CENTRO UNIVERSITÁRIO FEI
CAIO JORGE GAMARRA

MODELAGEM, OTIMIZAÇÃO E SIMULAÇÃO DE UMA ÁREA DE *PICKING*

São Bernardo do Campo
2021

CAIO JORGE GAMARRA

MODELAGEM, OTIMIZAÇÃO E SIMULAÇÃO DE UMA ÁREA DE *PICKING*

Dissertação de Mestrado apresentada ao Centro Universitário FEI para obtenção do título de Mestre em Engenharia Mecânica. Orientado pelo Prof. Dr. Fábio Lima.

São Bernardo do Campo

2021

Jorge Gamarra, Caio.
MODELAGEM, OTIMIZAÇÃO E SIMULAÇÃO DE UMA ÁREA DE PICKING / / Caio Jorge Gamarra. São Bernardo do Campo, 2021.
90 p. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Fábio Lima.

1. Modelagem. 2. Simulated Annealing. 3. Otimização. 4. Centro de distribuição. 5. Picking.. I. Lima, Fábio, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Mecânica

Mestrado

PGM-10

Aluno: Caio Jorge Gamarra **Matrícula:** 219306-8

Título do Trabalho: MODELAGEM, OTIMIZAÇÃO E SIMULAÇÃO DE UMA ÁREA DE PICKING.

Área de Concentração: Produção

Orientador: Prof.º Dr.º Fábio Lima

Data da realização da defesa: 10/08/2021

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

O trabalho foi aprovado por unanimidade pela banca examinadora. O aluno respondeu assertivamente aos questionamentos realizados. Há potencial significativo de geração de uma publicação científica em periódico, uma vez que a dissertação tem contribuições científicas e práticas.

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof.º Dr.º Fábio Lima	Ass.: _____
Prof.º Dr.º Mauro Sampaio	Ass.: _____
Prof.º Dr.º Claudio Barbieri da Cunha	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Rodrigo Magnabosco

Dedico este trabalho aos familiares e amigos que estiveram juntos nesta etapa demonstrando apoio para a conclusão deste desafio.

AGRADECIMENTOS

Agradeço à minha esposa, grande companheira que me incentivou e me apoiou durante esta empreitada.

Agradeço aos professores do Centro Universitário da FEI pelas aulas, ensinamentos e oportunidades.

Ao meu orientador Prof. Dr Fábio Lima por ter aceitado o desafio, apesar das pedras no caminho.

Aos meus colegas de trabalho que auxiliaram com conselhos e orientações para a conclusão deste trabalho

Ao Centro Universitário FEI pela infraestrutura e recursos disponibilizados para o desenvolvimento do presente trabalho

RESUMO

Diversos estudos indicam a importância do setor de logística nas empresas, principalmente naquelas cujos pedidos possuem grande variedade de produtos. Nestas, o setor de *picking* destaca-se, pois é a operação mais demorada para a empresa. Isto ocorre devido a variedade elevada de produtos e a demanda de movimentação de material para a composição dos pedidos. Uma forma de evitar desperdício de tempo e diminuir o esforço dos operadores de *picking* é realizar o estudo da rota de coleta dos itens para a montagem dos pedidos. A implementação de políticas de rotas promove diminuição na distância percorrida pelos funcionários e maior agilidade na montagem dos pedidos. A literatura apresenta diversas formas de promover benefícios na operação de *picking*. Este trabalho propõe uma metodologia combinando a modelagem do centro de distribuição (CD) em um software de simulação e a utilização de um algoritmo de otimização em busca de prover uma rota específica para realização das sequências de coleta para todos os pedidos com o objetivo de diminuir as distâncias percorridas pelos funcionários. Esta metodologia utiliza a simulação, em uma primeira iteração, para facilitar a obtenção de dados que seriam de grande dificuldade de coletar fisicamente, por exemplo a matriz de distâncias entre todas as posições. Com os dados disponibilizados pelo modelo, é possível então implementar um algoritmo que calcule sequências de coleta de menor distância. Por fim, o modelo de simulação pode ser utilizado novamente para a validação dos valores encontrados e teste da viabilidade de implementação da rota de coleta proposta pelo algoritmo. Neste trabalho, esta metodologia foi posta em prática com informações de uma grande empresa do setor de bebidas no Brasil, a qual teve um dos seus centros de distribuição modelado a fim de avaliar os desperdícios da operação de montagem dos pedidos. O algoritmo para estudo das rotas de coleta selecionado foi o *Simulated Annealing* devido a sua agilidade em convergir em um resultado. Aplicando esta metodologia neste Centro de Distribuição, obteve-se uma redução de mais de 7% na distância percorrida pelos operadores, sem a necessidade de investimentos em infraestrutura.

Palavras-chave: Modelagem, Simulated Annealing, Otimização, Centro de distribuição, *Picking*.

ABSTRACT

Several studies indicate the importance of the logistics in companies, especially in those which orders have a wide variety of products. In those Companies, the picking sector stands out, as it is a more time-consuming operation for the company. This is due to the high variety of products and high demand for material handling for order composition. One way to avoid wasting time and also decrease the effort of the operator is to study the route of collection of the items for order-fitting. The implementation of route policies reduces the distance traveled by employees and improve agility in order completion. The literature presents several ways to promote benefits in picking operation. This work proposes a methodology combining a model of a Distribution Center on a simulation software and the use of an optimization algorithm in search of a specific sequence to perform the collection routes for all orders with the objective of decreasing distances traveled by employees. This methodology uses simulation, in a first iteration, to facilitate obtaining data that would be really difficult to collect physically. With the data made available by the model, it is then possible to implement an algorithm that calculates collection sequences travelling shorter distances. Finally, the simulation model can be used again to validate the values found and test the feasibility of implementing the collection route proposed by the algorithm. In this work, this methodology was put into practice with information from a large company in the beverage industry in Brazil, which had one of its distribution centers modeled in order to assess the waste of the order-fitting operation. The selected algorithm for studying collection routes was Simulated Annealing due to its agility in converging on a result. Applying this methodology in this Distribution Center resulted in a reduction of more than 7% in the distance covered by the operators, without the need for investments in infrastructure.

Keywords: Modeling, Simulated Annealing, Optimization, Distribution center, Picking.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SKU - Stock keeping unit

AGV - Automated guided vehicle

CD - Centro de distribuição

WMS - Warehouse Management System

KPI - Key performance Indicator

ACD - Activity Cycle Diagram

SUMÁRIO

1 INTRODUÇÃO

Atualmente as empresas estão enfrentando um cenário de mudanças desafiador, onde cada nova tecnologia desenvolvida oferece melhorias em diversos setores. Porém, estas melhorias só podem ser alcançadas através da atualização de sistemas de manufatura e integração de todos setores da companhia, desde o desenvolvimento até a logística, lembrando que, para casos mais específicos pode ser necessária a integração do processo de descarte de produtos. Estas integrações estão ligadas aos sistemas de gerenciamento de informação (GENG *et al.*, 2017) e isto se aplica não somente aos setores produtivos, mas também aos setores que tratam da logística interna e externa dos produtos.

No setor logístico, mais precisamente nos centros de distribuição com operações de *picking*, como o que será estudado neste trabalho, os custos com o deslocamento e movimentação dos operadores podem passar de 50% dos custos totais (DE KOSTER et al., 2007, RICHARDS, 2018). A operação de *picking* consiste na seleção de produtos armazenados em locais específicos para a formação de um conjunto de itens que satisfaça o pedido realizado pelo cliente.

Visando reduzir custos e aumentar a eficiência das operações na zona de *picking*, algumas mudanças podem ser empregadas, como:

- realizar alterações no *layout*;
- utilização de estruturas porta *pallet*;
- utilização de maquinário para automatização do centro de distribuição;
- estudo da ordem de coleta dos itens

Sendo o "estudo da ordem de coleta dos itens", propondo a reestruturação da lista de *picking* dos operários, o objeto de estudo deste trabalho.

Como a logística e o processo de distribuição podem afetar diretamente a confiabilidade de uma empresa para os seus clientes, este fator serve como vantagem competitiva para as empresas que buscam estar à frente do mercado (MORESCO, 2017). Em busca de aumentar a capacidade de atendimento ao cliente, se faz necessária a diminuição de desperdícios nas áreas operacionais, e a tecnologia e o desenvolvimento de sistemas de informação, como o WMS - *Warehouse Management System*, apoiam este incremento nos níveis de atendimento ao consumidor.

Uma boa parte das empresas brasileiras apresenta uma deficiência no quesito atualização de informações dos seus sistemas. É comum encontrar centros de distribuição onde

as alterações no layout, informações relevantes para a operação, ou até mesmo o posicionamento dos produtos pode estar defasado no sistema. Quando se trata de *picking*, isto pode levar os operadores a cumprirem sequências mais longas do que o necessário na coleta dos produtos. Outro fator que pode dificultar a atualização constante dos seus sistemas é grande rotatividade de SKUs (*Stock Keeping Units*), que se adequam a demandas sazonais, além do constante desenvolvimento de novos produtos, como no caso da empresa estudada neste trabalho. Nestes casos a aplicação da metodologia estudada neste trabalho se torna uma maneira mais barata de atualizar a ordem de coleta dos itens destes centros de distribuição.

A pesquisa operacional pode auxiliar a busca de soluções para os problemas do setor de *picking*, definida como uma abordagem científica para a tomada de decisão que busca a melhor maneira de operar um sistema, considerando a escassez de recursos disponíveis (Winston, 2004). O problema das rotas pode ser explorado com a otimização combinatória, uma subárea da pesquisa operacional.

Dependendo da complexidade, a resolução dos problemas combinatórios pode ser realizada a partir de métodos exatos, métodos heurísticos, ou métodos híbridos, combinando os dois primeiros (MARTINS et al., 2003). Como o tempo de resolução torna-se muito grande para os problemas com muitas possiblidades, as heurísticas representam uma maneira de reduzir o número de possibilidades e obter uma solução em um tempo razoável, já que essas costumam ser procedimentos mais simples e flexíveis do que os métodos exatos de solução. Dessa forma, permite abordar modelos mais complexos e torná-los mais representativos com relação ao problema real (REEVES, 1995).

Outra ferramenta muito utilizada quando o estudo trata da análise de cenários e avaliação de impactos é a modelagem e simulação de eventos discretos. A utilização desta ferramenta traz benefícios como a possibilidade de analisar alterações de longo prazo em pouco tempo. Esta também costuma demonstrar como realmente um sistema opera, possibilitando confrontar os resultados disponibilizados com o que os envolvidos esperam da operação deste sistema (BANKS; CARSEN, 1984), além de estar inserida no contexto de manufatura digital.

Em busca de maiores produtividades dentro de um centro de distribuição, vários pontos de melhorias foram encontrados historicamente, como por exemplo o que será avaliado neste trabalho, a relação entre a *Picking list*, que é gerada através de um *software* de gerenciamento de armazém (WMS) e o *layout* do local, a qual pode se encontrar desatualizada. Para resolver este problema, propõe-se uma estratégia baseada em simulação e um algoritmo de busca por rotas mais curtas que faz a atualização da *Picking list* levando em conta a posição atual de todos os SKUs dentro do CD e outras possíveis alterações de *layout* que podem ter sido realizadas,

criando uma sequência de coleta dos produtos que resulte em rotas com menor deslocamento total dos operários.

1.1 OBJETIVO

Este trabalho tem por objetivo avaliar as contribuições da utilização de uma técnica de otimização que auxilie na minimização da distância percorrida pelos operários em uma área de *picking* em centros de distribuição de uma grande empresa do ramo de bebidas no Brasil.

1.1.1 Objetivo específico

Confirmar que a técnica proposta reduz as distâncias percorridas utilizando um modelo virtual em um software de simulação para percorrer a nova rota de coleta sugerida pelo algoritmo de otimização.

2 REVISÃO BIBLIOGRÁFICA

Nesta seção são apresentados diversos trabalhos envolvendo o tema de otimização em armazéns, principalmente os que apresentaram soluções focadas em melhorias para o setor de *picking* em centros de distribuição em busca de embasamento teórico para esta dissertação. Além disso, pesquisas envolvendo otimizações na área de *picking*, simulações e dados referentes à eficiência nos setores logísticos foram reunidos e estudados.

2.1 LOGÍSTICA E O SETOR DE PICKING

Dentro da área de logística, armazéns são considerados peças chave quando se trata de atendimento ao cliente de forma rápida e ágil. Muitas empresas já mudaram a forma de enxergar e gerenciar seus armazéns, não só como sendo apenas mais um meio de se manter o nível de estoque para atendimento ao cliente, mas como também uma maneira de estreitar a relação com o cliente, aumentando o nível de satisfação dos clientes.

> Numa perspectiva tradicional, os armazéns não acrescentam valor à cadeia de fornecimento, apenas disponibilizam os meios para manter o estoque de um determinado material nas quantidades requeridas, no ambiente apropriado e ao menor custo possível. Segundo uma visão mais atual, os armazéns deixaram de ser o ponto-morto do processo e passaram a ser uma parte integrante da cadeia de fornecimento, potenciando a excelência do serviço. (LOURENÇO 2014)

O autor ainda acrescenta que novos papéis passaram a ser desempenhados nos armazéns, como por exemplo operações de *cross-docking*, onde o material chega de sua origem, é rearranjado junto com outros produtos de um pedido e estes são despachados para o cliente final, operações de transbordo, onde se divide grandes volumes de materiais em quantidades menores que são despachadas em vários envios, entre outras operações diferentes. Entre as novas opções de operação de armazém, algumas delas fazem com que não seja mais imperativo a armazenagem de grandes quantidades de produtos por grandes períodos, tudo que se necessita é espaço suficiente para fazer o manuseio dos produtos entre o ponto de chegada e o de saída para a entrega (LOURENÇO, 2014).

Dentre as diversas áreas que um armazém ou um CD pode ter, uma das que mais se destaca em questão da importância de se manter uma alta eficiência de operação para melhores resultados e melhores taxas de atendimento ao consumidor, está o setor do *picking*. Neste setor ocorre a seleção e separação de produtos em *pallets* com os produtos solicitados em cada pedido para serem enviado aos cliente. O princípio de seleção de encomendas ou *Order Picking*, é

definido como a atividade onde um determinado número de bens são extraídos de um sistema de armazém, de forma a poder satisfazer os pedidos de um determinado cliente, seja ele um cliente independente, organização, máquina, etc. (FONTES, 2010).

De acordo com Kulak, Sahin e Taner (2012), o processo de *picking* pode ser realizado manualmente ou utilizando técnicas de automatização, sendo definidos como modelos de:

- *picker-to-part*: Processo manual, onde o operador coleta os itens escolhidos pelo cliente, seguindo até a posição do produto a ser coletado.
- *part-to-picker*: Processo automatizado, onde os itens são de alguma forma entregue ao operador.

No primeiro modelo, fatores que influenciam na produtividade são o *layout* e as distâncias percorridas. Por isso, diversas formas de operar são consideradas, como por exemplo (Kulak, Sahin e Taner, 2012):

- Separação em zonas de *picking*: a zona de *picking* é separada em áreas menores com poucos SKUs, sendo que cada área possui um operador responsável pela coleta dos itens.
- Percorrer totalmente a zona de *picking*: onde cada operador percorre totalmente os corredores da zona de *picking*, coletando os itens desejados.
- Percorrer parcialmente a zona de *picking*: nesta opção os operadores ficam livres para realizar rondas e visitar apenas os corredores onde precisam coletar itens, conforme relacionados em uma *picking list*.

Quando se trata de um setor de *picking* automatizado, onde as peças são entregues ao operador, existem algumas técnicas de fácil implementação e outras mais robustas e elaboradas, como:

- *Pick-by-Light:* Uma das formas mais simples, onde a maioria ou todas, quando possível, SKUs estão disponíveis para o operador, que fica em uma área reduzida, com estantes ou gavetas que contém os produtos, que, de acordo com os pedidos a serem entregues, acende uma luz (daí o nome *pick by light*) indicando qual produto deve ser coletado.
- Carrossel/Armazém vertical automatizados: Uma aplicação um pouco mais sofisticada do conceito de *parts-to-picker* e que compreende a necessidade de manusear um número mais elevado de SKUs, realiza a entrega de produtos a um operador, através do uso de um carrossel ou armazém vertical automatizado que

distribui os produtos nas gavetas ou estantes dos operadores conforme os pedidos são montados.

- AGVs (*Automated guided vehicle*): Uma outra aplicação bem mais tecnológica e sofisticada do conceito de *parts-to-picker* para realizar a montagem dos pedidos, é a utilização de AGVs que transportam estantes ou *pallets* até o operador que separa os itens de um pedido específico. Porém, esta opção envolve grandes investimentos em infraestrutura e sistemas de informação.

De acordo com Lourenço (2014), em um armazém convencional, os produtos são coletados por um funcionário que se movimenta à pé entre os *pallets* que contém grandes quantidades de produtos para montar o pedido a ser entregue. Já para Li, Huang, e Dai (2017), existem 2 classificações para o *picking* dentro de um armazém, se o operador se move até uma posição de uma SKU para fazer a coleta, este sistema então se classifica como *picker-to-part*, já se o produto é de alguma forma levado até o operador, ou seja, não há nenhuma ação que envolva o operador ir até a posição original do SKU, este sistema, então, é classificado com *part-to-picker*, e são sistemas relativamente novos e pouco utilizados devido ao elevado custo de implementação.

Além do tipo de movimentação, de SKUs ou de operadores, Richards (2018) destaca que os produtos podem ser selecionados em diferentes volumes, como por exemplo:

- Unitários, onde se retira de um local de armazenagem apenas 1 unidade deste produto por vez até se preencher o pedido;
- Em conjuntos, quando os produtos vêm embalados, por exemplo, 3 a 3 até se satisfazer a quantidade solicitada no pedido;
- Por embalagem, fardo ou caixa, com uma quantidade fixa de produtos em cada embalagem, sendo coletadas a quantidade dessas até preencher o pedido;
- Por camada, quando o operador pode coletar de um *pallet* que tem esse produto solicitado uma camada inteira, por exemplo 100 embalagens de um produto que é armazenado em um pallet com 10 unidades (Largura) X 10 unidades (Comprimento) X 6 unidades (Altura), isso lhe permite coletar uma camada inteira deste pallet de armazenamento;
- Por *pallet*, os operadores podem fazer a coleta levando um *pallet* inteiro para o despacho.

Diversas são as atividades que podem interferir no tempo de finalização de um pedido dentro do *picking* de um armazém, como por exemplo:

- o trajeto entre as posições de armazenagem das SKUs;
- a demora na coleta do item;
- a preparação de equipamentos utilizados;
- o recebimento das informações para a próxima coleta;
- a procura pelo item a ser retirado;

Porém, a atividade mais relevante é de fato o deslocamento realizado pelo operador que durante a coleta do material, que dependendo do trajeto escolhido pode acarretar em grande desperdício de tempo, diminuindo a eficiência da operação (Lu et al., 2016).

> O problema da definição da rota de recolha, para um determinado conjunto de localizações centra-se na correta sequência a ser dada à lista de peças a recolher, de modo a assegurar a minimização da distância percorrida, otimizando os percursos, reduzindo os tempos de viagem e consequentemente aumentando a taxa de expedição dos produtos. (LOURENÇO, 2014)

Da mesma forma, porém, de um ângulo diferente, Fontes (2010) sugere que uma boa implementação da sequência de coleta, ou *picking list*, minimizando as distâncias percorridas, permitem um aumento nos KPIs (*Key Performance Indicators*). Este fator justifica então a importância de realizar um estudo para prover uma melhoria, que seria a diminuição da distância percorrida pelo operador.

