UMA ABORDAGEM META-HEURÍSTICA PARA O PROBLEMA DE MARKETING DIRETO CONSIDERANDO O CANIBALISMO ENTRE PRODUTOS

Felipe Martins Müller
Universidade Federal de Santa Maria - UFSM
Av. Roraima, 1000 - 7 - Camobi, Santa Maria - RS, 97105-900
felipe@inf.ufsm.br

Veridiane Bárbara de Souza
Universidade Federal de Santa Maria - UFSM
Av. Roraima, 1000 - 7 - Camobi, Santa Maria - RS, 97105-900
veridianes@hotmail.com

Iaê Santos Bonilha
Universidade Federal de Santa Maria - UFSM
Av. Roraima, 1000 - 7 - Camobi, Santa Maria - RS, 97105-900
iaesb@hotmail.com

José Cardoso Sobrinho
Universidade Federal de Santa Maria - UFSM
Av. Roraima, 1000 - 7 - Camobi, Santa Maria - RS, 97105-900
jcardosos@smail.ufsm.br

Resumo: Este artigo apresenta um estudo sobre a aplicação de metaheurísticas em campanhas de marketing direto. Sendo assim, o objetivo é desenvolver e avaliar um algoritmo heurístico para resolução do problema de marketing direto considerando o canibalismo entre produtos, maximizando o feedback de compras dos clientes. O problema é classificado como NP-difícil, sendo relevante a utilização de métodos heurísticos para sua resolução. Além disso, a inclusão do efeito de canibalismo entre produtos ao problema é inédita na literatura. Foram propostos três algoritmos heurísticos, um Busca Tabu, um Tabu/GRASP e ainda um terceiro, Tabu/GRASP com efeito de canibalismo. As instâncias utilizadas foram as disponibilizadas na literatura e foram tratadas estatisticamente para simulação do efeito de canibalismo. O algoritmo de Tabu/GRASP obteve melhores resultados na maior parte das instâncias, porém o efeito do canibalismo compromete a qualidade das soluções. Os resultados demonstram a robustez e qualidade dos algoritmos propostos.
Palavras-chave: Marketing Direto, Heurística, GRASP, Busca Tabu.
Tópicos: MH – Metaheurísticas, OA – Outras Aplicações em PO

Abstract: This paper presents the application of metaheuristics in direct marketing campaigns. The aim is to develop and evaluate a heuristic algorithm to solve the problem of direct marketing considering the cannibalism between products, maximizing the feedback of customers' purchases. The problem is classified as NP-difficult, being relevant the use of heuristic methods for its resolution. In addition, the inclusion of the effect of cannibalism between products on the problem is unprecedented in the literature. Three heuristic algorithms were proposed: Tabu Search, Tabu / GRASP and Tabu / GRASP considering cannibalism. The instances used were those available in the literature and were treated statistically to simulate the effect of cannibalism. The algorithm of Tabu / GRASP obtained better results in most instances, whereas the cannibalism worses the solution quality. However, the results show the robustness and quality of algorithms proposed.

Keywords: Direct Marketing, heuristic, GRASP, Tabu Search.
Paper topics: MH – Metaheuristics, OA – Other Apllications of OR

https://proceedings.science/p/85305?lang=pt-br

1. Introdução

Os avanços tecnológicos das últimas décadas proporcionaram um considerável aumento na quantidade de dados disponíveis e dependência de técnicas de mineração de dados, o que tem levado à criação de inúmeros novos modelos de negócios e oportunidades. A área de marketing direto não é exceção e têm utilizado técnicas inovadoras como forma de obter vantagem competitiva, principalmente para identificar os clientes mais rentáveis ou aqueles com maior propensão a responder positivamente a uma campanha específica [Seret et al. 2012].

As campanhas de promoção são ferramentas fundamentais de marketing direto para melhorar o lucro econômico de uma empresa, seja adquirindo novos clientes ou gerando receita adicional de clientes existentes [Kotler e Armstrong, 2011]. Nesse sentido, a tendência atual para as campanhas de promoção com orientação para o cliente é baseada na crescente disponibilidade de dados do cliente e a evolução nas técnicas de mineração de dados [Abedi, 2017], além da prioridade estratégica de eficiência na alocação de recursos financeiros, principalmente, de modo que favoreça a qualidade da correspondência entre o cliente e a mensagem de marketing.

A atual pesquisa buscou explorar as possibilidades de meta-heurísticas GRASP e Busca Tabu para resolver eficientemente um problema de campanha de promoção, identificando quais os produtos devem ser ofertados e para quais clientes, maximizando o lucro e levando em consideração várias restrições de negócios. Desse modo, definiu-se como objetivo geral desenvolver e avaliar um algoritmo heurístico para resolução do problema de marketing direto, levando em consideração o canibalismo entre produtos, a fim de promover a maximização do feedback de compras dos clientes.

Resumidamente, o GRASP é um processo iterativo multipartida composto de uma fase de construção que busca gerar uma solução inicial de melhor qualidade e uma fase de busca local para melhorias o procedimento [Feo e Resende, 1995]. Já o procedimento de Busca de Tabu começa a partir de uma solução inicial, após a qual um conjunto de soluções vizinhas (espaço de busca) é determinado e avaliado. A heurística, em seguida, move-se para a melhor solução, não-tabu, e torna a solução anterior tabu por um número de iterações [Glover, 1989].

Considerou-se como base o modelo já implementado por Nobibon et al. [2011] e incorporou-se o efeito de canibalismo entre produtos que, segundo Traylor [1986], ocorre quando a venda de um dos produtos de uma empresa reduz a venda de outro produto da mesma empresa. Para resolver esse problema desenvolveu-se uma estratégia GRASP/Busca Tabu, considerando que se opte pela promoção de apenas um produto entre dois similares, e assim obter resultados significativamente satisfatórios, tanto no que diz respeito à qualidade da solução quanto ao tempo de computação.

Esta pesquisa busca auxiliar as organizações para otimização e tomada de decisão, principalmente em campanhas de promoção que visam a melhor alocação de recursos, combinando a redução de custos com a escolha de clientes com maior probabilidade de resposta e consequentemente maximizando o lucro. Além disso, proporciona embasamento teórico e contribuição para a comunidade científica, pois o efeito de canibalismo entre produtos, apesar de ser um fenômeno real, é um tópico inexplorado pela literatura.

