57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

# Otimização do Cross-Selling em Marketplaces Digitais: Um Algoritmo Multi-Start Random-Key Simulated Annealing

**Josicleyton Azevedo dos Santos, Fábio Francisco da Costa Fontes, Dario José Aloise**
Pós-Graduação em Ciência da Computação – UERN-UFERSA
Av. Francisco Mota, 572, Costa e Silva, Mossoró, RN – CEP: 59.625-900
josicleyton.santos@alunos.ufersa.edu.br, fabio_fontes@ufersa.edu.br, darioaloise@uern.br

**Luciano Carlos Azevedo da Costa**
Universidade Federal da Paraíba
Cidade Universitária, João Pessoa, PB – CEP 58.051-900
luciano.costa@academico.ufpb.br

**Matheus da Silva Menezes**
Universidade Federal do Rio Grande do Norte
Av. Senador Salgado Filho, s/n, Campus Universitário, Lagoa Nova, Natal, RN – CEP 59.078-970
matheus.menezes@ufrn.br

## RESUMO

Este trabalho propõe um algoritmo mateheurístico para otimização de estratégias de *cross-selling* em *marketplaces* digitais, combinando múltiplos pontos de partida com chaves aleatórias e *simulated annealing* (MSRKSA). A abordagem busca soluções de boa qualidade em menor tempo que métodos exatos, atendendo à necessidade de agilidade e personalização em ambientes dinâmicos. O modelo matemático ampliado considera *lift* entre itens, perfis de clientes, disponibilidade em estoque e histórico de compras. Os testes com instâncias artificiais mostraram que o MSRKSA, embora apresente desvio médio de 28,06% em relação ao ótimo do CPLEX, obteve tempos de execução significativamente menores, sendo mais viável para cenários de maior escala. A proposta se destaca como alternativa promissora à resolução exata, com potencial de uso prático em sistemas de recomendação personalizados em *marketplaces* digitais.

**PALAVRAS CHAVE. *Cross-selling*, Mateheurística, *Marketplaces*.**

**Tópicos (Metaheurísticas, Outras Aplicações em PO)**

## ABSTRACT

This work proposes a matheuristic algorithm for optimizing cross-selling strategies in digital marketplaces, combining multi-start with random keys and simulated annealing (MSRKSA). The approach aims to find high-quality solutions in less time than exact methods, addressing the need for agility and personalization in dynamic environments. The enhanced mathematical model considers item lift, customer profiles, stock availability, and purchase history. Tests on artificial instances showed that although MSRKSA presented an average deviation of 28.06% compared to the CPLEX optimum, it achieved significantly lower execution times, proving more feasible for large-scale scenarios. The proposed method stands out as a promising alternative to exact resolution, with practical potential for use in personalized recommendation systems in digital marketplaces.

**KEYWORDS. Cross-selling, Matheuristic, Marketplaces.**

**Paper topics (Metaheuristics, Other Applications in OR)**

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

## 1. Introdução

As plataformas digitais de *marketplace* desempenham um papel central na economia atual ao operarem como intermediárias entre fabricantes, varejistas e consumidores, facilitando a compra e venda de bens e serviços. Elas ampliam o acesso ao mercado, reduzem os custos de transação e promovem a internacionalização dos negócios [Yu et al., 2022; Cano et al., 2022]. Além disso, contribuem para o crescimento econômico ao viabilizarem novos modelos de negócio, como a economia do compartilhamento, e ao apoiarem diversos formatos de *e-commerce*, como *business-to-business*, *business-to-consumer* e *consumer-to-consumer* [Täuscher e Laudien, 2017; Cano et al., 2022]. Embora possam complementar os canais próprios de vendas dos varejistas, essas plataformas também apresentam desafios, como o risco de competição interna entre os canais, caso não haja um gerenciamento estratégico adequado [Maier e Wieringa, 2020; Alaei et al., 2020].

Para potencializar seu desempenho, estratégias como o *cross-selling* podem ser adotadas, aumentando o valor médio dos pedidos e o engajamento dos clientes por meio da recomendação de itens adicionais de interesse [Behera et al., 2020]. O uso de *marketing* digital personalizado e sistemas de recomendação permite adaptar ofertas às preferências individuais, elevando tanto as vendas quanto a satisfação dos consumidores [Behera et al., 2020]. Além disso, o efeito cruzado entre canais, em que as vendas *online* influenciam a demanda *offline*, pode ser explorado para otimizar estratégias e maximizar a lucratividade [Maier e Wieringa, 2020; Xu et al., 2023].

Embora o *cross-selling* já tenha sido amplamente estudado, a maior parte dos trabalhos se concentra em abordagens de *machine learning* para previsão ou segmentação [Boustani et al., 2023; Özdemir e Bayrakli, 2022]. Ainda são escassas as propostas que tratam o problema como uma tarefa de otimização combinatória. Desta forma, Santos et al. [2025] propuseram uma estratégia para otimização do *cross-selling* em *marketplaces*, baseada na identificação de lacunas na literatura quanto à adaptação de modelos ao comércio eletrônico e à personalização de recomendações. Para isso, desenvolveram um modelo de programação linear inteira binária que considera o *lift*, a disponibilidade entre itens e a segmentação por perfis de clientes, visando otimizar o *cross-selling* com base na previsão de compras. Porém, embora os testes com um *solver* comercial tenham validado o modelo, apresentaram tempos de execução elevados, mesmo em instâncias pequenas. Isso indica que depender apenas de métodos exatos pode ser inadequado, especialmente diante da dinamicidade e necessidade de respostas rápidas em plataformas digitais. Para tanto, este trabalho busca contribuir com a área ao propor e implementar um algoritmo mateheurístico para o problema apresentado por Santos et al. [2025], com o objetivo de obter soluções de qualidade em menor tempo. A abordagem combina a flexibilidade do *Simulated Annealing* com o uso de chaves aleatórias e a estratégia *multi-start*, tentando explorar de forma eficiente o espaço de soluções. Espera-se com esta proposta otimizar as decisões de *cross-selling* em *marketplaces* digitais, viabilizando recomendações mais ágeis e personalizadas, alinhadas às exigências de um ambiente comercial dinâmico e competitivo. Este artigo está organizado em: a metodologia (seção 2); formulação do modelo matemático (seção 3); mateheurística proposta (seção 4); os resultados computacionais (seção 5); e a conclusão do trabalho (seção 6).

