PROBLEMA DE ROTEIRIZAÇÃO DE ORDENS DE SERVIÇO COM FROTA LIMITADA E HETEROGENEA

Carlos Leonardo Ramos Póvoa
Universidade Estadual do Norte Fluminense Darcy Ribeiro (UENF)
Av. Alberto Lamego, 2000 - Pq. Califórnia, Campos dos Goytacazes - RJ, 28013-602
clrp@uenf.br

André Soares Velasco
Instituto Federal de Educação, Ciência e Tecnologia Fluminense (IFF)
Av. Souza Mota, 350 - Parque Fundão, Campos dos Goytacazes - RJ, 28060-010
asvelasco@iff.edu.br

Frederico Galaxe Paes
Instituto Federal de Educação, Ciência e Tecnologia Fluminense (IFF)
Av. Souza Mota, 350 - Parque Fundão, Campos dos Goytacazes - RJ, 28060-010
fpaes@iff.edu.br

Paola da Conceição da Penha da Silva
Universidade Estadual do Norte Fluminense Darcy Ribeiro (UENF)
Av. Alberto Lamego, 2000 - Pq. Califórnia, Campos dos Goytacazes - RJ, 28013-602
paolacpsilva@gmail.com

RESUMO

Nesse trabalho o problema de roteirização de ordens de serviço (SOVRP) é caracterizado como um caso específico do problema de roteirização de veículos com múltiplos compartimentos (MCVRP) onde a frota é heterogênea, as capacidades dos compartimentos são fixas, o número de compartimentos é predeterminado e os consumidores não podem ser visitados por mais de um veículo. Um modelo de programação inteira é proposto para o problema, bem como uma metaheurística híbrida GRASP e Busca Tabu. O algoritmo é comparado com 20 instâncias MCVRP e obteve melhores resultados em 10 instâncias.

PALAVRAS CHAVE: Roteirização de veículos, GRASP, Busca Tabu.

ABSTRACT

The service orders vehicle routing problem (SOVRP) is characterized as a specific case of the vehicle routing problem with multi-compartment (MCVRP) where the fleet is heterogeneous, the capacities of the compartments are fixed, the number of compartments is predetermined and consumers cannot be visited by more than one vehicle. An integer programming model is proposed for the problem, as well as a hybrid metaheuristic GRASP and Tabu Search. The algorithm is compared with 20 MCVRP instances and obtained better results in 10 instances.

KEYWORDS: Vehicle routing, GRASP, Tabu Search.

1. Introdução

Empresas prestadoras de serviço de telefonia, energia, água e esgoto, dentre outras, necessitam manter equipes externas para atender as solicitações dos seus clientes e realizarem serviços de manutenção em sua rede de equipamentos. Um gerenciamento eficiente dessas equipes permite um aumento na qualidade e redução de custos nos serviços prestados. A definição da distribuição das ordens de serviços entre as equipes é o ponto crítico dessa operação e deve ser feita de forma a minimizar o custo operacional e assegurar o atendimento de todos os consumidores.

O problema de roteirização de ordens de serviço (SOVRP) é caracterizado como uma adaptação do problema de roteirização de veículos com múltiplos compartimentos (MCVRP), onde a frota é heterogênea e os consumidores não podem ser visitados por mais de uma equipe de serviço. Veículos com múltiplos compartimentos são utilizados para o transporte de diferentes produtos que não podem ser misturados devido a alguma característica física, como diferença de temperatura de armazenamento, tipos de combustíveis, diferentes tipos de resíduos, dentre outros. No MCVRP deve ser adicionado a restrição de quais tipos de produtos o compartimento está apto a transportar.

No SOVRP os veículos são as equipes e cada uma possui um ou mais compartimentos que definem sua capacidade de executar diferentes tipos de serviços. Para exemplificar podemos citar equipes de manutenção de prestadoras de serviço de água e esgoto, onde existem equipes especializadas em manutenção da rede de esgoto, outras dedicadas a manutenção do sistema de abastecimento de água e equipes mistas que podem executar determinados serviços de água e esgoto. A capacidade das equipes em realizar os serviços é delimitada pela jornada diária de trabalho.

O objetivo do presente trabalho é modelar o problema de roteirização de ordens de serviço (SOVRP). Uma metaheurística híbrida GRASP e busca tabu é proposta para a solução do problema. Experimentos computacionais com instâncias MCVRP foram realizados para avaliar a eficiência do algoritmo proposto. O trabalho é justificado pela falta de estudo dessa variante do problema de roteirização na literatura, sendo a sua solução importante para reduzir o custo operacional das equipes de manutenção em empresas prestadoras de serviço.

O artigo é organizado da seguinte maneira. Uma revisão da literatura para o problema de roteirização de veículos com múltiplos compartimentos é descrita na seção 2. A seção 3 apresenta a formulação matemática do problema de SOVRP. Nas seções 4 e 5 o algoritmo GRASP com busca tabu para a solução do SOVRP é detalhada. Resultados computacionais com instâncias MCVRP e a comparação com resultados da literatura é apresentada na seção 6. Por fim, a seção 7 apresenta a conclusão e sugestões de pesquisas futuras.

2. Revisão da literatura

O problema de roteirização de veículos com múltiplos compartimentos (MCVRP) aparece pela primeira vez no trabalho de [Brown e Graves 1981] e em [Brown et al. 1987] onde é descrito como uma aplicação para o problema de distribuição de combustíveis.

[Ostermeier et al. 2020] identificaram um total de 84 publicações relacionadas ao MCVRP e montaram uma tipologia das diferentes variações do problema bem como suas aplicações. A tipologia desenvolvida considera os atributos relacionados aos compartimentos e aos requisitos de atendimento do pedido. Em relação aos compartimentos os autores identificaram três tipos de atributos: a) flexibilidade dos tamanhos dos compartimentos: fixos (FixS) ou flexíveis (FlexS). b) Atribuição de tipos de produtos aos compartimentos: fixo (FixA) ou flexível (FlexA). c) Compartilhamento dos compartimentos: compartilhado (SC) e não compartilhado (UC).

