**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA E DE MATERIAIS – PPGEM**

**TIAGO RODRIGUES WELLER**

# *FRAMEWORK* PARA OTIMIZAÇÃO DOS MOVIMENTOS DE REPOSICIONAMENTO EM MANUFATURA ADITIVA POR EXTRUSÃO DE MATERIAL

**TESE DE DOUTORADO**

**CURITIBA**
**2019**

**TIAGO RODRIGUES WELLER**

# *FRAMEWORK* PARA OTIMIZAÇÃO DOS MOVIMENTOS DE REPOSICIONAMENTO EM MANUFATURA ADITIVA POR EXTRUSÃO DE MATERIAL

Tese apresentada como requisito parcial para obtenção do grau de Doutor em Engenharia, do Programa de Pós-Graduação em Engenharia Mecânica e de Materiais, Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Neri Volpato, *Ph.D.*

Co-orientador: Prof. Dr. Luiz Carlos de Abreu Rodrigues.

**CURITIBA**

**2019**

Dados Internacionais de Catalogação na Publicação

Weller, Tiago Rodrigues
Framework para otimização dos movimentos de reposicionamento em manufatura aditiva por extrusão de material [recurso eletrônico] / Tiago Rodrigues Weller. -- 2020.
1 arquivo texto (120 f.) : PDF ; 4,95 MB.

Modo de acesso: World Wide Web
Título extraído da tela de título (visualizado em 7 fev. 2020)
Texto em português com resumo em inglês
Tese (Doutorado) - Universidade Tecnológica Federal do Paraná. Programa de pós-graduação em Engenharia Mecânica e de Materiais, Curitiba, 2019
Bibliografia: f. 112-117.

1. Engenharia mecânica - Teses. 2. Impressão tridimensional. 3. Ciclo de vida do produto. 4. Produtos industrializados. 5. Programação linear. 6. Plásticos - Extrusão. 7. Framework (Programa de computador). 8. Metaheurísticas. 9. Otimização combinatória. I. Volpato, Neri. II. Rodrigues, Luiz Carlos de Abreu. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Mecânica e de Materiais. IV. Título.

CDD: ed. 23 – 621.3

Biblioteca Central da UTFPR, Câmpus Curitiba
Bibliotecário: Adriano Lopes, CRB-9/1429

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

# TERMO DE APROVAÇÃO DE TESE Nº 22

A Tese de Doutorado intitulada: **FRAMEWORK PARA OTIMIZAÇÃO DOS MOVIMENTOS DE REPOSICIONAMENTO EM MANUFATURA ADITIVA POR EXTRUSÃO DE MATERIAL**, defendida em sessão pública pelo Candidato **Tiago Rodrigues Weller**, no dia 29 de novembro de 2019, foi julgada para a obtenção do título de Doutor em Engenharia, área de concentração: Engenharia de Manufatura, linha de pesquisa: Otimização de Processos de Fabricação, e aprovada em sua forma final, pelo Programa de Pós-Graduação em Engenharia Mecânica e de Materiais – PPGEM.

BANCA EXAMINADORA:

Prof. PhD. Neri Volpato - Presidente - UTFPR

Prof. Dr. José Aguiomar Foggiatto - UTFPR

Prof. Dr. Jorge Vicente Lopes da Silva - CTI-Renato Archer

Prof. Dr. Paulo Henrique Siqueira - UFPR

Prof. Dr. Leandro Magatão - UTFPR

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 29 de novembro de 2019.

## DEDICATÓRIA

Aos meus pais, que não mediram esforços para me ajudar no que foi necessário.

Aos meus filhos Ana Clara e João Pedro, os quais foram minha motivação para continuar, mesmo com todas as adversidades pelo caminho.

## AGRADECIMENTOS

Primeiramente a Deus, por me dar a dádiva da vida.

Aos meus pais, Nilva e Valdir Weller, que sempre fizeram tudo por mim e me permitiram chegar até aqui.

À minha esposa Daniele e aos meus filhos Ana Clara e João Pedro, que foram minha fonte de inspiração para vencer os percalços do caminho.

Ao professor Neri Volpato, por me aceitar na orientação do doutorado, pela paciência, disponibilidade e suporte.

Ao professor Luiz Carlos de Abreu Rodrigues, meu amigo pessoal, por aceitar a co-orientação do doutorado, pela amizade, paciência, disponibilidade, resiliência e suporte.

Ao estagiário Marcus Reis, que auxiliou na programação.

À UTFPR e aos colegas de departamento que me deram toda a estrutura e apoio necessários.

Agradeço a você, leitor, por dedicar seu precioso tempo e atenção a este trabalho.

# RESUMO

WELLER, Tiago Rodrigues. ***Framework* para otimização dos movimentos de reposicionamento em manufatura aditiva por extrusão de material.** 2019. 121p. Tese (Doutorado em Engenharia) - Programa de Pós-graduação em Engenharia Mecânica e de Materiais, Universidade Tecnológica Federal do Paraná, Curitiba, 2019.

O tempo de fabricação é uma questão importante na manufatura aditiva baseada no princípio de extrusão de material, devido ao princípio de deposição e também a necessidade de um grande número de reposicionamento do cabeçote extrusor entre as trajetórias de deposição (contornos e trechos de *rasters*). O tempo gasto em reposicionamentos, geralmente chamado de tempo improdutivo, pode ser minimizado através da aplicação de algoritmos de otimização. Neste trabalho, o problema de otimização dos movimentos de reposicionamento é detalhado buscando aumentar o seu entendimento, propondo assim formas de reduzir sua complexidade computacional. A partir do entendimento do problema foi proposto um *framework* como método para estruturar, decompor e simplificar a sua solução. O *framework* foi dividido em quatro (4) etapas principais e foi idealizado de forma a permitir a aplicação de diferentes métodos de pesquisa operacional. A primeira etapa envolve alguns métodos inovadores para reduzir o tamanho do problema. Para testar o *framework* proposto as demais etapas foram implementadas, através de modelos de programação linear inteira mista (MILP) e metaheurísticas de busca tabu. Três casos de diferentes complexidades foram analisados e comparados com um algoritmo guloso clássico e um algoritmo de inserção do vizinho mais próximo com 2-opt. Os resultados mostraram que a estrutura foi eficaz em lidar com esse problema e, para os casos analisados, foi possível reduzir consideravelmente a distância de reposicionamento do cabeçote extrusor. A simplificação do problema para casos mais complexos ainda precisa ser analisada.

**Palavras Chaves:** Manufatura Aditiva. *Framework.* Otimização de movimentos de reposicionamento. Programação linear inteira mista. Busca tabu.

# ABSTRACT

WELLER, Tiago Rodrigues. **A Framework for Tool-Path Airtime Optimization in Material Extrusion Additive Manufacturing,** 2019. 121p. Thesis (PhD in Mechanical Engineering and Materials), Federal University of Technology - Parana. Curitiba, 2019.

Building time is an important issue in material extrusion-based additive manufacturing due to the deposition principle and because head repositionings are always required between deposition segments (contours and rasters). The time waste in head repositioning, usually referred to as non-productive time can be minimized by applying optimization algorithms. In this work, the problem of repositioning movements optimization is detailed in order to increase its understanding, thus proposing ways to reduce its computational complexity. From the understanding of the problem a framework was proposed as a method to structure, decompose and simplify its solution. The framework has been divided into four (4) main steps and is designed to be used with different optimization methods. The first step was designed with some innovative methods to reduce the problem size. Mixed integer linear programming (MILP) models and tabu search metaheuristics were implemented to test the proposed framework. Three cases of different complexities were analyzed and compared with a classic greedy algorithm and the nearest neighbor insertion algorithm with 2-opt. The results show that the framework was effective to deal with this problem and, especially for the analyzed cases, it was possible to reduce the repositioning distance considerably. The simplification of the problem for more complex cases still needs to be analyzed.

**Keywords:** Additive Manufacturing. Framework. Tool-path airtime optimization. Mixed integer linear programming. Tabu search.

# LISTA DE FIGURAS

## LISTA DE QUADROS

# LISTA DE TABELAS

## LISTA DE ACRÔNICAS E SIGLAS

ACO Algoritmo de colônia de formigas (*Ant Colony Optimization*)

AG Algoritmos genéticos

AM Manufatura Aditiva (*Additive Manufacturing)*

AMF *Additive manufacturing format*

B&B *Branch-and-Bound*

BT Busca Tabu (*Tabu Search* – TS)

C Contorno

CAD Desenho Auxiliado por Computador (*Computer-Aided Design)*

CAM Manufatura Auxiliada por Computador (*Computer-Aided Manufacturing)*

CNC Comando numérico computadorizado

CRS Segmento contínuo de *Raster (Continuos Raster Segment)*

FDM Modelagem por Fusão e Deposição (*Fused Deposition Modelling*)

GAMS *General Algebraic Modeling System*

GTSP Problema do Caixeiro Viajante Generalizado (*Generalized Tavelling Salesman Problem)*

HGA Algoritmo genético híbrido (*Hybrid Genetic Algorithm)*

LP Programação linear (*Linear Programming)*

LOM Manufatura laminar de objetos (*Layered Object Manufacturing*)

MILP Programação linear inteira mista (*Mixed Linear Integer Programming)*

PC Policarbonato

PLA Ácido polilático (*Polylactic Acid*)

SA *Simulated Annealing*

SGA Algoritmo genético simples (*Simple Genetic Algorithm*)

SOP *Sequential Ordering Problem*

STL *STereoLithography*

TSP Problema do caixeiro viajante (*Travelling Salesman Problem*)

VP Polígono de visibilidade (*Visibility Polygon*)

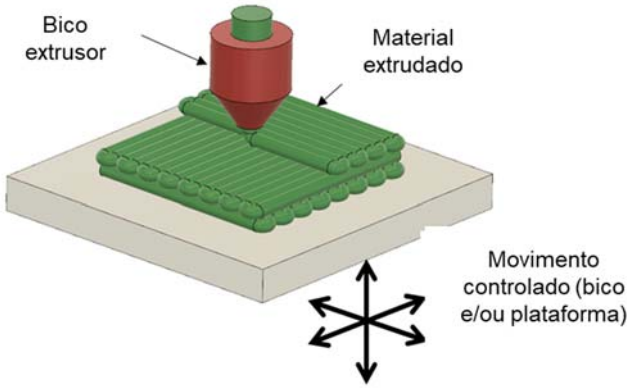
**SUMÁRIO**

# 1 INTRODUÇÃO

Os processos de Manufatura Aditiva (AM - *Additive Manufacturing*), comumente conhecidos como impressão 3D, constroem peças a partir da adição sucessiva de camadas. Com esta tecnologia, o processo de desenvolvimento de produto sofreu uma transformação significativa, podendo apresentar uma redução de custo e tempo em até 50% (JIN *et al*., 2011) se comparado aos processos tradicionais. Gibson *et al*. (2010) descrevem a AM como processo de fabricação que ocorre através da união de materiais para fabricar objetos diretamente a partir de um modelo geométrico digital tridimensional (3D), normalmente através do empilhamento de camadas.

Comparada aos processos de fabricação tradicionais (e.g. usinagem), a AM é considerada lenta (HAN *et al.,* 2003). Para tecnologias baseadas no princípio de extrusão de material (fundido ou pastoso), o tempo é um fator ainda mais relevante, pois, conforme ilustrado na Figura 1.1 a construção ocorre a partir de um cabeçote extrusor que deposita um fino filamento de material preenchendo as geometrias de cada camada (GIBSON *et al*., 2010).

Figura 1.1 – Princípio de funcionamento da manufatura aditiva por extrusão de material




Fonte: Volpato *et al*. (2017).

Adicionalmente, o preenchimento da camada não pode ser realizado de maneira contínua (AGARWALA *et al*. (1996); HABIB; KHODA (2017)), ou seja, o cabeçote deve interromper a extrusão e ser reposicionado em outra região para continuar o preenchimento. Sendo assim, o planejamento de trajetórias é uma tarefa

importante para o tempo de fabricação, pois define o caminho e a sequência de preenchimento em cada camada.

Nas tecnologias baseadas em extrusão de material, o preenchimento de cada camada normalmente é realizado de acordo com a seguinte sequência: *i*) contornos (Cs) (trajetória fechada com ponto de entrada e saída coincidentes) da camada e *ii*) preenchimento interno de cada contorno, normalmente através de uma estratégia que forma uma trajetória de ziguezague aberta, com pontos de entrada e saída distintos (e.g. formando trechos contínuos de *raster* – CRS). Sendo assim, a distância total percorrida com reposicionamentos é afetada pela geometria e tamanho da peça, tipo de preenchimento, quantidade de Cs presentes na camada e ainda o número total de camadas.

Na literatura encontram-se diferentes abordagens utilizadas na otimização de movimentos de reposicionamento nos processos de AM. Na otimização de uma camada contendo apenas Cs ou trechos abertos (semelhantes a CRSs) foram encontrados trabalhos utilizando algoritmos genéticos, adaptações de *solvers* do problema do caixeiro viajante, inserção do vizinho mais próximo, guloso e máxima inserção linear (TANG; PANG, 2003; WEIDONG, 2009; GANGANATH *et al.,*2016; FOK *et al*., 2016a; FOK *et al*., 2016b). Considerando a otimização entre Cs e CRSs para uma camada encontra-se trabalhos utilizando os algoritmos guloso e inserções dos pontos mais próximo e mais distante (VOLPATO *et al*., 2007a; VOLPATO *et al.,* 2013). Na otimização entre Cs e CRSs para múltiplas camadas, encontra-se trabalhos utilizando heurísticas desenvolvidas especificamente para o problema, algoritmo de colônia de formigas e *solvers* do problema do caixeiro viajante adaptados (FOK *et al*., 2019; IORI; NOVELLANI, 2019). O trabalho mais completo considerando a otimização entre Cs, CRSs, múltiplas camadas e movimentos de limpeza do cabeçote extrusor foi proposto utilizando os algoritmos de inserção do mais próximo com 2-opt e algoritmo guloso (VOLPATO *et al*., 2019). Cabe destacar que os trabalhos encontrados que se preocuparam na redução da complexidade computacional foram os propostos por Fok *et al.* (2016b) (considerando somente Cs e uma única camada) e uma das heurísticas de Iori e Novellani (2019) (considerando Cs, CRSs e múltiplas camadas).

Nenhum dos trabalhos revisados preocupou-se com o detalhamento do problema e a formalização de um método estruturado capaz de permitir sua solução através de diferentes métodos de pesquisa operacional.

## 1.1 OPORTUNIDADE DE PESQUISA

Na revisão bibliográfica não foram encontrados trabalhos focados em entender e detalhar o problema de otimização das distâncias de reposicionamento em AM por extrusão de material visando obter vantagem no momento de modelar o problema. Sendo assim, identificou-se a oportunidade de estudar mais detalhadamente o problema, aumentar o seu entendimento e formalizar uma estrutura (*framework*) contendo métodos inovadores e inéditos na sua decomposição e redução da complexidade computacional. Pretende-se desta forma, permitir a solução de diferentes instâncias do problema e de maneira independente do método de pesquisa operacional escolhido.

## 1.2 HIPÓTESE E OBJETIVOS

A principal hipótese desta pesquisa sugere que a decomposição e simplificação da complexidade computacional do problema de otimização dos movimentos de reposicionamento nos processos de AM por extrusão de material permitirá obter soluções melhores que as já conhecidas para os problemas propostos por Volpato et al. (2019).

### 1.2.1 Objetivo Geral

O objetivo geral deste trabalho é detalhar, estruturar, decompor e simplificar o problema de otimização dos movimentos de reposicionamento no processo de AM por extrusão de material, propondo formas de reduzir a sua complexidade computacional, e criar um framework que permita a sua solução, seguindo os passos propostos, através de diferentes técnicas de pesquisa operacional.

### 1.2.2 Objetivos Específicos

a) Analisar e detalhar o problema de otimização dos movimentos de reposicionamento em processos de AM por extrusão de material visando entender sua complexidade;

b) Propor alternativas de reduzir a complexidade computacional do problema através de pré-processamento dos dados e outros tratamentos durante o processamento;
c) Desenvolver um *framework* (método) para a solução do problema, estruturado com base na decomposição e simplificação proposta;
d) Propor modelos de programação linear inteira mista e algoritmos de busca tabu para a solução do problema dentro do *framework* proposto;
e) Analisar os resultados gerais comparando com casos de literatura.

## 1.3 JUSTIFICATIVA

Considerando que a tecnologia AM baseada no princípio de extrusão de material é uma das mais lentas, a redução do tempo gasto na construção de peças é um assunto relevante. Um dos pontos que contribui para este fato é a existência de vários movimentos de reposicionamento do bico extrusor durante o processo. Observa-se que cada camada pode conter dezenas de Cs e CRSs e cada peça pode ter centenas ou até milhares de camadas. Sendo assim, cada pequena redução de tempo por camada pode resultar em uma significativa redução ao final do processo. Este é o principal motivo para se implementar a otimização das distâncias de reposicionamento nos processos de AM. Como benefícios gerais da otimização de movimentos de reposicionamento pode-se destacar:

- Redução do tempo de fabricação, ou seja, aumento de produtividade;
- Aumento da disponibilidade do equipamento;
- Redução no custo de mão de obra especializada;
- Redução no consumo de energia;
- Redução nas manutenções do equipamento.

## 1.4 ORGANIZAÇÃO DO TRABALHO

Esta tese está organizada conforme segue: No Capítulo 1, apresenta-se a introdução ao tema, oportunidade de pesquisa, hipótese, objetivos e justificativa. No Capítulo 2, apresenta-se uma revisão da literatura sobre métodos de otimização e manufatura aditiva, seguido dos trabalhos relacionados ao tema. No Capítulo 3 são apresentados, propostos e justificados os materiais e métodos utilizados no trabalho.

No Capítulo 4 são apresentados os modelos implementados na solução do problema. O Capítulo 5 apresenta os resultados obtidos e discussões. Finalmente, o Capítulo 6 apresenta as conclusões e propostas de trabalhos futuros.

# 2 LEVANTAMENTO DO ESTADO DA ARTE

Neste capítulo são abordadas as informações da literatura que são relevantes para o desenvolvimento do trabalho. Inicialmente o problema da otimização de rotas é descrito e então os principais métodos de otimização que podem ser aplicados são revisados. Na sequência, as características dos processos de Manufatura Aditiva (AM) por extrusão de material foram detalhadas e foram revisados alguns trabalhos e suas técnicas de otimização que podem servir de inspiração na solução do problema estudado neste trabalho. Finalmente, encerra-se o capítulo com as considerações finais sobre a revisão bibliográfica.

## 2.1 PROBLEMA DO CAIXEIRO VIAJANTE (*TRAVELLING SALESMAN PROBLEM* – TSP)

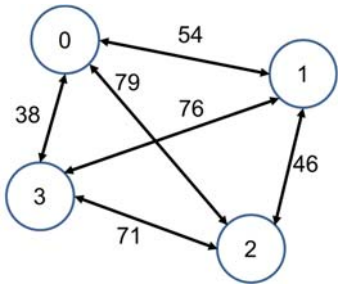
Um dos clássicos e mais estudados problemas de otimização de rotas é conhecido como O Problema do Caixeiro Viajante (*Travelling Salesman Problem* – TSP). Este problema consiste em tenta encontrar a menor rota para percorrer várias cidades (visitando cada uma delas uma única vez) e retornar à cidade de origem. Sua inspiração parte da necessidade dos vendedores realizarem entregas em diferentes locais (cidades) percorrendo a menor distância possível, reduzindo o tempo de deslocamento e custos com transporte e combustível. A formulação do TSP é muito aplicada em casos que tratam da otimização de rotas, como os problemas de roteamento de veículos, trajetórias de robôs, manufatura e planejamento da produção (WOEGINGER, 2003). Podem ser encontradas diversas formulações similares ao TSP que poderiam ser adaptadas à problemas de otimização de rotas, como o TSP generalizado (CASTELINO; SOUZA; WRIGHT, 2003), *Clustered* TSP (LAPORTE; PELEKAR, 2002), *Rural Postman Problem* (FOK *et al.,* 2019), *Open Vehicle Routing Problem* (LETCHFORD; LYSGAARD; EGLESE, 2007), entre outras.

Tratando-se de otimização de rotas, os deslocamentos entre cidades realizados no TSP podem também ser relacionados aos movimentos de reposicionamento existentes nos processos de AM. Baseados nisso, diferentes trabalhos como os apresentados por Khan, Hayhurst e Cannings (1999), Wah *et al.* (2002), Tang e Pang (2003), Castelino, Souza e Wright (2003), Woeginger (2003), Fok

*et al.* (2016), Volpato *et al.* (2007b, 2013 e 2019), formularam o problema de otimização de movimentos de reposicionamento em AM de maneira correlata ao TSP.

No TSP a representação das cidades pode ser realizada na forma de grafo, onde cada nó (*n*) representa uma cidade formando uma matriz *n x n* onde $c_{ij}$ é a distância (custo) de deslocamento entre os nós *i* e *j.* Na Figura 2.1 observa-se a construção do grafo de maneira simplificada com quatro nós. Cada nó *n* forma um vértice e o comprimento das arestas ligando estes vértices representa o custo de deslocamento.

Figura 2.1 – Grafo de um TSP com quatro nós




Fonte: Adaptado de Woeginger (2003).

Castelino, Souza e Wright (2003) colocam que o TSP pode ser formulado como um problema de programação linear inteira (PLI), ou seja, onde todas as variáveis pertencem ao conjunto dos números inteiros. Considerando variáveis $c_{ij}$ associadas com cada arco (*i, j*), ou seja, o custo de ir de *i* até *j*. Quando a ligação entre dois pontos é possível, atribui-se que a variável binária $x_{ij}$ = 1, ou seja, o nó *i* está ligado ao *j*, ou $x_{ij}$ = 0 em caso contrário.

A formulação é dada pela função objetivo mostrada na Equação 2.1, visando minimizar *C*.

$$C = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

Castelino, Souza e Wright (2003) colocam que o TSP é formado por um ciclo hamiltoniano, ou seja, um caminho que deve passar por todas as cidades (pontos no grafo) sem repetir nenhuma além do ponto final e inicial. Outro problema que pode ocorrer na solução do TSP é a ocorrência de subciclos, ou seja, caminhos fechados que não conectam todas as cidades, necessitando a imposição de restrições para

evitar esse problema. Sendo assim, o TSP pode ser um problema não polinomial completo, ou seja, que pode ser de difícil solução devido à complexidade computacional, tornando muito difícil achar uma solução ótima para problemas grandes em um tempo razoável de computação. Nestas situações é mais viável utilizar-se de heurísticas capazes de chegar a uma solução subótima em um tempo razoável.

Woeginger (2003) e Khan, Hayhurst e Cannings (1999) colocam que um TSP é considerado simétrico quando o custo de movimentação entre dois nós diferentes é igual, dado pela Equação 2.2:

$$c_{ij} = c_{ji} \tag{2.2}$$

Isto significa que o custo de movimentação entre dois nós é o mesmo em ambos os sentidos, ou seja, o custo de deslocar-se do nó *i* para *j* ou de *j* para *i* é o mesmo. Essa condição indica que podem existir soluções simétricas, ou seja, de igual custo e que não precisam ser solucionadas novamente. Dessa forma, pode-se reduzir consideravelmente o número de soluções viáveis formando um grafo não direcional (WOEGINGER, 2003). Mas, para outros casos que possuem restrições de precedência, ou seja, restrições que não permitem certos movimentos, o TSP é considerado assimétrico, sendo definido pela Equação 2.3:

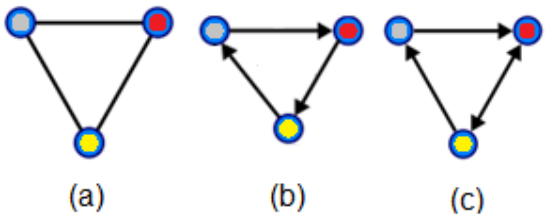
$$c_{ij} \neq c_{ji} \tag{2.3}$$

Nas formulações com precedências existem restrições quanto aos caminhos que podem não existir em ambas as direções ou os custos de movimentação entre os nós *i* para *j* ou de *j* para *i* são diferentes formando um grafo direcional (Figura 2.2). Esta situação é encontrada quando o movimento é proibido em um sentido (e.g. rotas que passam por ruas de mão única).

Na Figura 2.2a verifica-se o grafo não direcional característico do TSP simétrico e na Figura 2.2b consta o grafo direcional representando o TSP assimétrico. Uma operação de otimização para um problema complexo pode ocorrer com trajetórias entre nós que formam trechos assimétricos e simétricos, gerando neste

caso um TSP considerado misto (simétrico/assimétrico) conforme ilustrado na Figura 2.2c (WOEGINGER, 2003).

Figura 2.2 – Grafo de TSP com três nós: simétrico (a); assimétrico (b); misto (c)




Fonte: Adaptado de Woeginger (2003).

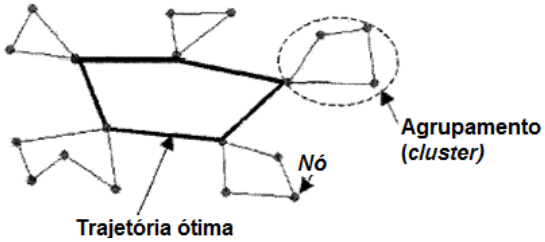
Os movimentos de reposicionamento em AM podem ser aproximados de um TSP simétrico, pois não há restrição quanto ao sentido do movimento. Tradicionalmente, o TSP pode ser tratado das seguintes maneiras (VOLPATO *et al.* (2019), CASTELINO; SOUZA; WRIGHT, 2003):

- Algoritmos de busca de soluções exatas, que funcionam bem para problemas de baixa complexidade.
- Algoritmos de heurísticas, os quais procuram soluções subótimas para problemas médios e complexos com um tempo de busca razoável.
- Casos especiais, onde o problema principal é resolvido em subproblemas menores de forma a facilitar a resolução.

TSP generalizado (GTSP)

Nesta variação do TSP, o caixeiro viajante visita regiões (com várias cidades), como foi aplicado no problema proposto por Castelino, Souza e Wright (2003). Neste problema, *n* cidades são agrupadas em *m* distritos (*clusters*) onde, para a análise ocorre a visita de apenas uma cidade em cada distrito (uma vez no distrito, as demais cidades são percorridas) e, posteriormente, ocorre o retorno para a posição inicial, conforme Figura 2.3.

Figura 2.3 – Esquema proposto pelo GTSP



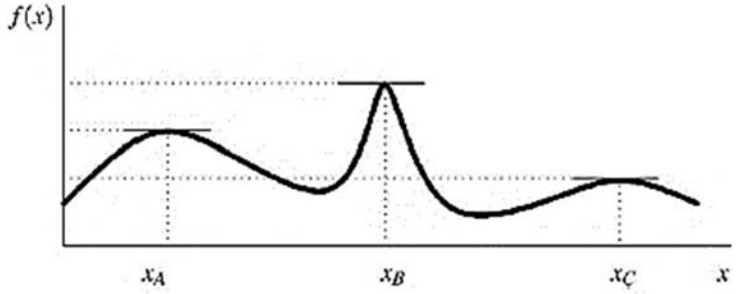
Fonte: Adaptado de Castelino; Souza; Wright (2003).

A AM por extrusão de material pode ser tratada como um caso especial de GTSP, pois as ilhas que deverão ser visitadas e depositadas podem ser associadas aos *clusters* desta formulação.

## 2.2 OTIMIZAÇÃO

A otimização de problemas consiste em buscar a melhor solução possível sem ter que testar todas as possibilidades envolvidas (LOPES; RODRIGUES; STEINER, 2013). Problemas de otimização são caracterizados por situações em que se deseja maximizar ou minimizar uma função numérica de várias variáveis, num contexto em que podem existir restrições. Tanto as funções como as restrições dependem dos valores assumidos pelas variáveis do modelo ao longo do procedimento de otimização. Na Figura 2.4, é apresentada uma função *f(x)* onde a função objetivo consiste em encontrar o valor para *x* onde *f(x)* alcança o valor máximo.

Figura 2.4 – Conjuntos De Soluções Ótimas Locais ($X_A$ e $X_C$ e global ($X_B$)




Fonte: Maringer (2005).

Como pode ser observado, os pontos $x_A$, $x_B$ e $x_C$ são as soluções ótimas locais. Entretanto, apenas $x_B$ é o ótimo global porque proporciona o maior valor global para a função objetivo, enquanto $x_A$ e $x_C$ são apenas ótimos locais. Segundo Gilli e Winker (2012), muitas vezes quando o espaço de busca é considerado complexo (e.g. muitos mínimos locais), torna-se difícil determinar se uma solução identificada é a ótima local ou global. Segundo Kargupta e Goldberg (1995), os algoritmos utilizados em otimização são classificados em duas categorias principais: métodos exatos (determinísticos) e métodos heurísticos (probabilísticos).

### 2.2.1 Métodos de Otimização Exatos

Os algoritmos exatos são aqueles que, dada uma certa entrada, ela produzirá sempre a mesma saída (HILLIER; LIEBERMAN, 2013). O algoritmo executa uma função matemática, normalmente seguindo um mesmo caminho de uma entrada até uma saída, tornando-se dependente do ponto de partida inserido. Os algoritmos exatos convergem para a solução ótima para qualquer instância do problema. Por outro lado, o tempo de processamento pode sofrer um crescimento exponencial com o aumento do problema, onde, normalmente apenas problemas de tamanho pequeno (e.g. TSP com poucos pontos) podem ser resolvidos num intervalo de tempo razoável.

### 2.2.2 Programação Linear – PL (*Linear Programming* – LP)

A PL é uma técnica de modelagem que usa expressões matemáticas que permite buscar a solução ótima para um problema de pesquisa operacional, onde a denominação linear refere-se às características das equações envolvidas na formulação do problema (ALMEIDA *et al.*, 2003; BRADLEY; HAX; MAGNANTI, 1977). A resolução de um modelo formulado via PL pode ser realizada, por exemplo, pelo algoritmo *Simplex* ou método dos pontos interiores.

Definições relacionadas a PL:

- Função objetivo: expressão matemática de função linear contendo as variáveis que serão otimizadas (maximizadas ou minimizadas) (ALMEIDA *et al.,* 2003);
- Variáveis de decisão: incógnitas definidas para a resolução do problema;
- Restrições: são equações ou inequações lineares que o método de solução deve respeitar na otimização do modelo;
- Região ou solução viável: conjunto que representa as combinações admissíveis de variáveis para otimização do problema (RITZMAN; KRAJEWSKI, 2004);
- Parâmetro: valor informado na implementação do problema e que não se altera quando a solução é implementada.

Os passos fundamentais na formulação do modelo são descritos por Lisboa (2002):

- Definição da função objetivo;

- Identificação das variáveis envolvidas no problema;
- Identificação das restrições que afetam as variáveis.

Hillier e Lieberman (2013) definem a PL por meio da formulação indicada na Equação 2.4:

$$\text{Minimize (ou Maximize) } c'x \tag{2.4}$$

Sujeito a:

$$Ax = b \; (\; Ax \geq b \text{ ou } Ax \leq b)$$

$$x \geq 0$$

O $c$ (custo) é um vetor linha $n$ dimensional de fatores de ponderação (pesos); $x$ é um vetor coluna $n$ dimensional de variáveis do modelo; $A$ é uma matriz $m$ por $n$; $b$ é um vetor coluna $m$ dimensional ($m$ define o número de equações do modelo); e $x \geq 0$ refere-se à restrição de não negatividade. Quando a PL possuir somente variáveis inteiras o modelo é caracterizado como programação inteira pura, ou, caso contrário, é caracterizado como programação inteira mista.

Hillier e Lieberman (2013) descrevem que dentre as várias aplicações práticas da PL pode-se citar: planejamento agregado de produção; otimização do fluxo produtivo; otimização do processo de produção; análise de produtividade; otimização do roteamento de veículos; balanceamento de linhas de produção, entre outros.

### 2.2.3 Programação Linear Inteira Mista - PLIM (*Mixed Integer Linear Programming* – MILP)

Taha (2014) cita que os modelos formulados como MILP definem que certas variáveis presentes no problema deverão assumir apenas valores inteiros (podendo ser variáveis binárias), conforme apresentado pela Equação 2.5.

$$\text{Minimize (ou Maximize) } c'x + hy \tag{2.5}$$

Sujeito a:

$$Ax + Gy = b$$

$$x \geq 0$$

$$y \in Z_+$$

Onde, $c$ é um vetor linha $n$ dimensional de ponderação (pesos); $x$ é um vetor coluna $m$ dimensional de variáveis do modelo; $A$ é uma matriz $m$ por $n$; $h$ é um vetor linha $p$ dimensional de pesos; $y$ é um vetor coluna $p$ dimensional de variáveis que assumem valores inteiros; $G$ é uma matriz $m$ por $p$; $b$ é um vetor coluna $m$ dimensional ($m$ define o número de equações do modelo); e $x \geq 0$ impõe uma restrição de não negatividade.

Arenales *et al.* (2006) citam que a diferença entre o número de variáveis ($n + p$) e o número de equações ($m$) do modelo é conhecida por graus de liberdade apresentados pelo sistema. Modelos MILP podem demandar uma carga computacional elevada, pois normalmente aplicam métodos de busca em árvore (e.g. *Branch-and-Bound* – B&B) associados a algoritmos *Simplex* em cada nó da árvore. Arenales *et al.* (2006) citam que o algoritmo *Simplex* é um método algébrico e iterativo capaz de fornecer a solução exata para problemas de PL geralmente em um número finito de iterações. O B&B é um método clássico de busca exata que consiste em dividir um problema inicial em subproblemas menores, de forma que a solução possa ser obtida através da solução destes subproblemas (GOLDBARG; LUNA, 2005).

### 2.2.4 Métodos Heurísticos

Lopes, Rodrigues e Steiner (2013) citam que a melhor opção na solução de problemas considerados complexos é através de metaheurísticas, onde, abre-se mão da obtenção de soluções ótimas em troca de obter-se soluções subótimas em um tempo reduzido. Os algoritmos metaheurísticos realizam a avaliação da função objetivo e parâmetros estocásticos onde, para uma determinada entrada aleatória, tendem a apresentar comportamentos, tempos e resultados diferentes em diferentes execuções, ao contrário dos algoritmos exatos (HOOS; STÜTZLE, 2004). Os mais conhecidos podem ser representados principalmente por: Algoritmos Genéticos (AG), Colônia de Formigas (ACO), *Simulated Annealing* (SA), Busca Tabu (BT), entre outros.

Blum e Raidl (2016) descrevem que as principais vantagens dos métodos metaheurísticos são:

- A função objetivo e as restrições não precisam, necessariamente, ser explicitadas por uma representação matemática;
- Podem trabalhar com variáveis contínuas, discretas ou ambas combinadas;

- Podem ser mais simples de implementar que métodos exatos;
- Não há restrição alguma quanto ao ponto de partida dentro do espaço de busca da solução;
- Podem trabalhar com buscas simultâneas dentro do espaço de buscas através de uma população de indivíduos;
- Podem trabalhar com um grande número de variáveis (com incremento do custo computacional).

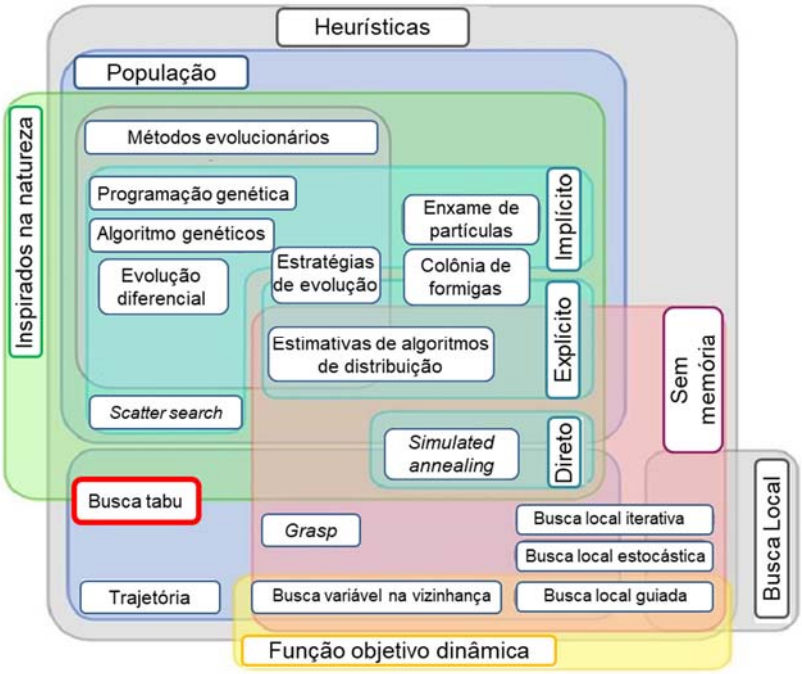
Como desvantagens, as metaheurísticas podem apresentar um custo computacional alto caso não sejam realizados procedimentos de “limpeza” dos dados utilizados, pois normalmente trabalham com grande número de variáveis e possibilidades dentro do espaço de busca. Segundo Lopes, Rodrigues e Steiner (2013) as metaheurísticas são consideradas técnicas de otimização confiáveis e possuem aplicações nos mais diferentes campos de engenharia e de outras ciências. Porém, estas técnicas podem apresentar algumas dificuldades numéricas e problemas de robustez relacionados com: a falta de continuidade das funções a serem otimizadas ou de suas restrições, funções não convexas, multimodalidade, existência de ruídos nas funções, necessidade de se trabalhar com valores discretos para as variáveis, existência de mínimos ou máximos locais, entre outros (DREO *et al.,* 2006).