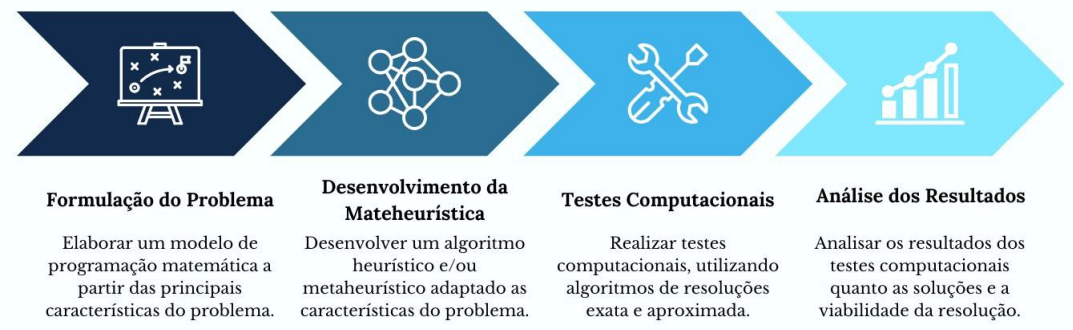
## 2. Metodologia

A metodologia deste estudo adota uma abordagem de pesquisa aplicada, com foco no desenvolvimento e implementação de uma mateheurística para otimizar o *cross-selling* em *marketplaces* digitais. O estudo também é exploratório e explicativo, buscando adaptar e compreender modelos existentes no contexto analisado. A abordagem é majoritariamente quantitativa, utilizando métodos matemáticos e análise de dados numéricos [Rodrigues, 2007].

A Figura 1 apresenta o processo de pesquisa. A primeira etapa consiste na formulação matemática do problema, com base em Santos et al. [2025], utilizando variáveis binárias para representar decisões de compra e relações de *cross-selling* entre produtos ou serviços. A formulação considera métricas de associação entre itens, disponibilidade em estoque, perfis de clientes e características que expandem o modelo original. Em seguida, apresenta-se um algoritmo mateheurístico desenvolvido, que integra resoluções exatas e heurísticas de forma colaborativa.

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

Figura 1: Etapas da pesquisa




A abordagem combina *Simulated Annealing*, chaves aleatórias e *multi-start*, integrando-se à resolução exata do modelo matemático para explorar melhor o espaço de soluções e a qualidade dos resultados. A sua eficácia foi avaliada por meio de testes computacionais em duas etapas: (i) resolução do modelo com o CPLEX; e (ii) aplicação do algoritmo proposto, permitindo comparar os resultados e analisar seu desempenho. As instâncias de teste foram geradas aleatoriamente e adaptadas às particularidades do problema, já que, por se tratar de uma abordagem inédita para o *cross-selling* em *marketplaces*, não há instâncias na literatura nem acesso a dados reais. Por fim, os resultados foram analisados com base no tempo de execução e qualidade das soluções em diferentes cenários, avaliando o seu desempenho, afim de identificar limitações e sugerir melhorias.

## 3. Modelo Matemático

O modelo matemático proposto por Santos et al. [2025] visa otimizar o *cross-selling* em *marketplaces*, associando diretamente produtos ou serviços aos clientes, diferentemente de abordagens anteriores focadas na disposição física dos itens, como visto em Ahn [2012], Sah et al. [2019] e Ryu et al. [2021]. O modelo considera o *lift* e a disponibilidade dos produtos ou serviços, e os perfis de clientes, a fim de otimizar o *cross-selling* com base em uma previsão de compras. Neste trabalho, considerou-se propor ao modelo de Santos et al. [2025] algumas modificações:

- Permitir que o cliente $k$ selecione ou compre múltiplos produtos ou serviços $i$; modelo de Santos et al. [2025] limitava a apenas 1;
- Acrescentar o histórico de compras de cada cliente como um dado de entrada, de modo a não recomendar produtos ou serviços já adquiridos anteriormente.

A Tabela 1 resume as entidades, constantes e as variáveis de decisão deste modelo:

Tabela 1: Notação e variáveis do modelo matemático

| **Conjunto de entidades** | |
|---|---|
| **Nomenclatura** | **Descrição** |
| $I$ | Conjunto de todos itens $i$ ou $j$. |
| $I'$ | Conjunto dos pares de itens $(i, j)$ que possuem o valor de $l_{ij}$ maior ou igual a 1. |
| $K$ | Conjunto de todos os clientes $k$ ou $t$. |
| **Constantes (dados de entrada)** | |
| **Nomenclatura** | **Descrição** |
| $l_{ij}$ | Matriz que indica o *lift* entre os itens $i$ e $j$. |
| $y_{kt}$ | Matriz que indica se os clientes $k$ e $t$ estão no mesmo perfil. |
| $w_{ik}$ | Matriz que indica se o item $i$ foi comprado anteriormente pelo cliente $k$. |
| $d_i$ | Vetor que indica a disponibilidade ou estoque do item $i$. |
| **Variáveis de decisão** | |
| **Nomenclatura** | **Descrição** |
| $x_{ik}$ | Denota se o item $i$ será comprado ou selecionado pelo cliente $k$. |
| $z_{ij}$ | Denota se irá ocorrer a recomendação de *cross-selling* entre os itens $i$ e $j$. |

O modelo é formulado como um problema de programação linear inteira binária.