Além da operacionalização através de operários se movimentando pela área total do *picking*, com acesso à todas as SKUs, existem outras opções de operação. De acordo com Fontes (2010), é possível se organizar o funcionamento das seguintes maneiras:

- *Basic Order Picking* (*picking* discreto);
- *Batch Picking* (por lotes);
- *Zone Picking* (*picking* por zona);
- *Wave Picking* (*picking* por ondas);
- *Bucket Brigades*.

Desta maneira, existem diferentes formas de finalizar o pedido.

No *picking* discreto, o operador faz a coleta de um produto por vez, respeitando a quantidade de cada pedido, e vai acrescentando produtos em um pallet que será depois enviado ao cliente que solicitou estes bens. Portanto, o *layout* do armazém deve ser organizado levando-se em consideração os produtos que têm mais saída, deixando-os próximos entre si, e se possível, próximos da expedição. Esta operação costuma se encaixar bem em empresas que não

realizam a montagem de muitos pedidos, mas os pedidos possuem grandes quantidades de produtos. (FONTES, 2010)

Este é o método de *picking* mais comum. Entretanto, pedidos com múltiplos SKUs e com grandes distancias entre as coletas podem ser muito trabalhosos. Em muitos casos ainda é necessário que haja a verificação do pedido para verificar se todos os itens estão corretos antes de enviar para despacho (RICHARDS, 2018).

Já no *picking* por lotes, os diversos produtos são disponibilizados em pequenos lotes em uma área de consolidação. Durante o processo de separação, onde o operador visa completar vários pedidos de uma vez, o colaborador passa por essas zonas de consolidação e coleta os itens necessários. Este sistema funciona muito bem nas empresas cujos pedidos tem poucos itens por ordem de venda. Geralmente permite uma eficiência boa, pois o operador pode realizar a coleta de produtos adicionais enquanto está em determinada área. Porém, é necessário que o colaborador tenha bastante atenção ao realizar a montagem dos pedidos, e, se possível, a empresa deve garantir uma etapa de verificação dos pedidos, pare prevenir a entrega errada de produtos (FONTES, 2010)

O *picking* por zona determina uma área de atuação para cada operador, sendo ele responsável por coletar e entregar uma gama restrita de produtos. É comum a utilização de equipamentos para movimentação dos produtos, por exemplo, esteiras, pois quando um operador finaliza sua coleta, ele envia estes produtos para a próxima zona, onde outro colaborador fará as coletas referentes aos produtos restritos a ele. (FONTES, 2010)

Richards (2018) ainda acrescenta que os pedidos podem ser coletados simultaneamente entre as zonas e consolidados depois, desde que as instruções sejam geradas pelo WMS para que cada zona faça a coleta simultânea.

No *picking* por ondas, os pedidos são acumulados por um período determinado, ao final deste período os operadores são alocados para a realização desta onda, realizando uma separação prévia dos produtos de cada pedido. Ao final desta separação prévia, os produtos vão então para uma etapa de consolidação do pedido, onde há uma verificação das quantidades e alocação dos produtos na embalagem para entrega. (FONTES, 2010)

As operações baseadas no modelo *Bucket Brigades* têm benefícios de produtividade, pois visa garantir com que todos colaboradores estejam sempre realizando tarefas. Os operários ficam divididos em zonas flutuantes, onde eles separam os produtos que estão na ordem do pedido, até que o operador da zona seguinte, ao ter finalizado sua tarefa, vem e assume este pedido, gerando uma reação em cadeia de cada colaborador ir tomar posse do pedido que está sendo montado pelo operador anterior. Este sistema costuma ser eficiente, pois ele é auto

regulado, onde o operário que tem maior eficiência vai trabalhar mais, porém ele não será afetado por atrasos dos demais colaboradores. (FONTES, 2010). Devido à essa grande variedade e opções de operação, deve-se escolher o melhor modelo de acordo com as características de operação de cada CD.

> É importante que dentre as estratégias de metodologia de *picking* e processos internos existentes, a empresa utilize a que melhor se aplique a seu sistema, levando em consideração todas as necessidades da mesma para que se possa realizar o processo de separação com a qualidade e rapidez exigida. (MORESCO 2017)

Quando se trata de *picking*, não existe uma solução única que resolva todos os tipos de operação ou, como descreve RICHARDS (2018) "*one size fits all*". Este autor relaciona ainda algumas operações e as aplicações típicas para elas, que podem ser observadas no Quadro 1.

Quadro 1 Quadro comparativo – entre algumas estratégias de coleta de itens

Método de coleta	Aplicações típicas	Benefícios	Desvantagens
Picking discreto	Maioria das operações	Operação em estágio único	Taxa de coleta baixa
		Flexível	Trabalho bem intenso
		Implementação rápida	Treinamento com curva de aprendizagem longa, dependendo de quais ferramentas utiliza
Coleta por lotes exatos	Pedidos do varejo em e-commerce	Coleta de múltiplos pedidos ao mesmo tempo	A separação do volume necessário demora mais do que na coleta por nível de pallet
		Diminui o deslocamento pelo armazem	Necessita de um setor para separação e mão de obra adicional
		Aumento na acurácia	É necessário auxílio de um sistema configurar os pedidos
		Pode ser usado em operação de cross-docking	É preciso fazer a reembalagem
Coleta por lotes baseado em camadas do pallet	Pedidos do varejo em e-commerce	Coleta de múltiplos pedidos ao mesmo tempo	Necessita de um setor para separação e mão de obra adicional
		Diminui o deslocamento pelo armazem	É ncessário retornar os itens não utilizados para o estoque
		Aumento na acurácia	É preciso fazer a reembalagem
Picking por zona	Operações onde há um grande número de SKUs e baixo número de itens por pedido	Menos deslocamento para os operarios	Normalmente precisa de esteiras para deslocamento dos itens
		Pedidos podem ser coletados simultaneamente ou em sequência	Custo dos equipamentos
		Pode acomodar diferentes famílias de itens, como produtos com temperatura controlada ou produtos perigosos	Pode ocasionar ociosidade entre os operadors se o trabalho não estiver balanceado entre as zonas
Picking por ondas	Quando os pedidos são liberados em períodos específicos ou para se adequar á saída dos caminhões de entrega	Permite o agendamento eficiente do trabalho	o uso do WMS se torna necessário
		Pedidos são separados em tempo para uma rodada da produção ou a saída de um veículo	Pedidos urgentes não podem ser separados tão facilmente
Parts-to-picker	Operações de picking com alta intensidade	Alta taxa de coleta	Alto custo dos equipamentos
		Grande acurácia	Alto custo com energia
		Reduz a necessidade de espaço	Potencial de falhas do sistema
		Estações de trabalho ergonomicas	Limitado a trabalhar principalmente com itens menores
		Treinamento menos intensivo	Alto custo de oportunidade

Fonte: Adaptado de Richards (2018).

Dentre as atividades que podem gerar aumento no tempo de coleta dos produtos, a distância percorrida pelos operários é a dominante e pode ser responsável por mais do que 50% do tempo de coleta, e como o tempo de deslocamento é considerado um desperdício, já que não está adicionando valor ao produto e gera encargos, a redução desse tempo de deslocamento em sistemas *picker-to-part* é crítico para a redução dos custos operacionais (Lu *et al.*, 2016). De acordo com De Koster (2007), diminuir a distância percorrida, total ou parcialmente, é um fator imperativo quando se trata de diminuir o tempo de *picking*, e aumentar a sua eficiência no processamento e operações do armazém.

A área de *picking* já foi intensamente estudada pela comunidade científica, de questões estratégicas a questões operacionais. Diefenbach e Glock (2019) delimitam os problemas estudados em armazéns em 5 áreas:

- *Layout*, como sendo o posicionamento das ilhas e corredores na área de *picking*;

- Alocação dos produtos no armazém, ou seja, a distribuição de todas SKUs em posições de coleta para os operadores realizarem a busca dos itens para preencher os pedidos;
- Zoneamento, a divisão da área de *picking* em zonas, onde cada uma terá como responsável um indivíduo ou grupo de operadores, eles somente coletam itens da zona a que foram alocados;
- Lotes, a utilização desta política implica que o operador não coleta os itens de apenas um pedido por vez, liberando-o para buscar itens de mais de um pedido simultaneamente;

Rotas de coleta do operador, quando há a aplicação de diferentes políticas ou estudo da sequência de coleta dos itens a ser realizada por cada operador.

Petersen e Aase (2017) já ressaltam que estudos anteriores geralmente se concentravam em uma das quatro principais políticas operacionais para melhorar o desempenho do sistema: políticas de coleta, roteamento, armazenamento e *layout*. As políticas de coleta envolvem a atribuição de itens ou pedidos para rotas de coleta e incluem pedido estrito, lote e zoneamento. As políticas de roteamento determinam a rota de um selecionador para um *tour* de coleta e variam de heurísticas simples a procedimentos ideais. As políticas de armazenamento atribuem SKUs aos locais de armazenamento. As políticas de armazenamento comumente usadas são armazenamento aleatório e armazenamento baseado em classe, onde os SKUs são classificados pela análise ABC em classes de armazenamento. Por último, o *layout* desempenha um papel importante na eficiência da coleta de pedidos, incluindo a orientação e a posição da coleta e dos corredores transversais.

Dentre as diversas alternativas de melhoria do setor de *picking*, este trabalho se aprofundará na questão das rotas de coleta visando a diminuição da distância percorrida pelos operadores.

2.2 POLÍTICA DE ROTAS DE COLETA

As políticas de rotas de coleta são implementadas nos CDDs em busca de minimizar a distância percorrida pelos operadores durante a coleta dos itens dos pedidos, em busca de diminuir a fadiga dos operadores, o tempo para realização das atividades, e consequentemente, aumentando a eficiência da operação. De acordo com Masae, Glock e Grosse, (2019), como o tempo de deslocamento é responsável por mais do que 50% do tempo total de separação do pedido, o problema das rotas de coleta já recebeu atenção da comunidade científica e existem

alguns compilados de trabalhos e levantamentos bibliográficos, como por exemplo os trabalhos de Grosse, Glock e Neumann (2017) e van Gils *et al.* (2018).

Dentre as políticas de rota, é comum aplicar os métodos "*S-shape*"; "*Return*"; "*Midpoint*"; "*Largest gap*" e "Combinado" (TEIXEIRA, 2018), métodos que serão abordados mais profundamente nas seções a seguir, quando serão discutidas as formas de se diminuir a distância percorrida pelos operadores.

2.3 OTIMIZAÇÃO

Quando se trata de armazéns com o sistema "*picker-to-part*", algumas áreas costumam ser estudadas. De acordo com Li, Huang, e Dai (2017) existem 4 caminhos a se seguir, sendo eles a otimização do layout, da alocação dos produtos nas posições, dos lotes de pedidos ou, então, da rota dos colaboradores quando da realização das coletas.

A otimização do layout consiste em arranjar os diferentes produtos dentro da zona de *picking* de forma a permitir uma operação eficiente. Vale ressaltar que, uma vez definido o *layout*, os custos para alterá-lo são altos, visto que estas alterações envolvem investimento em infraestrutura e, na maioria dos casos, a interrupção total ou parcial do local (LI; HUANG; DAI, 2017).

Quando se estuda a alocação dos produtos em uma área de *picking*, procura-se posicionar as SKUs em posições que minimizem a distância percorrida pelos operadores. Uma forma de se fazer isto é posicionar os produtos com maior volume de vendas mais próximos entre si, embora existam algumas outras restrições que devem ser respeitadas, de acordo com o tipo de produto que está sendo movimentado. (LI; HUANG; DAI, 2017)

Uma alternativa bastante utilizada quando se busca aumentar a eficiência no *picking* é o estudo dos pedidos em lotes, onde um operador realiza uma rota a fim de completar não apenas 1 pedido, mas um lote de pedidos. Neste cenário deve se colocar na balança o custo e o tempo para atendimento dos clientes. (LI; HUANG; DAI, 2017)

Outra opção é reavaliar a rota dos operadores que realizam a coleta dos itens, e já recebeu alguma atenção em outros estudos, geralmente fazendo um paralelo com o Problema do Caixeiro Viajante (LI; HUANG; DAI, 2017). Outros autores ainda classificam este estudo em armazéns com corredores múltiplos como uma um cenário específico do caixeiro viajante, o Problema do Caixeiro Viajante de Steiner (KULAK; SAHIN; TANER, 2012).

Este tipo de problema, a definição da rota do operador de *picking*, é classificado como um problema do tipo *NP-Hard (Non-deterministic Polynomial time)*, portanto, dentre as tentativas de solução opta-se por heurísticas ou por métodos otimizantes (van GILS *et al.* (2018)

e LI; HUANG; DAI, (2017)). Nestes casos, heurísticas são comumente utilizadas, mesmo sabendo que a solução não será ótima, pois elas permitem a convergência em um resultado, geralmente, bom, se a questão for bem entendida e modelada (DIEFENBACH; GLOCK, 2019). Porém isto não impede que métodos otimizantes sejam utilizados, mesmo que os algoritmos sejam muito complexos e demandem grande tempo computacional para chegar em um resultado ótimo.

Masae, Glock e Grosse (2019) dissertam sobre o assunto indicando que 3 tipos de algoritmos para a resolução deste problema foram propostos na literatura:

- Algoritmos exatos, que sempre acham uma solução ótima (exemplo: Problema do Caixeiro Viajante);
- Heurísticas, que são algoritmos dependentes de problemas criados de acordo com suas especificações, com o resultado na maioria dos casos não sendo ideal (exemplo: *S-shape*, *Midpoint*, etc);
- Meta-heurísticas, que são algoritmos independentes, de problemas de alto nível, que fornecem um conjunto de diretrizes ou estratégias para encontrar uma solução aproximada para o problema (exemplo: Colmeia de Abelha, Colônia de Formigas, etc).

2.3.1 Algoritmos exatos

Alguns exemplos de algoritmos exatos já utilizados na resolução deste problema são, o algoritmo do Caixeiro Viajante, sem a restrição de capacidade, e o problema da rota de veículos com capacidade, onde os operadores podem precisar dividir os pedidos em 2 ou mais partes, dependendo da capacidade de carregamento de materiais (nas mãos ou nos dispositivos utilizados para transporte) (MASAE; GLOCK; GROSSE, 2019). Os autores ainda acrescentam que o que torna possível a resolução destes algoritmos é a utilização de matrizes com as distâncias entre os corredores e dimensões do armazém.

De acordo com Dijkstra e Roodbergen (2017), o problema de rotas em um armazém pode ser classificado como um caso especial do Problema do Caixeiro Viajante de Steiner, que em alguns *layouts* pode ser resolvido com convergência a resultados ótimos em um tempo polinomial. O objetivo desta variação do Problema do Caixeiro Viajante é encontrar a "viagem de Steiner" com a menor distância, onde cada nó "Não Steiner" é visitado pelo menos 1 vez (KULAK; SAHIN; TANER, 2012).

2.3.2 Heurísticas

Apesar da possibilidade de resolução do problema de rotas em um armazém com algoritmos exatos, na prática, é mais comum utilizar-se de heurísticas, sendo várias delas descritas na literatura (DIJKSTRA; ROODBERGEN, 2017).

De acordo com van Gils et al. (2018), métodos heurísticos já comprovaram resolver problemas complexos de planejamento em um tempo computacional e resultados relativamente adequados. Alguns métodos são comumente utilizados como os descritos por van Gils *et al.*, (2018):

Corredor-por corredor, no qual o operador visita todos os corredores por inteiro desde que tenha pelo menos 1 item a ser coletado neste corredor.

Método do retorno, quando o operador entra e sai dos corredores onde contem pelo menos 1 item a ser coletado pelo mesmo lado.

Método do ponto médio, o operador entra em um corredor onde precisa coletar pelo menos um item e o percorre apenas até a metade e retorna por onde entrou. Caso um corredor tenha itens em ambos os lados, o operador primeiro coleta os itens entrando por uma ponta, depois coleta os itens entrando pela outra ponta.

Método combinado, onde o operador pode tanto atravessar um corredor por inteiro como pode também fazer a saída deste corredor pelo mesmo ponto que entrou (misto de corredor por corredor e ponto médio)

Fernandes (2017) ainda acrescenta os métodos S-*shape*, *Largest gap* e otimizado, descrevendo-os a seguir:

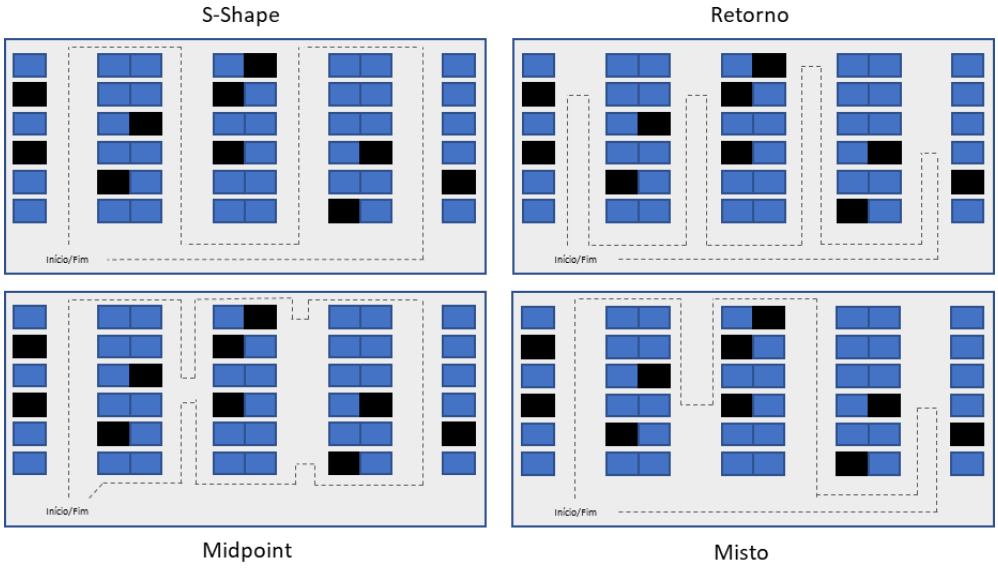
"*S-shape*: os operadores sequencialmente percorrem totalmente o corredor, desde que contenha pelo menos um item para ser recolhido. Se não existir artigos para recolher num determinado corredor, este não irá ser percorrido. Depois de efetuar a última recolha, o operador dirige-se ao depósito de recolha" (FERNANDES, 2017);

> *Largest Gap*: este método o operador também entra e sai dos corredores pelo mesmo lado quando recolhe um determinado artigo, à exceção do primeiro e do último corredor, que são percorridos na totalidade. O operador percorre um corredor utilizando a rota de retorno até se retirar a maior parte dos itens, e os restantes (caso existam) são recolhidos a partir do outro lado do corredor. A diferença é definida pela distância entre os dois picos adjacentes, ou a última escolha para o corredor de volta. Assim, os grandes intervalos nos corredores não são percorridos pelos operadores. (FERNANDES, 2017);

"Otimizado (*optimal*): um método otimizado combina a teoria de grafos e a programação dinâmica, que permita encontrar uma rota ótima, ou seja, a mais curta, eficaz e eficiente, no que se refere às distâncias percorridas e o tempo despendido no processo" (FERNANDES, 2017);

A Figura 1 exemplifica algumas destas dinâmicas de coleta de itens, para facilitar o entendimento.

Figura 1 - Exemplos de metodologias de coleta.



Fonte: Adaptado de Fernandes (2017).