Em relação a forma de atender a demanda do consumidor foram identificados dois tipos de atributos: a) número de visitas por consumidor: visita por um único veículo (SV) ou múltiplos veículos (MV). b) Atendimento da demanda: a demanda de um consumidor por um único produto pode ser atendida por mais de um veículo (SD) ou a demanda deve ser atendida por um único

veículo (UD). Em relação as principais aplicações os autores identificaram as seguintes: distribuição de combustível, coleta de materiais recicláveis, distribuição de alimentos e transporte marítimo.

[Chajakis e Guignard 2003] estudaram o problema aplicado a distribuição em lojas de conveniência com veículos com divisórias de tamanhos variáveis e desenvolveram uma solução utilizando relaxação lagrangiana. [Avella et al. 2004] descrevem uma aplicação no setor de distribuição de combustíveis com frota heterogênea. Os autores desenvolveram uma heurística simples para solução do problema e um algoritmo *branch and price*.

O trabalho de [El Fallahi et al. 2008] descreve o problema num contexto da distribuição de ração animal para fazendas, onde os tamanhos dos compartimentos são fixos e cada produto só pode ser alocado em um tipo de compartimento e não podem ser misturados com nenhum outro similar. Foi proposta uma solução para o problema usando algoritmos meméticos e busca tabu. [Tasar et al. 2019] estendem o problema considerando os tempos de carga e descarga e desenvolveram uma heurística de vizinhança variável (VNS).

[Lahyani et al. 2015] apresentam um estudo de caso de distribuição de azeite de oliva, que são categorizados em diferentes compartimentos em função de sua qualidade. Um algoritmo de busca em vizinhança variável (VNS) foi desenvolvido para a solução do problema. [Mendoza et al. 2010] resolveram o problema utilizando algoritmo memético. O trabalho de [Mendoza et al. 2011] resolve o MCVRP com demanda estocástica, utilizando heurísticas construtivas.

[Derigs et al. 2011] desenvolveram um modelo utilizando programação inteira para resolver de forma exata. No mesmo artigo os autores implementaram diversos algoritmos heurísticos e compararam os resultados. [Silvestrin e Ritt 2017] desenvolveram uma busca tabu iterativa (ITS) e obtiveram melhores resultados em quase todas as instâncias de [El Fallahi et al. 2008].

[Henke et al. 2015] apresentam uma versão do problema adaptado a coleta de material reciclado, onde os tamanhos dos compartimentos podem assumir valores discretos pré-definidos. O problema foi resolvido utilizando um algoritmo de busca em vizinhança variável (VNS). Os trabalhos de [Ostermeier e Hübner 2018] apresentam uma extensão do problema, onde a capacidade dos compartimentos é flexível e pode ser ajustada. Os autores desenvolveram um modelo de programação inteira e uma heurística *large neighborhood search* (LNS).

[Chen et al. 2019] aplicaram o MCVRP em uma empresa de distribuição de produtos congelados, considerando a restrição de janelas de tempo e o consumo do combustível. Os autores desenvolveram um modelo de programação matemática e uma heurística *adaptive large neighborhood search* (ALNS) para a solução do problema. Para um maior detalhamento dos trabalhos envolvendo MCVRP, suas diversas variantes e aplicações consultar [Ostermeier et al. 2020].

3. Formulação do Problema

De acordo com a tipologia desenvolvida por [Ostermeier et al. 2020] o problema de roteirização de ordens de serviço (SOVRP) é classificado como um MCVRP onde as capacidades dos compartimentos são fixas (FlexS) e o número de compartimentos é predeterminado para cada veículo. Quanto aos tipos de produtos que podem ser associados aos compartimentos são considerados fixos (FixA), ou seja, os produtos (serviços) somente podem ser atribuídos a compartimentos específicos e os compartimentos são compartilhados (SC) podendo conter a demanda de vários consumidores. Consequentemente, só pode ser alocado um veículo para atendimento ao consumidor se o mesmo tiver o conjunto de compartimentos compatível com a demanda do cliente. As demandas de um consumidor por múltiplos serviços devem ser atendidas por uma única equipe (SV) e a demanda de um consumidor por um único serviço não pode ser atendida por mais de uma equipe (UD).

O SOVRP pode ser formulado da seguinte forma: dado um grafo não direcionado $G = (N, E)$, o conjunto de vértices $N = \{0, 1, \ldots, n\}$ representa os pontos nos quais os serviços serão

executados. Sendo o elemento {0} a base de operações onde as equipes irão partir. Os serviços serão executados por um conjunto de equipes (veículos) $V = \{1, 2, \ldots, v\}$ não idênticos com diferentes custos. Cada equipe está habilitada para executar tipos específicos de serviços, que serão representados pelo seu conjunto de compartimentos $C = \{1, \ldots, m\}$. O conjunto de arestas $E = \{(i, j) \mid i, j \in N \text{ e } i < j\}$ representa as ligações entre os vértices e tem um custo de deslocamento não negativo $c_{ij} = c_{ji}$. Todos esses custos satisfazem a desigualdade triangular. O conjunto de tipos de serviços é definido por $P = \{1, 2, \ldots, p\}$. Em cada consumidor i há uma demanda por um ou mais serviços e o tempo de execução do serviço p no cliente i é definido por t_{ip}.

$$\min \quad \sum_{i\in N}\sum_{j\in N}\sum_{k\in V} x_{ij}^k d_{ij}\alpha_k + \sum_{j\in N}\sum_{k\in V} x_{0j}^k F_k \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in N} x_{ij}^k \leq 1 \qquad \forall i \in N\backslash\{0\}, \forall k \in V, \tag{2}$$

$$\sum_{j\in N}\sum_{k\in V} x_{ij}^k \leq 1 \qquad \forall i \in N\backslash\{0\}, \tag{3}$$

$$\sum_{j\in N} x_{ij}^k = \sum_{j\in N} x_{ji}^k \qquad \forall i \in N\backslash\{0\},\ \forall k \in V, \tag{4}$$

$$h_i^k - h_j^k + n x_{ij}^k \leq n-1 \quad \forall i \neq j \in N\backslash\{0\},\ \forall k \in V, \tag{5}$$

$$y_{jp}^k \leq \sum_{i\in N} x_{ij}^k \qquad \forall j \in N\backslash\{0\},\ \forall k \in V,\ \forall p \in P, \tag{6}$$