O restante do trabalho está organizado da seguinte forma: na seção 2 apresenta-se o problema de marketing direto, na seção 3 a metodologia utilizada, na seção 4 os algoritmos propostos, seguidos pelos experimentos computacionais na seção 5 e as conclusões na seção 6.

2. O Problema de Marketing Direto

Existem poucos trabalhos que tratam especificamente do problema de marketing direto com oferta de produtos. De acordo com Cohen [2004] e Nobibon et al. [2011], esse problema frequentemente é abordado com a aplicação de métodos estatísticos conhecidos como modelos de resposta de marketing (do inglês "*response models*) ". Esses métodos fornecem um apoio para o processo de seleção de clientes para uma campanha de promoção, fornecendo dados que explicam o comportamento de resposta do consumidor baseados em comportamento de compra anterior e

dessa forma prevendo a probabilidade de resposta futura [Piersma e Jonker, 2004]. Entretanto, a presente pesquisa se concentra em estudos específicos em Problema de Marketing Direto com Oferta de Produtos que utilizam métodos de otimização para maximização do lucro.

Bhaskar et al.[2009] consideram a seleção de clientes para uma campanha de promoção com venda cruzada em um banco comercial. O objetivo é maximizar o lucro total sujeito a duas restrições: o limite de orçamento de campanha e o volume de resposta esperado. O ponto principal do estudo reside na incerteza dos parâmetros e a aplicação lógica fuzzy em um conjunto de instâncias reais de venda cruzada de um banco. Cohen [2004] tratou do mesmo objetivo, porém, com um enfoque diferente. Seu estudo considerou os canais promocionais, restrições de orçamento, retorno mínimo e limites inferiores dos números de ofertas dos produtos da campanha promocional. Foi proposta uma formulação de programação matemática que validaram o estudo e apresentou resultados promissores para futuras campanhas.

Nobibon et al. [2011] desenvolveram um modelo de otimização para o problema de marketing direto com oferta de produtos. Foram consideradas restrições de retorno do investimento, número máximo de produtos por campanha e número mínimo de produtos oferecidos por cliente. Foram desenvolvidos oito algoritmos, sete baseados em métodos exatos e um baseado na meta-heurística Busca Tabu. Os melhores resultados foram apresentados pelo algoritmo Busca Tabu e *Branch-and-Price*, respectivamente.

Oliveira et al. [2014] partiram do modelo proposto por Nobibon et al. [2011] e propuseram modificações nas abordagens heurísticas para o problema. Oliveira et al. [2014] apresentaram um algoritmo heurístico composto pela combinação dos algoritmos GRASP (*Greedy Randomized Adaptive Search Procedure*) e GVNS (Busca Geral em Vizinhança Variável). Os resultados encontrados foram superiores aos encontrados na literatura, validando a proposta.

Cetin e Alabas-Uslu [2017] partiram do mesmo modelo, separando o problema em dois subproblemas: a seleção de produtos que serão incluídos na campanha de promoção e a distribuição desses produtos aos clientes. Na primeira fase um novo modelo de programação linear é utilizado para prever quais produtos são selecionados ou removidos da campanha. Na segunda fase, os produtos que foram selecionados na primeira fase são distribuídos aos clientes por outro modelo de otimização. As duas fases estão conectadas através de uma regra heurística. Foram propostas duas heurísticas, ambas apresentando desempenho superior aos apresentados pela literatura prévia.

3. Metodologia

Para o desenvolvimento do algoritmo proposto, foi utilizada a metodologia básica para resolução de problemas no contexto de pesquisa operacional, que incluem as seguintes fases: definição do problema, construção do modelo, solução do modelo, validação do modelo e a implementação da solução [Arenales et al. 2007; Belfiore e Fávero, 2013; Hillier e Lieberman, 2013]. Ressalta-se que, na prática, a implementação de cada fase pode variar ou ocorrer simultaneamente, pois é influenciada pelo tipo de problema e ambiente considerado.

3.1 Definição do Problema e Construção do Modelo

O Problema de Marketing Direto com Oferta de Produtos se concentra em duas decisões: se o produto *j* é ofertado ao cliente *i* e se o produto *j* irá fazer parte da campanha promocional. As variáveis de decisão são variáveis binárias, sendo $x_{ij} \in \{0,1\}$ onde 1, representa que o produto *j* foi ofertado ao cliente i, e 0 caso contrário, e $y_{ij} \in \{0,1\}$ onde 1, representa que o produto *j* foi utilizado na campanha, e 0, caso contrário.

A construção do modelo baseou-se no modelo proposto por Nobibon et al. [2011] e possui os seguintes parâmetros: $\boldsymbol{C}$ corresponde ao conjunto de clientes, $\boldsymbol{O}$ é o conjunto de oferta de produtos, $\boldsymbol{Can}$ corresponde ao conjunto de pares (i,j) de produtos canibais, isto é, aqueles que não podem ser ofertados simultaneamente, $\boldsymbol{j}$ representa o produto ofertado ($j \in O$), $\boldsymbol{i}$ representa o cliente ($i \in C$), $\boldsymbol{r_{ij}}$ representa a probabilidade de o cliente *i* reagir positivamente a uma oferta do produto *j*, $\boldsymbol{p_{ij}}$ é o retorno esperado (receita) do cliente *i* quando o produto *j* é ofertado, $\boldsymbol{c_{ij}}$ é o custo

associado com a oferta do produto *j* ao cliente *i*, $\boldsymbol{NPP_{ij}}$ corresponde ao lucro potencial líquido, do inglês *"net potential profit"*, que representa a diferença de receitas e custos $(p_{ij} - c_{ij})$ associados à oferta do produto *j* ao cliente *i*, $\boldsymbol{f_j}$ é o custo fixo incorrido quando o produto *j* é utilizado na campanha, $\boldsymbol{R}$ (*hurdle rate*) é a taxa mínima de retorno sobre o investimento global da campanha expresso em percentual e depende do risco do investimento, $\boldsymbol{B_{ij}}$ corresponde ao orçamento disponível para cada produto, $\boldsymbol{M_i}$ é o número máximo de ofertas que o cliente *i* pode receber e $\boldsymbol{O_j}$ é o compromisso de quantidade mínima (MQC – *minimum quantity commitment*), ou seja, o número mínimo de clientes que devem receber uma oferta do produto *j* para que este produto faça parte da campanha.

