LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

Estratégias de pricing para o problema de orientação de times com conjuntos

Pedro dos Santos Zanelato, Francisco Ferreira Lima Neto, Pedro Paulo A. de Paula e Silva, Edna A. Hoshino
Universidade Federal de Mato Grosso do Sul//Faculdade de Computação
Av. Costa e Silva,s/n. Cidade Universitária. Campo Grande-MS.
{pedro.zanelato,francisco.ferreira,p_paulo,edna.hoshino}@ufms.br

RESUMO

Nesse trabalho estuda-se o problema de orientação de times com conjuntos, que consiste em encontrar rotas para uma frota de veículos que respeitem uma duração máxima de percurso e maximizem os lucros coletados, os quais são associados a conjuntos de clientes. Propõe-se diferentes estratégias para resolver o problema de *pricing* que ocorre em um algoritmo de *branch-and-price*, proposto para resolver o problema estudado, motivado pelo sucesso deste tipo de algoritmo para outras variantes do problema de roteamento de veículos. Dado o alto custo computacional para resolver o problema de *pricing*, avalia-se o uso da relaxação *ng-route* e técnicas de aceleração para resolvê-la, como também heurísticas para gerar *ng-routes* a baixo custo computacional.

PALAVRAS CHAVE. branch-and-price, métodos de aceleração, heurísticas de pricing.

Tópicos: Programação matemática, Metaheurísticas, Otimização combinatória

ABSTRACT

This work studies the set team orienteering problem, which consists of finding routes for a fleet of vehicles that respect a maximum route length and maximize the profits collected, which are associated with sets of customers. Different strategies are proposed to solve the pricing problem that occurs in a branch-and-price algorithm, proposed to solve the problem, motivated by the success of this type of algorithm for other variants of the vehicle routing problem. Given the high computational cost to solve the pricing problem, the use of ng-route relaxation and acceleration techniques to solve it are evaluated, as well as heuristics to generate ng-routes at low computational cost.

KEYWORDS. branch-and-price, speedup tools, pricing heuristics.

Topics: Mathematical programming, Metaheuristics, Combinatorial Optimization

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

1. Introdução

O problema de orientação, em inglês *orienteering problem* (OP), é uma generalização do problema do caixeiro viajante em que não é obrigatório que todos os clientes sejam visitados e há um lucro associado a cada cliente, o qual é coletado se o cliente é visitado pelo percurso do viajante. O objetivo do OP é maximizar a soma dos lucros dos clientes visitados, respeitando-se um limite de tempo do percurso. Laporte e Martello [1990] provaram que o OP é um problema NP-difícil. Dentre as variantes do OP, duas delas se destacam. O problema de orientação de times, em inglês *Team Orienteering Problem* (TOP), trata-se de uma generalização do OP em que podem ser utilizados múltiplos veículos para coletar os lucros e cada um deles deve satisfazer o limite de tempo [Chao et al., 1996]. Outra generalização do OP é o problema de orientação com conjuntos, em inglês *Set Orienteering Problem* (SOP), que difere do OP ao considerar uma partição dos clientes em conjuntos e atrelar os lucros aos conjuntos, tal que o lucro de um conjunto é coletado quando, pelo menos, um dos clientes do conjunto é visitado [Archetti et al., 2018].

O problema de orientação de times com conjuntos, em inglês *Set Team Orienteering Problem* (STOP), é uma generalização tanto do TOP quanto do SOP. Ou seja, no STOP considera-se a existência de múltiplos veículos e uma partição dos clientes em conjuntos com lucros associados. Uma instância do TOP pode ser considerada uma instância do STOP, na qual a partição dos clientes é formada por conjuntos unitários, um para cada cliente. Já uma instância do SOP pode ser transformada em uma instância do STOP, na qual a frota de veículos possui apenas um veículo. Não é difícil de observar que o STOP é NP-difícil, pois generaliza o OP, por uma redução similar.

Pelo nosso conhecimento, o STOP não foi estudado anteriormente na literatura. O único problema relacionado é o problema de orientação de times com conjuntos e janelas de tempo, em inglês *Set Team Orienteering Problem with Time Windows* (STOPTW), que foi proposto e estudado por Yu et al. [2023]. Nesse trabalho, os autores propuseram um modelo compacto e uma metaheurística de *Simulated Annealing* com aprendizado por reforço para o STOPTW.

O uso do método da geração de colunas é uma prática consolidada para variantes do problema de roteamento de veículos [Costa et al., 2019]. É comum o uso de relaxações do problema de *pricing* (PP), dada a complexidade computacional deste. Dentre essas relaxações, a relaxação *ng-route* tornou-se o estado-da-arte desde a sua proposição por Baldacci et al. [2011]. Porém, mesmo com o uso dessas relaxações, resolver o problema *pricing* ainda exige elevado tempo computacional, o que torna necessário o uso de métodos de aceleração e heurísticas de *pricing*.

Neste trabalho, tem-se como objetivo avaliar o impacto de diferentes abordagens para resolver o PP em um algoritmo de *branch-and-price* (B&P) para o STOP. São avaliados métodos de aceleração do algoritmo de *labelling* exato conhecidos da literatura, mas pouco explorados em variantes do OP, assim como são propostas e avaliadas diferentes abordagens heurísticas para resolver a relaxação *ng-route*.