$$\sum_{k\in V} y_{jp}^k = 1 \qquad \forall j \in N\backslash\{0\},\ \forall p \in P, \tag{7}$$

$$\sum_{i\in N}\sum_{p\in P} t_{ip} y_{ip}^k \leq T_m \qquad \forall k \in V,\ \forall m \in C, \tag{8}$$

$$\sum_{i\in N} y_{ip}^k \leq |N| \sum_{m\in C} w_{pm}^k \quad \forall p \in P,\ \forall k \in V, \tag{9}$$

$$\sum_{p\in P} w_{pm}^k \leq \sum_{p\in P} \mu_{pm}^k \qquad \forall k \in V,\ \forall m \in C, \tag{10}$$

$$\sum_{i\in N}\sum_{j\in N} c_{ij} x_{ij}^k \leq L^k \qquad \forall k \in V, \tag{11}$$

$$x_{ij}^k \in \{0,1\} \quad \forall i, j \in N,\ i \neq j,\ \forall k \in V, \tag{12}$$

$$y_{ip}^k \in \{0,1\} \quad \forall i \in N\backslash\{0\},\ \forall p \in P,\ \forall k \in V, \tag{13}$$

$$w_{pm}^k \in \{0,1\} \quad \forall p \in P,\ \forall m \in C,\ \forall k \in V, \tag{14}$$

$$h_i^k \in \mathbb{Z}^+ \qquad \forall i \in N\backslash\{0\},\ \forall k \in V \tag{15}$$

As seguintes variáveis de decisão são utilizadas no modelo: x_{ij}^k indica se a equipe k viaja da localização i para j, y_{ip}^k indica se o serviço p requisitado pelo consumidor i vai ser atendido pelo veículo k e w_{pm}^k decide se o serviço p vai ser realizado pela equipe k utilizando o compartimento m. A variável $\mu_{pm}^k \in \{0, 1\}$, apesar de não ser uma variável de decisão, indica se o compartimento m da equipe k pode executar o serviço p.

A função objetivo (1) representa o custo total da rota, onde α_k é o custo variável (depende da distância) e F_k é o custo fixo de cada equipe. As restrições (2) asseguram que as equipes só podem ser utilizadas uma única vez. Todos os serviços demandados por cada consumidor só podem

ser atendidos por uma única equipe (3). A continuidade da rota é garantida pela restrição (4), onde se uma equipe chega no ponto deve partir do mesmo. A eliminação de *subtours* é feita pelas restrições (5). O consumidor precisa ser visitado na rota, se ao menos um dos seus serviços for associado a equipe k (6). A demanda do consumidor por um tipo de serviço deve ser atendida por uma única equipe (7), ou seja, a execução do serviço não pode ser dividida por mais de uma equipe. As restrições (8) limitam o tempo de operação das equipes, onde a soma dos tempos dos diferentes tipos de serviços não pode ultrapassar o tempo máximo de serviço para cada compartimento T_m. A alocação dos serviços para cada compartimento das equipes é garantida pelas restrições (9) e (10). A restrição (11) limita a distância máxima de cada equipe.

O objetivo do modelo é descrever de forma resumida o problema em questão. O SOVRP é *NP-hard*, já que é considerado uma generalização do VRP clássico. Apenas pequenas instâncias podem ser resolvidas de forma exata. Essa é a justificativa do uso de heurísticas para a solução de problemas reais [Silvestrin e Ritt 2017].

4. Algoritmo GRASP

A metaheurística GRASP (*Greedy Randomized Adaptive Search Procedures*) é um método iterativo, proposto por [Feo e Resende 1995]. O algoritmo desenvolvido (Algoritmo 1) é baseado no trabalho de [Chaovalitwongse et al. 2003] e [Kontoravdis e Bard 1995]. A construção de soluções viáveis é iniciada pelo procedimento seleção de sementes (Algoritmo 2), que tem como objetivo inicializar cada nova rota com um único consumidor. Essa seleção é feita de modo que os consumidores mais dispersos sejam alocados primeiro e o número de rotas iniciais seja igual ao número mínimo de equipes (I) necessárias para que todos os serviços sejam executados.

Algoritmo 1 GRASP - SOVRP

```
1:  procedure GRASP(P, V, δ1, δ2, δ3, lcr_i, lcr_f, iter)
2:      P = P − {0}
3:      ρ = SeedSelection(P, V)
4:      f* ← ∞
5:      for i = 1, ..., iter do
6:          x = ConstructGreedyRandomized (ρ, δ1, δ2, δ3, lcr_i, lcr_f,)
7:          ẋ = LocalSearch(x)
8:          if (f(ẋ) < f*) then
9:              S* = ẋ
10:             f* = f(ẋ)
11:         end if
12:     end for
13: return S*
14: end procedure
```

Algoritmo 2 SeedSelection

```
1:  procedure SEEDSELECTION(P, V)
2:      S = {}
3:      s = Find client c ∈ P that is farthest from departure point {0}
4:      while | S |< I do
5:          S = S ∪ s
6:          Create new route r with client s and associate a viable team v for the route
7:          ρ = ρ ∪ r
8:          V = V − v
9:          P = P − s
10:         s = Find costumer c ∈ P that maximize the sum of distances to elements ∈ S
11:     end while
12: return ρ
13: end procedure
```

O Algoritmo 3, apresenta a fase de construção da GRASP e sua execução considera inicialmente achar o menor custo de inserção viável em cada rota $r \in \rho$, para cada consumidor $c \in P$ não associado, utilizando a equipe $v \in V$. Então é calculado uma penalidade P_c, que mede o custo que se deve pagar posteriormente, se o consumidor não for associado na rota de menor custo. Consumidores com altos valores de penalização devem ser associados a uma rota primeiro, enquanto os de menores valores podem esperar. A lista de candidatos restritos (LCR) é construída com os λ maiores custos de penalização, e o cliente a ser roteirizado é escolhido de forma aleatória.