Dessa forma, após a determinação dos parâmetros e variáveis, define-se a função objetivo do problema como o objetivo de maximizar a soma dos lucros líquidos $(p_{ij} - c_{ij})$ de cada oferta do produto *j* ao cliente *i*, menos os custos fixos $\boldsymbol{f_j}$ de cada produto utilizado na campanha, considerando o efeito de canibalismo entre produtos.

$$\max \quad \sum_{i=1}^{m} \sum_{j=1}^{n} (p_{ij} - c_{ij}) x_{ij} - \sum_{j=1}^{n} f_j y_j \tag{1}$$

A função objetivo do problema está sujeita às seguintes restrições:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \geq (1 + \mathrm{R}) \left[\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} f_j y_j \right] \tag{2}$$

$$\sum_{i=1}^{m} c_{ij} x_{ij} \leq B_j \qquad j = 1, \ldots, n, \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} \leq M_i \qquad i = 1, \ldots, m, \tag{4}$$

$$\sum_{i=1}^{m} x_{ij} \leq m y_j \qquad j = 1, \ldots, n \tag{5}$$

$$\sum_{i=1}^{m} x_{ij} \geq O_j y_j \qquad j = 1, \ldots, n \tag{6}$$

$$y_i + y_j \leq 1 \qquad \forall (i,j) \in Can \tag{7}$$

$$y_j, x_{ij} \in \{0,1\} \qquad i = 1, \ldots, m, \; j = 1, \ldots, n. \tag{8}$$

A primeira restrição (2) diz respeito à taxa mínima de retorno, que garante que o total de receita exceda o total dos custos da campanha adicionados do percentual *R* (*hurdle rate*). A segunda restrição (3) garante que o orçamento $\boldsymbol{B_{ij}}$ alocado ao produto *j* não seja excedido. A terceira restrição (4) assegura que serão ofertados no máximo $\boldsymbol{M_i}$ produtos ao cliente *i*. As próximas restrições (5) e (6) referem-se ao número mínimo de ofertas que devem ser feitas para que o produto *j* faça parte da campanha. Logo, se o produto não faz parte da campanha ele não será ofertado a nenhum cliente, porém, se o produto *j* fizer parte da campanha, pelo menos $\boldsymbol{O_j} > 0$ clientes receberão a oferta. A seguinte restrição (7) impede a oferta de produtos similares (efeito canibalismo), tornando-os mutuamente excludentes. Por fim, a restrição de integralidade (8).

4 Algoritmos Propostos

4.1 Geração de uma Solução Inicial

O objetivo do algoritmo construtivo é gerar uma solução inicial para que a Busca Tabu possa explorar suas vizinhanças, gerando melhoras nesta solução. O algoritmo construtivo implementado por Nobibon et al. [2011] trata-se de um construtivo guloso, tentando inserir dentro da campanha os produtos com maior potencial de lucro e realizar as ofertas mais lucrativas para

cada um destes produtos. Buscando a melhoria do procedimento proposto pelos autores, foi realizada a implementação do método GRASP, alterando o algoritmo construtivo determinístico para um algoritmo não determinístico, sendo capaz de gerar soluções com um certo grau de aleatoriedade. Desta forma, possibilita-se que o algoritmo de Busca Tabu parta de diferentes pontos dentro espaço de soluções.

O algoritmo construtivo dos referidos autores pode ser dividido em três etapas principais: preparação das estruturas auxiliares, atribuição de ofertas mínimas e maximização do lucro. Na primeira etapa, entre as linhas 1 e 7, é construída uma lista ordenada de clientes em ordem decrescente em relação aos seus potenciais de lucro líquido ($\boldsymbol{NPP_{ij}}$) para cada um dos produtos. Durante esta etapa também é calculado o lucro projetado ($\boldsymbol{PR_j}$) do número mínimo de ofertas $\boldsymbol{O_j}$ para cada um dos produtos. O objetivo deste procedimento é identificar quais os melhores candidatos a receberem ofertas de cada um dos produtos disponíveis e quais são os produtos com maior potencial de lucro, informação útil para o processo de tomada de decisão nas duas próximas fases.

Na segunda etapa, linhas 8 a 15, é realizada a inclusão de produtos na campanha e atribuição do número mínimo de ofertas demandado pelos produtos escolhidos. Já com o cálculo do lucro projetado dos clientes referentes ao número mínimo de ofertas para cada um dos produtos, verifica-se se o conjunto mínimo de ofertas $\boldsymbol{O_j}$ de cada um dos produtos examinados respeita as restrições de orçamento ($\boldsymbol{B_{ij}}$) e taxa de retorno ($\boldsymbol{R}$). Após ser feita a avaliação de todos os produtos, o produto mais lucrativo (com o maior $\boldsymbol{PR_j}*$), é inserido na campanha e tem o número mínimo de ofertas atribuídas aos clientes previamente analisados (o mesmo conjunto de clientes que gerou a projeção $\boldsymbol{PR_j}$).

Após a inclusão do produto e suas ofertas, é feito o ajuste do número de ofertas possíveis de serem recebidos por cada cliente ($\boldsymbol{M_{ij}}$) e os procedimentos anteriores são reiniciados com o conjunto dos produtos restantes e os clientes que ainda estão disponíveis. Este é repetido até o ponto onde não se pode gerar um conjunto de ofertas válido para nenhum dos produtos ainda não ofertados.

Na última etapa, linhas 16 a 18, para cada um dos produtos incluídos na campanha são analisados na ordem decrescente em relação aos seus valores de $\boldsymbol{NPP_{ij}}$ os clientes ativos (que ainda possuem ofertas disponíveis para receber) a fim de realizar a atribuição de novas ofertas com o objetivo de esgotar o orçamento remanescente. Se existe orçamento para ser realizada a oferta e a adição dessa oferta não desrespeita a restrição de taxa de retorno, a oferta é efetivada. O procedimento termina quando não existem mais ofertas lucrativas possíveis de serem realizadas para nenhum dos produtos, dentro das restrições do problema.