### 2.2.5 Classificação das Heurísticas

Dreo *et al.* (2006) sugerem uma diferenciação entre heurísticas e metaheurísticas, onde colocam que de forma geral as heurísticas de otimização são desenvolvidas para resolução de problemas específicos, não sendo possível aplicá-las a outras classes de problemas. As metaheurísticas seriam estratégias de alto nível para a exploração do espaço de busca utilizando-se de diferentes métodos de diversificação e intensificação, pois possuem uma estrutura genérica que pode ser aplicada a uma grande gama de diferentes problemas. Na literatura as definições de heurísticas e metaheurísticas se confundem e são usadas como sinônimos, mas verifica-se uma tendência de chamar os algoritmos mais completos com randomização, busca local e hibridização como metaheurísticos, sendo aplicados principalmente para uma estrutura de otimização considerada genérica, isto é, que pode ser utilizada para diferentes problemas.

Na literatura existem diferentes classificações de metaheurísticas. Por exemplo, Dreo *et al.* (2006) propuseram uma classificação de acordo com os seguintes aspectos: computação evolucionária, populacional, inspiradas na natureza, sem uso de memória, com função objetivo dinâmica, implícita, explicita ou direta, conforme ilustra a Figura 2.5.

Figura 2.5 – Classificação das metaheurísticas proposta por Dreo *et al.* (2006)




Fonte: Adaptado de Dreo *et al.* (2006).

### 2.2.6 Métodos Baseados em População

Segundo Davis (1991), estas abordagens trabalham simultaneamente com conjuntos de soluções chamadas de população. Estes métodos podem ser mais eficientes para exploração global do problema em busca de soluções, porém aumentando o custo computacional e complexidade da estrutura de implementação. Os principais métodos que podem ser citados baseados em população são os algoritmos genéticos (AG) e colônia de formigas (ACO).

Koza (1997) descreve que os AGs são usados quando se necessita uma ferramenta computacional na busca de soluções para problemas complexos, como aprendizagem de máquinas, planejamento da movimentação de robôs, reconhecimento de padrões, detecção de imagem, entre outros. O AG é uma metaheurística onde uma população de indivíduos evolui baseada na seleção natural.

Dorigo e Stützle (2004) descrevem que os ACOs foram inspirados no comportamento organizado e coletivo baseados em feromônios que as formigas apresentam na busca por alimento. Encontram boa aplicação em problemas dinâmicos, cujas características se alteram ao longo da execução do algoritmo, mas, por outro lado, apresentam alto custo computacional.

### 2.2.7 Métodos de Busca Local

A implementação da busca local utiliza apenas informações sobre as soluções na vizinhança de uma solução atual (MLADENOVIĆ; HANSEN,1997; AARTS; LENSTRA, 2003). O método clássico de busca local para minimizar uma função objetivo dada por $f(x)$ é formalizado a seguir:

1. Gerar uma solução inicial $x^c$
2. Enquanto o critério de parada não for alcançado faça
3. Selecione $x^n \in N(x^c)$ (vizinha da solução atual)
4. se $f(x^n) < f(x^c)$ então $x^n = x^c$
5. fim enquanto

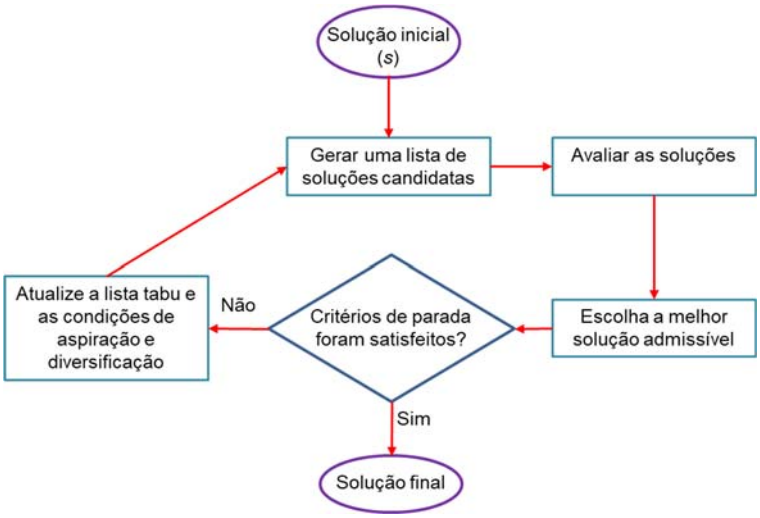
Os mecanismos de aceitação da solução mostrados na linha quatro são específicos de cada algoritmo, definindo como a busca é realizada dentre o espaço das soluções. O critério de parada adotado normalmente é baseado no número de iterações sem melhora na função objetivo. Dentre os principais métodos de busca local pode-se citar (LOURENÇO; MARTIN; STÜTZLE, 2003; KIRKPATRICK; GELLAT; VECCHI, 1983): algoritmo guloso (*greedy algorithm*); inserção do mais próximo (*Nearest Insertion* - NI); *k*-opt (2-opt e 3-opt); s*imulated* a*nnealing*; busca tabu; entre outros.

### 2.2.8 Busca Tabu – BT (*Tabu Search* - TS)

Segundo Glover e Laguna (1998), a BT é uma metaheurística que adquire e faz o uso de informações do histórico de busca (memória) de forma a tomar decisões melhores durante o processo iterativo de busca na vizinhança da solução vigente. É particularmente empregada em espaços de busca discretos, onde existe um número

finito de soluções vizinhas. A partir de uma solução vigente realiza-se uma pesquisa por soluções melhores (soluções elite) na vizinhança desta solução, empregando-se restrições (Lista Tabu) para impedir certos movimentos (ou soluções) por um número definido de iterações. Vizinhança é o conjunto de soluções possíveis de serem geradas a partir de uma solução dada. Caso não ocorram movimentos que melhorem a solução em relação à solução vigente, a BT poderá realizar um movimento que degrada menos o valor da função objetivo. O fluxograma básico de BT é apresentado na Figura 2.6.

Figura 2.6 – Fluxograma básico da metaheurística de BT




Fonte: Adaptado de Mohandessi (2017).

Entre outras aplicações, a BT é aplicada na otimização de sistemas de produção, otimização de roteamento de veículos, problemas de arranjo físico e otimização de operações de fabricação.

A BT apresenta três princípios: *i)* uso de uma estrutura de dados (lista) para memorizar o histórico do processo de busca; *ii)* uso de um mecanismo de controle para fazer um balanceamento entre a aceitação, ou não, de uma nova configuração, com base nas informações registradas na lista tabu referentes às restrições e aspirações desejadas e *iii*) incorporação de procedimentos que alternam as estratégias de diversificação e intensificação.

O pseudocódigo básico de BT (GLOVER; LAGUNA, 1998) é apresentado na sequência.

**1.** Enquanto o critério de parada da Diversificação não é atingido, faça:
**2.** Gerar uma solução inicial (*s*)
***3.*** $s^* = s;\ s^{best} = s;$
**4.** Enquanto o critério de parada da Intensificação não é encontrado, faça:
**5.** Gerar a vizinhança de *s* e selecionar a melhor solução vizinha (*s'*)
***6.*** $s = s'$
**7.** Se *s'* é melhor que *s** então
***8.*** $s^* = s'$
**9.** Se *s** é melhor que $s^{best}$ então
***10.*** $s^{best} = s^*$
***11.*** Retornar $s^{best}$

Na linha 1, o algoritmo será executado até que o critério de parada definido seja atingido (normalmente definido pelo número máximo de iterações sem melhora na função objetivo). No final do algoritmo o valor da melhor solução encontrada fica armazenada na variável $s^{best,}$ apresentado na linha 11. Na linha 2 é gerada a solução inicial, o que sempre ocorre quando o algoritmo é iniciado ou é realizada uma diversificação. Então, na linha 3, a solução armazenada na variável *s* é atribuída à variável *s** (solução estrela). Na linha 5 faz-se a busca na vizinhança e são geradas as soluções vizinhas da solução vigente *s*. Na linha 6 a melhor solução vizinha encontrada é atribuída à variável *s*'. Na linha 7 é verificado se a solução *s*' é melhor que a solução *s**. Caso afirmativo, na linha 8 a solução armazenada em *s*' é atribuída para *s**. O laço entre as linhas 4 e 8 é executado até que o critério de parada da intensificação ocorra. Este critério é um número máximo de iterações sem que *s** seja melhorado. Ao sair do laço entre as linhas 4 e 8, verifica-se na linha 9 se a solução *s** é melhor que a melhor solução encontrada até o momento. Se *s** é melhor então na linha 10 é armazenada na variável $s^{best}$.

Lista Tabu

A BT trabalha com estruturas de memórias entre as quais a mais usada é a Lista Tabu, onde ficam armazenados os últimos movimentos realizados, com a função de evitar que a busca fique presa em pontos de mínimo ou máximo local, provocando a realização de ciclos. Os movimentos armazenados na Lista Tabu ficam proibidos por um determinado número de iterações, sendo esse número normalmente

relacionado com o número de movimentos possíveis a partir da nova solução vigente e pode ainda ser relacionada com o número total de iterações do algoritmo (GLOVER; LAGUNA, 1998).

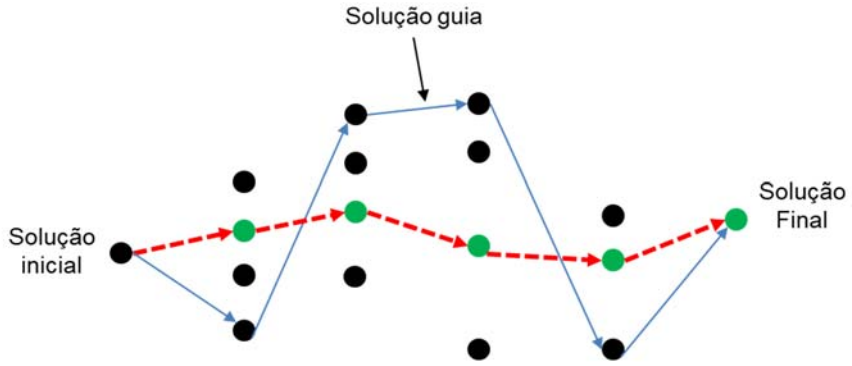
Busca de Soluções Vizinhas

Cada solução $x \in X$ , onde *X* representa o conjunto das soluções que podem ser geradas, têm um conjunto associado de soluções vizinhas $V(x) \subset X$ , que são chamadas de soluções vizinhas a *x*. Toda solução vizinha $x' \in V(x)$ pode ser gerada a partir de *x* através da realização de um movimento. Na BT, soluções vizinhas são simétricas, ou seja, *x*' é solução vizinha a *x* se e somente se *x* é solução vizinha a *x*'.

Diversificação e Intensificação

Diversificação e intensificação são estratégias que evitam que a busca fique presa ou deixe de explorar algumas regiões. A intensificação permite incrementar o esforço de busca em uma vizinhança considerada promissora. Uma técnica clássica de intensificação da busca na vizinhança que pode ser aplicada a partir de uma solução preliminar (solução guia) é o procedimento de *path-relinking* (reconexão dos caminhos). A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o melhor movimento é selecionado, ou seja, explora caminhos no espaço de soluções vizinhas realizando movimentos que melhorem a solução guia, conforme ilustrado pelas setas tracejadas na Figura 2.7.

Figura 2.7 – Caminho obtido após a busca de soluções vizinhas a partir de uma solução guia




Fonte: Adaptado de Gendreau; Potvin (2010).

Durante a intensificação, espera-se por melhoras frequentes na solução vigente. Caso a busca fique “presa” em uma região, ou seja, a realização de várias

iterações sem melhora da função objetivo realiza-se uma diversificação, onde a busca é deslocada para outra região do espaço de busca. Por isso, a diversificação permite que o algoritmo faça uma busca mais abrangente em todo o universo de soluções que podem ser obtidas, evitando a formação de ciclos (*loops*).As formas clássicas de diversificação são a geração de nova solução aleatória e/ou a realização de vários movimentos de troca (*swap*) consecutivos dentro da solução já obtida.

Critérios de Aspiração

Paul (2010) cita que na geração de uma solução vizinha, não pode ser realizado um movimento tabu, ou seja, contido na lista. Mas, podem ser definidos critérios que definem quando um movimento tabu pode ser quebrado, ou seja, é realizado um movimento proibido. Este critério é o “critério de aspiração” e é utilizado se a solução formada por um determinado movimento proibido for melhor que a melhor solução encontrada até o momento (GLOVER; LAGUNA, 1998).

Busca Tabu Reativa

Na BT reativa faz-se uso de uma lista tabu dinâmica, ou seja, de tamanho variável. Pode-se ainda penalizar a repetição de soluções, evitando que a busca fique presa em regiões de mínimo ou máximo local e prevenindo a ocorrência de ciclos. A mudança no tamanho da lista tabu pode ser obtida após um dado número de repetições da mesma solução e caso essa situação ocorra, executa-se o procedimento de diversificação, forçando a busca a sair da região que está gerando soluções repetidas.

### 2.2.9 Métodos Híbridos

Blum e Raidl (2016) citam que uma forma de melhorar a performance de busca da solução de problemas de otimização é a implementação de algoritmos híbridos, ou seja, realizar a combinação entre diferentes metaheurísticas ou ainda algoritmos que misturam métodos exatos com heurísticos (*matheuristics*), aproveitando vantagens específicas que se complementam. A combinação destas abordagens pode resultar no incremento do desempenho quanto ao tempo de processamento e a qualidade da solução obtida. Uma heurística é considerada híbrida quando é composta por diferentes algoritmos ou ainda quando são combinadas com técnicas clássicas de

otimização, como a programação inteira. Assim, permite o surgimento de um grande número de novas técnicas motivadas pela necessidade de alcançar um bom equilíbrio entre as capacidades de uma metaheurística para explorar o espaço de busca e a possibilidade de explorar a experiência acumulada durante a pesquisa.

O nível de combinação dos algoritmos pode ocorrer de duas maneiras:

- Alto nível – Nesta combinação, cada algoritmo trabalha de forma individual, mantendo suas características internas originais, e a hibridização ocorre através de uma interface de troca de dados.
- Baixo nível - Nesta combinação os algoritmos atuam de forma interligada, com uma forte dependência entre eles através de funções ou componentes compartilhados.

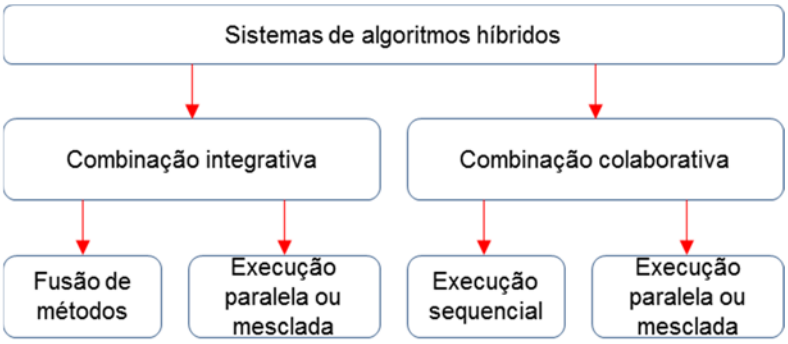
Blum e Raidl (2016) citam que a ordem de execução dos algoritmos híbridos é outro fator que deve ser levado em consideração no seu funcionamento. Uma das abordagens é a implementação do modelo de forma sequencial, onde um algoritmo passa as informações para o outro. Como um exemplo, pode-se citar o uso de um algoritmo de pré-processamento dos dados iniciais que repassará o seu resultado para outro algoritmo que efetivamente realizará a otimização do problema, conforme abordado nas implementações deste trabalho. Outra abordagem de implementação de algoritmos híbridos ocorre através de modelos paralelos, onde os algoritmos atuam ao mesmo tempo e necessariamente de maneira mais integrada.

Puchinger e Raidl (2005) citam ainda que as estratégias de controle são outra forma de se classificar os algoritmos híbridos. Esta combinação entre os algoritmos pode ocorrer de forma integrativa (coerciva) ou colaborativa (cooperativa), que, por sua vez podem ser executadas de forma sequencial ou paralela, conforme ilustrado na Figura 2.8.

Nas implementações onde as combinações ocorrem de forma integrativa, um método é subordinado a outro, funcionando como um componente do outro método. Por outro lado, nas combinações colaborativas surge a questão de qual é o espaço de busca que será explorado por cada algoritmo, pois, embora ocorra troca informações entre eles, um não faz parte do outro.

Raidl (2006) cita que no caso de problemas grandes, realizar uma decomposição em problemas menores é essencial para se obter uma solução de maneira satisfatória dentro de um limite de tempo aceitável.

Figura 2.8 – Classificação de sistemas híbridos de acordo com a estratégia de controle




Fonte: Adaptado de Puchinger; Raidl (2005).

Na prática, pode-se realizar a decomposição do problema de maneira implícita, obtendo-se diferentes soluções iniciais, valores de parâmetros, etc., e explícita, onde cada algoritmo atua exclusivamente num espaço de busca determinado.

### 2.2.10 Recomendações Gerais Para O Uso Das Metaheurísticas De Otimização

Marti e Reinelt (2011) colocam que as primeiras perguntas a serem respondidas para uma aplicação específica é se deve-se usar uma metaheurística de otimização para o problema inteiro e em caso afirmativo, qual empregar. Desta forma, fica difícil responder ambas as questões com uma resposta geral. No caso de problemas complexos computacionalmente, os autores recomendam aplicar procedimentos clássicos de otimização como um *benchmark* às metaheurísticas. Neste caso, se em poucas iterações as metaheurísticas apresentarem melhores resultados (função objetivo *vs* tempo de execução) que os métodos clássicos, esta é uma indicação clara para sua aplicação. A segunda escolha refere-se à própria metaheurística. Uma escolha pode ocorrer encima das propriedades do espaço de busca e da função objetivo, por exemplo, a BT vem sendo particularmente empregada em espaços de busca discretos, onde existe um número finito de soluções vizinhas (GLOVER; LAGUNA, 1998). Outra motivação para se escolher uma metaheurística específica é o seu sucesso de implementação em problemas semelhantes.

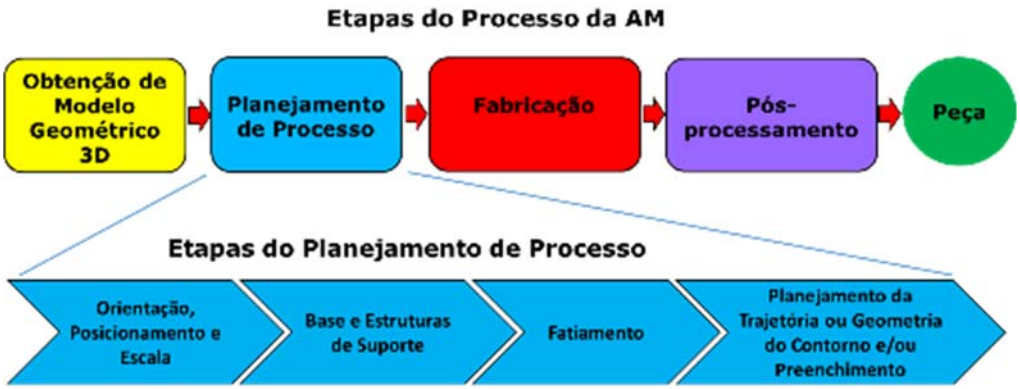
Independentemente do método escolhido, Puchinger e Raidl (2005) citam que uma boa prática é que deve-se iniciar por um algoritmo simples para posteriormente incrementá-lo e, se for o caso, torná-lo híbrido. Qualquer implementação dos

algoritmos apresentados anteriormente necessita de atenção quanto ao número de detalhes, uma tarefa que é geralmente deixada para o usuário.

## 2.3 PLANEJAMENTO DE PROCESSO NA MANUFATURA ADITIVA

Os processos de AM são popularmente conhecidos como impressão 3D, e, segundo Volpato e Silva (2017), de maneira geral as etapas do processo podem ser divididas da seguinte forma: obtenção do modelo geométrico CAD (*Computer-Aided Design*) 3D, planejamento do processo, fabricação em um equipamento de AM e pós-processamento, conforme ilustrado na Figura 2.9.

Figura 2.9 - Etapas do processo e do planejamento de processo da AM




Fonte: Volpato; Silva (2017).

A etapa de pré-processamento inicia-se a partir da utilização de um modelo geométrico CAD 3D e sua conversão para um formato de arquivo utilizado em planejadores de processos de AM (e.g. STL - *STereoLithography*, AMF - *Additive Manufacturing Format,* ou ainda outro formato 3D aceito pelo equipamento). O planejamento do processo propriamente dito inicia-se com a orientação e posicionamento do modelo 3D sobre a plataforma de construção, execução do fatiamento, cálculo da base e estruturas de suporte (caso necessário), geração das trajetórias de preenchimento das camadas e geração dos dados a serem enviados à máquina de AM.

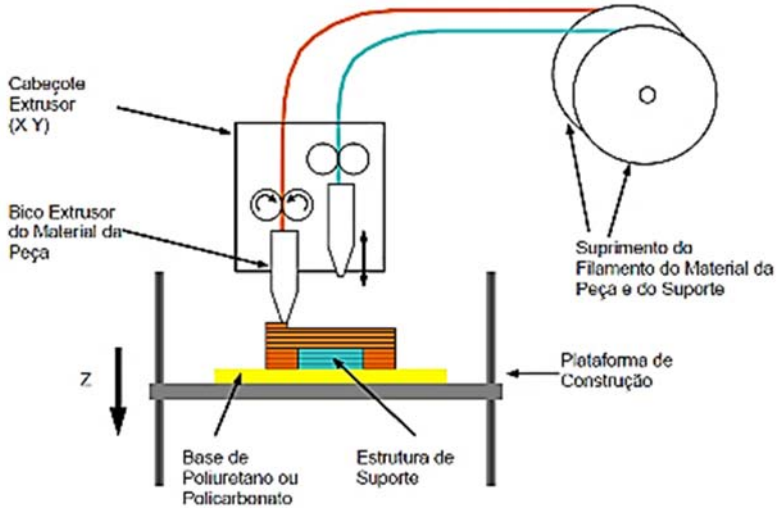
O planejamento de trajetórias destaca-se como uma tarefa crítica dentro do planejamento de processo, pois a ferramenta de construção (e.g. cabeçote extrusor) deve ser guiada para o preenchimento de cada camada, apresentando-se como uma etapa bastante intensiva computacionalmente (FOK *et al.,* 2019). Por fim, as

trajetórias de deposição influem diretamente nas propriedades da peça obtida, no tempo de construção e, consequentemente, nos custos operacionais.

## 2.4 PROCESSOS DE AM POR EXTRUSÃO DE MATERIAL

Volpato *et al.* (2013) citam que processos de AM por extrusão de material utilizam-se de um cabeçote extrusor, com um ou mais bicos, que é transladado nos planos X-Y extrudando e depositando o filamento de material que preencherá a peça em cada camada. A tecnologia FDM (*Fused Deposition Modeling*) introduzida no mercado nos anos 1990 pela empresa Stratasys Ltd é muito difundida atualmente. Ao final do preenchimento de cada camada, a plataforma se desloca no eixo Z com uma distância igual à espessura de camada, formando camadas superpostas de filamento até se obter o objeto pretendido (Figura 2.10).

Figura 2.10 - Princípio de funcionamento do processo FDM da Stratasys

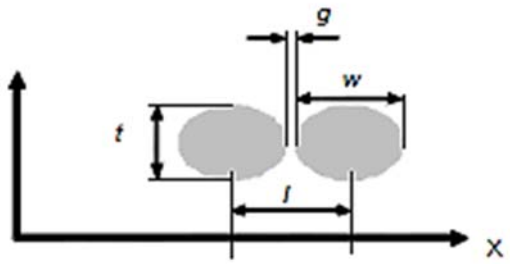



Fonte: Volpato *et al.* (2007b).

Os materiais utilizados neste processo comumente são polímeros termoplásticos como acrilonitrila-butadieno-estireno (ABS), ácido polilático (PLA), policarbonato (PC), PC-ABS, poliuretano (PU), entre outros. As tecnologias de AM por extrusão de material possuem algumas características de geometria de filamento e parâmetros de processo que podem ser controlados para obtenção de propriedades diferentes nas peças construídas, conforme descrevem Volpato *et al.* (2013).

A Figura 2.11 ilustra esquematicamente dois filamentos adjacentes onde pode-se observar os seguintes parâmetros:

Figura 2.11 - Parâmetros de processo no FDM




Fonte: Volpato *et al.* (2013).

- ***w***: largura do filamento
- ***t****:* espessura do filamento
- ***g***: espaçamento entre filamentos
- ***l***: distância entre filamentos

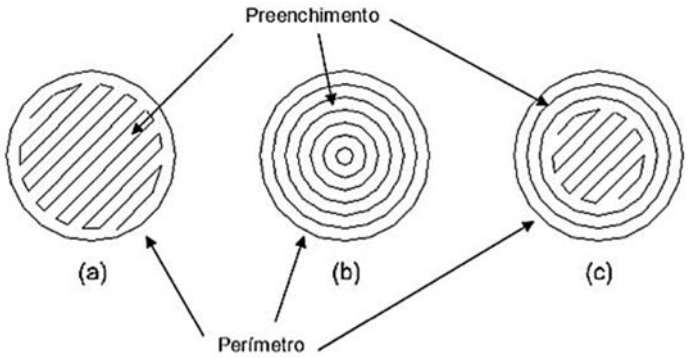
O planejamento da trajetória para preenchimento da área de cada camada permite a utilização de diferentes estratégias de preenchimento. Volpato *et al.* (2007b) citam que a estratégia de deposição normalmente empregada para estas tecnologias é depositar primeiramente os contornos (C) das peças (formados por uma sequência de pontos, compondo segmentos de reta que formam polígonos fechados) e então preencher as áreas internas de cada contorno com a estratégia definida pelo usuário.

O preenchimento pode ser realizado de três maneiras:

- *Raster*: preenchimento em movimentos ziguezague (paralelos) (Figura 2.12a).
- Contornos paralelos: preenchimento em contornos concêntricos (Figura 2.12b).
- Contorno e *raster*: estas duas estratégias de preenchimento são combinadas em cada camada (Figura 2.12c).

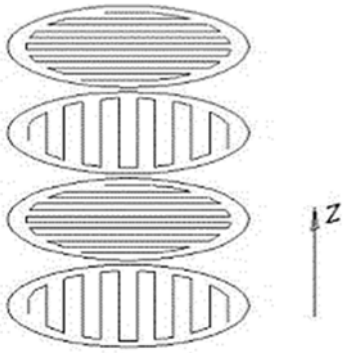
Conforme Volpato *et al.* (2013), o preenchimento tipo *raster* é muito empregado devido a sua maior velocidade de construção. É comum também ocorrer uma rotação no ângulo de deposição entre camadas adjacentes, de forma que os filamentos fiquem dispostos transversalmente entre as camadas, melhorando a resistência da peça construída (AHN *et al., 2002*), conforme ilustrado pela Figura 2.13.

Figura 2.12 - Tipos de preenchimento comumente utilizados em AM por extrusão de material




Fonte: Adaptado de Volpato (2007b).

Figura 2.13 - Preenchimento com alternância de direção entre camadas

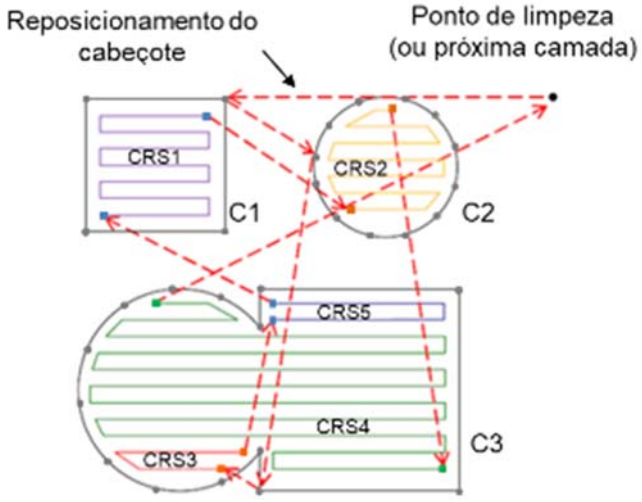



Fonte: Adaptado de Volpato *et al.* (2007b).

Como normalmente o preenchimento da camada não pode ser realizado de maneira contínua, o cabeçote deve interromper a extrusão e ser reposicionado em outra região para continuar o preenchimento. Os deslocamentos entre os trechos de deposição (Contornos – C e Segmentos Contínuos de *Raster* – CRS) são realizados em movimentos rápidos pelo equipamento, representados por linhas retas tracejadas na Figura 2.14.

A Figura 2.14 ilustra o preenchimento de uma camada de maneira sequencial. O cabeçote extrusor inicia o deslocamento no ponto de limpeza e é reposicionado passando pelos contornos C1 / C2 / C3. Posteriormente, o cabeçote extrusor é reposicionado passando pelos trechos de *raster* CRS3 / CRS5 / CRS1 / CRS2 / CRS4 e retorna ao ponto de limpeza. Como nos processos baseados no princípio de extrusão o preenchimento da camada é realizado por um único filamento fino de material (tornando-o mais lento que outros processos de AM), a otimização dos movimentos de reposicionamento é especialmente importante para a redução no tempo de fabricação (VOLPATO *et al.,* 2013).

Figura 2.14 – Trajetórias de deposição e movimentos de reposicionamento para processos de AM baseados em extrusão de material

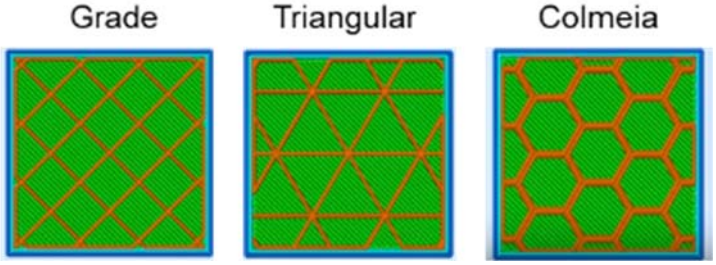



Fonte: Adaptado de Volpato *et al.* (2013).

Agarwala *et al.* (1996) e Habib e Khoda (2017) descrevem que, em Cs com polígonos não convexos, um CRS não é capaz de preencher a área interna de forma completa, sendo necessária a divisão da área para preenchimento com mais de um CRS. Na Figura 2.14 observa-se que o contorno C3 precisou de três CRSs (CRS3, CRS4 e CRS5) para o seu preenchimento completo. Analisando a camada completa verifica-se que para o preenchimento dos três Cs existe a necessidade de cinco CRSs. Baseado nesta característica do processo, Volpato *et al.* (2013) indicam que a otimização da distância de reposicionamento é promissora na redução do tempo total de construção de peças.

Outras formas de preenchimento também podem ser encontradas nos *softwares* planejadores de processo, utilizadas quando os preenchimentos são inferiores a 100%, ou seja, peças que não serão completamente preenchidas, conforme ilustrado na Figura 2.15.

Figura 2.15 – Alguns tipos de preenchimento utilizados em *softwares* planejadores de processo encontrados no mercado: grade (a), triangular (b) e colmeia (c)

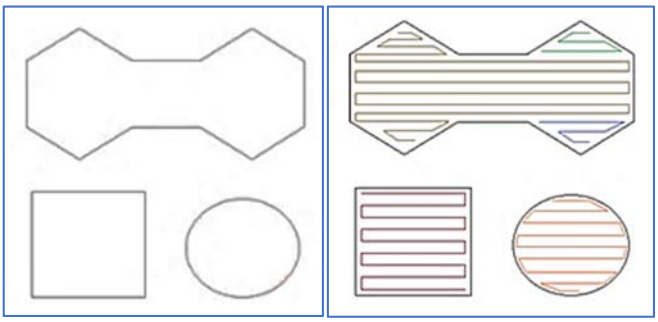



Fonte: Adaptado de Cura (2019).

Outra abordagem ligada ao preenchimento das camadas é a sequência de deposição, que consiste em determinar a ordem em que serão depositados os Cs e os CRSs, existindo duas opções: deposição sequencial e deposição intercalada (GIBSON *et al*., 2010).

A deposição sequencial é adotada por equipamentos da empresa Stratasys Ltd, onde ocorre primeiramente a deposição de todos os Cs de uma camada (Figura 2.16a) e posteriormente de todos os CRSs (Figura 2.16b).

Figura 2.16 – Exemplo da deposição sequencial

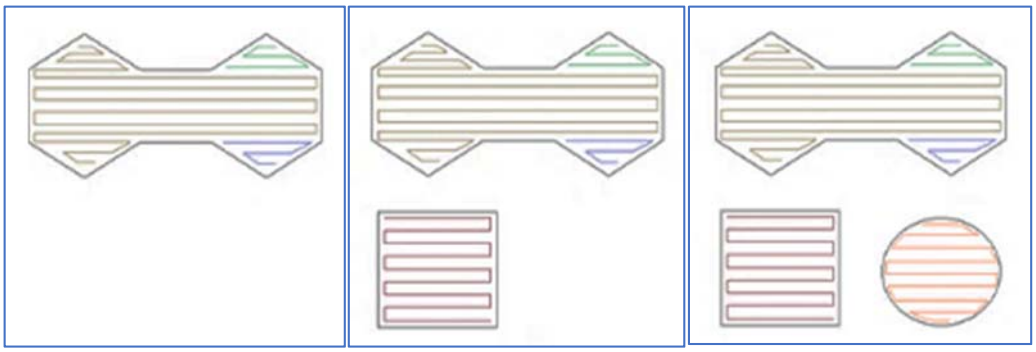


(a) (b)

Fonte: O autor (2019) baseado em Gibson *et al.* (2010).

A sequência de deposição intercalada consiste em depositar um determinado C externo (e posteriormente os internos, se existirem) e depois depositar os CRSs que preencherão esta região (Figura 2.17a). Respectivamente, na Figura 2.17b e 2.17c são adicionados o segundo e o terceiro Cs intercalados com seus respectivos CRSs, seguindo esta estratégia até o término da camada atual. Esta deposição intercalada entre Cs e CRSs repete-se até o preenchimento de toda a camada.

Figura 2.17 – Exemplo da deposição intercalada



(a) (b) (c)

Fonte: O autor (2019) baseado em Gibson *et al.* (2010).

## 2.5 OTIMIZAÇÃO DOS MOVIMENTOS DE REPOSICIONAMENTO EM PROCESSOS DE FABRICAÇÃO

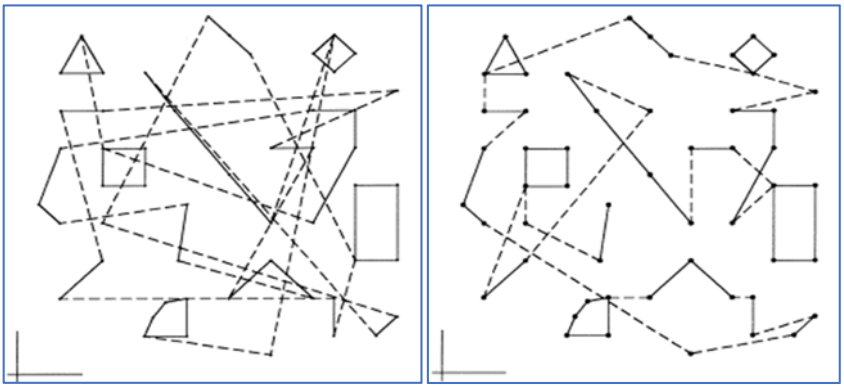
Existem diferentes formas de se otimizar os processos de fabricação, onde comumente encontram-se estudos envolvendo a otimização de parâmetros do processo, ferramentas, trajetórias, sequência de tarefas, entre outras. O problema de otimização dos movimentos de reposicionamento pode ser aproximado dos movimentos de deslocamento existentes no TSP e podem ser encontrados tanto em processos de fabricação considerados tradicionais (e.g. fresamento, puncionamento, furação, entre outros) como nos processos de fabricação por AM. Independentemente da característica do processo, o problema pode ser considerado similar, ou seja, movimentos de reposicionamento são necessários, mas não agregam valor. Sendo assim, otimizar os movimentos de reposicionamento torna-se importante para aumentar a produtividade e reduzir desperdícios. Nesta seção são apresentadas algumas das implementações de otimização dos movimentos de reposicionamento encontradas para os processos de fabricação tradicionais (que podem de alguma forma ser adaptados à AM) e também as implementações encontradas para processos de AM.