4.1. Busca local

O algoritmo de busca local (Algoritmo 4) utiliza dois tipos de movimentos para melhorar a solução. O primeiro movimento $Shift$ (1,0) transfere um cliente c de uma rota r_1 para uma rota r_2. O segundo $Swap$ (1, 1) permuta um cliente c_1, de uma rota r_1, e um cliente c_2, de uma rota r_2. O movimento que irá atualizar a solução é escolhido de forma aleatória entre os cinco melhores movimentos viáveis de S.

O algoritmo é interrompido quando não é mais possível melhorar a solução S. Caso alguma equipe seja incompatível, tenta-se trocar as equipes das rotas ou utilizar alguma outra equipe disponível.

Algoritmo 3 ConstructGreedyRandomized

```
1: procedure CONSTRUCTGREEDYRANDOMIZED(ρ, δ1, δ2, δ3, lcr_i, lcr_f)
2:     λ = Random(lcr_i, lcr_f)
3:     while P ≠ ∅ do
4:         for c = 1, ..., | P | do
5:             for r = 1, ..., | ρ | do
6:                 for v = 1, ..., | V | do
7:                     if viable then
8:                         Find the insertion cost c between clients i and j:
9:                         C_{i,j,c,r,v} = min{C_{i,j,c} ∀i, j ∈ r}
10:                    end if
11:                end for
12:            end for
13:        end for
14:        for c = 1, ..., | P | do
15:            Find route r* with overall minimum insertion cost:
16:            C_{c,r*,v} = min{C_{c,r,v} ∀r ∈ ρ, ∀v ∈ V}
17:        end for
18:        for c = 1, ..., | P | do
19:            Find Penalty cost P_c = ∑(C_{c,r,v} − C_{c,r*,v}), ∀r ∈ ρ, ∀v ∈ V
20:        end for
21:        RCL = Build a list with the λ largest Penalty costs (P_c)
22:        ĉ = Randomly select one consumer from RCL to be routed
23:        P = P − ĉ
24:    end while
25: return solution (S)
26: end procedure
```

Algoritmo 4 LocalSearch

```
1: procedure LOCALSEARCH(S)
2:     improve = true
3:     MoveList = ∅
4:     while improve = true do
5:         Find all Shift(1, 0) and Swap(1, 1) viable moves ∈ S
6:         MoveList = Build a list with the 5 best moves
7:         Randomly select a move from MoveList and update solution S
8:         if MoveList = ∅ then
9:             improve = false
10:        end if
11:    end while
12: return S
13: end procedure
```

5. Algoritmo busca tabu

A busca tabu (TS) armazena movimentos recentes em uma memória de curto prazo e declara movimentos como sendo tabu para evitar uma solução anteriormente obtida. O desempenho do algoritmo depende da estrutura de vizinhança utilizada, do gerenciamento das soluções inviáveis e da memória de curto prazo adotada [Silvestrin e Ritt 2017]. A busca tabu desenvolvida é inspirada nos trabalhos de [Cordeau e Maischberger 2012], [Silvestrin e Ritt 2017] e [El Fallahi et al. 2008].

A solução inicial é proveniente da GRASP. A cada iteração é escolhido o melhor movimento não tabu que gere uma solução vizinha *S'*. Para garantir uma ampla exploração do espaço de busca, são usados mecanismos de relaxação das restrições (permite soluções inviáveis) e de diversificação. O algoritmo encerra quando o número máximo de iterações sem melhorias ($noimpMax$) ou o número máximo de iterações ($iterMax$) é atingido (Algoritmo 5).

5.1. Função objetivo

Uma vantagem da busca tabu é permitir a possibilidade de soluções inviáveis durante o processo de busca local. Sendo R o conjunto de rotas da solução S, a função objetivo deve ser penalizada sempre que uma restrição é violada. Baseada no trabalho de [Cordeau e Maischberger 2012] a função utilizada para avaliar os movimentos é calculada da seguinte forma: $f(s) = \sum_{r \in R} c(r) + \alpha t(r) + \beta d(r)$, onde $c(r)$ é o custo total da rota r, $t(r)$ é a soma das violações de tempo da equipe (compartimentos) para cada tipo de serviço e $d(r)$ é a violação do comprimento máximo da rota. Os multiplicadores α e β são pesos autoajustáveis. Esses multiplicadores são inicialmente iguais a 1 e seus valores são aumentados por um fator $1 + \delta$ quando as correspondentes restrições são violadas, caso contrário, os multiplicadores são decrescidos pelo fator 1- δ. No início do procedimento, o parâmetro δ é escolhido de maneira aleatória no intervalo [0, 1].

5.2. Vizinhança de busca e lista tabu

O movimento *Shift* (1,0) é o único utilizado para montar a vizinhança de busca e é definido pela tupla (r, r', i), onde r é a rota de origem, r' é a rota de destino e i o cliente. Apenas as rotas mais próximas do cliente i são consideradas na busca. A distância entre o cliente e a rota é calculada usando o centro de gravidade da rota ($\sum_{i \in n} x_i / n$, $\sum_{i \in n} y_i / n$). Uma lista de rotas mais próximas, ordenadas pela proximidade, é mantida para cada cliente i e o seguinte critério é usado para preencher a lista: apenas 30% das rotas mais próximas serão usadas na busca. Se o número de

rotas for inferior a 5, serão consideradas as 5 mais próximas. O custo $f(s)$ de todos os movimentos é calculado e o melhor movimento não tabu é escolhido para atualizar a solução.

Algoritmo 5 Tabu Search

```
1: procedure TABUSEARCH(S, iterMax, noimpMax)
2:     α = 1, β = 1, noimp = 0, inv = 0
3:     S* = S, f* = f(S)
4:     Choose δ ∈ [0,1], ζ ∈ [0,1], τ ∈ [1, √nm] randomly
5:     repeat
6:         iter = iter + 1
7:         S = bestMove(S, α, β, ζ)
8:         updatePenalties(α, β, δ)
9:         updateTabuList(τ)
10:        if S is viable and f(S) < f* then
11:            S* = S, f* = f(S)
12:            noimp = 0
13:        else
14:            noimp = noimp + 1
15:        end if
16:        if S is viable then
17:            inv = 0
18:        else
19:            inv = inv + 1
20:            if inv = 5 then
21:                S = perturb(S*)
22:                inv = 0
23:                α = 1, β = 1
24:                Choose δ ∈ [0,1], ζ ∈ [0,1], τ ∈ [1, √nm] randomly
25:            end if
26:        end if
27:    until (iter = iterMax) or (noimp = noimpMax)
28: return S*
29: end procedure
```

Um movimento (r, r', i) é considerado tabu por τ interações. A tabu ternura (τ) é escolhida aleatoriamente em $[1, \sqrt{mn}]$ no início da TS, onde m é o número de rotas na solução inicial e n o número de clientes [Cordeau e Maischberger 2012]. Um único critério de aspiração é utilizado: se um movimento melhora a melhor solução geral, ele é realizado, mesmo que seja tabu (aspiração por objetivo). Uma estrutura de armazenamento da lista tabu foi criada para cada cliente, com atualização a cada iteração.

