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

Método Exato para o Roteamento de Veículos com Empacotamento Bidimensional e Conflitos

Ana Clara Nascimento dos Santos, Pedro Henrique Del Bianco Hokama
Universidade Federal de Itajubá
Av. B P S, 1303 – Pinheirinho, Itajubá – MG
anaclarans@unifei.edu.br, hokama@unifei.edu.br

Mário César San Felice
Universidade Federal de São Carlos
Rod. Washington Luís, s/n – Monjolinho, São Carlos – SP, 13565-905
felice@ufscar.br

RESUMO

Neste artigo, apresentamos o Problema do Roteamento de Veículos Capacitados com Restrições de Carregamento Bidimensional e Conflitos, que consiste em encontrar rotas para uma frota de veículos que deve atender as demandas de um conjunto de clientes sem exceder a capacidade dos veículos. Além disso, é necessário encontrar um empacotamento bidimensional para os itens, respeitando a ordem de descarregamento e os conflitos entre eles. Apresentamos um algoritmo de *Branch-and-Cut* que utiliza um modelo em Programação Linear Inteira e propomos melhorias envolvendo quebras de simetrias e pré-processamentos. Utilizamos um modelo de Programação por Restrição para determinar se um conjunto de itens pode ser alocado em um veículo e dessa forma encontrar cortes violados. Testes foram realizados em instâncias adaptadas da literatura e constatamos que as melhorias implementadas obtiveram um impacto modesto no tempo de execução e possibilitaram ao modelo dobrar o número de soluções ótimas alcançadas.

PALAVRAS CHAVE. Roteamento, Empacotamento com Conflitos, Programação Inteira.

Tópicos (Otimização Combinatória, Logística e Transportes)

ABSTRACT

In this paper, we present the Problem of Capacitated Vehicle Routing with Two-Dimensional Loading Constraints and Conflicts, which consists in finding routes for a fleet of vehicles that must meet the demands of a set of customers without exceeding the capacity of the vehicles. In addition, it is necessary to find a two-dimensional packing for the items, respecting the unloading order and the conflicts between them. We present a Branch-and-Cut algorithm that uses an Integer Linear Programming model and propose improvements involving breaking symmetry and preprocessing. We use a Constraint Programming model to determine whether a set of items can be allocated to a vehicle and thus find violated cuts. Tests were performed on instances adapted from the literature and we found that the implemented improvements had a modest impact on the execution time, but enabled the model to double the number of optimal solutions achieved.

KEYWORDS. Routing, Packing with Conflicts, Integer Programming.

Paper topics (Combinatorial Optimization, Logistics and Transport)

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

1. Introdução

Nos setores de logística e distribuição, há uma busca constante por eficiência, a fim de reduzir custos e tornar o processo mais sustentável. Entretanto, existem situações que aumentam esse desafio, como quando as etapas de armazenamento e distribuição envolvem itens que não podem ser alocados juntos devido ao risco de contaminação ou incompatibilidade entre materiais. Neste trabalho consideramos a versão bidimensional, que ocorre quando os itens a serem transportados são frágeis, altos ou possuem um formato específico que impossibilita o empilhamento vertical.

Neste contexto, surge o Problema do Roteamento de Veículos Capacitados com Restrições de Carregamento Bidimensional e Conflitos (*Capacitated Vehicle Routing Problem with Two-dimensional Loading Constraints and Conflicts* - 2L-CVRPC), caracterizado por demandar rotas para uma frota de veículos, considerando a necessidade de atender a um conjunto específico de clientes sem ultrapassar a capacidade dos veículos. Além disso, a complexidade do problema é ampliada pela exigência de encontrar um empacotamento bidimensional viável dos itens a serem entregues, respeitando a ordem de descarregamento e considerando os conflitos entre itens.

O 2L-CVRPC é NP-difícil e vem da combinação de dois problemas de otimização. O primeiro é o clássico Problema de Roteamento de Veículos Capacitados (*Capacitated Vehicle Routing Problem* - CVRP), em que se buscam as melhores rotas para atender a uma série de clientes, de modo que a capacidade dos veículos seja respeitada. O segundo é o Problema do Empacotamento Bidimensional em Contêineres na Presença de Conflitos (*Two-dimensional Bin Packing Problem With Conflicts* - 2BPPC), em que itens retangulares de tamanhos variados devem ser alocados no menor número possível de recipientes idênticos, respeitando as restrições de conflito entre os itens.

O problema do empacotamento tem sido amplamente explorado devido a sua relevância na otimização de espaço para armazenamento e transporte de produtos. Uma de suas variações é a que considera conflitos entre os itens. Gendreau et al. [2004] estão entre os primeiros a abordar o Problema do Empacotamento Unidimensional em Contêineres na Presença de Conflitos, utilizando heurísticas baseadas no algoritmo *First Fit Decreasing* (FFD), em procedimentos de coloração de grafos, em cliques e em limitantes inferiores. Muritiba et al. [2010], ao trabalharem com o mesmo problema, empregaram uma abordagem exata baseada na formulação do problema da cobertura por conjuntos. Capua et al. [2015] também abordaram este problema, usando um método baseado na metaheurística *Iterated Local Search*, além de um algoritmo *Large Neighborhood Search*. Epstein et al. [2008], ao abordarem o 2BPPC utilizaram uma estratégia de resolução baseada em algoritmos de aproximação, centrada na coloração de grafos. Khanafer et al. [2012] também exploraram este mesmo problema, propuseram um *framework* baseado na decomposição em árvore para resolvê-lo, e apresentaram várias heurísticas para tornar o uso da decomposição eficaz.

Paralelamente, o problema de Roteamento de Veículos (VRP) também tem sido extensivamente estudado, devido a suas aplicações na logística e distribuição, onde a eficiência nas rotas é essencial para a redução de custos e tempo. Dantzig e Ramser [1959] são responsáveis por aquele que é considerado o artigo seminal do VRP. Nele abordaram o Problema do Despacho de Caminhões, que consiste em buscar um roteamento de distância mínima para uma frota de caminhões que transportam gasolina da central de distribuição para um grande número de postos a serem abastecidos. Clarke e Wright [1964] generalizaram esse problema para atender a um conjunto de clientes localizados ao redor do depósito de forma dispersa, empregando uma frota de veículos com capacidades variadas, que foi denominado Problema do Roteamento de Veículos.