### 2.5.1 Fresamento

Khan, Hayhurst e Cannings (1999) implementaram um algoritmo de *Simulated Annealling* (SA) na otimização de movimentos de reposicionamento para operações de fresamento utilizando restrições de entrada e saída. A formulação ocorreu como um TSP de larga escala para os casos simétrico, assimétrico e misto em problemas considerados simples e complexos, variando de 10 até 641 nós. Os autores citam que uma formulação como TSP simétrico ocorre normalmente para contornos fechados (e.g. fresamento de contornos concêntricos, deposição dos contornos em FDM, corte a plasma, *laser* ou processos similares). Os casos formulados como TSP assimétrico são normalmente encontrados em operações de fresamento onde o sentido de corte da ferramenta influi no resultado da usinagem. O TSP misto é a aplicação mais próxima da realidade onde nós, segmentos abertos e contornos podem aparecer aglomerados, comumente encontrados em operações bidimensionais como fresamento. Um dos problemas resolvidos foi a otimização de um problema com 19

contornos e 57 nós distribuídos entre contornos fechados e segmentos abertos, conforme observa-se na Figura 2.18a e 2.18b as soluções inicial e final, respectivamente.

Figura 2.18 – Representação da solução inicial (a) e otimizada (b) para o TSP misto



(a) (b)

Fonte: Khan; Hayhurst; Cannings (1999).

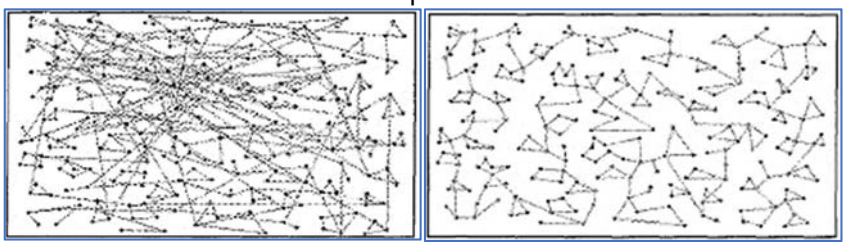
Como resultado foi observado que o algoritmo SA possibilitou o encontro de soluções ótimas ou subótimas dependendo da escolha do limite de iterações e a distribuição inicial dos nós. No trabalho não foram consideradas restrições de precedência entre os Cs e posteriormente CRSs, como acontece numa estratégia tradicional de AM por extrusão de material. Não foram previstas ainda a possibilidade de troca ou limpeza da ferramenta de corte e nem a otimização de múltiplas camadas de usinagem. Outro problema encontrado foi a busca ter ficado “presa” em regiões subótimas, gerando um grande custo computacional e necessitando da implementação de critérios de parada adicionais para o algoritmo de SA.

Wah *et al.* (2002) formularam um problema genérico de otimização de trajetórias de fabricação como um TSP assimétrico, utilizando arcos (semelhante a Cs e CRSs) e utilizaram-se de algoritmos híbridos baseados em algoritmo genético (AG) com 2-opt e 3-opt e compararam com um modelo de programação linear. Estes modelos foram realizados para um problema genérico de trajetórias de fabricação, mas podem ser adaptados em processos de AM por extrusão de material. Como restrições os autores colocam que o cálculo foi realizado para trechos abertos tomando duas decisões: *i) e*scolha do próximo trecho a ser percorrido e *ii*) escolha do sentido que este trecho será percorrido. Os autores descrevem que a implementação apenas do AG apresentou um elevado custo computacional, que foi reduzido através da hibridização, obtendo uma redução de até 50% nos movimentos de

reposicionamento comparada à solução não otimizada. No trabalho não foram consideradas restrições de precedência entre Cs e CRSs, nem a possibilidade de troca ou limpeza da ferramenta de corte e nem a otimização de múltiplas camadas de fabricação.

Castelino, Souza e Wright (2003) implementaram GTSP e *Sequencial Ordering Problem* (SOP) que é um TSP assimétrico com restrições adicionais de precedência na otimização de movimentos de reposicionamento em operações de fresamento de contornos paralelos. A implementação começa a partir de um sistema de planejamento de processo baseado em *features* a partir de um modelo geométrico e base de dados das ferramentas usadas no processo. Formulando o problema desta forma, os autores consideraram o problema como um GTSP com restrições de precedência, agrupando vários pontos de uma região em apenas um ponto (*cluster)*. As implementações propostas foram aplicadas para um problema com 82 polígonos e 242 pontos, conforme solução inicial e final mostradas respectivamente na Figura 2.19a e 2.19b. O resultado do algoritmo foi uma melhora de 544% em relação a solução inicial não otimizada e uma melhora de 65-70% em comparação a um algoritmo guloso. Os autores concluíram que ambas as heurísticas implementadas deram soluções ótimas para as peças usadas no sistema de planejamento de processo.

Figura 2.19 – Resultado não otimizado (a) e otimizado (b) obtidos no problema com 82 polígonos e 242 pontos



(a) (b)

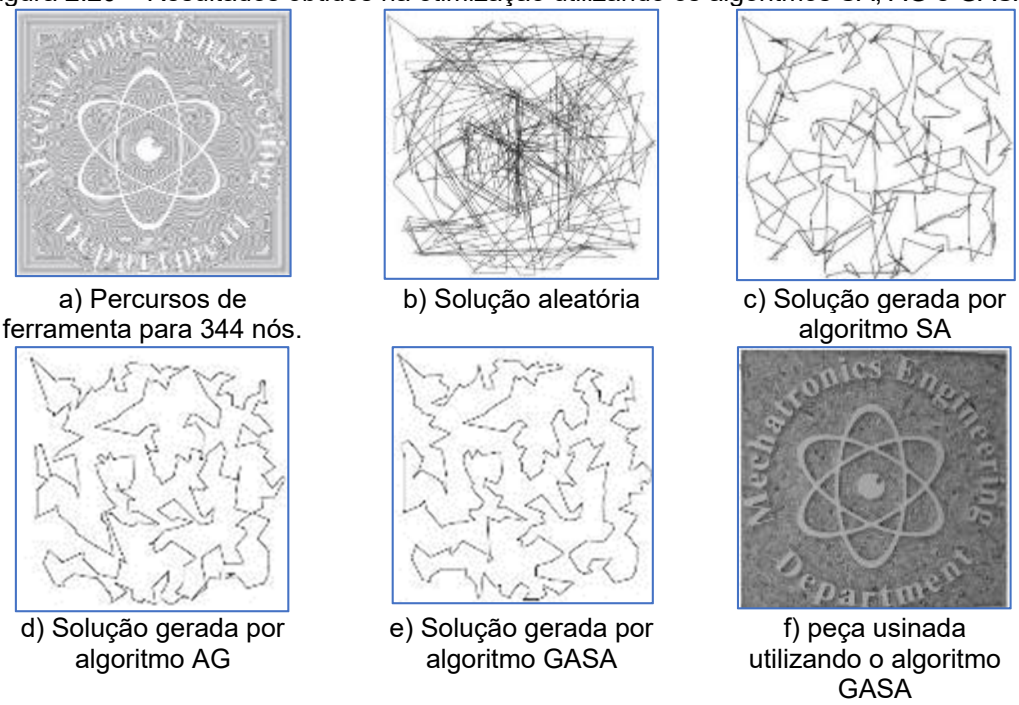
Fonte: Castelino; Souza; Wright (2003).

Neste trabalho não foi tratada a otimização de trajetórias abertas (e.g. CRS) e nem a otimização de múltiplas camadas.

Oysu e Bingul (2009) realizaram a otimização do tempo de reposicionamento em um sistema de fresagem, trabalhando apenas com contornos paralelos devido ao ponto de entrada/saída serem coincidentes, reduzindo o problema para análise semelhante ao TSP. O artigo trata da comparação de SA, AG e um algoritmo hibrido

entre os ambos chamado GASA. Os autores implementaram vários testes para problemas simples (88 nós) e complexos (344 nós). O algoritmo GASA apresentou melhores resultados em problemas mais complexos, porém com um expressivo custo computacional. Na Figura 2.20 os autores mostram a comparação dos diferentes algoritmos para o problema de 344 nós.

Figura 2.20 – Resultados obtidos na otimização utilizando os algoritmos SA, AG e GASA



a) Percursos de ferramenta para 344 nós.

b) Solução aleatória

c) Solução gerada por algoritmo SA

d) Solução gerada por algoritmo AG

e) Solução gerada por algoritmo GASA

f) peça usinada utilizando o algoritmo GASA

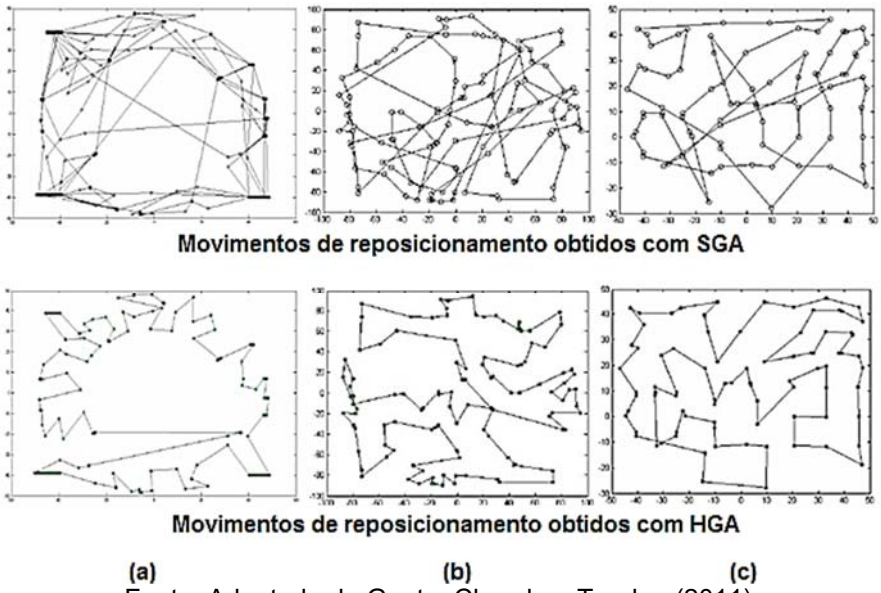
Fonte: Adaptado de Oysu; Bingul (2009).

A Figura 2.20b apresenta à solução inicial aleatória, gerando uma movimentação de reposicionamento excessiva. A otimização através do algoritmo GASA (Figura 2.20e) apresentou uma redução de até 84,1% no tempo de usinagem para o problema complexo em relação a solução aleatória. Como resultado Oysu e Bingul (2009) colocam que o ganho com o algoritmo GASA foi expressivo, gerando a redução no tempo e custo do processo de fresamento, podendo o algoritmo ser facilmente integrado ao processo *off-line* de sistemas *Computer-Aided Manufacturing* (CAM), de forma a ser integrado como um módulo adicional. Neste trabalho não foi tratada a otimização de trajetórias abertas (e.g. CRS) e nem a otimização de múltiplas camadas.

Seguindo nesta mesma linha de pesquisa, Gupta, Chandna e Tandon (2011) também realizaram a otimização do tempo de reposicionamento em operações de fresagem. Os autores propuseram um AG híbrido (HGA) que possui um algoritmo que

faz o pré-processamento dos pontos existentes na trajetória, partindo assim de uma solução inicial já pré-otimizada, comparando-o com um AG simples (SGA). Três problemas de alta (275 nós), média (150 nós) e baixa (85 nós) complexidade foram estudados, onde o HGA apresentou melhores resultados para todos os problemas testados, obtendo uma melhora de 3,00%, 6,42% e 11,01%, respectivamente. Um exemplo dos resultados obtidos é apresentado na Figura 2.21. Verifica-se na Figura 2.21 que o HGA gerou trajetórias com menos movimentos de reposicionamento em relação ao SGA.

Figura 2.21 – Trajetórias obtidas para os problemas analisados: 275 nós (a); 150 nós e (b); 85 nós (c)




Fonte: Adaptado de Gupta; Chandna; Tandon (2011).

Como resultado da análise, o HGA mostrou-se melhor para os três problemas propostos, apresentando um resultado superior para os problemas mais complexos, principalmente para problemas com população maior que 60 nós. Neste trabalho não foi abordada a possibilidade de otimização de trajetórias abertas (e.g. CRS) e nem a otimização de múltiplas camadas.

### 2.5.2 Puncionamento

Roychoudhury e Muth (1995) compararam a efetividade de heurísticas de sequenciamento de operações em um processo de puncionamento controlado por comando numérico computadorizado (CNC) para chapas metálicas. Este tipo de processo é fortemente afetado pela sequência de operações, pois o tempo de contato

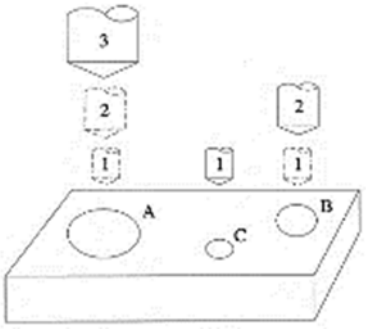
entre a ferramenta e a peça é muito pequeno comparado ao tempo de deslocamento de reposicionamento e troca de ferramental. Foram analisados 42 problemas reais de uma empresa do ramo através de uma formulação baseada no TSP e implementação com uso das heurísticas do vizinho mais próximo, inserção aleatória, inserção do ponto mais próximo e mais distante e ainda algumas combinações entre estas. Como método de melhora foram realizados métodos 2-opt combinados com os métodos iniciais e ainda com diferentes características de problemas. Como bases de avaliação foram comparadas as distâncias e tempos de processamentos obtidos. Em relação ao resultado, foi observado que as heurísticas de inserção do vizinho mais próximo e inserção aleatória foram as que obtiveram os melhores resultados.

### 2.5.3 Furação

Kolahan e Liang (2000) implementaram BT para otimização em operações de furação. O problema foi estruturado de forma semelhante a um TSP assimétrico, porém com diferença de ser mais complexo devido à associação de custos dependentes de cada sequência, operação e posição. Os autores colocam como fator complicador que dependendo do diâmetro do furo, da geometria da ferramenta e qualidade superficial desejada, um furo pode necessitar de várias operações com diferentes conjuntos de ferramentas, conforme esquema apresentado na Figura 2.22. O cálculo da função objetivo buscou minimizar o custo total da operação de furação considerando simultaneamente a combinação do conjunto de ferramentas disponíveis, a sequência de operações, o custo da troca de ferramenta, os custos de deslocamento de reposicionamento e velocidade de corte para uma placa de fixação de um molde de injeção. Como parâmetros de entrada foram utilizados 12 tipos de ferramentas diferentes com velocidades de corte, avanço, diâmetro, custo unitário, ciclo de vida e ainda sua posição no porta ferramentas. Como cada furo pode ser realizado por diferentes combinações de ferramentas, a solução inicial constrói uma possível sequência de operações viáveis. A peça utilizada no estudo de caso possuía 26 furos com posições, profundidades, operações e acabamentos diferentes. Dentre os resultados apresentados, os autores citam que modificando os parâmetros de tamanho da lista tabu e critério de diversificação, o algoritmo reduziu de 80 possíveis operações dadas na solução inicial para 56 operações requeridas, representando uma significativa redução de 47% sobre a solução inicial em 10 minutos de execução. Foi

verificado um incremento expressivo de performance do algoritmo com o aumento do problema em um tempo de execução considerado razoável.

Figura 2.22 – Representação esquemática do processo de furação utilizando diferentes ferramentas




Fonte: Kolahan; Liang (2000).

Ghaiebi e Solimanpur (2007) aplicaram um algoritmo de colônia de formigas (ACO) na otimização dos tempos de deslocamento em reposicionamento e troca de ferramentas em um processo de furação. O problema foi tratado como um TSP assimétrico com restrições de precedência e formulado através de um modelo matemático binário não linear. Os autores colocam que em uma operação de furação múltipla com furos de grande diâmetro a operação de furação deve ser realizada em etapas com diferentes ferramentas, e sua otimização deve ser realizada na sequência de furação e nas ferramentas usadas de forma a minimizar os deslocamentos de reposicionamento. Um diferencial encontrado neste trabalho é a possibilidade de se definir e inserir no algoritmo um conjunto de informações de restrição da sequência de ferramentas que devem ser utilizadas em cada furo. Os resultados do algoritmo ACO foram comparados com um algoritmo de LP, onde a redução média entre ambos os métodos no tempo de furação chegou a 52%. Nota-se que este algoritmo pode ser aplicado a outros problemas de otimização de rotas como fresamento e processos de AM.

## 2.6 OTIMIZAÇÃO DE MOVIMENTOS DE REPOSICIONAMENTO EM PROCESSOS DE AM

A popularização de tecnologias de AM trouxe à tona um dos seus principais problemas: o elevado tempo de construção das peças. Deve ser citado que otimizações também podem ser aplicadas em outras etapas do processo (JIN *et al.,*

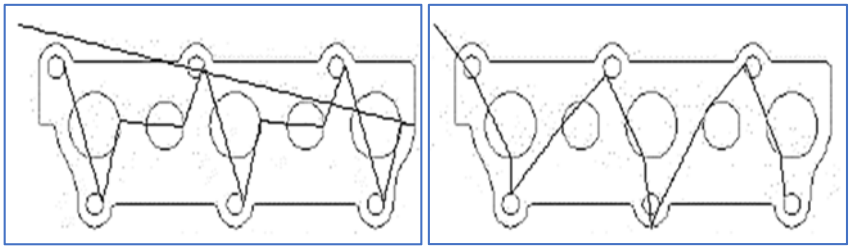
2013; JIN *et al.,* 2017; AHASAN; KHODA, 2016; KULKARNI; MARSAN; DUTTA, 2000), principalmente no que diz respeito à geração do arquivo que contém os dados geométricos do modelo 3D (e.g. refinamento da malha de triângulos no arquivo stl), orientação, posição, planejamento das trajetórias de construção, entre outras, as quais não fazem parte do escopo deste trabalho. Uma extensa revisão sobre otimização de parâmetros de processo em AM por extrusão de material foi apresentada por Mohamed, Masood e Bhowmik (2015). Nesta seção são apresentadas as implementações de otimização de movimentos de reposicionamento em processos de AM que mais se assemelham ao trabalho proposto.

### 2.6.1 Manufatura Laminar de Objetos (*Laminated Object Manufacturing* - LOM)

Tang e Pang (2003) estruturaram o problema de otimização das distâncias de reposicionamento entre Cs para a tecnologia LOM como um TSP simétrico. Os autores propuseram um algoritmo denominado de máxima inserção linear (MLI) e compararam os resultados com AG. O algoritmo MLI proposto mostrou uma vantagem considerável sobre o AG, obtendo uma redução de até 50% nos deslocamentos de reposicionamento com apenas 500 iterações e apresentando um tempo de execução de duas a três vezes inferiores para as geometrias mais complexas. Os autores colocam que apesar de obter um bom resultado, o algoritmo apresentou elevado custo computacional para as geometrias mais complexas resolvidas (mais de quatro Cs), mesmo considerando apenas o reposicionamento entre Cs e a otimização de apenas uma camada.

Weidong (2009) também propôs um AG para minimizar as distâncias de reposicionamento entre Cs no processo LOM, formulando o problema como um TSP simétrico. O algoritmo identifica os possíveis pontos de entrada e saída nos Cs para então calcular a distância de reposicionamento total a partir de diferentes trajetórias possíveis (Figura 2.23). O autor obteve com apenas 50 iterações do algoritmo uma redução de até 53% nos reposicionamentos para uma camada da peça analisada, significando uma redução de 1,4 horas no tempo de escaneamento com *laser* necessário para fabricação. A Figura 2.23b apresenta um exemplo de otimização realizada para uma das geometrias propostas. Neste trabalho não foi tratada a otimização de reposicionamento entre trajetórias abertas (e.g. CRS) e nem a otimização de múltiplas camadas.

Figura 2.23 – Solução inicial não otimizada (a) e após 50 iterações (b)



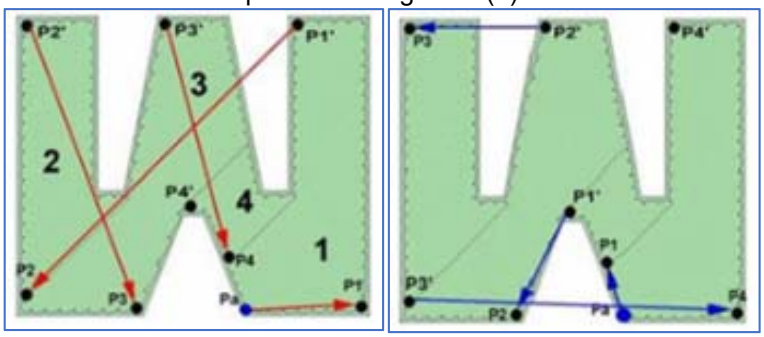
(a) (b)

Fonte: Weidong (2009).

### 2.6.2 Extrusão de Material

Volpato *et al.* (2007) observaram que no *software* comercial Insight de um equipamento FDM da fabricante Stratasys Ltd, os movimentos de reposicionamento são gerados de forma não otimizada, possibilitando ganho de produtividade no processo através de sua otimização. Os autores implementaram um algoritmo guloso para otimizar as distâncias de reposicionamento (Figura 2.24), adotando estratégias de deposição sequencial e intercalada (seção 2.3), onde, na média, a redução de tempo obtida por camada foi de 8,92% com relação ao *software* comercial.

Figura 2.24 – Solução inicial aleatória de movimentos de reposicionamento (a) e solução otimizada pelo método guloso (b)




(a) (b)

Fonte: Volpato *et al.* (2007).

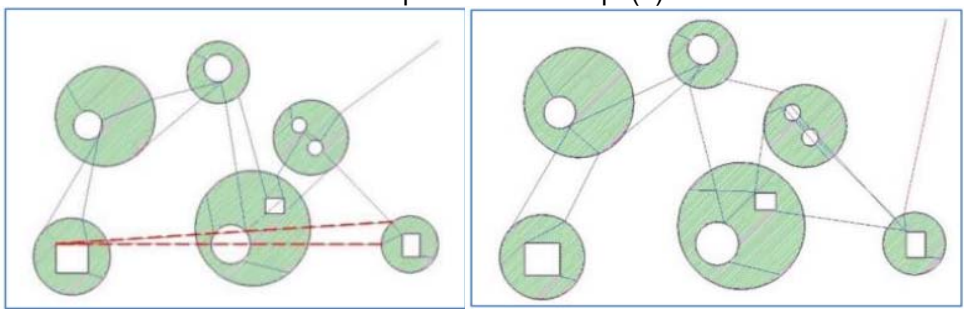
Neste trabalho, os autores trataram o problema de maneira parcial considerando a otimização de uma camada e, mesmo com algoritmos considerados simples conseguiram bons resultados.

Volpato *et al.* (2013) apresentaram duas propostas de otimização, uma baseada em um método guloso e a outra combinando a inserção do ponto mais próximo (para Cs) e mais distante (para CRS). Os autores relataram que o método guloso obteve melhor resultado para geometrias simples, chegando a obter uma

redução de deslocamento de 39,2% em relação a reposicionamentos sem otimização. Já o segundo método obteve melhores resultados para geometrias mais complexas, apresentando uma redução de até 58,5% nas distâncias em relação a reposicionamentos sem otimização. Neste novo trabalho, Volpato *et al.* (2013) apresentaram algoritmos considerados simples, trataram o problema de forma parcial (apenas uma camada por vez).

Volpato *et al.* (2019) propuseram a otimização dos reposicionamentos considerando deposição sequencial e combinando as heurísticas de inserção do mais próximo com 2-opt e comparando-as com um algoritmo guloso, testando casos de diferentes complexidades. A otimização inicia pelos Cs e posteriormente otimiza os CRS. Na proposta, ocorre uma segunda etapa de otimização utilizando o algoritmo de 2-opt, testando a inversão no sentido de preenchimento dos CRS, tentando obter um refinamento da solução inicial. Neste trabalho, o problema foi tratado de maneira completa, ou seja, o algoritmo foi estruturado de forma a tratar da otimização de múltiplas camadas e com a possibilidade de incluir deslocamentos para limpeza do cabeçote, como acontece em muitos processos comerciais, porém dificilmente encontrada em equipamentos de baixo custo. Ao total, foi realizada a otimização entre 393 camadas, com limpeza a cada duas ou dez camadas, ou ainda a não realização da limpeza durante o processo.

Figura 2.25 – Comparação da trajetória gerada através do método guloso (a) e o método de inserção do mais próximo com 2-opt (b)



(a) (b)

Fonte: Volpato *et al.* (2019).

A conexão entre as camadas pode ocorrer de duas formas, onde: *i)* se não houver necessidade de limpeza do cabeçote extrusor, ele é deslocado e inicia o preenchimento no ponto de C mais próximo na camada seguinte; *ii)* se houver necessidade de limpeza, o cabeçote é deslocado até o ponto de limpeza e posteriormente inicia o preenchimento pelo ponto de C mais próximo. A Figura 2.25

ilustra um dos resultados obtidos pelos autores, comparando o método guloso (Figura 2.25a) em relação ao método de inserção do mais próximo com 2-opt (Figura 2.25b), cujo resultado apresentou ganho de 25,1%. Para esta mesma peça, em relação a solução sem otimização, os ganhos chegaram a 46,3%. Estes resultados mostram que os ganhos com otimização de distâncias de reposicionamento são significativos, mesmo com algoritmos considerados simples.

Ganganath *et al.* (2016) propuseram a otimização dos tempos para um processo de AM genérico. Os autores propuseram a modificação das curvas de velocidade do cabeçote extrusor e otimização das distâncias de reposicionamento. O problema foi formulado como um TSP e solucionado através dos algoritmos de inserção do mais próximo e Christofides (1976) com 2-opt e aplicado para um caso com segmentos de deposição isolados e distribuídos aleatoriamente na área de deposição. Os resultados apontaram uma redução de até 50% em relação a solução não otimizada.

Fok *et al*. (2016a) propuseram a otimização do reposicionamento em processo AM por extrusão de material utilizando um *solver* baseado no algoritmo de Christofides (1976) modificado e compararam os resultados obtidos pelo *software* de planejamento de processo denominado CURA. Os autores reportaram uma redução média de 8,58% nas distâncias de reposicionamento. Salienta-se que o trabalho considerou apenas Cs para uma única camada.

Fok *et al*. (2019) propuseram um *solver* em duas etapas, onde, na primeira etapa formulada como sendo TSP a menor distância de reposicionamentos é encontrada e na segunda etapa formulada como sendo *Undirected Rural Postman Problem* a rota é refinada com uma restrição para minimizar os cruzamentos do cabeçote com o C externo das ilhas, minimizando assim a formação de fiapos (*strings*) de filamento aderidos na peça. Ou seja, adota a estratégia de deposição intercalada, onde toda a ilha deve ser preenchida antes de se reposicionar o cabeçote em outra região. O trabalho adaptou o modelo de controle de aceleração do cabeçote extrusor proposto por Fok *et al.* (2016b) para otimizar a velocidade dos deslocamentos, os autores implementaram *solvers* baseados nos algoritmos de Christofides-Frederickson, colônia de formigas clássico e modificado para otimizar os movimentos de reposicionamento. Segundo os autores a principal vantagem dos algoritmos de colônia de formigas é a possibilidade de processamento paralelo, buscando a redução dos tempos de processo e comparado os resultados com os obtidos no *software*

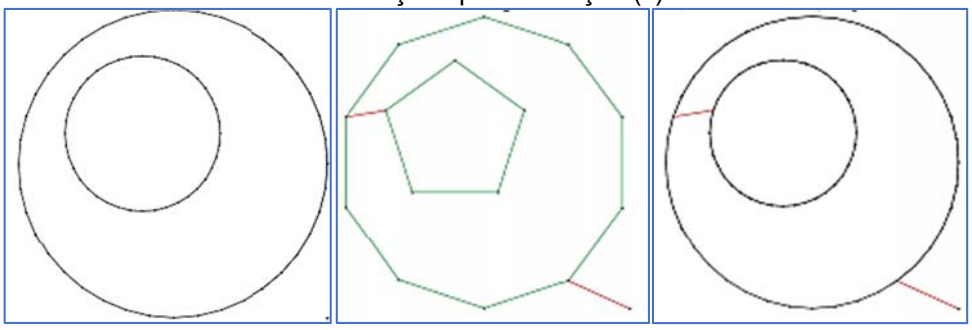
CURA. Os autores citam que os algoritmos de colônia de formigas obtiveram os melhores resultados com tempos de processamento considerados razoáveis. Neste trabalho foi solucionada a otimização de múltiplas camadas, considerando Cs e CRSs, mas sem permitir a inclusão de movimentos para limpeza do cabeçote.

Iori e Novellani (2019) também propuseram a otimização dos movimentos de reposicionamento para AM por extrusão de material formulando-o como um *Rural Postman Problem*. Os autores formalizaram o problema como programação linear inteira e solucionaram utilizando quatro propostas heurísticas, comparando ainda com o *software* CURA. A primeira heurística foi denominada de *Closest* 3DP e consiste em, partindo da origem buscar o segmento de deposição mais próximo até completar a camada. A segunda heurística denominada *Clustered* 3DP funciona de maneira similar à primeira, mas divide a área de deposição em três *clusters* (semelhante a definição de ilhas – seção 3.1), partindo da origem, preenchendo um *cluster* completamente (preenchimento intercalado – seção 2.4) e em seguida reposicionando até o *cluster* mais próximo. A terceira heurística denominada *Look Ahead* 3DP utiliza como base a primeira heurística e analisa três possíveis futuros movimentos a partir do ponto atual, optando pelo movimento que forneça o menor custo ao final da camada. A quarta heurística denominada *Shortest Path Based* 3DP realiza a simplificação da complexidade computacional através da limitação de pontos que serão usados no cálculo, podendo ser definido pelo usuário como valor fixo ou como uma porcentagem do total de vértices da camada. O cálculo utiliza o algoritmo de Dijkstra em conjunto com a heurística *Closest* 3DP na solução do problema. Os autores citam que os melhores resultados foram obtidos com a heurística *Look Ahead* 3DP, obtendo uma redução de até 36%, mas com elevado custo computacional e que a melhor relação entre custo computacional e redução de reposicionamentos é esperado da heurística *Shortest Path Based* 3DP, mas isso precisa ser mais testado. Neste trabalho foi realizada a otimização de Cs, CRSs e múltiplas camadas, mas sem permitir a inclusão de movimentos para limpeza do cabeçote.

Fok *et al*. (2016b) propuseram um modelo para otimização das curvas de velocidade do cabeçote extrusor e outro para as distâncias de reposicionamento. Os autores formularam o problema de reposicionamento baseado no TSP e para reduzir a complexidade computacional foi proposto um esquema de relaxamento que consiste em particionar e consolidar os segmentos de deposição. Primeiro, a área de deposição foi dividida em pequenas partições para serem resolvidas individualmente.

A otimização ocorre inter-partições e posteriormente intra-partições. Na primeira etapa identifica-se as partições que contenham segmentos de deposição e seus pontos centrais. O *solver* do TSP com 2-opt adaptado de Fok *et al*. (2016a) determina a sequência de partições a serem visitadas objetivando minimizar a distância percorrida em reposicionamentos entre seus pontos centrais. Na etapa de consolidação, define-se um parâmetro que deve ser ajustado conforme a geometria da peça e que controla o comprimento máximo dos segmentos de deposição a serem considerados no cálculo. Assim, vários segmentos podem ser unidos e considerados como um segmento maior, simplificando o problema para então realizar a otimização interna de cada partição através do mesmo *solver*. Após as otimizações inter e intra-partições serem realizadas para definição das rotas de reposicionamento a geometria completa da peça é recuperada. Os autores citam que a geometria original de estudo contém dois polígonos totalizando 80 segmentos (Figura 2.26a). A aplicação da relaxação e consolidação permitiu a redução para 15 segmentos (Figura 2.26b), facilitando a resolução através do *solver* para definição dos movimentos de reposicionamento. Por fim, a geometria da peça é recuperada, mantendo-se os movimentos de reposicionamento definidos anteriormente (Figura 2.26c).

Figura 2.26 – Comparação entre a geometria original (a), a aplicação da relaxação e consolidação (b) e a solução após relaxação (c).



(a) (b) (c)

Fonte: Adaptado de Fok *et al.* (2016).

Na prática o esquema de relaxação proposta visou simplificar a geometria das peças reduzindo o número de pontos que devem ser otimizados pelo *solver* TSP, reduzindo o custo computacional.

Os autores reportaram bons resultados para peças mais complexas, atingindo uma redução de tempo de construção de até 11,06% em relação ao *software* de planejamento de processo denominado CURA sem introduzir grandes impactos sobre

a qualidade das soluções obtidas. Os autores trataram o problema de forma parcial, considerando apenas Cs para uma camada, não considerando a existência de CRSs, múltiplas camadas ou movimentos de limpeza do cabeçote.

## 2.7 CONSIDERAÇÕES SOBRE A REVISÃO BIBLIOGRÁFICA

A otimização de distâncias de reposicionamento é um problema bastante complexo a nível computacional, devido ao número de pontos existentes nas camadas e das várias camadas que compõe uma peça.

Os trabalhos em AM que otimizaram apenas os movimentos de reposicionamento entre Cs ou trechos de deposição abertos (semelhantes a CRSs) foram propostos por Tang e Pang (2003), Weidong (2009), Ganganath *et al.* (2016), Fok *et al*. (2016a), Fok *et al*. (2016b). Estes trabalhos consideraram apenas uma camada. Os trabalhos que consideraram Cs e CRSs para uma camada foram propostos por Volpato *et al*. (2007a) e Volpato *et al.* (2013). Na otimização entre Cs e CRSs para múltiplas camadas encontra-se os trabalhos de Iori e Novellani (2019), Fok *et al*. (2019) e Volpato *et al*. *(*2019), onde, somente Volpato *et al*. *(*2019) preocuparam-se em inserir o procedimento de limpeza do cabeçote extrusor. Cabe destacar que os trabalhos encontrados que se preocuparam na redução da complexidade computacional foram os propostos por Fok *et al.* (2016b) e uma das heurísticas propostas por Iori e Novellani (2019). Nenhum dos trabalhos revisados preocupou-se com o detalhamento do problema e a formalização de um método estruturado capaz de permitir sua solução através de diferentes métodos de pesquisa operacional.

Embora existam publicações otimizando os movimentos de reposicionamento e duas delas propondo formas de reduzir a complexidade computacional, observa-se que o problema não foi formalizado e estruturado de maneira genérica para que possa ser solucionado através de diferentes técnicas de pesquisa operacional. Neste sentido, propõe-se desenvolver um método (*framework*) específico para estruturar, decompor e simplificar este problema. Seguindo-se a estrutura de trabalho proposta no *framework,* é possível realizar a otimização utilizando métodos de pesquisa operacional destinados a este fim. Neste caso, foram propostos a utilização de modelos de programação linear inteira mista como *benchmark* e metaheurísticas de busca tabu, e destaca-se que a busca tabu ainda não foi aplicada para esta finalidade.