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

$$max \sum_{(i,j)\in I'} l_{ij} \cdot z_{ij} \quad (0)$$

s.a:

$$\sum_{i\in I} x_{ik} \geq 1, \quad \forall k \in K. \quad (1)$$

$$\sum_{k\in K} x_{ik} + \sum_{j|(j,i)\in I'} z_{ji} \leq d_i, \quad \forall i \in I. \quad (2)$$

$$z_{ij} + x_{ik} + x_{jt} \leq 2 + y_{kt}, \quad \forall (i,j) \in I' \text{ e } \forall k,t \in K. \quad (3)$$

$$z_{ij} \leq \sum_{k\in K} x_{ik}, \quad \forall (i,j) \in I'. \quad (4)$$

$$z_{ij} \leq \sum_{t\in K} x_{jt}, \quad \forall (i,j) \in I'. \quad (5)$$

$$x_{ik} \leq \sum_{j|(j,i)\in I'} z_{ji}, \quad \forall i \in I, \text{ e } \forall k \in K. \quad (6)$$

$$z_{ij} + x_{ik} \leq 2 - w_{jk} \quad \forall (i,j) \in I' \text{ e } \forall k \in K. \quad (7)$$

$$x_{ik} \in \{0,1\}, \quad \forall i \in I, \text{ e } \forall k \in K. \quad (8)$$

$$z_{ij} \in \{0,1\}, \quad \forall (i,j) \in I'. \quad (9)$$

A função objetivo (0) busca maximizar o *cross-selling* entre produtos ou serviços adquiridos $i$ e os recomendados $j$, por meio do produto entre $l_{ij}$ e $z_{ij}$. As restrições (1) garantem que cada cliente $k$ escolha ao menos um item $i$, permitindo múltiplas seleções.

As restrições (2) limitam a soma das unidades vendidas e recomendadas de cada item $i$ ao seu estoque $d_i$. Nas restrições (3), o *cross-selling* entre os itens $i$ e $j$, com *lift* $\geq 1$, ocorre quando $i$ é comprado por $k$ e $j$ por $t$, e que $k$ e $t$ estejam no mesmo perfil de clientes $y_{kt}$.

As restrições (4) e (5) impõem que o *cross-selling* entre $i$ e $j$ só ocorra se forem adquiridos por clientes distintos. As restrições (6) exigem que cada cliente $k$ que adquire um item $i$ receba ao menos um outro item $j$ na recomendação. As restrições (7) impedem que um item $j$, já adquirido pelo cliente $k$, seja recomendado no *cross-selling* $z_{ij}$, se $k$ selecionar ou comprar o item $i$. Por fim, as restrições (8) e (9) definem as variáveis de decisão como binárias.

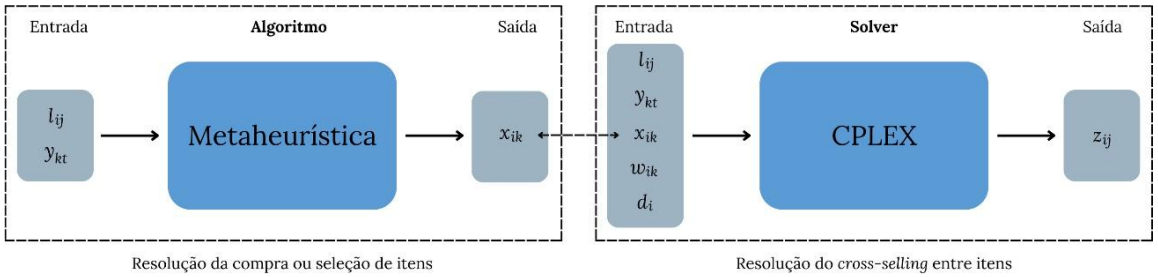
## 4. Mateheurística

O modelo matemático proposto por Santos et al. [2025], ampliado na Seção 3, destaca-se como uma ferramenta promissora para aprimorar estratégias de *cross-selling*. Ao considerar o histórico de compras, perfis dos clientes e interações entre itens, proporciona uma visão mais fundamentada do comportamento de consumo. Apesar do potencial, a complexidade do modelo pode dificultar sua aplicação prática. Para contornar isso, propõe-se uma estratégia híbrida, baseada em mateheurística, em que uma metaheurística determina os valores das variáveis $x_{ik}$ (compras por cliente); e esses valores alimentam um *solver* exato para a resolução de $z_{ij}$, conforme resumido na Figura 2. A metaheurística adotada é o *Simulated Annealing* com chaves aleatórias e múltiplos pontos de partida (*multi-start*), formando o algoritmo *Multi-Start Random-Key Simulated Annealing* (*MSRKSA*), a fim de otimizar o *cross-selling* em plataformas digitais de *marketplace*.

O algoritmo *Simulated Annealing* é uma metaheurística inspirada no processo físico de recozimento de metais, que busca soluções otimizadas simulando o resfriamento gradual de um sistema [Kirkpatrick et al., 1983; Černý, 1985]. Sua principal característica é a aceitação probabilística de soluções piores, com base na distribuição de Boltzmann, permitindo escapar de ótimos locais e explorar melhor o espaço de busca. O seu desempenho depende da escolha adequada de parâmetros como temperatura inicial, função de resfriamento e critérios de parada, sendo amplamente aplicado em problemas complexos de otimização [Talbi, 2009].