Dada a contextualização do problema e o objetivo do trabalho, as demais partes do texto estão organizadas da seguinte forma: a Seção 2 apresenta a descrição formal e um modelo estendido para o STOP, a Seção 3 descreve os métodos de aceleração do algoritmo de *labelling* utilizados, a Seção 4 apresenta as heurísticas de *pricing* propostas e utilizadas, a Seção 5 descreve os resultados computacionais dos métodos utilizados e, por fim, a Seção 6 consiste nas considerações finais.

2. Descrição formal do problema

Seja $G = (V, E)$ um grafo não orientado completo, com $V = \{0, 1, \ldots, N\}$, sendo que 0 e N representam depósitos e os vértices do conjunto $V' = V \setminus \{0, N\}$ os clientes. Uma rota r é um caminho com extremidades em 0 e em N, passando por um ou mais clientes. Os clientes e as arestas que ocorrem em uma rota são ditos *visitados* pela rota. Dada uma rota r denotamos por $V(r)$ e $E(r)$ os vértices e as arestas visitados pela rota r. Considere uma coleção $\mathcal{C}$ dos subconjuntos de

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

V, formando uma partição de V e na qual $D = \{0, N\}$ é o conjunto de depósitos e pertence a $\mathcal{C}$. Considere que cada conjunto $s \in \mathcal{C}$ está associado a um lucro w_s e que cada aresta $e \in E$ está associada a custo de tempo t_e. O tamanho da frota de veículos é representado por m. Em qualquer rota r, a soma dos custos de tempo t_e das arestas $e \in E(r)$ não pode exceder um limite de tempo dado por uma constante T. O lucro associado a um conjunto s é coletado se, e somente se, pelo menos um vértice $i \in s$ for visitado por um rota. Dessa forma, o *problema de orientação de times com conjuntos* consiste em encontrar até m rotas, cada uma delas respeitando o limite de tempo T e visitando subconjuntos disjuntos de clientes, com o objetivo de maximizar a soma dos lucros coletados.

Seja $\mathcal{R}$ o conjunto de todas as rotas válidas para uma instância do STOP. Consideramos uma variável de decisão binária λ_r, para cada rota $r \in \mathcal{R}$, que assume o valor 1 se a rota r é selecionada na solução. Adicionalmente, consideramos uma variável y_i, para cada vértice $i \in V'$, uma variável x_e, para cada $e \in E$ e uma variável v_s, para cada $s \in \mathcal{C}$. Dada uma rota $r \in \mathcal{R}$, as constantes a_i^r e a_e^r representam a presença do vértice $i \in V'$ e da aresta $e \in E$ em $V(r)$ e em $E(r)$, respectivamente.

Um modelo estendido para o STOP é dado por:

$$(P)\ \max \quad \sum_{s \in \mathcal{C}} w_s v_s \tag{1}$$

$$\text{s.a.} \quad \sum_{r \in \mathcal{R}} \lambda_r \leq m \tag{2}$$

$$\sum_{r \in \mathcal{R}} a_i^r \lambda_r + y_i = 1 \qquad \forall i \in V' \tag{3}$$

$$\sum_{r \in \mathcal{R}} a_e^r \lambda_r - x_e = 0 \qquad \forall e \in E \tag{4}$$

$$\sum_{r \in \mathcal{R}} \sum_{i \in s} a_i^r \lambda_r - v_s \geq 0 \qquad \forall s \in \mathcal{C} \tag{5}$$

$$y_i, x_e, v_s, \lambda_r \in \{0, 1\}, \qquad \forall i \in V', \forall e \in E, \forall s \in \mathcal{C}, \forall r \in \mathcal{R}. \tag{6}$$

A restrição (2) limita o total de rotas da solução, enquanto as restrições (3) garantem que as rotas sejam disjuntas em V' e obrigam que cada vértice i seja visitado por uma rota sempre que $y_i = 0$. As restrições (4) e (5) obrigam, respectivamente, $x_e = 1$ se alguma rota visita a aresta e e $v_s = 0$ se nenhuma rota da solução visita vértices do conjunto s. As demais restrições são de integralidade.

Sejam π_0, π_i, β_e e θ_s as variáveis duais relacionadas às restrições (2), (3), (4) e (5), respectivamente. Dessa forma, resolver PP equivale a encontrar $r' = \arg\max_{r \in \mathcal{R}} \overline{c}_r$, sendo $\overline{c}_r$ o custo reduzido da rota r, o qual é calculado por:

$$\overline{c}_r = -\pi_0 - \sum_{i \in V'} \pi_i a_i^r - \sum_{e \in E} \beta_e a_e^r - \sum_{s \in \mathcal{C}} \sum_{i \in s} \theta_s a_i^r. \tag{7}$$

Analogamente, o custo reduzido $\overline{c}_{ij}$ associado ao par ordenado de vértices (i, j) , no qual s é o conjunto em $\mathcal{C}$ que contém o vértice j, é dado por:

$$\overline{c}_{ij} = -\pi_j - \beta_{ij} - \theta_s. \tag{8}$$

Observe que o STOP pode ser modelado como um problema de minimização, sendo sufi-

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

ciente substituir a função objetivo (1) do modelo por:

$$\min \quad \sum_{s \in \mathcal{C}} -w_s v_s.$$

Deste ponto em diante, adotaremos essa versão de minimização do problema, com o intuito de facilitar o uso das mesmas nomenclaturas usadas na literatura para problemas de roteamento de veículos.

Portanto, consideramos, sem perda de generalidade, que o PP consiste em $\min_{r \in \mathcal{R}} \bar{c}_r$. Como o objetivo de PP é encontrar uma rota $r \in R$ de menor custo reduzido $\bar{c}_r$, o PP pode ser modelado como o problema de caminho mínimo elementar com restrição de recursos (PCMERR).