A combinação do 2BPP e do VRP deu origem ao Problema de Roteamento de Veículos Capacitados com Restrições de Carregamento Bidimensionais (*Two-Dimensional Loading Capacitated Vehicle Routing Problem - 2L-CVRP*), que une a complexidade de definir rotas e alocar itens. Iori et al. [2007] introduziram o 2L-CVRP e apresentaram uma abordagem exata, que visa a

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

minimização do custo de roteamento, invocando de maneira iterativa um algoritmo de *Branch-and-Bound* para verificar a viabilidade dos carregamentos. Além de adotar heurísticas para melhoria do desempenho geral. Gendreau et al. [2008], ao trabalharem com o mesmo problema, propuseram uma *Tabu Search*, na qual a resolução do componente de carregamento do problema é abordada por meio de heurísticas, limitantes inferiores e um procedimento de *Branch-and-Bound*. Leung et al. [2011] também investigaram o 2L-CVRP e utilizaram um algoritmo que une *Tabu Search* e *Extended Guided Local Search*, além de propor novas heurísticas para o 2BPP.

Hamdi-Dhaoui et al. [2011] analisaram a variação do VRP denonimada *The Vehicle Routing Problem With Conflicts* (VRPC), que leva em consideração propriedades físicas dos itens, em especial casos de materiais perigosos, de modo que itens conflitantes não possam ser transportados no mesmo veículo. Os autores apresentaram um modelo matemático, heurísticas e meta-heurísticas, incluindo uma *Iterated Local Search* e um *Greedy Randomized Adaptive Search Procedure - Evolutionary Local Search (GRASP-ELS)*. Eles também propuseram um limitante inferior.

Abdal-Hammed et al. [2014] proposeram um algoritmo *Large Neighborhood Search* para solucionar o 2L-CVRP, que possui duas fases, a primeira gera uma ordem de posicionamento e aplica uma estratégia *bottom-left* a esta ordem para empacotar os pedidos nos veículos e uma segunda fase que encontra uma rota viável para os veículos. Wei et al. [2015] também abordaram o mesmo problema, propondo um algoritmo *Variable neighborhood search* para o roteamento e adaptando uma heurística *skyline* para examinar as restrições de empacotamento. Hokama et al. [2016] propuseram um algoritmo *Branch-and-Cut* para resolver o 2L-CVRP e o 3L-CVRP, que adota múltiplas abordagens para podar a árvore de enumeração. O empacotamento é atacado com diferentes abordagens algorítmicas, entre elas *Branch-and-Bound*, programação por restrições e meta-heurísticas. Os autores também apresentaram novas heurísticas, uma delas baseada no algoritmo Bottom-Left e a outra no Biased Random-Key Genetic Algorithm (BRKGA).

Arpini e Rosa [2017] forneceram uma extensa revisão sistemática da literatura sobre o 2L-CVRP. Ferreira et al. [2021] exploraram o 2L-CVRP e três variantes: (i) permitindo entrega fracionada (2L-SDVRP), na qual um cliente pode ser atendido por mais de um veículo; (ii) com requisitos verdes (G2L-CVRP), que leva em consideração as emissões de CO2; e (iii) a integração da entrega fracionada com requisitos verdes (G2L-SDVRP). Para cada variante, são apresentados modelos matemáticos e cada caso é solucionado com *Branch-and-Cut*. Os autores desenvolveram um procedimento personalizado para resolver o subproblema de empacotamento, incluindo limites inferiores, uma heurística de base construtiva e uma formulação em Programação por Restrições.

O presente trabalho propõe uma formulação em Programação Linear Inteira para o 2L-CVRPC e, para fortalecer esta formulação, restrições obtidas via pré-processamentos e cortes são apresentadas. Duas das famílias de restrições propostas apresentam um número exponencial de restrições, a que elimina subciclos e a que garante que os itens sejam empacotáveis, o que torna inviável a adição explícita de ambas ao modelo. Portanto, utilizamos a técnica de *Branch-and-Cut*, adicionando essas restrições apenas quando detectamos que foram violadas. Como resultado conseguimos mais que dobrar o número de soluções ótimas para alguns casos.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 apresentamos a definição formal do 2L-CVRPC, as notações que serão utilizadas e a formulação matemática do problema. A Seção 3 apresenta o Problema do Empacotamento Bidimensional com Restrições de Descarregamento (*Two-Dimensional Orthogonal Packing Problem With Unloading Constraints* - 2OPP-UL) e sua modelagem em Programação por Restrição. A seção 4 detalha as melhorias. Os resultados dos experimentos para o 2L-CVRPC com instâncias adaptadas da literatura e a conclusão são encontrados nas Seções 5 e 6, respectivamente.

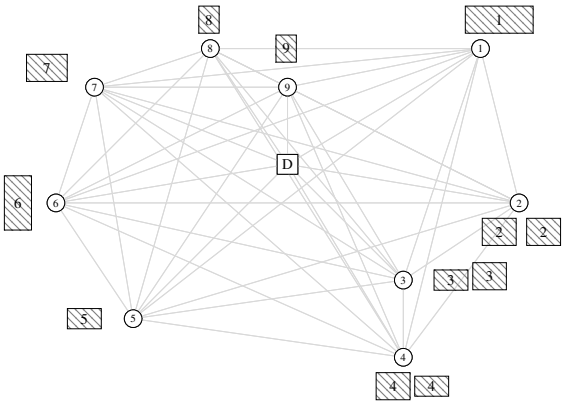
DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

2. Descrição do 2L-CVRPC e Modelagem em Programação Linear Inteira

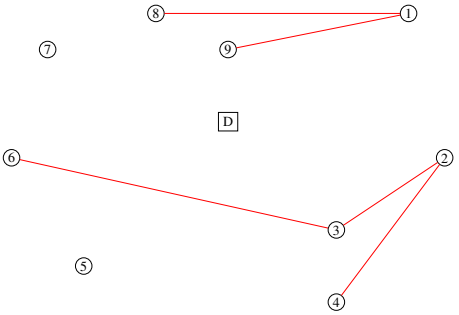
Nesta seção apresentamos a descrição formal do 2L-CVRPC, e uma modelagem em Programação Linear Inteira (PLI) para o problema, além de melhorias para fortalecer o modelo.