5.3. Diversificação

Os procedimentos de diversificação guiam a solução por regiões menos exploradas. Existem várias maneiras de diversificar a busca em um procedimento TS, sendo a forma mais simples e direta penalizar soluções inviáveis conforme explicado na seção 5.1. O segundo mecanismo de diversificação aplicado é penalizar os movimentos frequentes que são adicionados à solução. Para um movimento $M = (r, r', i)$ a função objetivo modificada passa a ser: $f'(s) = f(s)(1 + \zeta\, freq_M/iter)$, onde $freq_M$ é o número de vezes que o movimento M foi realizada durante a busca. O parâmetro ζ é escolhido aleatoriamente em [0, 1] no início do procedimento.

A última estratégia de diversificação aplicada consiste em um procedimento de reinicialização. A ideia é que a BT reinicie a exploração do espaço de solução a partir de uma solução S^* sempre que ela ficar presa em um mínimo local. Depois de cinco soluções não viáveis consecutivas, S^* é perturbado e os parâmetros α, β, δ, ζ e τ são reiniciados. Para perturbar uma solução, um cliente é escolhido aleatoriamente e então é efetuado a permutação com um cliente pertencente a uma outra rota: $Swap(1, 1)$. Somente trocas viáveis são aceitas e o número de clientes permutados é escolhido aleatoriamente no intervalo [1, 4].

6. Experimentos computacionais

As instâncias utilizadas para avaliar a performance do algoritmo GRASP/TS foram as propostas por [El Fallahi et al. 2008]. Os resultados foram comparados com os trabalhos de

[Silvestrin e Ritt 2017], [Mendoza et al. 2010] e [El Fallahi et al. 2008]. O SOVRP não permite que um consumidor seja atendido por mais de uma equipe por questões operacionais, sendo assim, os resultados obtidos pelos trabalhos de [Muyldermans e Pang 2010] e [Derigs et al. 2011] não foram utilizados. Esses autores não fizeram testes não permitindo a divisão de entregas e ainda não implementaram a restrição de tempo de rota. Instâncias aleatórias foram geradas para validar o modelo matemático proposto para o SOVRP. Resultados utilizando somente a heurística GRASP para o MCVRP estão em [Póvoa et al. 2020].

6.1. Metodologia

[El Falhahi et al. 2008] transformam 20 instâncias conhecidas para o VRP dos trabalhos de [Christofides et al. 1979] e [Christofides e Eilon 1969] em dois conjuntos de problemas MCVRP com dois compartimentos. As instâncias variam de 50 a 484 consumidores. Em 16 dos 30 problemas os consumidores são distribuídos de forma aleatória no espaço Euclidiano, enquanto os 4 restantes têm seu arranjo em *clusters* [Mendoza et al. 2010]. Nas instâncias 6-10, 13 e 14 os consumidores tem tempo de serviço e existe a restrição de distância da rota.

Os dois conjuntos foram gerados utilizando dois tipos de estratégia de divisão. A primeira estratégia S1 divide a capacidade do veículo e a demanda de cada consumidor em duas partes iguais. Nessa estratégia a solução do VRP corresponde a solução do MCVRP. Se a divisão do conjunto de demandas de cada consumidor não for permitida, as soluções ótimas são as mesmas. O objetivo dessa estratégia é verificar se o resultado fica muito longe das melhores soluções para o VRP [El Fallahi et al. 2008].

A segunda estratégia S2 é obtida dividindo randomicamente a demanda de cada consumidor. A demanda para o primeiro compartimento é calculada utilizando a equação $q_{i1} = q_i/k$, onde q_i é a demanda do VRP original e k é um número aleatório inteiro entre 3 e 5. A demanda para o segundo compartimento q_{i2} é a diferença entre a demanda total (q_i) e a utilizada no primeiro compartimento. As capacidades de cada compartimento Q_1 e Q_2 são determinadas da seguinte forma: $Q_1 = (Q \times \bar{Q}_1)/(\bar{Q}_1 + \bar{Q}_2)$ e $Q_2 = (Q \times \bar{Q}_2)/(\bar{Q}_1 + \bar{Q}_2)$, onde Q é capacidade do veículo para o VRP original, $\bar{Q}_1$ é a demanda média para o primeiro compartimento e $\bar{Q}_2$ para o segundo compartimento. O conjunto de instancias da estratégia S2 foram obtidos com os autores, já que o método de geração aleatório utilizado não permite reproduzir com fidelidade os valores originais obtidos pelo trabalho de [El Falhahi et al. 2008].

Os algoritmos foram codificados em *object pascal* e testados em um PC Intel Core i5-2400, 3.1 GHz e somente um núcleo de processamento foi utilizado. A Tabela 1 sumariza o ambiente computacional utilizado em cada trabalho e o fator de redução usado para comparar os tempos de processamento. O trabalho de [Mendoza et al. 2010] utilizou processamento paralelo em um *grid* com 35 computadores, impossibilitando a conversão de tempo. Todos os fatores de conversão foram obtidos no *benchmark* do *PassMark* (https://www.passmark.com).