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

Figura 2: Esquema do algoritmo mateheurístico para a otimização do *cross-selling* em *marketplaces* digitais pela decomposição do problema

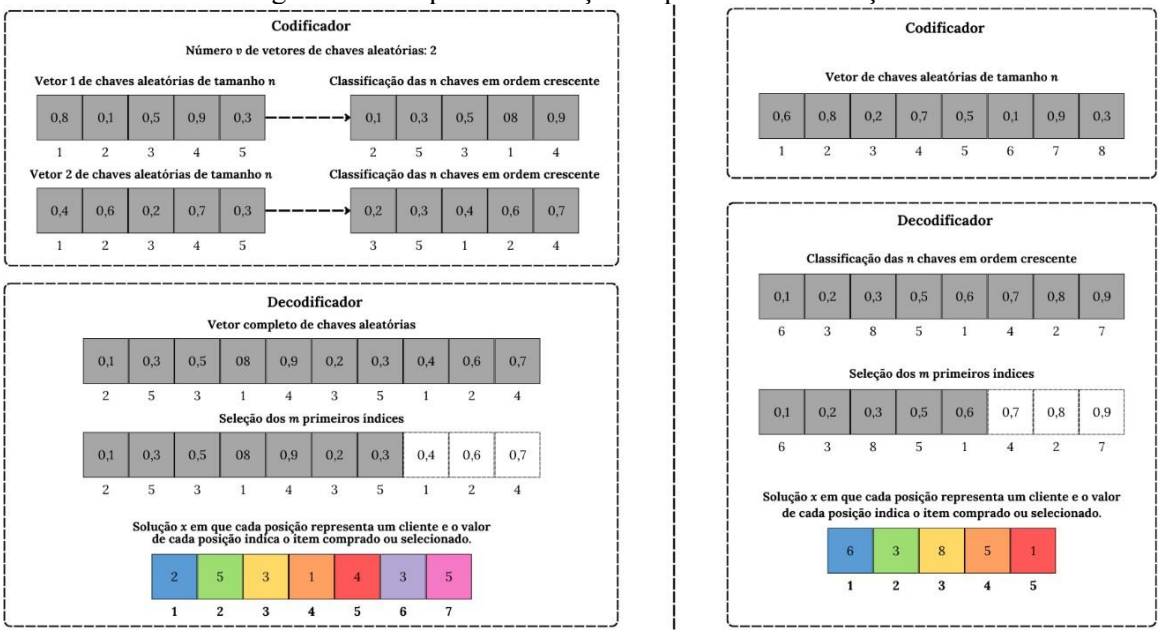



O paradigma das chaves aleatórias consiste em representar soluções por vetores contínuos no intervalo [0,1), os quais são transformados em soluções viáveis no espaço discreto por meio de um decodificador específico. Essa abordagem, introduzida por Bean [1994] e amplamente empregada em algoritmos como o RKGA e o BRKGA, permite separar claramente o problema da heurística de solução. Trabalhos recentes demonstram sua versatilidade ao aplicá-la a diferentes contextos, como roteamento, particionamento e localização de *hub*s [Mangussi et al., 2023; Chaves et al., 2024]. Essa independência entre método e problema motiva esta abordagem neste trabalho, para a geração da solução inicial de $x_{ik}$ – compra ou seleção de itens pelos clientes.

### *4.1. Solução inicial*

Com base na ideia de chaves aleatórias, esta seção descreve a implementação do codificador e decodificador utilizados para gerar a solução inicial do problema proposto. A proposta consiste em construir um vetor solução de $x_{ik}$, em que cada posição representa um cliente $k$ e o valor associado indica o item $i$ escolhido por esse cliente. O algoritmo considera dois cenários, conforme a relação entre o número de itens $n$ e de clientes $m$. A Figura 3 ilustra: à esquerda, o caso $n < m$; e à direita o caso $n \geq m$. Cada cor representa um cliente e o valor, o item atribuído.

Figura 3: Exemplos de alocação de produtos ou serviços




Caso $n < m$, múltiplos vetores de chaves aleatórias de tamanho $n$ são gerados, e cada vetor é ordenado individualmente e seus índices concatenados, formando um único vetor completo

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

ao final; os $m$ primeiros são utilizados na solução final. Caso $n \geq m$, um único vetor de chaves aleatórias de tamanho $n$ é gerado, e após a ordenação, os $m$ primeiros itens são atribuídos aos clientes, e os demais são armazenados em um vetor *unItems* para uso posterior em estratégias de vizinhança. A função $generateSolution$ implementa esse processo. Ela recebe como entrada $n$, $m$ e a disponibilidade dos produtos $d$, retornando à solução gerada $x$, o vetor de itens não utilizados $unItems$ e a disponibilidade atualizada $d'$. A função garante que cada cliente receba exatamente um item, conforme exigido pelas restrições (1) do modelo matemático. A redistribuição de itens adicionais será abordada nas estruturas de vizinhança descritas na próxima seção.

**Algoritmo 1**. $generateSolution(n, m, d')$

```
1   unItems ← [ ]; // Inicializa uma lista vazia para armazenar os itens não escolhidos
2   x ← [ ]; // Inicializa a solução como um vetor vazio de tamanho m
3   if n < m then
4       v = ⌈m/n⌉;
5       if m/n ≠ 0 then
6           v ← v + 1;
7       endif
8       vetor_itens ← [ ]; // Inicializa uma lista vazia para guardar os itens
9       for h ← 1 to v do
10          RK_h ← [ ]; // Inicializa um vetor vazio de chaves aleatórias de tamanho n
11          vetor_h ← [1, ..., n];
12          for i ← 1 to n do
13              Gere um número aleatório p ∈ [0, 1);
14              RK_v[i] ← p;
15          end for
16          Ordene vetor_h com base nos valores de RK_h;
17          Adicione vetor_h ao final de vetor_itens;
18      end for
19      for k ← 1 to m do
20          x[k] ← [vetor_itens[k]];
21          d'[vetor_itens[k]] ← d'[vetor_itens[k]] − 1;
22      end for
23  else
24      RK ← [ ]; // Inicializa um vetor vazio de chaves aleatórias de tamanho n
25      for i ← 1 to n do
26      Gere um número aleatório p ∈ [0, 1);
27      RK[i] ← p;
28      end for
29      vetor_itens ← [1, ..., n];
30      Ordene vetor_itens com base nos valores de RK;
31      for k ← 1 to m do
32          x[k] ← [vetor_itens[k]];
33          d'[vetor_itens[k]] ← d'[vetor_itens[k]] − 1;
34      end for
35      for k ← m + 1 to n do
36          Acrescente vetor_itens[k] a unItems;
37      end for
38  end if
39  return x, unItems, d';
```

### *4.2. Busca local*

Após a geração da solução inicial, o algoritmo $MSRKSA$ explora o espaço de busca por meio da função $generateNeighborSolution$, que cria soluções vizinhas $x'$ a partir da solução atual $x$, a lista de itens não escolhidos ($unItems$) e a disponibilidade atualizada dos itens ($d'$). A função seleciona aleatoriamente uma entre quatro estratégias:

1. Troca: troca os itens de dois clientes escolhidos aleatoriamente.
2. Duplicação: duplica um item de um cliente e o atribui a outro, desde que haja disponibilidade do item e o mesmo ainda não esteja com o cliente alvo.