No geral, heurísticas como as apresentadas acima, isoladamente ou em combinação, são amplamente implementadas para solucionar o problema da rota do operador devido a vantagem relacionada na eficiência computacional e facilidade de implementação(CORTÉS *et al.*, 2017).

2.3.3 Meta-heurísticas

Uma outra forma de determinar a melhor rota pode ser através da utilização de meta heurísticas, que são um conjunto de diretrizes para desenvolver um algoritmo de otimização para a rota dos operadores (KULAK; SAHIN; TANER, 2012). Cortés *et al.* (2017) ressaltam que estudos já foram desenvolvidos utilizando Algoritmos Genéticos, Colônia de Formigas e Enxame de Partículas para o problema da rota dos operadores.

Pinedo (2016) ainda acrescenta que é comum encontrarmos os algoritmos *"Simulated Annealing"*, *"Tabu Search"*, *"Genetic Algorithm"*, *"Ant Colony"*, *"Bee hive"* entre outros. Eles são classificados como algoritmos de busca local e são divididos em dois tipos: *Improvement type* e *contructive type*. Algoritmos de melhoria iniciam o processo com uma

solução pronta aleatória e buscam obter um resultado melhor manipulando as variáveis previamente selecionadas no início do processo. Um processo de busca local não garante uma solução ótima, eles sempre buscam uma solução melhor do que a atual. Cabe ao programador definir parâmetros de parada de acordo com o nível de melhoria alcançado com as iterações.

Ardjmand, Sanei Bajgiran e Youssef (2019) comentam que no trabalho de Matusiak *et al.* (2014) foi proposta a utilização de *Simulated Annealing* para estudar a formação de lotes com restrição de precedências e rotas de operadores. O método deles era composto por 2 sub algoritmos, sendo um algoritmo "*A-star*" para a solução das rotas e o *Simulated Annealing* para a solução dos lotes, e obtiveram resultados com erros menores do que 1,2% comparando com soluções ótimas contendo mais do que três pedidos.

Ardjmand, Sanei Bajgiran e Youssef (2019) também comentam sobre o trabalho de Chen *et al.* (2015) no qual foi formulado um modelo de programação inteira mista não linear para o problema de lotes, sequenciamentos e rotas de operador para minimizar o tempo de execução dos pedidos. Eles propuseram uma combinação de Algoritmo Genético e Colônia de Formigas para a solução dos problemas. De forma semelhante, Cheng *et al.* (2015) usaram um método hibrido de enxame de partículas com colônia de formigas para a resolução do lote de pedidos e das rotas de operador.

De acordo com Ardjmand, Sanei Bajgiran e Youssef (2019)a maioria dos trabalhos e estudos desenvolvidos são aplicados apenas no layout predefinido do armazém, o que não necessariamente limita a aplicabilidade dos algoritmos supracitados, mas pode afetar a generalização da aplicação deles. O autor diz também que muitos estudos investigando as operações de *picking* trabalham também a alocação dos itens ou o próprio layout da área de *picking*. Ele ainda comenta que, com relação a essas limitações, o trabalho deles busca formular uma solução que seja independente do *layout* do armazém. Neste trabalho, o método de melhoria escolhido também será focado no layout atual da área de *picking*, porém poderá ser replicado em outros CDs com diferentes *layouts* e estruturas.

2.3.3.1 Simulated Annealing

O *Simulated Annealing* é um algoritmo do tipo "heurística de refinamento", ou seja, começa com uma solução inicial qualquer e tenta obter um resultado melhor através da manipulação das variáveis utilizando procedimentos de busca local, o que não garante uma solução ótima. O que o método busca fazer é encontrar uma solução próxima que seja melhor do que a solução atual, e a cada iteração ele faz essa busca por possíveis soluções melhores,

podendo aceitar ou não uma possível solução baseada em seus critérios próprios, a fim de sair de possíveis soluções ótimas locais e ir em busca de soluções ótimas globais. (PINEDO, 2016).

Este algoritmo trabalha baseado em princípios da mecânica estatística. O algoritmo inicia com uma solução aleatória e em cada iteração uma solução vizinha é explorada. Se esta segunda solução é melhor em termos de qualidade, ela será aceita como a nova solução, caso contrário, seguindo uma probabilidade de aceitação de "resultados ruins", parâmetro baseado na "Temperatura", esta solução pode ser aceita como a solução atual (ARDJMAND; SANEI BAJGIRAN; YOUSSEF, 2019). Conforme o *Simulated Annealing* se desenvolve, a Temperatura vai diminuindo, e consequentemente, diminui-se a aceitação de soluções "piores". Este processo é chamado de resfriamento e pode ser feito de diversas formas, sendo duas as mais utilizadas, o resfriamento através de uma taxa fixa ou através de uma porcentagem até que uma temperatura baixa seja alcançada. Ardjmand, Sanei Bajgiran e Youssef (2019) ainda acrescentam que o *Simulated annealing* é bem sensível a este parâmetro e um resfriamento genérico pode não funcionar em qualquer problema, não permitindo a convergência para resultados aceitáveis, ou com longo período até se chegar a alguma solução.

2.3.3.2 Ant Colony optimization

O algoritmo de Colônia de Formiga combina técnicas de busca local, regras de despacho e outras técnicas em uma estrutura. Este algoritmo teve seus paradigmas inspirados pelo comportamento das colônias de formigas, que depositam feromônios ao longo do caminho sinalizando a rota até o destino para que outras formigas possam seguir. Este algoritmo funciona assumindo que uma colônia de formigas (Artificiais) constroem soluções, iterativamente, para o problema a ser estudado deixando trilhas de feromônios (artificiais também) indicando o caminho para resultados previamente encontrados. Estas formigas se comunicam apenas através da quantidade de feromônio depositado nas trilhas para cada solução durante a execução do programa.(PINEDO, 2016). O autor ainda acrescenta que devido à natureza do algoritmo, que pode encontrar soluções que são apenas ótimos locais, é comum implementar algum procedimento de busca local para aumentar a performance do algoritmo.

Li, Huang e Dai (2017) adicionam que o algoritmo imita o comportamento das formigas quando buscando por alimentos. As formigas liberam o feromônio na rota em que estão procurando para informar outras formigas de um caminho promissor (solução) de acordo com a quantidade depositada. Ele acrescenta que este algoritmo já foi adaptado para a solução do Problema do Caixeiro viajante com performance promissora.

Assim como mencionado por Pinedo (2016), Li, Huang e Dai (2017) considerou a combinação do algoritmo de Colônia de Formigas com Busca Local (*Ant Colony Optimization Local Search*), onde o algoritmo de busca local faz a varredura na vizinhança e melhora a solução, e somente após esta busca ele atualiza a quantidade de feromônio é atualizada para as soluções encontradas.

De Santis *et al.* (2018) declara que a aplicação deste algoritmo no contexto do p*icking* se daria na minimização da distância percorrida pelos operários na realização da coleta dos itens, porém em sua busca encontraram poucos estudos realizados em armazéns onde o *picking* tem a coleta manual.

2.3.3.3 Particle swarm optimization

De acordo com Lin *et al.* (2016) este algoritmo imita o movimento de uma população de partículas em busca da solução ótima. Na "enésima" iteração do algoritmo, cada particular "i" se move em direção à solução ótima pela atualização da sua velocidade e posição tomando como base a experiencia específica de outras partículas e da experiência global que obteve o melhor resultado até o momento.

Este autor ainda propôs a utilização de "experiências ruins" para atualizar as partículas ao redor, chamando-o de algoritmo melhorado de enxame de partículas. Ele sugere que isto pode prevenir buscas repetitivas em posições (soluções) ruins, portanto, a busca pela solução ótima é acelerada.

2.3.3.4 Busca Tabu

O Algoritmo de Busca Tabu opera de forma semelhante ao *Simulated Annealing*, sendo a diferença entre eles principalmente nos critérios de aceite e rejeição de solução. No *Simulated Annealing* estes critérios são baseados em processos probabilísticos, ao passo que na Busca Tabu são baseados em processos determinísticos. (PINEDO 2016)

Uma Busca Tabu genérica é implementada através de uma lista tabu, onde estão relacionados os movimentos aplicados em soluções anteriores. É esta lista que previne que o algoritmo vicie em alguma solução específica visitada anteriormente, evitando ciclagem de resultados e promovendo a busca em outras zonas dentro do espaço de soluções que ainda não foram exploradas (CORTÉS *et al.,* 2017) Este autor aplica este algoritmo para a proposta de rotas de deslocamento em um *picking* propondo a mudança da posição dos SKUs em um

armazém. Além disso, o algoritmo prevê movimentos de *Swap* e *Shift* para a obtenção de novas rotas, sendo o movimento de *swap* a permutação de 2 posições e o movimento de *Shift* é a remoção de uma SKU e reinserção em outra posição dentro da rota do operador. Masae, Glock e Grosse (2019) comentam que a busca tabu depende dessa troca (*Swap* e *Shift*) para explorar soluções vizinhas em busca de um resultado melhor.

Cortés *et al.* (2017) aplicou a busca tabu ao problema de rota do *picking* considerando restrições de disponibilidade de inventário e equipamentos de manejo de material variados, e quando comparado com Algoritmos Genéticos e *Simulated Annealing* encontraram resultados que eram até 9% melhores do que estes outros métodos.

2.3.3.5 Algoritmos genéticos

Algoritmos genéticos são mais generalistas do que, por exemplo, Busca Tabu e *Simulated Annealing*, que podem até ser vistos como casos específicos do Algoritmo Genético. Neste processo, os Indivíduos são avaliados de acordo com o valor associado com a função objetivo, um conjunto de indivíduos é chamado de geração, e cada iteração do processo iterativa gera uma nova geração de indivíduos, criada a partir da reprodução, em busca de promover novos resultados. Destes resultados, apenas os que promovem alguma melhoria são eleitos como aptos para reprodução, e os que não apresentam performance boa são eliminados. Os processos de nascimento, morte e reprodução que vão determinar a próxima geração podem ser complexos e, geralmente, dependem dos níveis de adequação da geração atual (PINEDO 2016). O autor ainda acrescenta que o que faz o Algoritmo Genético ser diferente dos outros mencionados é a questão da quantidade de novas soluções que são levadas para a próxima iteração. Com o *Simulated Annealing* e a Busca Tabu, apenas uma solução está sendo levada para a próxima iteração, e no algoritmo genético uma porção das soluções aplicáveis são levadas para a próxima repetição, baseada no tamanho da população. Portanto, estes dois outros métodos podem ser considerados como uma versão específica do Algoritmo Genético, onde a população tem tamanho 1.

Este tipo de algoritmo é popularmente aplicado como solução meta heurística na aplicação do problema de rotas do operador. Como em qualquer modelagem de sistema, é importante decidir como representar as variáveis, e para este problema é comum se comparar os genes de um cromossomo do algoritmo genético com a sequência de coleta dos pedidos (MASAE; GLOCK; GROSSE, 2019).

2.4 SIMULAÇÃO E VALIDAÇÃO DE RESULTADOS

A simulação computacional, pode ser classificada em três categorias. A primeira é representada pela simulação de Monte Carlo, que utiliza geradores de números aleatórios para simular sistemas físicos ou matemáticos, nos quais não se considera o tempo explicitamente como uma variável. A segunda engloba a Simulação Contínua, que se utiliza de equações diferenciais para o cálculo da mudança das variáveis de estado ao longo do tempo. Por fim, a Simulação de Eventos Discretos é utilizada para modelar sistemas que mudam em instantes discretos no tempo, a partir da ocorrência de eventos (CHWIF e MEDINA, 2015). A simulação permite testes do tipo *what if*, em que se verifica como a mudança de um ou mais parâmetros afetam o comportamento de um determinado sistema. Para isso, é necessário que o usuário por traz do modelo de simulação seja capaz de modelar corretamente, analisar e tomar as decisões para que sejam encontrados os resultados de saída. Essa ferramenta permite a realização de testes de cenários operacionais, não fornecendo, necessariamente, resultados ótimos. Contudo, é possível alcançar bons resultados se utilizada corretamente.

De forma análoga ao trabalho de Ardjmand, Sanei Bajgiran e Youssef (2019)**,** este trabalho busca também a utilização de ferramentas que possibilitem de forma ágil a avaliação da rota dos operadores em um determinado layout utilizando-se de modelos de simulação para obtenção de parâmetros importantes para a melhoria das rotas de *picking*.

De acordo com Dijkstra e Roodbergen (2017)**,** existem dois métodos principais para avaliação da distância percorrida nas rotas de coleta, através da criação de um modelo de simulação ou através do desenvolvimento de formulas baseadas em propriedades estatísticas dos modelos de rotas adotados. Os autores utilizaram os dois métodos para poder avaliar a alocação de locais de armazenamento em um armazém em busca da menor distância percorrida e encontraram poucas discrepâncias entre os resultados extraídos do modelo e dos métodos matemáticos desenvolvidos.

Em um trabalho bastante semelhante ao proposto aqui, Quader e Castillo-Villar (2018) comentam que a gestão das decisões em um armazém são fatores chave em busca da melhoria e eficiência da operação para redução de custos. Eles propuseram um modelo de simulação que pudesse testar diferentes heurísticas de rota e alocação de SKU’s que maximizassem a utilização dos operadores ou diminuísse o tempo de ciclo dos pedidos e que permitisse a avaliação de como o sistema se comportaria dinamicamente em uma zona de *picking* com vários corredores.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção se encontram informações específicas sobre as ferramentas que serão utilizadas a fim de construir uma abordagem para a solução do problema estudado.

3.1 SIMULATED ANNEALING

O *Annealing* (recozimento) é o processo de aquecimento de um sólido, por exemplo, até seu ponto de fusão, seguido de um gradual e vagaroso resfriamento, até que se alcance o enrijecimento novamente. Nesse processo, é essencial ter um resfriamento vagaroso para que os átomos encontrem tempo suficiente para se reorganizarem em estruturas uniformes com energia mínima. Caso o sólido seja resfriado de maneira brusca, os átomos formariam uma estrutura irregular e fraca, com alta energia por conta do esforço interno gasto (FAYYAZ *et al.*, 2018).

O método *Simulated Annealing* foi proposto inicialmente por Kirkpatrick et al. (1983) e Cerny (1985) em trabalhos independentes. Eles entenderam como um algoritmo desenvolvido anteriormente por Metropolis *et al* (1953), um modelo computacional para determinar a organização dos átomos de um sólido, que apresente energia mínima (resfriamento), poderia ser utilizado em problemas de otimização, em que a função objetivo a ser minimizada corresponde à energia dos estados do sólido. Para isso, realizaram uma analogia entre o sistema físico e o problema de otimização combinatória:

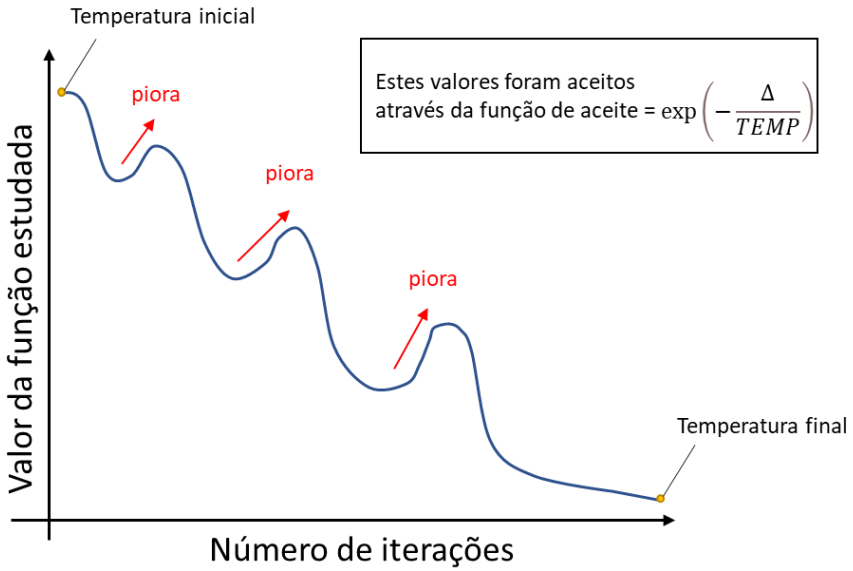
1. As soluções dos problemas de otimização equivalem aos estados do sistema físico;
2. O custo de uma solução é equivalente à energia de um estado;
3. A escolha de uma solução vizinha em um problema de otimização equivale à perturbação de um estado;
4. O ótimo global é equivalente ao estado fundamental;
5. O ótimo local é equivalente ao resfriamento rápido no sistema físico.

Partindo dessas analogias e da aplicação sucessiva do algoritmo *Metropolis* eles desenvolveram o método para problemas de otimização.

Esse método computacional oferece uma maneira de escapar de ótimos locais analisando a vizinhança da solução corrente e aceitando outras que tragam resultados melhores, além de também aceitar algumas soluções que piorem com uma certa probabilidade. A condição para aceitar ou rejeitar um movimento que aumente a função objetivo é determinada por uma sequência de números aleatórios, mas com uma probabilidade controlada (KIRKPATRICK et al, 1983).

Dessa forma, o *Simulated Annealing* é um algoritmo para otimização utilizado para encontrar o ponto ótimo global em um vasto espaço de busca através da simulação do processo de resfriamento, diminuindo gradualmente a temperatura do sistema até a convergência em um estado estável (FAYYAZ *et al.*, 2018). A Figura 2 indica o comportamento esperado ao se adotar este algoritmo. O método é iniciado com um valor alto de temperatura, permitindo que alcance algum valor mínimo para a função estudada. Como o método adota algumas medidas para aceitar "resultados piores", acaba permitindo que se encontre valores menores por não ficar preso em mínimos locais, dado tempo ou número de iterações suficientes. Conforme a temperatura diminui, torna-se menos e menos favorável a aceitações destes "resultados piores" (FAYYAZ *et al.*, 2018).

Figura 2 - Comportamento esperado ao aceitar pioras.



Fonte: Adaptado de Fayyaz *et al.* (2018).

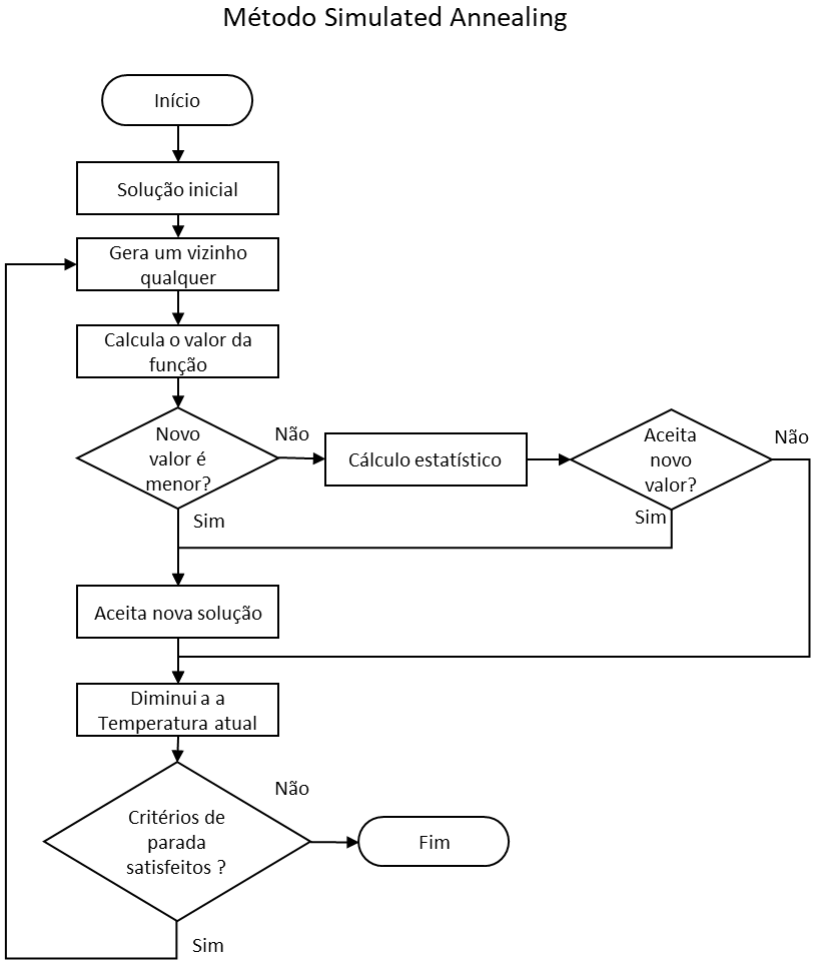
A probabilidade de aceitar essa piora é chamada de função de aceite ($g(\delta,T)$) e é geralmente representada por exp($-\delta/T$), onde δ é a diferença entre as soluções e T é um parâmetro de controle correspondente à temperatura, análogo ao processo de recozimento . Com essa função, as mudanças que implicam em pequenos aumentos da função objetivo são mais prováveis de serem aceitas do que as que gerem grandes aumentos. E em relação à temperatura, quanto mais alta, mais movimentos são aceitos. Já quanto mais se aproxima de zero, muitos movimentos que aumentem o resultado serão rejeitados (KIRKPATRICK et al, 1993).