3. Metodos exatos de *pricing*

Como o PCMERR é NP-Difícil [Dror, 1994], é comum na literatura a utilização de relaxações para o PP, que relaxam a elementariedade do caminho, permitindo determinadas repetições de vértices. Entre elas pode-se destacar: a *k-cycle elimination* [Irnich e Villeneuve, 2006] que relaxa a elementariedade do caminho mínimo com restrição de recursos, mas proíbe subciclos de tamanho menor ou igual a k, a *partial elementarity* [Desaulniers et al., 2008] na qual estabelece que somente um subconjunto de vértices deve respeitar a elementariedade do caminho e a *ng-route* [Baldacci et al., 2011] que utiliza conjuntos de vértices chamados *ng-sets* para determinar quais vértices podem ser repetidos em um caminho. Nessa seção serão discutidos um algoritmo e técnicas de aceleração para resolver o PP do STOP, usando-se a relaxação *ng-route* ao invés do PCMERR, os quais são bem conhecidos na literatura para variantes do problema de roteamento de veículos mas pouco explorados para variantes do OP.

3.1. Relaxação *ng-route*

A relaxação *ng-route* é uma das possíveis estratégias para resolver o PP. Sua utilização permite que o PP seja resolvido por um algoritmo de tempo pseudopolinomial. Ela estabelece que cada vertice $i \in V$ está associado a um conjunto de vértices $NG_i \subseteq V$ (também chamado de *ng-set*) de tamanho Δ. Cada NG_i é composto pelo próprio vértice i e pelos $\Delta - 1$ vértices mais próximos de i. Nessa relaxação uma rota pode repetir um vértice i se, e somente se, entre as duas ocorrências de i, existir um vértice j tal que $i \notin NG_j$.

Cada caminho $p_u = (v_0, v_1, ..., v_n)$ que começa no vértice v_0 e termina em v_n, possui um conjunto associado de vértices proibidos Π_u para os quais p_u não pode ser estendido. Ou seja, um vértice $i \in V$ pode estender p_u se, e somente se, $i \notin \Pi_u$. Π_u é dado por:

$$\Pi_u = \{v_k, k = 1, 2, ..., n-1 : v_k \in \bigcap_{s=k+1}^{n} NG_{v_s}\} \cup \{v_n\} \tag{9}$$

Utilizamos essa relaxação, juntamente com um algoritmo de *labelling* para resolver o PP do STOP. Os conjuntos Π_u foram armazenados utilizando máscaras de bits já que tornam operações entre conjuntos, como a intersecção, em operações mais eficientes. Definimos uma tabela de programação dinâmica M, com dimensão $T \times |V|$, contendo *buckets* de *labels*. Cada entrada $M[t][i]$ contém os *labels* que possuem um tempo acumulado l_t no intervalo $]t-1, t]$ e que terminam no vértice i. Um intervalo é utilizado para guardar os *labels* em M porque l_t muitas vezes não é inteiro e também porque agrupar *labels* facilita a aplicação de regras de dominância.

Labels representam caminhos em G que começam no vértice inicial. Um *label* l de um caminho p pode ser representado como uma tupla $l = (l_{prev}, l_i, l_c, l_t)$, sendo l_{prev} o *label* que l estende, l_i o último vértice em p, l_c o custo reduzido de p e l_t o tempo acumulado de p. O custo reduzido l_c é dado por $l_c = l'_c + \bar{c}_{l'_i l_i}$, sendo l'_c o custo reduzido do *label* l_{prev}, $\bar{c}_{l'_i l_i}$ o custo

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

reduzido associado à aresta (l'_i, l_i), l'_i o último vértice de l_{prev} e l_i o último vértice de l. Além disso, um *label* l é dominado, ou seja pode ser descartado, por outro *label* l' se e somente se:

$$l'_i = l_i \tag{10}$$
$$l'_c \leq l_c \tag{11}$$
$$l'_t \leq l_t \tag{12}$$
$$\Pi(l') \subseteq \Pi(l) \tag{13}$$

Inicialmente, a tabela M é inicializada apenas com os *labels* $(\emptyset, i, \bar{c}_{0,i}, t_{0,i})$, que representam rotas que saem do depósito e visitam somente um outro vértice. A cada iteração o algoritmo tenta adicionar novos *labels* em uma entrada da tabela a partir da extensão de outros *labels* já existentes. Ele repete esse processo até que todas as entradas da tabela tenham sido preenchidas. Depois disso, o algoritmo termina e a(s) rota(s) de menor custo reduzido encontrada(s) é(são) retornada(s).

3.2. Decremental State-Space Relaxation

A Relaxação Decrescente do Espaço de Estados [Martinelli et al., 2014], em inglês *Decremental State-Space Relaxation* (DSSR), é um algoritmo iterativo que relaxa espaço de estados dos conjuntos NG_i. Ele considera que para cada conjunto NG_i existe um subconjunto $NGD_i \subseteq NG_i$ que o substitui no algoritmo *ng-route*. No início do algoritmo, os conjuntos $NGD_i = \emptyset, \forall i \in V$ e a cada iteração verifica-se se a rota com menor custo reduzido respeita a relaxação *ng-route* original. Se a relaxação é desrespeitada, os NGD_i dos vértices que desrespeitam a relaxação original e formam subciclo são atualizados de forma a adicionar o vértice que foi repetido. Após isso, o algoritmo reinicia sua execução. Já se a relaxação é respeitada, o algoritmo termina sua execução e a melhor rota é retornada.