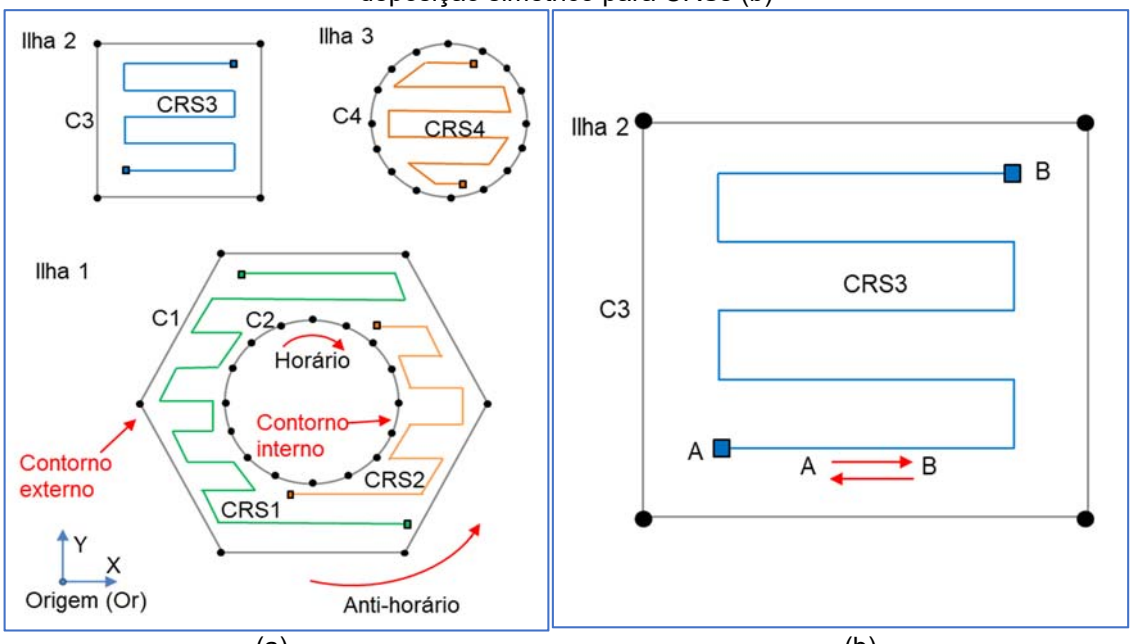
# 3 MATERIAIS E MÉTODOS

Nesse capítulo são apresentadas a definição do problema e os métodos que foram utilizados no desenvolvimento desta pesquisa. Foi proposto um *framework* que tratará separadamente contornos (Cs) e segmentos contínuos de *rasters* (CRSs), para uma estratégia de deposição sequencial, através de diferentes abordagens para a otimização de distâncias de reposicionamento para o processo de manufatura aditiva (AM) por extrusão de material. O *framework* decompõe o problema em quatro passos principais e propõe o uso de métodos de simplificação que reduzem o tamanho do problema e o custo computacional.

## 3.1 DEFINIÇÃO DO PROBLEMA

A camada de deposição pode ser dividida esquematicamente nos elementos apresentados na Figura 3.1. Inicialmente, cada região onde será depositado material é denominada de ilha. Uma ilha é delimitada por um C externo e dentro da mesma, há pelo menos um CRS, podendo também conter outros Cs internos ou até outras ilhas.

Figura 3.1 – Representação esquemática das ilhas e seus componentes (a) e detalhe do sentido de deposição simétrico para CRSs (b)




(a) (b)

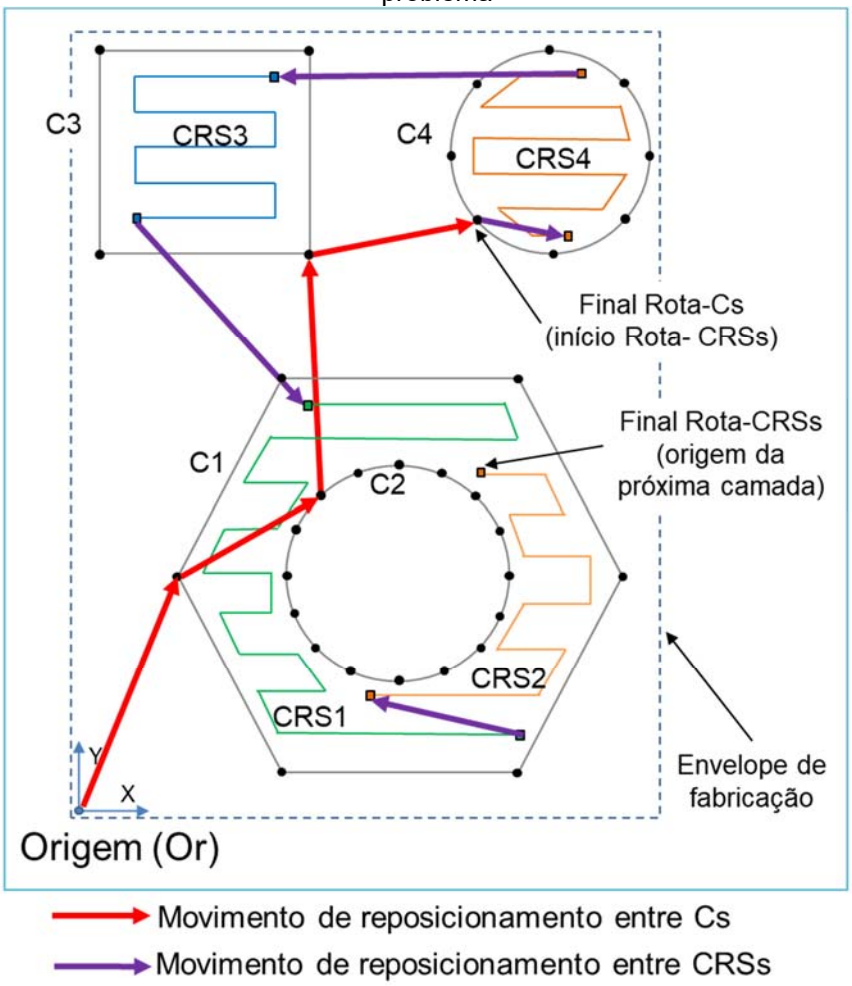
Fonte: O autor (2019).

Conforme ilustrado na Figura 3.1a, a camada é constituída de três ilhas, quatro Cs (C1-C4) e quatro CRSs (CRS1-CRS4), sendo que a ilha 1 possui um C externo, um C interno e dois CRSs. Os Cs são polígonos formados por um conjunto de pontos e segmentos de reta que constituem uma trajetória fechada e, sendo assim, apenas um dos pontos do contorno precisa ser escolhido para compor a rota de reposicionamento, porque os pontos inicial e final são coincidentes. A Figura 3.1b apresenta de maneira detalhada um CRS, que pode ter vários pontos intermediários (para formar o caminho em ziguezague), mas apenas os pontos inicial e final precisam ser considerados, ou seja, cada CRS é representado por um par de pontos distintos, formando uma trajetória aberta. Para ambos os casos (Cs e CRSs) a trajetória de deposição é considerada simétrica (um caso especial do TSP simétrico – ver seção 2.1), ou seja, o sentido de deposição não é considerado como restrição. No caso de Cs a deposição pode ocorrer no sentido horário ou anti-horário, mas, por convenção os *softwares* de planejamento de processo definem que o sentido de deposição dos Cs externos é anti-horário e dos Cs internos horário (conforme ilustrado na Ilha 1 da Figura 3.1a), mas isso não é uma restrição na solução do problema. No caso dos CRSs (Figura 3.1b), a simetria de movimento significa que o deslocamento é permitido em qualquer sentido, ou seja, do ponto A para B, ou vice-versa.

É importante mencionar que, o número e a posição dos pontos dos Cs são determinados pelo *software* de planejamento de processo de AM, que depende da geometria da peça e da precisão da malha 3D. Os CRSs também são obtidos pelo *software* de planejamento de processo, mas os pontos são influenciados por parâmetros de processo definidos pelo usuário, como ângulo de *raster*, espaçamento de *raster* (*raster gap*) e ainda largura de *raster* (*raster width*), conforme apresentado na Seção 2.4.

Todos os Cs e CRSs (e seus respectivos pontos) em uma ilha específica podem ser identificados e mapeados pelo *software* de planejamento de processo. Em outras palavras, cada ilha em cada camada terá um conjunto de Cs e CRSs agrupados e identificados. Tendo todas as ilhas mapeadas, o problema pode ser tratado com um caso especial de Caixeiro Viajante Generalizado (GTSP – ver seção 2.1.1).

Figura 3.2 – Representação esquemática da deposição de uma camada utilizada na formulação do problema




Fonte: O autor (2019).

Neste trabalho, a abordagem de deposição adotada será a sequencial (apresentada na Seção 2.4), que é utilizada nas impressoras comerciais da empresa Stratasys Ltd. Isso significa que o cabeçote extrusor começará a partir de um ponto de origem (Or) na plataforma, visitará todos os Cs primeiro e depois todos os CRSs, apenas uma vez, sem retornar à origem ao final de uma camada. Na Figura 3.2 a deposição sequencial é exemplificada, com o cabeçote extrusor partindo do ponto Or e sendo reposicionado nos contornos C1 / C2 / C3 / C4 e posteriormente nos segmentos de *raster* CRS4 / CRS3 / CRS1 / CRS2, completando a camada.

Por uma questão de simplicidade, supõe-se que os reposicionamentos do cabeçote extrusor sejam linhas retas, permitindo calcular a distância euclidiana entre os dois pontos de reposicionamento. O tamanho da plataforma neste estudo é dado pelo tamanho do envelope de fabricação que contém todas as peças a serem

fabricadas (caixa delimitadora das peças), conforme ilustrado na Figura 3.2. No caso da primeira camada, a origem de reposicionamento é definida na origem do plano cartesiano da área de trabalho, ou seja, no ponto inferior esquerdo do envelope de fabricação. Para as camadas posteriores, a origem será sempre o último ponto de CRS visitado na camada anterior. Este processo repete-se para todas as camadas, ou seja, com exceção da primeira camada que inicia na origem do plano cartesiano, todas as camadas seguintes sempre serão iniciadas a partir do último ponto de CRS visitado na camada anterior.

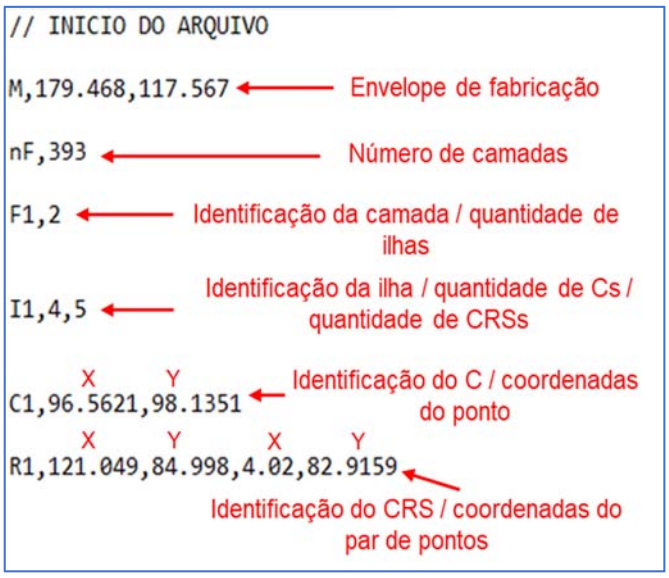
Ao final da construção da peça, a distância total dos reposicionamentos será o somatório dos reposicionamentos realizados em cada camada. Como simplificação adotada no método, neste trabalho não está prevista a implementação do procedimento de limpeza do cabeçote extrusor, uma vez que nas impressoras de baixo custo, isto não é observado, mas isto pode ser implementado futuramente.

Dados de entrada

Os dados estruturados de entrada para o modelo proposto serão obtidos a partir de um arquivo de texto (txt) denominado DAMAT. Este arquivo é gerado por um sistema de planejamento de processo de AM, chamado de *Rapid Prototyping Process Planning* (RP3) (VOLPATO; FOGGIATTO, 2009). Dentre as informações que podem ser obtidas do arquivo DAMAT, cujo cabeçalho é apresentado na Figura 3.3, pode-se destacar:

- Dimensão do envelope de fabricação;
- Espessura e número de camadas;
- Identificação das ilhas, Cs e CRSs;
- Coordenadas cartesianas dos pontos que deverão ser percorridos pelo cabeçote extrusor (todos os pontos dos C e somente os pontos de início e fim de CRSs);
- Coordenadas cartesianas dos pontos.

Figura 3.3 – Dados do arquivo Damat contendo todas as coordenadas das camadas

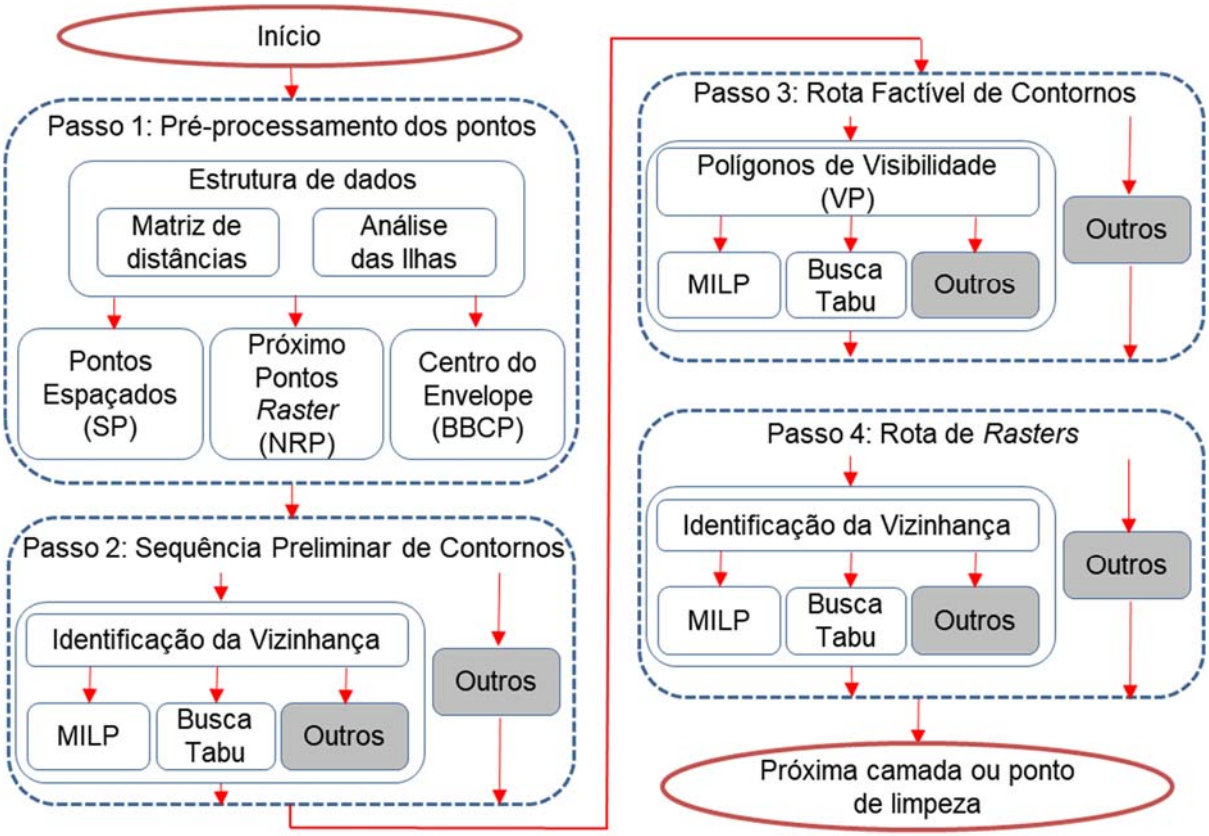



Fonte: Adaptado de Volpato; Foggiatto (2009).

## 3.2 *FRAMEWORK* PROPOSTO

O *framework* proposto divide a otimização dos movimentos de reposicionamento em quatro passos principais (Figura 3.4). No passo um (1) são propostos métodos de pré-processamento objetivando reduzir o número de pontos de Cs, simplificando o problema e reduzindo o custo computacional. Este passo poderia ser empregado para qualquer proposta de otimização de rotas de Cs e CRSs. Os passos dois (2), três (3) e quatro (4) correspondem a uma proposta deste trabalho que envolvem basicamente definir rota preliminar, refinar a rota de Cs e depois gerar rotas de CRSs. Estes passos podem ser resolvidos usando diferentes algoritmos de otimização que se adequem às características do problema. Devido à estratégia de deposição sequencial adotada, os Cs e os CRSs devem ser processados separadamente, pois a rota dos CRSs depende da definição da rota dos Cs. O *framework* assume que os dados estão organizados em “ilhas”, como mencionado anteriormente.

Figura 3.4 – *Framework* do método de otimização proposto




Fonte: O autor (2019).

Passo 1 – Pré-processamento dos pontos

Como o problema é complexo para obter-se a solução ótima em um tempo computacional razoável alguns métodos de decomposição e simplificação foram propostos para reduzir o número de pontos (dados) em cada camada. Ao fazer isso, espera-se obter boas soluções (não ótimas) em um tempo computacional aceitável. O Passo 1 foi idealizado para ser genérico, ou seja, as simplificações propostas aqui servem para a sequência proposta nesse *framework* e também para outros métodos de solução que venham a ser utilizados futuramente.

(a) Estrutura de Dados - Matriz de distâncias

Uma informação relevante para o *framework* proposto é a matriz de distâncias que registra todas as distâncias euclidianas (bidimensionais) entre todos os pontos relevantes (Cs, CRSs, e os outros a serem criados e definidos em função da redução do número dos pontos apresentada a seguir) calculados usando a Equação 3.1. Os pontos são armazenados em uma estrutura de dados sendo definidos pelas suas coordenadas *x*-*y* ($x_1$-$y_1$, $x_2$-$y_2$, ..., $x_n$-$y_n$) e devidamente identificados por Cs, CRSs, etc.

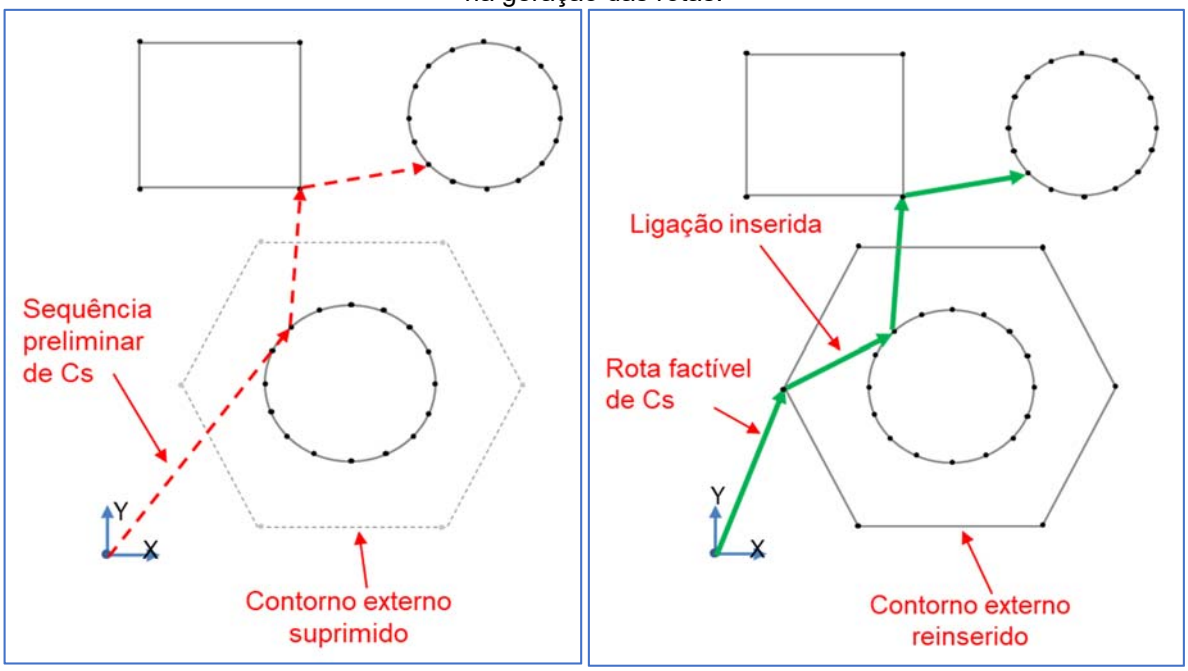
$$dist_{i,i+1} = \sum_{i,i}^{n} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}\,, \quad \forall i \in I \tag{3.1}$$

(b) Eliminação de pontos de Cs através da Análise das ilhas

Essa simplificação adotada tira proveito de um atributo das ilhas (definido no início desta seção). Se uma ilha tiver Cs internos, seu C externo não será considerado nesta etapa, ou seja, apenas os Cs internos serão analisados (Figura 3.5). Desta forma, no Passo 2 evita-se o uso dos pontos existentes nos Cs externos, simplificando o problema. Essa simplificação é baseada na observação de que, para atingir qualquer C interno, o reposicionamento do cabeçote extrusor deve cruzar o respectivo C externo.

Posteriormente, na etapa 3, cada C externo removido nesta fase será incluído entre a trajetória linear que conecta dois pontos de C, perturbando pouco a rota definida no passo 2.

Figura 3.5 – Análise das ilhas possibilitando a supressão (a) e posterior reinserção (b) do C externo na geração das rotas.




(a) (b)

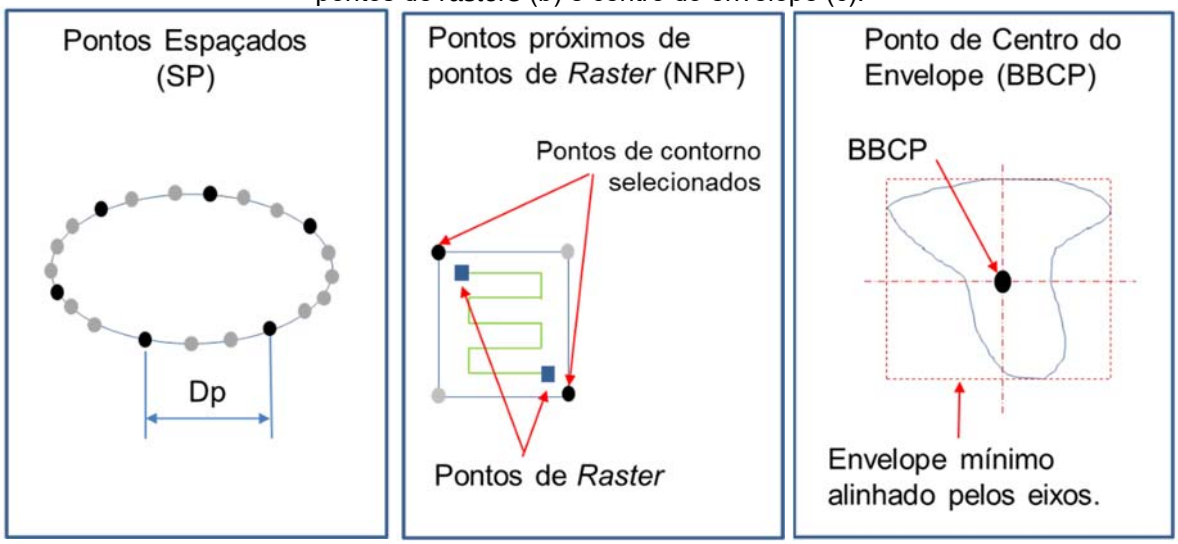
Fonte: O autor (2019).

(c) Pontos espaçados (*Spaced Points* – SP)

Usando a matriz de distâncias, essa opção de simplificação (poda de pontos) é um pré-tratamento aplicado em todos os Cs da camada, baseado na observação de

que existem pontos muito próximos entre si que podem ser desprezados. Neste caso, define-se uma distância mínima (*Dp*) que deve existir entre pontos subsequentes pertencentes a um mesmo C (Figura 3.6a), de forma que os pontos muito próximos serão descartados (e.g. $Dp \leq 2{,}5mm$). Essa distância foi definida arbitrariamente e pode ser ajustada ao lidar com Cs muito pequenos ou grandes. Esse método cria um conjunto de pontos espaçados (SP) e, dependendo do número de pontos de C existentes no problema, a expectativa inicial é que ele não seja eficiente para atender o passo 2 (a redução obtida no número de pontos pode não ser suficiente). No entanto, esta simplificação ajudará a reduzir a busca no passo 3.

Figura 3.6 – Métodos de simplificação dos pontos de contorno: pontos espaçados (a), próximo dos pontos de *rasters* (b) e centro do envelope (c).




(a) (b) (c)

Fonte: O autor (2019).

(d) Pontos próximos dos pontos de *Raster (Near Raster Points* – NRP*)*

Essa opção de simplificação dos pontos de C considera apenas os pontos próximos aos pontos dos CRS (pontos inicial e final), conforme mostrado na Figura 3.6b. Isso significa que todos os pontos de um C serão considerados como 2 (dois), 4 (quatro), 6 (seis), etc, variando de acordo com o número de CRSs dentro do C. Essa ideia de poda vem da observação do processo, ou seja, a sequência de deposição, onde, uma vez que um C é depositado, o próximo ponto será um ponto de um CRS. Isso pode ter alguma vantagem principalmente se for adotada futuramente uma estratégia de deposição intercalada.

(e) Centro do Envelope (*Bounding Box Center Point* – BBCP)

Nesta etapa, é obtido o envelope mínimo que contém cada C, ou seja, um polígono circunscrito passando pelos seus pontos extremos (Figura 3.6c) e calculado o seu ponto central (BBCP). Ao fazer isso, todos os pontos pertencentes a um determinado C são substituídos por apenas um (1) ponto. O envelope mínimo pode ser facilmente obtido através das coordenadas mínimas e máximas nos eixos x e y dos pontos de cada C a partir da estrutura de dados acima. O conjunto de BBCPs gerados será utilizado em diferentes estágios e com diferentes finalidades no *framework.*

Passo 2: Sequência preliminar de Contornos

A definição da rota entre Cs foi dividida em duas etapas para reduzir o espaço de busca e o custo computacional, principalmente para os modelos MILP. No passo 2, com base nas simplificações adotadas no passo 1, é gerada uma sequência preliminar de Cs. Então, no passo 3, uma rota factível entre Cs é criada com base nessa sequência preliminar.

Identificação da vizinhança

Para reduzir ainda mais o espaço de busca no Passo 2 e com isso o custo computacional, foi aplicado um caso especial de identificação de vizinhança proposto por Arya *et al.* (1998). A ideia principal é identificar os Cs que estão próximos um do outro. Em outras palavras, criar uma relação de vizinhança entre os Cs. Um conjunto de vizinhança é formado para cada C. O ponto de origem (Or) também é incluído neste procedimento e os conjuntos de vizinhanças formados são então comparados e somados de forma a identificar os vizinhos dos vizinhos (vizinhos recíprocos).

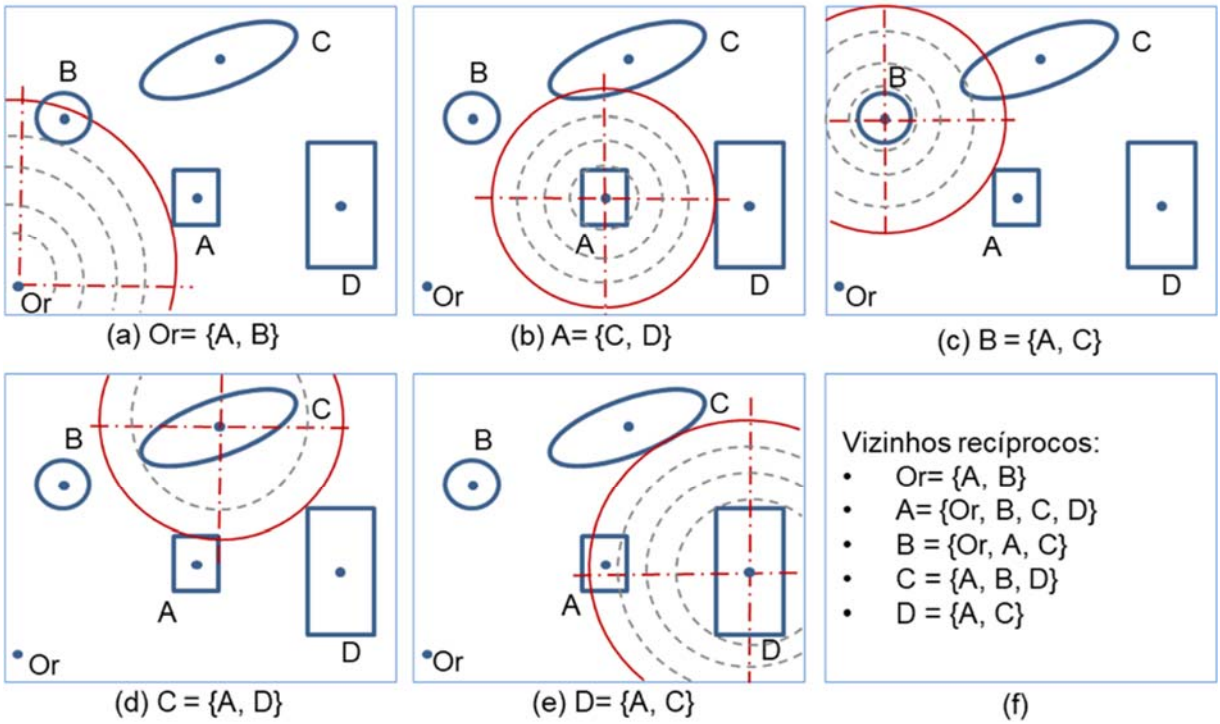
A Figura 3.7 ilustra graficamente a geração dos conjuntos de vizinhança para uma camada específica. As Figuras 3.7a até 3.7e apresentam o conjunto de vizinhos para cada C, enquanto a Figura 3.7f indica os vizinhos recíprocos.

Neste exemplo, devido à análise de vizinhos recíprocos, o contorno A terá quatro vizinhos (Or, B, C, D) em vez dos dois (C, D) identificados anteriormente.

O pseudocódigo para fazer isso é apresentado a seguir:

1. Para cada C, gere um círculo a partir do ponto BBCP e de raio Ri igual a menor aresta do envelope;
2. Enquanto o critério de parada não for alcançado faça:
   Obs: O critério de parada foi definido para encontrar dois vizinhos ou alcançar o canto mais distante da área de deposição.
   Procure interseções do círculo com cada C (usando o conjunto de SP).
3. Gere círculos de $R = Ri + 5mm$
   Retorne o conjunto de vizinhos do C atual.
4. fim;

Figura 3.7 – Representação esquemática da geração da vizinhança de uma camada com uma origem e quatro Cs: vizinhança de Or (a); vizinhança de A (b); vizinhança de B (c); vizinhança de C (d); vizinhança de D (e) e conjunto dos vizinhos recíprocos (f)




Fonte: O autor (2019).

Para evitar a ocorrência de *clusters* (aglomerações) de Cs desconectados na formação de conjuntos de vizinhança (ou seja, para verificar a existência de subciclos de grupos isolados), foi proposto um teste seguindo o pseudocódigo a seguir:

1. $i$ =1 (conte o número de *clusters*);
2. Adicione todos os Cs ao conjunto *NEWCLUSTER*;
3. Gere um teste de agrupamento para o conjunto *NEWCLUSTER*, conforme indicado na revisão de Saxena *et al.* (2017);

4. Adicione todos os Cs pertencentes ao ciclo identificado ao conjunto *CLUSTER* (*i*);
5. Todos os Cs $\in$ *NEWCLUSTER* estão incluídos *CLUSTER* (*i*)?
    a. Se SIM, interrompa a busca (vá para a linha 8);
    b. Se NÃO, execute a linha 6;
6. $i = i + 1$;
7. Identifique todos os Cs $\in$ *NEWCLUSTER* que não pertencem ao *CLUSTER* (*i*) e forme um novo conjunto de Cs (*NEWCLUSTER*). Retorne à linha 3;
8. Se $i > 1$ então,
    a. Para todos os Cs, encontre a menor distância entre *clusters* (*single-linkage clustering* apresentado por Saxena *et al.*, 2017) – para todos os *clusters* existentes - e inclua-o no conjunto de vizinhanças do C analisado;
9. fim;

Existem muitos métodos para análise de *clusters* descritos na literatura. Uma extensa revisão foi apresentada por Saxena *et al.* (2017). Na linha 1 do pseudocódigo, há pelo menos um *cluster* por camada. Na linha 3 do pseudocódigo, um teste de agrupamento verifica se todos os Cs identificados na formação de conjuntos de vizinhança estão contidos no CLUSTER (i). Se não, as linhas 5 a 7 verificam todos os Cs que não pertencem a *clusters* identificados anteriormente e formam um novo *cluster*. As linhas 3 a 7 são repetidas até que nenhum *cluster* adicional seja encontrado. Na linha 8, se houver mais de um *cluster*, será encontrada a menor distância entre os Cs de dois *clusters* diferentes, atualizando o conjunto de vizinhanças para todos os Cs.

No Passo 2, o método de otimização escolhido (MILP ou BT - detalhados no capítulo 4) é aplicado para conectar todos os Cs seguindo a restrição do conjunto de vizinhança, em uma sequência que minimizará a distância para visitar todos os Cs. A matriz de distâncias fornecida pelo Passo 1 é usada para realizar esta análise. Essa implementação pressupõe que é improvável que os *links* entre Cs fora da vizinhança tragam melhorias na função objetivo. Ao fazer isso, uma restrição é criada. Neste *framework*, este passo pode ser realizado considerando qualquer uma das três opções de simplificação apresentadas anteriormente (SP, NRP e BBCP). Devido a esta e outras simplificações adotadas no Passo 1 (e.g. nem todos os Cs são considerados), a sequência de Cs gerada pode não representar uma rota válida. Uma rota não válida é observada principalmente quando ocorre a simplificação com a

supressão de Cs externos proposta na análise das ilhas e quando as ilhas são representadas pelos pontos BBCP (centro do envelope).

Passo 3: Rota factível de Contornos

Este passo consiste em criar uma rota factível entre Cs baseado na sequência preliminar fornecida pelo Passo 2, aplicando um caso especial do conceito de Polígonos de Visibilidade (*Visibility Poligons* - VP) apresentado por Joe e Simpson (1987). O polígono de visibilidade é formado pelos pontos "visíveis" entre Cs adjacentes na sequência preliminar de Cs. Consiste em identificar apenas os conjuntos de pontos capazes de criar uma solução aprimorada, descartando todos os pontos que provavelmente não trarão melhorias para a função objetivo. O procedimento de formação dos VPs pode ser descrito pelo seguinte pseudocódigo:

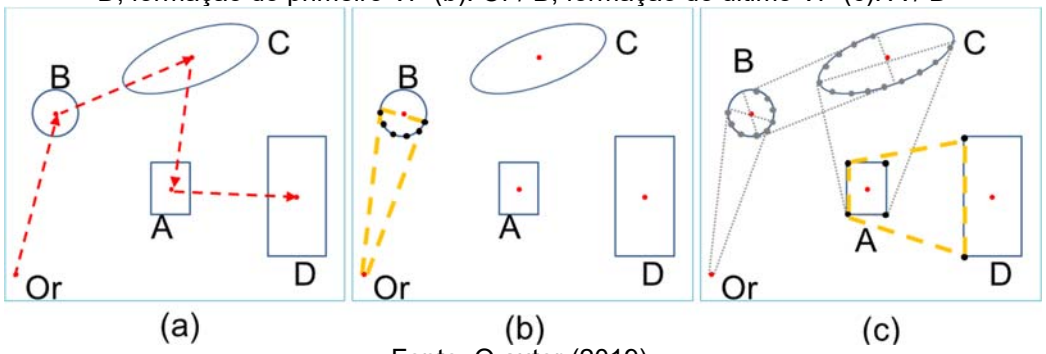
1. Dada uma sequência preliminar de Cs:
2. Desenhe duas linhas tangentes aos dois Cs analisados, conectando as extremidades desses Cs adjacentes, formando um VP;
3. Identifique os intervalos dos pontos (do conjunto SP) pertencentes a cada C contidos pelo VP;
4. Retorne um conjunto de pontos VP associados a cada C.

A Figura 3.8 exemplifica o processo de criação dos VPs para uma sequência preliminar de Cs obtida no Passo 2: Or / B / C / A / D (Figura 3.8a). As Figuras 3.8b e 3.8c mostram o primeiro VP gerado entre Or e B e o último entre A e D, respectivamente.

Seguindo a linha 2 do pseudocódigo, o VP formará um polígono contendo pontos pertencentes ao conjunto SP de cada C. Uma rota de Cs factível é obtida a partir da escolha do melhor par de pontos dentro desse conjunto VP. A rota de reposicionamento é gerada com base nos pontos do conjunto SP contidos nos VPs, também usando a matriz de distância fornecida pelo Passo 1.

Ressalta-se que esse é outro método inovador utilizado para reduzir o tamanho do problema, evitando o uso de todos os pontos na análise.

Figura 3.8 – Polígonos de visibilidade (VPs) para uma sequência preliminar de Cs (a): OR / B / C / A / D; formação do primeiro VP (b): Or / B; formação do último VP (c): A / D




Fonte: O autor (2019).

Passo 4: Rota de *Rasters*

A determinação da rota de CRSs é semelhante à rota de Cs descrita no Passo 3, ou seja, usando a matriz de distâncias fornecida pelo Passo 1 e as restrições de vizinhança obtidas no Passo 2. Para reduzir o número total de conexões possíveis e, consequentemente, reduzir o custo computacional, definiu-se que os CRSs pertencentes à mesma ilha sejam ligados entre si na rota, mas sem uma imposição de ordem. Sem essa restrição, a busca pode levar mais tempo para convergir devido ao tamanho do espaço de pesquisa a ser explorado. Como mencionado anteriormente, a principal diferença entre Cs e CRSs é que cada CRS possui um par de pontos e, por esse motivo, os pontos de entrada e saída não são coincidentes. A otimização dos CRSs começa no final da rota de Cs (definido como "*First*"), percorrendo todos os CRSs. A sequência que obtiver o menor valor da função objetivo é definida com a sequência de CRS a ser depositada. O último ponto CRS a ser visitado (definido como "*Last*") se torna o ponto inicial da próxima camada de impressão.