O algoritmo construtivo original foi transformado em um guloso aleatorizado. Esta alteração foi feita determinando-se um intervalo de melhores clientes, relativo à população total de clientes, no qual serão escolhidas as ofertas. O intervalo definido foi de 10% dos clientes com maior $\boldsymbol{NPP_{ij}}$. Portanto, no ponto de escolha de ofertas na segunda etapa do algoritmo construtivo de Nobibon et al. [2011], citada anteriormente, ao invés de ser sempre escolhido o cliente de maior $\boldsymbol{NPP_{ij}}$, será escolhido, de forma pseudoaleatória, um dos clientes contidos no intervalo previamente delimitado. As alterações aqui discutidas são ilustradas nas linhas 6 a 8 no pseudocódigo a seguir (Algoritmo 1).

Além disso, realizou-se uma segunda alteração no código para considerar a relação de canibalismo entre os produtos. Considerando-se a existência desta relação evita-se inserir ambos produtos canibais durante a construção da solução. Quando um produto é selecionado para ser inserido na campanha durante a segunda etapa do algoritmo construtivo, seu produto canibal não é considerado para as tentativas de inserção de produtos posteriores. Os pseudocódigos a seguir (Algoritmos 1 e 2) ilustram estas alterações, sendo a inclusão do canibalismo especificada na linha 4 do algoritmo 2.

Algoritmos 1 e 2 – Algoritmos construtivos guloso aleatorizado (com e sem canibalismo)

Construtivo Guloso Aleatorizado

1: **lucro_total**:=0, **custo_total**:=0, **V** <- Lista de clients, **Sol** := 0, **ofertas_aleatorias** := 0, **quantidade_ofertas** := 0
2: **para cada** cliente em **V** calcular $\mathbf{NPP_{ij}} := \frac{P_{ij}+C_{ij}}{C_{ij}}$
3: ordena os clientes em **V** em ordem não crescente de $\mathbf{NPP_{ij}}$
4: **para cada** produto **j** $\notin$ **Sol**
5: | **ofertas_aleatorias** := 0
6: | **enquanto quantidade_ofertas** < $\mathbf{O_j}$
7: | | insere oferta do produto **j** para o cliente **i** escolhido aleatoriamente entre
| | 10% clientes em **V** com os maiores $\mathbf{NPP_{ij}}$ em **ofertas_aleatorias**
8: | | **quantidade_ofertas** := **quantidade_ofertas** + **1**
9: | $\mathbf{C_j}$:= custo das ofertas contidas em **ofertas_aleatorias**
10:| $\mathbf{P_j}$:= lucro das ofertas contidas em **ofertas_aleatorias**
11:| $\mathbf{PR_j} := \mathbf{P_j} - \mathbf{C_j} - \mathbf{f_j}$
12: seleciona o produto **j*** com o maior $\mathbf{PR_j}$ que satisfaça $\mathbf{C_{j*}} \leq B_{j*}$ e
lucro_total + $\mathbf{P_{j*}} \geq (\mathbf{1} + \mathbf{R})$(**custo_total**+ $\mathbf{C_{j*}} + \mathbf{f_{j*}}$)
13: **se** existe um produto **j*** e $\mathbf{PR_{j*}} > \mathbf{0}$ **então**
14:| inclui as primeiras $\mathbf{O_{j*}}$ ofertas previamente calculadas em **Sol**, $\mathbf{y_{j*}} := \mathbf{1}$,
| **lucro_total := lucro_total** + P_{j*}, **custo_total := custo_total** + $\mathbf{C_{j*}} + \mathbf{f_{j*}}$
15:| **para cada** cliente pertencente as primeiras $\mathbf{O_{j*}}$ entradas na lista ordenada
16:| | $\mathbf{x_{ij}} := \mathbf{1}$, $\mathbf{M_{ij}} := \mathbf{M_{ij}} - \mathbf{1}$
17:| | **se** $\mathbf{M_{ij}} = \mathbf{0}$, **então**
18:| | | remove cliente **i** de **V**
19:| volte para 4:
20:para cada produto **j** ∈ **Sol**
21:| **para cada** cliente **i** ∈ **V**

Construtivo Guloso Aleatorizado com Canibalismo

1: **lucro_total**:=0, **custo_total**:=0, **V** <- Lista de clients, **Sol** := 0, **ofertas_aleatorias** := 0, **quantidade_ofertas** := 0
2: **para cada** cliente em **V** calcular $\mathbf{NPP_{ij}} := \frac{P_{ij}+C_{ij}}{C_{ij}}$
3: ordena os clientes em **V** em ordem não crescente de $\mathbf{NPP_{ij}}$
4: **para cada** produto **j** $\notin$ **Sol** e que não é canibal aos produtos já adicionados a **Sol**
5: | **ofertas_aleatorias** := 0
6: | **enquanto quantidade_ofertas** < $\mathbf{O_j}$
7: | | insere oferta do produto **j** para o cliente **i** escolhido aleatoriamente entre
| | 10% clientes em **V** com os maiores $\mathbf{NPP_{ij}}$ em **ofertas_aleatorias**
8: | | **quantidade_ofertas** := **quantidade_ofertas** + **1**
9: | $\mathbf{C_j}$:= custo das ofertas contidas em **ofertas_aleatorias**
10:| $\mathbf{P_j}$:= lucro das ofertas contidas em **ofertas_aleatorias**
11:| $\mathbf{PR_j} := \mathbf{P_j} - \mathbf{C_j} - \mathbf{f_j}$
12: seleciona o produto **j*** com o maior $\mathbf{PR_j}$ que satisfaça $\mathbf{C_{j*}} \leq B_{j*}$ e
lucro_total + $\mathbf{P_{j*}} \geq (\mathbf{1} + \mathbf{R})$(**custo_total**+ $\mathbf{C_{j*}} + \mathbf{f_{j*}}$)
13: **se** existe um produto **j*** e $\mathbf{PR_{j*}} > \mathbf{0}$ **então**
14:| inclui as primeiras $\mathbf{O_{j*}}$ ofertas previamente calculadas em **Sol**, $\mathbf{y_{j*}} := \mathbf{1}$,
| **lucro_total := lucro_total** + P_{j*}, **custo_total := custo_total** + $\mathbf{C_{j*}} + \mathbf{f_{j*}}$
15:| **para cada** cliente pertencente as primeiras $\mathbf{O_{j*}}$ entradas na lista ordenada
16:| | $\mathbf{x_{ij}} := \mathbf{1}$, $\mathbf{M_{ij}} := \mathbf{M_{ij}} - \mathbf{1}$
17:| | **se** $\mathbf{M_{ij}} = \mathbf{0}$, **então**
18:| | | remove cliente **i** de **V**
19:| volte para 4:
20:para cada produto **j** ∈ **Sol**
21:| **para cada** cliente **i** ∈ **V**
22:| | **se** $\mathbf{P_{ij}} > \mathbf{C_{ij}}$ e a oferta do produto **j** para o cliente **i** respeita as restrições
| de orçamento e taxa de retorno, faça a oferta e ajuste os valores da solução