Definição 2.1 (2L-CVRPC) Problema do Roteamento de Veículos Capacitados com Restrições de Carregamento Bidimensional e Conflitos. *Uma instância do 2L-CVRPC é composta por: (i) Um conjunto K de veículos idênticos com compartimento de carga de largura W, altura H e peso suportado D. Os itens devem ser retirados a partir da borda superior do veículo e a remoção dos itens de um cliente não pode ser obstruída por itens a serem descarregados posteriormente. (ii) um grafo completo não direcionado $G = (V, E)$, no qual $V = \{0, 1, \ldots, n\}$ é um conjunto com $n + 1$ vértices, que correspondem ao depósito (vértice 0) e seus n clientes. Cada aresta $(i, j) \in E$ representa o caminho entre dois clientes e possui um custo associado c_{ij} que é caracterizado pela distância euclidiana entre os vértices. (iii) Cada cliente $i \in \{1, \ldots, n\}$ apresenta um conjunto I_i de itens, com peso total d_i. Cada item $p \in I_i$ tem largura w_p e altura h_p. (iv) Uma matriz Z, cujo elemento z_{ij} assume o valor 1 se e somente se, as demandas dos clientes i e j forem conflitantes. O objetivo é minimizar o custo total percorrido e uma solução deve satisfazer às seguintes restrições:*

1. *Toda rota começa e termina no depósito;*
2. *Todos os veículos devem ser utilizados;*
3. *As demandas de todos os clientes devem ser completamente atendidas;*
4. *Cada cliente deve ser visitado apenas uma vez e por apenas um veículo;*
5. *O peso suportado pelo veículo não deve ser excedido;*
6. *Clientes com itens conflitantes não devem estar na mesma rota;*
7. *Cada item deve ser alocado em exatamente um veículo;*
8. *A orientação de cada item é fixa, não são permitidas rotações;*
9. *Itens não devem se sobrepor ou exceder as bordas do veículo.*
10. *Os itens devem ser posicionados de forma que possam ser retirados movendo-se no sentido da abertura do contêiner, sem mover itens que serão entregues posteriormente.*



(a) Conjunto de 12 itens a serem entregues.



(b) Conjunto de 5 conflitos entre os clientes.

Figura 1: Exemplo de instância do 2L-CVRPC com 9 clientes.

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

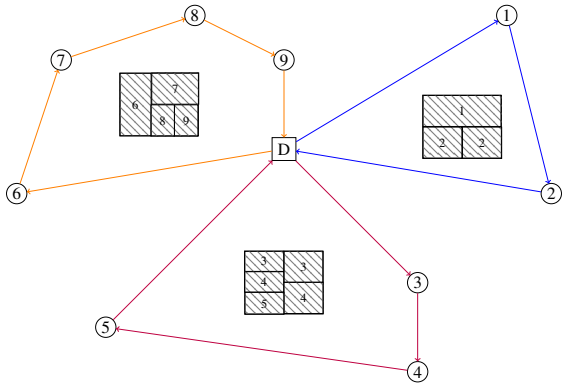


Figura 2: Exemplo de solução para a instância da Figura 1a, na qual a subrota azul representa o trajeto do veículo 1, a subrota roxa representa o trajeto do veículo 2 e a subrota laranja representa o trajeto do veículo 3. Ao centro de cada trajeto é apresentado o empacotamento dos itens no veículo.

A Figura 1 apresenta uma instância do 2L-CVRPC com $n = 9$ clientes, 3 veículos e 12 itens no total, sendo os itens representados ao lado do cliente de destino. A Figura 1b apresenta os conflitos entre os clientes: $1-8$, $1-9$, $2-3$, $2-4$, $3-6$. A solução para a instância, apresentada na Figura 2, respeita a ordem de descarregamento dos itens nas subrotas, bem como os conflitos.

2.1. Formulação Matemática

A formulação em PLI para o 2L-CVRPC é uma adaptação da formulação do VRPC [Hamdi-Dhaoui et al., 2011] com a adição de restrições que satisfaçam o empacotamento com ordem de descarregamento. Apesar do grafo de entrada não ser orientado, nossa formulação utiliza variáveis com três índices [Toth e Vigo, 2014] indicando o sentido em que uma aresta foi percorrida. Mais precisamente, temos variáveis binárias dos tipos α_{ij}^k e β_i^k. Para cada trio i, j e k, α_{ij}^k indica que o veículo k percorre a aresta $\{i, j\}$ no sentido de i para j, enquanto β_i^k indica que o vértice i é visitado pelo veículo k. Considere $\mathcal{P}$ como o conjunto das subrotas cujos itens dos clientes não conseguem ser empacotados naquela ordem.

$$\text{Minimize: } \sum_{k=1}^{|K|} \sum_{i,j \in V : i \neq j} c_{ij} \alpha_{ij}^k \tag{1}$$

$$\text{Sujeito à: } \sum_{i=1}^{n} d_i \beta_i^k \leq D, \quad \forall k \in K \tag{2}$$

$$\sum_{k=1}^{|K|} \beta_i^k = 1, \quad \forall i \in V \setminus \{0\} \tag{3}$$

$$\sum_{k=1}^{|K|} \beta_0^k = |K| \tag{4}$$

$$\sum_{i \in V : i \neq j} \alpha_{ij}^k = \beta_j^k, \quad \forall j \in V, \forall k \in K \tag{5}$$

$$\sum_{j \in V : j \neq i} \alpha_{ij}^k = \beta_i^k, \quad \forall i \in V, \forall k \in K \tag{6}$$