Seguindo as recomendações de Marti e Reinelt (2011) (seção 2.2.10), optou-se por testar o *framework* proposto através de um procedimento clássico de otimização utilizando-se de modelos de Programação Linear Inteira Mista (MILP) como *benchmark* de comparação para a metaheurística de Busca Tabu (BT). A escolha dos métodos baseou-se nas características do problema e propriedades do espaço de busca, onde os métodos propostos mostram-se adequados para otimização de problemas combinatórios.

## 3.3 FERRAMENTAS COMPUTACIONAIS

Algumas etapas dentro do *framework* proposto foram realizadas manualmente (não automatizadas), pois o objetivo desta tese é testar o funcionamento do método proposto para a otimização das distâncias de reposicionamento. No Passo 1, a obtenção das coordenadas, análise das ilhas, pontos espaçados e pontos próximos dos pontos de *raster* são realizadas manualmente. Nos Passos 2 e 3, a identificação da vizinhança e a formação dos polígonos de visibilidade, respectivamente, são realizadas manualmente. Com os dados devidamente organizados, a solução de cada passo é obtida automaticamente através do ambiente de modelagem GAMS e dos *solvers* de Busca Tabu. Posteriormente, algumas das rotas obtidas foram desenhadas e analisadas manualmente, conforme apresentado no Capítulo 5.

Para o desenvolvimento de modelos matemáticos encontram-se ambientes de modelagem específicos para modelagem e programação matemática como: *Lingo*, *MatLab®, OPL Studio, Excel/Solver, GAMS*, entre outros. Em geral, estes ambientes de modelagem também possuem *solvers* de resolução como CPLEX, Gurobi, Lingo, entre outros. Ou seja, necessita-se implementar o modelo matemático e resolvê-lo utilizando o *solver* escolhido. Os modelos MILP propostos foram implementados utilizando-se o ambiente de modelagem GAMS (*General Algebraic Modeling System*) versão 23.3.3, que é um sistema de modelagem de alto nível para programação matemática. O sistema GAMS possui *solvers* comerciais robustos e consolidados, permitindo aplicações de modelagem complexas e em larga escala, entre os quais foi selecionado o *solver* CPLEX 12.1.0 que é reconhecidamente eficiente para este tipo de aplicação (MITTELMANN, 2007).

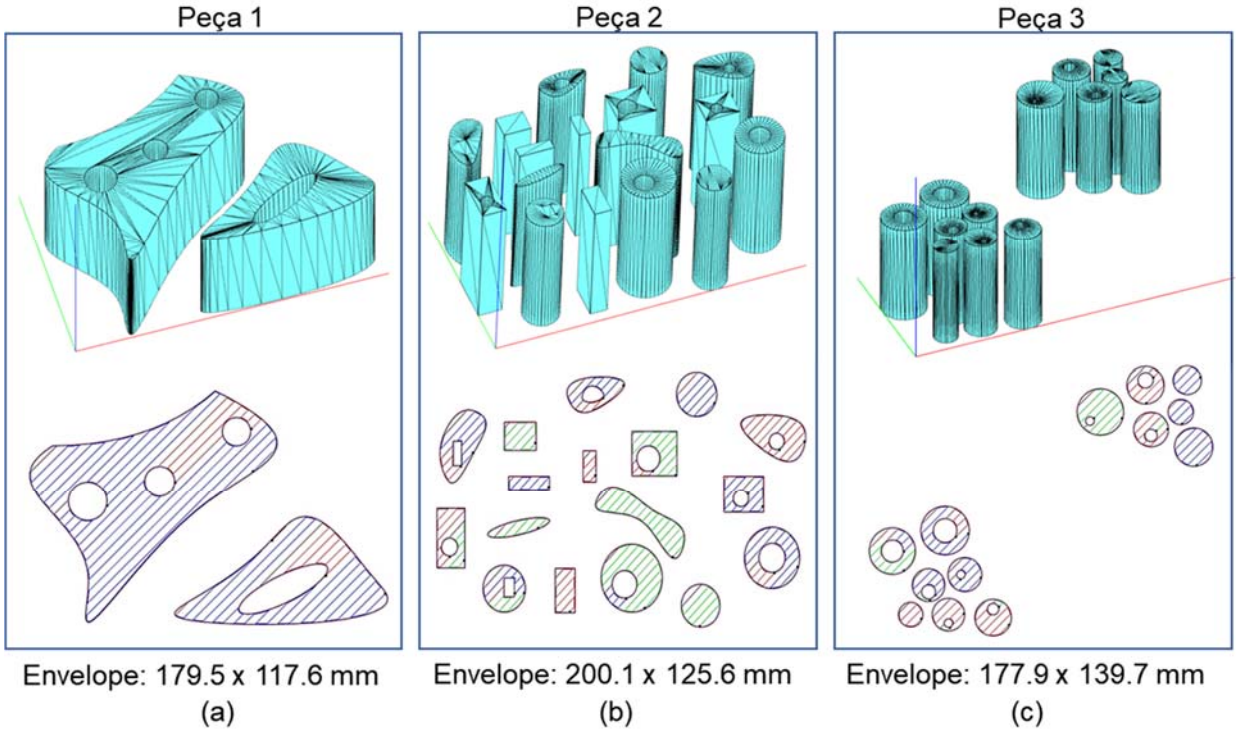
As metaheurísticas de BT foram implementadas em linguagem C++ no ambiente de modelagem Visual Studio 2017 utilizando o compilador MSVC e utilizando *flags* de otimização ativa /Ox.

Todos os experimentos foram executados em um computador com as seguintes configurações: Intel Core i5-3210M CPU @ 2.5Ghz, 8GB de RAM e sistema operacional Windows 10 Pro 64-bit.

## 3.4 GEOMETRIAS SELECIONADAS PARA ANALISAR O *FRAMEWORK* PROPOSTO

Três peças propostas por Volpato *et al.* (2019) foram utilizadas para testar o *framework* proposto. As geometrias apresentadas na Figura 3.9 representam diferentes configurações de deposição partindo de geometrias consideradas simples até geometrias mais complexas (Cs simples, Cs simples com furos, múltiplos Cs posicionados aleatoriamente). A peça 1 (Figura 3.9a) é considerada simples e possui apenas duas ilhas. A peça 2 (Figura 3.9b) tem como característica múltiplas ilhas dispostas aleatoriamente. A peça 3 (Figura 3.9c) possui múltiplas ilhas dispostas em dois *clusters*.

Figura 3.9 – Geometrias de teste que serão utilizadas na análise do método: peça 1 (a), peça 2 (b) e peça 3 (c)




Fonte: Volpato *et al.* (2019)

Estas geometrias podem representar boa parte dos problemas reais. As peças foram obtidas por um comando de extrusão no sistema CAD (*Computer-Aided Design*), portanto, as geometrias das fatias são semelhantes em todas as camadas. Os dados geométricos (pontos de coordenadas dos Cs e CRSs) foram obtidos de um sistema de planejamento de processo AM chamado *Rapid Prototyping Process*

*Planning* (RP3) (VOLPATO; FOGGIATTO, 2009). O ângulo de *raster* é incrementado em 90 graus entre as camadas, portanto, observou-se que os resultados obtidos nas cinco primeiras camadas podem ser considerados representativos para todas as camadas. Por esse motivo, apenas as cinco primeiras camadas foram otimizadas neste estudo. Os resultados obtidos foram analisados considerando as três opções de simplificação no Passo 2 (SP, NRP e BBCP) para a peça 1 e duas opções de simplificação para as peças 2 e 3 (NRP e BBCP).

Assumindo uma velocidade de reposicionamento do cabeçote extrusor de 20,32 mm/s (HAN *et al.,* 2003), também foi estimada a redução do tempo improdutivo.

Os resultados obtidos foram analisados considerando os seguintes critérios:

- Comprimento das distâncias de reposicionamento (mm);
- Tempo não produtivo (ocioso) estimado (s);
- Tempo computacional (s);

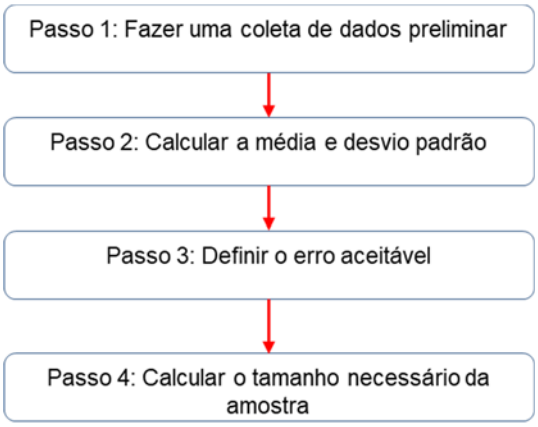
As rotas geradas foram comparadas com a Não Otimizada (NO) e obtidas com os algoritmos Guloso e Inserção do Vizinho Mais Próximo com 2-opt (NI2OPT), propostas por VOLPATO *et al.* (2019).

## 3.5 CÁLCULO DO TAMANHO AMOSTRAL

Existem diversos métodos de amostragem descritos em literatura (MEYER; 2000; BUSSAB; MORETTIN, 2006; SPIEGEL, 1977, entre outros). Neste estudo, o cálculo do tamanho amostral foi utilizado apenas na determinação do número de repetições realizadas para obter uma média confiável de resultados para a metaheurística de BT, pois a BT parte de uma solução inicial aleatória e pode apresentar diferentes resultados para o mesmo problema.

A amostragem aleatória simples é a técnica de amostragem probabilística mais elementar e pode ser utilizada para diversos problemas. O cálculo do tamanho da amostra pode ser um problema complexo e depende dos tipos de parâmetros que estão sendo analisados (proporção, média, etc.). A extensão da amostra determina quão representativa ela é e depende principalmente dos seguintes fatores: amplitude do universo, nível de confiança estabelecido e erro máximo permitido. Basicamente, a definição do tamanho da amostra pode ser divido em quatro passos (Figura 3.10):

Figura 3.10 – Passos utilizados na determinação do tamanho das amostras




Fonte: O autor (2019).

Walpole e Myers (2009) citam que o número exato de amostragens *n* deve ser determinado a partir de uma dedução da expressão do intervalo de confiança da distribuição por amostragem da média de uma variável distribuída normalmente, resultando na Equação 3.2:

$$n \geq \left(\frac{2t_{\alpha/2;GL}S}{k\bar{x}}\right)^2 \tag{3.2}$$

Neste cálculo têm-se os seguintes parâmetros:

- Média amostral $x$;
- Desvio padrão amostral $S$;
- Grau de confiança α;
- Número de amostras $n$;
- Coeficiente da distribuição normal padrão para uma probabilidade determinada: $\frac{t_\alpha}{2}; GL$ (calculado ou tabelado);
- Tolerância desejada $k$;

Martins e Laugeni (2005) citam que, na prática, normalmente o número de amostragem preliminar deve ser em torno de 10 execuções, com grau de confiança entre de 90% e 95% e o erro relativo variando entre 5% e 10%.O caso utilizado na definição do número de amostras foi a peça 2 com 2500 diversificações, pois apresentou o maior desvio padrão. A Tabela 3.1 apresenta os parâmetros utilizados na Equação 3.2 e a Tabela 3.2 apresenta o cálculo do tamanho amostral.

Tabela 3.1 – Parâmetros utilizados na definição da quantidade de amostras necessárias

| **Parâmetros** | |
|---|---|
| Grau confiança | 95% |
| Valor *Alpha* | 5% |
| Tolerância (*k*) | 10% |
| Amostras iniciais (*n*) | 10 |
| Graus liberdade | 9 |
| Valor T (*alpha/2;GL*) | 2.262 |

Fonte: O autor (2019).

Tabela 3.2 – Valores obtidos no cálculo da quantidade amostral

| **AMOSTRAS INICIAIS** | **Contorno** | | ***Raster*** | |
|---|---|---|---|---|
| | **Diversificações** | | **Diversificações** | |
| | **2500** | **5000** | **2500** | **5000** |
| **1** | 543,98 | 542,28 | 604,23 | 628,86 |
| **2** | 611,47 | 540,65 | 617,28 | 585,69 |
| **3** | 593,42 | 552,68 | 613,20 | 599,62 |
| **4** | 559,15 | 550,33 | 634,39 | 628,07 |
| **5** | 559,82 | 560,21 | 637,23 | 607,73 |
| **6** | 551,18 | 552,77 | 640,19 | 615,18 |
| **7** | 555,14 | 551,90 | 631,08 | 643,55 |
| **8** | 547,58 | 544,81 | 653,45 | 606,76 |
| **9** | 541,46 | 528,88 | 627,43 | 610,18 |
| **10** | 575,58 | 542,28 | 607,66 | 615,54 |
| **MÉDIA** | **563,88** | **546,68** | **626,61** | **614,12** |
| **DESVIO PADRÃO** | **22,89** | 8,77 | 15,73 | 16,36 |
| **Amostras necessárias (*n*)** | 3,37 | 0,53 | 1,29 | 1,45 |
| **Amostras realizadas (*n*)** | **5** | **5** | **5** | **5** |

Fonte: O autor (2019).

Conforme observado na Tabela 3.2, a simulação que teve maior variação nos resultados foi a definição da rota de Cs, necessitando 3,37 amostras. Baseado neste resultado foi definido um número de 5 amostras como suficientes para obter uma confiabilidade de 95% nos resultados.

# 4 ABORDAGENS PROPOSTAS

Nesta seção são apresentados os modelos MILP e Busca Tabu que foram implementados na solução dos problemas utilizando o *framework* proposto.

## 4.1 MODELOS MILP

A fim de apresentar o modelo matemático, antes, são definidos os índices, parâmetros, variáveis e conjuntos que restringem o âmbito de restrições e variáveis. Estes estão apresentados nos Quadros 4.1, 4.2, 4.3 que estabelecem, respectivamente.

Quadro 4.1 - Índices e parâmetros usados nos modelos matemáticos

| | |
|---|---|
| $i, j$ | Cs presentes no problema |
| $k$ | Posição dos Cs |
| $kr$ | Posição dos RS |
| $m$ | CRSs presentes no problema |
| $p, p_j$ | Pontos de Cs ou CRSs |
| $dist_{p,pj}$ | Distância entre o ponto $p$ e $p_j$ |

Fonte: O autor (2019).

Quadro 4.2 - Conjuntos usados nos modelos matemáticos

| | |
|---|---|
| $ASSIGN(i)$ | Conjunto que define a sequência de deposição dos Cs *i* |
| $CT(i)$ | Conjunto dos pontos associados ao C *i* |
| $First(p)$ | Conjunto associado ao último ponto de C visitado |
| $Last(p)$ | Conjunto associado ao último ponto de CRS visitado |
| $LC(i, j)$ | Conjunto que identifica que os Cs *i* e *j* ocupam, respectivamente, as posições *k* e *k+1* |
| $LIM(i)$ | Conjunto de pontos do VP associados a um C *i* |
| $Pair(m)$ | Conjunto de pontos que define um CRS |
| $PTraster$ | Conjunto de todos os pontos de CRS |

Fonte: O autor (2019).

Quadro 4.3 - Variáveis usadas nos modelos matemáticos

| | |
|---|---|
| $linkbn_{p,\,p\,j,\,kr}=1$ | Indica que os pontos *p* e $p_j$ estarão ligados na posição *kr,* formando um CRS. Caso contrário, esta variável será nula. |
| $link_{p,p\,j,k}=1$ | Indica que os pontos *p* e $p_j$ estarão ligados na posição *k* da rota de Cs. Caso contrário, esta variável será nula. |
| $link_{p,p\,j,kr}=1$ | Indica que os pontos *p* e $p_j$ estarão ligados na posição *kr* da rota de CRSs. Caso contrário, esta variável será nula. |
| $pt\_rank_{p,k}=1$ | Indica que o ponto *p* está "ativo" (no C *i*) e ocupa a posição *k* da rota de Cs. Caso contrário, esta variável será nula. |
| $pt\_rank_{p,kr}=1$ | Indica que o ponto *p* está "ativo" (no CRS *m*) e ocupa a posição *kr* da rota de CRSs. Caso contrário, esta variável será nula. |
| $rank_{i,k}=1$ | Indica que o C *i* ocupa a posição *k* da rota de Cs. Caso contrário, esta variável será nula. |
| $rank_{m,kr}=1$ | Indica que o CRS *m* ocupa a posição *kr* da rota de CRSs. Caso contrário, esta variável será nula. |
| $rank\_rst_{p,kr}=1$ | Indica que o ponto *p* está "ativo" (no CRS *m*) e ocupa a posição *kr* da rota de CRSs. Caso contrário, esta variável será nula. |

Fonte: O autor (2019).

Tabela 4.1 – Critérios de parada adotados nos modelos MILP

| **Critérios de parada** | |
|---|---|
| *Gap* de integralidade | 15% |
| Tempo de execução | 7200 segundos |

Fonte: O autor (2019).

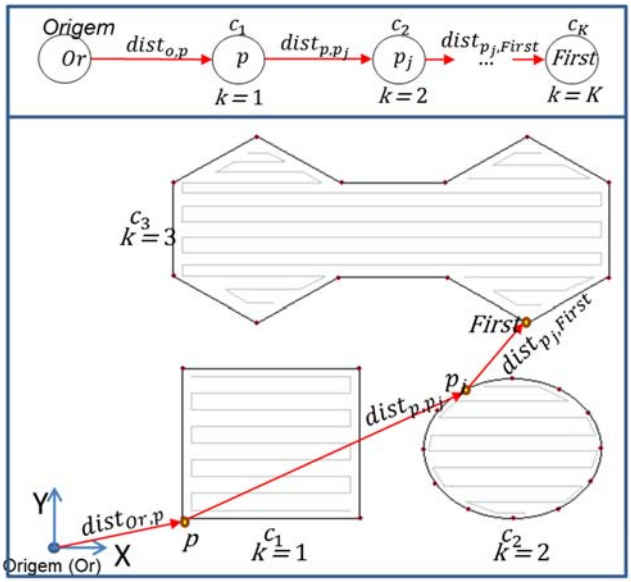
### 4.1.1 Modelo matemático proposto para definição da sequência preliminar de Cs (Passo 2)

A Figura 4.1 apresenta de maneira simplificada o modelo matemático proposto para a solução de otimização do reposicionamento entre contornos, conforme detalhado na sequência.

O deslocamento do cabeçote extrusor inicia-se na origem (Or) (posição *home* da área de trabalho ou último ponto da camada anterior), devendo percorrer todos os Cs, conforme sequência de pontos definidos pelo algoritmo. No exemplo apresentado na Figura 4.1, a trajetória de reposicionamento inicia-se na origem e percorre os pontos $p$, $p_j$ e $First$ que pertencem aos Cs $c_1$, $c_2$ e $c_3$, respectivamente. O

deslocamento total é calculado pelo somatório das distâncias entre estes pontos, representados pelas setas.

Figura 4.1– Representação simplificada do método proposto para otimização de reposicionamento entre contornos




Fonte: O autor (2019).

A função objetivo, Equação 4.1, visa minimizar o deslocamento total, partindo do ponto “Or” e percorrendo apenas uma camada (sem retornar à origem). A restrição apresentada na Equação 4.2 é gerada para toda posição *k* (da sequência preliminar de Cs) e impõe que qualquer posição *k* tem que ser ocupada por apenas um C *i*, necessariamente. De forma similar, a restrição apresentada na Equação 4.3 é gerada para todo C *i* e impõe que qualquer C *i* tem que ocupar uma posição *k*, necessariamente.

$$MinZ1 = \sum_{i} \sum_{j>i} \sum_{p \in CT(i)} \sum_{p_j \in CT(j)} \sum_{k} dist_{p,p_j} * link_{p,p_j,k} \tag{4.1}$$

Sujeito a:

$$\sum_{i} rank_{i,k} = 1 \ \forall k \tag{4.2}$$

$$\sum_{k} rank_{i,k} = 1 \ \forall i \tag{4.3}$$

Como a variável binária $rank_{i,k}$ = 1 identifica que o C *i* ocupa a posição *k* da sequência preliminar de Cs, na restrição apresentada na Equação 4.4, se $rank_{i,k}$ = 1, necessariamente haverá um ponto *p*, pertencente ao C *i*, que estará "ativado" (onde o C é iniciado e finalizado), ou seja, onde $pt\_rank_{p,k}$ = 1.

$$\sum_{p \in CT(i)} pt_{rank_{p,k}} = rank_{i,k}, \forall i, k \tag{4.4}$$

Na restrição 4.5, a variável binária $link_{p,p_j,k} = 1$ se e somente se os pontos *p* e $p_j$ estão "ativos" nas posições *k* e *k+1*. Para evitar soluções redundantes (realizar cortes válidos), como $link_{p,p_j,k} = 1$ ou $link_{p_j,p,k} = 1$, que são equivalentes, na restrição 4.5, os pontos *p* e $p_j$ podem ocupar qualquer uma das posições *k* ou *k+1* se $link_{p,p_j,k} = 1$.

$$link_{p,p_j,k} \geq pt_{rank_{p,k}} + pt_{rank_{p_j,k}} + pt_{rank_{p,k+1}} + pt_{rank_{p_j,k+1}} - 1 \tag{4.5}$$

$$\forall i, j > i, k < K, p \in CT(i), p_j \in CT(j)$$

A restrição 4.6 contribui para a convergência da busca. Basicamente, se o ponto *p* está "ativo" na variável binária $link_{p,p_j,k} = 1$ ou $link_{p_j,p,k} = 1$, então, necessariamente, $pt\_rank_{p,k}$ = 1 ou $pt\_rank_{p,k+1}$ = 1. A restrição 4.7 define as variáveis binárias envolvidas neste modelo matemático.

$$\sum_{\substack{j \\ j>i}} \sum_{p_j \in CT(j)} link_{p,p_j,k} + \sum_{\substack{j \\ j<i}} \sum_{p_j \in CT(j)} link_{p_j,p,k} \leq pt_{rank_{p,k}} + pt_{rank_{p,k+1}} \tag{4.6}$$

$$\forall i, p \in CT(i), k < K$$

$link_{p,p_j,k}$ , $pt\_rank_{p,k}$ , $rank_{i,k}$ são variáreis binárias (4.7)

$$\forall p \in CT(i), p_j \in CT(j), k < K$$

### 4.1.2 Modelo Matemático Proposto para a Rota Factível de Cs (Passo 3)

Esse modelo é semelhante ao anterior, mas agora todos os Cs e o conjunto dos pontos SP são incluídos na análise, transformando a solução preliminar em uma solução factível. Portanto, a nova função objetivo é a Equação 4.8 (semelhante à Equação 4.1, mas com novos conjuntos de pontos). Para um C *i*, *LIM*(*i*) limita a busca aos pontos do VP associados a este C *i.*

$$Min\ Z2 = \sum_{i} \sum_{j>i} \sum_{p \in LIM(i)} \sum_{p_j \in LIM(j)} \sum_{\substack{k<K \\ k \in LC(i,j)}} dist_{p,p_j} * link_{p,p_j,k} \tag{4.8}$$

Sujeito a:

$$\sum_{p \in LIM(i)} pt_{rank_{p,k}} = 1, \forall i, k \in ASSIGN(i) \tag{4.9}$$

Como, neste terceiro passo, a sequência preliminar de Cs é conhecida, identificada por *k ∈ ASSIGN (i),* a restrição imposta na Equação 4.9 é gerada para todo C *i*, impondo que necessariamente haverá um ponto *p* ∈ *LIM*(*i*) onde ocorrerá o início/fim de deposição do C *i*. Na restrição 4.10, a variável binária $link_{p,p_j,k} = 1$ se e somente se os pontos *p* e $p_j$ estão “ativos”, ou seja, onde são iniciados e finalizados os Cs *i* e *j*, respectivamente, nas posições *k* e *k+1*. A principal diferença está na consideração da sequência de deposição dos Cs, que já é conhecida, identificada por *k* ∈ *ASSIGN*(*i*), por exemplo.

$$link_{p,p_j,k} \geq \underset{k \in ASSIGN(i)}{pt\_rank_{p,k}} + \underset{k \in ASSIGN(j)}{pt\_rank_{p_j,k}} + \underset{k+1 \in ASSIGN(i)}{pt\_rank_{p,k+1}} + \underset{k+1 \in ASSIGN(j)}{pt\_rank_{p_j,k+1}} - 1 \tag{4.10}$$

$$\forall\ i, j > i, k < K\ \therefore\ k \in LC(i,j), p \in LIM(i), p_j \in LIM(j)$$

A restrição 4.11 objetiva limitar a busca no conjunto de pontos *LIM*(*i*), ou seja, limita a busca aos pontos do conjunto VP considerado para este C (*i)*.

$$\sum_{\substack{j \\ j>i}} \sum_{p_j \in LIM(j)} link_{p,p_j,k} + \sum_{\substack{j \\ j<i}} \sum_{p_j \in LIM(j)} link_{p_j,p,k} \tag{4.11}$$

$$\leq pt_{rank_{p,k}} + pt_{rank_{p,k+1}}$$

$$\forall\, i, p \in LIM(i), k < K$$

Na restrição 4.12, a partir dos resultados do passo anterior (Passo 2), o conjunto *LC(i, j)* identifica quais Cs *i* e *j* ocupam, respectivamente, as posições *k* e *k + 1* da rota de Cs e, portanto, o conjunto *LC(i, j)* que está associado à posição *k*. Nesse caso, necessariamente, um par de pontos *p ∈ LIM(i)* e *pj ∈ LIM(j)*, pertencentes aos Cs *i* e *j*, respectivamente, estão associados à posição *k*, de modo que $link_{p,p_j,k} = 1$. Por fim, a restrição 4.13 define as variáveis binárias envolvidas nesse modelo matemático.

$$\sum_{p \in LIM(i)} \sum_{p_j \in LIM(j)} link_{p,p_j,k} = 1 \tag{4.12}$$

$$\forall\, i, j, k \in LC(i, j)$$

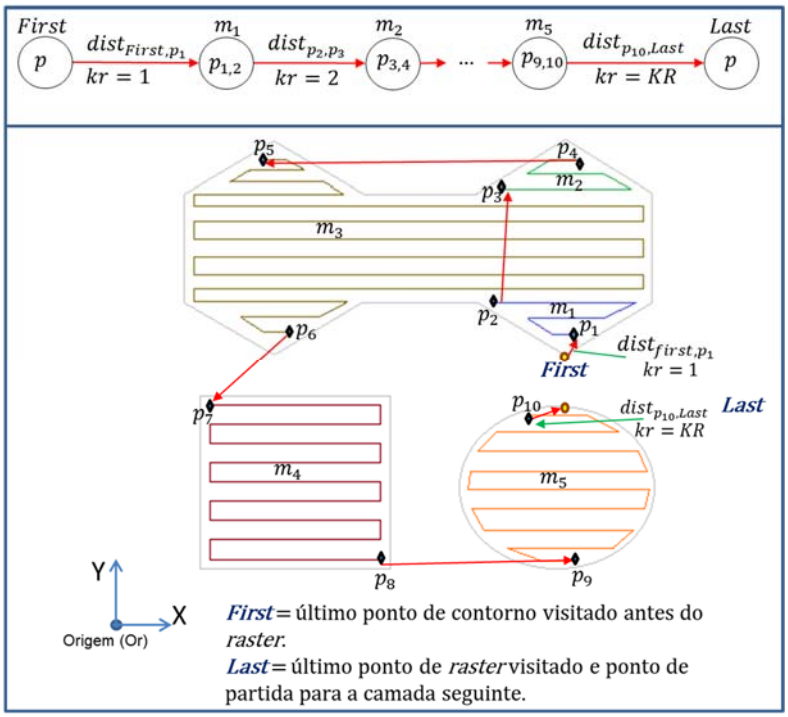
$$link_{p,p_j,k}\ ,\ pt\_rank_{p,k} \text{ são variáreis binárias} \tag{4.13}$$

$$\forall\ p \in LIM(i), p_j \in LIM(j), k < K$$

### 4.1.3 Modelo Matemático Proposto para a Rota de *Rasters* (Passo 4)

A rota de CRSs inicia-se a partir do último ponto de C visitado (ponto denominado *First).* Partindo de *First*, o cabeçote extrusor deve visitar todos os CRS (*m*). Ao terminar o último CRS o cabeçote extrusor será reposicionado em *Last,* onde estará o ponto de partida para a próxima camada. Na Figura 4.2, observa-se que o deslocamento total é calculado pela somatória das distâncias de reposicionamento representadas pelas setas.

Figura 4.2– Representação simplificada do modelo MILP proposto para a rota de CRSs




Fonte: O autor (2019).

A função objetivo, Equação 4.14, visa minimizar o deslocamento total, partindo do último ponto de C visitado ($p \in First$) e percorrendo apenas uma camada (sem retornar à origem). Terminando o CRSs, o último ponto visitado será o ponto de início da próxima camada ($p \in Last$).

$$Min\ Z3 = \sum_{p \in PTraster} \sum_{\substack{p_j \in PTraster \\ p < p_j}} \sum_{kr} dist_{p,p_j} * raster_{p,p_j,kr} + \sum_{p \in First} \sum_{p_j \in PTraster} \sum_{kr} dist_{p,p_j} * raster_{p,p_j,kr} \quad (4.14)$$

A restrição 4.15 impõe que cada ponto de CRS $p \in PTraster$ precisa ser posicionado na posição *kr*. Nesta restrição, no primeiro e no terceiro somatórios, os pontos *p* e *pj*, respectivamente, identificam o final de um CRS e um reposicionamento para o início do próximo CRS. A única diferença entre o primeiro e o terceiro somatório é, no primeiro caso, *p* < *pj* e, no segundo caso, *p* > *pj*. Isso é feito para eliminar soluções redundantes. Esses somatórios são gerados para cada posição *kr* dos CRSs

e exige que qualquer posição *kr* seja ocupada por apenas um segmento de reposicionamento, conectando os pontos *p* e *pj*. O segundo somatório, para *kr* = *1*, define o reposicionamento do ponto *pj*, que identifica o último ponto C visitado ($pj \in First$), até o ponto *p* ($p \in PTraster$), onde ocorre o início do primeiro CRS. O quarto somatório define o reposicionamento do ponto *p* ($p \in PTraster$), que identifica o ponto inicial da próxima camada para um ponto fictício (*dummy point*) *pj* ($pj \in Last$). No quarto somatório, o ponto *p* ($p \in PTraster$) é o local onde ocorre o final do último CRS, para *kr* = *KR*. Como o modelo matemático exige que cada ponto *p* ($p \in PTraster$) seja ligado a outro ponto (representando um reposicionamento) *pj* ($pj \in Last$) seria um ponto da próxima camada e não é considerado na função objetivo.

$$\sum_{\substack{p_j \in PTraster \\ p<p_j}} \sum_{1<kr<K} raster_{p,p_j,kr} + \sum_{p_j \in First} raster_{p_j,p,1} + \tag{4.15}$$

$$\sum_{\substack{p_j \in PTraster \\ p>p_j}} \sum_{1<kr<K} raster_{p_j,p,kr} + \sum_{p_j \in Last} raster_{p_j,p,KR} =$$

$$\forall p < P \wedge p \in PTraster$$

Como o reposicionamento implica a associação entre dois pontos, tal que $raster_{p,p_j,kr} = 1$, a restrição 4.16 impõe que esse reposicionamento ocorra necessariamente se o ponto *pj* for alocado na posição *kr* da sequência de pontos de CRS, ou seja, $rank_{rst_{p_j,kr}} = 1$.

Na restrição 4.17, a variável binária $raster_{p,p_j,1} = 1$ identifica o reposicionamento para a execução do primeiro CRS, movendo-se do ponto $p \in$ *First* para o ponto $p_j \in$ *PTraster*, onde o início do primeiro CRS ocorre, para *kr* = 1.