4.2 Busca Tabu

Após ser construída uma solução inicial passa-se a utilizar a meta-heurística Busca Tabu, a qual possui uma memória adaptativa para o refinamento da solução. A Busca Tabu realiza melhoras na solução, partindo de uma solução inicial, através de uma busca local que realiza a exploração no espaço de soluções através do conceito de vizinhança. Foram utilizadas as três vizinhanças descritas por Nobibon et al. [2011] juntamente com uma quarta vizinhança incluída durante a implementação deste trabalho.

A primeira vizinhança consiste na troca de uma oferta de um produto entre dois clientes. Na segunda vizinhança escolhem-se dois clientes e dois produtos, sendo que nenhum dos dois clientes recebe a oferta do mesmo produto, dentre os dois produtos escolhidos, e permuta-se as ofertas de ambos produtos entre os dois clientes. Na terceira vizinhança escolhem-se dois clientes e permutam-se as ofertas de todos os produtos que estejam sendo ofertado para um dos clientes e não estejam sendo ofertados para o outro cliente. Neste movimento, devem-se considerar restrições de orçamento e número máximo de ofertas por cliente para cada uma das permutações isoladamente, caso uma das permutações inflija uma das restrições, esta permutação, isoladamente, não é realizada.

Já a quarta vizinhança consiste na permutação de uma oferta de um produto para outro. Essa vizinhança foi criada com o objetivo de mudar-se a cardinalidade da solução, pois as vizinhanças previamente discutidas não são capazes de modificar o número total de clientes alocados a cada produto.

Como critério tabu foram considerados como atributos: produtos e clientes envolvidos nos movimentos efetuados, utilizando-se listas tabu separadas para cada uma das vizinhanças. Os

critérios de parada utilizados para busca tabu foram de 50 iterações sem melhoras ou 1 hora de tempo de execução.

Algoritmo 3– Algoritmo da Busca Tabu

Busca Tabu

```
1:  Sol_atual := Sol_incumbente
2:  enquanto iterações < 50 e tempo < 1 hora
3:  |   vizinhança_1(Sol_atual) => Sol_vizinhança1
4:  |   se Sol_vizinhança_1 > Sol_incumbente então
5:  |   |       Sol_atual := Sol_vizinhança_1,
    |   |       Sol_incumbente := Sol_vizinhança_1
6:  |   senão
7:  |   |       vizinhança_2(Sol_atual) => Sol_vizinhança2
8:  |   |       se Sol_vizinhança_2 > Sol_incumbente então
9:  |   |       |       Sol_atual := Sol_vizinhança_2,
    |   |       |       Sol_incumbente :=Sol_vizinhança_2
10:|   |       senão
11:|   |       |       vizinhança_3(Sol_atual) => Sol_vizinhança3
12:|   |       |       se Sol_vizinhança_3 > Sol_incumbente então
13:|   |       |       |       Sol_atual := Sol_vizinhança_3,
    |   |       |       |       Sol_incumbente :=Sol_vizinhança_3
14:|   |       |       senão
15:|   |       |       |       vizinhança_4(Sol_atual) => Sol_vizinhança4
16:|   |       |       |       se Sol_vizinhança_3 > Sol_incumbente então
17:|   |       |       |       |       Sol_atual := Sol_vizinhança_4,
    |   |       |       |       |       Sol_incumbente :=Sol_vizinhança_4
18:|   |       |       |       senão
19:|   |       |       |       |       Sol_atual = melhor solução entre as vizinhanças
20:|   se foram realizadas 5 iterações sem melhora na solução incumbente
21:|   diversificação(Sol_atual)
```

Em síntese, o fluxo de execução da Busca Tabu começa pela exploração de todas as soluções não tabu de cada uma das vizinhanças declaradas, resultando na escolha de uma solução candidata a melhor solução para cada uma das vizinhanças. Então, a melhor entre as melhores soluções de cada uma das vizinhanças é escolhida como nova solução atual, a lista tabu da vizinhança escolhida é atualizada e o processo de exploração das vizinhanças é reiniciado com a nova solução atual. O Algoritmo 3 especifica o procedimento. Como critério de aspiração aceita-se um movimento tabu que leva a uma solução melhor que a incumbente;

4.3 Diversificação

Após cinco iterações sem melhora durante a Busca Tabu, o algoritmo iniciará um procedimento de diversificação. Este procedimento tem como objetivo principal variar a combinação de produtos ofertados, pois, durante o processo de melhoramento da solução pela Busca Tabu, a lista de produtos ofertados não é alterada.

O processo de diversificação consiste na remoção da oferta de um dos produtos selecionados na solução atual e a inserção da oferta de um dos produtos que não foram selecionados. É tentada a remoção de cada um dos produtos da solução atual, tentando-se inserir um dos produtos que ainda não foram ofertados através da construção de sua base de ofertas de acordo com a lógica de inserção do algoritmo construtivo. Caso a inserção do produto gere uma solução que não respeite as restrições do problema, ela será desconsiderada e a inserção do próximo produto será testada. O procedimento continua até que seja achada uma substituição válida ou todas as possíveis combinações forem testadas. O processo de diversificação da solução é ilustrado no pseudocódigo a seguir (Algoritmo 4).

Como o processo de diversificação altera a composição da solução, o produto selecionado para substituir um dos produtos pertencentes a solução atual não poderá ser um produto com

relação de canibalismo com os produtos que não serão removidos da solução. Esta alteração é demonstrada no pseudocódigo a seguir (Algoritmo 5).