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

$$\sum_{i,j \in S} \alpha_{ij}^k \leq |S| - 1, \quad S \subseteq V \setminus \{0\}, 2 \leq |S| \leq n-1, \forall k \in K \tag{7}$$

$$\sum_{(i,j) \in P} \alpha_{ij}^k \leq |P| - 1, \quad P \in \mathcal{P}, k \in K \tag{8}$$

$$\sum_{j=1}^{n} z_{ij}.\beta_j^k \leq n.(1 - \beta_i^k), \quad \forall i \in V \setminus \{0\}, \forall k \in K \tag{9}$$

$$\alpha_{ij}^k \in \{0,1\}, \quad \forall i,j \in V^2, \forall k \in K \tag{10}$$

$$\beta_i^k \in \{0,1\}, \quad \forall i \in V, \forall k \in K \tag{11}$$

O objetivo definido por (1) consiste em minimizar o custo total ao atender a demanda dos clientes. As restrições (2) garantem que o peso total dos itens seja suportado pelo veículo ao qual foram atribuídos. O conjunto de restrições (3) certifica que cada cliente será atendido por exatamente um veículo. A restrição (4) assegura que serão feitas $|K|$ rotas, ou seja, toda a frota será utilizada. As restrições (5) garantem que o veículo chegue ao vértice que lhe foi atribuído, enquanto (6) garantem que ele deve deixar o vértice após a entrega. O conjunto de restrições (7) elimina os subciclos que não contêm o depósito. As restrições (8) garantem que as rotas sejam viáveis em relação ao empacotamento dos itens a ela atribuídos. As restrições (9) asseguram que itens conflitantes não sejam atribuídos à mesma rota. Por fim, as restrições (10) e (11) apontam o domínio das variáveis binárias.

É importante ressaltar que o número de subciclos é exponencial, o que impossibilita que todas as restrições (7) sejam inseridas. Recorremos então ao uso de *Lazy Constraints* para relaxar esse conjunto de restrições e adicioná-las sob demanda. De modo semelhante, após detectar que a rota passa pelo depósito, é necessário resolver um problema NP-difícil para identificar se ela pertence ao conjunto $\mathcal{P}$. Nesse ponto, surge o Problema do Empacotamento Bidimensional com Restrições de Descarregamento (2OPP-UL), que verifica se o empacotamento de um conjunto de itens é factível ou não. Para resolvê-lo, utilizamos a técnica de Programação por Restrições, abordada mais adiante na Seção 3.

2.2. Separação de Cortes

Como as restrições (7) são em número exponencial, o modelo não começa com todas elas. Temos de resolver o problema de separação, a fim de adicionar algumas destas restrições sob demanda. Para tanto, a partir de uma solução inteira do *Branch-and-Bound* realizamos uma busca em largura para identificar subciclos, que são ciclos isolados, sem o depósito. Quando encontrado um subciclo, adicionamos a restrição (7) correspondente (L1). Uma variação desse método é adicionar uma restrição correspondente para os possíveis veículos (L2). Caso contrário, se a subrota passa pelo depósito, verificamos sua viabilidade em relação ao empacotamento.

Para saber se o empacotamento é ou não viável, é necessário solucionar o 2OPP-UL. Com o objetivo de evitar chamar o *solver* para empacotamentos que já foram testados antes, armazenamos uma tabela de empacotamentos viáveis e outra de inviáveis. No caso dos empacotamentos viáveis, guardamos a chave (ordem dos nós) e o valor (coordenada x e y) de cada um deles. Para os empacotamentos inviáveis, guardamos apenas a ordem dos nós. Se o empacotamento for encontrado na tabela de viáveis, não é necessário inserir qualquer restrição. Se ele já estiver armazenado na tabela de inviáveis, uma restrição do tipo (8) é inserida. O mesmo ocorre se ele não foi encontrado em nenhuma das tabelas e a resolução do 2OPP-UL retornou que o empacotamento era inviável.

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

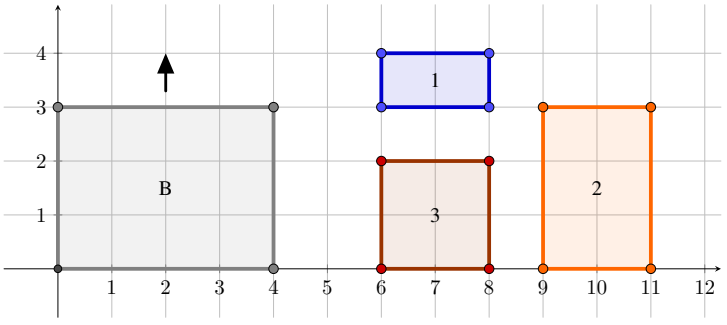
Além disso, foi proposta uma variação (S) na qual é possível considerar também as subsequências ao consultar as tabelas. Seja seq a sequência dos vértices do empacotamento atual. Se seq for subsequência de alguma das chaves da tabela de viáveis, sabemos que seq possui um empacotamento viável. De modo complementar, se alguma sequência da tabela de inviáveis é subsequência de seq, então não será necessário resolver novamente o empacotamento para seq, pois sabemos que ela é inviável.

Atribuímos as siglas L1, L2 e S às diferentes versões da *Lazy Constraints* para facilitar a identificação das mesmas na Seção 5.

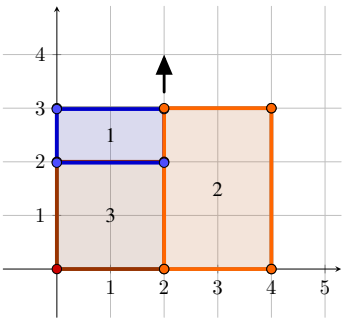
3. Descrição 2OPP-UL e Modelagem em Programação Por Restrições

Nesta seção será abordada a versão de decisão do 2OPP-UL e sua modelagem em programação por restrições.

