Pesquisa Operacional para o Desenvolvimento
Vol. 18, e18004, p. 1-18, 2024

ISSN 1984-3534

MINIMIZAÇÃO DO COMPRIMENTO DE CICLO EM MÁQUINAS DE CORTE DE VIDRO[1]

Ernée Kozyreff Filho[a] *, **Silvio Alexandre de Araujo**[b]

[a]Instituto de Ciências e Engenharia, Departamento de Engenharia
Universidade Estadual Paulista (Unesp), Itapeva, SP, Brazil

[b]Instituto de Biociências, Letras e Ciências Exatas, Departamento de Matemática
Universidade Estadual Paulista (Unesp), São José do Rio Preto, SP, Brazil

Recebido 05/05/2023, aceito 03/07/2024

RESUMO

Estudamos o problema de determinar um ciclo de comprimento mínimo para uma máquina de corte de vidro usada para criar vincos verticais e horizontais em chapas de vidro retangulares. Mostramos que o problema pode ser modelado como um Problema do Caixeiro Viajante (*Traveling Salesman Problem*-TSP) e como um Problema do Carteiro Rural (*Rural Postman Problem*-RPP) usando o mesmo grafo. Algumas características específicas do problema tornam a formulação usando a abordagem RPP equivalente à formulação TSP se um certo tipo de restrição de eliminação de sub-rotas for usado. Experimentos computacionais realizados em um grande conjunto de instâncias indicam que o tempo necessário para resolver uma instância está relacionado com o tamanho do grafo e com a paridade do número de vincos verticais e horizontais no padrão de corte.

Palavras-chave: Otimização inteira-mista, Corte de vidro, Indústria automotiva.

ABSTRACT

We study the problem of determining a minimum length cycle for a glass cutting machine used to create vertical and horizontal scores on rectangular glass plates. We show that the problem can be modeled as a Traveling Salesman Problem (TSP) and as a Rural Postman Problem (RPP) using the same graph. Some specific features of the problem make the formulation using the RPP approach equivalent to the TSP formulation if a certain type of subtour elimination constraint is used. Computational results conducted on a large set of instances indicate that the time necessary to solve an instance is related to the size of the graph and and the parity of the number of vertical and horizontal lines in the cutting pattern.

Keywords: Mixed-integer optimization, Glass cutting, Automotive industry.

* Autor para correspondência. E-mail: ekozyreff@gmail.com
DOI: https://doi.org/10.4322/PODes.2024.006

[1]*Todos os autores assumem a responsabilidade pelo conteúdo do artigo.*

1. Introdução

O Problema de Determinação do Caminho de Corte (*Cutting Path Determination Problem*-CPDP) é o problema de encontrar a melhor sequência de movimentos para uma ferramenta ou uma máquina usada para cortar peças menores de uma chapa grande de matéria-prima. O objetivo é minimizar a distância total percorrida pela ferramenta em um ciclo de corte. O CPDP surge em uma variedade de ambientes industriais, como têxtil, metal, plástico e vidro. Na produção de vidros para a indústria automotiva, o problema ocorre em uma das primeiras etapas do processo, quando uma máquina de corte é utilizada para criar vincos na superfície de uma chapa retangular de vidro para produzir peças menores. Neste artigo, abordamos o problema de encontrar um ciclo com comprimento mínimo para o cabeçote de uma máquina de corte de vidro.

A maioria dos trabalhos relacionados a máquinas de corte na indústria de vidro considera o problema de posicionamento de peças que precisam ser cortadas em uma chapa com o objetivo de minimizar o desperdício de material, conhecido como problema de Corte e Empacotamento (*Cutting and Packing*-C&P) (Wäscher et al., 2007). A disposição geométrica das peças na chapa é chamada de padrão de corte, e o desenho de bons padrões de corte bidimensionais é fundamental para reduzir custos no processo. No entanto, questões relacionadas ao processo de corte nem sempre são consideradas durante a criação dos padrões de corte.

Apesar de encontrarmos diversos trabalhos aplicando o CPDP em diferentes situações reais, principalmente focadas no corte de chapas metálicas (Moreira et al., 2007; Dewil et al., 2011, 2014), não encontramos trabalhos que considerem o CPDP em contexto da fabricação de vidros automotivos, em que as chapas são cortadas em peças retangulares menores usando uma máquina controlada por computador numérico (*Computer Numeric Control*-CNC). Para ajudar a preencher essa lacuna, abordamos o CPDP aplicado a um determinado tipo de padrão de corte muito utilizado nesse cenário, que possui apenas linhas verticais e horizontais atravessando a chapa de borda a borda, usualmente chamado de padrão quadriculado. As principais contribuições deste trabalho residem na adaptação de modelos matemáticos para representar o problema prático estudado, na exploração de características específicas do problema que tornam a formulação usando a abordagem do Problema do Carteiro Rural (*Rural Postman Problem*-RPP) equivalente à formulação do Problema do Caixeiro Viajante (*Traveling Salesman Problem*-TSP), e na apresentação de resultados computacionais usando conjuntos de dados baseados na prática, que indicam que o tempo necessário para resolver uma instância está relacionado ao tamanho do grafo e à paridade do número de linhas verticais e horizontais no padrão de corte, isto é, se esse número é par ou ímpar.

Este artigo está dividido em seis seções, incluindo esta introdução (Seção 1). Na Seção 2, é apresentada uma revisão da literatura. Na Seção 3, o processo de produção de vidros automotivos é descrito com foco no problema considerado neste trabalho. Na Seção 4, três modelos matemáticos são propostos, sendo dois com base no TSP e um com base no RPP. Os experimentos computacionais são apresentados na Seção 5 e as conclusões, na Seção 6.

2. Revisão da Literatura

Nesta seção apresentamos uma revisão da literatura de artigos relevantes que estudam o CPDP e também alguns artigos que desenvolvem ferramentas de otimização aplicadas a decisões sobre corte de vidro.

Em relação ao CPDP e aplicações gerais, Manber e Israni (1984) consideram a minimização do número de pontos de perfuração ao cortar formas irregulares de uma chapa. Os autores desenvolvem três algoritmos de acordo com a característica do grafo associado. Williams e Gupta (1988) examinam o CPDP com barreiras entre nós e Williams e Lee (1993) estendem o problema para o caso tridimensional. Hoeft e Palekar (1997) estudam o problema de minimizar o ciclo de corte de formas poligonais usando um modelo TSP e uma heurística de decomposição

de Lagrange. Han e Na (1999) consideram uma aplicação em um processo de corte CNC e propõem uma heurística de recozimento simulado para minimizar o comprimento do caminho sem superaquecer o material. Castelino et al. (2003) apresentam algoritmos para redução do tempo não produtivo (ou "tempo de ar") de uma ferramenta utilizada para fresamento. Os autores formulam o problema como um TSP generalizado com restrições de precedência. Wang e Xie (2005) desenvolvem algoritmos de otimização de colônias de formigas para uma máquina perfuradora para determinar o tipo e a sequência de operações aplicadas para maximizar a eficiência da fabricação.

Ainda relacionado ao CPDP, Lee e Kwon (2006) consideram o problema em um processo de corte CNC para minimizar a distância percorrida pela ferramenta de corte. Os autores formulam o problema como um TSP generalizado e desenvolvem um algoritmo genético para otimizar as localizações dos pontos de perfuração e a sequência de corte. Moreira et al. (2007) estudam um RPP "dinâmico", no qual uma chapa de metal é suspensa e novas possibilidades de caminhos de corte são criadas à medida em que as peças se desprendem da chapa ao serem cortadas. Duas novas heurísticas são propostas para resolver o problema. Imahori et al. (2008), considerando uma aplicação em materiais duros como rochas e pedras, desenvolvem um procedimento heurístico em que várias restrições específicas do setor são consideradas. Rodrigues e Ferreira (2012) também apresentam abordagens heurísticas, baseadas em algoritmos meméticos, para o CPDP, considerando aplicações gerais em que o processo de corte é contínuo, ou seja, a ferramenta de corte nunca deixa a superfície. Dewil et al. (2011) propõem um modelo de programação inteira e várias heurísticas baseadas em busca tabu para minimizar a distância total percorrida por uma ferramenta de corte de chapa metálica. Dewil et al. (2014) também consideram uma aplicação na indústria metalúrgica e apresentam um modelo matemático baseado em um TSP generalizado, bem como um conjunto de heurísticas construtivas.