Tabela 1 – Ambientes computacionais

Referência	Ambiente	Fator
El Fallahi et al. (2008)	Pentium 4, 2.4 GHz	3
Mendoza et al. (2011)	Computer grid 35 x (1.85 Ghz Intel Core 2 Duo)	-
Silvestrin and Ritt (2017)	AMD FX 8150, 3.6 GHz	1
This paper	Intel Core i5-2400, 3.1 GHz	1

Após alguns experimentos iniciais, os parâmetros do algoritmo GRASP/TS foram definidos. O número de iterações ($iter$) de GRASP foi definido como 100 e os valores de $\delta_1 = 0{,}1$, $\delta_2 = 0{,}8$ e $\delta_3 = 0{,}1$. O tamanho da lista de candidatos restritos (RCL) é escolhido aleatoriamente para cada iteração no intervalo [2, 10]. Os parâmetros de parada da busca tabu, $iterMax$ e $noimpMax$ foram configurados em 1500 e 1000. Todos os experimentos foram replicados 10 vezes, uma vez que o algoritmo desenvolvido é estocástico.

O modelo proposto para o SOVRP foi testado utilizando um conjunto de 14 instâncias geradas aleatoriamente. Os testes foram realizados no ambiente de processamento *IBM Watson Studio* com um computador de 2 núcleos e 8Gb de memória. A codificação do modelo foi feita em *Python* e resolvido utilizando a API do *software* de otimização *IBM ILOG CPLEX Optimization Studio 20.1* com o tempo de processamento limitado em 30 minutos.

6.2. Resultados

A Tabela 2 apresenta os resultados do modelo de programação inteira proposto para o SOVRP. Nas instâncias de até 12 clientes a solução ótima foi encontrada com rapidez, com exceção da Inst_10 que foi resolvida em aproximadamente 28 minutos. Nas instâncias de 15 clientes houve aumento do tempo de processamento, inviabilizando os testes com instâncias maiores. Todas as instâncias foram testadas com o algoritmo GRASP/TS e todos os resultados se igualaram com o valor ótimo do modelo proposto, com tempo de processamento inferior a 1 segundo em todas as instâncias.

Tabela 2 – Testes do Modelo proposto SOVRP

Instância	Tipos de Serviços	Tipos de Compartimentos	Equipes	Clientes	Função Objetivo	t(s)
Inst_1	2	2	2	5	*872,28	< 1
Inst_2	2	2	2	5	*300,84	< 1
Inst_3	3	3	2	5	*769,44	< 1
Inst_4	3	3	2	5	*345,76	< 1
Inst_5	2	2	3	10	*1041,79	4
Inst_6	2	2	3	10	*518,31	13
Inst_7	3	3	3	10	*483,13	2
Inst_8	3	3	3	10	*807,70	4
Inst_9	2	2	3	12	*679,92	23
Inst_10	3	3	3	12	*1026,91	1670
Inst_11	2	2	3	15	*585,06	133
Inst_12	3	3	3	15	*1019,05	1172
Inst_13	2	2	3	15	*629,60	164
Inst_14	3	3	3	15	*1450,40	894

Os resultados do experimento utilizando a estratégia de divisão S1 estão na Tabela 3. As colunas *File*, *n* e *BKV* indicam respectivamente o nome da instância testada, o total de clientes e o melhor resultado encontrado na literatura para o VRP. A qualidade da solução de cada instância é medida pelo desvio relativo Gap%. O tempo de processamento, em segundos, normalizado de cada algoritmo está na coluna t(s). Os tempos de processamento do trabalho de [El Fallahi et al. 2008] foram divididos por três.

O algoritmo proposto obteve uma redução de 1.8% no valor médio das instâncias em relação ao algoritmo memético (MA) proposto por [El. Fallahi et al. 2008], com melhora de resultados em treze instâncias. Em relação a busca tabu (TS), também proposta por [El. Fallahi et al. 2008], foi obtido uma redução de 0.5% com melhora de resultados em dez instâncias. Comparado com o algoritmo memético (MA/SCS) proposto por [Mendoza et al. 2010] foi obtida uma redução de 0.9% na média e melhora de resultados em 13 instâncias. Na comparação com a busca tabu iterativa (ITS) proposta por [Silvestrin e Ritt 2017] o algoritmo proposto não obteve melhora do valor médio ficando 1.2% acima da média dos resultados. Mesmo tendo uma piora em relação a ITS o algoritmo obteve redução de valores em 4 instâncias. Em relação aos melhores valores encontrados na literatura (BKV) o algoritmo ficou 2.8% acima da média desses resultados, igualando a valores de duas instâncias. Vários desses resultados foram obtidos por métodos exatos e apresentam valores ótimos [Uchoa *et al.*, 2017]. O algoritmo proposto não se distanciou das

melhores soluções do PRV e apesar de não ter sido elaborado para resolver o problema clássico de roteirização de veículos mostrou-se com desempenho competitivo.