Dessa maneira, caso $\delta=f(j)-f(i)$ for menor que zero, a solução j será aceita como nova solução corrente. Do contrário, só será aceita caso a função de aceite seja maior do que um valor aleatório calculado ($g(\delta,T)>random(0,1)$).

O método computacional inicia com um valor de temperatura relativamente alto para evitar que o algoritmo fique preso a um mínimo local precipitadamente. Esse parâmetro é gradualmente diminuído e, para cada valor, são realizadas diversas tentativas de alcançar melhores soluções nas vizinhanças da solução corrente. O algoritmo para quando algum critério de parada for satisfeito (pode ser estabelecido quando um valor da função objetivo, a cada mudança, é praticamente o mesmo para um determinado número de temperaturas consecutivas).

A Figura 3 mostra um fluxograma exemplificando o funcionamento do método *Simulated Annealing* de acordo com Souza (2008).

Figura 3 - Fluxograma do *Simulated Annealing*.



Fonte: Adaptado de Souza (2008).

O autor acrescenta que este algoritmo adota os seguintes parâmetros:

1. Valor inicial da temperatura;
2. Função da temperatura T(t), que determina como a temperatura deve mudar ("*Cooling Schedule*"). Parâmetro $\propto$;
3. Critério de parada do algoritmo.

Como critérios de parada pode-se adotar a temperatura final, quando a temperatura corrente atinge o valor da temperatura final o algoritmo se encerra. Porém, dependendo da taxa de resfriamento isto pode levar muito mais tempo, além de exigir muito mais capacidade de processamento. É comum adotar também um limite do número de iterações para cada temperatura, e assim que este valor é alcançado o algoritmo aceita o valor da função, independentemente de ter efetuado o processamento e resfriamento total da função, conforme desejado.

3.2 SIMULAÇÃO

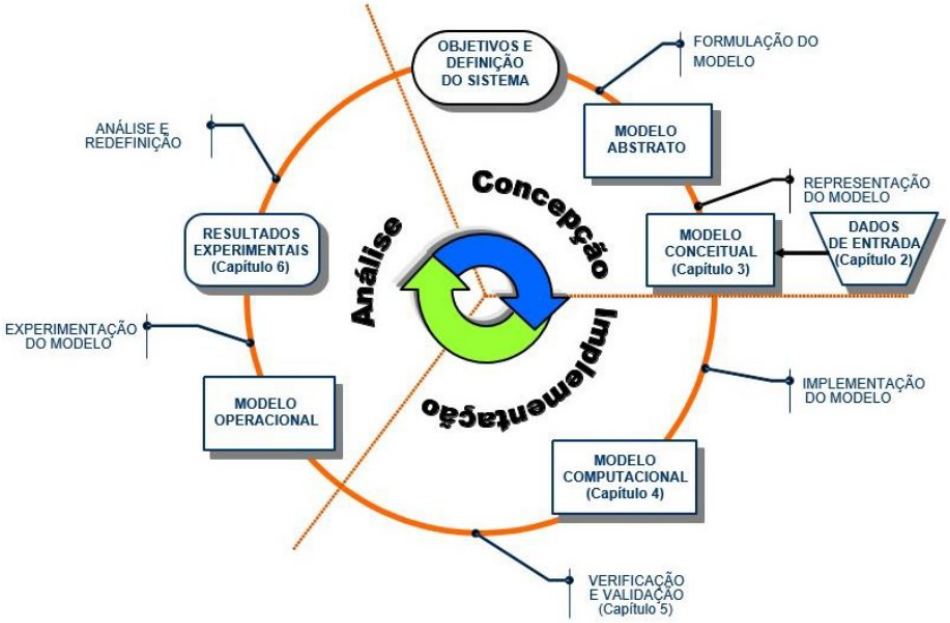
Simulação Computacional pode ser descrita como uma maneira de analisar as possibilidades de um sistema pela construção de um modelo matemático baseado em computador. Para Silva et al. (1998) a simulação tem como objetivo reproduzir um sistema por um modelo, para testar hipótese de valores com variáveis controladas. McLean e Leong (2001) descrevem que os modelos de simulação são construídos para dar suporte às decisões, estimar investimentos em novas tecnologias, expansão de capacidade de produção, gerenciamento de materiais, gerenciamento de recursos humanos, entre outros.

Essa ferramenta permite o teste de cenários operacionais, não fornecendo, necessariamente, resultados ótimos. Contudo, é possível alcançar bons resultados se utilizada de maneira inteligente, ou seja, a simulação permite testes do tipo *what if*, em que se verifica como a mudança de um ou mais parâmetros afeta o comportamento do sistema (CHWIF e MEDINA, 2015). Para isso, é necessário que o usuário por trás do modelo de simulação seja capaz de analisar e tomar as melhores decisões para que sejam encontrados bons resultados de saída (SELLITTO, 2009).

3.2.1 Metodologia de modelagem e simulação

Para que um modelo seja adequado ao sistema que se deseja simular, é comum que algumas etapas conceituais sejam seguidas a fim de se atingir um nível de confiabilidade alto no modelo construído. Existem algumas metodologias mais comumente seguidas pelos programadores, como por exemplo esta metodologia dividida em três grandes etapas: concepção ou formulação do modelo, implementação do modelo e análise dos resultados (Figura 4).

Figura 4 - Metodologia de modelagem e simulação.



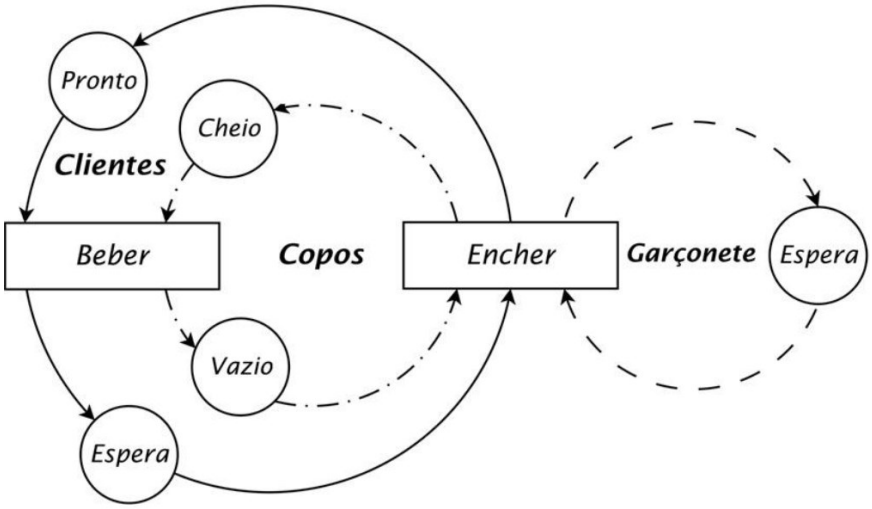
Fonte: Chwif e Medina (2015).

O processo inicia-se pela fase de concepção, em que é necessário entender o sistema, o problema em questão e seus objetivos. Nessa etapa, também são coletados os dados de entrada, lembrando que existe grande importância em se obter dados adequados para alimentar o modelo. Caso elementos “errados” sejam utilizados na simulação, dados “errados” sairão como resultado (CHWIF e MEDINA, 2015).

Segundo Shannon (1998), quanto mais esforços forem direcionados à fase inicial do projeto, o que inclui da definição do problema à criação do modelo computacional, mais rápida será a construção do modelo de simulação e a finalização do projeto.

Ao final da etapa de concepção, o analista, que tem o modelo abstrato em mente, deve transformar este conhecimento tácito em algo mais explícito, desenvolvendo um modelo conceitual, uma representação do sistema em estudo através de diagramas, fluxogramas, entre outras opções. Robinson (2004) comenta que o modelo conceitual é uma descrição de um modelo de simulação a ser desenvolvido, apresentando os objetivos, parâmetros de entrada e saída, componentes, premissas e simplificações do sistema. Essa etapa do processo de um projeto de simulação é considerada por Law e Kelton (2000) o aspecto mais importante de uma simulação. Para Chwif e Medina (2015), uma das técnicas de representação desse modelo de mais fácil entendimento é o *Activity Cycle Diagram* (ACD). Um exemplo dele está indicado na Figura 5.

Figura 5 – Exemplo de um ACD.



Fonte: Chwif e Medina (2015).

Somente após a definição e finalização do modelo conceitual, muitas vezes discutido entre o analista e o gerente do processo e algumas vezes pode incluir também os operadores para ser possível a inclusão de *insights* e processos que ocorram e possam não estar documentados, inicia-se a etapa de implementação. O modelo conceitual é convertido em um modelo computacional, e deve ser comparado com o conceitual para avaliar se a sua operação atende realmente aquilo que foi estabelecido na etapa de concepção. Para essa validação, alguns resultados precisam ser gerados e é necessário verificar se o modelo é uma representação precisa da realidade.

Nesta etapa o modelo computacional torna-se o modelo operacional, surgindo os resultados da simulação, e permitindo que conclusões e recomendações possam ser geradas. (CHWIF e MEDINA, 2015).

Para Harrell et al. (2002) o estudo de simulação é um processo interativo, em que as atividades são refinadas e algumas vezes redefinidas em cada interação. Algumas etapas do estudo também podem ser conduzidas em paralelo com outra etapa, ou seja, não existe a necessidade de completar uma etapa predecessora para seguir com o estudo.

3.2.2 Processos de Verificação e Validação

Por mais que esses muitas vezes sejam confundidos, não possuem a mesma função, uma vez que a validação está relacionada ao modelo conceitual e a verificação ao modelo computacional (CHWIF e MEDINA, 2015) e são partes essenciais na modelagem e simulação.

A validação conceitual está relacionada ao modelo conceitual, quando se avaliam as considerações feitas, o nível de detalhamento e o escopo, verificando se eles representam de forma adequada o sistema. Assim, a validação é o processo de certificar que o modelo reflete a operação do sistema real de maneira que dê andamento ao problema definido (BATEMAN et al., 2013). Já a verificação computacional está relacionada ao modelo computacional, como demonstra a Figura 6. Essa pode ser entendida como o processo de retirada de bugs do modelo ou elementos que estão causando o mau funcionamento.

Figura 6 – representação de um modelo computacional.

Fonte: Chwif e Medina (2015).

A aplicação dos processos de validação e verificação, portanto auxiliam a conferir ao modelo em construção um nível aceitável de confiabilidade, permitindo uma melhor representação do sistema em estudo.

3.2.3 Implementação computacional da simulação

Após a modelagem conceitual, é iniciada a etapa de implementação do modelo computacional, que por sua vez pode ser complexa e depende muito do conhecimento técnico de simulação ou programação do analista por trás da modelagem. Ele deve decidir em qual plataforma o modelo será executado e, para isso, é possível utilizar linguagens de simulação (GPSS, SLAM, SIMAM e etc.), simuladores (ARENA, SIMUL8, Extend, Plant Simulation, dentre outros) ou linguagens de programação (Python, Basics, C, Fortran, etc) (CHWIF e MEDINA, 2015).

Segundo Chwif e Medina (2015), o modelo de simulação é uma ferramenta poderosa de experimentos estatísticos utilizados no processo de análise do comportamento do sistema. Com

isso, essa ferramenta pode ser utilizada para analisar o comportamento de diversos cenários e propor melhorias para problemas do cenário real.

3.2.4 Simulação em Sistemas logísticos

De acordo com Ballou (2006), os modelos de simulação computacional são adequados para avaliação das operações logísticas dentro de uma empresa, uma vez que possibilitam testes de novos cenários de maneira rápida e de baixo custo em relação à realização de mudanças no cenário real. Dessa maneira, é possível verificar o impacto de cada mudança nos custos operacionais e no nível de serviço oferecido pela empresa.

Para Tozi et al. (2011), a modelagem computacional é o elemento essencial para apoiar o entendimento e a análise de sistemas complexos, principalmente quando se pretende analisar cadeias logísticas desde uma perspectiva tática e/ou estratégica.

Segundo Oliveira (2004), a simulação de uma cadeia de suprimentos (CS) deve respeitar a lógica de coordenação da simulação e a representação fiel das regras de negócio da empresa e da própria CS. Dessa forma, é possível analisar melhorias para a logística de armazéns e sistemas de distribuição, obtendo respostas de melhores *layouts*, fluxo de materiais, alocação de recursos e pessoas.

A simulação possibilita o estudo da operação por intermédio da análise de diversos parâmetros de forma simultânea (MEIRELLES et al., 2009). Dessa maneira, ela pode ser empregada nos estudos de layout para estimar os parâmetros ligados a tarefas como:

1. Elaborar uma série de arranjos otimizados, originados pelo uso de rotinas clássicas de *layouts*;
2. Detectar gargalos em estruturas de arranjos físicos e pontos de melhoria;
3. Comparar diferentes composições de layout em relação a parâmetros operacionais.

Assim sendo, os modelos de simulação buscam representar os sistemas reais com o máximo de semelhança possível.

3.2.5 Vantagens da simulação

Prado (2010) afirma que a simulação possibilita a imitação da operação de um sistema real, podendo ser visto o funcionamento dos modelos elaborados em uma tela, como se estivesse vendo um filme. Dessa forma, essa ferramenta é empregada para estabelecer e averiguar um

procedimento real em menor tempo e com menor custo, propiciando um estudo acurado de acontecimentos passados, presentes e futuros (LAW e KELTON, 2000).

Segundo Pedgen, Shanon e Sadowsky (1990), na simulação podem ser experimentadas hipóteses para confirmar como e por que certos fatos ocorrem. Assim sendo, possibilita que se mensure o desempenho de um sistema real em relação a situações operacionais distintas (LAW, 2007). Dessa forma, a simulação computacional consegue evitar decisões desacertadas que possam pôr em risco o funcionamento da empresa ou resultem em investimentos inadequados (KRAJEWSKI e RITZMAN, 2001).

Segundo Law e Kelton (2000) e Harrel et al. (2002), a simulação apresenta como benefícios:

1. Desenvolvimento de modelos adaptáveis à realidade, testando diferentes cenários e possibilidades de operação de um sistema, sem comprometimento de recursos;
2. Capacidade de simulação de sistemas complexos (dotados de elementos estocásticos), os quais não são adequadamente descritos por modelos matemáticos determinísticos;
3. Avaliação da distribuição dos recursos disponíveis, alocando-os de forma adequada ao processo e garantindo níveis elevados de produção;
4. Melhor controle sobre as condições experimentais em comparação prática no sistema real;
5. Análise de longos períodos de tempo de uma opção em um tempo reduzido de simulação;
6. Determinação de gargalos existente no sistema e estudos relacionados à otimização do processo.

Dessa maneira, a simulação é um meio poderoso para amparar a decisão, possibilitando que sejam obtidas respostas potencialmente boas para os problemas do sistema real (CORRÊA, GIANESI e CAON (2001). No entanto, para que se obtenha resultados aplicáveis, em que seja possível representar a realidade de um sistema de forma confiável, é necessário aplicar as técnicas de modelagem e de simulação bem definidas.

4 METODOLOGIA

Este trabalho pode ser classificado, quanto à natureza como uma pesquisa explanatória, que consiste em verificar quais fatores podem contribuir ou determinar o acontecimento de certos eventos, estudando a realidade e explicando a razão das ocorrencias. (GIL, 2010).

Este trabalho apresenta a aplicação de um modelo para o setor de picking com os dados reais de funcionamento do centro de distribuição estudado, além de analisar comparativamente os resultados de eficiência obtidos com a rota de coleta atual e a proposta pelo algoritmo de estudo da rota de coleta realizada pelos operadores.

O método selecionado é a modelagem e simulação em uma abordagem quantitativa. Hollocks (1992) descreve a simulação computacional como uma técnica de pesquisa operacional que envolve a criação de um programa computacional representando alguma parte do mundo real, de forma que experimentos no modelo original predizem o que acontecerá na realidade. Além disso, a otimização combinatória, uma subárea da pesquisa operacional foi explorada a fim de chegar a resultados com menores custos de deslocamento. O algoritmo selecionado para a exploração das possibilidades combinatórias é o Simulated Annealing.

Em relação aos procedimentos técnicos, as formas de pesquisa utilizadas são:

1. Pesquisa bibliográfica, a fim de se obter informações de estudos já publicados por outros autores acerca deste tema, levando em consideração as soluções propostas e as ferramentas utilizadas para a solução dos problemas em centros de distribuição;
2. Análise documental, utilizando dados obtidos através de planilhas eletrônicas, plantas do local e outros documentos relevantes ao processo;
3. Entrevistas com funcionários, visando o entendimento da operação, demandas e resultados a ser disponibilizado para a empresa.

As informações coletadas servem para alimentar os modelos de simulação e o algoritmo de estudo das rotas, a fim de se obter uma nova sequência de coleta dos itens dos pedidos percorrendo uma distância menor.

5 DESENVOLVIMENTO EXPERIMENTAL

As informações referentes aos métodos escolhidos, informações sobre a operação do *picking* neste armazém e formas de integração dos modelos desenvolvidos estão descritas nesta seção.

5.1 ESTUDO DAS ROTAS DE PICKING

No CD estudado neste trabalho, o armazém da empresa é dividido entre a área de armazenagem, áreas de administração, áreas de operação, e o setor de *Picking*, onde os operadores, que são terceirizados através de uma empresa de logística, recebem os pedidos e montam estes pedidos em *pallets* que serão alocados em caminhões, de acordo com as informações liberadas pelo WMS. A operação ocorre no período noturno, a fim de que no início da manhã seguinte os caminhões já carregados possam seguir para as suas rotas de entrega.

Essas rotas são montadas por um *software* que roteiriza os caminhões de acordo com os pedidos recebidos até as 17:00 do dia em questão, e, de acordo com essas informações, o WMS faz então a programação de coleta dos itens para paletização. No geral, a roteirização dos caminhões e a programação vinda do sistema de gerenciamento do armazém necessitam de algumas horas para disponibilizar o resultado. Os operadores iniciam seu turno às 21:00 e realizam reuniões e outras tarefas não relacionadas à movimentação de material por até uma hora, iniciando as montagens de pedidos a partir das 22:00 com os roteiros fornecidos pelo WMS. Outra informação relevante é que a este setor de *picking* utiliza o modo de operação discreta.