Mais recentemente, Silva et al. (2019) utilizam duas abordagens para o CPDP: uma baseada no RPP e outra baseada no TSP. Nos resultados computacionais apresentados, o modelo RPP obtém melhor desempenho na maioria dos casos. Oliveira et al. (2020) propõem um modelo matemático para o problema de empacotamento irregular de tiras e para o CPDP. Os autores apresentam resultados computacionais que indicam vantagens em se considerar um problema integrado com dois objetivos conflitantes: minimizar o desperdício de material e a distância percorrida pela máquina de corte.

Foram encontrados vários estudos que tratam de diferentes problemas de otimização relacionados à indústria de vidro, mas, conforme destacado anteriormente, não encontramos nenhum artigo sobre CPDP aplicado a esse tipo de indústria. Dyson e Gregory (1974) e Madsen (1988) enfocam o problema de corte de estoque que visa minimizar o desperdício de material e otimizar a sequência de padrões de corte. Chambers e Dyson (1976) consideram o problema de determinar tamanhos de estoque ótimos. Nicholls (1993) considera todo o conjunto de fabricantes de vidro da Austrália e propõe um modelo matemático considerando decisões sobre produção, distribuição e armazenamento. Al-Khayyal et al. (2001) consideram um problema de empilhamento e escalonamento em um ambiente de produção de vidro e fornecem um procedimento de resolução baseado em decomposição. Puchinger et al. (2004) descrevem diferentes algoritmos heurísticos para o problema de corte de estoque na fabricação de vidro com restrições na disponibilidade de docas de carregamento. As heurísticas consideradas no artigo são: heurística gulosa, heurísticas baseadas em branch-and-bound e heurísticas evolutivas.

Ainda considerando aplicações na indústria de vidros, Arbib e Marinelli (2007) lidam com o problema de minimização de perda de bordas aplicado ao corte de vidro em uma fábrica italiana de vidro automotivo. Os autores consideram duas fases da produção de vidro e apresentam um algoritmo heurístico baseado num modelo matemático para o problema de p-mediana, bem como resultados computacionais baseados em instâncias reais. Na et al. (2013) desenvolvem métodos de solução baseados em heurísticas de construção seguidas de busca local para minimizar o desperdício de material em linhas automatizadas de produção de vidro e apresentam resultados

computacionais baseados em instâncias reais. Na et al. (2014) propõem um algoritmo heurístico construtivo e analisam a complexidade do problema relacionado ao desperdício no tempo de ciclo. Em Parreño et al. (2020), os autores consideram um processo de corte guilhotinado em três estágios. Um procedimento heurístico de busca em feixe é desenvolvido e um estudo computacional com conjuntos de dados reais é apresentado.

Existem outros artigos aplicados à indústria vidreira, mas não diretamente relacionados a máquinas de corte, tais como: Alvarez-Valdes et al. (2005), que consideram o problema de job-shop flexível; Chambers e Dyson (1976), que abordam o problema de seleção de tamanhos de estoque; Lee e Lee (2020), que estudam o problema de dimensionamento e sequenciamento de lotes; Lin (2018), que examina o problema de desenvolvimento de produto; e Taskin e Ünal (2009), que investiga o problema de planejamento tático.

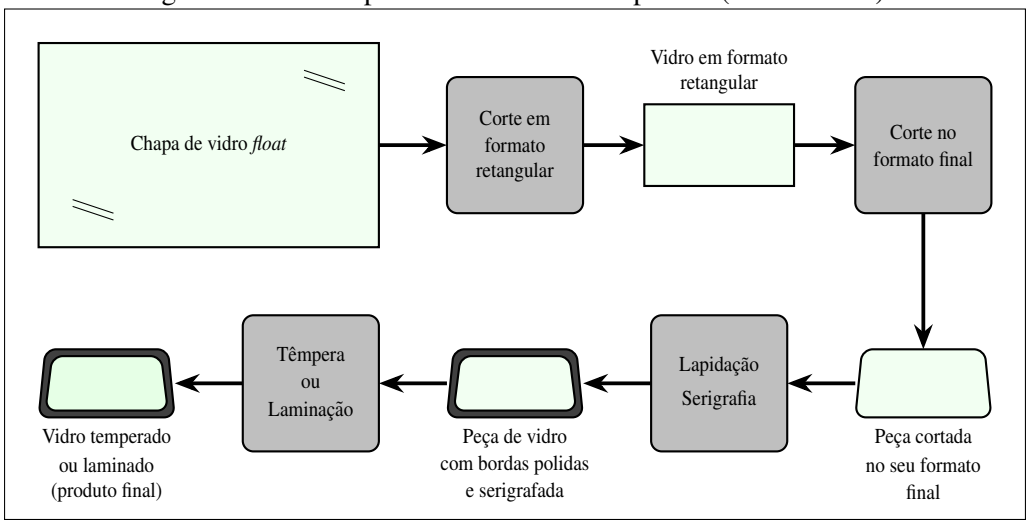
3. Definição do Problema

O vidro utilizado como matéria-prima para a indústria automotiva é produzido por meio de um processo que envolve um banho de estanho no qual o vidro fundido flutua e esfria durante a etapa de recozimento. Por isso, é chamado de vidro *float*. Este tipo de vidro tem propriedades importantes para a indústria vidreira: é plano e sem distorções, tem espessura uniforme e baixíssima incidência de bolhas ou outros materiais estranhos. Por outro lado, o vidro *float* não é considerado seguro para aplicação imediata porque, em caso de quebra, as peças resultantes tendem a ser objetos pontiagudos com arestas vivas. Por esse motivo, o vidro precisa ser *temperado* ou *laminado* antes de ser incorporado ao veículo.

O temperamento de uma peça de vidro *float* consiste em aquecê-la a aproximadamente 600 ^{o}C e resfriá-la logo em seguida com ar de alta pressão. Esse procedimento cria uma diferença de tensões entre as camadas interna e externa do vidro, aumentando sua resistência mecânica. Além disso, se o vidro quebrar, ele se estilhaçará em pequenos fragmentos pouco cortantes.

A laminação envolve a colagem de duas folhas de vidro com um filme plástico entre elas. O produto resultante tem a importante propriedade de segurança de permanecer como uma peça única, retida pelo filme, em caso de quebra. O vidro laminado é amplamente utilizado em pára-brisas de veículos, também porque o filme plástico ajuda a bloquear a radiação ultravioleta.

Figura 1: Processo produtivo do vidro temperado (ou laminado).



Fonte: Elaborada pelos autores.

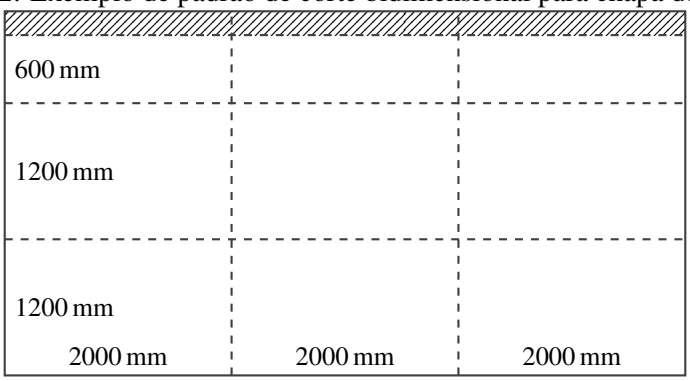
A produção de vidros temperados e laminados começa de forma semelhante, mas difere em suas etapas finais. A Figura 1 ilustra ambos os processos. Eles iniciam com uma grande chapa retangular de vidro *float* cortada em partes menores (geralmente também retangulares). Estas

peças seguem então para outro posto de trabalho onde são novamente cortadas, mas já na sua forma final. As próximas etapas dos processos são a lapidação (para polir as arestas vivas do vidro) e a serigrafia. Finalmente, o vidro é temperado ou laminado.

Este trabalho é focado na primeira etapa do processo ilustrado na Figura 1: o corte da chapa de vidro em peças retangulares menores. Nesta etapa, a chapa é colocada sobre uma mesa e traçada por uma ferramenta que cria vincos na superfície do vidro, permitindo que a chapa seja facilmente dividida em partes menores, seja por outra máquina ou por operadores. Existem diferentes tipos de equipamentos de corte que podem ser utilizados para realizar esses vincos, e neste trabalho tratamos de máquinas CNC dotadas de um cabeçote que pode se mover em qualquer direção no plano.

As chapas de vidro mais comumente usadas medem 6000 mm $\times$ 3210 mm e são chamadas de chapas *jumbo*. As peças recortadas da chapa possuem dimensões que podem variar de 150 mm (largura do vidro lateral de um carro) a mais de 2000 mm (largura do vidro traseiro de um ônibus). A Figura 2 mostra um exemplo de padrão de corte para a produção de dois tipos de peças: 2000 mm $\times$ 1200 mm e 2000 mm $\times$ 600 mm. Nesse padrão de corte bidimensional, cada chapa jumbo produzirá seis peças do primeiro tipo e três do segundo tipo. Além disso, haverá uma perda de 6,54 % de material (área hachurada).