Algoritmos 4 e 5– Algoritmo da diversificação sem canibalismo e com canibalismo

Diversificação

```
1:para cada produto j ∈ Sol_atual
2:|   para cada produto j* ∉ Sol_atual
3:|   |   remove ofertas de j presentes em Sol_atual e ajusta valores da solução
4:|   |   para cada cliente i ∈ V
5:|   |   |   se P_ij > C_ij e a oferta do produto j* para o cliente i respeita as
  |   |   |   restrições de orçamento e taxa de retorno, faça a oferta e ajuste
  |   |   |   os valores da solução
6:|   |   se as ofertas para o produto j* gerarem uma solução válida
7:|   |   |   termina o processo de diversificação
8:|   |   senão retorna a Sol_atual a sua condição original
```

Diversificação com Canibalismo

```
1:para cada produto j ∈ Sol_atual
2:|   para cada produto j* ∉ Sol_atual sendo que j* não é canibal aos produtos
  |   remanescentes em Sol_atual
3:|   |   remove ofertas de j presentes em Sol_atual e ajusta valores da solução
4:|   |   para cada cliente i ∈ V
5:|   |   |   se P_ij > C_ij e a oferta do produto j* para o cliente i respeita as
  |   |   |   restrições de orçamento e taxa de retorno, faça a oferta e ajuste
  |   |   |   os valores da solução
6:|   |   se as ofertas para o produto j* gerarem uma solução válida
7:|   |   |   termina o processo de diversificação
8:|   |   senão retorna a Sol_atual a sua condição original
```

4.4 GRASP em conjunto com Busca Tabu

O conceito central do GRASP refere-se a múltiplas execuções de uma busca local, partindo-se de soluções iniciais diferentes. Desta forma é possível cobrir-se uma região maior do espaço de busca possibilitando que a busca local evite certos mínimos locais.

Utilizando-se do algoritmo construtivo guloso aleatorizado previamente descrito, torna-se possível reiniciar a Busca Tabu em diferentes pontos do espaço de busca. Para este trabalho, definiu-se que serão realizados cinco reinícios aleatórios para a Busca Tabu, e assim, após cada execução da Busca Tabu, seu resultado é guardado e uma nova solução inicial é gerada pelo algoritmo construtivo e é iniciada uma nova execução da Busca Tabu. O melhor resultado dentre os obtidos é utilizado como o final.

Algoritmo 6 - Algoritmo do GRASP/TABU

GRASP

```
1:Sol:= 0, Melhor_Sol := 0, Sol_Resultante
2: enquanto iterações < 5
3:|   construtivo_guloso_aleatorizado() => Sol
4:|   busca_tabu(Sol) => Sol_Resultante
5:|   se Sol_Resultante for melhor que Melhor_Sol
6:|   |   Melhor_Sol = Sol_Resultante
```

Na próxima sessão são apresentados em três etapas os testes e os desempenhos dos algoritmos desenvolvidos em várias instâncias de campanha de promoção. A primeira etapa constitui na reprodução do algoritmo de Busca Tabu, a segunda etapa a aplicação do algoritmo de Busca Tabu em conjunto com GRASP e na terceira o algoritmo de Busca Tabu em conjunto com GRASP com a inclusão de canibalismo entre produtos.

5. Experimentos computacionais

Os testes realizados foram efetuados em três etapas, totalizando 324 execuções em cada etapa, sendo uma execução para cada instância. A primeira fase dos testes, consistiu na reprodução do algoritmo proposto por Nobibon et al. [2011]. Apesar da reprodução ter sido exatamente à implementação descrita na literatura e testada no mesmo conjunto de instâncias, os resultados não foram os mesmos. Justifica-se tais resultados devido ao fato de o algoritmo ter sido desenvolvido

de acordo com a descrição provida pelos referidos autores, pois não se obteve acesso aos códigos originais.

Na segunda etapa, compreendeu a implementação do algoritmo construtivo guloso proposto pelos autores com a alteração para um guloso aleatorizado, com a utilização da meta-heurística Busca Tabu com acréscimo do procedimento GRASP. Uma execução do GRASP compreendeu cinco reinícios aleatórios na Busca Tabu, sendo as soluções geradas pelo guloso aleatorizado. A Busca Tabu teve como critério de parada 50 iterações sem melhoras ou uma hora de execução.

Finalmente, a terceira etapa consistiu na implementação do algoritmo com inclusão do canibalismo entre produtos. Tanto na construção da solução inicial quanto na diversificação, quando um produto é ofertado na campanha, seu produto canibal não é considerado para as tentativas de oferta de produtos posteriores.

Utilizou-se o programa Statistica, pelo qual foram identificados os produtos similares, tratados como canibais. A análise foi realizada no nível de produto em função do lucro simulado por cliente, que correspondeu à diferença entre receitas ($\boldsymbol{p_{ij}}$) e os custos ($\boldsymbol{c_{ij}}$). Os conjuntos de instâncias avaliados contaram com cinco, dez e quinze variáveis e uma variação de 100 a 10.000 clientes. Procedeu-se a seleção de duas variáveis mais próximas no dendrograma da análise de cluster onde foram avaliados cinco tipos de produtos, quatro variáveis nas instâncias com dez tipos de produtos e seis variáveis nas instâncias que continham quinze tipos de produtos. Optou-se pela atribuição de relações de canibalismo que afetassem 40% do conjunto de produtos presentes em cada instância, pois não houve a intenção de que as relações de canibalismo afetassem todos os produtos dos conjuntos. Após a determinação das variáveis similares nos dendrogramas, passou-se a uma segunda fase que consistiu da reanálise dos bancos com quatro e seis variáveis, visando escolher somente um par de variável similar em cada grupo, a fim de identificar os pares mais próximos.

Os algoritmos propostos foram desenvolvidos em linguagem C++ e resolvidos utilizando o software Visual Studio Community, em um conjunto de instâncias disponibilizados por Nobibon et al. [2011]. Estas instâncias estão divididas em quatro grupos: pequeno porte (*small*) S1, S2 e S3 com 100, 200 e 300 clientes, respectivamente, médio porte (*medium*) M1 e M2 com mil e dois mil clientes e ainda o quarto grupo de grande porte (*large*) L com dez mil clientes. Cada conjunto possui três tipos de taxas de retorno, sendo 5%, 10% ou 15% e três tipos de produtos (n) com 5, 10 ou 15 produtos. As instâncias s com um limite de M_i entre 1, 2 ou 3, ou seja, um orçamento pequeno, médio ou grande, respectivamente. Os valores de número máximo de ofertas e orçamento foram gerados mantendo a viabilidade das instâncias.