Da mesma forma, na restrição 4.18, a variável binária $raster_{p,p_j,KR} = 1$ identifica que o reposicionamento ocorrerá entre o último ponto $p_j \in$ *PTraster* do CRS e o ponto fictício *Last* ($p \in$ *Last*), que será o ponto de partida para a próxima camada.

$$\sum_{\substack{p \in PTraster \\ p<p_j}} raster_{p,p_j,kr} + \sum_{\substack{p_j \in PTraster \\ p>p_j}} raster_{p_j,p,kr} + \tag{4.16}$$

$$\sum_{p \in First} raster_{p,p_j,1} + \sum_{p \in Last} raster_{p,p_j,KR} \geq rank_{rst_{p_j,kr}}$$

$$\forall p_j \in PTraster, kr$$

$$\sum_{p_j \in PTraster} raster_{p,p_j 1} = 1 \forall p \in First \tag{4.17}$$

$$\sum_{p_j \in PTraster} raster_{p,p_j,KR} = 1 \forall p \in Last \tag{4.18}$$

As restrições 4.19, 4.20 e 4.21 exigem que exista apenas uma única variável binária $raster_{p,p_j,kr}$ = 1 em cada uma delas, gerando uma ordenação dos reposicionamentos. Na restrição 4.19, para *kr = 1,* o reposicionamento passará do ponto $p \in First$, até o ponto inicial do primeiro CRS $p_j \in PTraster$. Da mesma forma, na restrição 4.20, para $kr = KR$, o reposicionamento passará do ponto final do último CRS $p_j \in PTraster$ até o ponto $p \in Last$. A restrição 4.21 é gerada para cada posição *kr* dos CRSs, isto é, para *1<kr<KR*, impondo que qualquer posição *kr* de CRS seja ocupada por apenas um segmento de reposicionamento, conectando os pontos *p* e *pj*.

$$\sum_{p \in First} \sum_{p_j \in PTraster} raster_{p,p_j,1} = 1 \tag{4.19}$$

$$\sum_{p \in Last} \sum_{p_j \in PTraster} raster_{p,p_j,KR} = 1 \tag{4.20}$$

$$\sum_{p \in PTraster} \sum_{\substack{p_j \in PTraster \\ p< p_j}} raster_{p,p_j kr} = 1 \tag{4.21}$$

$$\forall\ 1 < kr < KR$$

A restrição 4.22 é gerada para toda sequência *kr* de reposicionamento e impõe que haja dois pontos associados a cada *kr*, já que o reposicionamento é composto

pelo deslocamento entre dois pontos. Em função disso, a restrição 4.23 impõe que qualquer ponto $p \in PTraster$ deve ser associado a um *kr*. Dois casos particulares disso, indicados nas restrições 4.24 e 4.25, são que o último ponto de C visitado ($p \in First$) e o ponto ($p \in Last$) de partida para a próxima camada (ponto fictício) serão associados às posições *kr = 1* e *kr = KR*, respectivamente.

$$\sum_{p \in First} rank_{rst_{p,1}} + \sum_{p \in Last} rank_{rst_{p,KR}} + \sum_{p \in PTraster} rank_{rst_{p,kr}} \leq 2 \forall\ kr \tag{4.22}$$

$$\sum_{kr} rank_{rst_{p,kr}} \geq 1 \forall\ p \in PTraster \tag{4.23}$$

$$rank_{rst_{p,1}} \geq 1\ \forall\ p \in First \tag{4.24}$$

$$rank_{rst_{p,KR}} \geq 1\ \forall\ p \in Last \tag{4.25}$$

A restrição 4.26 explora a associação entre um par *m* de pontos de cada CRS, $p \in Pair(m)$. Neste caso, um dos pontos, *p* ou $p_j$, estará na posição *kr* e outro na posição *kr+1*. Somente se os pontos *p* e $p_j$ estiverem nas posições *kr* e *kr+1*, ou vice-versa, como indicado no lado direito da restrição, é que teremos a variável binária $linkbn_{p,p_j,kr} = 1$.

$$linkbn_{p,p_j,kr} \geq rank_{rst_{p,kr}} + rank_{rst_{p_j,kr}} + rank_{rst_{p,kr+1}} + rank_{rst_{p_j,kr+1}} - 1 \tag{4.26}$$

$$\forall m, p \in Pair(m), p_j \in Pair(m), kr < KR$$

Como o CRS implica na associação entre dois pontos, tal que $linkbn_{p,p_j,kr} = 1$, na restrição 4.27, se o ponto *p* for alocado na posição *kr* isso impõe que o CRS (entre $p$ e $p_j$) necessariamente ocorrerá na posição *kr-1 ou kr*. A restrição 4.28 atua em conjunto com a restrição 4.27 e indica que cada CRS está associado a apenas um

par de pontos $p, p_j \in Pair(m)$, cuja posição *kr* na rota de reposicionamento precisa ser definida.

$$\sum_{\substack{p_j \in Pair(m) \\ p < p_j}} (linkbn_{p,p_j,kr} + linkbn_{p,p_j,kr-1}) + \tag{4.27}$$

$$\sum_{\substack{p_j \in Pair(m) \\ p > p_j}} (linkbn_{p_j,p,kr} + linkbn_{p_j,p,kr-1}) \geq rank\_rst_{p,kr}$$

$$\forall m, p \in Pair(m), kr$$

$$\sum_{p \in Pair(m)} \sum_{\substack{p_j \in Pair(m) \\ p < p_j}} \sum_{kr} linkbn_{p,p_j,kr} \leq 1 \qquad \forall\ m \tag{4.28}$$

## 4.2 METAHEURÍSTICAS DE BUSCA TABU (BT)

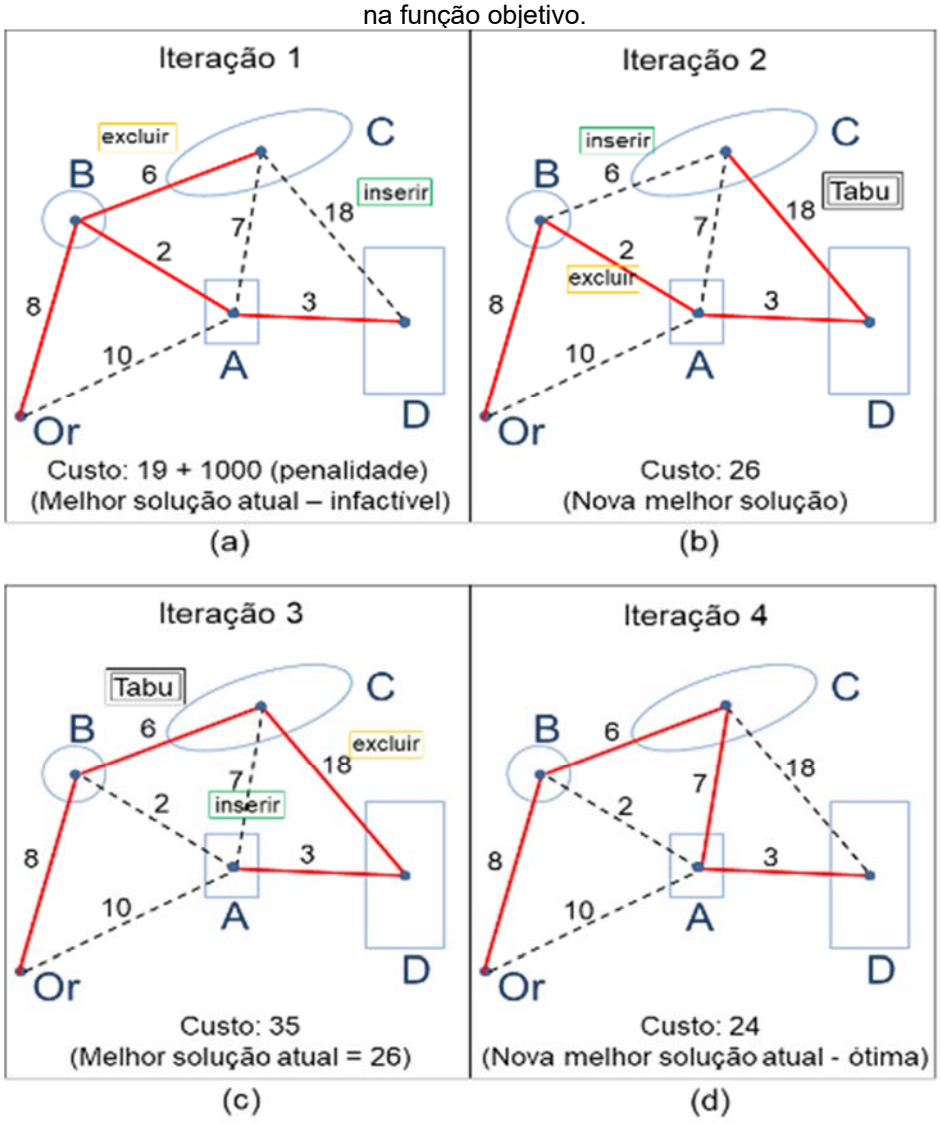
A BT será implementada no lugar dos métodos MILP para uma estratégia de deposição sequencial, seguindo o *framework* ilustrado na Figura 3.3, ou seja, será utilizada na geração da sequência preliminar de Cs (Passo 2), na rota factível de Cs (Passo 3) e ainda na rota de CRSs (Passo 4). Uma das vantagens da BT é a possibilidade de implementação de estrutura de memória (lista tabu), que pode ser adotada no intuito de alimentar a busca e para evitar que a busca fique presa em regiões de ótimos locais. Quando um movimento é realizado, este movimento torna-se proibido (ou tabu) por *N* iterações, ou seja, este movimento é inserido na lista tabu em uma tentativa de evitar ciclos do algoritmo.

A partir do pseudocódigo de BT apresentado na seção 2.2.8, apresenta-se, na Figura 4.3 o exemplo de aplicação da BT para uma camada com origem e quatro Cs (grafo de cinco nós), onde, considerando certas restrições deve-se conectar todos os nós com o custo mínimo. Existem duas restrições neste exemplo:

- Restrição 1: Se um C fizer mais de duas ligações deve-se aplicar penalidade de 1000 (valor definido arbitrariamente que inviabiliza a solução) por ligação (a solução é infactível pois o cabeçote extrusor não pode visitar duas vezes o mesmo ponto);

- Restrição 2: Se as ligações Or-D, Or-C ou B-D ocorrerem deve-se aplicar uma penalidade de 100 (valor definido arbitrariamente que penaliza o desrespeito da vizinhança mas não é demasiadamente elevado a ponto de inviabilizar o critério de aspiração) por ligação;

Figura 4.3 – Solução ilustrada de quatro iterações para uma camada com origem e quatro Cs: a cada iteração um movimento *swap* é realizado para inserir e excluir ligações entre Cs até atingir a melhora na função objetivo.




Fonte: O autor (2019).

Neste exemplo, realizam-se quatro iterações que mostram os movimentos (troca de ligações entre Cs denominados *swaps*) realizados para atingir o custo mínimo de movimentação. As iterações são repetidas até que o melhor resultado seja encontrado ou algum critério de parada for atingido. Para evitar voltar às soluções visitadas anteriormente, a cada iteração insere-se a ligação que gerou a solução vigente na lista Tabu. Um movimento Tabu poderá ser aceito apenas se resultar em

uma solução melhor do que a melhor solução encontrada anteriormente (critério de aspiração).

Na iteração 1 (Figura 4.3a) é gerada uma solução infactível, pois desrespeita a restrição 1. Na iteração 2 (Figura 4.3b) é explorada uma solução vizinha, ou seja, é excluída a ligação BC e inserida a ligação CD, obtendo melhora na função objetivo. Neste caso, o movimento que provocou melhora na função objetivo irá para a lista tabu e ficará proibido por uma iteração. Na solução vizinha da iteração 3 (Figura 4.3c) foi excluída a ligação AB e refeita a ligação BC, mas este movimento não resulta em melhora da função objetivo e não é considerado. Na solução vizinha obtida na iteração 4 (Figura 4.3d) é excluída a ligação CD e realizada a ligação AC, obtendo o melhor valor possível para a função objetivo, ou seja, a solução ótima, interrompendo-se então a busca. De forma geral, o passo 2 do *framework* é realizado de maneira similar ao exemplo da Figura 4.3. Utilizando-se das restrições de vizinhança geradas no passo 2 do *framework* define-se que a posição *i=1* sempre será ocupada pelo ponto Or e então testa-se a permutação de um C que está momentaneamente na posição *i* com o C na posição *i+1*, *i+2*, ... até *K,* (o número total de Cs que fazem parte do conjunto da vizinhança). Quando a troca produzir melhora no valor da função objetivo, ela será aceita como nova solução vigente e esse movimento irá para a lista tabu por um número definido de iterações. No passo 3, explorando o fato de que a sequência preliminar (solução guia) de Cs já é conhecida, realiza-se a intensificação da busca na vizinhança através de um procedimento de *path-relinking* uilizando os pontos contidos nos conjuntos de VPs. No passo 4, o cálculo é realizado de maneira semelhante ao passo 2, ou seja, cada CRS(*m*) formado por um par de pontos é considerado como dois nós que possuem a restrição de proibição da ligação entre si.

A partir dos vários testes realizados obtiveram-se os melhores resultados com os seguntes parâmetros:

Tabela 4.2 – Parâmetros utilizados na busca tabu

| **Parâmetros** | |
|---|---|
| **Tamanho da Lista Tabu** | 1,5 vezes o número de Cs |
| **Iterações na Lista Tabu** | 10% do número de diversificações |

Fonte: O autor (2019).

Futuramente é possível que os valores definidos na Tabela 4.4 possam ser definidos automaticamente em função da geometria da camada, tamanho do envelope de fabricação, número de Cs e CRSs, entre outros parâmetros.

# 5 RESULTADOS

Neste capítulo são apresentados e discutidos os resultados obtidos pelos modelos de Programação Linear Inteira Mista - MILP e Busca Tabu – BT em relação às rotas Não Otimizadas - NO e obtidas com os algoritmos Guloso e Inserção do Mais Próximo com 2-opt - NI2OPT (VOLPATO *et al.,* 2019). Foram apresentadas, as distâncias de reposicionamento, os tempos ociosos, os tempos de processamento e ainda possibilidades de melhorar a performance da BT.

## 5.1 DISTÂNCIAS DE REPOSICIONAMENTO

A Tabela 5.1 apresenta a quantidade original dos dados (número de pontos de Contornos - C) para cada camada e a redução obtida em relação a quantidade de dados original após cada simplificação proposta (Análise de ilhas, Pontos espaçados - SP, Pontos Próximo de Pontos de *Raster* – NRP, Centro do Envelope - BBCP). Como esperado, as simplificações BBCP e NRP, respectivamente, apresentaram uma redução substancial de pontos em relação a SP.

Tabela 5.1 – Quantidade de dados antes e depois da realização do pré-processamento (Passo 1)

| **Peça** | **Número de pontos de Contorno** | **Número total de pontos de C após pré-processamento** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Análise de Ilhas** | | **SP** | | **NRP** | | **BBCP** | |
| | | Pontos | Redução | Pontos | Redução | Pontos | Redução | Pontos | Redução |
| 1 | 477 | 309 | 35,22% | 262 | 61,6% | 15 | 96,90% | 4 | 99,16% |
| 2 | 1058 | 788 | 25,52% | 415 | 59,5% | 27 | 97,40% | 13 | 98,77% |
| 3 | 1111 | 796 | 28,35% | 330 | 64,2% | 34 | 96,90% | 17 | 98,47% |

Fonte: O autor (2019).

A Tabela 5.2 apresenta os resultados para as distâncias de reposicionamento com o *framework* proposto utilizando os modelos MILP e BT e considerando as três opções para reduzir o número de pontos de C (SP, NRP e BBCP). Os resultados são comparados às rotas NO e otimizadas com os algoritmos Guloso (*greedy)* e NI2OPT obtidas diretamente do sistema RP3.

Tabela 5.2 – Distâncias de reposicionamento e ganho (redução) obtidas com o *framework* e métodos MILP e BT (considerando as simplificações SP, NRP e BBCP no Passo 2) para NO, Guloso e NI2OPT

| | | | | | **MODELOS MILP** | | | **BUSCA TABU** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | **Simplificações de pontos de contorno propostas** | | | | | | |
| | | | | | | | | **NRP** | | **BBCP** | |
| | **PEÇA** | **NO** | **GULOSO** | **NI2OPT** | **SP** | **NRP** | **BBCP** | **Diversificações** | | **Diversificações** | |
| | | | | | | | | **2500** | **5000** | **2500** | **5000** |
| 1 | Rota-C (mm) | 1742,29 | 995,11 | 1090,21 | 880,80 | 913,26 | 913,66 | 927,54 | 927,51 | 928,19 | 926,45 |
| | Rota-CRS (mm) | 2048,32 | 831,45 | 943,66 | 795,95 | 826,72 | 826,72 | 873,38 | 873,38 | 873,38 | 873,38 |
| | **Total (mm)** | **3790,62** | **1826,57** | **2033,87** | **1676,75** | **1739,99** | **1740,39** | **1800,92** | **1800,89** | **1801,58** | **1799,83** |
| | Tempo ocioso (s) | 186,55 | 89,89 | 100,09 | 82,52 | 85,63 | 85,65 | 88,63 | 88,63 | 88,66 | 88,57 |
| | Ganho sobre NO (%) | - | - | - | 55,8% | 54,1% | 54,1% | 52,5% | 52,5% | 52,5% | 52,5% |
| | Ganho sobre Guloso (%) | - | - | - | 8,2% | 4,7% | 4,7% | 1,4% | 1,4% | 1,4% | 1,5% |
| | Ganho sobre NI2OPT (%) | - | - | - | 17,6% | 14,4% | 14,4% | 11,5% | 11,5% | 11,4% | 11,5% |
| 2 | Rota-C (mm) | 6582,47 | 2988,23 | 3042,15 | NA | 2407,99 | 2452,25 | 2737,72 | 2678,96 | 2717,11 | 2645,13 |
| | Rota-CRS (mm) | 7327,42 | 3049,77 | 2629,67 | NA | 2553,51 | 2534,17 | 3161,26 | 3073,96 | 3135,58 | 3066,37 |
| | **Total (mm)** | **13909,89** | **6038,00** | **5671,82** | **NA** | **4961,50** | **4986,41** | **5898,98** | **5752,92** | **5852,69** | **5711,50** |
| | Tempo ocioso (s) | 684,54 | 297,15 | 279,13 | NA | 244,17 | 245,39 | 290,30 | 283,12 | 288,03 | 281,08 |
| | Ganho sobre NO (%) | - | - | - | NA | 64,3% | 64,2% | 57,6% | 58,6% | 57,9% | 58,9% |
| | Ganho sobre Guloso (%) | - | - | - | NA | 17,8% | 17,4% | 2,0% | 4,7% | 3,1% | 5,4% |
| | Ganho sobre NI2OPT (%) | - | - | - | NA | 12,5% | 12,1% | -4,0% | -1,4% | -3,2% | -0,7% |
| 3 | Rota-C (mm) | 3552,68 | 1762,69 | 1758,65 | NA | 1506,61 | 1515,51 | 1644,94 | 1645,37 | 1644,86 | 1595,82 |
| | Rota-CRS (mm) | 6330,55 | 1658,73 | 1662,20 | NA | 1570,39 | 1552,75 | 1813,51 | 1794,18 | 1809,37 | 1781,50 |
| | **Total (mm)** | **9883,23** | **3421,42** | **3420,85** | **NA** | **3077,00** | **3068,27** | **3458,46** | **3439,55** | **3454,22** | **3377,32** |
| | Tempo ocioso (s) | 486,38 | 168,38 | 168,35 | NA | 151,43 | 151,00 | 170,20 | 169,27 | 169,99 | 166,21 |
| | Ganho sobre NO (%) | - | - | - | NA | 68,9% | 69,0% | 65,0% | 65,2% | 65,0% | 65,8% |
| | Ganho sobre Guloso (%) | - | - | - | NA | 10,1% | 10,3% | -1,1% | -0,5% | -1,0% | 1,3% |
| | Ganho sobre NI2OPT (%) | - | - | - | NA | 10,1% | 10,3% | -1,1% | -0,5% | -1,0% | 1,3% |

Fonte: O autor (2019).

Os resultados NO (não otimizados) significam que a rota é gerada seguindo a lógica interna do sistema RP3, sem a execução de algum algoritmo de otimização (essa lógica pode variar para cada sistema de planejamento de processos). As rotas NI2OPT são obtidas a partir de uma otimização dupla (em duas etapas), isto é, primeiro é gerada uma rota utilizando inserção do mais próximo (NI) e posteriormente esta rota é refinada utilizando o método 2-opt (VOLPATO *et al.,* 2019).

Para facilitar a análise, esses resultados são apresentados graficamente nas Figuras 5.1, 5.2 e 5.3. Conforme detalhado na Seção 3.5, os resultados obtidos para as implementações de BT representam a média de cinco execuções do algoritmo, cujo desvio padrão médio é apresentado na Tabela 5.3.

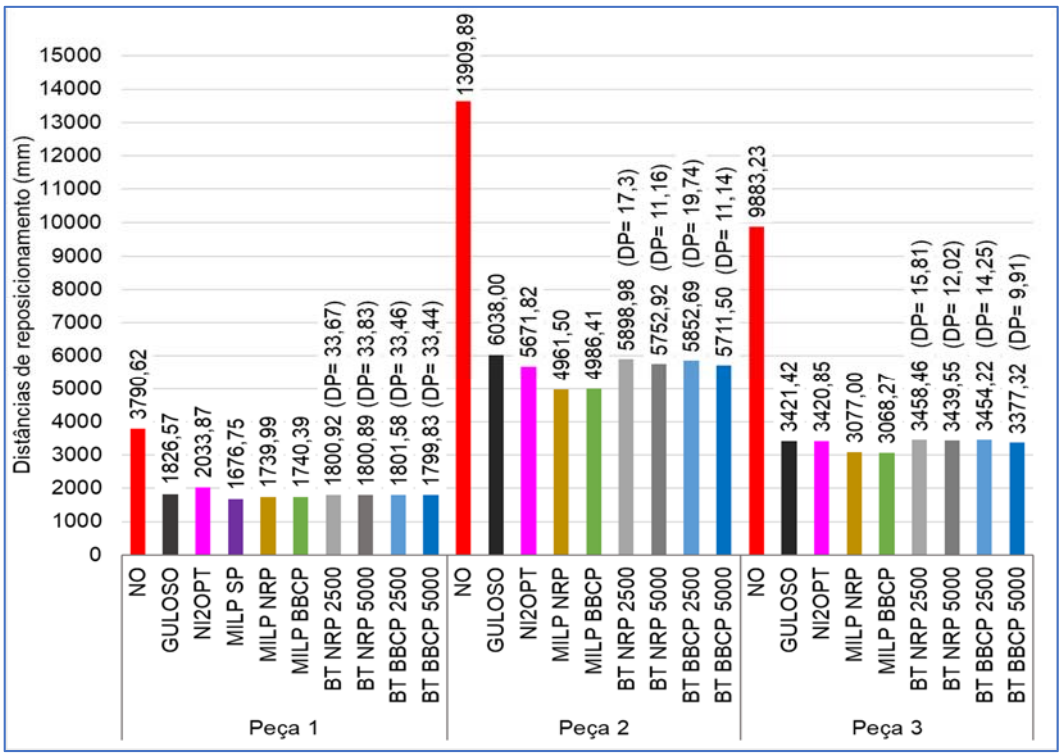
Tabela 5.3 – Média e desvio padrão (DP) das distâncias de reposicionamento obtidos para as execuções utilizando BT com o número de diversificações ajustado para 2500 e 5000

| Peça | Modelo | Média | DP Médio |
|---|---|---|---|
| **1** | BT NRP 2500 | 1800,92 | 33,46 |
| | BT NRP 5000 | 1800,89 | 33,45 |
| | BT BBCP 2500 | 1801,58 | 33,68 |
| | BT BBCP 5000 | 1799,83 | 33,84 |
| **2** | BT NRP 2500 | 5898,98 | 17,31 |
| | BT NRP 5000 | 5752,92 | 11,16 |
| | BT BBCP 2500 | 5852,69 | 19,74 |
| | BT BBCP 5000 | 5711,50 | 11,15 |
| **3** | BT NRP 2500 | 3458,46 | 15,82 |
| | BT NRP 5000 | 3439,55 | 12,02 |
| | BT BBCP 2500 | 3454,22 | 14,26 |
| | BT BBCP 5000 | 3377,32 | 9,92 |

Fonte: O autor (2019).

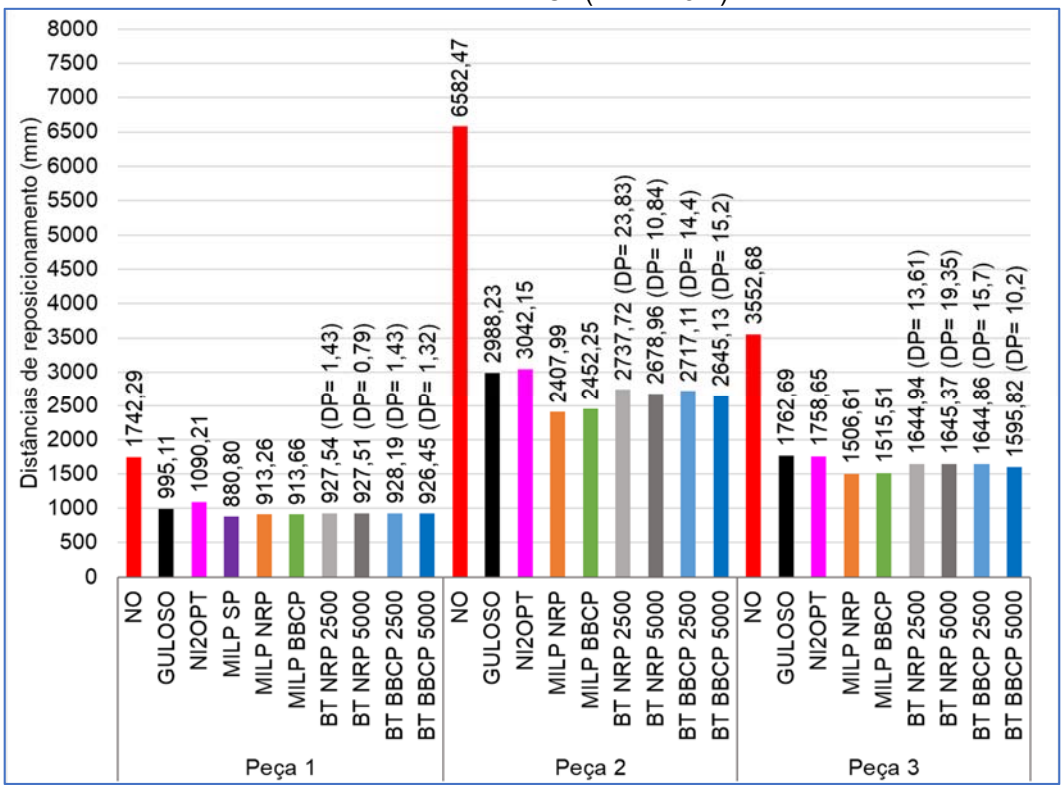
A redução no número de pontos de C alcançados pela opção SP para o caso das peças 2 e 3, que têm geometrias mais complexas, não possibilitou os modelos MILP obterem resultados em 7200s (critério de parada). Com a simplificação mais eficaz dos pontos de C obtida pelas opções NRP e BBCP, os modelos MILP foram bem-sucedidos. Para efeitos de comparação, os modelos utilizando BT também foram executados sem considerar a opção de simplificação por SP.

Figura 5.1 – Representação gráfica dos totais das distâncias de reposicionamento (Tabela 5.2)



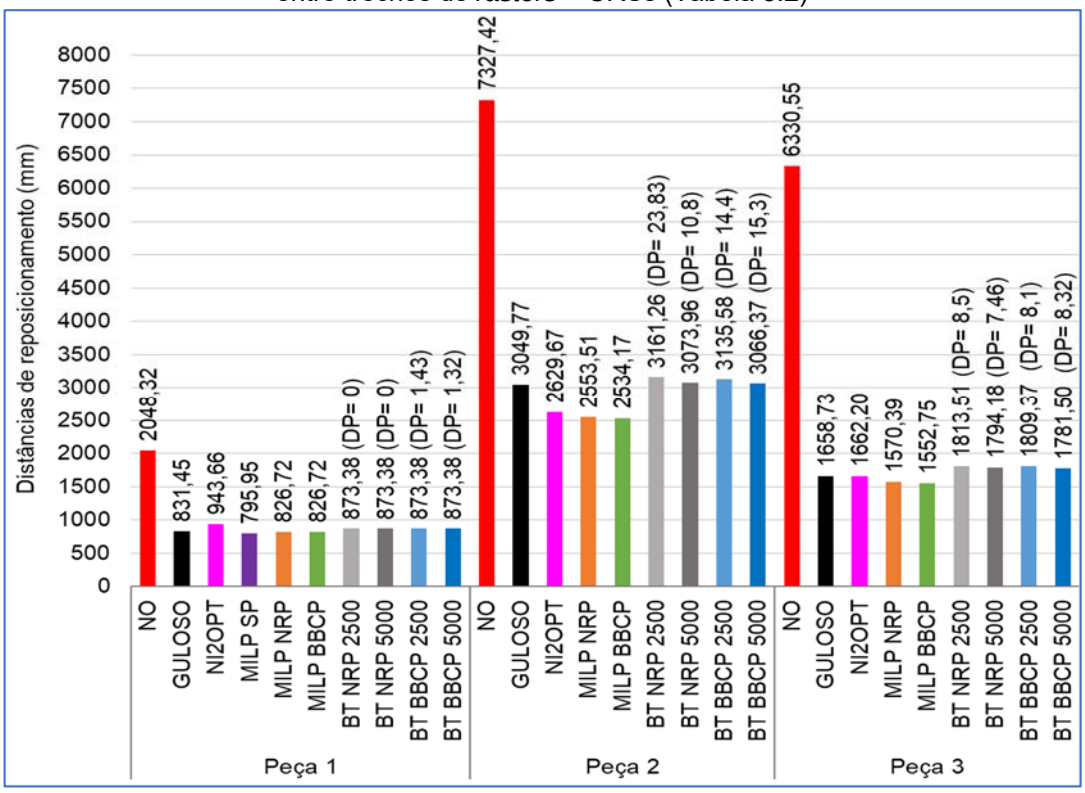

Fonte: O autor (2019).

Figura 5.2 – Representação gráfica dos totais das distâncias de reposicionamento obtidas apenas entre contornos - Cs (Tabela 5.2)




Fonte: O autor (2019).

Figura 5.3 – Representação gráfica dos totais das distâncias de reposicionamento obtidas apenas entre trechos de *rasters* – CRSs (Tabela 5.2)




Fonte: O autor (2019).

Comparando os resultados das distâncias de reposicionamentos para a peça 1, observa-se uma diferença mínima entre todos os métodos, mas o melhor resultado foi obtido pelo MILP SP. Os métodos MILP NRP e BBCP obtiveram resultados iguais entre si, da mesma forma como ocorreu com a BT NRP e BBCP. Para as peças 2 e 3 utilizando MILP NRP e BBCP, observou-se uma pequena diferença. Para a peça 2, o NRP obteve um ganho de 0,4% sobre o BBCP e, para a peça 3, o BBCP obteve um ganho de 0,2% sobre o NRP. No caso da BT, a opção BBCP sempre teve resultados um pouco melhores sobre a opção NRP, chegando a 0,7% e 1,8% para as peças 2 e 3, respectivamente.

De maneira geral, os resultados do *framework* proposto utilizando os modelos MILP com as simplificações para reduzir o número de pontos de C (SP, NRP e BBCP) superaram os resultados das rotas obtidas utilizando NO, Guloso, NI2OPT e BT em todas as peças, mostrando um comportamento superior constante (Tabela 5.2), mas apresentaram um considerável custo computacional, principalmente nas peças 2 e 3.

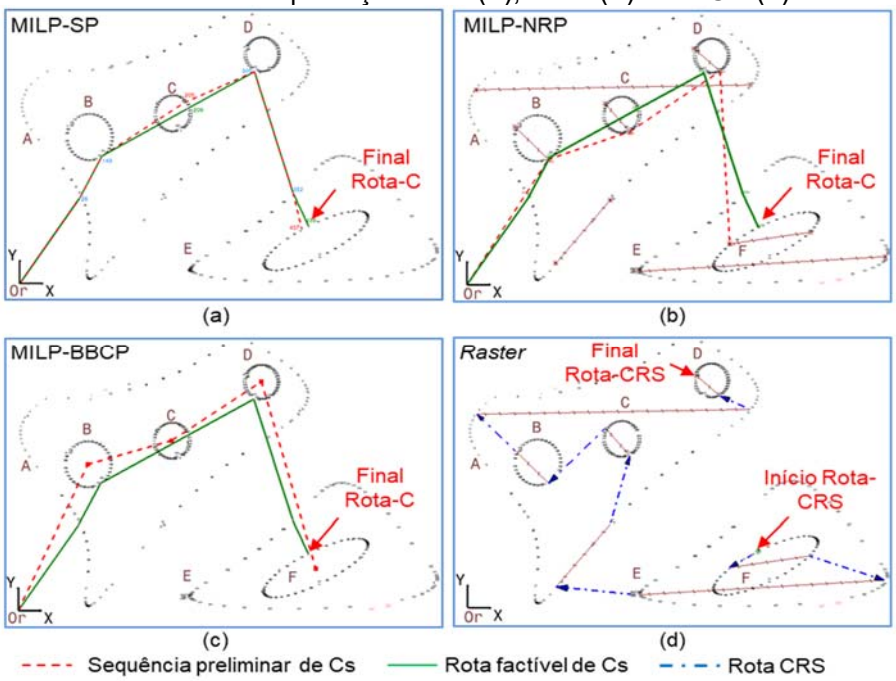
Para os três casos estudados, as três abordagens de simplificação são comparáveis entre si. Portanto, é possível supor que o NRP e o BBCP permitiram

solucionar o problema sem comprometer os resultados das distâncias de reposicionamento, embora tenham gerado rotas diferentes, conforme descrito à frente. Considerando que a diferença entre NRP e BBCP foi pequena, há uma indicação que utilizar a simplificação BBCP seja mais vantajoso em função do menor número de pontos utilizados no cálculo da sequência preliminar e da independência em relação aos parâmetros de processo. Cabe destacar que o método de simplificação NRP pode sofrer interferência de parâmetros de processo relacionados ao preenchimento tipo *raster* (e.g. ângulo e espaçamento de *raster*), pois esses determinarão onde serão criados os pontos de *raster*. Sendo assim, os parâmetros de processo ajustados pelo usuário poderão ter influência direta nos resultados obtidos pelo MILP ou BT utilizando a simplificação NRP. Este efeito não é observado para a simplificação BBCP, pois ela depende somente das coordenadas do ponto central do envelope de cada ilha, independentemente do tipo de preenchimento que será adotado.

## 5.2 COMPARAÇÃO DAS ROTAS OBTIDAS

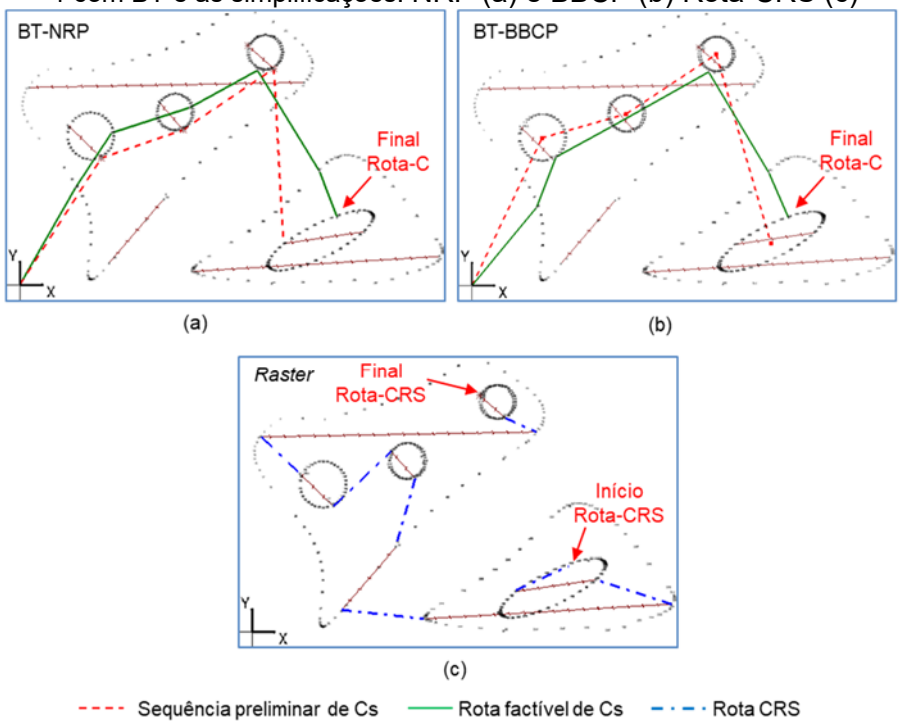
Para a peça 1 (geometria mais simples), os ganhos em reposicionamento do MILP (SP, NRP e BBCP) sobre os algoritmos NO, Guloso e NI2OPT foram de 55,8%, 8,2% e 17,6%, respectivamente. As execuções de BT para NRP e BBCP (2500 e 5000 diversificações) obtiveram resultados semelhantes entre si, apresentando um ganho de 52,5%, 1,5% e 11,5% sobre NO, Guloso e NI2OPT, respectivamente. Como pode ocorrer com o Guloso, neste caso em particular, ele apresentou um bom resultado e, por isso, o percentual do ganho foi de apenas 1,5%, o que ocorreu também com o MILP, que teve somente 8,2% de ganho. Para o MILP, as Figuras 5.4a, 5.4b e 5.4c apresentam a sequência preliminar de Cs (Passo 2) e a rota factível de Cs (Passo 3) obtidas para a peça 1 ao aplicar SP, NRP e BBCP, respectivamente. Para esta peça, SP, NRP e BBCP retornaram os mesmos resultados para a rota de Cs. A Figura 5.4d apresenta a rota CRS (Passo 4) obtida para a peça 1, que foi exatamente a mesma para todos os métodos de simplificação de pontos. Para a BT, as Figuras 5.5a e 5.5b apresentam sequência preliminar de Cs (Passo 2) e a rota factível de Cs (Passo 3) obtidas na peça 1 aplicando NRP e BBCP, respectivamente. Neste caso, de maneira semelhante aos resultados obtidos nos modelos MILP, as rotas foram praticamente idênticas, devido à geometria simples da peça.

Figura 5.4 – Sequência preliminar de Cs e rota factível de Cs obtidas para a primeira camada da peça 1 com os modelos MILP e as simplificações: SP (a); NRP (b) e BBCP (c) e a Rota-CRS (d)




Fonte: O autor (2019).

Figura 5.5 – Sequência preliminar de Cs e rota factível de Cs obtidas para a primeira camada da peça 1 com BT e as simplificações: NRP (a) e BBCP (b) Rota-CRS (c)

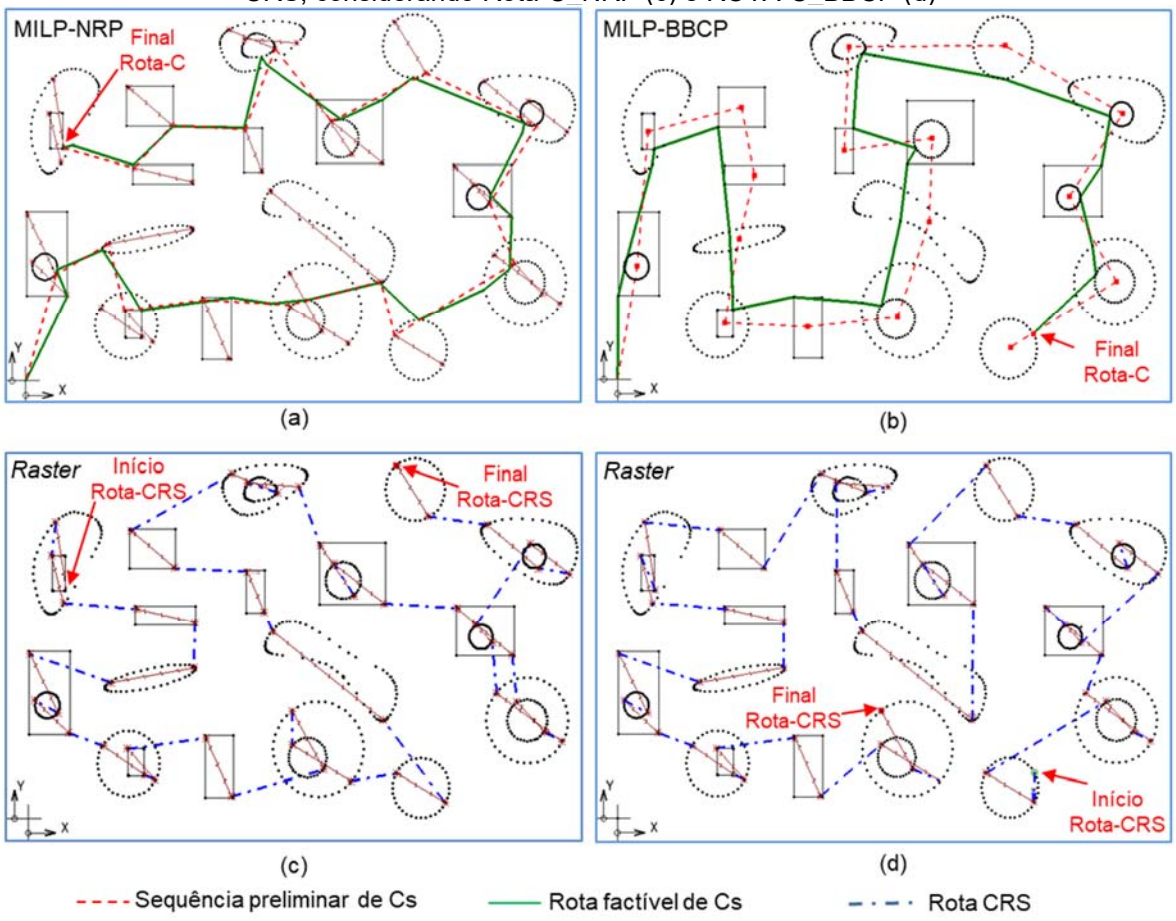



Fonte: O autor (2019).