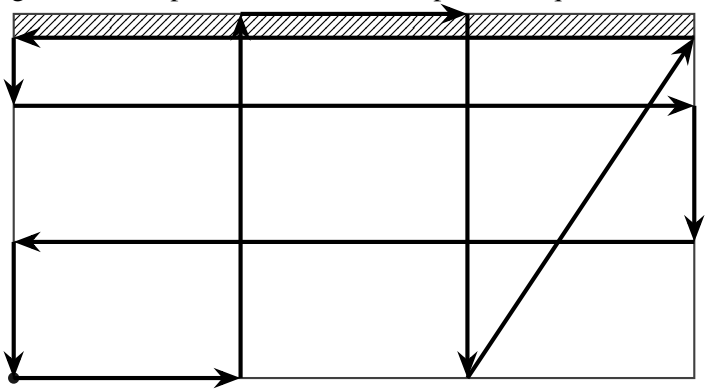
Figura 2: Exemplo de padrão de corte bidimensional para chapa de vidro.



Fonte: Elaborada pelos autores.

O cabeçote da máquina de corte inicia e termina cada ciclo na mesma posição. Supondo que este ponto é o canto inferior esquerdo da chapa de vidro, a Figura 3 mostra um possível caminho que o cabeçote da máquina pode seguir. Este caminho tem uma distância total de 35625 mm e é ótimo, ou seja, possui comprimento mínimo.

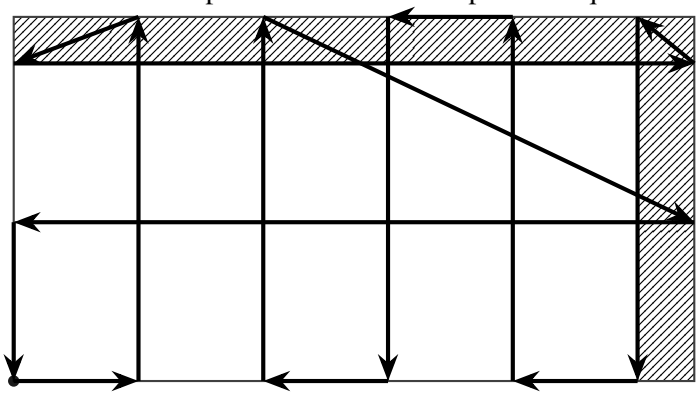
Figura 3: Exemplo de caminho ótimo para a máquina de corte.



Fonte: Elaborada pelos autores.

Determinar um caminho de comprimento mínimo para o cabeçote da máquina é importante porque a etapa de corte pode ser um gargalo no processo. A aparência simples do caminho exibido na Figura 3 pode sugerir que encontrar um caminho de corte ideal é uma tarefa fácil. No entanto, alguns padrões de corte podem resultar em caminhos ótimos que não parecem intuitivos. Como exemplo, a Figura 4 mostra um caminho ótimo para cortar 10 peças de tamanho 1100 mm $\times$ 1400 mm de uma chapa jumbo, com uma distância total de 39880 mm.

Figura 4: Outro exemplo de caminho ótimo para a máquina de corte.



Fonte: Elaborada pelos autores.

Na Seção 4, propomos o uso de dois modelos matemáticos para reduzir o tempo do ciclo de corte, minimizando a distância percorrida pelo cabeçote da máquina ao traçar um determinado padrão de corte, e resolvemos um grande conjunto de instâncias do problema, cujos resultados são apresentados na Seção 5.

4. Modelagem Matemática

O TSP é o problema de encontrar um ciclo de custo mínimo que visite todos os nós de um grafo conectado pelo menos uma vez. O RPP é o problema de encontrar um ciclo de custo mínimo que percorra um certo subconjunto das arestas do grafo (chamadas *arestas necessárias*) pelo menos uma vez. Qualquer instância TSP pode ser convertida em uma instância RPP (Christofides et al., 1981), e vice-versa (Laporte, 1997). Essas conversões envolvem transformações no grafo original que podem aumentar o número de arestas e nós (ver, por exemplo, Silva et al., 2019). Para o caso particular do CPDP estudado neste artigo, entretanto, o problema pode ser modelado como um TSP ou um RPP usando o mesmo grafo, ou seja, sem nenhuma transformação.

Representamos o padrão de corte na chapa de vidro com um grafo euclidiano completo não direcionado $G = (N, E)$, em que $N = \{0, 1, \ldots, n\}$ é o conjunto de nós e E é o conjunto de arestas. Seja v o número de linhas verticais e seja h o número de linhas horizontais do padrão de corte bidimensional. Definimos a posição inicial (e final) do cabeçote da máquina em cada ciclo de corte como o canto inferior esquerdo da chapa e fixamos esse ponto como o nó 0. Os nós restantes correspondem às extremidades de cada linha do padrão de corte bidimensional (então $n = 2v + 2h$). O número total de nós no grafo é $n' = n + 1$.

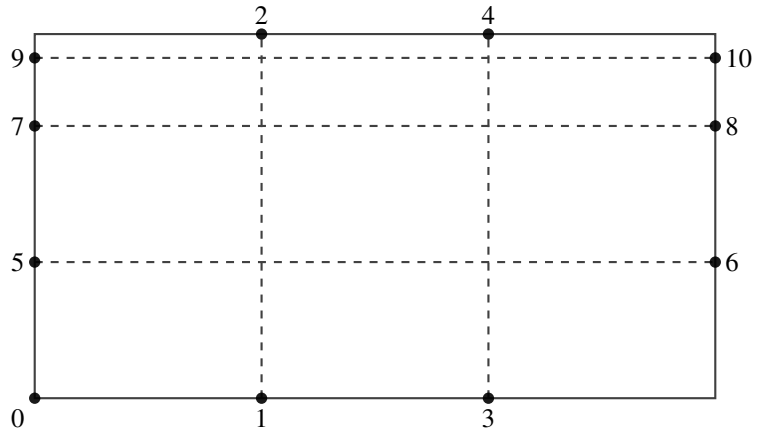
Cada aresta do grafo representa um possível movimento do cabeçote da máquina. Quando esta ferramenta percorre as bordas correspondentes às linhas verticais e horizontais do padrão, ela cria vincos na chapa de vidro, portanto, essas são as arestas necessárias. As demais arestas também representam movimentos do cabeçote sobre a chapa, porém sem tocá-la.

A Figura 5 mostra os nós e as arestas necessárias do grafo associado ao padrão de corte exibido na Figura 2, para o qual $v = 2$ e $h = 3$. Os nós estão representados com pontos pretos, as arestas necessárias estão representadas com linhas tracejadas, e todas as outras arestas estão omitidas.

Denotamos por R o conjunto de arestas necessárias e por G_R o grafo induzido por R. Cada aresta necessária é um componente conexo de G_R e, portanto, o número de componentes conexos neste grafo é $v + h$. O cabeçote inicia no nó 0 e deve percorrer todas as arestas em R antes de retornar à sua posição inicial. Assim, o problema pode ser modelado como um RPP. Como o nó inicial é 0 e como G_R contém todos os outros nós $(1, \ldots, n)$, o cabeçote deve visitar todos os nós em N. Portanto, o problema pode ser modelado como um TSP usando o mesmo grafo.

O grafo G representando o padrão de corte, conforme descrito, possui algumas características que podem ser utilizadas para simplificar a modelagem do problema. Como G é completo e euclidiano (portanto, a desigualdade triangular é válida) o caminho mais curto entre qualquer par de nós é sempre a aresta que os conecta. Como os componentes conexos de G_R são arestas únicas, uma vez que a máquina atravessa uma dessas arestas, ela não precisa retornar a ela (ou a qualquer um de seus nós finais). Assim, em uma solução ótima, todos os nós serão visitados exatamente uma vez.

Figura 5: Nós e arestas necessárias de um padrão de corte.



Fonte: Elaborada pelos autores.

Para o restante do artigo, uma rota é um ciclo que contém todos os nós do grafo, e uma sub-rota é um ciclo que contém apenas um subconjunto próprio dos nós no grafo. Os modelos TSP e RPP apresentados na Seção 4.1 usam as seguintes variáveis e parâmetros:

Variáveis

. $x_e = 1$ se a aresta e está na rota e $x_e = 0$ caso contrário, $e \in E$.

Parâmetros

. c_e é o custo da aresta $e \in E$.