Definição 3.1 (2OPP-UL) ***Problema do Empacotamento Bidimensional com Restrições de Descarregamento*** *Uma instância do 2OPP-UL é formada por um contêiner B de altura H e largura W, e um conjunto de itens T, no qual cada item i possui sua respectiva altura h_i e largura w_i. Em uma solução para este problema, cada item de T deve ser empacotado em B, os itens não devem se sobrepor ou exceder as bordas do contêiner e não é permitido rotacioná-los. Além disso, o empacotamento deve respeitar a ordem de descarregamento, os itens que serão retirados primeiro devem estar posicionados próximo à abertura superior de B. Sem perder a generalidade, consideramos as dimensões do contêiner e dos itens como inteiros.*



(a) Instância com contêiner B de altura 3 e largura 4 e o conjunto de 3 itens, cujos números identificam a ordem em que devem ser descarregados.



(b) Solução respeitando sentido de descarregamento.

Figura 3: Exemplo para o Problema do Empacotamento com Restrições de Descarregamento.

A Figura 3a é um exemplo de uma instância do 2OPP-UL com 3 itens. O item azul tem largura 2 e altura 1 e será descarregado primeiro, o item laranja tem largura 2 e altura 3 e será descarregado após o azul, o último a ser descarregado será o item vermelho, um quadrado com altura e largura iguais a 2. O contêiner é um retângulo com $W = 4$ e $H = 3$. Já a Figura 3b apresenta uma solução para essa instância, na qual é possível empacotar os itens com o vermelho no ponto (0, 0), o laranja à direita dele e o azul acima do vermelho. Vale notar que o empacotamento deve ser possível sem rotacionar os itens e que nesse momento não existem conflitos. E em ambas as figuras, a seta indica a direção e sentido do descarregamento.

3.1. Modelagem Matemática

Optamos por utilizar a Programação por Restrições (*Constraint Programming* - CP), método que utiliza um conjunto de variáveis e outro de restrições (condições, propriedades ou requisitos) para modelar o problema. As variáveis podem assumir diferentes valores contidos em

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

um conjunto finito, nomeado Domínio. A solução é encontrada pesquisando sistematicamente as possíveis atribuições de valores às variáveis, geralmente guiadas por heurísticas [Smith, 1995]. A solução é considerada viável quando satisfaz todas as restrições.

Na modelagem do 2OPP-UL são usadas duas variáveis para cada item i, X_i é a posição da borda esquerda do item no eixo x (direção da largura), e Y_i é a posição da borda inferior do item no eixo y (direção da altura). O domínio dessas variáveis é $D(X_i) = \{0, \ldots, W - w_i\}$ e $D(Y_i) = \{0, \ldots, H - h_i\}$ para cada i, com os valores nesses intervalos sendo discretos. Estes domínios asseguram que nenhum item ultrapasse os limites do veículo. Para garantir que os itens não se sobreponham, temos a seguinte restrição para cada par i e j de um mesmo cliente:

$$[X_i + w_i \leq X_j] \text{ ou } [X_j + w_j \leq X_i] \text{ ou } [Y_i + h_i \leq Y_j] \text{ ou } [Y_j + h_j \leq Y_i] \ . \tag{12}$$

Caso o item i pertença a um cliente que será visitado antes do cliente que receberá o item j, a seguinte restrição é adicionada para assegurar que os itens serão facilmente retirados pela abertura superior do veículo conforme a ordem de descarregamento:

$$[X_i + w_i \leq X_j] \text{ ou } [X_j + w_j \leq X_i] \text{ ou } [Y_j + h_j \leq Y_i] \ . \tag{13}$$

4. Melhorias

Foram desenvolvidas melhorias para o modelo apresentado. Ainda que essas novas restrições não sejam necessárias para o funcionamento do modelo, adicioná-las faz com que as soluções sejam encontradas mais rapidamente. Cada melhoria recebeu uma sigla entre parênteses após o seu nome para que possa ser identificada com mais facilidade na Seção 5.

Veículos Consecutivos (V): É uma quebra de simetrias para aumentar eficiência, em que restringimos as opções de veículo pelo qual cada um dos $|K|$ primeiros clientes pode ser atendido. Dessa forma o cliente $i \in \{1, \ldots, |K|\}$ deverá estar em um dos i primeiros veículos.

$$\sum_{k=0}^{i-1} \beta_i^k = 1 \qquad i \in \{1, \ldots, |K|\} \tag{14}$$

Adicionando Conflitos por Cliques (C): Considere o grafo $G_c = (V, E_c)$, no qual os vértices são os clientes e as arestas representam os conflitos entre eles. Seja C uma clique em G_c, no 2L-CVRPC ela representa um conjunto de clientes que devem ser atendidos por $|C|$ veículos distintos, já que possuem conflitos com todos os outros clientes que também compõem a clique. Portanto, são inseridas as restrições de que itens da mesma clique não podem ser transportados no mesmo veículo, (15), fortalecendo o modelo. Para essa abordagem, foi utilizada a biblioteca Cliquer [Niskanen, 2002] adaptada para C++, que apresenta funções para a busca de Cliques Maximais em grafos. Por meio dessas funções, as cliques são encontradas e adicionadas a uma coleção de cliques $\mathcal{C}$.

$$\sum_{i \in C} \beta_i^k \leq 1 \qquad k \in K, C \in \mathcal{C} \tag{15}$$

5. Resultados Computacionais

Todos os modelos e algoritmos descritos foram implementados em C++ e compilados com g++ 9.4 em ambiente Linux utilizando um computador com processador de 2.30GHz e 8GB de memória RAM. O resolvedor de PLI foi o IBM Cplex 22.1 e o resolvedor de CP foi o IBM CP Optimizer 22.1, ambos do mesmo pacote.

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

Os testes foram executados nas instâncias apresentadas por Iori et al. [2007] para o problema do roteamento de veículos com restrições de carregamento bidimensional. Nelas são consideradas as classes $r = 2, \ldots, 5$, e o número de itens gerados para cada cliente é um valor inteiro uniformemente aleatório no intervalo $[1, r]$. Foram selecionadas 95 instâncias, com base no número de clientes, valor que se correlaciona com o nível de dificuldade do problema do roteamento. Optamos por realizar os testes com as instâncias que continham até 50 clientes.