A fim de avaliar o desempenho dos algoritmos desenvolvidos, adotou-se como medida a qualidade da solução (GAP) expressa em valor percentual de acordo com a equação abaixo (9) onde Z_{UB} é o valor ótimo para as instâncias S1 e S2 ou, quando o ótimo não for encontrado, um valor de limite superior para S3, M1, M2 e L, disponibilizado por Nobibon, Leus e Spieksma, e ainda Z_{AP} é o valor da função objetivo obtido pelos algoritmos implementados.

$$GAP = \frac{ZUB - ZAP}{ZUB} x100 \quad (9)$$

Para uma melhor visualização dos resultados, realizou-se uma comparação entre as médias dos GAPs das implementações desenvolvidas neste trabalho, a implementação de Busca Tabu do trabalho de Nobibon et al. [2011] e os melhores GAPs das implementações de Oliveira et al. [2014].

A Tabela 1 apresenta as médias obtidas pelos experimentos com as instâncias S1 e S2. No trabalho de Nobibon et al. [2011] estes resultados não foram apresentados, pois esse grupo de instâncias foi resolvido na otimalidade, por essa mesma razão Oliveira et al. [2014] não apresentaram seus resultados para S1 e S2, não possibilitando a comparação entre as heurísticas para esse grupo de instâncias.

Tabela 1 - Média GAP (%) para S1 e S2

N	S1			S2		
	5	**10**	**15**	**5**	**10**	**15**
Tabu/GRASP	4.04	4.82	7.18	2.89	4.12	5.28
Canibalismo	12.51	11.99	12.26	10.22	12.40	13.77

No entanto, os resultados são aqui apresentados para demonstrar o desempenho da heurística, que manteve uma qualidade similar de resultados em relação as outras categorias de instâncias.

Já a Tabela 2 apresenta as médias obtidas pelos experimentos com as instâncias S3, M1, M2 e L, onde compara-se os resultados apresentados pelos referidos trabalhos e os resultados deste trabalho.

Tabela 2 – Comparação de resultados - Média GAP (%)

N	S3			M1			M2			L		
	5	**10**	**15**	**5**	**10**	**15**	**5**	**10**	**15**	**5**	**10**	**15**
Nobibon et al. [2011]	6.86	6.52	7.76	7.22	8.54	7.60	9.75	9.58	9.11	**10.86**	11.04	10.23
Oliveira et al. [2014]	6.77	6.45	7.58	7.22	8.42	7.63	9.68	9.59	9.34	**10.86**	**10.94**	**10.22**
Reprodução Tabu	6.59	8.47	7.14	4.66	6.88	7.91	7.78	10.20	10.13	15.31	18.65	19.17
Tabu/GRASP	**4.36**	**6.44**	**5.48**	**3.76**	**5.24**	**5.43**	**5.12**	**8.95**	**7.95**	13.16	18.06	18.25
Canibalismo	12.01	13.72	14.54	11.28	13.96	14.77	12.10	15.06	17.11	16.50	17.90	16.79

Como demonstrado, a implementação da Busca Tabu em conjunto com GRASP obteve melhoras em todas instâncias de pequeno e médio porte. Como o algoritmo de Busca Tabu utiliza um procedimento construtivo determinístico que recebe a instância do problema como entrada, não pode variar a entrada em um mesmo problema, e as vizinhanças serem exploradas em sua totalidade durante a Busca Tabu, o algoritmo sempre explorará a instância do problema da mesma forma. A inclusão de reinícios aleatórios proporcionada pelo GRASP, variando a solução inicial a fim de modificar o local de partida no espaço de soluções e exploração de vizinhanças, de fato potencializou as chances do algoritmo de achar novos máximos locais.

No entanto, para as instâncias L os resultados apresentados pelo algoritmo provaram-se significativamente piores que os resultados obtidos pelos outros autores. Possivelmente esta piora foi resultado do limite de tempo atribuído para a busca local, pois a otimização do código não foi o ponto central da implementação durante este trabalho, resultando na interrupção por limite de tempo na busca local para as instâncias da categoria L.

Com a inclusão de canibalismo estre produtos na implementação, nas instâncias menores, de pequeno e médio porte, a inclusão da restrição gerou piora dos resultados. Possivelmente, o par de produtos canibais pode ter bloqueado uma combinação para qual os outros algoritmos tendiam, ou seja, dois bons produtos se tornaram canibais impossibilitando a melhor combinação de produtos. A combinação inicial do algoritmo construtivo é importante pois, que pelo que foi demonstrado durante os testes, o processo de diversificação normalmente não consegue gerar soluções melhores. A solução normalmente é diversificada se tornando pior e ela não consegue chegar a uma melhora em relação a solução incumbente antes de um novo processo de diversificação ser acionado.

No entanto, o algoritmo construtivo avalia apenas o lucro provido pelo número mínimo de ofertas exigidos por produto, podendo ocorrer, algumas vezes, uma avaliação equivocada de certos produtos. Pode haver casos onde um produto que possua alta lucratividade no número mínimo de ofertas exigidas possua o restante das ofertas ruins, enquanto outros produtos com um lucro no número mínimo menor possuem um conjunto total mais balanceado e podem prover mais lucro no total. Nesses casos, se um dos pares canibais possuir um produto lucrativo que antes estava na solução e o outro produto for problemático, ou seja, nem todas ofertas lucrativas, isso pode

acarretar em uma melhora perceptível nas soluções, pois pode ocorrer de o produto canibal "problemático" ser bloqueado e melhorar a solução, o que ocorreu em alguns casos específicos de instâncias de vários tamanhos.

Portanto, ter eliminado algumas combinações possíveis pode ter auxiliado o construtivo a encontrar uma melhor combinação de ofertas, melhorando alguns resultados. Nas instâncias L, com 10.000 clientes, isso ocorreu com uma maior frequência, em especial nos conjuntos de 10 e 15 produtos, onde a implementação com efeito de canibalismo gerou melhoras em relação a reprodução da Busca Tabu e também da Busca Tabu em conjunto com GRASP, sem o efeito. Esse fenômeno também pode ser atribuído à redução do espaço de busca causada pelo bloqueio de combinações de produtos.