. Δ_i é o conjunto de arestas incidentes no nó $i \in N$.

. E_{UV} é o conjunto de arestas com uma extremidade em $U \subset N$ e a outra em $V \subset N$. (Assim, E_{UU} é o conjunto de arestas do grafo induzido por U, e $E_{U\overline{U}}$ é o conjunto de corte entre U e $\overline{U} = N \setminus U$.)

. N_T é o conjunto de nós que são extremidades das arestas em $T \subset E$.

4.1. Modelos Usando o TSP

Com base na formulação proposta por Dantzig et al. (1954), um modelo TSP para CPDP é:

$$\text{minimizar} \quad \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{sujeito a} \quad x_e = 1, \ \ \forall e \in R \tag{2}$$

$$\sum_{e \in \Delta_i} x_e = 2, \ \ \forall i \in N \tag{3}$$

$$\sum_{e \in E_{SS}} x_e \leq |S| - 1, \ \ \forall S \subset N, \ 3 \leq |S| \leq \frac{n}{2} \tag{4}$$

$$x_e \in \{0, 1\}. \tag{5}$$

A função objetivo (1) minimiza a distância percorrida pelo cabeçote da máquina. As restrições (2) forçam cada aresta necessária a estar na rota. As restrições (3) garantem que todo nó tenha grau dois, e as restrições (4) impedem soluções com sub-rotas. Finalmente, (5) define o domínio das variáveis.

Uma forma alternativa de evitar sub-rotas é usar as seguintes restrições:

$$\sum_{e \in E_{S\overline{S}}} x_e \geq 2, \ \ \forall S \subset N, \ 3 \leq |S| \leq \frac{n}{2}. \tag{6}$$

As restrições (6) garantem que a solução é um único componente conexo e podem ser usadas no lugar de (4). Chamaremos o modelo (1)–(5) de TSP1 e o modelo (1)–(3), (5), (6) de TSP2.

4.2. Modelo Usando o RPP

Com base na formulação proposta por Christofides et al. (1981) e incluindo o nó 0 na rota, um modelo RPP para CPDP é:

$$\text{minimizar} \quad \sum_{e \in E} c_e x_e \tag{7}$$

$$\text{sujeito a} \quad x_e \geq 1, \ \ \forall e \in R \tag{8}$$

$$\sum_{e \in \Delta_i} x_e \equiv 0 \mod 2, \ \ \forall i \in N \tag{9}$$

$$\sum_{e \in \Delta_0} x_e \geq 2 \tag{10}$$

$$\sum_{e \in E_{N_T \overline{N_T}}} x_e \geq 2, \ \ \forall T \subsetneq R \tag{11}$$

$$x_e \in \mathbb{Z}_+. \tag{12}$$

A função objetivo (7) minimiza a distância percorrida pelo cabeçote da máquina. As restrições (8) forçam cada aresta necessária a estar na rota. As restrições (9) garantem que cada nó tenha grau par e a restrição (10) inclui o nó 0 na rota. As restrições (11) garantem que todos os subconjuntos de R (que são os componentes conexos de G_R) sejam conectados ao seu complemento em E. Finalmente, (12) define o domínio das variáveis.

Para o caso particular do CPDP considerado neste artigo, o modelo RPP é equivalente ao modelo TSP2. Primeiro observemos que (1) é igual a (7). Pelos argumentos discutidos no início desta seção, em uma rota ótima, cada aresta necessária é percorrida exatamente uma vez e o grau de cada nó é dois. Assim (8) e (9) são simplificados para (2) e (3), respectivamente, e (10) pode

ser descartado. Além disso, cada aresta não necessária é percorrida no máximo uma vez, então (12) torna-se (5).

Para completar a demonstração, precisamos provar que (6) e (11) são equivalentes para todas as soluções que satisfazem (2), (3) e (5). Assim, seja x essa solução e seja $Q = \{e \in E \mid x_e = 1\}$. Se as arestas em Q formam um único ciclo, então x satisfaz tanto (6) quanto (11). Caso contrário, as arestas em Q formam duas ou mais sub-rotas desconectadas. Agora mostraremos que, neste caso, (i) existe uma restrição correspondente (6) para cada restrição (11) e (ii) vice-versa.

(i) Seja S o conjunto de nós de uma sub-rota na solução com $3 \leq |S| \leq \frac{n}{2}$. Se $|S|$ é ímpar, então S contém o nó 0, e escolhendo T para ser o conjunto de arestas necessárias com nós em $\overline{S}$, as restrições (6) e (11) são iguais. Se $|S|$ é par, então S não contém nó 0, e escolhendo T para ser o conjunto de arestas necessárias com nós em S, as restrições (6) e (11) são iguais. Assim, toda restrição (6) tem uma restrição (11) correspondente.

(ii) Seja T o conjunto de arestas necessárias em uma sub-rota da solução que não inclui o nó 0. Se $|N_T| \leq \frac{n}{2}$, então escolhendo $S = N_T$, (11) e (6) são iguais. Se $|N_T| > \frac{n}{2}$, então escolhendo $S = \overline{N_T}$, (11) e (6) são iguais. Assim, toda restrição (11) tem uma restrição (6) correspondente.

Em suma, demonstramos que, para o problema estudado, os modelos RPP e TSP2 são equivalentes. Assim, a partir deste ponto, nos referimos apenas aos modelos TSP1 e TSP2.

5. Experimentos Computacionais

Realizamos testes computacionais para avaliar o desempenho de TSP1 e TSP2 para resolver instâncias do CPDP. Os modelos foram implementados em Python e resolvidos com o *solver* Gurobi 9.0.0 em um laptop Intel i5 com velocidade de CPU de 2,67 GHz e 4 GB de RAM. Para cada instância, permitimos um tempo computacional de 600 segundos e definimos o parâmetro "MIPGapAbs" como zero, o que faz com que o Gurobi interrompa a otimização somente quando o valor ótimo absoluto da função objetivo for alcançado. O código-fonte com os modelos implementados, as instâncias testadas e os resultados computacionais completos estão disponíveis em `http://dx.doi.org/10.17632/8jyf6s695p.1`.

5.1. Instâncias

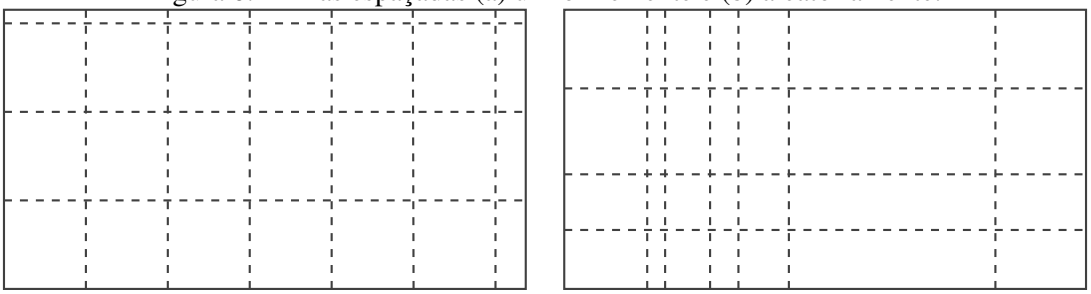
As instâncias estão divididas em dois conjuntos: no primeiro, as linhas verticais e horizontais são espaçadas uniformemente, enquanto no segundo elas são espaçadas aleatoriamente. No conjunto com linhas espaçadas uniformemente, todas os retângulos formados são idênticos, retratando a produção de peças com as mesmas dimensões. Já no conjunto com linhas espaçadas aleatoriamente, os retângulos têm tamanhos diferentes, o que reflete os casos em que as peças produzidas são distintas. Este conjunto também contém instâncias com retângulos cujas dimensões não são encontradas na prática, mas que foram mantidas para expandir a abrangência dos experimentos computacionais. A Figura 6 mostra duas instâncias com seis linhas verticais e três horizontais. Na Figura 6a as linhas são espaçadas uniformemente e na Figura 6b elas são espaçadas aleatoriamente. Em todos os casos, o tamanho da chapa de vidro foi de 6000 mm $\times$ 3210 mm.

Para cada conjunto, foram geradas 3000 instâncias. O número de linhas verticais, v, variou de 1 a 30, e o número de linhas horizontais, h, variou de 1 a 20. Para cada combinação de v e h, cinco instâncias foram geradas usando as seguintes regras (todos os valores estão em milímetros):

- Para instâncias com linhas uniformemente espaçadas, o espaço entre linhas verticais consecutivas é um inteiro sorteado de $\left\{\lfloor\frac{6001}{v+1}\rfloor, \ldots, \lfloor\frac{5999}{v}\rfloor\right\}$ e o espaço entre linhas consecutivas horizontais é um inteiro sorteado de $\left\{\lfloor\frac{3211}{h+1}\rfloor, \ldots, \lfloor\frac{3209}{h}\rfloor\right\}$. Após o procedimento de geração destas instâncias, as peças obtidas têm larguras entre 193 mm e 5958 mm e alturas entre 152 mm e 3202 mm.