Para cada instância, uma matriz simétrica de conflitos foi gerada entre os nós (clientes), onde a presença de um conflito é determinada aleatoriamente com base na probabilidade ψ fornecida, i.e., cada par de nós é avaliado uma vez de modo independente, sendo registrado um conflito entre eles na matriz com probabilidade ψ. Conjecturamos que a presença de conflitos contribui para a complexidade do problema, de modo a torná-lo mais difícil. Porém, parece razoável que uma quantidade excessiva de conflitos limita tanto as escolhas, a ponto de eliminar soluções viáveis e tornar o problema mais fácil. Para verificar essas hipóteses, testamos nossos algoritmos com conjuntos de instâncias que apresentam diferentes probabilidades ψ de conflitos: 5%, 10% e 20%.

A partir do modelo (1)-(11) do 2L-CVRPC foram implementas as versões de *Lazy Constraints* L1, L2 e S (descritas na Seção 2.2), bem como as melhorias V e C (descritas na Seção 4). Optamos por agrupar essas versões e melhorias em configurações incrementais, verificando o tempo em que cada uma consegue resolver as instâncias. Por exemplo, enquanto L1 corresponde à implementação do modelo (1)-(11) apenas com a primeira variação da Lazy Constraint, L1VSC indica que foram adicionadas as melhorias V, S e C.

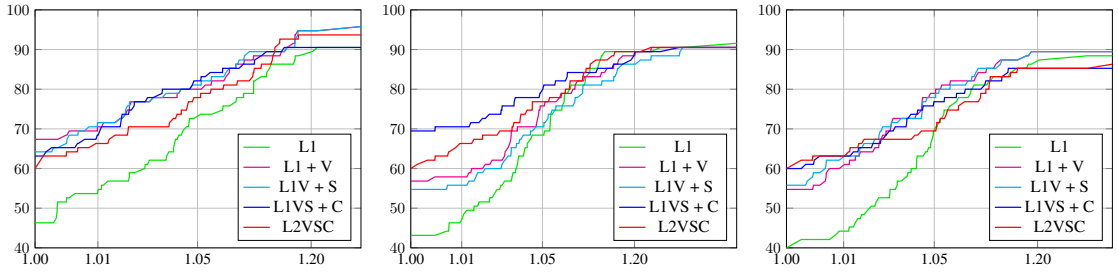
Os resultados dos testes são apresentados por meio de gráficos de perfis de desempenho (*performance profiles*) [Dolan e Moré, 2002], utilizados para comparar o desempenho de vários algoritmos sobre um conjunto de instâncias. Dada uma instância, temos a solução de cada algoritmo e a melhor solução obtida entre eles. Assim, para cada algoritmo podemos calcular a razão entre sua solução e a melhor obtida, o que chamamos de *performance gap*. O eixo y do gráfico representa a porcentagem das instâncias, enquanto o eixo x corresponde ao *performance gap*. Nos gráficos abaixo, cada curva representa uma configuração do nosso algoritmo, identificada pelas siglas das melhorias e variações de parâmetro que a compõem. Em perfis de desempenho, consideramos que a curva com uma maior área sob ela fornece os melhores resultados.

Vale ressaltar que, independente das melhorias, com tempo suficiente é possível atingir uma solução ótima. Contudo, o algoritmo para solucionar o empacotamento pode ser chamado diversas vezes durante a solução de uma instância. Por isso, limitamos a 60 segundos o tempo de execução do *solver* para cada empacotamento, de forma que não sendo encontrado um empacotamento nesse tempo ele é considerado inviável. Quando esse tempo é excedido, não garantimos a solução ótima, mas é possível identificar os casos em que isso ocorre durante a execução. Se desejável, pode-se não limitar o tempo do empacotamento para garantir soluções ótimas. Além disso, o tempo total de execução do *solver* para o 2L-CVRPC foi limitado a 10 minutos.

A Figura 4a apresenta a comparação entre as variações do algoritmo para instâncias com 5% de chance de conflito. Nela é possível notar que a configuração L1+V obteve as melhores soluções para um pouco menos de 70% das instâncias. No entanto, a variação L1V + S obteve melhor desempenho geral. A Figura 4b monstra que para as instâncias com 10% de conflitos inicialmente L1VS + C está acima das demais variações, com as melhores soluções para aproximadamente 70% das instancias. A Figura 4c apresenta a comparação das configurações do algoritmo para 20% de conflitos . É possível observar que a configuração L2VSC inicialmente obteve as melhores soluções para cerca de 60% das instâncias. Entretanto, a variação do algoritmo com melhor performance para esse tipo de instância foi L1V + S.

Por fim, as Tabelas 1, 2 e 3 mostram os tempos médios de resolução das instâncias por

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br



(a) 5% de conflito. (b) 10% de conflito. (c) 20% de conflito.

Figura 4: Comparação do impacto de cada variação do algoritmo de acordo com quantidade de conflitos. O eixo x está em escala logarítmica.