Em nossa abordagem, consideramos que h^2 rotas de menor custo reduzido devem respeitar a relaxação inicial. Ou seja, a cada iteração do algoritmo, h^2 rotas são verificadas e, se necessário, os NGD_i dos vértices de cada rota são atualizados. Essa implementação foi escolhida com o propósito de diminuir o número de iterações do DSSR. Ao final da execução, as h rotas de menor custo reduzido são retornadas pelo algoritmo.

3.3. Bidirectional labelling

Outra possível estratégia para acelerar a execução do PP é utilizar o *Bidirectional labelling* [Giovanni Righini, 2006]. Ela consiste em propagar os *labels* nas duas direções (do vértice inicial ao vértice final como também do vértice final ao vértice inicial). A motivação dessa estratégia é tentar diminuir a quantidade de *labels* que são criados, já que divide o crescimento exponencial dos mesmos entre as duas direções, aumentando assim a eficiência do algoritmo.

Os *labels* são divididos entre *foward labels*, que representam um caminho do vértice inicial em direção ao final, e *backward labels*, que representam um caminho do vértice final em direção ao inicial. Um *forward label* e um *backward label* podem ser estendidos se e somente se seus tempos acumulados $l_t < \lceil T/2 \rceil$. Utilizamos o mesmo algoritmo de *labelling* citado em 3.1 de forma que duas tabelas de programação dinâmica foram criadas: M_{fw} e M_{bw} que armazenam os *foward labels* e *backward labels* respectivamente. Seja $\hat{e} = \arg\max_{e \in E} t_e$, ou seja, $\hat{e}$ é uma aresta em E que maximiza o custo de tempo t dentre todas as arestas do grafo. Temos que $t_{\hat{e}} + \lceil T/2 \rceil \leq T$, M_{fw} tem dimensões $|V| \times (\lceil T/2 \rceil + t_{\hat{e}})$ e M_{bw} possui dimensões $|V| \times \lceil T/2 \rceil$. Caso $t_{\hat{e}} + \lceil T/2 \rceil > T$, M_{fw} terá dimensões $|V| \times T$.

Assim como no algoritmo original, todas as entradas das duas tabelas (M_{fw} e M_{bw}) são percorridas com o objetivo de estender novos *labels*. Ao fim das iterações, rotas pertencentes à matriz M_{fw} com custo reduzido negativo são adicionadas na solução. Dado que $l^{fw} =$

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

$(l^{fw}_{\text{prev}}, l^{fw}_i, l^{fw}_c, l^{fw}_t)$ é um *foward label*, $l^{bw} = (l^{bw}_{\text{prev}}, l^{bw}_i, l^{bw}_c, l^{bw}_t)$ é um *backward label*, Π^{fw} e Π^{bw} são os conjuntos de vértices proibidos associados a l^{fw} e l^{bw} respectivamente, l^{fw} pode ser combinado com l^{bw}, para dar origem a um *label* l, se e somente se $\Pi^{fw} \cap \Pi^{bw} = l^{fw}_i = l^{bw}_i$ e $l^{fw}_t + l^{bw}_t \leq T$.

O algoritmo percorre todas as entradas de M_{fw} e M_{bw} e tenta combinar os *labels* das duas matrizes. Visto que l^{fw} e l^{bw} possuem um custo reduzido associado de l^{fw}_c e l^{bw}_c respectivamente, o custo reduzido l_c de l é dado por $l_c = l^{fw}_c + l^{bw}_c$. Se $l_c < 0$, l é adicionado no conjunto de soluções. Ao final de todos esses passos, o algoritmo termina e as soluções encontradas são retornadas.

4. Heurísticas de *pricing*

Embora o uso de relaxações e técnicas de aceleração tenham impactos positivos no tempo gasto pelos algoritmos exatos de *pricing*, o tempo utilizado pelo algoritmo ainda é elevado [Costa et al., 2019]. O uso de heurísticas de *pricing* permite que o número de chamadas do algoritmo exato seja reduzido, tal que este só é executado quando as heurísticas falham em encontrar uma coluna com custo reduzido negativo. As heurísticas mais utilizadas pela literatura são variações no algoritmo de *labelling* do *ng-route* [Costa et al., 2019], de modo que é perdida a garantia de encontrar o melhor caminho porém a complexidade do algoritmo é reduzida consideravelmente. Algumas dessas heurísticas consistem em: uso de *buckets* com tamanhos limitados [Fukasawa et al., 2006], relaxação parcial das regras de dominância [Archetti et al., 2011] e redução do número de arestas consideradas pelo algoritmo de *labelling* [Contardo e Martinelli, 2014].

O uso de metaheurísticas para resolver o problema de *pricing* foi proposto por [Desaulniers et al., 2008], porém como o PP é resolvido de forma elementar estas tendem a falhar mais em encontrar colunas de custo reduzido negativo do que as heurísticas baseadas em *labelling*. Assim, além da avaliação de heurísticas baseadas no algoritmo de *labelling* para o *ng-route*, propõe-se um abordagens metaheurísticas para encontrar *ng-paths* de custo reduzido negativo.

4.1. Heurística baseada em *labelling*

Dado o uso de *buckets* para armazenar *labels* e a verificação de regras de dominância somente entre *labels* contidos no mesmo *bucket*, uma heurística de *pricing* consiste em manter somente os melhores *labels* de cada *bucket*, o que vale relaxar a condição de dominância expressa em (13). Denomina-se essa heurística como *Aggressive Dominance* (AD).