• Para instâncias com linhas espaçadas aleatoriamente, a coordenada horizontal de cada linha vertical é um inteiro sorteado de $\{0, \ldots, 6000\}$ e a coordenada vertical de cada linha horizontal é um inteiro sorteado de $\{0, \ldots, 3210\}$. Após o procedimento de geração destas instâncias, as peças obtidas têm larguras entre 1 mm e 5938 mm e alturas entre 1 mm e 3208 mm.

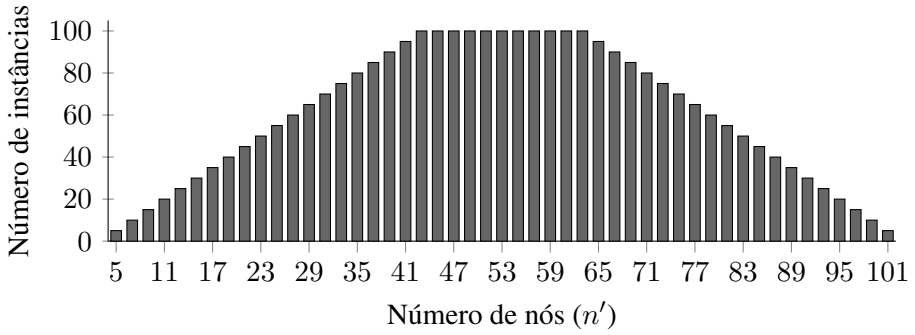
Figura 6: Linhas espaçadas (a) uniformemente e (b) aleatoriamente.



Fonte: Elaborada pelos autores.

Dividimos a análise dos resultados por tipo de instância: primeiro consideramos o conjunto com linhas espaçadas uniformemente, depois o conjunto com linhas espaçadas aleatoriamente. Para cada conjunto, o número de nós (n') varia de 5 (quando $v = h = 1$) a 101 (quando $v = 30, h = 20$). A quantidade de instâncias para cada valor de n' não é a mesma, como mostra a Figura 7, pois diferentes combinações de v e h podem resultar no mesmo valor de n'. Por exemplo, um grafo com $n' = 11$ ocorre em quatro configurações diferentes: $v = 1, h = 4$; $v = 2, h = 3$; $v = 3, h = 2$; e $v = 4, h = 1$. Como será mostrado, instâncias com mais nós tendem a ser mais difíceis de resolver, e a paridade de v e h também contribui para a dificuldade do problema.

Figura 7: Número de instâncias em função do número de nós (n').

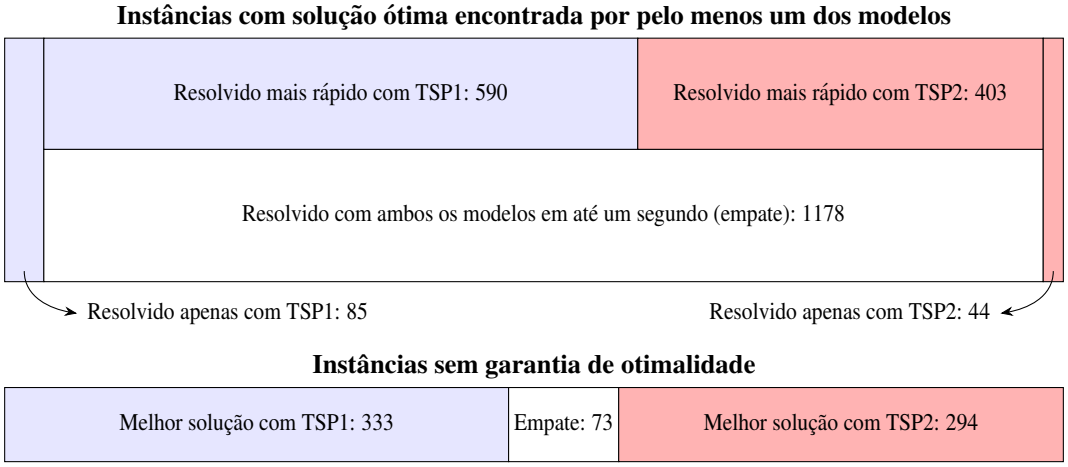


Fonte: Elaborada pelos autores.

5.2. Resultados para Instâncias com Linhas Espaçadas Uniformemente

Iniciamos nossa análise comparando o desempenho dos modelos TSP1 e TSP2. Dizemos que uma instância foi resolvida quando a otimização foi finalizada pelo *solver* dentro do tempo permitido (e, portanto, uma solução comprovadamente ótima foi encontrada). Para decidir qual modelo teve melhor desempenho, usamos as seguintes regras: (i) se ambos os modelos terminam em menos de um segundo, houve empate; (ii) se um modelo resolveu antes do outro (pelo menos um deles após um segundo), o modelo mais rápido venceu; (iii) se ambos os modelos atingiram o limite de tempo, o modelo vencedor foi aquele que encontrou a melhor solução (se ambos atingiram o mesmo valor da função objetivo, houve empate).

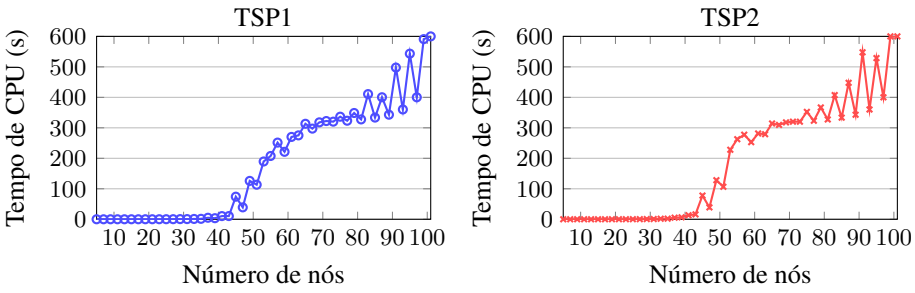
Figura 8: Comparação entre TSP1 e TSP2 na resolução de instâncias com linhas espaçadas uniformemente.



Fonte: Elaborada pelos autores.

De um total de 3000 instâncias, o modelo TSP1 resolveu 2256 e o modelo TSP2 resolveu 2215. O número de instâncias resolvidas por ambos os modelos foi de 2171, das quais 1178 foram resolvidas em menos de um segundo. As outras 993 demoraram mais para serem resolvidas com pelo menos um modelo (sendo 590 mais rápidas com TSP1 e 403 mais rápidas com TSP2). No total, 700 instâncias não terminaram dentro do tempo limite com nenhum dos modelos, então neste caso comparamos o valor da melhor solução encontrada: TSP1 foi melhor em 333 instâncias, TSP2 foi melhor em 294 instâncias e nas outras 73 houve empate. O *gap* de otimalidade relativo para essas instâncias ao final da otimização foi muito pequeno, com valor máximo de 0,0282 %. Ao todo, TSP1 venceu em 1008 instâncias (33,6 %), TSP2 venceu em 741 instâncias (24,7 %) e houve empate em 1251 instâncias (41,7 %). A Figura 8 resume esses resultados.

Figura 9: Tempo médio de CPU em função de n' para instâncias com linhas espaçadas uniformemente.



Fonte: Elaborada pelos autores.

Consideremos agora o tempo de CPU (ou seja, o tempo computacional) gasto em cada instância. Na Figura 9, o gráfico à esquerda mostra os tempos médios de CPU das instâncias em função do número de nós usando TSP1, enquanto o gráfico à direita apresenta a mesma informação com TSP2. Em ambos os gráficos, observamos que as instâncias com até 43 nós foram resolvidas muito rapidamente e que os tempos de CPU crescem a partir desse ponto. Para as instâncias maiores, com 101 nós, o tempo médio de CPU foi de 600 s com ambos os modelos, o que significa que nenhuma dessas instâncias foi resolvida com otimalidade comprovada. O tempo médio de CPU para todo o conjunto de instâncias foi de 170 s com TSP1 e 178 s com TSP2.