		L1			L1 + V			L1V + S			L1VS + C			L2VSC		
Nº de Clientes	Freq.	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O
15	10	331.1471	0.0240	7	101.5001	0	10	98.2511	0	10	60.8627	0	10	98.4405	0	10
20	10	599.8310	0.2377	0	599.9343	0.2075	0	599.8813	0.2039	0	586.3614	0.1313	1	562.8862	0.1147	2
21	10	603.9915	0.2267	0	511.4724	0.1167	2	511.6644	0.1149	2	488.9257	0.1765	3	457.3420	0.1155	4
22	10	513.5859	0.2377	2	472.2166	0.1431	3	462.5440	0.1364	3	498.5674	0.1143	2	435.1564	0.1418	3
25	5	599.8284	0.4367	0	599.8290	0.3755	0	599.8128	0.3782	0	610.6572	0.3921	0	605.6800	0.3968	0
29	10	614.7983	0.8255	0	613.3976	0.7586	0	613.6164	0.7586	0	628.1025	0.8023	0	608.2360	0.7820	0
30	5	599.6054	0.6734	0	599.8174	0.4457	0	599.8150	0.4416	0	599.8172	0.5299	0	599.9470	0.4160	0
32	15	608.0855	0.6846	0	600.2618	0.7469	1	589.6103	0.7469	1	611.6827	0.7375	1	620.1259	0.7533	0
35	5	599.8896	0.5567	0	599.9032	0.4443	0	599.8482	0.4434	0	599.8636	0.4240	0	599.8594	0.5348	0
40	5	599.9438	0.6866	0	599.9238	0.4226	0	599.8748	0.4262	0	599.8700	0.6541	0	599.8426	0.5391	0
44	5	609.4168	0.8972	0	606.2882	0.9028	0	600.5114	0.9028	0	641.6902	0.9035	0	619.2332	0.8913	0
50	5	602.4476	0.8827	0	610.0710	0.7786	0	607.7836	0.7786	0	648.4444	0.8863	0	601.9782	0.8788	0
Total		566.42608	0.48897	9	527.03474	0.42431	16	523.59953	0.42312	16	529.52790	0.44481	17	516.37054	0.43289	19

Tabela 1: Instâncias com 5% de conflito.

		L1			L1 + V			L1V + S			L1VS + C			L2VSC		
Nº de Clientes	Freq.	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O
15	10	392.3338	0.0356	6	33.2287	0	10	35.3723	0	10	47.9242	0	10	54.4216	0	10
20	10	600.9447	0.2489	0	546.9822	0.1090	2	556.8915	0.1168	1	554.7576	0.1074	2	531.2450	0.1265	3
21	10	599.8443	0.2634	0	547.7539	0.1306	2	553.3225	0.1475	2	592.8609	0.1330	1	599.8161	0.1244	0
22	10	562.9176	0.1859	1	442.5485	0.1497	3	434.7599	0.1618	3	460.5798	0.1254	3	470.1541	0.1487	3
25	5	599.8660	0.3854	0	599.5652	0.3681	0	599.7970	0.3837	0	599.8504	0.3431	0	599.7966	0.2894	0
29	10	554.8653	0.6935	1	555.7751	0.7289	1	552.3565	0.7333	1	555.7283	0.6977	1	552.4993	0.6883	1
30	5	600.1712	0.6779	0	599.8750	0.5629	0	600.1292	0.5674	0	599.8638	0.4344	0	599.8802	0.5738	0
32	15	621.3685	0.7189	0	573.1391	0.6673	1	571.7273	0.6715	1	572.6989	0.6583	1	583.7232	0.6568	1
35	5	610.3860	0.5402	0	599.9404	0.6590	0	599.9096	0.6645	0	602.1816	0.5697	0	599.9072	0.6664	0
40	5	600.6300	0.5533	0	599.8782	0.4426	0	602.0546	0.4457	0	599.8956	0.7877	0	599.8782	0.6531	0
44	5	600.1628	0.8347	0	651.1200	0.8828	0	604.3222	0.8839	0	610.2218	0.8855	0	606.5514	0.8185	0
50	5	619.3536	0.8937	0	615.9080	0.8942	0	625.1730	0.8947	0	611.1010	0.8846	0	607.4020	0.8834	0
Total		574.55191	0.46824	8	507.27795	0.42356	19	505.89330	0.43017	18	513.94276	0.42141	18	514.78198	0.42267	18

Tabela 2: Instâncias com 10% de conflito.

número de clientes. Apresentamos nestas tabelas a quantidade de instâncias por número de clientes "Freq.", a comparação dos tempos médios "T(s)", MIP Gap médio e número de soluções ótimas encontradas "O". O MIP gap é dado por $\frac{lp-ld}{lp}$, onde lp é o limitante primal e ld é o limitante dual. Quando a inviabilidade de uma instância é comprovada, fazemos Mip gap igual a 0. Caso uma solução viável não seja encontrada, consideramos o Mip gap como 1.

Na Tabela 1 podemos notar que, adicionando as melhorias encontramos soluções ótimas para instâncias em que anteriormente o ótimo não havia sido atingido no limite de tempo. Para instâncias com mais de 32 clientes, a dificuldade aumenta significativamente, como evidenciado pela ausência de soluções ótimas. Na Tabela 2 obtivemos o maior número de soluções ótimas na configuração L1 + V, além de redução de tempo médio para a maioria das categorias de cliente. Por último, na Tabela 3 podemos notar que, como conjecturado, foi possível resolver de modo ótimo

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

		L1			L1 + V			L1V + S			L1VS + C			L2VSC		
Nº de Clientes	Freq.	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O	T(s)	MIP Gap	O
15	10	268.2909	0.0184	8	18.3814	0	10	12.2314	0	10	10.2004	0	10	8.8938	0.0000	10
20	10	599.8118	0.2363	0	524.4688	0.1182	3	524.0891	0.1178	3	452.5543	0.1093	3	471.9124	0.1114	3
21	10	599.8198	0.3003	0	423.5886	0.0726	5	419.2479	0.0697	5	418.7724	0.0882	4	407.7764	0.0929	5
22	10	546.2055	0.1989	1	476.8498	0.0945	3	443.3649	0.0864	4	441.9067	0.0947	4	434.0243	0.1082	4
25	5	599.8184	0.4029	0	603.7512	0.3883	0	599.1742	0.3862	0	599.7508	0.3674	0	597.1474	0.3492	0
29	10	543.5316	0.6752	1	544.1061	0.6760	1	547.8203	0.6760	1	559.5059	0.5649	1	628.5426	0.6759	1
30	5	599.8452	0.5650	0	602.4672	0.5683	0	599.7942	0.5677	0	611.6638	0.5284	0	599.4386	0.5437	0
32	15	565.7979	0.5776	1	566.3696	0.4521	1	567.7471	0.4506	1	569.2544	0.5374	1	568.4584	0.6441	1
35	5	599.8558	0.5665	0	599.7854	0.5401	0	599.8474	0.5453	0	599.8644	0.6542	0	599.8164	0.5497	0
40	5	599.9134	0.6788	0	599.8250	0.5671	0	599.8798	0.5671	0	600.0000	1.0000	0	599.7980	0.4510	0
44	5	560.0536	0.7297	1	482.4860	0.8000	1	482.4020	0.8000	1	481.7938	0.7128	1	484.2106	0.7234	1
50	5	615.7620	0.9083	0	599.9282	0.9005	0	599.9640	0.9005	0	613.2106	0.8978	0	599.9800	0.8910	0
Total		546.73480	0.44434	12	482.21795	0.37070	24	477.77948	0.36938	25	472.62769	0.39405	24	478.31922	0.39036	25