Tabela 3 – Resultados **S1**; sem divisão de demanda

File	n	BKV	Gap%	El Fallahi et al. (2008)						Mendoza et al. (2010)			Silvestrin and Ritt (2017)			SOVRP	
				MA	t(s)	Gap%	TS	t(s)	Gap%	MA/SCS	t(s)	Gap%	ITS (10^4)	t(s)	Gap%	GRASP /TS	t(s)
vrpnc1	50	524.6	0.0	524.6	6	0.0	524.6	7	0.0	524.6	10	0.0	524.6	6	0.0	**524.6**	4
vrpnc2	75	835.3	1.2	855.8	8	-1.3	850	19	-0.6	857.3	16	-1.4	839.1	8	0.7	844.9	5
vrpnc3	100	826.1	1.1	876.8	7	-4.8	831.3	17	0.4	841.6	24	-0.8	830.4	10	0.5	834.8	14
vrpnc4	150	1028.4	2.2	1089.6	31	-3.6	1061.1	95	-1.0	1045.9	47	0.5	1052.5	14	-0.2	1050.8	44
vrpnc5	199	1291.4	4.9	1389.6	39	-2.5	1348.3	135	0.5	1381.9	86	-1.9	1339.0	23	1.2	1355.0	77
vrpnc6	50	555.4	0.0	571.4	5	-2.8	575.9	4	-3.6	556.7	8	-0.2	555.4	4	0.0	**555.4**	5
vrpnc7	75	909.7	2.0	933	13	-0.5	970.8	7	-4.4	924.3	16	0.4	912.8	9	1.7	928.2	10
vrpnc8	100	865.9	1.2	969.2	6	-9.6	888.6	20	-1.4	877.6	24	-0.2	874.3	9	0.2	876.2	16
vrpnc9	150	1162.5	2.7	1230.9	33	-3.0	1232.1	33	-3.1	1223.1	47	-2.3	1180.6	15	1.2	1194.4	40
vrpnc10	199	1395.8	5.3	1520.1	47	-3.3	1538.6	45	-4.5	1497.5	67	-1.9	1426.8	21	3.0	1469.6	69
vrpnc11	120	1042.1	0.9	1046.1	16	0.5	1043.3	48	0.8	1046.9	38	0.4	1044.7	13	0.6	1051.3	54
vrpnc12	100	819.6	2.9	820.6	6	2.8	819.5	21	2.9	821	24	2.8	819.6	6	2.9	843.7	13
vrpnc13	120	1541.1	1.9	1547.4	25	1.5	1582.2	15	-0.7	1586.9	44	-1.0	1575.5	11	-0.3	1571.0	23
vrpnc14	100	866.4	1.7	866.9	8	1.6	868.6	15	1.4	868.1	24	1.5	866.7	12	1.6	880.9	18
e072-04f	72	237	4.9	241.9	4	2.7	244.5	9	1.7	242	19	2.7	242.0	5	2.7	248.5	21
e076-08s	76	735	1.2	761.9	5	-2.4	748.5	8	-0.6	757.1	16	-1.8	744.1	8	-0.0	743.8	5
e076-07u	76	682	1.6	703.5	5	-1.5	692.2	8	0.1	698.8	16	-0.9	691.7	9	0.2	692.8	10
e135-07f	135	1162.9	3.4	1193.2	16	0.8	1174.4	32	2.4	1195.3	59	0.6	1173.1	24	2.5	1202.5	97
e241-22k	241	707.8	6.0	748.6	168	0.2	741.1	129	1.2	763.5	110	-1.8	727.4	35	3.1	750.0	81
e484-19k	484	1101.5	7.2	1250.1	548	-5.5	1149.5	1197	2.8	1250.9	1011	-5.6	1159.3	62	1.9	1181.1	850
Avg.		914.5	2.8	957.06	50	-1.8	944.3	93	-0.5	948.1	85	-0.9	929.0	15	1.2	940.0	73

A estratégia de divisão S2 utiliza um número aleatório inteiro *k* para a divisão da demanda entre os dois compartimentos. Essa característica não permite que as instâncias sejam geradas sempre da mesma forma, sendo assim, as instâncias utilizadas no trabalho de [El Fallahi et al. 2008] foram obtidas com os autores. O trabalho de [Silvestrin e Ritt 2017] gerou novos valores de divisão de demanda para os produtos, impossibilitando a comparação com os trabalhos de [Medonza et al. 2010] e [El Fallahi et al. 2008]. A Tabela 4 apresenta os resultados da estratégia de divisão S2.

Tabela 4 – Resultados **S2**; sem divisão de demanda

File	n	BKV	Gap%	El Fallahi et al. (2008)						Mendoza et al. (2010)			SOVRP	
				MA	t(s)	Gap%	TS	t(s)	Gap%	MA/SCS	t(s)	Gap%	GRASP/TS	t(s)
vrpnc1	50	550.2	-0.6	558.8	6	-2.1	556.1	7	-1.6	550.2	10	-0.6	**547.0**	4
vrpnc2	75	863.6	1.5	888.6	8	-1.4	863.6	19	1.5	884.3	17	-0.9	876.5	5
vrpnc3	100	837.6	0.2	878.4	7	-4.4	837.6	17	0.2	852.7	25	-1.6	839.3	14
vrpnc4	150	1070.7	0.7	1089.1	31	-1.0	1070.7	95	0.7	1080	51	-0.2	1078.2	44
vrpnc5	199	1361.4	1.8	1408.5	39	-1.6	1361.4	135	1.8	1403.5	91	-1.3	1385.7	77
vrpnc6	50	562.7	-0.9	569.4	5	-2.1	563.4	4	-1.0	562.7	9	-0.9	**557.5**	5
vrpnc7	75	946.3	-1.2	955.1	13	-2.2	949	7	-1.5	946.3	16	-1.2	**934.5**	10
vrpnc8	100	874.7	2.9	958.9	6	-6.1	916.2	20	-1.8	874.7	24	2.9	900.2	16
vrpnc9	150	1240.5	-3.4	1262.7	33	-5.1	1290.8	33	-7.2	1240.5	51	-3.4	**1198.1**	40
vrpnc10	199	1490.2	-0.6	1509.1	47	-1.8	1490.2	45	-0.6	1515.6	78	-2.2	**1481.9**	69
vrpnc11	120	1122.9	6.6	1122.9	16	6.6	1201.6	48	-0.4	1162.4	48	3.0	1196.7	54
vrpnc12	100	916.6	-1.5	926.5	6	-2.6	934.1	21	-3.3	916.6	24	-1.5	**902.9**	13
vrpnc13	120	1542.4	4.2	1542.4	25	4.2	1582.3	15	1.5	1607.4	37	-0.0	1606.7	23
vrpnc14	100	937.6	-0.1	966.5	8	-3.1	1141.6	15	-17.9	937.6	24	-0.1	**936.9**	18
e072-04f	72	262.3	0.6	263.6	4	0.1	262.3	9	0.6	262.5	22	0.5	263.9	21
e076-08s	76	772.2	-1.6	793.5	5	-4.3	772.2	8	-1.6	778.2	15	-2.4	**759.5**	5
e076-07u	76	697.8	-0.3	702.2	5	-0.9	697.8	8	-0.3	704.1	16	-1.2	**696.0**	10
e135-07f	135	1233.2	6.3	1233.2	16	6.3	1235.2	32	6.1	1293.8	49	1.3	1310.7	97
e241-22k	241	787.8	-0.5	796.7	168	-1.6	787.8	129	-0.5	810.7	117	-3.3	**783.9**	81
e484-19k	484	1177.3	0.3	1240.9	548	-4.8	1177.3	1197	0.3	1269	782	-7.0	1180.8	850
Avg.		962.4	1.0	983.35	50	-1.2	984.6	93	-1.3	982.6	75	-1.1	**971.8**	73

O procedimento GRASP/TS obteve uma redução média acima de 1% nos resultados em relação aos algoritmos propostos por [El Fallahi et al. 2008] e [Medonza et al. 2010]. Em relação aos algoritmos de [El. Fallahi et al. 2008] houve uma melhora em 16 instâncias quando comparado com o algoritmo memético (MA) e de 12 instâncias quando comparado a busca tabu (TS).

