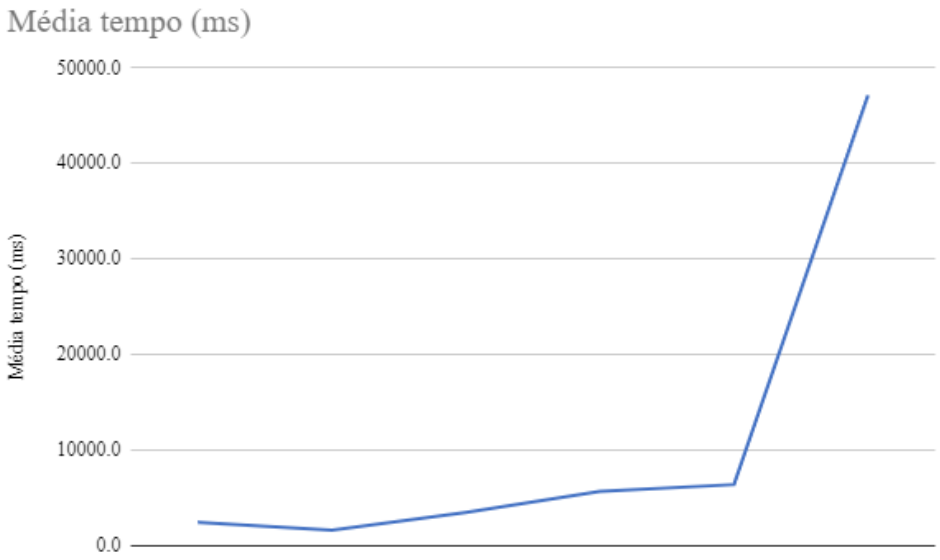

Figura 1: Gráfico de medias de tempo para instâncias pequenas

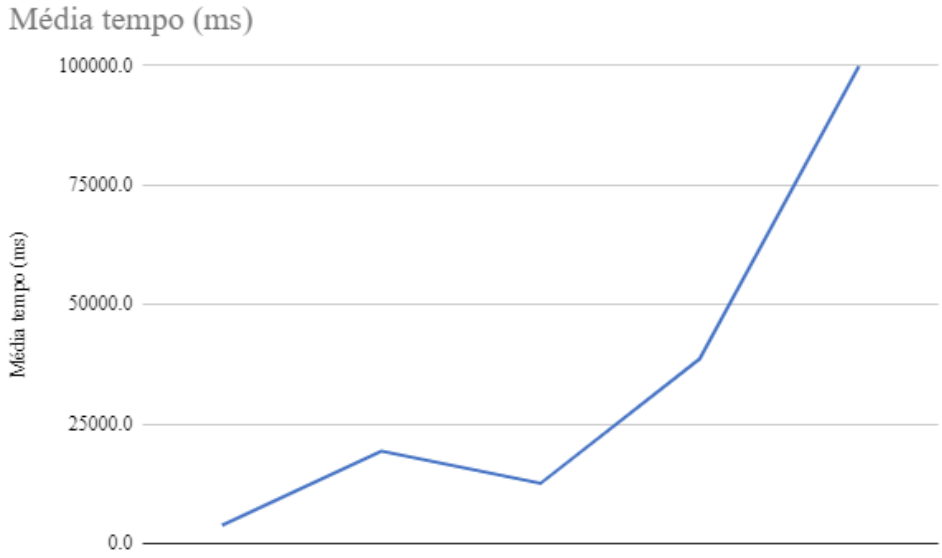

Figura 2: Gráfico de medias de tempo para instâncias médias

Por fim, houve a tentativa de executar o programa para instâncias com 100 nós ou maiores, mas estas tentativas não deram fruto, uma decorrer do elevado tempo de execução, passando de 20 horas com melhorias progressivamente menores para o valor ótimo encon-

trado.

5 Conclusão

Em suma, para o problema de roteamento de veículos para frotas heterogêneas, o uso de um algoritmo exato para instâncias de pequeno e médio porte se mostra viável para uso corriqueiro, embora o mesmo não seja eficiente em relação ao seu tempo de processamento. Entretanto, temos que a execução de algoritmos exatos se mostra muito onerosa para instâncias de tamanhos elevados, sendo necessário o uso de outras técnicas como heurísticas e meta-heurísticas.

6 Trabalhos Futuros

Devido a restrições de tempo do projeto, não foi possível experimentar com heurísticas de variadas complexidades. Futuramente, poderiam ser analisadas desde heurísticas simples como restrições para tempo de processamento, até algumas heurísticas de implementação mais complexa como o relaxamento de variáveis.

Referências

[1] S. Onut, M. R. Kamber and G. Altay, *A heterogeneous fleet vehicle routing model for solving the LPG distribution problem: A case study*. Journal of Physics: Conference Series, **490**, 12043–12047 (March 2014).

[2] M.B. Nidhi and B. Anil, *A cost optimisation strategy for a single warehouse multi-distributor vehicle routing system in stochastic scenario*. International Journal of Logistics Systems and Management, **10** (1), 110–121 (2011).

[3] R. Baldacci, M. Battarra and D. Vigo, *Routing a Heterogeneous Fleet of Vehicles*. Operations Research/ Computer Science Interfaces Series, **43**, 3–27 (2008).

[4] G.B. Dantzig and J.H. Ramser, *The truck dispatching problem*. Management Science, **6** (1), 80–91 (1959).

[5] G. Clarke and J. Wright, *Scheduling of vehicles from a central depot to a number of delivery points*, **12** (4), 568–581 (1964).

[6] B.Golden, A. Assad, L. Levy and Filip Gheysens, *The fleet size and mix vehicle routing problem*. Computers & Operations Research, **11** (1), 305–548 (1984).

[7] Biblioteca Auxiliar para Grafos por Prof. Dr. Flávio Keidi Miyazawa (2022). Prof. Dr. Flávio Keidi Miyazawa. Disponível em: `https://www.ic.unicamp.br/~fkm/courses.html` [Acesso em 11/08/2022]

[8] Lemon Graph Library Documentation (2022). Disponível em: `http://lemon.cs.elte.hu/trac/lemon/wiki/Documentation` [Acesso em: 11/08/2022]

[9] Gurobi Optimizer Reference Manual, version 9.0. (2020). Gurobi Optimization, Inc. Disponível em: `https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf` [Acesso em 11/08/2022]

[10] Repositório no GitHub contendo os programas desenvolvidos ao longo deste trabalho. Disponível em: `https://github.com/coppofe/MC_030_PFG` [Acesso em 11/08/2022]