Os tempos médios de CPU exibidos nos gráficos da Figura 9 possuem um padrão semelhante ao de uma serra, em vez de uma tendência mais suave. Vamos explorar esse fato

particionando o conjunto de instâncias de acordo com a paridade do número de linhas verticais e horizontais (v e h, respectivamente) dos padrões de corte. Como será evidenciado, instâncias com um número par de linhas verticais e horizontais tendem a ser mais difíceis de resolver. A Tabela 1 descreve a divisão do conjunto de instâncias em quatro subconjuntos de tamanho igual e apresenta o tempo médio de CPU de cada subconjunto com TSP1 e TSP2.

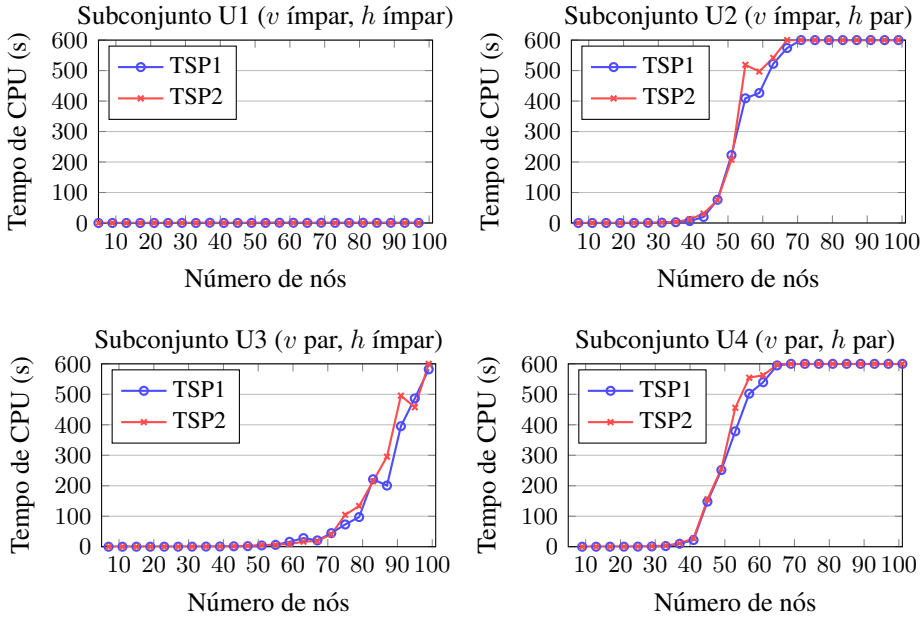
Tabela 1: Subconjuntos de instâncias com linhas espaçadas uniformemente e tempos médios de CPU.

Subconjunto	Paridade de v e h	Tempo médio de CPU	
		TSP1	TSP2
U1	v ímpar, h ímpar	0,4 s	0,2 s
U2	v ímpar, h par	290,8 s	305,8 s
U3	v par, h ímpar	45,9 s	51,4 s
U4	v par, h par	343,0 s	354,6 s

Fonte: Elaborada pelos autores.

Para todas as instâncias, o número de nós é sempre ímpar. No subconjunto U1, $n' \in \{5, 9, \ldots, 97\}$; nos subconjuntos U2 e U3, $n' \in \{7, 11, \ldots, 99\}$; e no subconjunto U4, $n' \in \{9, 13, \ldots, 101\}$. Assim, quando calculamos a média dos tempos de CPU para todas as instâncias com o mesmo número de nós para construir os gráficos da Figura 9, aproximadamente metade dos pontos vieram de U1 e U4, enquanto os outros pontos vieram de U2 e U3. A Figura 10 exibe os gráficos dos tempos médios de CPU em função do número de nós para cada subconjunto.

Figura 10: Tempo médio de CPU em função de n' nos subconjuntos U1, U2, U3 e U4.



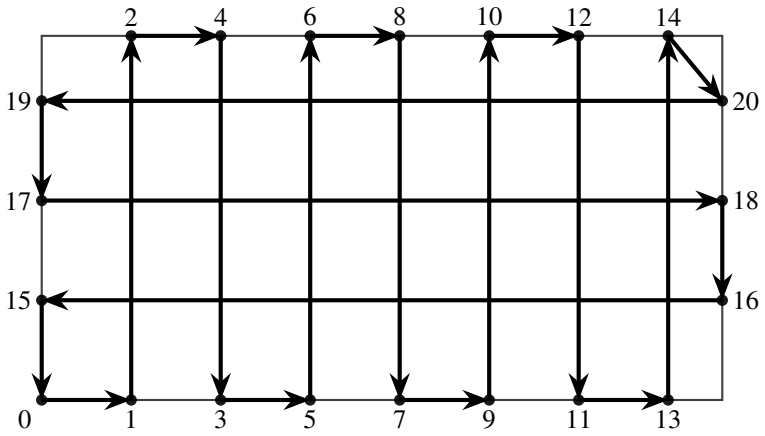
Fonte: Elaborada pelos autores.

Observamos nestes gráficos como a paridade de v e h influencia no tempo de CPU. Especificamente, as instâncias começam a ficar mais difíceis com valores menores de n' se v e/ou h forem pares. Nos gráficos da Figura 9, os pontos que vêm dos subconjuntos U2 e U3 são os "localmente altos", enquanto os pontos que vêm de U1 e U4 são os "localmente baixos".

O tempo médio de CPU em U1 foi extremamente pequeno quando comparado com os outros subconjuntos, e mesmo as instâncias grandes foram resolvidas muito rapidamente. Em todas as instâncias deste subconjunto, notamos uma sequência "intuitiva" de movimentos nos percursos

ótimos, em que o cabeçote executa primeiro todos os cortes verticais, faz um único movimento diagonal e depois executa todos os cortes horizontais.

Figura 11: Rota ótima para uma instância com $v = 7$ e $h = 3$.



Fonte: Elaborada pelos autores.

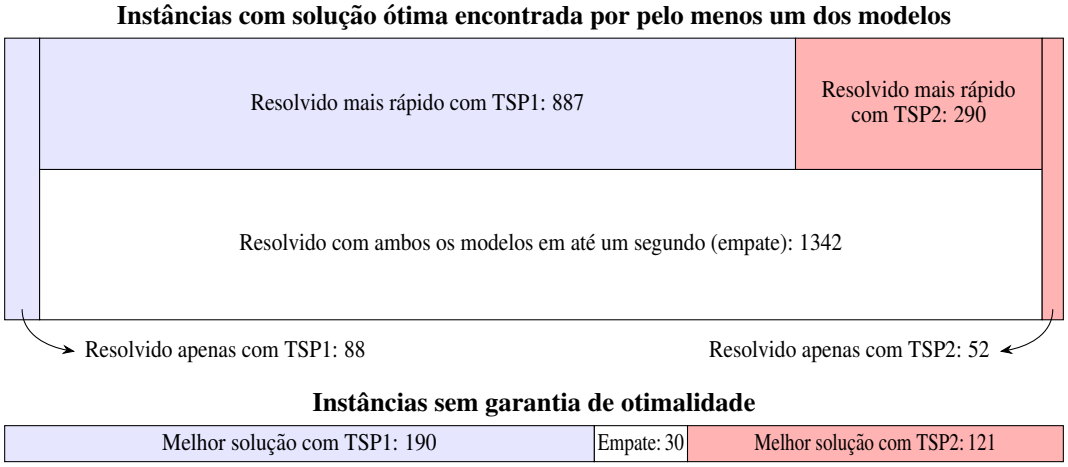
A Figura 11 ilustra o comportamento de uma instância com $v = 7$ e $h = 3$: partindo do nó 0, o cabeçote se move para 1, executa um corte e para no nó 2, depois segue para traçar a linha paralela seguinte, de 4 a 3, e continua desta forma da esquerda para a direita até ter traçado todas as linhas verticais, parando no nó 14; agora a ferramenta realiza um movimento diagonal até o nó 20 e traça todas as linhas horizontais, de cima para baixo; ao atingir o nó 15, todas as linhas foram traçadas e o cabeçote retorna ao nó 0.

Esse tipo de trajeto coincide com a solução que seria fornecida por um algoritmo de vizinho mais próximo para o TSP, então a rapidez da solução poderia ser fruto das heurísticas do Gurobi, que são executadas automaticamente por padrão. Para testar essa hipótese, rodamos todas as instâncias do subconjunto U1 desabilitando as heurísticas no Gurobi (ou seja, com o parâmetro "Heuristics" definido como zero) e os resultados foram semelhantes. Portanto, não foi este parâmetro que causou os tempos de CPU baixos. Além disso, como será mostrado na Seção 5.3, as instâncias com linhas espaçadas aleatoriamente tiveram um comportamento semelhante com relação aos tempos de CPU quando v e h são ímpares, mas as rotas ótimas nem sempre tiveram esse mesmo tipo de geometria em zigue-zague.