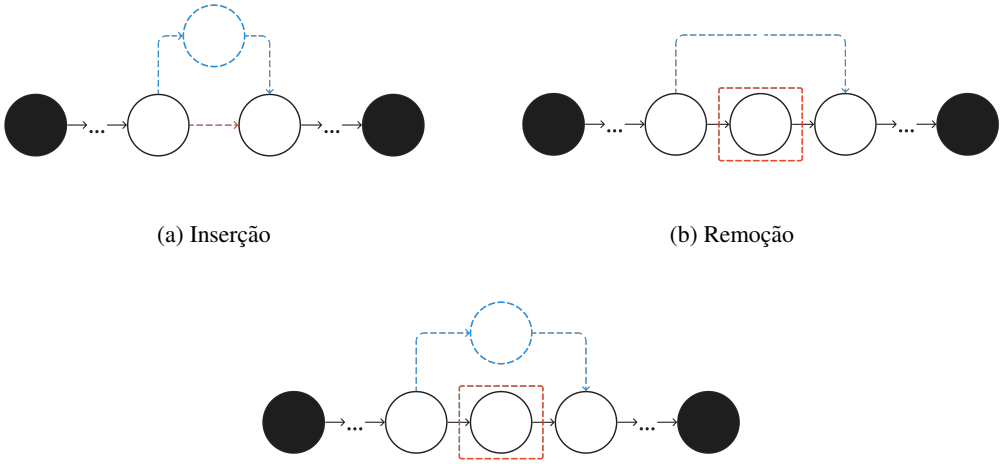
4.2. Metaheurísticas

Propõe-se o uso de metaheurísticas para encontrar *ng-paths* válidos para o *pricing*. Dados dois caminhos parciais a e b com término em um mesmo vértice i, um saindo do depósito de origem e outro do de destino, a concatenação de tais caminhos é um *ng-path* válido se $\Pi(a) \cap \Pi(b) = \{i\}$. Isto se deve ao fato de que se a intersecção dos conjuntos de memória conter qualquer outro vértice, uma repetição inválida ocorre ao concatenar os dois caminhos.

Dado um caminho $p_u = (v_o, v_1, ..., v_n)$ que é um *ng-path* e já tem inicio e fim nos vértices desejados, é possível a realização de buscas locais para encontrar *ng-paths*. São propostas 3 operações de buscas locais de tempo polinomial: troca, remoção e inserção unitárias, como também um procedimento de *Destroy&Repair* baseado em *labelling*.

São propostas dois procedimentos metaheurísticos para encontrar *ng-paths* válidos, uma heurística de *Simulated Annealing* e uma heurística GRASP. A fase construtiva utilizada para as duas heurísticas é baseada na proposta por Feo e Resende [1995] para a GRASP, em que cada visita é adicionada sequencialmente e a etapa de construção da lista restrita de candidatos utiliza a função de avaliação $f = \frac{\bar{c}_{ij}}{t_{ij}}$ ao avaliar inserir o vértice j em uma solução parcial S terminada no vértice i, como também a memória da solução S é atualizada a cada adição de um elemento à rota de forma que um vértice j só pode ser adicionado à lista de candidatos se $j \notin \Pi(S)$ e ainda é possível alcançar o depósito destino após a visita de j.

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br



(a) Inserção

(b) Remoção

(c) Troca

Figura 1: Operações de busca local.

4.2.1. Operações de busca

Dado um caminho p, denomina-se como $\Pi_l(j)$ a memória do caminho entre o depósito origem até o vértice da posição j e, de forma análoga, $\Pi_r(j)$ o caminho do depósito destino até o vértice j. A fim de simplificar a descrição das condições de suficiência para a validade de uma operação de busca, considera-se que as operações em questão já respeitam as restrições de recurso, o que é verificado em tempo $\mathcal{O}(1)$ ao computar os pesos das arestas removidas e as que seriam adicionadas no caso da operação ser realizada.

Uma operação de **troca** de um vértice qualquer por um vértice i na posição j é valida se, e somente se, $\Pi_l(j-1) \cap N(i) \cap \Pi_r(j+1) = \emptyset$ e $i \notin \Pi_l(j-1) \cup \Pi_r(j+1)$. Uma operação de **inserção** de um vértice i na posição entre os vértices das posições $j-1$ e j é válida se, e somente se, $\Pi_l(j-1) \cap N(i) \cap \Pi_r(j) = \emptyset$ e $i \notin \Pi_l(j-1) \cup \Pi_r(j)$. Uma operação de **remoção** de um vértice da posição j é valida se, e somente se, $\Pi_l(j-1) \cap \Pi_r(j+1) = \emptyset$. A verificação de validade de tais procedimentos pode ser realizada em $\mathcal{O}(\frac{n}{b})$, em que b está relacionado à estrutura de dados utilizada para representar os conjuntos. Avaliar todas as possíveis operações de troca ou inserção em uma rota com q visitas tem complexidade $\mathcal{O}(\frac{n^2}{b}q)$, enquanto a avaliação de todas as possíveis remoções tem complexidade $\mathcal{O}(\frac{n}{b}q)$. Como após a aplicação de procedimentos de busca a memória tem de ser atualiza em certas posições, realizar qualquer um dos procedimentos tem complexidade $\mathcal{O}(\frac{n}{b}q)$. Tais procedimentos estão ilustrados na Figura 1.