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

3. Adição: insere um item da lista $unItems$ em um cliente aleatório, desde de que haja disponibilidade do item e a lista $unItems$ não esteja vazia.
4. Remoção: remove um item de um cliente, desde que este continua com, pelo menos, um item, adicionando-o a $unItems$ e atualizando a disponibilidade.

Como a segunda, terceira e quarta estratégia dependem de condições específicas, caso alguma delas seja selecionada, mas não gere uma nova solução, a função força a aplicação da primeira estratégia. Cada operação mantém a viabilidade da solução. Ao final, a função retorna à solução vizinha $x'$, a lista $unItems$ e a disponibilidade $d'$ atualizadas.

**Algoritmo 2. generateNeighborSolution($x$, $unItems$, $d'$)**

```
Inicialize um novo vetor dinâmico de vetores x' a partir de x;
Selecione aleatoriamente strategy como um número inteiro ∈ {1,…,4};
strategyApplied ← false;
switch strategy do
    case 1 // Estratégia 1: Trocar itens de dois clientes escolhidos aleatoriamente
        Escolha um índice inteiro client1 ∈ {1,…,m} aleatoriamente;
        Escolha um índice inteiro client2 ∈ {1,…,m} aleatoriamente, com client2 ≠
        client1; Troque x'[client1] e x'[client2];
        strategyApplied ← true;
    case 2 // Estratégia 2: Duplicar um item de um cliente em outro
        Escolha um índice inteiro clientIndex ∈ {1,…,m} aleatoriamente;
        Defina itemToDuplicate como um item aleatório de x'[clientIndex];
        Escolha um índice inteiro targetClient ∈ {1,…,m} aleatoriamente;
        if targetClient ≠ clientIndex and itemToDuplicate ∉ x'[targetClient]
        x'[targetClient] and d'(itemT oDuplicate) > 0 then
            Adicione itemToDuplicate a x'[targetClient];
            d'(itemT oDuplicate) ← d'(itemT oDuplicate) − 1;
            strategyApplied ← true;
        end if
    case 3 // Estratégia 3: Inserir um item não escolhido em um cliente aleatório
        if unItems ≠ ∅ then
            Selecione um itemToInsert ∈ unItems aleatoriamente;
            Escolha um índice inteiro targetClient ∈ {1,…,m} aleatoriamente;
            if d'(itemToInsert) > 0 then
                Adicione itemToInsert a x'[targetClient];
                Remova itemToInsert de unItems;
                d'(itemT oInsert) ← d'(itemT oInsert) − 1;
                strategyApplied ← true;
            end if
        end if
    case 4 // Estratégia 4: Remover um item aleatório de um cliente
        Escolha um índice inteiro clientIndex ∈ {1,…,m} aleatoriamente;
        if |x'[clientIndex]| > 1 then
            Selecione um itemToRemove ∈ x'[clientIndex] aleatoriamente;
            Remova itemToRemove de x'[clientIndex];
            Adicione itemToRemove a unItems;
            d'(itemT oRemove) ← d'(itemT oRemove) + 1;
            strategyApplied ← true;
        end if
if strategyApplied ≠ true then // Se nenhuma solução for gerada, força a estratégia 1
    Escolha um índice inteiro client1 ∈ {1,…,m} aleatoriamente;
    Escolha um índice inteiro client2 ∈ {1,…,m} aleatoriamente, com client2 ≠
    client1; Troque x'[client1] e x'[client2];
end if
return x, unItems, d';
```

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

### *4.3. Algoritmo MSRKSA para otimização do cross-selling em marketplaces digitais*

Após a descrição das funções auxiliares responsáveis pela geração da solução inicial ($generateSolution$) e da solução vizinha ($generateNeighborSolution$), apresenta-se o funcionamento do algoritmo $MSRKSA$ proposto. O algoritmo recebe como entrada: as matrizes $l$ $w$ e $y$, o vetor $d$, os números de itens $n$ e de clientes $m$. O processo de resfriamento é controlado por $T_{max}$, $T_{min}$, $\alpha$ e $k_{max}$. A Tabela 2 resume as variáveis e constantes utilizadas pelo algoritmo.