Para o caso da peça 2, as rotas obtidas nos modelos MILP são apresentadas nas Figuras 5.6a-b e 5.6c-d, mostrando a sequência preliminar de Cs, a rota factível

de Cs e a rota de CRSs, considerando os métodos de simplificação NRP e BBCP, respectivamente.

Figura 5.6 – Sequência preliminar de Cs e rota factível de Cs obtidas para a primeira camada da peça 2 obtidas pelos modelos MILP com as simplificações: NRP (a) e BBCP (b), e as respectivas Rotas-CRS, considerando Rota-C_NRP (c) e ROTA-C_BBCP (d)

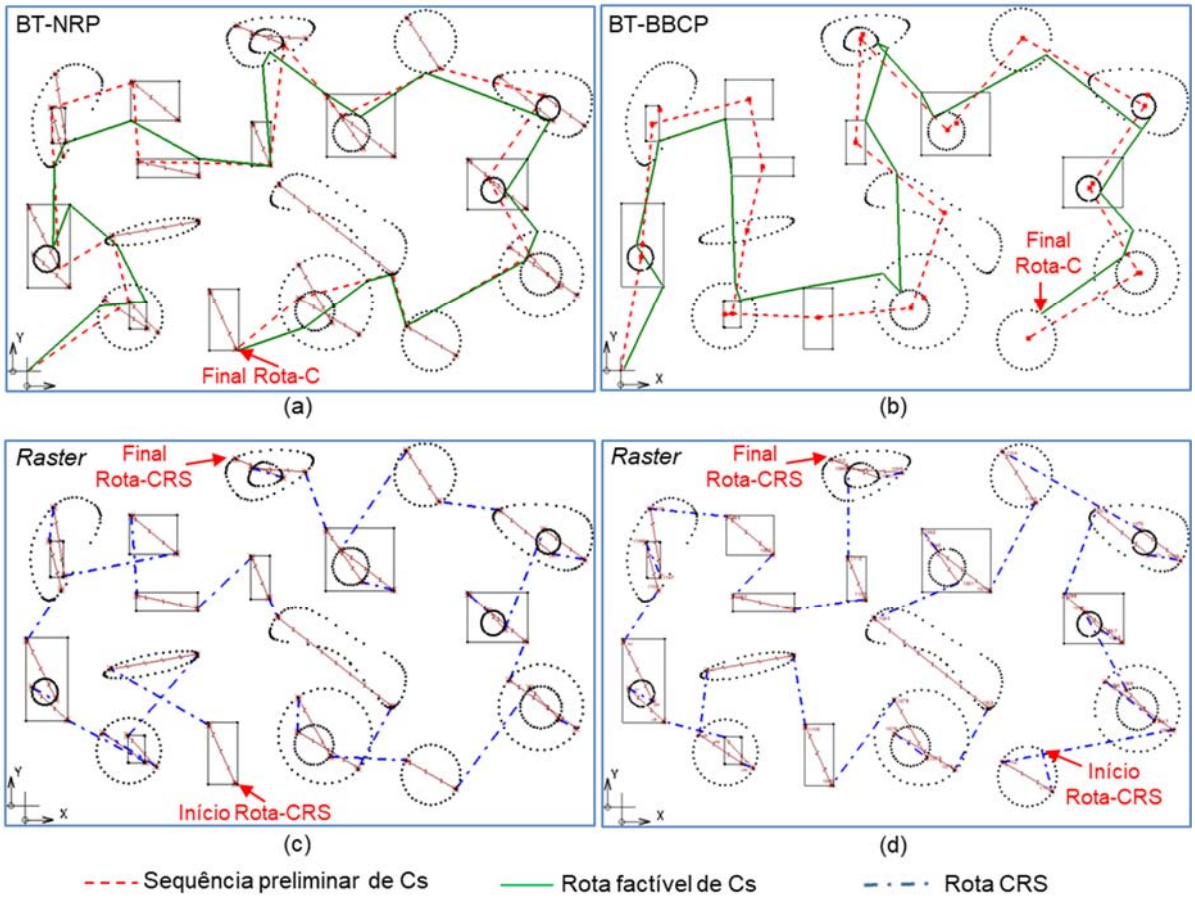



Fonte: O autor (2019).

Analisando as rotas MILP NRP e BBCP ilustradas na Figura 5.6 verifica-se que, embora os valores das distâncias obtidas tenham sido muito próximos (4961,50mm e 4986,41mm, respectivamente) as rotas geradas foram diferentes para Cs e consequentemente para CRSs. Portanto, mesmo apresentando rotas diferentes, o que é esperado devido às diferenças na geração da sequência preliminar, o resultado da distância de reposicionamento foi similar.

As melhores rotas obtidas na peça 2 através dos algoritmos de BT são apresentadas nas Figuras 5.7a-b e 5.7c-d, mostrando a sequência preliminar de Cs, a rota factível de Cs e a rota de CRSs, considerando os métodos de simplificação NRP e BBCP, respectivamente.

Figura 5.7 – Sequência preliminar de Cs e rota factível de Cs obtidas para a primeira camada da peça 2 obtidas pela BT com as simplificações: NRP (a) E BBCP (b), e as respectivas Rotas-CRS, considerando Rota-C_NRP (c) e Rota-C_BBCP (d).



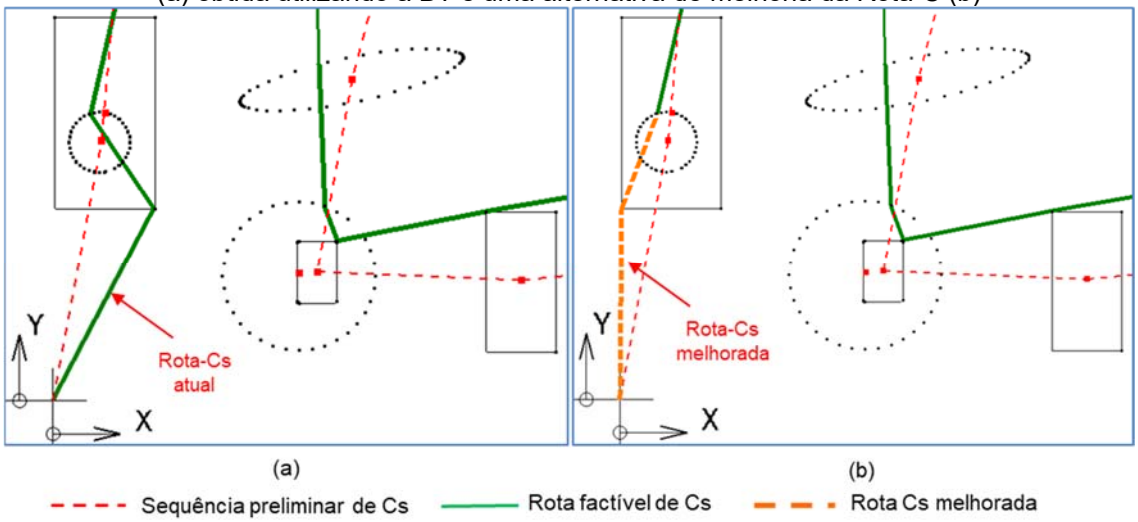

Fonte: O autor (2019).

Para a peça 2, os resultados da BT utilizando NRP (2500 e 5000 diversificações), indicaram ganho de 57,6% e 58,6% sobre NO e 2% e 4,7% sobre o Guloso, respectivamente. Em comparação com o algoritmo NI2OPT houve uma perda de 4% e 1,4%, respectivamente. A BT utilizando BBCP (2500 e 5000 diversificações) obteve ganhos de 57,9% e 58,9% sobre a rota NO, 3,1% e 5,4% sobre o algoritmo Guloso. Já a comparação com o algoritmo NI2OPT revelou uma perda de 3,2% e 0,7%, respectivamente. O desempenho melhor sobre o Guloso, da mesma forma que aconteceu com os modelos MILP pode ser explicado pela geometria da camada, conforme explicado anteriormente. Já a perda da BT em relação ao NI2OPT é explicada pelo seu mau desempenho nos trechos de *raster,* conforme explicado mais à frente. Assim como aconteceu com MILP, os resultados da BT utilizando as simplificações NRP e BBCP foram semelhantes.

Comparando as rotas obtidas por MILP e BT, observa-se que as rotas utilizando simplificação NRP foram consideravelmente diferentes. Como pode acontecer, o melhor resultado da BT obtido pela simplificação BBCP e o resultado MILP foram semelhantes, ou seja, a sequência preliminar obtida no Passo 2 foi a mesma, mas a rota factível foi levemente diferente. Isto pode indicar que a BT pode obter resultados próximos aos dos modelos MILP, necessitando de mais estudos quanto ao refinamento de seus parâmetros. Para as rotas CRS, observa-se que todas foram consideravelmente diferentes. Estas rotas diferentes obtidas entre MILP e BT são esperadas, pois no caso da BT a solução inicial aleatória e as diversificações provavelmente levarão a soluções distintas.

Embora a sequência de Cs e CRSs a serem depositados tenha sido boa, ou seja, o critério de vizinhança entre as ilhas foi respeitado, o desempenho inferior da BT para algumas comparações nas peças 2 e 3 se deve a escolhas ruins dos pontos de C que compõem a rota factível (Passo 3) e principalmente ao excesso de cruzamentos na rota entre CRSs (Passo 4). A Figura 5.8a apresenta parte de uma rota obtida pela BT para Cs na peça 2 através da simplificação BBCP que apresentou este problema e, em função da observação deste resultado, pode-se pensar em uma sugestão de melhoria de rota apresentada na Figura 5.8b.

Figura 5.8 – Rota factível de Cs obtida para a primeira camada da peça 2 com a simplificação BBCP (a) obtida utilizando a BT e uma alternativa de melhoria da Rota-C (b)

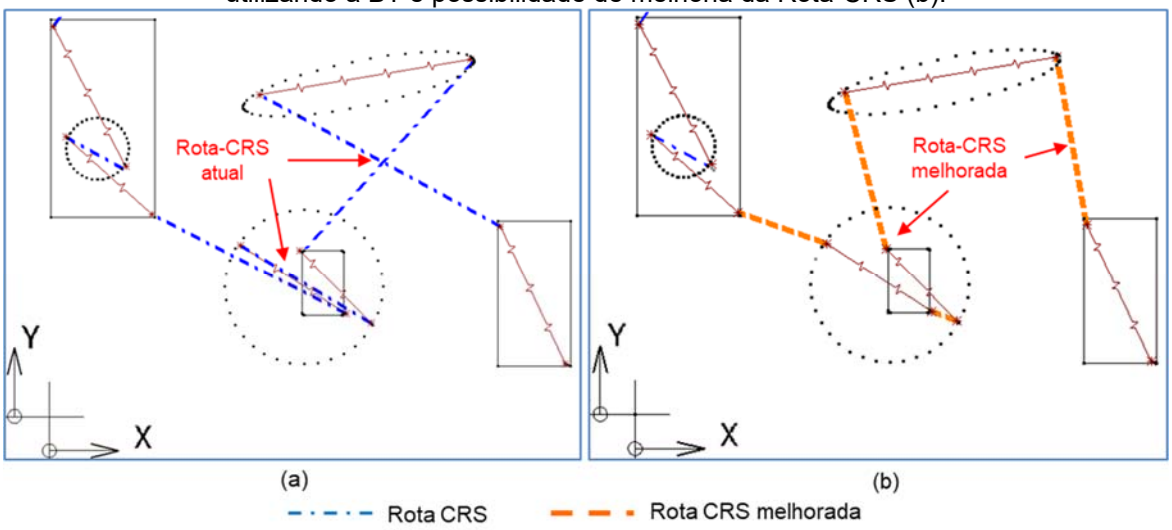



Fonte: O autor (2019).

Nas rotas de CRSs fica mais evidente a causa da BT ter obtido piores resultados em comparação aos demais métodos. Nesse caso, a melhoria pode ser realizada eliminando o excesso de cruzamentos dos reposicionamentos entre os

CRSs, realizando uma segunda etapa de otimização com métodos clássicos de busca local, por exemplo, através do uso de 2-opt (da mesma forma como ocorre no algoritmo NI2OPT). A Figura 5.9a mostra a rota-CRS com vários cruzamentos, gerando uma rota maior. Já a Figura 5.9b sugere uma rota-CRS melhorada, evitando os cruzamentos.

Figura 5.9 –Rota-CRS obtida para a primeira camada da peça 2 com a simplificação NRP (a) utilizando a BT e possibilidade de melhoria da Rota-CRS (b).

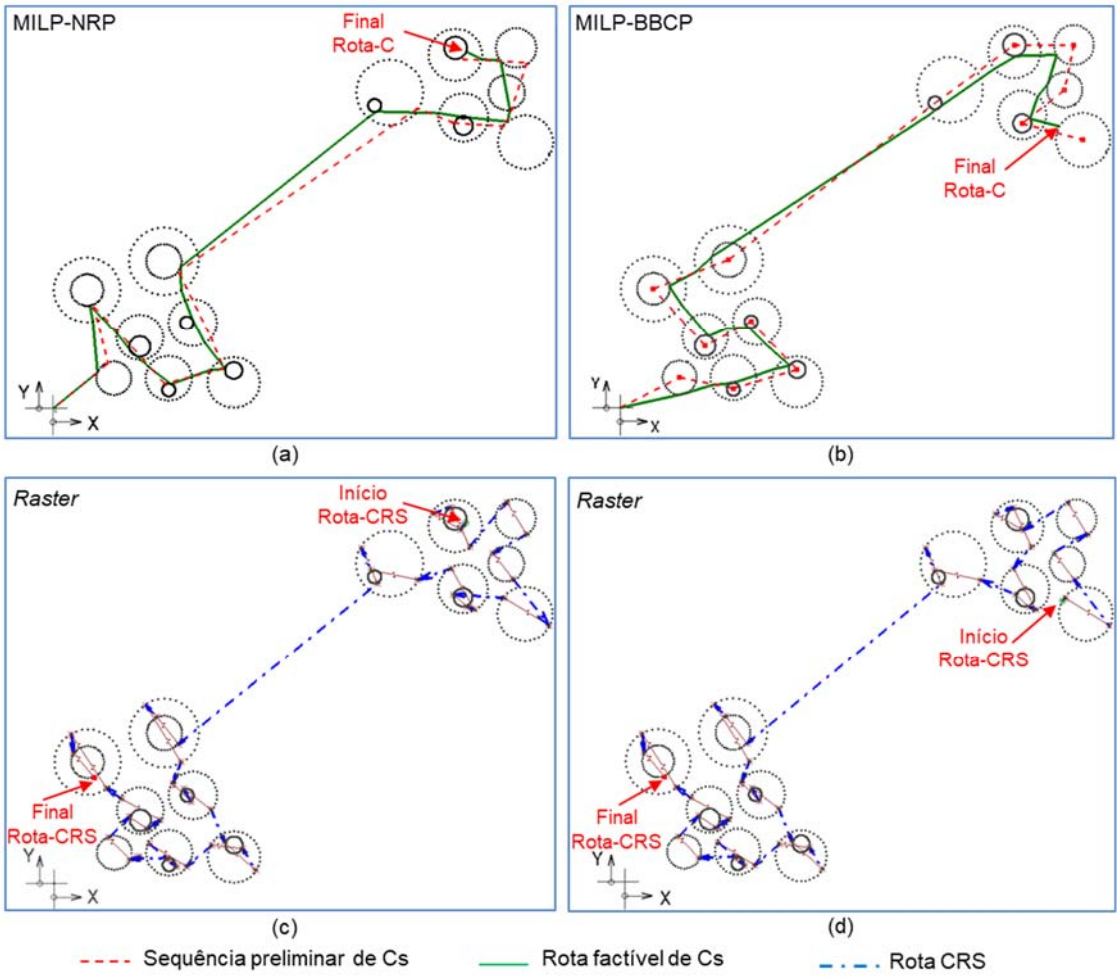



Fonte: O autor (2019).

Para a peça 3, as rotas obtidas pelos modelos MILP são apresentadas nas Figuras 5.10a-b e 5.10c-d, apresentando a sequência preliminar de Cs, a rota factível de Cs e a rota de CRSs, considerando os métodos de simplificação NRP e BBCP, respectivamente. As rotas MILP, NRP e BBCP obtiveram ganhos sobre NO de 68,9% e 69%, respectivamente. Sobre os algoritmos Guloso e NI2OPT os ganhos foram semelhantes, de 10,1% e 10,3%, respectivamente. Este desempenho semelhante entre Guloso e NI2OPT e o respectivo baixo ganho do MILP sobre estes dois métodos pode ser explicado pela geometria da camada, pois, embora existam várias ilhas e um grande número de pontos (mais que na peça 2), as ilhas estão divididas em dois *clusters,* permitindo assim realizar a otimização local do primeiro *cluster* de maneira eficiente e posteriormente passar para o segundo *cluster.*

As melhores rotas obtidas na peça 3 através da BT são ilustradas nas Figuras 5.11a-b e 5.11c-d, apresentando a sequência preliminar de Cs, a rota factível de Cs e a rota de CRSs, considerando os métodos de simplificação NRP e BBCP, respectivamente.

Figura 5.10 – Sequência preliminar de Cs e rota factível de Cs obtidas para a primeira camada da peça 3 obtidas pelos modelos MILP com as simplificações: NRP (a) e BBCP (b), e as respectivas Rotas-CRS, considerando Rota-C_NRP (c) e Rota-C_BBCP (d)

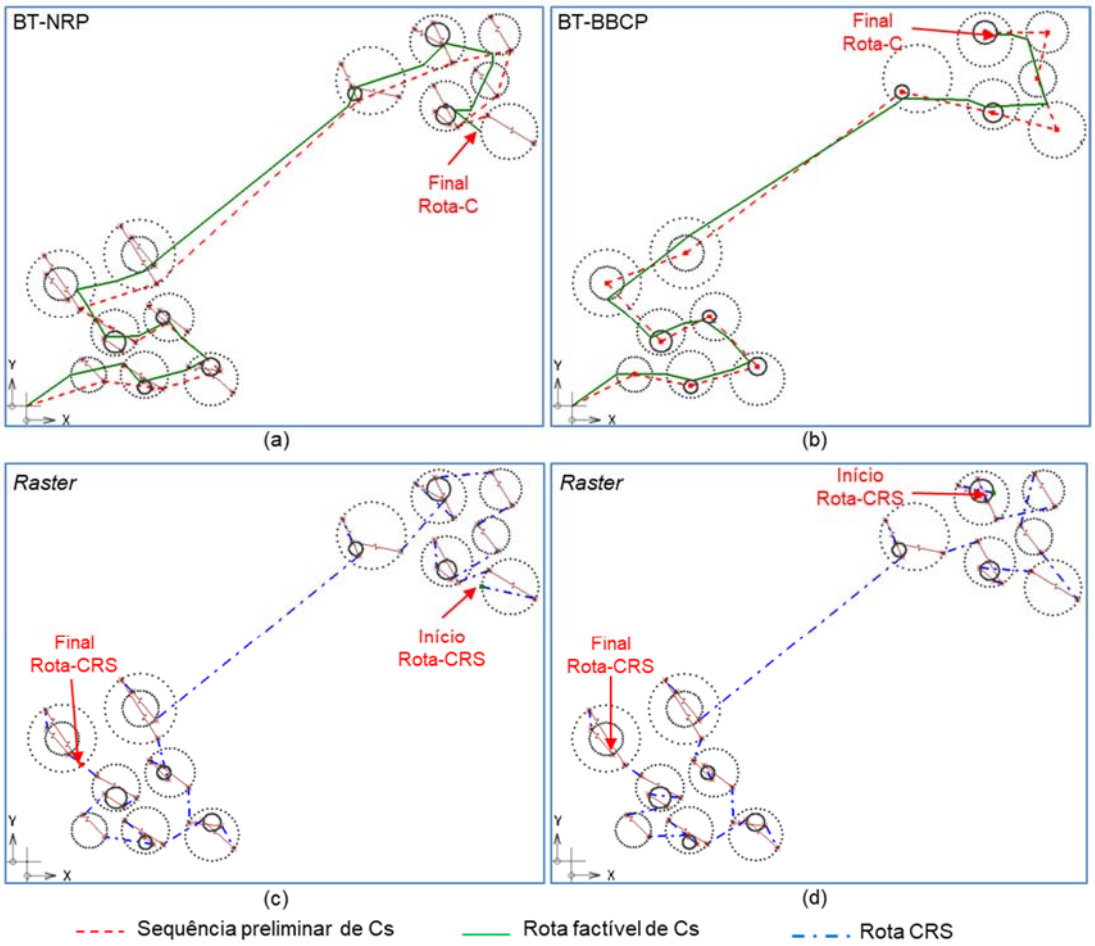



Fonte: O autor (2019).

Para BT NRP (2500 e 5000 diversificações) os resultados indicaram ganho de 65% e 65,2% sobre NO. Em comparação com os algoritmos Guloso e NI2OPT houve perdas semelhantes de 1,1% e 0,5%, respectivamente. Para a BT BBCP (2500 e 5000 diversificações) os ganhos sobre a rota NO foram semelhantes de 65% e 65,8%, respectivamente. A BT BBCP (2500 diversificações), obteve perdas semelhantes de 1% em relação aos algoritmos Guloso e NI2OPT. Já a opção de BT BBCP com 5000 diversificações obteve ganhos semelhantes de 1,3% sobre o Guloso e NI2OPT.

Este desempenho aquém ao esperado da BT para a peça 3 aconteceu também com os modelos MILP, devido principalmente à geometria da camada, conforme já citado acima. Assim como ocorreu com os modelos MILP, os resultados entre NRP e BBCP foram equivalentes, mas geraram rotas parcialmente diferentes.

Figura 5.11 – Sequência preliminar de Cs e rota factível de Cs obtidas para a primeira camada da peça 3 obtidas com BT e as simplificações: NRP (a) e BBCP (b), e as respectivas Rotas-CRS, considerando Rota-C_NRP (c) e Rota-C_BBCP (d)




Fonte: O autor (2019).

Isto pode ser explicado pela geometria da camada, separada em dois *clusters.* Outro aspecto que também contribuiu negativamente na rota da BT foi a má escolha de pontos que conectam as ilhas na rota de Cs e pelo excesso de cruzamentos na rota CRS, embora menos evidente em comparação à peça 2.

Conforme descrito no *framework*, o Passo 3 fez uso de uma nova aplicação dos Polígonos de Visibilidade – VP, como forma de reduzir mais o espaço de busca durante a geração da rota factível de Cs. A formação dos VPs faz uso da sequência preliminar de Cs (Passo2), selecionando apenas os pontos pertencentes ao conjunto SP que estejam contidos nos VPs. Desta forma, o espaço de busca é reduzido consideravelmente. Como os VPs dependem apenas da rota preliminar (que pode variar entre as várias camadas, simplificações e modelos implementados), a Tabela 5.4 apresenta a média da redução de pontos obtidas considerando as simplificações NRP e BBCP.

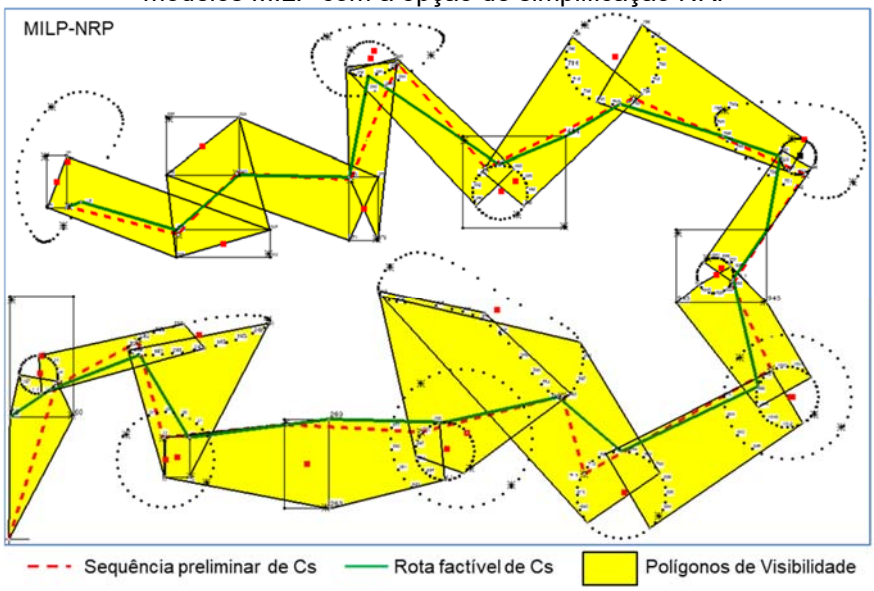
Tabela 5.4 – Média da quantidade de pontos selecionados após a simplificação dos polígonos de visibilidade

| Peça | Pontos de contorno obtidos na simplificação SP | Média do número de pontos de contorno após simplificação dos Polígonos de Visibilidade | | | |
|---|---|---|---|---|---|
| | | NRP | | BBCP | |
| | Pontos | Pontos | Redução | Pontos | Redução |
| **1** | 262 | 64 | 78,0% | 64 | 78,1% |
| **2** | 415 | 140 | 66,2% | 135 | 67,4% |
| **3** | 330 | 126 | 61,8% | 120 | 63,6% |

Fonte: O autor (2019).

Analisando os resultados da Tabela 5.4, observa-se que o potencial de redução de pontos obtidos através dos VPs é substancial e foi equivalente entre as simplificações NRP e BBCP. A Figura 5.12 ilustra os VPs formados na primeira camada da peça 2 para a rota preliminar obtida pelo modelo MILP utilizando a simplificação NRP.

Figura 5.12 – Representação gráfica dos polígonos de visibilidade criados para a peça 2 utilizando os modelos MILP com a opção de simplificação NRP




Fonte: O autor (2019).

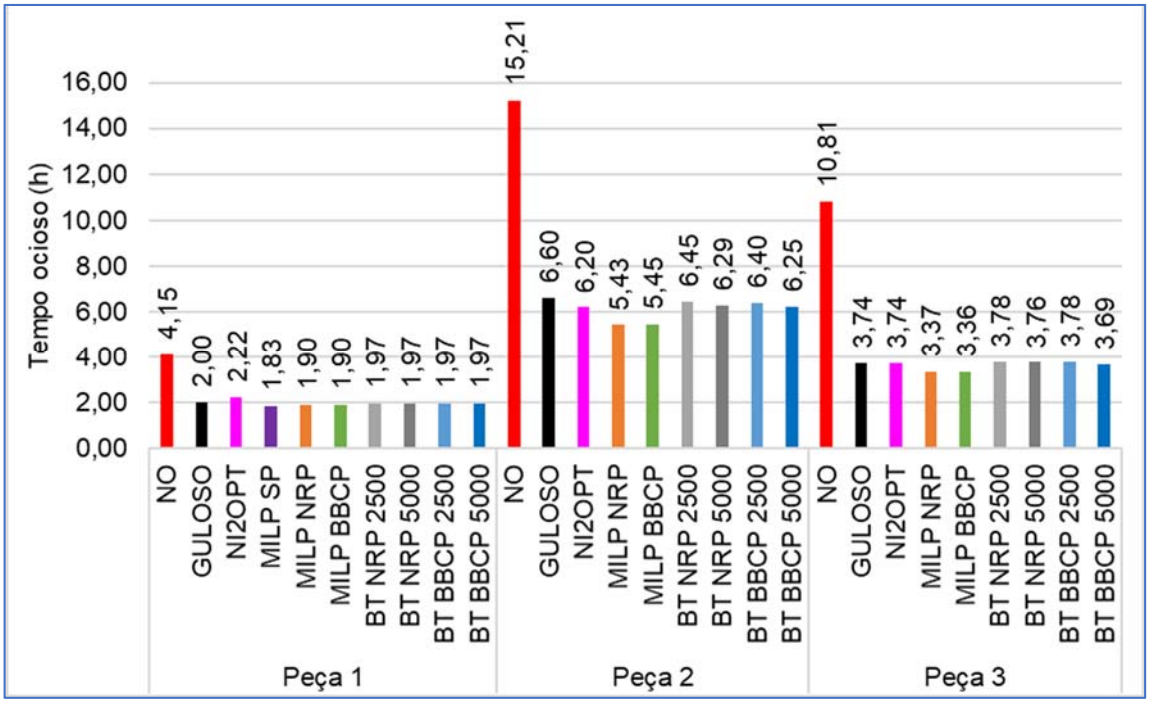
Com relação às rotas obtidas, é esperado que utilizando BT elas sejam consideravelmente diferentes entre si, em consequência da geração de uma solução inicial aleatória e também das diversificações realizadas. Também é esperado que um número maior de diversificações apresente resultados melhores na função objetivo e menor desvio padrão, conforme apresentado na Tabela 5.3. Por outro lado, o tempo

de execução da BT com maior número de diversificações é sensivelmente maior, conforme apresentado mais à frente.

## 5.3 ANÁLISE DO TEMPO OCIOSO

Baseado nas distâncias de reposicionamento apresentadas anteriormente na Tabela 5.2, o tempo ocioso existente durante a construção das peças pode ser estimado. A Figura 5.13 apresenta uma projeção desses tempos (em horas) para a construção das peças estudadas com 400 camadas (100mm de altura), considerando a deposição de apenas um perímetro de C (ver Seção 2.4), deposição interna do tipo *raster* com 100% de preenchimento e espessura de camada de 0,25mm. O melhor resultado obtido utilizando as simplificações NRP ou BBCP através dos modelos MILP ou BT foi usado na comparação de ganho dos tempos ociosos. Para a peça 1 (menor complexidade) considerando os resultados NO, Guloso, NI2OPT e o modelo MILP-SP os tempos de ociosidade foram de 4,15h, 2,0h, 2,22h e 1,83h, respectivamente. Isto representa que o modelo MILP-SP obteve uma redução na ociosidade de 55,9% (2,32h), 8,5% (0,17h) e 17,5% (0,39h) em relação as rotas NO, Guloso e NI2OPT, respectivamente.

Figura 5.13 – Projeção dos totais dos tempos ociosos (em horas) para construção das peças com 400 camadas




Fonte: O autor (2019).

Para a peça 2 considerando os resultados NO, Guloso, NI2OPT e o modelo MILP-NRP os tempos de ociosidade foram de 15,21h, 6,60h, 6,20h e 5,43h, respectivamente. Desta forma, estima-se que o tempo de ciclo para construção da peça 2 utilizando MILP-NRP foi reduzido em 64,29% (9,78h), 18,83% (1,17h) e 14,18% (0,77h) em relação as rotas NO, Guloso e NI2OPT, respectivamente. Já na peça 3, o uso do modelo MILP-BBCP possibilitou uma redução de 68,82% (7,45h) em relação a NO e 10,16% (0,38h) considerando o Guloso e NI2OPT.

## 5.4 ANÁLISE DO TEMPO DE PROCESSAMENTO COMPUTACIONAL

De maneira detalhada, as Tabelas 5.6, 5.7 e 5.8 apresentam o tempo de processamento computacional para cada passo do *framework* (2, 3 e 4) considerando os modelos MILP, BT (2500) e BT (5000). Os tempos computacionais para os algoritmos Guloso e NI2OPT não foram coletados devido a impossibilidade de obter estes dados de forma isolada no *software* RP3. No entanto, pode-se fazer uma aproximação dos tempos de processamento computacional (em segundos) apresentados no trabalho de Volpato *et al*. (2019) para cinco camadas, conforme apresentado na Tabela 5.5.

Tabela 5.5 – Tempo de processamento computacional (em segundos) estimado para cinco camadas para os algoritmos Guloso e NI2OPT

| Peça | Guloso | NI2OPT |
|---|---|---|
| **1** | 0,015 | 0,013 |
| **2** | 0,46 | 1,29 |
| **3** | 0,049 | 0,89 |

Fonte: Adaptado de Volpato *et al.* (2019).

Tabela 5.6 – Tempo de processamento computacional nos modelos MILP (em segundos) para os passos 2-3 (C) e 4 (CRS) de cada método de simplificação para cinco camadas

| | **Modelos MILP** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **SP** | | | | **NRP** | | | | **BBCP** | | | |
| **Peça** | **Passo 2** | **Passo 3** | **Passo 4** | **Total (s)** | **Passo 2** | **Passo 3** | **Passo 4** | **Total (s)** | **Passo 2** | **Passo 3** | **Passo 4** | **Total (s)** |
| **1** | 1001,4 | 515,6 | 0,3 | **1517,3** | 0,6 | 0,8 | 0,7 | **2,0** | 0,6 | 0,7 | 0,7 | **1,9** |
| **2** | NA | NA | NA | NA | 1504,5 | 23,2 | 2352,1 | **3879,7** | 3273,4 | 17,3 | 2402,5 | **5693,1** |
| **3** | NA | NA | NA | NA | 112,7 | 21,1 | 721,6 | **855,3** | 131,6 | 23,7 | 778,5 | **933,8** |

Fonte: O autor (2019).

Tabela 5.7 – Tempo de processamento computacional (em segundos) nos algoritmos de BT realizando 2500 diversificações para os passos 2-3 (C) e 4 (CRS) para cada método de simplificação para cinco camadas

| | Busca Tabu | | | | Diversificações: 2500 | | | |
|---|---|---|---|---|---|---|---|---|
| | NRP | | | | BBCP | | | |
| Peça | Passo 2 | Passo 3 | Passo 4 | Total (s) | Passo 2 | Passo 3 | Passo 4 | Total (s) |
| **1** | 20,0 | 2,0 | 1,0 | **23,0** | 20,0 | 5,0 | 1,0 | **26,0** |
| **2** | 60,0 | 10,0 | 36,5 | **106,5** | 45,0 | 10,5 | 37,5 | **93,0** |
| **3** | 40,0 | 10,0 | 30,0 | **80,0** | 35,0 | 10,0 | 31,0 | **76,0** |

Fonte: O autor (2019).

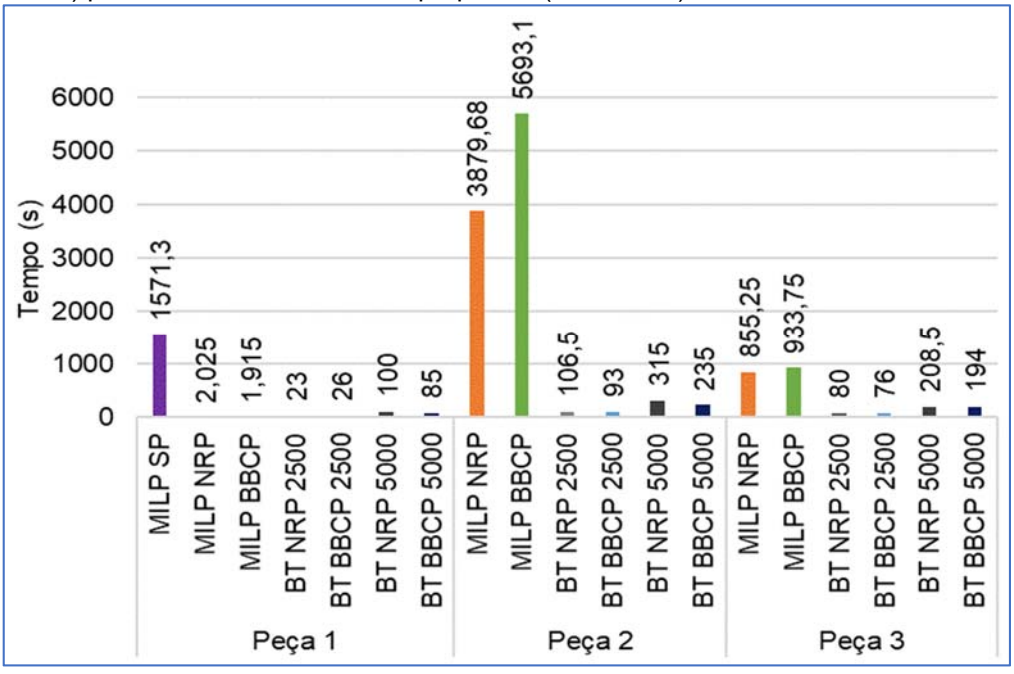
Tabela 5.8 – Tempo de processamento computacional (em segundos) nos algoritmos de BT realizando 5000 diversificações para os passos 2-3 (C) e 4 (CRS) para cada método de simplificação para cinco camadas

| | Busca Tabu | | | | Diversificações: 5000 | | | |
|---|---|---|---|---|---|---|---|---|
| | NRP | | | | BBCP | | | |
| Peça | Passo 2 | Passo 3 | Passo 4 | Total (s) | Passo 2 | Passo 3 | Passo 4 | Total (s) |
| **1** | 90,0 | 5,0 | 5,0 | **100,0** | 75,0 | 5,0 | 5,0 | **85,0** |
| **2** | 210,0 | 25,0 | 80,0 | **315,0** | 130,0 | 25,0 | 80,0 | **235,0** |
| **3** | 125,0 | 20,0 | 63,5 | **208,5** | 110,0 | 20,0 | 64,0 | **194,0** |

Fonte: O autor (2019).