Também é possível utilizar um algoritmo de *labelling* para reconstruir parte do caminho. Assim, dado um vértice s na posição j do caminho e um vértice o na posição $j+k$ do caminho, é possível por meio um um *labelling* exato ou heurístico encontrar um novo trecho para ser inserido no lugar do trecho das posições $j+1$ a $j+k-1$. Este *labelling* deve, porém, ser inicialmente povoado somente com os caminhos parciais unitários que não violam $\Pi_l(j)$. Como também, um *label* l, que ainda não foi estendido até o vértice destino o, só poderá ser adicionado ao caminho se $\Pi(l) \cap \Pi_r(j+k) = \emptyset$. A complexidade dos procedimentos de busca com *labelling* é pseudopolinomial. Ilustra-se a realização destes procedimentos na Figura 2.

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

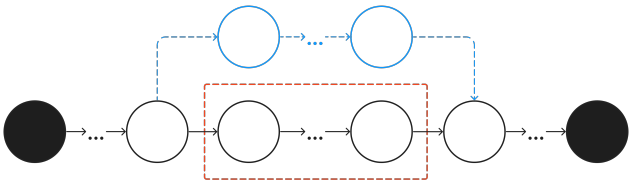


Figura 2: Operação de *Destroy&Repair*

4.2.2. Uma heurística de *Simulated Annealing*

Baseado nos procedimentos de busca de troca, inserção e remoção, propõe-se uma heurística de Têmpera Simulada, em inglês *Simulated Annealing* (SA) [Kirkpatrick et al., 1983], para encontrar *ng-paths* de custo reduzido negativo. O Algoritmo 1 descreve o funcionamento da heurística de *Simulated Annealing* utilizada.

Algoritmo 1 *Simulated Annealing*

Input: Instância do PP *I*, Temperatura T, fator de resfriamento β
Output: Solução S

```
S ← GreedyRandomizedInitialSolution(I)
while Critério de parada não atingido do
    S' ← MovimentoAleatório(S)
    r ← NúmeroAleatório()
    if f(S') ≤ f(S) ou r ≤ e^(−|f(S')−f(S)|/T) then
        S ← S'
    end if
    T ← βT
end while
```

Foi utilizado como critério de parada $T < 10^{-4}$ e não existir nenhum movimento que melhora a qualidade da solução, ou seja, quando $T < 10^{-4}$ o algoritmo continua até chegar a um ótimo local.

4.2.3. Uma heurística GRASP

Foi utilizada uma metaheurística GRASP, do nome em inglês *Greedy Randomized Adaptative Search Procedure* [Feo e Resende, 1995]. As etapas de uma metaheurística GRASP estão descritas no Algoritmo 2. Também foi avaliado para a etapa de busca local da GRASP um procedimento de *Variable Neighborhood Search*, de forma que os procedimentos de busca são chamados sequencialmente com relação ao custo computacional destes enquanto nenhum procedimento que melhora o valor da solução atual não for melhorado. Assim, foi utilizada uma sequência com as operações de busca unitárias, uma busca com *labelling* heurístico e uma busca com *labelling* exato. O procedimento de *labelling* heurístico consiste em um *Destroy&Repair* com o uso de um algoritmo análogo da heurística AD.

5. Experimentos computacionais

Testes computacionais foram realizados em uma máquina com um processador Intel Core i7-4790 (3.6GHz), 32GB de memória RAM e sistema operacional Linux. Foi utilizada a biblioteca

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

Algoritmo 2 *GRASP*

Input: Instância do PP I, fator de aleatoriedade α, número de iterações K
Output: Solução S
1: $S \leftarrow \emptyset$
2: **for each** $k \in \{1, 2, \ldots, K\}$ **do**
3: $S' \leftarrow$*GreedyRandomizedInitialSolution(*I,α*)*
4: $S' \leftarrow$BuscaLocal(S')
5: **if** $f(S') < f(S)$ **then** $S \leftarrow S'$
6: **end if**
7: **end for**

SCIP v3.2.1 [Gamrath et al., 2016] como resolvedor de B&P e o CPLEX v12.6.1.0 como resolvedor da relaxação linear.

Dado que não foram encontradas instâncias apropriadas para o STOP na literatura, os testes foram realizados com a adaptação para o STOP das instâncias do TOP propostas por Chao et al. [1996] por meio do procedimento realizado por Archetti et al. [2018] ao adaptar instâncias do OP para o SOP. Os testes para os algoritmos do B&P foram realizados com $\Delta = 9$ e tempo limite de 1800 s.

Inicialmente, foram realizados testes para avaliar os diferentes métodos propostos e discutidos nas Seções 3 e 4. Esses testes foram realizados no nó raiz da árvore de *branch-and-price* e considerando-se apenas as instâncias do grupo 4 do STOP. Como forma de observar o desempenho comparativo entre os métodos, foram utilizados gráficos de *performance profile* [Dolan e Moré, 2002]. Um ponto no gráfico é representativo da porcentagem do total instâncias de teste em que o algoritmo gastou até τ vezes o tempo do melhor algoritmo para cada instância.

O gráfico da Figura 3 apresenta um comparativo de desempenho entre os diferentes métodos de *pricing exato*. O algoritmo de *labelling* sem técnicas de aceleração é representado por ***labelling* original**. Outrossim, denota-se como ***labelling* bidirecional** e ***labelling* com DSSR** o uso de algoritmos de *labelling* com as respectivas técnicas de aceleração. Estes testes de *labelling* com DSSR utilizaram $h = 10$.