5.3. Resultados para Instâncias com Linhas Espaçadas Aleatoriamente

Seguimos o mesmo critério utilizado na Seção 5.2 para comparar o desempenho dos modelos TSP1 e TSP2 para resolver instâncias com linhas espaçadas aleatoriamente. Das 3000 instâncias, TSP1 resolveu 2607 e TSP2 resolveu 2571. O número de instâncias resolvidas até a otimalidade por ambos os modelos foi de 2519, das quais 1342 foram resolvidas em menos de um segundo. As outras 1177 demoraram mais para terminar com pelo menos um modelo (sendo 887 mais rápidas com TSP1 e 290 mais rápidas com TSP2). No total, 341 instâncias não terminaram dentro do tempo limite com nenhum dos modelos, então neste caso comparamos o valor da melhor solução encontrada: TSP1 foi melhor em 190 instâncias, TSP2 foi melhor em 121 instâncias e nas outras 30 houve empate. O *gap* de otimalidade relativa para essas instâncias ao final da otimização foi muito pequeno, com valor máximo de 0,0227 %. Ao todo, TSP1 venceu em 1165 instâncias (38,8 %), TSP2 venceu em 463 instâncias (15,4 %) e houve empate em 1372 instâncias (45,8 %). A Figura 12 resume esses resultados.

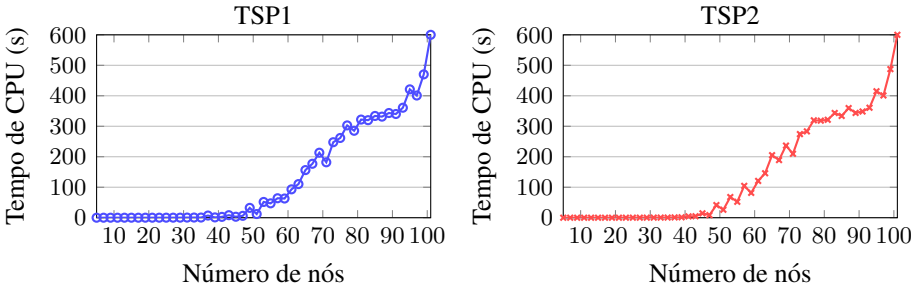
Figura 12: Comparação entre TSP1 e TSP2 na resolução de instâncias com linhas espaçadas aleatoriamente.



Fonte: Elaborada pelos autores.

Na Figura 13 estão os tempos médios de CPU em função do número de nós usando TSP1 e TSP2. Novamente observamos, em ambos os gráficos, que instâncias com até 45 nós foram resolvidas muito rapidamente e que os tempos de CPU crescem a partir deste ponto. Para as instâncias maiores, com 101 nós, o tempo médio de CPU foi de 600 s com ambos os modelos, o que significa que nenhuma dessas instâncias foi resolvida com otimalidade comprovada. O tempo médio de CPU para todo o conjunto de instâncias foi de 101 s com TSP1 e 113 s com TSP2.

Figura 13: Tempo médio de CPU em função de n' para instâncias com linhas espaçadas aleatoriamente.



Fonte: Elaborada pelos autores.

Os gráficos da Figura 13 exibem um comportamento de serra menos pronunciado do que sua contraparte na Figura 9. Novamente, particionamos o conjunto de instâncias de acordo com a paridade do número de linhas verticais e horizontais dos padrões de corte, conforme mostrado na Tabela 2, que também traz o tempo médio de CPU para cada subconjunto com TSP1 e TSP2.

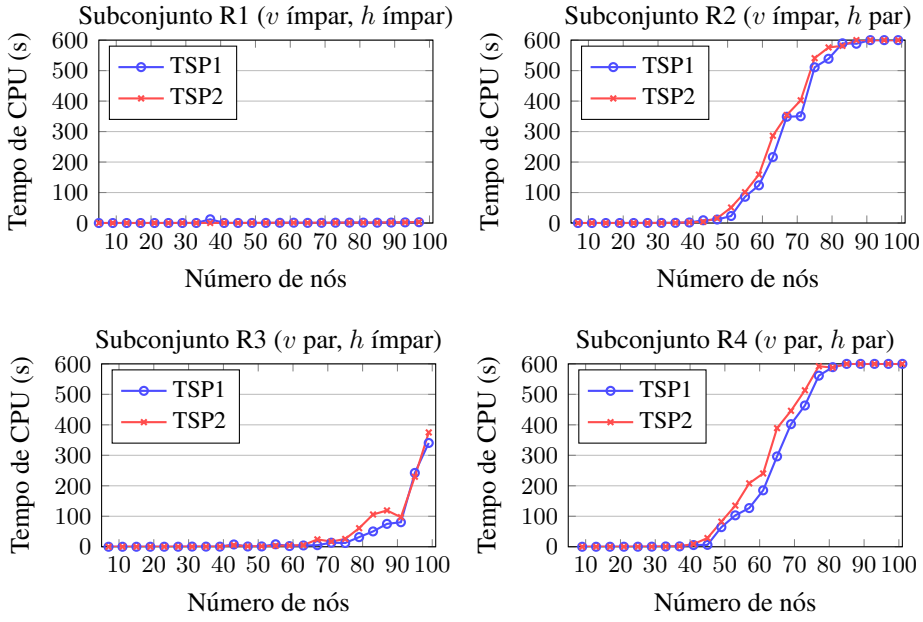
Tabela 2: Subconjuntos de instâncias com linhas espaçadas aleatoriamente e tempos médios de CPU.

Subconjunto	Paridade de v e h	Tempo médio de CPU	
		TSP1	TSP2
R1	v ímpar, h ímpar	1,3 s	0,6 s
R2	v ímpar, h par	175,8 s	191,7 s
R3	v par, h ímpar	15,3 s	21,7 s
R4	v par, h par	211,2 s	238,2 s

Fonte: Elaborada pelos autores.

A Figura 14 apresenta os gráficos dos tempos médios de CPU em função do número de nós para cada subconjunto. Mais uma vez, vemos que as instâncias começam a ficar mais difíceis com valores menores de n' se v e/ou h forem pares. Nesses gráficos é possível observar que a paridade de h parece ter uma influência muito maior no tempo de CPU do que a paridade de v.

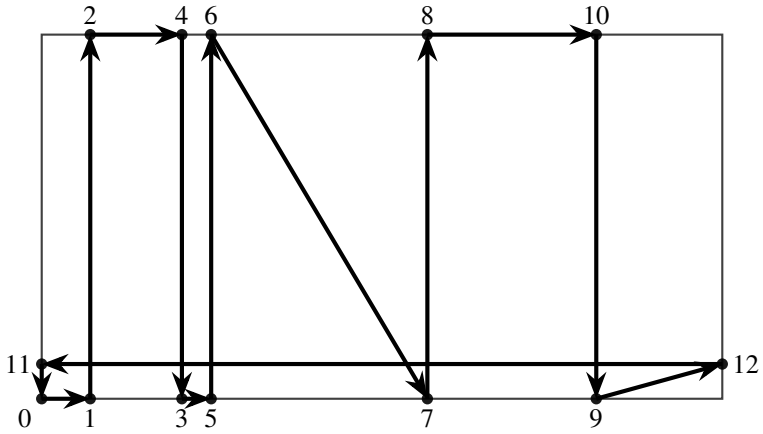
Figura 14: Tempo médio de CPU em função de n' nos subconjuntos R1, R2, R3 e R4.



Fonte: Elaborada pelos autores.

Como no caso de linhas uniformemente espaçadas, o tempo médio de CPU do subconjunto com v e h ímpares, R1, foi extremamente pequeno. No entanto, a sequência "intuitiva" de movimentos observada nas rotas ótimas de todas as instâncias de U1 não ocorreu em todas as instâncias de R1. A Figura 15 mostra o percurso ótimo de uma instância com $v = 5$ e $h = 1$ (ambos ímpares) em que há dois movimentos diagonais do cabeçote da máquina. Além disso, essa rota ótima não seria a solução fornecida por um algoritmo que segue a regra do vizinho mais próximo.

Figura 15: Rota ótima para uma instância com $v = 5$ e $h = 1$.



Fonte: Elaborada pelos autores.

6. Conclusões

Este artigo abordou o problema de minimizar o comprimento de um ciclo que inclui todos os nós em um grafo usado para modelar os possíveis movimentos do cabeçote de determinadas máquinas de corte de vidro. Mostramos como o problema pode ser modelado como um TSP e um RPP usando o mesmo grafo. Apresentamos duas formulações usando TSP e uma formulação usando RPP, e demonstramos que, para o problema específico estudado, a formulação RPP é equivalente a uma das formulações TSP.