Tabela 3: Instâncias com 20% de conflito.

mais instâncias com alta probabilidade de conflitos, inclusive para uma instância com 44 clientes.

No geral, as melhorias colaboraram para uma leve redução no tempo de execução, e possibilitaram a ampliação do número de soluções ótimas encontradas. Além disso, comparando os resultados para instâncias com 20% de conflitos em contraste com aquelas com 10% ou 5%, podemos observar que para as primeiras foi obtido maior número de soluções ótimas em menor tempo. Isto corrobora nossa hipótese de que muitos conflitos acabam simplificando o problema ao eliminar soluções viáveis. Por outro lado, não observamos diferenças significativas no tempo ou qualidade das soluções obtidas ao variar os conflitos entre 5% e 10%. Parece interessante em trabalhos futuros testar instâncias com menos conflitos, para avaliar o impacto nos resultados.

6. Conclusões

Neste artigo abordamos o 2L-CVRPC que combina o problema do roteamento de veículos com o problema do empacotamento bidimensional com conflitos. Apresentamos um modelo em PLI para o problema que possui um número exponencial de restrições, aquelas que eliminam subciclos e conjuntos de itens que não são empacotáveis em um mesmo veículo. Por esse motivo, não adicionamos essas restrições *a priori*, mas apenas quando detectamos que uma delas foi violada durante o *Branch-and-Cut*. Notadamente, descobrir se um conjunto de itens possui um empacotamento em um contêiner é um problema NP-Difícil. Para resolver o problema do empacotamento utilizamos uma formulação em Programação por Restrições. Para a Formulação em PLI foram apresentadas duas melhorias envolvendo pré-processamentos e quebras de simetria, que, juntamente com as variações da *Lazy Constraint*, possibilitaram o aumento no número de soluções ótimas encontradas.

Nos próximos passos desta pesquisa pretendemos melhorar o desempenho do algoritmo apresentado, por meio de novas abordagens, como heurísticas de inicialização, novas rotinas de separação e outras formas de quebra de simetria.

Referências

Abdal-Hammed, M. K., Hifi, M., e Wu, L. (2014). Large neighborhood search for the vehicle routing problem with two-dimensional loading constraints. In *2014 International Conference on Control, Decision and Information Technologies (CoDIT)*, p. 054–059. IEEE.

Arpini, B. P. e Rosa, R. A. (2017). Uma revisão sistemática da literatura sobre o problema de roteirização de veículos capacitados com restrições de carregamento bidimensional (2l-cvrp). *TRANSPORTES*, 25(1):61–72.

Capua, R., Frota, Y., Vidal, T., e Ochi, L. S. (2015). Um algoritmo heurístico para o problema de bin packing com conflitos. *Anais do XLVII Simpósio Brasileiro de Pesquisa Operacional*.

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br

Clarke, G. e Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581.

Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1): 80–91.

Dolan, E. D. e Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213.

Epstein, L., Levin, A., e Van Stee, R. (2008). Two-dimensional packing with conflicts. *Acta Informatica*, 45:155–175.

Ferreira, K. M., de Queiroz, T. A., e Toledo, F. M. B. (2021). An exact approach for the green vehicle routing problem with two-dimensional loading constraints and split delivery. *Computers & Operations Research*, 136:105452.

Gendreau, M., Iori, M., Laporte, G., e Martello, S. (2008). A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks: An International Journal*, 51(1):4–18.

Gendreau, M., Laporte, G., e Semet, F. (2004). Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research*, 31(3):347–358.

Hamdi-Dhaoui, K., Labadie, N., e Yalaoui, A. (2011). The vehicle routing problem with conflicts. *IFAC Proceedings Volumes*, 44(1):9799–9804.

Hokama, P., Miyazawa, F. K., e Xavier, E. C. (2016). A branch-and-cut approach for the vehicle routing problem with loading constraints. *Expert Systems with Applications*, 47:1–13.

Iori, M., Salazar-González, J.-J., e Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation science*, 41(2):253–264.

Khanafer, A., Clautiaux, F., e Talbi, E.-G. (2012). Tree-decomposition based heuristics for the two-dimensional bin packing problem with conflicts. *Computers & Operations Research*, 39(1): 54–63.

Leung, S. C., Zhou, X., Zhang, D., e Zheng, J. (2011). Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 38(1):205–215.

Muritiba, A. E. F., Iori, M., Malaguti, E., e Toth, P. (2010). Algorithms for the bin packing problem with conflicts. *Informs Journal on computing*, 22(3):401–415.

Niskanen, S. (2002). Cliquer - routines for clique searching. URL `https://users.aalto.fi/~pat/cliquer.html`. Acesso em: 09/09/2022.

Smith, B. M. (1995). A tutorial on constraint programming. Technical Report 95.14, University of Leeds, School of Computer Studies.

Toth, P. e Vigo, D. (2014). Introduction to vehicle routing problems. In *Vehicle routing: problems, methods, and applications*, chapter 1, p. 1–23. SIAM.

Wei, L., Zhang, Z., Zhang, D., e Lim, A. (2015). A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 243(3):798–814.

DOI: 10.59254/sbpo-2024-193481
https://proceedings.science/p/193481?lang=pt-br