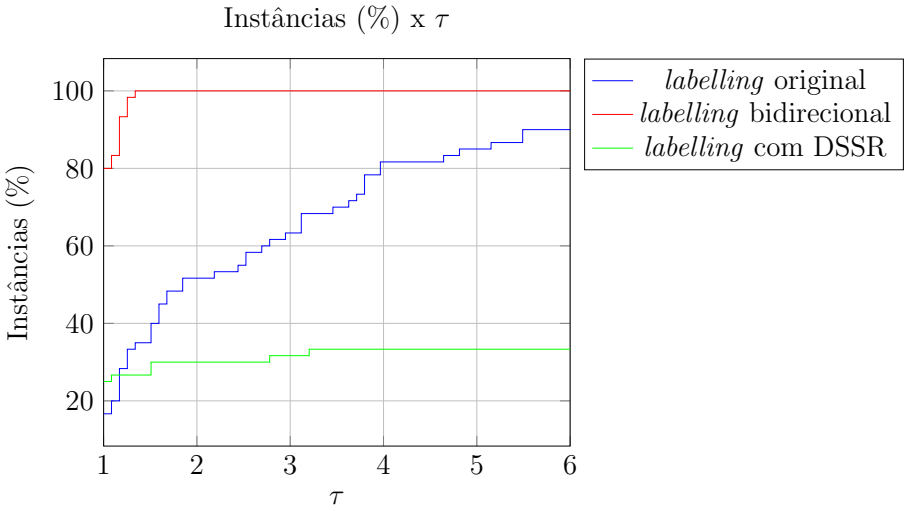


Figura 3: Performance profile de tempo gasto pelos métodos exatos no nó raiz do set 4 do STOP

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

Observando-se a Figura 3, notadamente o uso de *labelling* bidirecional tem um grande impacto no desempenho do algoritmo para resolver o PP. Ele resolve 80% das instâncias em menos tempo que as demais estratégias e 100% delas com um tempo de $1,2$ vezes da estratégia mais rápida.

O resultado comparativo entre as heurísticas de *pricing* propostas nesse trabalho está representado no gráfico de *performance profile* na Figura 4. Nessa análise, avalia-se o tempo gasto para resolver o PP no nó raiz da árvore de B&P, habilitando-se diferentes heurísticas de *pricing*, novamente considerando-se apenas as instâncias do grupo 4 do STOP. No gráfico, denomina-se como **AD** o uso da heurística *agressive dominance*. São representados por **GRASP** os testes com a metaheurística GRASP com $K = 200$. **GRASP-AD** representado os testes da metaheurística GRASP com $K = 20$, porém quando esta falha a heurística AD é chamada em seguida. **GRASPVNS** indica o uso da heurística GRASP com $K = 10$ e o procedimento de VNS para a fase de busca local. Representa-se como **SA** a metaheurística de *Simulated Annealing* com três reinicios. Para a GRASP foi utilizado o parâmetro $\alpha = 0.2$. Para o *Simulated Annealing* foram utilizados os parâmetros $\beta = 0.995$ e $T = 100000$.

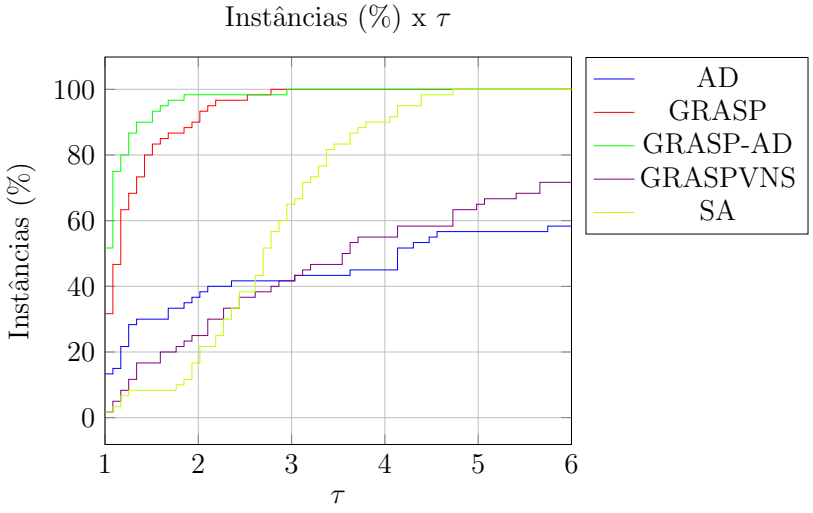


Figura 4: Performance profile de tempo gasto pelas heurísticas no nó raiz do set 4 do STOP

A heurística de *pricing* que combina o uso da heurística **GRASP** e *agressive dominance* tem um desempenho superior que as demais estratégias. Essa abordagem teve o melhor desempenho em mais de 50% das instâncias e resolveu mais de 90% das instâncias dentro de um fator de aproximadamente 1.5 da abordagem mais rápida.

Finalmente, considerando as melhores estratégias para resolver o PP, identificados nos experimentos iniciais, avaliamos o impacto dessas estratégias no desempenho do algoritmo B&P em todas as instâncias do STOP e sem impor limites no total de nós explorados na árvore de enumeração. Assim, foram realizados testes com a composição dos métodos que obtiveram melhores resultados na análise comparativa, o *labelling* bidirecional como algoritmo exato em conjunto ao GRASP-AD como heurística de *pricing*. Como forma de obter limitantes primais, foi utilizada uma heurística gulosa randomizada similar à fase construtiva do GRASP [Feo e Resende, 1995], de forma que as rotas elementares resultantes da fase de geração de colunas eram avaliadas para a composição de uma solução válida para o STOP.