Tabela 2: Descrição das variáveis e constantes utilizadas no algoritmo $MSRKSA$

| Variável | Descrição |
|---|---|
| $\boldsymbol{l}$ | Matriz do *lift* entre os produtos ou serviços. |
| $\boldsymbol{y}$ | Matriz binária de perfis entre os clientes. |
| $\boldsymbol{w}$ | Matriz binária de histórico de compras. |
| $\boldsymbol{d}$ | Vetor de disponibilidades dos produtos ou serviços. |
| $\boldsymbol{d'}$ | Vetor de disponibilidades dos produtos ou serviços atualizada. |
| $\boldsymbol{n}$ | Número de produtos ou serviços. |
| $\boldsymbol{m}$ | Número de clientes. |
| $\boldsymbol{unItems}$ | Lista de produtos ou serviços não selecionados ou comprados. |
| $\boldsymbol{x}$ | Solução atual de $x$. |
| $\boldsymbol{z}$ | Solução atual de $z$. |
| $\boldsymbol{f(z)}$ | Custo atual de $z$. |
| $\boldsymbol{x'}$ | Solução vizinha de $x$. |
| $\boldsymbol{z'}$ | Solução vizinha de $z$. |
| $\boldsymbol{f(z')}$ | Custo atual de $z'$. |
| $\boldsymbol{x^*}$ | Melhor solução de $x$ para o problema. |
| $\boldsymbol{z^*}$ | Melhor solução de $z$ para o problema. |
| $\boldsymbol{f(z^*)}$ | Melhor custo de $z$ para o problema. |
| $\boldsymbol{T}$ | Temperatura de controle. |
| $\boldsymbol{T_{max}}$ | Temperatura máxima. |
| $\boldsymbol{T_{min}}$ | Temperatura mínima. |
| $\boldsymbol{k}$ | Variável que controlará o número de iterações. |
| $\boldsymbol{k_{max}}$ | Número máximo de iterações por temperatura. |
| $\boldsymbol{\Delta E}$ | Diferença de energia ou valores entre as soluções atual e vizinha. |
| $\boldsymbol{rand}$ | Número aleatório entre 0 e 1, usado na probabilidade de aceitar soluções piores. |
| $\boldsymbol{\alpha}$ | Parâmetro usado na função de decaimento da temperatura. |

Inicialmente, as soluções $x^*$ e $z^*$são iniciadas como vetores vazios, com custo $f(z^*) = 0$, e a temperatura $T$ é definida como $T_{max}$. O algoritmo segue a lógica do *Simulated Annealing*, com dois laços: o externo gera uma solução inicial com a função $generateSolution$, enquanto o interno busca aprimorá-la em até $k_{max}$ iterações, reduzindo a temperatura ao final de cada ciclo por meio do fator $\alpha$. A solução inicial $x$ gera a lista $unItems$ e o vetor $d'$ atualizado. A função $solveModel$ calcula a solução exata $z$ e seu custo $f(z)$. Essa função considera $l$, $y$, $w$, $d$, $n$, $m$ e $x$, e exclui as restrições (6) do modelo matemático, pois $x$ já é fornecido. Se $f(z) > f(z^*)$, a melhor solução é atualizada. Em seguida, no laço interno, uma nova solução vizinha $x'$ é gerada por $generateNeighborSolution$, atualizando $unItems$ e $d'$. A nova solução $z'$ e custo $f(z')$ é calculado com $solveModel$, e a diferença $\Delta E$ entre as soluções é avaliada. Se $\Delta E < 0$, a nova solução é aceita. Caso contrário, ela só é aceita com probabilidade baseada em $\Delta E/T$. Se aceita, $x$ e $z$ são atualizados. Se a nova solução for superior à melhor encontrada, as variáveis $x^*$, $z^*$ e $f(z^*)$ também são atualizadas. A temperatura é então reduzida ($T \leftarrow T \cdot \alpha$) e o processo continua até que $T \leq T_{min}$. Ao final, o algoritmo retorna as melhores soluções $x^*$ e $z^*$, e o custo $f(z^*)$.

Abaixo está o pseudocódigo detalhado do algoritmo $MSRKSA$:

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

**Algoritmo 3.** $MSRKSA(l, y, w, d, n, m, Tmax, Tmin, \alpha, kmax)$

```
1   x*, z* ← []; // Inicializa vetores vazios para armazenar x∗ e z∗
2   f(z*) ← 0;
3   T ← T_max; // Temperatura recebe o valor da temperatura máxima
4   while T > T_min do
5       d' ← d;
6       x, unItems, d' ← generateSolution(n, m, d);
7       z, f(z) ← solveModel(l, y, w, d, n, m, x);
8       if f(z) > f(z∗) then
9           x* ← x;
10          z* ← z;
11          f(z*) ← f(z);
12      end if
13      for k ← 1 to k_max do
14          x', unItems, d' ← generateNeighborSolution(x, unItems, d');
15          z', f(z') ← solveModel(l, y, w, d, n, m, x');
16          ΔE ← f(z) − f(z');
17          if ΔE ≤ 0 or ΔE/T > rand ∈ [0, 1] then
18              x ← x';
19              z ← z';
20              f(z) ← f(z');
21              if f(z) > f(z*) then
22                  x* ← x;
23                  z* ← z;
24                  f(z*) ← f(z);
25              end if
26          end if
27      end for
28      T ← T · α; // Temperatura atualizada
29  end while
30  return x*, z*, f(z*);
```

## 5. Resultados Computacionais

Foram realizados testes com o método exato CPLEX e a metaheurística $MSRKSA$ para resolver o problema de otimização do *cross-selling* em plataformas digitais de *marketplace*. Ambos os métodos foram implementados em C++. Foram criadas 45 instâncias pequenas com as seguintes características: matrizes $l_{ij}$ (valores entre 0 e 5), simétricas e com diagonal nula, conforme o modelo de Sah et al. [2019]; matrizes $y_{kt}$ (valores binários), também simétricas e com diagonal nula; matrizes $w_{ik}$ com valores binários (0 ou 1); vetores $d_i$ com três níveis de disponibilidade: baixa (5), média (10) e alta (15). As instâncias foram divididas em três grupos: 18 com mais produtos $n$ do que clientes $m$; 9 com $n = m$; 18 com mais clientes do que produtos. Os testes foram executados em um computador com 60 GB de RAM, processador Intel Core i9-10900 (20 núcleos, 2,80 GHz) e GPU RTX 3070. O tempo limite para o CPLEX foi de 48 horas por instância. A metaheurística MSRKSA foi executada 30 vezes por instância.

A calibração da MSRKSA utilizou otimização bayesiana, escolhida por sua eficiência na busca por bons parâmetros com menos avaliações [Sano et al., 2019]. O processo foi aplicado a 5 instâncias escolhidas aleatoriamente e repetido durante 72 horas. Os parâmetros calibrados para o algoritmo MSRKSA foram definidos como: $k_{max}$ igual a 72, $T_{min}$ de 0,01, $T_{max}$ de 80,00, e $\alpha$ igual a 0,9409. A Tabela 3 compara o desempenho do CPLEX e do algoritmo MSRKSA em 45 instâncias do problema de *cross-selling* em *marketplaces*. As colunas indicam o valor da função objetivo (F. Obj.), tempo de execução (T(s)) e o GAP (%) entre a solução encontrada e a ótima.