A área de picking deste CD dispõe de 240 posições pallet, todas a nível do chão. Além disso, existem 3 racks que possuem 12 posições para produtos, separados em 4 níveis diferentes. Estes racks armazenam produtos em menor em menor quantidade do que as posições pallet e servem para produtos com baixo volume de vendas. Para a montagem dos pedidos, os operadores coletam um pallet vazio na mesma área de entrega dos pallets¸ localizada na parte inferior. Neste local há a conferencia dos pedidos já montados, confirmando que os produtos estão em quantidades corretas de acordo com o requisitado, e posteriormente são colocados no caminhão que fará a entrega.

5.2 ESTRATÉGIA DE COLETA DE DADOS

As informações de operação foram obtidas através de visitas a mais de um CD dessa empresa para verificar especificidades da operação de *picking*, quando os operadores estão

fazendo a montagem dos pedidos. Além das visitas, a empresa disponibilizou planilhas com dados sobre os pedidos, os *layouts* e dimensões da área de *picking*. A Figura 7 mostra uma parte da curva ABC com a venda dos produtos em um período separados em 3 níveis.

Figura 7 - Curva ABC disponibilizada pela empresa.

Curva	Item	Descrição	Qnt	Pareto
A	988	PRODUTO A1	28638	27.86%
A	2538	PRODUTO A2	17542	44.92%
A	1695	PRODUTO A3	9318	53.99%
A	2546	PRODUTO A4	14347	67.94%
B	1743	PRODUTO B1	3769	71.61%
B	978	PRODUTO B2	9737	81.08%
B	2532	PRODUTO B3	4300	85.26%
B	2544	PRODUTO B4	2269	87.47%
B	18677	PRODUTO B5	1010	88.45%
B	982	PRODUTO B6	1195	89.61%
C	3733	PRODUTO C1	3304	92.83%
C	1388	PRODUTO C2	415	93.23%
C	2585	PRODUTO C3	826	94.04%
C	2231	PRODUTO C4	807	94.82%
C	14074	PRODUTO C5	324	95.14%
C	2237	PRODUTO C6	339	95.47%
C	10537	PRODUTO C7	393	95.85%

Fonte: Autor.

A sequência de coleta realizada pelos operadores, alvo deste trabalho, pode ser visualizada na Figura 8. Ela representa um trecho da planilha com todos os *pallets*, a hora de início e fim de cada montagem, quais produtos, quantos fardos ou caixas de cada item e qual operador realizou a separação de cada *pallet*.

Figura 8 - Planilha de montagem dos *pallets*.

Data	Palete	UsuarioNome	CodigoItem	Item	Qtd	Hora Inicio	Hora Fim
03/03/2020	P01_M_01_1/42	Ajudante 1	2585	PRODUTO A3	1	23:02:49	23:03:36
03/03/2020	P01_M_01_1/42	Ajudante 1	11593	PRODUTO B1	1	23:03:37	23:04:06
03/03/2020	P01_M_01_1/42	Ajudante 1	15135	PRODUTO A1	1	23:04:06	23:04:48
03/03/2020	P03_A_03_1/42	Ajudante 2	838	PRODUTO A1	3	23:05:06	23:08:02
03/03/2020	P03_A_03_1/42	Ajudante 2	8037	PRODUTO B1	1	23:08:03	23:08:44
03/03/2020	P02_A_02_1/42	Ajudante 1	18836	PRODUTO C1	3	23:11:35	23:13:30
03/03/2020	P02_A_02_1/42	Ajudante 1	17808	PRODUTO B6	1	23:13:31	23:14:10
03/03/2020	P02_A_02_1/42	Ajudante 1	17757	PRODUTO A1	3	23:14:10	23:14:46
03/03/2020	P02_A_02_1/42	Ajudante 1	18807	PRODUTO C2	3	23:14:46	23:15:59
03/03/2020	P02_A_02_1/42	Ajudante 1	12951	PRODUTO B2	1	23:15:59	23:16:26
03/03/2020	P02_A_02_1/42	Ajudante 1	3735	PRODUTO B4	1	23:16:27	23:17:05
03/03/2020	P02_A_02_1/42	Ajudante 1	19166	PRODUTO A2	1	23:17:06	23:17:36
03/03/2020	P02_A_02_1/42	Ajudante 1	2243	PRODUTO C5	1	23:17:36	23:17:57
03/03/2020	P02_A_02_1/42	Ajudante 1	16011	PRODUTO A4	1	23:17:58	23:19:43
03/03/2020	P02_A_02_1/42	Ajudante 1	15971	PRODUTO A3	1	23:19:43	23:19:46
03/03/2020	P03_M_03_1/42	Ajudante 2	2349	PRODUTO A2	24	23:22:48	23:29:28
03/03/2020	P03_M_03_1/42	Ajudante 2	7703	PRODUTO B3	2	23:37:39	23:37:55
03/03/2020	P03_M_03_1/42	Ajudante 2	504	PRODUTO A1	19	23:29:28	23:35:49
03/03/2020	P03_M_03_1/42	Ajudante 2	4409	PRODUTO C1	1	23:35:49	23:35:58
03/03/2020	P03_M_03_1/42	Ajudante 2	2350	PRODUTO C6	1	23:35:59	23:36:09

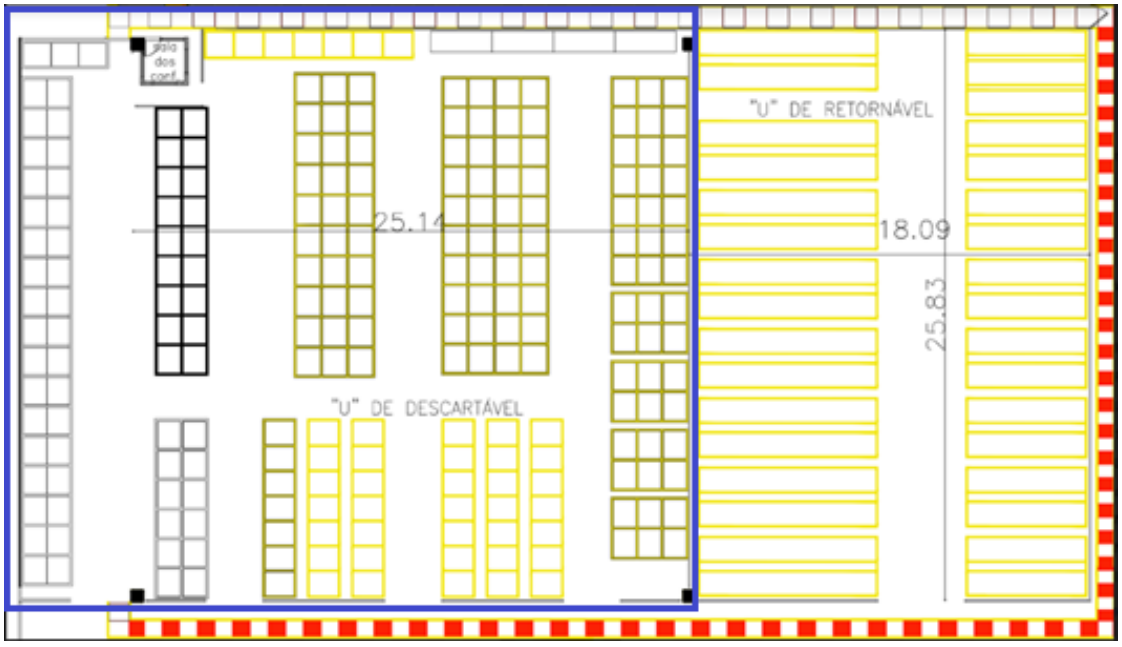
Fonte: Autor.

Tendo em vista as dimensões das áreas de *picking* e a comum falta de documentação detalhada, a obtenção de informações precisas, como, por exemplo, as distâncias entre as

posições dos SKUs, se torna uma tarefa árdua. Uma alternativa seria o uso de GPS (*Global Positioning System*) para a coleta destas informações. Porém, o investimento em sistemas para captação da movimentação via GPS dos operadores pode se tornar inviável devido à falta de infraestrutura. Estes sistemas comumente necessitam de acesso a uma rede de internet confiável e segura para enviar os dados pra um servidor, além da necessidade de visibilidade dos aparelhos GPS pela rede de satélites. Como pode ocorrer de alguns CDs não terem uma cobertura do sistema de posicionamento global eficiente, seja por localização ou por outras interferências físicas, a implementação deste sistema para a coleta de dados das rotas pode se tornar inviável.

A Figura 9 é um dos materiais que foi disponibilizado pela empresa para a modelagem do *picking*. O setor destacado em azul é o objeto de estudo utilizado neste trabalho, onde se localizam os produtos descartáveis. Foi necessário que uma pessoa fosse ao local medir a distância dos corredores entre as ilhas e outras cotas que não estavam presentes neste arquivo para atualizar e complementar o layout.

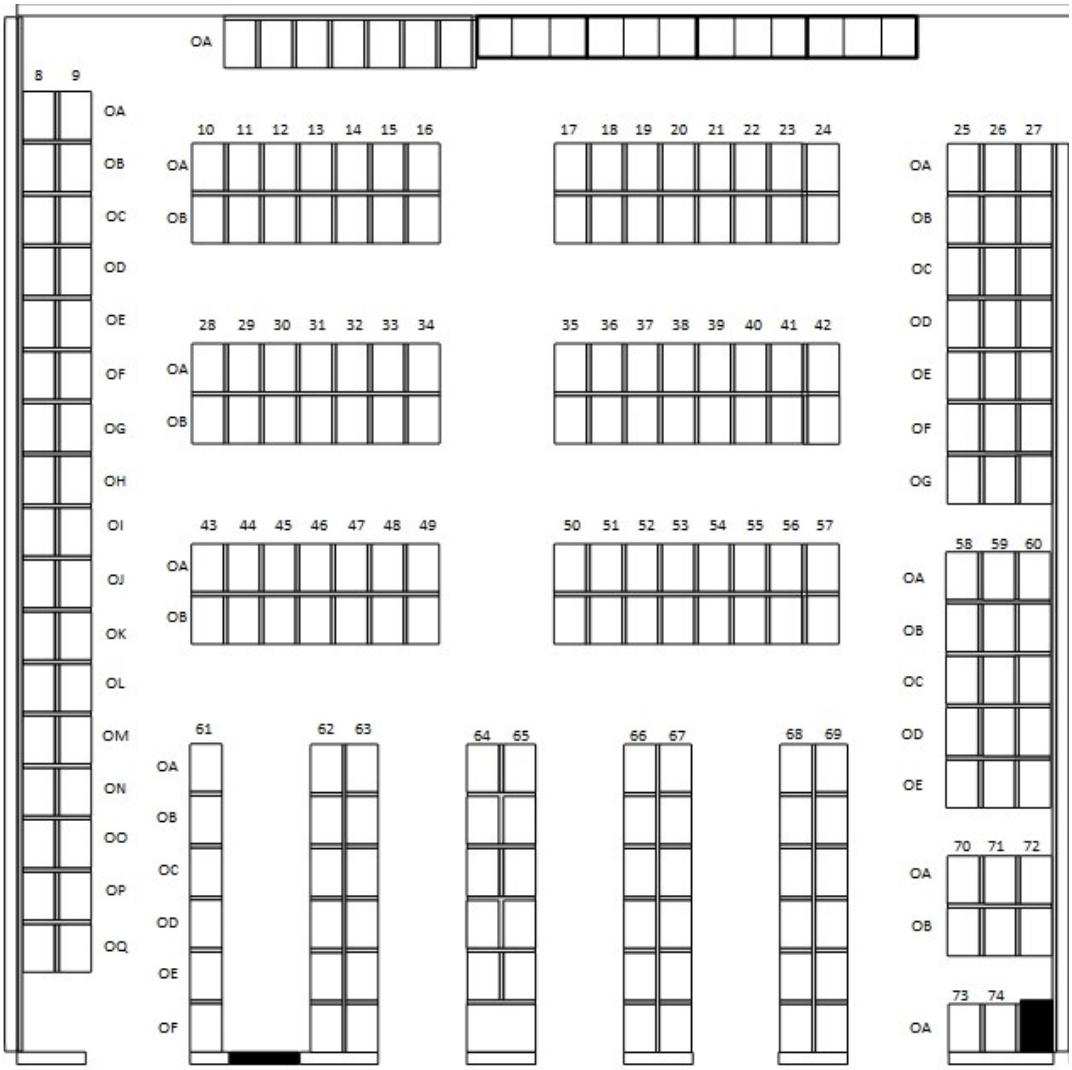
Figura 9 - Layout com medidas fornecido pela empresa.



Fonte: Autor.

Outro ponto a se destacar é que o material fornecido não estava atualizado, por exemplo, a disposição do corredor à direita já não era a mesma indicada nesta figura, nem a existência de posições *pallet* na parte superior esquerda da imagem. O novo posicionamento pode ser visto na Figura 10.

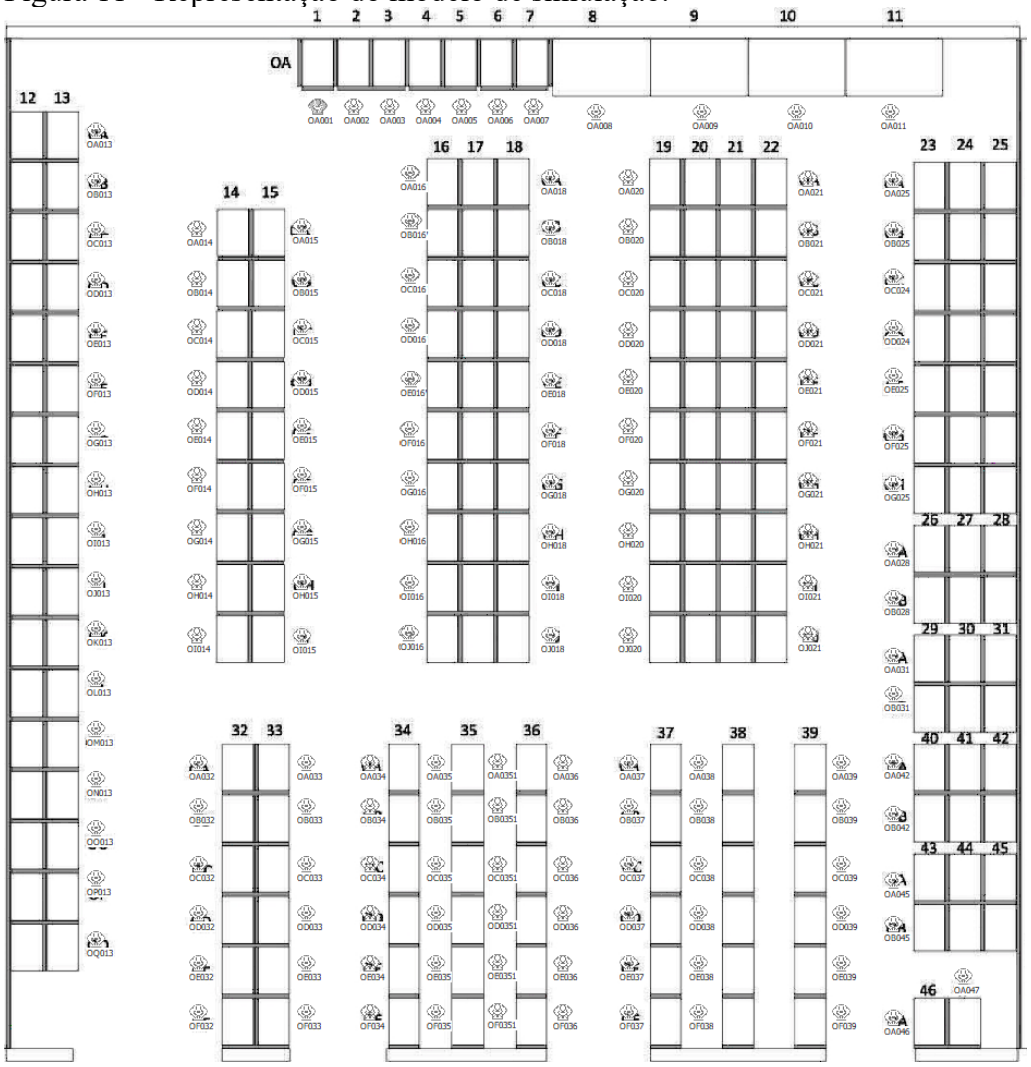
Figura 10 – Representação do layout corrigido utilizado para as análises.



Fonte: Autor.

A partir deste *layout* o modelo de simulação foi desenvolvido (Figura 11) para que se torne possível, de maneira precisa, a aquisição de informações que seriam impossíveis de se coletar manualmente, como por exemplo as distâncias entre uma posição de coleta A e todas as outras posições existentes e para a obtenção dos dados de distância percorrida pelos operadores.

Figura 11 - Representação do modelo de simulação.



Fonte: Autor.

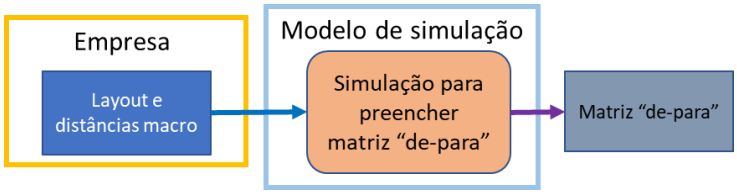
Utilizando-se destas planilhas e *layouts* foi possível então organizar as informações necessárias para alimentar o modelo de simulação a fim de se obter as informações desejadas.

5.3 MODELO DE SIMULAÇÃO

O modelo proposto para este trabalho foi desenvolvido utilizando o *software Plant Simulation*, da desenvolvedora Siemens, que fornece ferramentas para diversas áreas como análises estruturais, simulação contínua para fluidos, desenvolvimento de equipamentos e possui esta ferramenta de simulação de eventos discretos. Para este estudo se torna necessária a modelagem do sistema e operação de um CD, porém a parte de eventos discretos, a qual utiliza dados de distribuições estatísticas para inserir aleatoriedades no sistema, não será utilizada. Este modelo deve realizar duas funções, sendo a primeira uma execução para a criação de uma tabela "de-para" listando a distância de todas as posições entre si (Figura 12). Esta informação é

extremamente importante para a execução do algoritmo de estudo das rotas e de grande dificuldade de obtenção no cenário real, devido à demora para a obtenção destes valores e a inviabilidade de atribuir alguém para realização desta tarefa em um setor da empresa que tem atividades distribuídas quase que durante as 24 horas do dia.

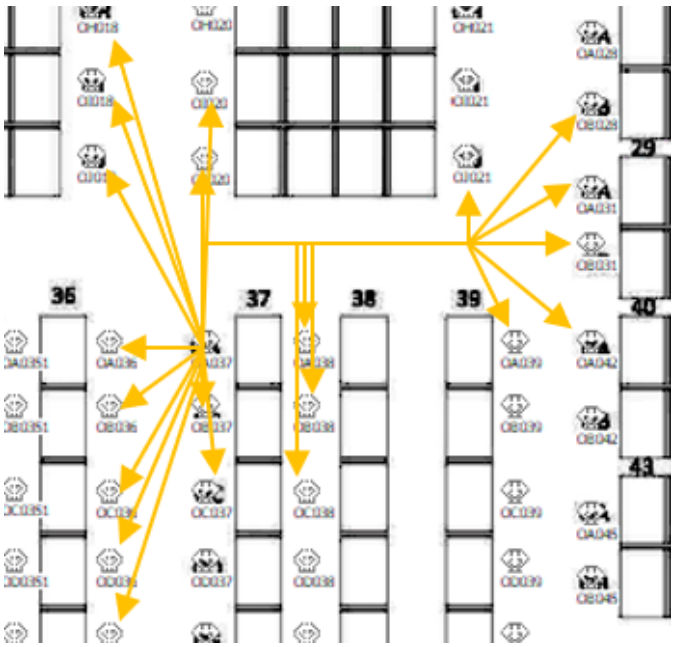
Figura 12 - Representação da primeira função do modelo de simulação.