Foram realizados testes com o algoritmo B&P sem nenhuma das estratégias de aceleração propostas neste trabalho e, posteriormente, com os métodos selecionados. Para os testes sem aceleração, foi utilizado $\Delta = 3$ devido ao alto custo computacional para resolver o PP. Como os

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

métodos de aceleração viabilizam valores de Δ mais elevados, foi utilizado $\Delta = 9$. Os resultados computacionais destes testes estão representados na Tabela 1.

	$\Delta = 3$				$\Delta = 9$ + métodos			
grupo	opt	t	gap	w/ub	opt	t	gap	w/ub
1(54)	54	0.45	*	0	54	0.01	*	0
1(33)	33	0.03	*	0	33	0.01	*	0
3(60)	60	1.76	*	0	60	0.21	*	0
4(60)	35	168.68	0.17	12	46	54.60	0.09	1
5(78)	75	126.03	0.16	0	74	9.71	0.11	0
6(42)	42	116.58	*	0	38	1.16	0.06	0
7(60)	39	146.10	0.10	4	52	128.39	0.05	0
total(387)	338	79.95	0.13	16	357	27.73	0.08	1

Tabela 1: Desempenho do *branch-and-price* no STOP nas diferentes configurações

A coluna **grupo** indica o grupo de instâncias e, entre parênteses é indicado o total de instâncias do grupo. A coluna **opt** representa o número de instâncias resolvidas de forma ótima. O tempo médio gasto nas instâncias resolvidas na otimalidade está indicado na coluna t. O gap médio de dualidade das instâncias com limitante primal não ótimo está indicado na coluna **gap**. O gap é calculado pela fórmula gap $= \frac{DB-LB}{DB}$, em que DB e LB são os valores dos limitantes duais e primais, respectivamente. Quando todas as instâncias foram resolvidas de forma ótima, o gap é representado por "*". O número de instâncias em que a heurística primal não teve tempo suficiente para encontrar uma solução, devido ao tempo limite ser atingido, está representado na coluna **w/ub**.

Observa-se que, com o uso do *labelling* bidirecional e da composição de heurísticas GRASP-AD, o número de instâncias resolvidas na otimalidade aumentou em 19 instâncias e o número de instâncias sem limitante primal reduziu em 11 instâncias. Além disso, o tempo médio para todas as instâncias foi reduzido em 32.42%. Com ressalva do grupo 6, concluí-se que os métodos utilizados contribuíram para a melhora de desempenho do algoritmo de B&P.

6. Conclusão

Neste trabalho, foram propostas e avaliadas diferentes abordagens de aceleração para resolver o problema de *pricing* em um algoritmo exato de B&P proposto para o STOP. Considerando-se a relaxação *ng-route* para o PP, foi avaliado o uso das técnicas de relaxação decrescente do espaço de estados e *labelling* bidirecional para a resolução do problema de *pricing* de forma exata. Além disso, foi avaliado o uso de uma heurística baseada na relaxação de regras de dominância denominada *aggressive dominance* e propôs-se o uso de duas diferentes metaheurísticas, GRASP e *Simulated Annealing*, para encontrar *ng-paths* de custo reduzido negativo. Os testes realizados indicam que a combinação do *labelling* bidirecional com as heurísticas GRASP e AD impactaram positivamente o desempenho do algoritmo utilizado. Potenciais trabalhos futuros incluem a avaliação dos métodos propostos para outras variantes do problema de orientação e de roteamento de veículos.

7. Agradecimentos

O presente trabalho foi realizado com apoio da Universidade Federal de Mato Grosso do Sul – UFMS/MEC – Brasil.

Referências

Archetti, C., Bouchard, M., e Desaulniers, G. (2011). Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science*, 45(3):285–298.

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br

Archetti, C., Carrabs, F., e Cerulli, R. (2018). The set orienteering problem. *European Journal of Operational Research*, 267(1):264–272.

Baldacci, R., Mingozzi, A., e Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *OPERATIONS RESEARCH*, p. 1269–1283.

Chao, I.-M., Golden, B. L., e Wasil, E. A. (1996). The team orienteering problem. *European journal of operational research*, 88(3):464–474.

Contardo, C. e Martinelli, R. (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129–146.

Costa, L., Contardo, C., e Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985.

Desaulniers, G., Lessard, F., e Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404.

Dolan, E. D. e Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Math. Progam*, 91:201–213.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978.

Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. d., Reis, M., Uchoa, E., e Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106:491–511.

Gamrath, G., Fischer, T., Gally, T., Gleixner, A., Hendel, G., Koch, T., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M., et al. (2016). The scip optimization suite 3.2.

Giovanni Righini, M. S. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, p. 255–273.

Irnich, S. e Villeneuve, D. (2006). The shortest-path problem with resource constraints and k-cycle elimination for k $\geq$ 3. *Informs Journal on Computing*, 18(3):391–406.

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

Laporte, G. e Martello, S. (1990). The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2):193–207. ISSN 0166-218X. URL `https://www.sciencedirect.com/science/article/pii/0166218X90901000`.

Martinelli, R., Pecin, D., e Poggi, M. (2014). Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research*, 239:102–111.

Yu, V. F., Salsabila, N. Y., Lin, S.-W., e Gunawan, A. (2023). Simulated annealing with reinforcement learning for the set team orienteering problem with time windows. *Expert Systems with Applications*, p. 121996.

DOI: 10.59254/sbpo-2024-193381
https://proceedings.science/p/193381?lang=pt-br