Comparado com o algoritmo memético (MA/SCS) proposto por [Mendoza et al. 2010] foi obtida uma redução em 16 instâncias. A comparação com os melhores valores (BKV) obtidos por cada algoritmo foi obtido uma melhora em 10 instâncias e a média dos valores ficou apenas 1% acima dos melhores resultados.

O tempo de processamento foi em torno de 20% menor quando comparado a busca tabu (TS) proposto por [El. Falahhi et al. 2008] e 50% maior que o algoritmo memético (MA). Em relação ao algoritmo memético (MA/SCS) proposto por [Mendoza et al. 2010] o tempo médio de solução foi praticamente o mesmo, apesar da utilização de processamento paralelo em um grid com 35 computadores.

7. Conclusão

Esse artigo apresenta o SOVRP, uma variante do MCVRP, e apesar de sua aplicabilidade em diversas empresas, não tem recebido a devida atenção na literatura. O problema consiste em estabelecer o conjunto otimizado das rotas que as equipes de serviço irão fazer para atender a demanda dos consumidores respeitando as restrições operacionais. Uma heurística híbrida GRASP/TS foi proposta para resolver o problema. O resultado da GRASP foi utilizado como solução inicial da TS. O mecanismo elaborado para reduzir a vizinhança de busca na TS apresentou redução do tempo de processamento e não interferiu na qualidade da solução.

As principais contribuições do trabalho incluem (1) a apresentação de uma nova variante do MCVRP inspirada em um problema real; (2) nova modelagem para o SOVRP; (3) o desenvolvimento de uma heurística híbrida (GRASP/TS) para a solução do problema; e (4) experimentos computacionais para testar o desempenho do algoritmo com instâncias MCVRP.

O algoritmo desenvolvido apresentou bom desempenho, superando os resultados disponíveis da literatura em diversas instâncias. O SOVRP requer mais estudos de suas variantes, como janela de tempo de atendimento das ordens de serviço, dentre outros.

8. Referências bibliográficas

Avella, P., Boccia, M., Sforza, A. (2004) Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research* 152, 170 – 179.

Brown, G. G., Ellis, C. J., Graves, G. W., Ronen, D. (1987) Real-time, wide area dispatch of mobil tank trucks. *Interfaces* 17, 107–120.

Brown, G., Graves, G. W. (1981) Real-time dispatch of petroleum tank trucks. *Management Science* 27, 19 – 32.

Chajakis, E., Guignard, M. (2003) Deliveries in vehicles with multiple compartments. *Journal of Global Optimization* 26, 43 – 78.

Chaovalitwongse, W., Kim, D., Pardalos, P. M. (2003) GRASP with a New Local Search Scheme for Vehicle Routing Problems with Time Windows. *Journal of Combinatorial Optimization* 7, 179–207.

Chen, L., Liu, Y., Langevin, A. (2019) A multi-compartment vehicle routing problem in cold-chain distribution. *Computers & Operations Research* 111, 58–66.

Christofides, N., Eilon, S. (1969) An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly* 20, 309–318.

Christofides, N., Mingozzi, A., Toth, P. (1979) *Combinatorial optimization*. The vehicle routing problem. John Wiley, Chichester 1, 315–338.

Cordeau, J.F., Maischberger, M. (2012) A parallel iterated tabu search heuristic for vehicle routing problems. *Computers and Operations Research* 39, 2033–2050.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., Vogel, U. (2011) Vehicle routing with compartments: Applications, modelling and heuristics. *OR Spectrum* 33, 885–914.

Fallahi, A.El., Prins, C., Wolfler Calvo, R. (2008) A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers and Operations Research* 35, 1725–1741.

Feo, T., Resende, M. (1995) Greedy randomized adaptive search procedures. *Journal of Global Optimization* 6, 109–13.

Henke, T., Speranza, M.G., Wäscher, G. (2015) The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research* 246, 730–743.

Kontoravdis, G., Bard, J.F. (1995) A GRASP for the Vehicle Routing Problem with Time Windows. *Informs Journal on Computing* 7, 10–23.

Lahyani, R., Laporte, G., Coelho, L.C., Khemakhem, M. (2015) A Multi-Compartment Vehicle Routing Problem Arising in the Collection of Olive Oil in Tunisia. *Omega* 51, 1–10.

Mendoza, J.E., Castanier, B., Guéret, C., Medaglia, A.L., Velasco, N. (2010) A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers and Operations Research* 37, 1886–1898.

Mendoza, J.E., Castanier, B., Guéret, C., Medaglia, A.L., Velasco, N. (2011) Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transportation Science* 45, 346–363.

Muyldermans, L., Pang, G. (2010) On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Journal of Operational Research* 206, 93–103.

Ostermeier, M., Henke, T., Hübner, A., Wäscher, G. (2020) Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions. *European Journal of Operational Research*.

Ostermeier, M., Hübner, A. (2018) Vehicle selection for a multi-compartment vehicle routing problem. *European Journal of Operational Research* 269, 682–694.

Póvoa, C.L.R; A. S. Velasco; F. G. Paes. Geo-Rota - Sistema web de suporte a decisão para o problema de roteirização de veículos com multi-compartimentos. *Gestão da Produção, Operações e Sistemas,* v. 15, n. 3, p. 133–154, 2020.

Silvestrin, P.V., Ritt, M. (2017) An Iterated Tabu Search for the Multi-compartment Vehicle Routing Problem. *Computers & Operations Research* 81, 192–202.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A. (2017) New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research* 257, 845–858.

Tasar, B., Türsel Eliiyi, D., Kandiller, L. (2019) Vehicle routing with compartments under product incompatibility constraints. *Traffic Management* 31, 25–36.