O CPLEX resolveu 26 instâncias, principalmente as menores, mas falhou em 10 (GAPs entre 5,56% e 74,16%) e excedeu a memória em 9. Esses resultados evidenciam a limitação dos métodos exatos frente ao crescimento da complexidade do problema, o que os torna inviáveis em aplicações que exigem escalabilidade ou resposta em tempo real.

O MSRKSA proposto, por sua vez, apresentou um desvio médio de 28,06% em relação ao ótimo do CPLEX, mas com tempo de execução 47,31% menor (52,69% do tempo do CPLEX).

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

Apesar da perda de qualidade, oferece soluções aceitáveis em menos tempo, o que representa uma vantagem em contextos onde o tempo de resposta é mais crítico que a exatidão da solução.

Contudo, o MSRKSA também apresenta limitações. A qualidade das soluções pode variar bastante entre instâncias, e a estratégia atual de busca local baseada em *first improvement* pode limitar sua capacidade de encontrar soluções melhores. Estratégias como *best improvement* ou a hibridização com outras metaheurísticas poderiam tornar o algoritmo mais robusto e eficaz.

Tabela 3: Resultados computacionais

| Instância | CPLEX | | | MSRKSA | |
|---|---|---|---|---|---|
| | F.O. | T(s) | GAP | F.O. | T(s) |
| 8_6_5 | **86,7** | 2,20 | - | 74,46 | 5.506,71 |
| 8_6_10 | **86,7** | 8,78 | - | 74,27 | 5.529,12 |
| 8_6_15 | **86,7** | 8,31 | - | 75,37 | 4.476,24 |
| 8_8_5 | **81,8** | 42,47 | - | 67,27 | 9.721,37 |
| 8_8_10 | **81,8** | 43,59 | - | 69,37 | 9.100,68 |
| 8_8_15 | **81,8** | 40,63 | - | 68,91 | 9.477,22 |
| 8_10_5 | **72,6** | 3.004,52 | - | 49,42 | 15.325,21 |
| 8_10_10 | **72,6** | 6.822,23 | - | 49,98 | 11.513,19 |
| 8_10_15 | **72,6** | 4.725,00 | - | 49,15 | 15.627,66 |
| 8_12_5 | **57** | 6.942,53 | - | 36,69 | 19.267,70 |
| 8_12_10 | **57** | 11.260,77 | - | 36,37 | 20.173,89 |
| 8_12_15 | **57** | 25.376,61 | - | 36,78 | 21.349,96 |
| 8_14_5 | **66,8** | 55.434,78 | - | 28,36 | 24.967,17 |
| 8_14_10 | **66,8** | 137.950,41 | - | 30,02 | 26.146,68 |
| 8_14_15 | **66,8** | 161.434,88 | - | 28,84 | 32.985,30 |
| 10_6_5 | **120,5** | 83,74 | - | 97,58 | 8.551,64 |
| 10_6_10 | **134,2** | 2.149,59 | - | 101,85 | 7.069,96 |
| 10_6_15 | **134,2** | 2.429,84 | - | 101,43 | 9.747,90 |
| 10_8_5 | **113,6** | 8.426,89 | - | 91,36 | 15.211,57 |
| 10_8_10 | **125,4** | 26.783,17 | - | 94,51 | 13.919,61 |
| 10_8_15 | **125,4** | 37.422,19 | - | 96,29 | 15.153,28 |
| 10_10_5 | 106,8 | 172.800,00 | 7,95% | 80,51 | 20.781,56 |
| 10_10_10 | 110,8 | 172.800,00 | 32,84% | 80,24 | 17.867,21 |
| 10_10_15 | 109,6 | 172.800,00 | 31,93% | 80,56 | 23.800,80 |
| 10_12_5 | 86,6 | 172.800,00 | 19,46% | 62,39 | 27.770,42 |
| 10_12_10 | 87,4 | 172.800,00 | 67,89% | 62,16 | 29.408,55 |
| 10_12_15 | 85,6 | 172.800,00 | 74,16% | 61,68 | 34.384,50 |
| 10_14_5 | *[1] | * | * | 54,94 | 40.584,85 |
| 10_14_10 | 104,2 | 172.800,00 | 55,18% | 53,76 | 46.169,14 |
| 10_14_15 | 101,3 | 172.800,00 | 59,09% | 54,38 | 56.099,13 |
| 12_6_5 | **183,5** | 399,80 | - | 139,89 | 14.131,23 |
| 12_6_10 | **206,9** | 84.194,81 | - | 150,65 | 16.523,98 |
| 12_6_15 | **206,9** | 120.870,54 | - | 156,73 | 15.935,48 |
| 12_8_5 | **182,3** | 12.890,97 | - | 135,98 | 29.255,49 |
| 12_8_10 | **218,4** | 76.401,33 | - | 149,19 | 37.034,10 |
| 12_8_15 | 218,4 | 172.800,00 | 5,56% | 147,72 | 34.971,24 |
| 12_10_5 | 173,2 | 172.800,00 | 6,49% | 124,43 | 25.531,42 |
| 12_10_10 | * | * | * | 126,30 | 31.570,45 |
| 12_10_15 | * | * | * | 126,01 | 28.548,69 |
| 12_12_5 | * | * | * | 114,56 | 49.089,97 |
| 12_12_10 | * | * | * | 115,79 | 46.426,61 |
| 12_12_15 | * | * | * | 115,00 | 50.590,14 |
| 12_14_5 | * | * | * | 114,61 | 54.032,60 |
| 12_14_10 | * | * | * | 118,17 | 72.483,81 |
| 12_14_15 | * | * | * | 115,75 | 48.247,85 |

[1] O símbolo * indica que a árvore de soluções do *branch and cut* se tornou tão grande que não houve memória suficiente para resolver o problema.