Testamos computacionalmente as formulações TSP1 e TSP2 para resolver um conjunto de 6000 instâncias, divididas em dois conjuntos de mesmo tamanho. No primeiro conjunto, as linhas verticais e horizontais são espaçadas uniformemente e, no segundo, são espaçadas aleatoriamente. No conjunto com linhas espaçadas uniformemente, o TSP1 venceu em 33,6 % das instâncias e o TSP2 venceu em 24,7 % das instâncias, havendo empate nos 41,7 % das instâncias restantes. No conjunto com linhas espaçadas aleatoriamente, o TSP1 venceu em 38,8 % das instâncias e o TSP2 venceu em 15,4 % das instâncias, havendo empate nos 45,8 % das instâncias restantes. Portanto, os resultados indicam superioridade do TSP1 em relação ao TSP2, mas não de forma consistente, pois houve casos em que o TSP2 teve melhor desempenho.

Para as duas formulações testadas, além da quantidade de nós no grafo, a paridade do número de linhas verticais e horizontais (ou seja, se o número de linhas é par ou ímpar) no padrão de corte exerce um papel importante no tempo necessário para resolver as instâncias. Uma questão que permanece em aberto é explicar por que instâncias com um número par de linhas verticais e horizontais tendem a ser mais difíceis de resolver.

Dadas as características particulares do problema estudado (grafo euclidiano completo com nós nas bordas de um retângulo e arestas necessárias horizontais e verticais), um possível tópico a ser explorado é o desenvolvimento de um algoritmo que determine uma rota ótima sem a utilização de *solvers* de otimização matemática. Por fim, observamos que, embora a indústria fabricante de vidro automotivo tenha servido como motivação, podem existir aplicações semelhantes em outros setores industriais.

Agradecimentos. Este trabalho teve o apoio financeiro do CNPq (processos números 302998/2022-5 e 402240/2023-5) e FAPESP (processos números 2013/07375-0 e 2022/05803-3).

Referências

Al-Khayyal, F., Griffin, P. M. e Smith, N. R. Solution of a large-scale two-stage decision and scheduling problem using decomposition. *European Journal of Operational Research*, v. 132, n. 2, p. 453–465, 2001.

Alvarez-Valdes, R., Fuertes, A., Tamarit, J. M., Giménez, G. e Ramos, R. A heuristic to schedule flexible job-shop in a glass factory. *European Journal of Operational Research*, v. 165, n. 2, p. 525–534, 2005.

Arbib, C. e Marinelli, F. An optimization model for trim loss minimization in an automotive glass plant. *European Journal of Operational Research*, v. 183, n. 3, p. 1421–1432, 2007.

Castelino, K., D'Souza, R. e Wright, P. K. Toolpath optimization for minimizing airtime during machining. *Journal of Manufacturing Systems*, v. 22, n. 3, p. 173–180, 2003.

Chambers, M. L. e Dyson, R. G. The cutting stock problem in the flat glass industry – selection of stock sizes. *Operational Research Quarterly*, v. 27, n. 4, p. 949–957, 1976.

Christofides, N., Corberán, A., Campos, V. e Mota, E. *An algorithm for the rural postman problem*. Imperial College Report IC.O.R.81.5, London, England, 1981.

Dantzig, G., Fulkerson, R. e Johnson, S. Solution of a large-scale traveling-salesman problem. *Operations Research*, v. 2, n. 4, p. 393–410, 1954.

Dewil, R., Vansteenwegen, P. e Cattrysse, D. Cutting path optimization using tabu search. *Key Engineering Materials*, v. 473, p. 739–748, 2011.

Dewil, R., Vansteenwegen, P. e Cattrysse, D. Construction heuristics for generating tool paths for laser cutters. *International Journal of Production Research*, v. 52, n. 20, p. 5965–5984, 2014.

Dyson, A. R. G. e Gregory, A. S. The cutting stock problem in the flat glass industry. *Operational Research Quarterly*, v. 25, n. 1, p. 41–53, 1974.

Han, G. C. e Na, S. J. A study on torch path planning in laser cutting processes, part 2: cutting path optimization using simulated annealing. *Journal of Manufacturing Processes*, v. 1, n. 1, p. 62–70, 1999.

Hoeft, J. e Palekar, U. S. Heuristics for the plate-cutting traveling salesman problem. *IIE Transactions (Institute of Industrial Engineers)*, v. 29, n. 9, p. 719–731, 1997.

Imahori, S., Kushiya, M., Nakashima, T. e Sugihara, K. Generation of cutter paths for hard material in wire EDM. *Journal of Materials Processing Technology*, v. 206, p. 453–461, 2008.

Laporte, G. Modeling and solving several classes of arc routing problems as traveling salesman problems. *Computers & Operations Research*, v. 24, n. 11, p. 1057–1061, 1997.

Lee, M. K. e Kwon, K. B. Cutting path optimization in CNC cutting processes using a two-step genetic algorithm. *International Journal of Production Research*, v. 44, n. 24, p. 5307–5326, 2006.

Lee, Y. e Lee, K. Lot-sizing and scheduling in flat-panel display manufacturing process. *Omega (United Kingdom)*, v. 93, p. 102036, 2020.

Lin, K. Y. User experience-based product design for smart production to empower industry 4.0 in the glass recycling circular economy. *Computers & Industrial Engineering*, v. 125, p. 729–738, 2018.

Madsen, O. B. G. An application of travelling-salesman routines to solve pattern-allocation problems in the glass industry. *The Journal of the Operational Research Society*, v. 39, n. 3, p. 249–256, 1988.

Manber, U. e Israni, S. Pierce point minimization and optimal torch path determination in flame cutting. *Journal of Manufacturing Systems*, v. 3, n. 1, p. 81–89, 1984.

Moreira, L. M., Oliveira, J. F., Gomes, A. M. e Ferreira, J. S. Heuristics for a dynamic rural postman problem. *Computers & Operations Research*, v. 34, n. 11, p. 3281–3294, 2007.

Na, B., Ahmed, S., Nemhauser, G. e Sokol, J. Optimization of automated float glass lines. *International Journal of Production Economics*, v. 145, n. 2, p. 561–572, 2013.

Na, B., Ahmed, S., Nemhauser, G. e Sokol, J. A cutting and scheduling problem in float glass manufacturing. *Journal of Scheduling*, v. 17, n. 1, p. 95–107, 2014.

Nicholls, M. G. The development of a mathematical model of the operations of a national glass manufacturer. *The Journal of the Operational Research Society*, v. 44, n. 9, p. 935–943, 1993.

Oliveira, L. T., Silva, E. F., Oliveira, J. F. e Toledo, F. M. B. Integrating irregular strip packing and cutting path determination problems: a discrete exact approach. *Computers & Industrial Engineering*, v. 149, p. 106757, 2020.

Parreño, F., Alonso, M. T. e Alvarez-Valdes, R. Solving a large cutting problem in the glass manufacturing industry. *European Journal of Operational Research*, v. 287, p. 378–388, 2020.

Puchinger, J., Koller, G. e Raidl, G. Solving a real-world glass cutting problem. In: *Evolutionary Computation in Combinatorial Optimization*. Coimbra, Portugal, 2004. p. 165–176.

Rodrigues, A. M. e Ferreira, J. S. Cutting path as a rural postman problem: solutions by memetic algorithms. *International Journal of Combinatorial Optimization Problems and Informatics*, v. 3, n. 1, p. 31–46, 2012.

Silva, E. F., Oliveira, L. T., Oliveira, J. F. e Toledo, F. M. B. Exact approaches for the cutting path determination problem. *Computers & Operations Research*, v. 112, p. 104772, 2019.

Taskin, Z. C. e Ünal, A. T. Tactical level planning in float glass manufacturing with co-production, random yields and substitutable products. *European Journal of Operational Research*, v. 199, n. 1, p. 252–261, 2009.

Wang, G. G. e Xie, S. Q. Optimal process planning for a combined punch-and-laser cutting machine using ant colony optimization. *International Journal of Production Research*, v. 43, n. 11, p. 2195–2216, 2005.

Wäscher, G., Haußner, H. e Schumann, H. An improved typology of cutting and packing problems. *European Journal of Operational Research*, v. 183, n. 3, p. 1109–1130, 2007.

Williams, R. L. e Gupta, A. K. Determination of minimum cutting tool paths in the presence of barriers. *Computers & Industrial Engineering*, v. 14, n. 4, p. 495–502, 1988.

Williams, R. L. e Lee, Y.-G. Determination of minimum three-dimensional cutting-tool paths in the presence of barriers. *Computers & Industrial Engineering*, v. 25, p. 211–214, 1993.