Para facilitar o entendimento, a Figura 5.14 apresenta graficamente todos os tempos de processamento computacional.

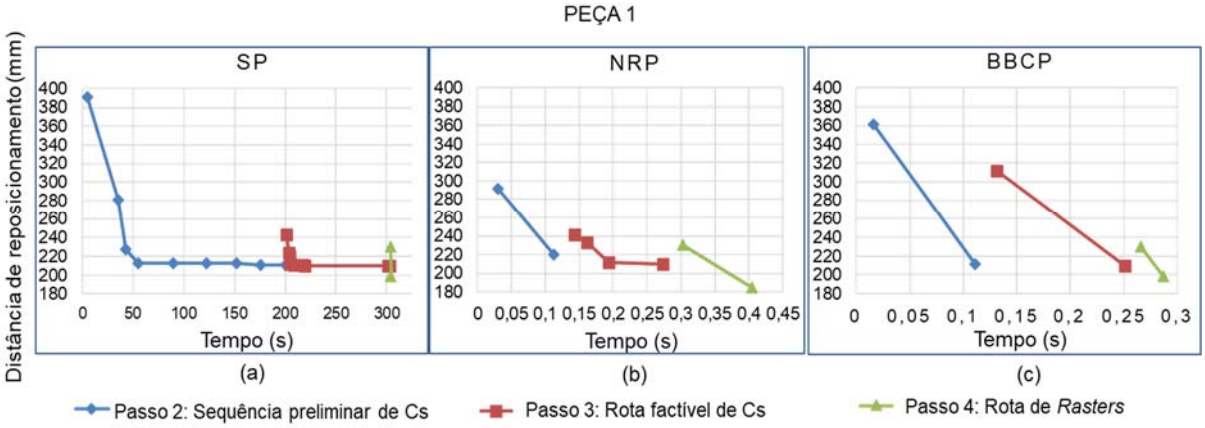
Figura 5.14 – Representação gráfica dos totais dos tempos de processamento computacional (em segundos) para os diferentes métodos propostos (Tabela 5.2)




Fonte: O autor (2019).

Em geral, observa-se que a geração da sequência preliminar de Cs (Passo 2) e rota CRS (Passo 4) exige muito mais tempo do que a rota factível de Cs (Passo 3). Para entender o comportamento dos modelos MILP visando definir os critérios de parada, os tempos de processamento das funções objetivo foram avaliados na primeira camada, conforme apresentado nas Figuras 5.15, 5.16 e 5.17 para as peças 1, 2 e 3, respectivamente.

Figura 5.15 – Evolução da função objetivo para a peça 1: MILP com SP (a), NRP (b) e BBCP (c)

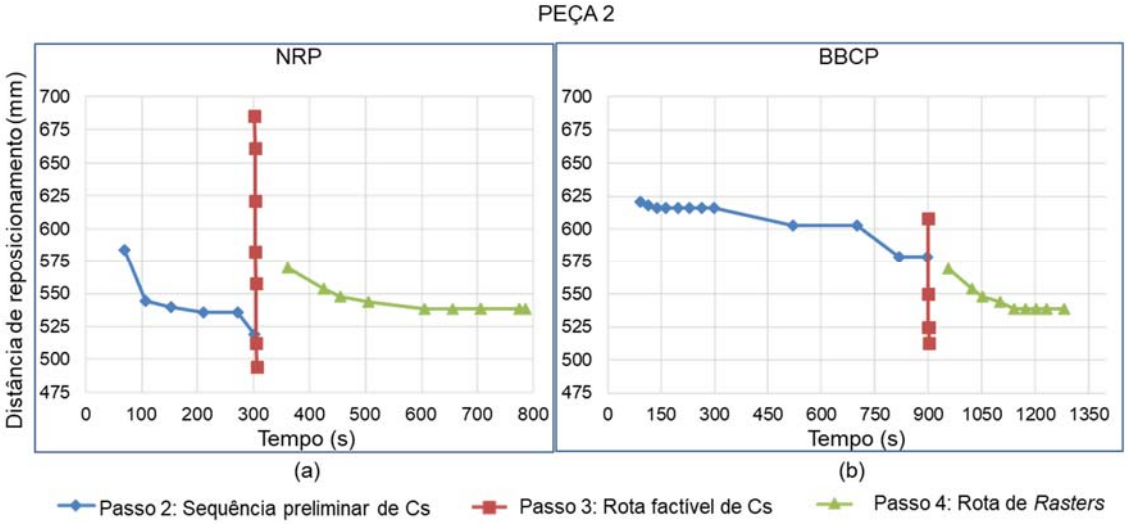



Fonte: O autor (2019).

Analisando as Figuras 5.15, 5.16 e 5.17, a importância dos passos 1 e 2 se torna evidente para os modelos MILP. Por exemplo, para a peça 1 (Figura 5.15), usando a simplificação de SP, o passo 2 alcançou uma boa solução em aproximadamente 200 segundos. As opções NRP e BBCP obtiveram soluções semelhantes em aproximadamente 0,11 segundos. Assim, verifica-se que é vantajoso investir mais tempo na geração de uma boa sequência preliminar de Cs (Passo 2), porque determina muito da qualidade do resultado final, levando a um resultado geral melhor no final de todos os passos.

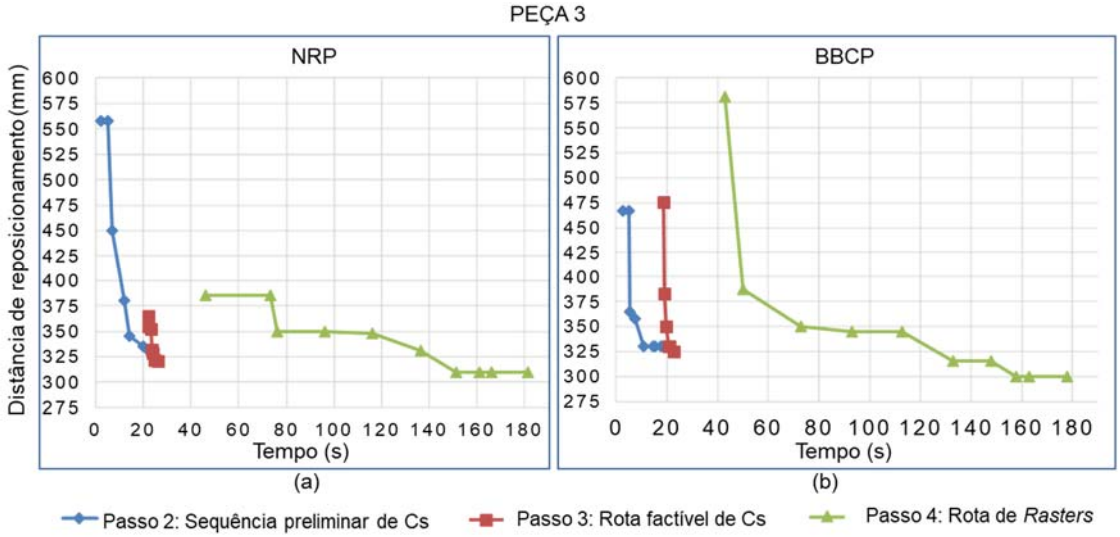
Foi observado que a peça 2 (1111 pontos), contendo 26 Cs e 26 CRSs em 17 ilhas esparsas, e a peça 3 (1058 pontos), com 22 Cs e 22 CRSs em 13 ilhas formando dois *clusters* foram resolvidas pelos modelos MILP e BT em tempos significativamente diferentes para as duas opções de simplificação (NRP e BBCP). Para a peça 2 (Figura 5.16), o Passo 2 levou aproximadamente 300 e 900 segundos, para as simplificações NRP e BBCP. Para a peça 3 (Figura 5.17), os modelos MILP utilizando as simplificações NRP e BBCP obtiveram resultados para a sequência preliminar de Cs em aproximadamente 20 segundos.

Figura 5.16 – Evolução da função objetivo para a peça 2: MILP com NRP (a) e BBCP (b)




Fonte: O autor (2019).

Figura 5.17 – Evolução da função objetivo para a peça 3: MILP com NRP (a) e BBCP (b).




Fonte: O autor (2019).

Comparando os tempos de processamento para as peças 2 e 3, fica claro que o tamanho do problema (quantidade de pontos e ilhas) e a geometria da camada (distribuição dos pontos e ilhas) influenciam diretamente nos tempos de processamento computacional. Sendo assim, quanto mais ilhas existirem em uma camada (por mais simples que sejam), maior será o tempo de processamento e, consequentemente o potencial de ganho, pois, necessariamente, ocorrerão mais reposicionamentos, conforme observado na peça 2.

É esperado também que os resultados obtidos pelas metaheurísticas de BT para as distâncias de reposicionamento sejam piores àqueles obtidos pelos modelos MILP, mas isso é compensado pelo tempo de processamento muito menor, embora

isso não tenha sido observado na peça 1 devido à sua geometria simples. Analisando os tempos de processamento investidos para as simplificações NRP e BBCP, verifica-se que para as peças 2 e 3 o tempo de processamento BBCP foi bem maior, quando deveria ter sido ao contrário, pois a simplificação BBCP é a que possibilita a maior redução dos pontos de C. Este comportamento inesperado dos modelos MILP BBCP não foi investigado, mas acredita-se que possam existir simetrias durante os cálculos, ou seja, cálculos que levam ao mesmo resultado são executados várias vezes.

Embora os resultados obtidos pelos modelos MILP sejam considerados bons do ponto de vista das distâncias de reposicionamento, o custo computacional desse método foi elevado para as peças 2 e 3. Dessa forma, sua aplicação na otimização de problemas complexos com centenas de camadas pode ser limitada em função do tempo de processamento, principalmente se a produção de peças for baixa. Por exemplo, considerando os tempos de processamento atuais, o investimento no cálculo de otimização para 400 camadas poderia ser de até 455.448 segundos (126,51 horas) e 74.704 segundos (20,75 horas) para as peças 2 e 3, considerando a simplificação BBCP (maior tempo), ou seja, o tempo investido no cálculo seria muito maior que o tempo economizado na construção da peça. Já para a otimização de peças consideradas simples, como a peça 1, com os tempos atuais de processamento computacional é vantajoso utilizar os métodos MILP, pois o tempo investido na otimização é de aproximadamente 160 segundos, obtendo uma economia de 612 segundos (0,17 horas) no tempo de construção.

Por outro lado, imaginando-se um cenário de produção em massa utilizando a AM como processo de fabricação, torna-se viável investir bastante tempo na otimização das rotas, pois uma vez otimizadas, não existe limite no número de peças que podem ser construídas, diluindo o investimento de tempo de processamento no lote de produção, onde, qualquer redução no tempo de ciclo de construção das peças impactará positivamente na produtividade do processo.

De maneira geral, uma forma de se reduzir os tempos de processamento nos modelos MILP é alterando os critérios de parada a partir da análise da evolução da função objetivo. Por exemplo, analisando-se a Figura 5.15a, verifica-se que a partir do tempo de 50s a função objetivo do Passo 2 praticamente não obteve melhora, permitindo, neste caso aumentar o *gap* de integralidade (percentual da diferença estimada para se atingir a solução ótima) ou reduzir o tempo de execução, forçando a parada da execução.

Os tempos de processamento computacional da BT foram muito inferiores aos tempos dos modelos MILP. Mesmo assim, não podem ser considerados desprezíveis. Por exemplo, o tempo de otimização da peça 2 com 400 camadas pela simplificação NRP com 5000 diversificações (pior caso) seria de consideráveis 25.200 segundos (7 horas), e mesmo assim, como a BT teve um desempenho insatisfatório, conforme descrito anteriormente, considerando a performance atual não há garantias de se conseguir reduzir o tempo de construção da peça. Da mesma forma que nos modelos MILP, a BT também poderia ter os critérios de parada alterados visando minimizar o tempo de execução, mas isso não foi testado porque o objetivo é conseguir a menor distância possível de reposicionamentos.

## 5.5 CONSIDERAÇÕES FINAIS

Pode-se dizer que os métodos MILP e BT mostraram-se adequados para a otimização das distâncias de reposicionamento dentro do *framework* proposto. Como esperado, os resultados dos modelos MILP com as simplificações para reduzir o número de pontos de C (SP, NRP e BBCP) superaram os resultados das rotas geradas utilizando NO, Guloso, NI2OPT e BT em todas as peças, mostrando sempre um comportamento superior. Porém, também como esperado, estes apresentaram um elevado custo computacional. Cabe ressaltar que apesar deste custo maior, o MILP é importante pois serve como *benchmark* aos demais métodos, uma vez que apresenta a solução ótima dentro das restrições impostas no *framework*. No caso da BT, seu desempenho aquém do esperado (adequações na metaheurística serão investigadas futuramente) pode ser explicado principalmente pelo mau desempenho nas rotas CRS, onde sempre apresentou resultados inferiores aos demais algoritmos. Isto pode ser observado pelo excesso de cruzamentos nos reposicionamentos e pode ser minimizado através da aplicação de um método de otimização local como 2-opt.

O *framework* mostrou-se adequado dentro da proposta de utilizar-se uma estratégia de deposição sequencial. A divisão da geração da rota entre Cs nos Passos 2 e 3 surgiu da grande dificuldade dos modelos MILP em ligar com os problemas mais complexos (peças 2 e 3). Após todo o trabalho ter sido realizado, ficou claro que o que poderia ter sido melhorado foi a aquisição e tratamento dos dados, que foi realizada de forma manual (força bruta), utilizando um sistema de desenho auxiliado por

computador (CAD) e planilhas eletrônicas, demandando muito tempo e esforço e ainda suscetível a erros humanos, como aconteceu várias vezes.

Outro ponto que poderia ter sido melhorado foi o tempo de processamento computacional. Os modelos MILP e BT foram implementados para testar o *framework* proposto objetivando a otimização das distâncias de reposicionamento, sem maiores preocupações com os tempos de processamento, embora eles sejam importantes. Os modelos foram implementados, testados e refeitos algumas vezes, melhorando sua performance gradativamente. Mesmo assim, o tempo de processamento continua considerável e permite a implementação de mais melhorias, tanto de programação quanto de processamento. O que também pode contribuir neste sentido, é a modificação de parâmetros e critérios de parada, por exemplo, *gap* de integralidade, tempo de execução nos modelos MILP e para a BT o tamanho da lista tabu, número de iterações que um movimento é considerado tabu, uso de lista tabu dinâmica, alteração do número de diversificações, entre outros. Também é relevante a forma de processamento dos dados. Sabe-se que existem novas técnicas como o processamento paralelo em múltiplos processadores, o que sem dúvida contribuiria muito na redução do tempo computacional.

Também é esperado que o número de vizinhos (Passo 2) influencie diretamente o resultado, uma vez que uma vizinhança menor pode limitar o espaço de busca e uma vizinhança maior pode dificultar a obtenção de uma solução “boa” em um tempo computacional aceitável. Resultados de boa qualidade foram obtidos com o número de vizinhos ajustado para dois. Independentemente da proposta de simplificação dos pontos de C escolhida, ficou clara a importância da análise da vizinhança na geração da sequência preliminar de Cs e da rota de CRS, pois limita o número de soluções e, consequentemente, permite que os modelos MILP e BT obtenham uma solução dentro dos critérios de parada.

# 6 CONCLUSÕES E RECOMENDAÇÕES

## 6.1 CONCLUSÕES

O problema de otimização dos movimentos de reposicionamento em processos de AM por extrusão de material foi analisado e detalhado, visando entender suas características, peculiaridades e complexidade computacional. Baseado na compreensão do problema alguns métodos de simplificação foram propostos visando reduzir o seu tamanho, permitindo obter soluções de boa qualidade sem comprometer o resultado. Primeiro, foi proposto o pré-processamento dos dados com três opções inovadoras para reduzir o número de pontos de contorno, denominadas Pontos Espaçados (SP), Pontos Próximos de Pontos de *Raster* (NRP) e Ponto de Centro do Envelope (BBCP). Ainda para a simplificação do problema, posteriormente foram propostas a aplicação de um caso especial de geração da vizinhança e uma nova aplicação dos polígonos de visibilidade, permitindo reduzir ainda mais o espaço de busca.

A estrutura (*framework*) proposta para otimizar os movimentos de reposicionamento seguindo a estratégia de deposição sequencial (todos os contornos primeiro e depois os trechos de *raster*) utilizando a divisão e simplificações propostas foi dividida em quatro passos principais. Com este *framework* é possível obter soluções através de outras técnicas de pesquisa operacional que se adequem ao problema. Inicialmente, foi proposta uma etapa de Pré-Processamento dos Dados (Passo 1), idealizado para ser genérico, ou seja, as simplificações adotadas objetivam reduzir o tamanho do problema, permitindo também sua aplicação com outros métodos de otimização. Em seguida, a geração da Sequência Preliminar de Contornos (Passo 2) faz uso das simplificações adotadas no Passo 1 e das restrições de vizinhança, como forma de simplificar e reduzir ainda mais o espaço de busca. Na geração da Rota Factível de Contornos (Passo 3), o uso dos polígonos de visibilidade permite eliminar os pontos de contorno que provavelmente não trarão melhorias na função objetivo e, como resultado, reduzirão ainda mais o espaço de busca e o custo computacional. Por fim, a Rota de *Rasters* (Passo 4) é gerada utilizando também a análise da vizinhança como forma de reduzir o espaço de busca.

O *framework* foi testado utilizando modelos de Programação Linear Inteira Mista Híbrida (MILP) e metaheurísticas de Busca Tabu (BT) aplicados na solução dos

Passos 2, 3 e 4. Os modelos MILP foram usados como *benchmark* aos demais métodos implementados e comparados. Os resultados foram comparados com os obtidos a partir de um sistema de planejamento de processo denominado RP3 que gerou as rotas de reposicionamento Não Otimizadas (NO) e otimizadas utilizando os algoritmos Guloso e Inserção do mais próximo com 2-opt (NI2OPT). Os resultados obtidos mostraram que, utilizando o *framework* com as simplificações, os modelos MILP obtiveram resultados superiores aos demais algoritmos. Mas, conforme esperado, os melhores resultados dos modelos MILP foram obtidos com um elevado custo computacional. Para a otimização utilizando BT o resultado foi aquém do esperado, principalmente devido ao baixo desempenho na otimização entre trechos de *raster.*

Os resultados mostraram que as três abordagens de simplificação (SP), NRP e BBCP) são comparáveis entre si, não comprometendo os resultados. No entanto, isto não foi exaustivamente testado. A principal desvantagem do *framework* é que as simplificações propostas não permitem obter a solução ótima. Isso é causado principalmente pelos Passos 1 e 2, pois as simplificações realizadas e as restrições de vizinhança reduzem o espaço de busca para reduzir o custo computacional.

De maneira geral, pode-se afirmar que a hipótese dessa pesquisa foi comprovada, ou seja, a decomposição e simplificação permitiram a solução de instâncias maiores do problema, mas, o tempo atual de processamento computacional pode limitar o uso destes modelos nas otimizações de problemas do mundo real, com centenas de camadas. Sendo assim, melhorias no tempo computacional precisam ser implementadas.

Finalmente, este trabalho contribuiu cientificamente com a área de AM, principalmente com relação aos processos baseados no princípio de extrusão de material, detalhando e ampliando os conhecimentos sobre otimização dos movimentos de reposicionamento que são caraterísticos dos mesmos.

## 6.2 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Embora a estrutura proposta seja eficaz e tenha produzido bons resultados com o MILP e aquém do esperado para o BT, cita-se alguns pontos de melhoria que podem gerar trabalhos futuros. Os dados obtidos e utilizados nas análises são uma simulação de uma impressão real, sendo assim é necessário futuramente realizar o

teste das propostas apresentadas neste trabalho em impressões reais. Outro ponto que deve ser implementado é a realização de mais testes com outras geometrias de peças, para verificar se os resultados obtidos neste trabalho se confirmam. Também é necessário implementar de maneira automatizada a obtenção e tratamento dos dados utilizados no pré-processamento, geração da vizinhança e formação dos polígonos de visibilidade. Outro ponto importante que deve ser considerado é o estudo de formas de se reduzir o tempo de processamento computacional, que mostrou-se muito elevado para os modelos MILP e considerável para a BT. Como o *framework* foi idealizado para uma sequência de deposição sequencial, como acontece nos equipamentos da marca Stratasys Ltd, é interessante para efeitos de comparação, permitir a realização de movimentos de limpeza do cabeçote extrusor. Também é interessante estudar formas de se adaptar o *framework* e os modelos implementados para realizar a otimização seguindo uma abordagem de deposição intercalada. Por fim, é necessário entender e propor formas de melhorar a performance da BT, principalmente na geração das rotas de CRSs.

Neste sentido, as sugestões para trabalhos futuros são:

- Testar o *framework* proposto em mais casos;
- Realizar impressões reais e comparar os resultados com as simulações;
- Automatização dos métodos de simplificação (Passo 1), identificação de vizinhança (Passo 2) e formação de polígonos de visibilidade (Passo 3);
- Incluir a possibilidade de limpeza do cabeçote extrusor;
- Estudar formas para redução do tempo de processamento computacional nos modelos utilizados;
- Estudar formar de melhoria das rotas geradas nos trechos de *rasters* (CRSs) utilizando a BT, e;
- Adaptar o *framework* para uma estratégia de deposição intercalada.

# REFERÊNCIAS


AARTS, E.; AARTS, E.H.L.; LENSTRA, J.K. (Ed.). **Local search in combinatorial optimization**. Princeton: University Press, 2003.

AGARWALA, M.K.; JAMALABAD, V.C.R.S.; LANGRANA, N.A.; SAFARI, A., WHALEN, P.J; DANFORTH, S.C. **Structural quality of parts processed by fused deposition, Rapid Prototyping Journal**, 2(4): p4-19, 1996.

AHASAN N; KHODA, B. **AM optimization framework for part and process attributes through geometric analysis**. Journal of Additive Manufacturing,11 pp. 85-96, 2016.

AHN, S. H.; MONTERO, M.; ODELL, D.; ROUNDY, S; WRIGHT, P. K. **Anisotropic material properties of fused deposition modeling ABS**. Rapid Prototyping Journal, v. 8, n. 4, p. 248-257, 2002.

ALMEIDA, D.; LOURENÇO, P.; PINTO, S. **Programação Linear.** Faculdade de Ciências e Tecnologia da Universidade de Coimbra / Departamento de Matemática. Coimbra, 2003.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional: para cursos de engenharia**. Rio de Janeiro, RJ: Elsevier. 2007.

ARYA, S. MOUNT, D.M. NETANYAHU, N.S. SILVERMAN, R. WU, A.Y. **An optimal algorithm for approximate nearest neighbor searching in fixed dimensions**, J. ACM. 45, 891–923, 1998.

BLUM, C.; RAIDL, G.R. **Hybrid Metaheuristics: Powerful Tools for Optimization**. Switzerland: Springer, 2016.

BRADLEY, S.P.; HAX, A.C.; MAGNANTI, T.L. **Applied mathematical programming**, Reading, MA, USA: Addison-Wesley, 1977.

BUSSAB, W.O.; MORETTIN, P.A. **Estatística Básica**, 5ª Edição, São Paulo: Saraiva, 2006.

CASTELINO K.; D'SOUZA C.R.S.; WRIGHT P.K. **Toolpath optimization for minimizing airtime during machining.** Journal of Manufacturing Systems, v. 22, n. 3, p. 173–180, 1999.

CHRISTOFIDES, N. **Worst-case analysis of a new heuristic for the travelling salesman problem**, DTIC Document, Tech. Rep., 1976.

CURA 3D. **Printing slicing software**. [Online]. Disponível em: https://ultimaker.com/en/products/cura-software, Acessado: 19-05-2019.

DAVIS, L. **Handbook of genetic algorithms.** New York: Van Nostrand Reinhold, 1991.

DORIGO, M.; STÜTZLE, T. **Ant Colony Optimization**. Massachusetts Institute of Technology. Cambridge, 2004.

DREO, J.; AUMASSON, J.P.; TFAILI, W.; SIARRY, P. **Adaptive learning search, a new tool to help comprehending metaheuristics**. International Journal on Artificial Intelligence Tools, p. 23, 2006.

FOK, K.Y.; GANGANATH. N.; CHENG, C.T; TSE, C. **A 3D printing path optimizer based on Christofides algorithm.** In: 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW). IEEE, p. 1-2, 2016.

FOK, K.Y.; CHENG, C.T; GANGANATH. N.; IU, H.; TSE, C. **An ACO-Based Tool-Path Optimizer for 3-D Printing Applications**. IEEE Transactions on Industrial Informatics, v. 15, n. 4, p. 2277-2287, 2018.

FOK, K.Y.; CHENG, C.T; TSE, C.; GANGANATH. N. **A relaxation scheme for TSP-based 3D printing path optimizer**. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). IEEE, p. 382-385, 2016b.

GANGANATH, N.; CHENG, C.T; FOK, K.Y.; TSE, C. **Trajectory planning for 3D printing: A revisit to traveling salesman problem.** In: 2016 2nd International Conference on Control, Automation and Robotics (ICCAR). IEEE, 2016. p. 287-290.

GIBSON, I.; ROSEN, D.W.; STUCKER, B. **Additive manufacturing technologies**, New York: Springer, 2010.

GILLI, M.; WINKER, P., **A Review of Heuristic Optimization Methods in Econometrics.** Swiss Finance Institute Research Paper No. 08-12, 2012.

GHAIEBI, H.; SOLIMANPUR, M. **An ant algorithm for optimization of hole-making operations.** Computers & Industrial Engineering, v. 52, n. 2, p. 308-319, 2007.

GLOVER, F.; LAGUNA, M. **Tabu search**. In: Handbook of combinatorial optimization., Boston, MA: Springer, 1998.

GOLDBARG, M. C.; LUNA, H. P. C., **Otimização combinatória e Programação Linear: Modelos e Algoritmos**, 2 ed. São Paulo: Elsevier, 2005.

GUPTA, A.K.; CHANDNA, P.; TANDON, P. „Optimization of machining parameters and tool selection in 2.5 D milling using genetic algorithm". **International Journal of Innovative Technology & Creative Engineering**, v. 1, n. 8, 2011.

HABIB, A.; KHODA, B. **Attribute driven process architecture for additive manufacturing.** Robotics and Computer-Integrated Manufacturing, v. 44, p. 253-265, 2017.

HAN, W.; JAFARI, M. A.; SEYED, K. **Process speeding up via deposition planning in fused deposition-based layered manufacturing processes**. Rapid Prototyping Journal, v. 9, n. 4, p. 212-218, 2003.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à pesquisa operacional**. Tio de Janeiro, Brasil: McGraw Hill, 2013.

HOOS, H.; STÜTZLE, T. **Stochastic local search: Foundations and applications**. Amsterdã: Elsevier, 2004.

IORI, M; NOVELLANI, S. **Optimizing the nozzle path in the 3D Printing Process**. In: International Conference on Design, Simulation, Manufacturing: The Innovation Exchange. Springer, Cham, p. 912-924, 2019.

JIN, G. Q.; LI, W. D.; TSAI, C. F.; WANG, L. **Adaptive tool-path generation of rapid prototyping for complex product models**. Journal of manufacturing systems, v. 30, n. 3, p. 154-164, 2011.

JIN, Y. A.; HE, Y.; FU, J. Z.; GAN, W. F.; LIN, Z. W. **Optimization of tool-path generation for material extrusion-based additive manufacturing technology.** Additive Manufacturing, 2013.

JIN, Y.; HE, Y.; FU, G.; ZANG, A.; DU, J. **A non-retraction path planning approach for extrusion-based additive manufacturing**. Robotics and Computer-Integrated Manufacturing, v. 48, p. 132-144, 2017.

JOE, B.; SIMPSON, R.B. **Corrections to Lee's visibility polygon algorithm**. BIT Numerical Mathematics, v. 27, n. 4, p. 458-473, 1987.

KARGUPTA, H.; GOLDBERG, D. E., **Black box Optimization: Implications of SEARCH"**, University of Illinois, Urbana-Champaign, 1995.

KHAN, W. A.; HAYHURST, D. C.R.S.; CANNINGS, C. **Determination of optimal path under approach and exit constraints**. European journal of operational research, v. 117, n. 2, p. 310-325, 1999.

KIRKPATRICK, S.; GELLAT, D. C. e VECCHI, M. P., **Optimization by Simulated Annealing. Science**, pp. 671-680, 1983.

KOLAHAN, F.; LIANG, M. **Optimization of hole-making operations: a tabu-search approach.** International Journal of Machine Tools and Manufacture, v. 40, n. 12, p. 1735-1753, 2000.

KOZA, J.R. **Genetic Programming: On the Programming of Computers by Means of Natural Selection**. MIT - Massachusetts Institute of Technology, Cambridge, 1997.

KULKARNI P., MARSAN A., DUTTA D. **A review of process planning techniques in layered manufacturing**. Additive Manufacturing Journal, v. 6, n. 1, p.18–35, 2000.
LAPORTE, G; PALEKAR, U. **Some applications of the clustered travelling salesman problem**. Journal of the operational Research Society, v. 53, n. 9, p. 972-976, 2002.

LETCHFORD, A.N.; LYSGAARD, J.; EGLESE, R.W. **A branch-and-cut algorithm for the capacitated open vehicle routing problem**. Journal of the Operational Research Society, v. 58, n. 12, p. 1642-1651, 2007.

LOPES, H.S; RODRIGUES, L.C.A; STEINER, M.T.A. **Meta-heurísticas em pesquisa operacional**. Curitiba, PR: Omnipax, 2013.

LOURENÇO, H.R.; MARTIN, O.C.; STÜTZLE, T. **Iterated local search. In**: **Handbook of metaheuristics**. Boston, MA: Springer, 2003.

MARINGER, D.G. **Distribution assumptions and risk constraints in portfolio optimization.** Computational Management Science, v. 2, n. 2, p. 139-153, 2005.

MARTÍ, R.; REINELT, G. **Heuristic methods. In: The Linear Ordering Problem**., Berlin: Springer, p. 17-40, 2011.

MARTINS, P. G.; LAUGENI, F. P. **Production management**. São Paulo: Saraiva, 2005.

MEYER, P.L. **Probabilidade: aplicações à estatística**. Curitiba, PR: Livro Técnico, 2000.

MITTELMANN, H. D. **Recent Benchmarks of Optimization Software**, 22nd European Conference on Operational Research, EURO XXII Prague, Czech Republic: Dept of Math and Stats Arizona State University, 2007.

MLADENOVIĆ, N; HANSEN, P. **Variable neighborhood search**. Computers & operations research, v. 24, n. 11, p. 1097-1100, 1997.

MOHAMED, O.A.; MASOOD, S.H.; BHOWMIK, J.L. **Optimization of fused deposition modeling process parameters: a review of current research and future prospects**. Advances in Manufacturing, v. 3, n. 1, p. 42-53, 2015.

MOHANDESSI, A. **Commercial Vehicle Scheduling with Time window by Applying Tabu Search and Cooperative Hopfield Algorithms**. Tese de Doutorado. California State Polytechnic University, Pomona, 2017.

OYSU, C.; BINGUL, Z. **Application of heuristic and hybrid-GASA algorithms to tool-path optimization problem for minimizing airtime during machining**. Eng. Appl. of AI 22(3), p. 389-396, 2009.

PAUL, G. **Comparative performance of tabu search and simulated annealing heuristics for the quadratic assignment problem**. Operations Research Letters, v. 38, n. 6, p. 577-581, 2010.

GENDREAU; POTVIN. **Handbook of metaheuristics**. New York: Springer, 2010.

PUCHINGER J., RAIDL G.C.R.S. **Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification.** Lecture Notes in Computer Science, vol 3562. Springer, Berlin, Heidelberg, 2005.

RAIDL, G.R. **A unified view on hybrid metaheuristics. In: International Workshop on Hybrid Metaheuristics. Springer, Berlin, Heidelber**g, p. 1-12, 2006.

RITZMAN, L.P.; KRAJEWSKI, L.J. **Administração da produção e operações**. São Paulo: Pearson Education, 2004.

ROYCHOUDHURY, B; MUTH, J.F. **Tool path optimization procedures for machine tools**. Computers & industrial engineering, v. 28, n. 2, p. 367-377, 1995.

SAXENA, A., PRASAD, M., GUPTA, A., BHARILL, N., PATEL, O.P., TIWARI, A., ER, M.J., DING, W., LIN, C.T. **A review of clustering techniques and developments,** Neurocomputing**.** 267, 664–681, 2017.

SPIEGEL, M. R.; **ESTATÍSTICA, Coleção Schaum**. São Paulo: Mcgraw-Hill, 1977.

TAHA, H A. **Integer programming: theory, applications, and computations**. Arkansas: Academic Press, 2014.

TANG, K; PANG, A. **Optimal connection of loops in laminated object manufacturing.** Computer-Aided Design, v. 35, n. 11, p. 1011-1022, 2003.

VOLPATO, N.; FOGGIATTO, J.A.; LIMA, M.V.A. de.; MANCZAK, T. **Uma Otimização da Estratégia de Preenchimento do Processo FDM**. In: 4º Congresso Brasileiro de Engenharia de Fabricação, Estância de São Pedro. Anais do Congresso COBEF. São Paulo: ABCM, v. 1, 2007a.

VOLPATO, N.; AHRENS, C. H.; FERREIRA, C. V.; GÜNTHER, P.; CARVALHO, J. de.; SANTOS, J. CRS. L. dos.; SILVA, J. V. L. da. **Prototipagem Rápida: Tecnologia e Aplicações**. Primeira Edição. São Paulo: Blücher, 2007b.

VOLPATO, N.; FOGGIATTO, J. A. **The development of a generic Rapid Prototyping process planning system.** In: Innovative Developments in Design and Manufacturing. CRC Press, p. 399-406, 2009.

VOLPATO, N.; NAKASHIMA, C.R.S.T.; GALVÃO, L.C.; BARBOZA, A.O.; BENEVIDES, P.F.; NUNES, L.F. **Reducing repositioning distances in fused deposition-based processes using optimization algorithms. In: High Value Manufacturing**: Advanced Research in Virtual and Rapid Prototyping: Proceedings of the 6th International Conference on Advanced Research in Virtual and Rapid Prototyping, Leiria, Portugal, 1-5 October, 2013.

VOLPATO, N.; GALVÃO, L.C.; NUNES, L.F.; SOUZA, R.I.; OGUIDO, K. **Combining heuristics for tool-path optimisation in material extrusion additive manufacturing**. Journal of the Operational Research Society, p. 1-11, 2019.

VOLPATO, N.; SILVA, J.V.L. **Planejamentos de processo para tecnologias de AM**. In: VOLPATO, N. (ed.), Manufatura Aditiva – Tecnologias e aplicações da impressão 3D. São Paulo: Blucher, 2017.

WAH P.K.; MURTY K.G.; JONEJA A.; CHIU L.C. **Tool path optimization in layered manufacturing**. IIE TRANS, v. 34, n. 4, p. 335–347, 2002.

WALPOLE, R.E.; MYERS, R.H. **Probabilidade e Estatística para Engenharia**. São Paulo: Pearson, 2009.

WEIDONG, Y. **Optimal path planning in Rapid Prototyping based on genetic algorithm**. In: Control and Decision Conference, 2009. CCDC'09. Chinese. IEEE, p. 5068-5072, 2009.

WOEGINGER, G.J. **Exact Algorithms for NP-Hard Problems: A Survey, Combinatorial Optimization**. Lecture notes in computer science, vol. 2570, Springer, pp. 185–207, 2003.