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

Em síntese, o CPLEX é adequado apenas para instâncias pequenas e quando se exige solução ótima, acaba sendo limitado por tempo e memória em cenários maiores. O MSRKSA surge como uma alternativa viável para esses casos, entregando soluções aceitáveis com tempo menor, o que sugere que o algoritmo continue sendo aperfeiçoado para trabalhos futuros para melhorias na qualidade e consistência das soluções.

## 6. Conclusão

Diante da crescente relevância das plataformas digitais de *marketplace* no cenário econômico contemporâneo, este trabalho apresentou uma contribuição significativa ao propor uma abordagem mateheurística para a otimização do *cross-selling*, alinhada às exigências de personalização, agilidade e escalabilidade desses ambientes. A partir da identificação de lacunas na literatura, especialmente no que tange à adaptação de modelos matemáticos à realidade dinâmica dos *marketplaces*, desenvolveu-se e avaliou-se uma estratégia baseada na combinação do *Simulated Annealing*, chaves aleatórias e *multi-start*. Embora a qualidade das soluções obtidas ainda possa ser aprimorada, especialmente em comparação aos métodos exatos, a proposta se mostra como alternativa prática frente à complexidade computacional envolvida. As análises e sugestões apresentadas para futuras melhorias reforçam a abertura de novas frentes de investigação, seja na modelagem, nas técnicas de otimização aplicadas ou na validação com dados reais.

Assim, esta pesquisa não apenas avança na busca por soluções mais eficientes para o problema de *cross-selling* em *marketplaces*, como também oferece subsídios para que gestores e desenvolvedores de plataformas possam tomar decisões mais informadas e estratégicas. A integração entre teoria e prática, aliada à contínua inovação metodológica, se mostra essencial para lidar com os desafios e potencialidades dos negócios digitais.

## Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

## Referências

Ahn, K. I. (2012). Effective product assignment based on association rule mining in retail. *Expert Systems with Applications*, *39*(16), 12551-12556.

Alaei, A. M., Taleizadeh, A. A., e Rabbani, M. (2022). Marketplace, reseller, or web-store channel: The impact of return policy and cross-channel spillover from marketplace to web-store. *Journal of Retailing and Consumer Services*, *65*, 102271.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, *6*(2), 154-160.

Behera, R. K., Gunasekaran, A., Gupta, S., Kamboj, S., e Bala, P. K. (2020). Personalized digital marketing recommender engine. *Journal of Retailing and Consumer Services*, *53*, 101799.

Boustani, N., Emrouznejad, A., Gholami, R., Despic, O., & Ioannou, A. (2023). Improving the predictive accuracy of the cross-selling of consumer loans using deep learning networks. *Ann. Oper. Res.*, 339, 613-630.

Cano, J. A., Londoño-Pineda, A., Castro, M. F., Paz, H. B., Rodas, C., e Arias, T. (2022). A Bibliometric Analysis and Systematic Review on E-Marketplaces, Open Innovation, and Sustainability. *Sustainability*, *14*(9), 5456.

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, *45*, 41-51.

Chaves, A. A., Resende, M. G., & Silva, R. (2024). A random-key GRASP for combinatorial optimization. *arXiv preprint arXiv:2405.18681*.

Kirkpatrick, S., Gelatt Jr, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

Maier, E., e Wieringa, J. (2021). Acquiring customers through online marketplaces? The effect of marketplace sales on sales in a retailer's own channels. *International Journal of Research in Marketing*, *38*(2), 311-328.

Mangussi, A. D., Pola, H., Macedo, H. G., Juliao, L. A., Proença, M. P. T., Gianfelice, P. R. L., e Salezze, B. V. (2023). Meta-heurısticas via chaves aleatórias aplicadas ao problema de localização de hubs em árvore. *Anais do Simpósio Brasileiro de Pesquisa Operacional.*

Özdemir, Y., & Bayrakli, S. (2022). A Case Study on Building a Cross-Selling Model through Machine Learning in the Insurance Industry. *European Journal of Science and Technology*.

Ryu, M., Ahn, K. I., e Lee, K. (2021). Finding effective item assignment plans with weighted item associations using a hybrid genetic algorithm. *Applied Sciences*, *11*(5), 2209.

Sah, B., Titiyal, R., e Bhandari, D. D. (2019). Product assignment using quadratic assignment model in retail. *International Journal of Services and Operations Management*, *32*(1), 25-43.

Sano, S., Kadowaki, T., Tsuda, K., & Kimura, S. (2020). Application of Bayesian optimization for pharmaceutical product development. *Journal of Pharmaceutical Innovation*, *15*, 333-343.

Santos, J. A., Da Costa Fontes, F. F., Aloise, D. J. e Borges, R. P. (2025). Otimização do cross-selling em marketplaces digitais: um modelo matemático para aumentar a satisfação do cliente e lucratividade dos fornecedores. *Trabalho apresentado em Anais do LVI Simpósio Brasileiro de Pesquisa Operacional*.

Talbi, E. G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.

Täuscher, K., E Laudien, S. M. (2018). Understanding platform business models: A mixed methods study of marketplaces. *European management journal*, *36*(3), 319-329.

Xu, X., He, P., Zhou, L., e Cheng, T. C. E. (2023). Coordination of a platform-based supply chain in the marketplace or reselling mode considering cross-channel effect and blockchain technology. *European Journal of Operational Research*, *309*(1), 170-187.

Yu, J., Zhao, J., Zhou, C., e Ren, Y. (2022). Strategic business mode choices for e-commerce platforms under brand competition. *Journal of Theoretical and Applied Electronic Commerce Research*, *17*(4), 1769-1790.

**DOI:** 10.59254/sbpo-2025-212284
https://proceedings.science/p/212284?lang=pt-br