Fonte: Autor.

Para a realização desta tarefa, o programador deve utilizar o *Layout* da empresa com as escalas, corredores e posicionamentos reais, criar o modelo baseado nestas informações e incluir os pontos de coleta de cada posição *pallet*. Com o modelo construído, é necessário executar o comando para que um operador percorra de uma posição A para todas as outras posições anotando a distância em uma planilha a tabela "de-para" para que ela possa ser utilizada futuramente pelo algoritmo de estudo da rota de coleta. A Figura 13 apresenta um exemplo simplificado do funcionamento deste código, de como o operador se movimenta com as instruções inseridas nele.

Figura 13 - Exemplo de coleta de distância.



Fonte: Autor.

A programação de um operador para percorrer e coletar estas distâncias foi realizada no *Plant Simulation*, por meio de "Métodos" que utilizam uma linguagem própria, o "*SimTalk* 2.0" (Figura 14). Este método em específico percorre a tabela "Distancias", presente no modelo de simulação fazendo a leitura das células da primeira linha, indicando a posição atual que o operador deve ir e as próximas posições. Ao realizar o percurso, a distância percorrida em cada trecho é anotada na tabela, nas células referentes ao deslocamento. O operador realiza o deslocamento entre as posições até não haver mais nenhuma célula em branco desta tabela.

Figura 14 – Exemplo de um método com a linguagem *SimTalk 2*.0.

```
for i:= 1 to distancias.Ydim-1
    for j:=1 to distancias.Xdim -1
        A := str_to_obj(distancias[1,i+1])
        B := str_to_obj(distancias [j+1,1])
        @.goto(A)
        waituntil @.Location = A
        w:=@.StatTraveledDistance
        @.goto(B)
        waituntil @.Location = B
        @.goto(A)
        waituntil @.Location = A
        k:= @.StatTraveledDistance-w
        distancias[j+1,i+1] := to_str(k/2)
        distancia[j+1,i+1] := to_str(k/2)
    next
next
```

Fonte: Autor.

Um trecho da planilha preenchida utilizando o modelo de simulação está representado pela Figura 15, onde na primeira coluna estão os nomes de todas as posições e na primeira linha está o "endereço" para onde o operador deve se deslocar.

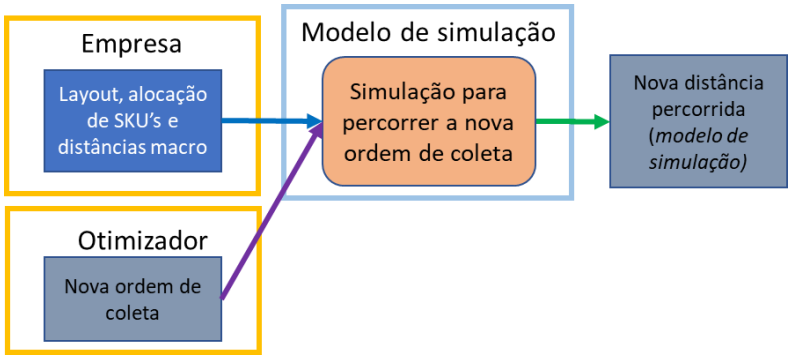
Figura 15 - Exemplo de tabela "de-para".

	string 1	string 2	string 3	string 4	string 5
1		.CDD.Norte.OA001	.CDD.Norte.OA002	.CDD.Norte.OA003	.CDD.Norte.OA004
2	OA001	0	1.060546875	2.12109375	3.181640625
3	OA002	1.060546875	0	1.060546875	2.12109375
4	OA003	2.12109375	1.060546875	0	1.060546875
5	OA004	3.181640625	2.12109375	1.060546875	0
6	OA005	4.2421875	3.181640625	2.12109375	1.060546875
7	OA006	5.302734375	4.2421875	3.181640625	2.12109375
8	OA007	6.4638671875	5.4033203125	4.3427734375	3.2822265625
9	OA009	4.04271457628784	5.08155279817584	6.1278328864355	7.17830148960638
10	OA010	1.81452171434648	2.5190680972737	3.41298087508767	4.38185944284487
11	OA011	1.57759100310795	1.81452171434648	2.51906809725915	3.41298087508767

Fonte: Autor.

Após a execução desta função, com a tabela “de-para” inteiramente preenchida, é possível alimentar estes dados no algoritmo de estudo das rotas e extrair dados necessários para a realização da segunda função do modelo de simulação (Figura 16). Nesta etapa, além dos dados utilizados anteriormente, novas informações, como, por exemplo, a alocação dos produtos nas posições e a nova ordem de coleta, disponibilizadas pelo algoritmo de estudo das rotas de coleta, permitem que o modelo seja utilizado para avaliar a distância total percorrida pelos operadores para a realização de todos os pedidos do período estudado. Com este valor disponibilizado, é possível confrontar este dado com o resultado obtido através do algoritmo de estudo das rotas a fim de se avaliar a eficácia de ambas as ferramentas.

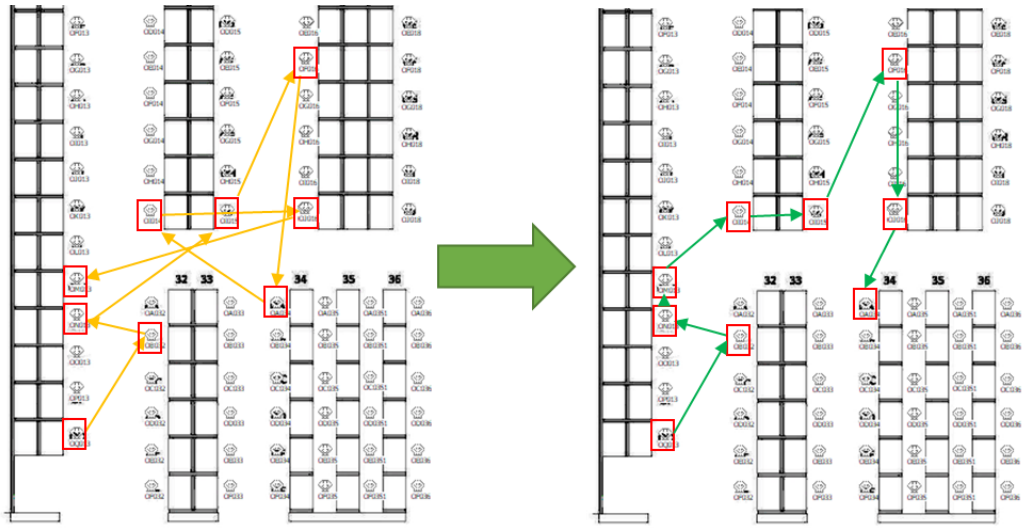
Figura 16 - Representação da segunda função do modelo de simulação.



Fonte: Autor.

Nesta etapa, o operador percorre a nova rota de coleta dos itens, que, em alguns casos, poderia apresentar algumas ineficiências que aumentam a distância total percorrida e foram reorganizadas através do método escolhido, o *Simulated Annealing* (Figura 17). Desta forma é possível comparar se a reorganização da sequência de coleta realmente diminui a distância total. Para realizar este comparativo, a sequência original é simulada, o valor percorrido pelo operador é anotado. O mesmo acontece com a sequência sugerida pelo algoritmo, e se torna possível confrontar os valores obtidos.

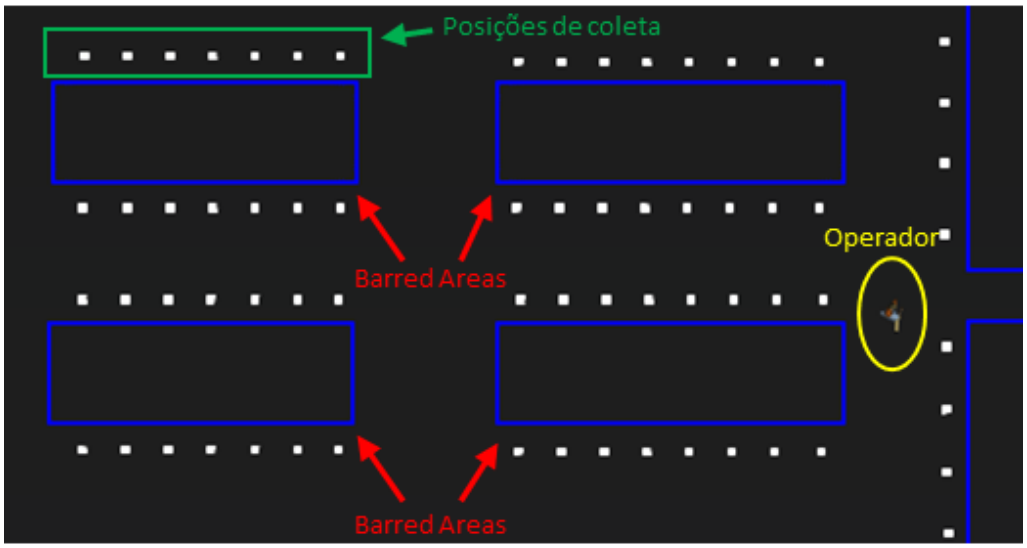
Figura 17 - Exemplo de uma rota de coleta reorganizada.



Fonte: Autor.

Tanto para esta função como para a função de preenchimento da tabela "de-para", o modelo leva em consideração as dimensões reais do CD, inclusive impedindo a movimentação em linha reta entre posições que possuem ilhas de *pallets* entre elas, isso é possível apenas com a utilização de *"barred areas"*, que fazem com que o operador desvie de obstáculos ao se movimentar. A Figura 18 demonstra estes obstáculos na vista superior da representação tridimensional do modelo, indicado com setas em vermelho, o operador se movimentando com um destaque em amarelo e as posições de coleta destacadas em verde.

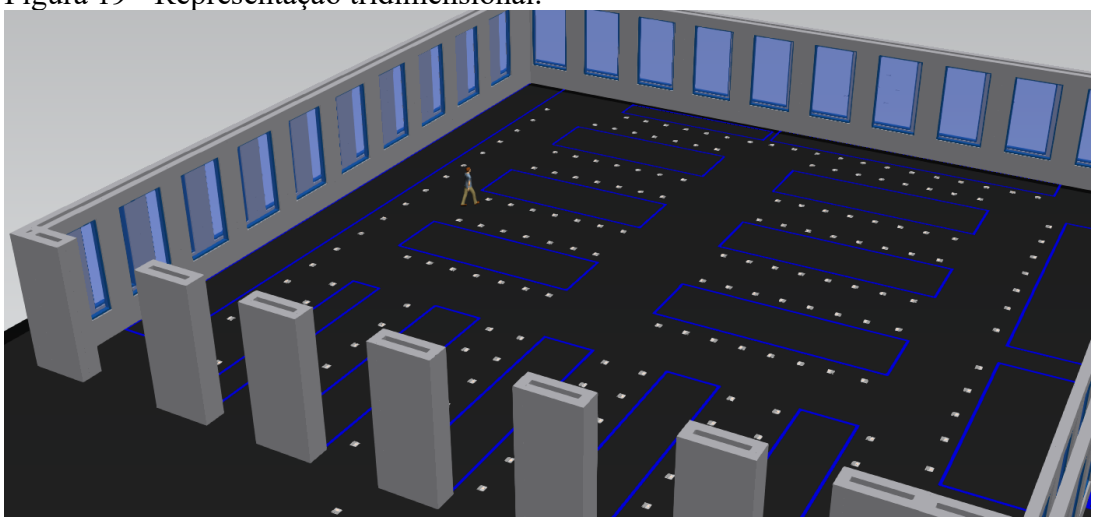
Figura 18 - Representação tridimensional do modelo em detalhes



Fonte: Autor.

Este modelo tridimensional é uma das características que diferenciam a ferramenta *Plant simulation*, que permitem a visualização dos equipamentos e *layouts*, integração com as outras ferramentas de desenvolvimento e programação de robôs e dispositivos, além de ser um pré-requisito para a configuração da locomoção de operadores livremente pela área de produção. A Figura 19 exibe a representação tridimensional para este CD.

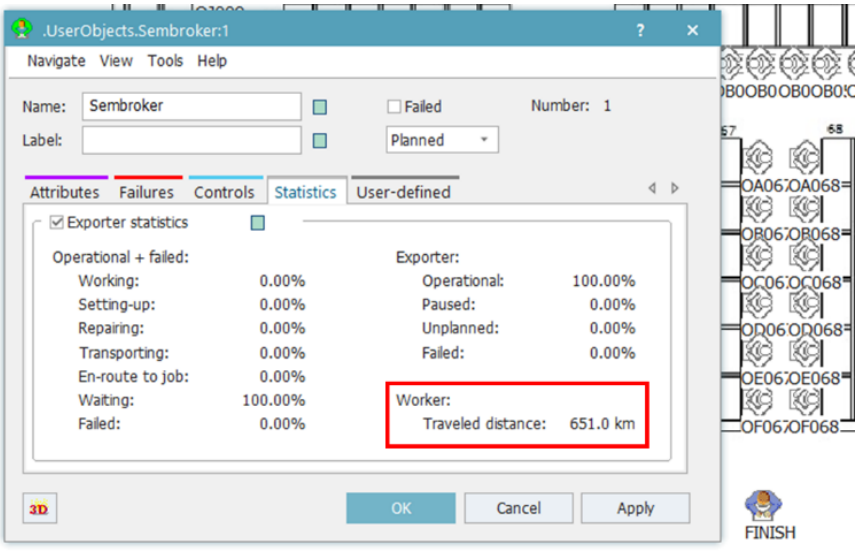
Figura 19 - Representação tridimensional.



Fonte: Autor.

Após percorrer todas as rotas de coleta o modelo de simulação deve apresentar o resultado da distância percorrida (Figura 20) a fim de se comparar com o valor obtido pelo cálculo do algoritmo de estudo das rotas e validar que para o layout estudado a nova sequência de coleta dos itens para cada pedido seja mais eficiente.

Figura 20 - Apresentação de resultado

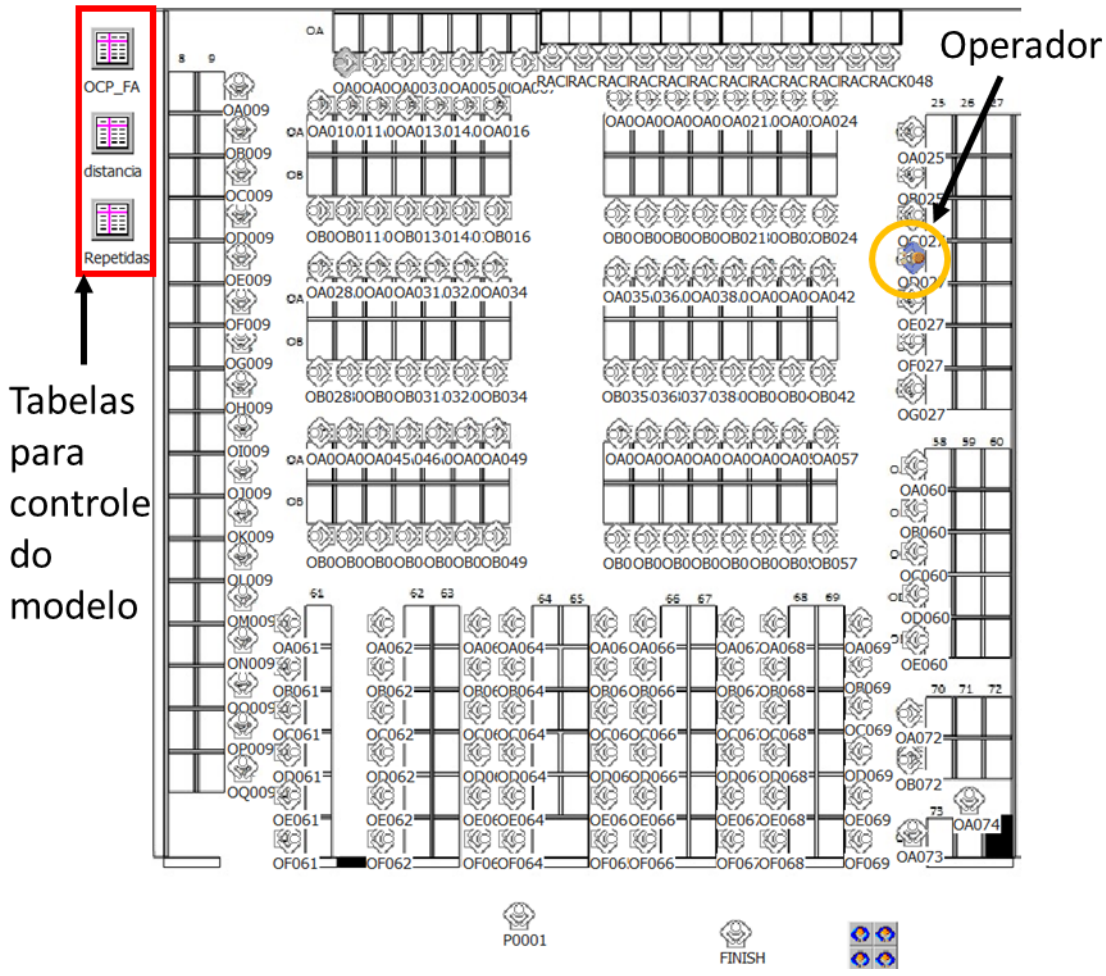


Fonte: Autor.

Esta função também foi programada por “Método” do *Plant Simulation*, com uma programação para que o operador realize a coleta dos itens seguindo uma ordem. Esta ordem é coletada em uma tabela do modelo chamada “OCP_FA”, com a sequência de coleta do operador. Mais informações sobre os “Métodos” utilizados neste modelo podem ser visualizadas no Apêndice A.

A Figura 21 demonstra o modelo em funcionamento, as tabelas utilizadas com destaque no lado esquerdo, enquanto o operador, destacado no lado direito da figura percorre as posições de coleta de itens.

Figura 21 - Modelo de simulação em funcionamento



Fonte: Autor

Com a utilização destas tabelas e Métodos, é possível comandar o operador a percorrer pelas ordens de coleta dos pedidos, a fim de se obter a distância percorrida em cada um dos cenários.

.

5.4 ALGORITMO PARA ESTUDO DAS ROTAS

O algoritmo de estudo das rotas foi construído com a aplicação do método *Simulated Annealing*, selecionado por sua capacidade de convergir para resultados bons com agilidade suficiente para utilização no dia-a-dia das empresas com este tipo de operação de *picking* e também por sua simplicidade de implementação. Além deste método, outras opções foram testas, como por exemplo o algoritmo *Particle Swarm* e *Ant Colony*, porém não foi possível convergir em um resultado em um deles, enquanto no outro algoritmo não foi possível construí-lo com as informações e variáveis disponibilizadas para este estudo.

O algoritmo inicia com a definição dos parâmetros de controle que são utilizados pelo método *Simulated Annealing*, como a temperatura inicial, a temperatura final e a taxa de resfriamento. Na sequência, é necessária a introdução de dados iniciais, como, por exemplo, a sequência atual de coleta, a planilha de distâncias "de-para" e outras informações referentes a produtos com mais de uma "posição *pallet*" disponível. Com estas informações, o algoritmo deve então calcular a distância total percorrida com a solução inicial.