Observa-se que a inclusão de canibalismo gera um número menor de combinações possíveis tornando mais fácil para a busca local explorar a maior parte do espaço de busca. Quando a dimensão do problema aumenta, o número de combinações aumenta exponencialmente, o que faz com que algoritmos de busca local, especialmente os que fazem exploração exaustiva, comecem a ter problemas de tempo de execução. Como o canibalismo, que efetivamente está reduzindo o espaço efetivo de busca, mesmo que não seja uma redução direcionada para melhora, gerou melhoras nas maiores instâncias, pois com o espaço de busca reduzido a busca local conseguiu realizar a exploração mais eficientemente.

Além disso, a inserção da restrição de canibalismo possivelmente restringiu combinações de produtos e alterou o caminho percorrido pela Busca Tabu durante a exploração. Esse bloqueio pode ter forçado a Busca Tabu a percorrer caminhos alternativos que pode possibilitar que a busca chegue em máximos locais melhores do que previamente havia chegado pois, como a busca é determinística, dada uma instância e suas restrições, ela sempre percorrerá o mesmo caminho. Qualquer mudança na configuração do problema, ou seja, alteração nas restrições de uma instância, efetivamente bloqueia certos trechos do espaço de busca, mudando o caminho que a Busca Tabu percorre. Estes caminhos alternativos podem possibilitar que a busca local chegue em melhores máximos locais, gerando uma melhora na solução final.

Optou-se por não se apresentar os tempos computacionais, pois eles foram similares aos apresentados anteriormente na literatura.

6. Conclusões

Este trabalho buscou explorar as possibilidades das meta-heurísticas para resolver eficientemente um problema de campanha de promoção. O marketing direto, cujas estratégias são diretamente apoiadas pelos dados de clientes, busca uma vantagem competitiva na oferta de seus produtos e/ou serviços. É um marketing de relacionamento voltado ao cliente em nível individual, promovendo campanhas de promoção mais focadas a fim obter ofertas cada vez mais assertivas e, consequentemente, a maximização do retorno financeiro. Como objetivo principal, buscou-se desenvolver e avaliar um algoritmo heurístico para resolução do problema de marketing direto, levando em consideração o canibalismo entre produtos, a fim de promover a maximização do feedback de compras dos clientes.

O algoritmo proposto foi baseado na implementação de Busca Tabu de Nobibon *et al*. [2011] e alterado, acrescentando a meta-heurísticas GRASP. O modelo teve também a adição da restrição de canibalismo entre produtos, que ocorre quando a venda de um dos produtos de uma empresa reduz a venda de outro produto da mesma empresa. Mais especificamente, o algoritmo deverá optar pela promoção de apenas um produto entre dois similares. A inclusão do efeito ao problema é um tópico inexplorado na literatura, apesar de ser um fenômeno real e que deve ser considerado sempre que houver ameaça interna de um produto similar.

Os experimentos computacionais apresentaram resultados de maior qualidade em grande parte das instâncias. Não se obteve melhora em todos os conjuntos apenas nas instâncias de grande porte, possivelmente ao tempo limitado de busca local. O tratamento de canibalismo no problema em questão foi realizado pela análise se agrupamentos, onde se classificam uma amostra de dados em grupos menores mutuamente excludentes, com base nas similaridades. Já com a inclusão de

canibalismo estre produtos na implementação, nas instâncias menores a inclusão da restrição gerou piora dos resultados na maior parte das instâncias. No entanto, o trabalho mostrou-se relevante e significativo com os resultados apresentados.

Como trabalhos futuros sugere-se como alterações no processo de diversificação e vizinhanças, além da adição o efeito halo ao modelo, que ocorre quando a promoção de um produto pode aumentar a venda de outro produto, levando a complementaridades e incremento no lucro total. Até o momento não foi encontrado nenhum trabalho relacionado com isso na literatura.

Referências

ABEDI, V. S. (2017) Allocation of advertising budget between multiple channels to support sales in multiple markets. *Journal of the Operational Research Society***.** v. 68, n. 2, p. 134 – 146.

ARENALES, M. N.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. (2007) Pesquisa Operacional Para Cursos De Engenharia. Rio De Janeiro: Elsevier.

BELFIORE, P.; FÁVERO, L.P. (2013) Pesquisa operacional para cursos de engenharia. Rio de Janeiro: Elsevier.

CETIN, F.; ALABAS-USLU, C. (2017) Heuristic solution to the product targeting problem based on mathematical programming. *International Journal of Production Research*, v. 55, n.1, p. 3 – 17.

COHEN, M.-D. (2004) Exploiting response models: optimizing cross-sell and up-sell opportunities in banking. *Information Systems*, v. 29, n. 4, p. 327–341, jun.

FEO, T.; RESENDE, M. A. Greed Randomized Adaptive Search Procedures. *Journal of Global Optimization*, v. 6, p. 1-27, 1995.

GLOVER, F. Tabu search—part I. *ORSA Journal on Computing*. v. 1, n.3, pp 190–206, 1989.

HILLIER, F. S.; LIEBERMAN, G. J. (2013) Introdução à pesquisa operacional. 9. ed. Porto Alegre: AMGH.

KOTLER, P.; ARMSTRONG, G. (2011) Principles of Marketing. 14. ed. Prentice Hall.

NOBIBON, F. T.; LEUS, R.; SPIEKSMA, F. C. (2011) Optimization models for targeted offers in direct marketing: Exact and heuristic algorithms. *European Journal of Operational Research*, v. 210, n. 3, p. 670 – 683.

OLIVEIRA, T.A.; COELHO, V.N.; SOUZA, M.J.F.; BOAVA, D.L.T.; BOAVA, F.M.F.M.; COELHO, L.M.; COELHO, B.N. (2014). Um algoritmo heurístico aplicado ao problema de oferta de produtos em campanhas de marketing direto. *Anais do XLVI Simpósio Brasileiro de Pesquisa Operacional*, p. 103 – 114.

SERET, A.; VERBRAKEN, T.; VERSAILLES, S.; BAESENS, B. (2012) A new SOM-based method for profile generation: Theory and an application in direct marketing. *European Journal of Operational Research*, v. 220, n. 1, p. 199 – 209.

TRAYLOR, M. B. (1986) Cannibalism in multibrand firms. *Journal of Consumer Marketing*, v. 3, n. 2, p. 69-75, Spring.