No próximo passo começa a implementação do algoritmo referente ao *Simulated Annealing* que atua enquanto a temperatura atual for maior do que a temperatura final (parâmetro de parada), realizando permutações, trocando um item por outro na sequência de coleta , a fim de verificar possíveis diminuições nas rotas percorridas pelos ajudantes. Após cada permuta é realizado o cálculo da distância utilizando a regra do vizinho mais próximo. Uma alternativa a utilização desta regra seria a utilização de um método exato, como, por exemplo, o algoritmo do caixeiro viajante. Este não foi implementado pois aumentaria grandemente a complexidade do programa, e, consequentemente, custo computacional e o tempo para execução aumentariam possivelmente para níveis que tornariam inviável a utilização deste algoritmo, levando em consideração que a linguagem de programação selecionada é compilada e não interpretada, aumentando o tempo de realização dos cálculos. Outra informação importante é que as permutas são realizadas apenas entre os itens do mesmo *pallet*, evitando que produtos sejam erroneamente alocados em pedidos onde não foram solicitados.

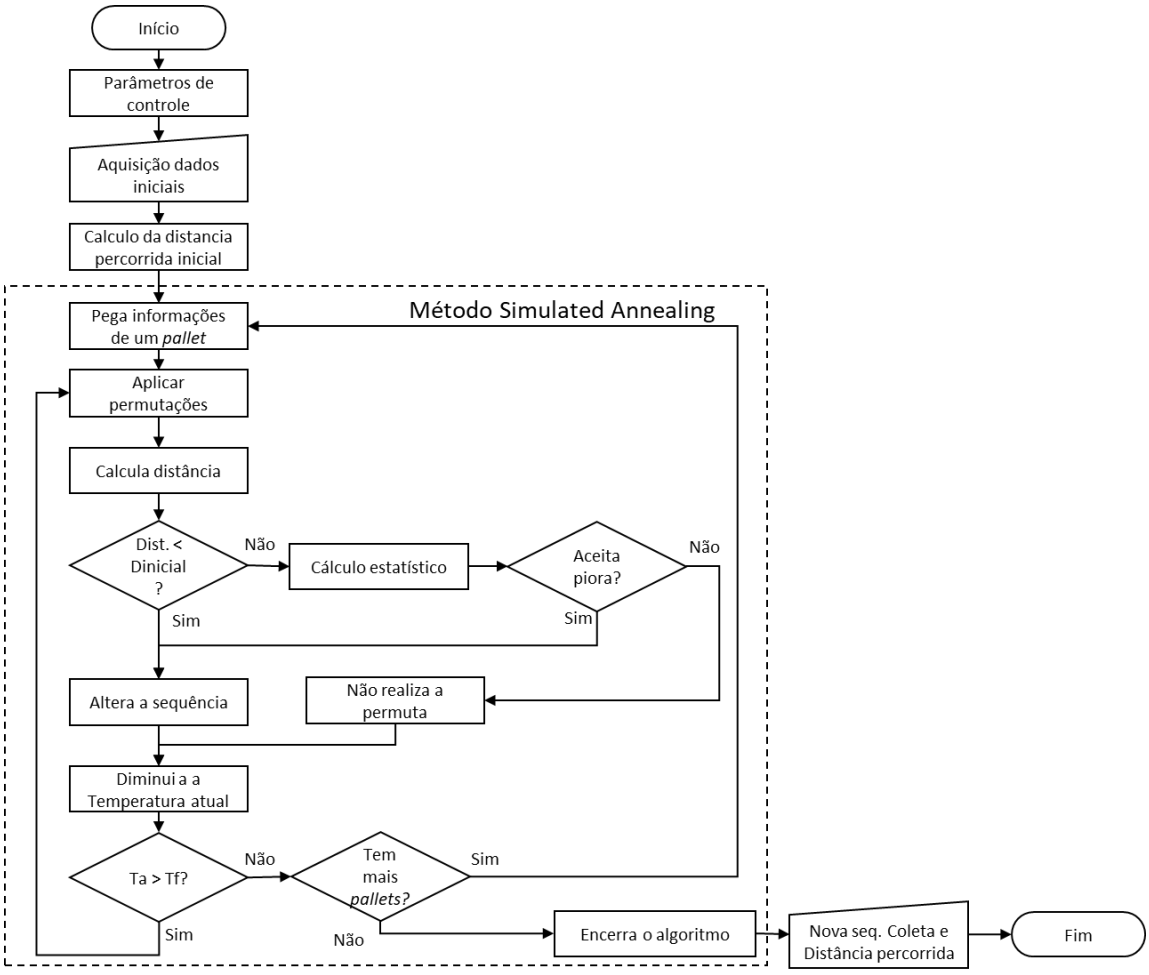
Com a distância calculada após a permuta, verifica-se a diferença entre o valor atual e o valor original. No caso de a distância ser menor do que a original, o algoritmo substitui a sequência antiga pela nova. Seguindo as diretrizes do *Simulated Annealing*, existe um cálculo estatístico para a aceitação ou não de resultados piores. Esta lógica atua em busca de evitar travar o resultado final em um mínimo local, permitindo que o algoritmo rompa barreiras em

busca de resultados melhores. Portanto, mesmo que a distância calculada seja maior do que a original, realiza-se uma consulta a um valor dentro de uma função que vai definir se este valor pode ser aceito como uma resposta, gerando uma nova sequência.

Após cada iteração, a temperatura diminui de acordo com a taxa de resfriamento configurada no início do processo. O algoritmo fará então a troca da sequência de coleta dos itens do pedido realizando permutações entre 2 itens. Estas trocas são aplicadas até que a temperatura seja menor do que a temperatura final ou se esgotarem as possibilidades de trocas dentro deste pedido/*pallet.* Há então a verificação se existe outro *pallet* a ser analisado. O processo se repete até que não haja mais *pallets* para realização das permutas e o algoritmo se encerra gerando como resultado uma nova ordem de coleta e o cálculo da distância para realização dessa ordem de coleta.

A Figura 22 apresenta um fluxograma com as operações realizadas pelo algoritmo, conforme o que foi descrito anteriormente.

Figura 22 - Fluxograma funcionamento do algoritmo proposto.



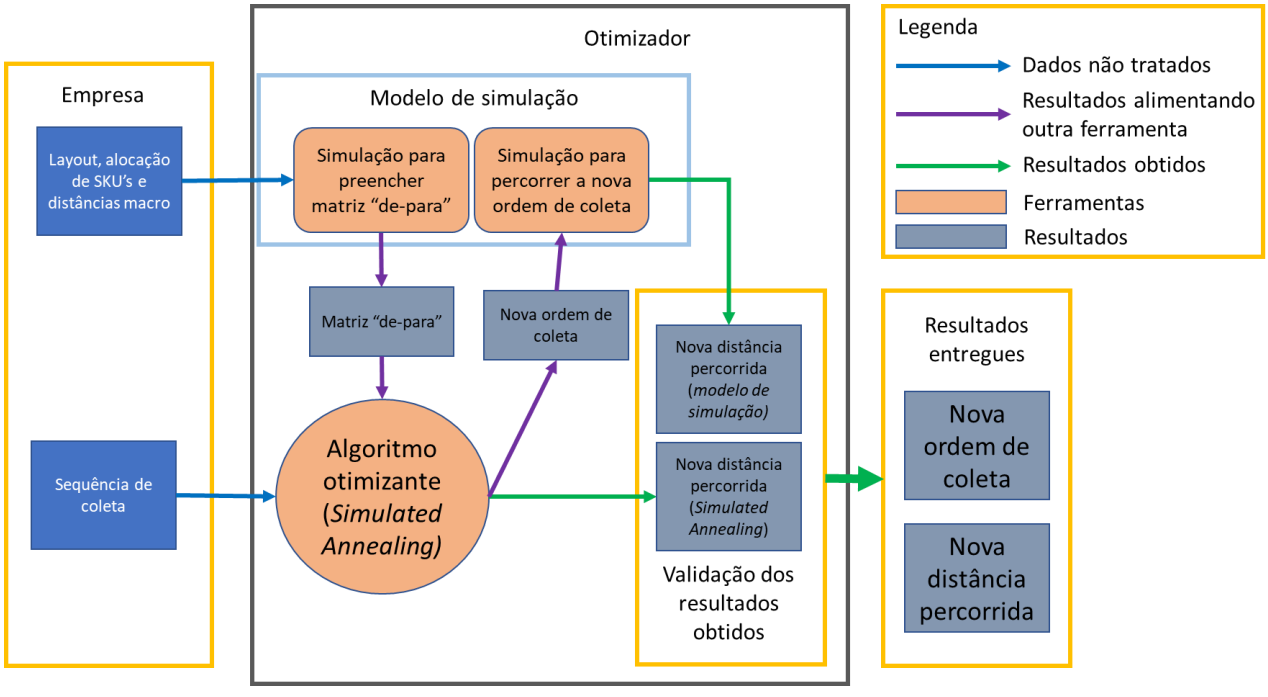
Fonte: Autor.

5.5 *INTEGRAÇÃO DOS MODELOS*

Após a construção do modelo de simulação e a programação do algoritmo de estudo das rotas, utilizando o algoritmo *Simulated Annealing*, é necessário definir como as duas ferramentas devem trabalhar juntas.

A Figura 23 indica brevemente como é feita a entrada de dados no modelo proposto e a integração entre os processos. Os dados fornecidos pela empresa (*layout*, distâncias entre corredores, alocação dos SKU's e sequência de coleta) são as entradas externas usadas no algoritmo de estudo das rotas. A sequência de coleta é utilizada neste algoritmo que vai reordenar a sequência usando o *Simulated Annealing* como base para fazer as alterações. Os outros dados fornecidos são utilizados no modelo de simulação. Este modelo, como mencionado anteriormente, tem duas funções, sendo a primeira a criação de uma matriz "de-para" que serve para alimentar o algoritmo de estudo das rotas com dados que seriam virtualmente impossíveis de se coletar *in loco*.

Figura 23 - Entrada de dados.



Fonte: Autor.

Após a disponibilização dos dados necessários (matriz "de-para", posição dos SKU's e ordem de coleta) o algoritmo de estudo das rotas entra em operação e traz como resultado uma nova ordem de coleta e o cálculo da distância percorrida com esta nova coleta. Neste momento o modelo de simulação se torna necessário novamente para a validação da distância calculada

pelo algoritmo. A nova ordem de coleta é então inserida no modelo de simulação a fim de se extrair a distância percorrida. Os valores obtidos são então confrontados a fim de validar que a nova sequência de coleta reduz a distância total percorrida pelos operadores.

6 ANÁLISE DOS RESULTADOS

Nesta seção são apresentados e discutidos os resultados obtidos através da metodologia proposta no capítulo anterior, a combinação de um modelo de simulação e do algoritmo *Simulated Annealing*. Os modelos de simulação foram desenvolvidos no *Plant Simulation* da desenvolvedora *Siemens*. A linguagem de programação utilizada para o algoritmo foi o *Python*. A escolha dessa linguagem foi baseada em conhecimento prévio em utilizá-la. A programação foi implementada na versão *python 3.8.4*, com a adição da biblioteca *Numpy 1.19.0*. Outras bibliotecas que foram utilizadas já são nativas do *python*, sem a necessidade de serem adicionadas externamente, são disponibilizadas apenas com comandos de importação, por exemplo a "*math*", que contém funções matemáticas mais complexas e a "*threading*", que possibilita a realização do *multithreading*, uma técnica de computação de paralelismo, permitindo uma agilidade maior na realização das tarefas contidas no programa.

6.1 DESIGN DO EXPERIMENTO

Como está sendo adotada uma abordagem dependendo de 3 etapas, cada uma tem suas variáveis e pode exigir análises mais robustas, ou não.

Sendo a primeira etapa o modelo de simulação para coleta dos dados para a matriz "de-para", este experimento não exige que esta operação seja repetida diversas vezes, visto que não há elementos que possam ocasionar variações. O modelo simplesmente percorre todas as posições sem aleatoriedades a fim de preencher a tabela necessária para a próxima etapa.

Na sequência, o algoritmo para o estudo das rotas, *Simulated Annealing*, possui algumas variáveis que permitem alcançar maior acurácia, porém, podem exigir mais processamento computacional, e eventualmente demorar mais para chegar em algum resultado. Neste experimento a única variável que foi utilizada para este controle foi a taxa de resfriamento, mantendo temperatura inicial e final (Ti = 1 e Tf = 0) sempre as mesmas durante as execuções. Foram executados 3 valores diferentes para a taxa de resfriamento, sendo eles 0,5, 0,9 e 0,99, selecionados empiricamente, a fim de se avaliar o *trade-off* entre o tempo de execução e melhores resultados.

A primeira execução do algoritmo foi realizada com a taxa de resfriamento em 0,5 para avaliar se com esta taxa seria possível convergir em algum resultado rapidamente. Como o algoritmo chegou a uma proposta de reorganização de todos os pedidos de um mês de operação em um pouco mais de 48 minutos, conclui-se que seria possível aumentar esta taxa. Na sequência foi testada a opção de 0,9 que convergiu para um resultado melhor demandando cerca

de cinco horas e meia de execução do algoritmo. O último teste foi realizado com o valor de 0,99 para verificar quanto ainda seria possível obter de melhoria no resultado, não importando o tempo de execução. Este teste demandou cerca de 46 horas, e os resultados apresentaram uma redução de mais 0,2% na distância absoluta percorrida pelos operários quando comparado ao teste anterior.

Por último, efetua-se a simulação para validação do resultado obtido com a nova ordem de coleta, e, assim como não houve variações para o modelo de preenchimento da matriz "de-para", os resultados permanecem constantes quando comparando as três rodadas de teste que foram realizadas.

6.2 RESULTADOS

A princípio, esta metodologia foi aplicada em uma das unidades desta grande empresa do setor de bebidas, a qual forneceu os dados de montagem de *pallets* e pedidos de um mês inteiro, as distâncias macro do *layout* e as alocações dos produtos. A Tabela 1 apresenta um resumo dos resultados encontrados do algoritmo de estudo das rotas, indicando a distância percorrida pelo período analisado (um mês), a diferença entre a distância inicial e a distância reorganizada e o tempo de execução, de acordo com as diferentes taxas de resfriamento.

Tabela 1 - Resumo dos resultados do *Simulated Annealing*.

CR	Distância (km)		Diferença		Tempo de execução (h)
	inicial	reorganizada	km	%	
0,5	706,26	652,61	53,65	7,60%	0,7
0,9	706,26	649,13	57,13	8,09%	5,4
0,99	706,26	648,01	58,25	8,25%	46

Fonte: Autor.

Para este CD estudado, portanto, o algoritmo *Simulated Annealing* encontra mais do que 7,5% de diminuição na distância total percorrida pelos operadores. Estes valores devem ser confrontados agora pelo modelo de simulação. A Tabela 2 apresenta os resultados obtidos após esta execução, o qual indica que, utilizando a sequência proposta com a menor taxa de resfriamento, seria possível reduzir 7,17% da distância original percorrida ao longo de um mês, representando mais do que 50 km de redução da distância percorrida no mês.

Tabela 2 - Resultados após execução no modelo de simulação.

C.R.	Distância (km)		Diferença	
	inicial	otimizada	km	%
0,5	705,68	655,05	50,63	7,17%
0,9	705,68	651,88	53,8	7,62%
0,99	705,68	651,01	54,67	7,75%

Fonte: Autor.

Ao comparar os resultados obtidos pelos dois métodos, pode se concluir que a aplicação do algoritmo *Simulated Annealing* apresenta redução da distância percorrida e foi validado pela aplicação da sequência sugerida em um modelo de simulação, visto que, em comparação entre os cenários, não houve nenhum que apresentou diferença maior do que 0,5%. A comparação entre os resultados pode ser vista na Tabela 3 - Comparação entre os cenários.Tabela 3.

Tabela 3 - Comparação entre os cenários.

Cenário	Distância (km)		Diferença	
	S.A.	Simulação	km	%
Inicial	706,26	705,68	0,58	0,08%
C.R. = 0,5	652,61	655,05	2,44	0,37%
C.R. = 0,9	649,13	651,88	2,75	0,42%
C.R. = 0,99	648,01	651,01	3,00	0,46%

Fonte: Autor.

Portanto, com baixos índices de erro, quando comparando o algoritmo e o modelo de simulação, e com resultados apresentando diminuição da distância total percorrida, o algoritmo *Simulated Annealing* pode ser considerado uma opção viável para aplicação neste problema.

6.3 ANÁLISE

Sobre os resultados obtidos, três aspectos valem a pena ser discutidos. O primeiro é o tempo de execução, o segundo é o valor encontrado e o que este representa e o último é a peculiaridade de cada CD.

6.3.1 Tempo de execução

Quando alterado o valor da taxa de resfriamento, houve grande diferença entre os tempos de execução. Para o valor mais baixo (0,5) a ferramenta executou os processos muito rapidamente, em menos de uma hora os resultados foram disponibilizados. Quando utilizado o

valor 0,9 a execução demorou consideravelmente mais, 5 horas e meia, um valor ainda aceitável, visto que estamos falando de um algoritmo que faz trocas aleatórias e aceita alguns pontos de “piora” no resultado. Porém, para o valor mais alto (0,99) o tempo de execução aumentou exponencialmente, conforme o esperado, demorando mais de 45 horas para a execução do algoritmo, realizando muito mais trocas com esta taxa de resfriamento e não necessariamente obtendo um resultado muito melhor, neste caso, com a distância total percorrida apenas 0,2% menor.

Para o dia-a-dia da empresa, utilizar o valor de 0,99 torna-se inviável para o estudo que foi realizado, com o horizonte de análise de 30 dias, devido ao elevado tempo de execução com os métodos que foram empregados. Já os outros dois valores vão apresentar desempenhos bons em tempos aceitáveis. Neste estudo foram analisados os pedidos de um período de 26 dias, porém a aplicação ideal desta metodologia seria a execução do algoritmo diariamente, reorganizando as rotas definidas pelo WMS a fim de se realizar a operação de coleta dos itens de maneira mais eficiente todos os dias. Com isto, o tempo de execução será menor do que os tempos exigidos para completar a análise apresentada neste trabalho.

Estas análises foram desenvolvidas em um computador com processador Intel Core i7 de sétima geração com 16 gigabytes de memória RAM rodando com Windows 10. Espera-se tempos de execução diferentes em computadores com outras configurações.

6.3.2 Valores encontrados

Neste CD estudado, o setor de *picking* opera com 11 ajudantes, sendo cada um responsável por aproximadamente 9% da distância total percorrida. Os resultados encontrados podem parecer baixos, porém não exigem investimentos em infraestrutura e aumentam a eficiência da operação. Isto pode garantir com que a operação não “capote” - Termo utilizado quando a operação ultrapassa o período reservado para tal, resultando no atraso da saída de alguns veículos para entrega; permitindo que, pela manhã, todos os caminhões já estejam abastecidos com os produtos e prontos para cumprir suas rotas. Além disso, a redução e distância total percorrida, aliada com algumas outras poucas melhorias, permitiria reduzir 1 operador, diminuindo este custo ao longo do ano. Como esta empresa possui mais de 100 centros de distribuição com operação de *picking*, o impacto da utilização da ferramenta na rede dos CD pode trazer ganhos relevantes e reduções de custo importantes para a operação e com menos desperdícios.

6.3.3 Peculiaridades entre os CDDs

Cada centro de distribuição é responsável pelo atendimento de uma região específica, sujeito a sazonalidades, ações de *marketing* localizado, negociações com redes e parceiros, entre muitas outras variáveis. Isto faz com que cada unidade opere com quantidades médias diferentes de SKUs nos *pallets* montados e entregues aos clientes. No caso do CD estudado, este valor é baixo, ilustrado pela Figura 24, apresentando média de 7,21 produtos diferentes em cada *pallet*. Isto limita a possibilidade de trocas na rota de coleta e consequente diminuição da distância. Em um *pallet* com 2 SKUs diferentes, existem apenas duas opções de rota, sendo, uma "de A pra B" e outra "de B pra A". Quanto maior a quantidade, maior a possibilidade de realização de permutas pelo algoritmo, maior a chance de se encontrar rotas menores, diminuindo mais significativamente a distância total.

Figura 24 - Tabela dinâmica dos pedidos.

Rótulos de Linha	Contagem de Item	Soma de Quantidade
864442	**4**	**6**
P01_A_01_1/35	1	1
P01_M_01_1/35	1	1
P02_A_02_1/35	1	1
P02_M_02_1/35	1	3
864443	**38**	**295**
P02_A_02_1/42	4	42
P02_M_02_1/42	12	125
P03_A_03_1/42	15	63
P03_M_03_1/42	6	61
Z_ITEM_NAO_PALLETIZADO	1	4
864444	**56**	**377**
P01_A_01_1/42	12	85
P01_M_01_1/42	16	45
P02_A_02_1/42	4	9
P02_M_02_1/42	21	213
P03_A_03_1/42	1	12
P03_M_03_1/42	1	12
Z_ITEM_NAO_PALLETIZADO	1	1
864445	**28**	**185**
P01_A_01_1/42	1	20
P01_M_01_1/42	15	111
P02_A_02_1/42	3	3
P02_M_02_1/42	6	26
P03_A_03_1/42	1	12
P03_M_03_1/42	1	12
Z_ITEM_NAO_PALLETIZADO	1	1

Fonte: Autor.

Em outros centros de distribuição a montagem do pallet pode contar com mais do que 20 produtos em média. Neste caso, especula-se que o benefício em utilizar a metodologia proposta apresente resultados ainda mais relevantes.

7 CONCLUSÃO

Levando-se em consideração a importância dos setores de *picking* em armazéns e o custo de tempo durante esta operação ser, em sua maioria, relacionado ao deslocamento dos operadores se movimentando para a coleta dos itens, a aplicação de um algoritmo que realize o estudo e reorganização da sequência de coleta pode trazer benefícios relevantes para empresas de logística que operam com *picking* discreto.

A metodologia utilizada neste trabalho certificou que a utilização do algoritmo *Simulated Annealing* é aplicável a este cenário e conseguiu reduzir a distância percorrida pelos operadores, tendo como base um estudo realizado em um centro de distribuição de uma grande empresa do ramo de bebidas no Brasil. Utilizou-se um modelo de simulação para conseguir informações de posição e distância entre cada local de coleta de itens, informações importantes para a utilização do algoritmo. Este modelo também foi utilizado para validar que os cálculos e alterações realizadas pelo *Simulated Annealing*, fazendo uma rodada de simulação para certificar que a distância percorrida pelo operador foi reduzida.

Vale salientar que para esta metodologia é necessário utilizar dados de *layout*, posição das ilhas e locais de coletas e dados de operação atualizados para que a sequência sugerida esteja alinhada com os parâmetros de *picking*. Outro ponto importante é a característica das vendas e operação de cada centro de distribuição. O CD estudado apresenta baixo número de SKUs diferentes por *pallet* e o algoritmo encontrou uma diminuição de pelo menos 7,6% da distância. Em um CD com pedidos mais variados e com volume maior, os ganhos podem ser ainda melhores, devido a possibilidade de haver mais trocas de sequência em cada *pallet*.

A utilização do *Simulated Annealing* se mostrou viável pois, além de apresentar redução das distâncias percorridas, executou os cálculos em um período relativamente curto. Foi realizado um teste para avaliar o quanto o parâmetro “taxa de resfriamento” interfere na diminuição da distância e no tempo de execução. Com um valor intermediário os resultados foram disponibilizados em menos de uma hora, enquanto com um valor maior, que trouxe um resultado consideravelmente melhor, demorou um pouco mais de 5 horas. Um último teste com o valor mais alto possível demorou em torno de 45 horas para convergir ao resultado final, trazendo uma melhoria de apenas 0,2% com relação ao cenário anterior.

Outra questão relevante quanto ao método escolhido é que, apesar de ter apresentado resultados de forma ágil, a linguagem de programação interfere no tempo de realização dos cálculos, podendo convergir muito mais rapidamente usando uma linguagem compilada ao invés de uma interpretada, como a que foi utilizada neste trabalho.

Quanto às limitações do modelo estudado, pode ser relacionado a quantidade de pedidos e *pallets* montados, que se for maior do que a quantidade atual deste CD pode demorar mais a convergir, se utilizada a linguagem de programação atual. Outro ponto importante é a questão da aquisição das informações reais do *layout* da área de *picking*¸ que se feito de maneira pouco precisa não garante assertividade dos resultados encontrados.

Portanto, conclui-se que a metodologia proposta neste trabalho apresenta resultados que diminuem a distância de deslocamento dos operadores, além de ser aplicável em vários centros de distribuição, desde que seja possível a obtenção dos dados e parâmetros de entrada.

REFERÊNCIAS

ARDJMAND, E.; SANEI BAJGIRAN, O.; YOUSSEF, E. **Using list-based simulated annealing and genetic algorithm for order batching and picker routing in put wall-based picking systems**, Applied Soft Computing Journal. Elsevier B.V., 75, pp. 106–119. doi: 10.1016/j.asoc.2018.11.019. 2019.

BALLOU, R. H. **Gerenciamento da cadeira de suprimentos: logística empresarial**. Tradução de Raul Rubenich. 5. ed. Porto Alegre: Bookman, 2006. 616 p. ISBN 8536305916.

BANKS, J.; CARSEN, J. S., **Discrete event system simulation**, Prendice-Hall, Englewood Cliffs, NJ, 1984.

BATEMAN, R. E. **Simulação de sistemas: aprimorando processos de logística, serviços e manufatura**. Elsevier, 2013.

ČERNÝ, V. **Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications**, v. 45, n. 1, p. 41-51, 1985.

Chen, T. L. et al. **An efficient hybrid algorithm for integrated order batching, sequencing and routing problem**, International Journal of Production Economics. Elsevier, 159, pp. 158–167. doi: 10.1016/j.ijpe.2014.09.029. 2015.

Cheng, C. Y. et al. **Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem**, International Journal of Production Economics. Elsevier, 170, pp. 805–814. doi: 10.1016/j.ijpe.2015.03.021. 2015.

CHWIF, L.; MEDINA, A. C. **Modelagem e simulação de eventos discretos: teoria e aplicações**. 4 ed. Rio de Janeiro: Elsevier. 309 p. 2015.

CORRÊA, H. L.; GIANESI, I. G. N; CAON, M. **Planejamento, programação e controle da produção**. São Paulo: Atlas, v. 1, 2001.

CORTÉS, P. et al. **A tabu search approach to solving the picking routing problem for large- and medium-size distribution centres considering the availability of inventory and K**

heterogeneous material handling equipment, Applied Soft Computing Journal, 53(April), pp. 61–73. doi:10.1016/j.asoc.2016.12.026. 2017.

DIEFENBACH, H.; GLOCK, C. H. (2019) **Ergonomic and economic optimization of layout and item assignment of a U-shaped order picking zone**, Computers and Industrial Engineering. Elsevier Ltd, 138, p. 106094. doi:10.1016/j.cie.2019.106094. 2019.

DIJKSTRA, A. S.; ROODBERGEN, K. J. **Exact route-length formulas and a storage location assignment heuristic for picker-to-parts warehouses'**, Transportation Research Part E: Logistics and Transportation Review. Elsevier Ltd, 102, pp. 38–59. doi: 10.1016/j.tre.2017.04.003. 2017.

FAYYAZ, Z. et al. **Simulated annealing optimization in wavefront shaping controlled transmission**, Applied Optics, 57(21), p. 6233. doi: 10.1364/ao.57.006233. 2018.

FERNANDES, P. D. C. **Otimização do Processo de Picking**. Setúbal, 2017.

FONTES, F. J. T. **Redesenho e Optimização de Armazém**. Porto, 2010.

GENG, Z. et al. **Energy optimization and analysis modeling based on extreme learning machine integrated index decomposition analysis: Application to complex chemical processes**'. doi: 10.1016/j.energy.2016.12.090. 2017.

GIL, A.C.G.I.L. **Como elaborar projetos de pesquisa**. [S.l.]: Atlas, 2010.

VAN GILS, T. et al. **Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review**, European Journal of Operational Research. Elsevier B.V., 267(1), pp. 1–15. doi: 10.1016/j.ejor.2017.09.002. 2018.

GROSSE, E. H., GLOCK, C. H.; NEUMANN, W. P. **Human factors in order picking: a content analysis of the literature**, International Journal of Production Research. Taylor & Francis, 55(5), pp. 1260–1276. doi: 10.1080/00207543.2016.1186296. 2017.

HARREL; C. R.; MOTT J. R. A.; BATEMAN R. E.; BOWDEN R. G.; GOGG T. J. **Simulação: otimizando os sistemas**. 2ed., São Paulo: IMAM, 2002.

HOLLOCKS, B. **A well –kept secret? Simulation in manufacturing industry reviewed. Or Insight**, p. 12-17, out. – dez. 1992.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. **Optimization by simulated annealing**. science, v. 220, n. 4598, p. 671-680, 1983.

DE KOSTER, R.; LE-DUC, T.; ROODBERGEN, K. J. **Design and control of warehouse order picking: A literature review. European journal of operational research**, v. 182, n. 2, p. 481-501, 2007.

KRAJEWSKI, L. J.; RITZMAN, L. P. **Operations Management: strategy and analysis**. 6 ed. New Jersey: Prentice Hall. 882 p. 2001.

KULAK, O., SAHIN, Y.; TANER, M. E. **Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms**, Flexible Services and Manufacturing Journal, 24(1), pp. 52–80. doi: 10.1007/s10696-011-9101-8. 2012.

LAW, A.M.; KELTON, W.D. **Simulation Modelling & Analysis**. 3ªed. Nova Iorque: McGraw-Hill Books. 2000.

LAW, A. M. **Simulation modeling and analysis**. 4ª ed. McGraw-Hill, 2007.

LI, J., HUANG, R. E DAI, J. B. **Joint optimisation of order batching and picker routing in the online retailer's warehouse in China**, International Journal of Production Research, 55(2), pp. 447–461. doi: 10.1080/00207543.2016.1187313. 2017.

LIN, C. C. et al. **Joint order batching and picker Manhattan routing problem**, Computers and Industrial Engineering. Elsevier Ltd, 95, pp. 164–174. doi: 10.1016/j.cie.2016.03.009. 2016.

LOURENÇO, Á. J. G. **Otimização da atividade de picking num armazém de peças – um caso de estudo**. Porto 2014.

LU, W. et al. **An algorithm for dynamic order-picking in warehouse operations**, European Journal of Operational Research. Elsevier Ltd., 248(1), pp. 107–122. doi: 10.1016/j.ejor.2015.06.074. 2016.

MARTINS, V. C. et al. **Otimização de *layouts* industriais com base em busca tabu**. Gestão & Produção, v. 10, n. 1, p. 69-88, 2003.

MASAE, M., GLOCK, C. H.; GROSSE, E. H. **Order picker routing in warehouses: A systematic literature review**, International Journal of Production Economics. Elsevier B.V., 224, p. 107564. doi: 10.1016/j.ijpe.2019.107564. 2019.

MATUSIAK, M. et al. **A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse**, European Journal of Operational Research. Elsevier B.V., 236(3), pp. 968–977. doi: 10.1016/j.ejor.2013.06.001. 2014.

MCLEAN, C.; LEONG, S., **The Expanding Role of Simulation in Future Manufacturing**, In: Proceedings of the 2001 Winter Simulation Conference, p. 1478-1486, 2001.

MEIRELLES, A. F.; MEIRELLES, L. A.; BARBASTEFANO, R. G.; FLEXA, R. C. **Simulação e Layout - Um estudo de caso**. In: XXIX ENEGEP. Salvador, BA, 2009.

METROPOLIS, N. et al. **Equation of state calculations by fast computing machines**. The journal of chemical physics, v. 21, n. 6, p. 1087-1092, 1953.

MORESCO, G. H. **Análise da eficiência operacional na atividade de picking em uma empresa de tubos e conexões**. Universidade Federal de Santa Catarina. 2017.

OLIVEIRA, Carlos Machado de et al. **Desenvolvimento de um sistema de simulação para cadeias de suprimentos**. Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas. 111p. Tese (Doutorado) 2004.

PEGDEN, C.D., SHANON, R.E., SADOWSKY R. **Introduction to simulation using siman**, McGraw-Hilt New Jersey, 1990.

PETERSEN, C.G. AND AASE, G.R. **Improving Order Picking Efficiency with the Use of Cross Aisles and Storage Policies**. Open Journal of Business and Management, 5, 95-104. doi: 10.4236/ojbm.2017.51009 , 2017

PINEDO, M. L. **Scheduling: Theory, algorithms, and systems**, fifth edition. doi: 10.1007/978-3-319-26580-3. 2018.

PRADO, D. S. **Usando o ARENA em Simulação**. 4. ed. Belo Horizonte: INDG Tecnologia e Serviços Ltda. 2010.

QUADER, S.; CASTILLO-VILLAR, K. K. **Design of an enhanced multi-aisle order-picking system considering storage assignments and routing heuristics**, Robotics and Computer-Integrated Manufacturing. Elsevier, 50, pp. 13–29. doi:10.1016/j.rcim.2015.12.009. 2018.

REEVES, C. R., **Modern heuristic techniques for combinatorial problems**. Editora McGraw-Hill. ISBN 0077092392, 9780077092399. 320 p. (Advanced topics in computer science series) 1995.

RICHARDS, G. Warehouse Managenent: **A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse**. Third Edit. Kogan Page. 2018.

ROBINSON, Stewart. **Simulation: the practice of model development and use**. Chichester: Wiley, 2004.

DE SANTIS, R. et al. **An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses**, European Journal of Operational Research. Elsevier B.V., 267(1), pp. 120–137. doi:10.1016/j.ejor.2017.11.017. 2018.

SELLITTO, M. A.; BORCHARDT, M.; PEREIRA, G. M. **Análise de uma operação logística de carregamento e expedição de cimento por simulação computacional**. Revista Gestão Industrial, v. 5, n. 4, 2009.

SHANNON, R. E**. Introduction to the art and science of simulation**. Winter Simulation Conference. Proceedings (Cat. No. 98CH36274). IEEE, 1998. p. 7-14. 1998.

SILVA, G. Q. et al. **Análise de estratégias de picking aplicada a armazém de empresas de autopeças por meio de simulação discreta**. Simpósio em excelência em gestão da tecnologia. Resende, RJ, 2015.

SOUZA, M. J. F. **Inteligência computacional para otimização**. Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto. Disponível em http://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.pdf. Acesso em: 20 jul. 2020.

TEIXEIRA, E. **Análise e melhoria dos processos de arrumação e picking do armazém exterior de uma empresa do setor da construção civil**, p. 94, 2018.

TOZI, L. A.; SALAZAR, M. S.; CRISTOVÃO, E. S. **Análise do Congestionamento de Caminhões no Sistema de Docas de um Terminal de Carga Aéreo Brasileiro**. In: Anais do X Simpósio de Pesquisa em Transporte Aéreo, Ouro Preto, 2011.

WINSTON, W. L.; GOLDBERG, J. B. **Operations research: applications and algorithms**. Belmont: Thomson Brooks/Cole, 2004.

APÊNDICE A – PROGRAMAÇÃO DO MODELO DE SIMULAÇÃO MÉTODO PARA PREENCHIMENTO DA TABELA "DE-PARA":

```
var i: integer
var j : integer
var A : object
var B : object
var w : real
var k : real
@.AutomaticMediation := false



for i:= 1 to distancias.Ydim-1
        for j:=1 to distancias.Xdim -1
                A := str_to_obj(distancias[1,i+1])
                B := str_to_obj(distancias [j+1,1])
                @.goto(A)
                waituntil @.Location = A
                w:=@.StatTraveledDistance
                @.goto(B)
                waituntil @.Location = B
                @.goto(A)
                waituntil @.Location = A
                k:= @.StatTraveledDistance-w
                distancias[j+1,i+1] := to_str(k/2)
                distancia[j+1,i+1] := to_str(k/2)


        next
        next
```

APÊNDICE B – PROGRAMAÇÃO DO MODELO DE SIMULAÇÃO MÉTODO PARA REALIZAÇÃO DA ORDEM DE COLETA:

```
var i : integer
var j : integer
var x : integer
var k : integer
var l : integer
var Dist: real
var y: integer




for var col := 1 to @.Current_OCP.XDim  -- Este trecho está apagando a OCP dentro do Ajudante
        for var row := 1 to @.Current_OCP.YDim
            @.Current_OCP [col,row] := VOID
        next
    next

    for var row := 1 to OCP_FB.YDim          -- este trecho está preenchendo a OCP dentro do ajudante
        @.Current_OCP [1,row] := OCP_FB [1,row]
        @.Current_OCP [2,row] := OCP_FB [2,row]
        @.Current_OCP [3,row] := OCP_FB [3,row]
        @.Current_OCP [4,row] := OCP_FB [4,row]
        @.Current_OCP [5,row] := OCP_FB [5,row]
    next



--@.goto(Begin_A)
--waituntil @.Location = Begin_A
```

```
var dists : real := 10000
    @.step := 1
    for  i := 1 to @.current_OCP.Ydim
        var A : object := str_to_obj(@.Current_OCP [4,i])
        for y:= 1 to Repetidas.YDim
            if Repetidas[1,y] = A then
                for j:= 1 to distancia.Xdim
                    if distancia[j,1] = obj_to_str(B) then
                        for x:= 1 to distancia.Ydim
                            if distancia[1,x] =Repetidas[5,y] or distancia[1,x] =Repetidas[2,y] or distancia[1,x] =Repetidas[3,y] or distancia[1,x] =Repetidas[4,y] then
                        dist := str_to_num(distancia[j,x])
                        if dist < dists then  -- verifica se a distância da posição é menor que a anterior
                            dists := dist -- se for menor substitui a distância e a posição para qual o ajudante irá
                            A := str_to_obj(distancia[1,x])
                        end
                    end
                next
            end
        next
            end
            next
        for p :=1 to Posições.Ydim
                    if Posições[1,p] = A then
                        A := str_to_obj(Posições[2, p])
                    end
            next

        dists := 10000
        @.goto(A)
        waituntil @.Location = A
```

APÊNDICE C – PROGRAMAÇÃO PARA O ALGORITMO DE ESTUDO DAS ROTAS - CRIAÇÃO E DEFINIÇÃO DAS FUNÇÕES:

```
# -*- coding: utf-8 -*-
import csv
import numpy as np
import os
from math import exp
import time
import smtplib, ssl
import sys
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import threading as th
os.system("cls")

csv.register_dialect('csvPadrao',
                     delimiter = ';',
                     quoting=csv.QUOTE_NONE,
                     skipinitialspace=True)

def DebugOt(debug, pos, dist, fluxo, skus_juntados):
    if debug:
        print("\n\n########## DebugOt START ##########\n\n")

        print("POS\n")
        print(pos)

        print("\n\n\n\n\nDIST\n")
        print(dist)

        print("\n\n\n\n\nFLUXO\n")
        print(fluxo)

        print("\n\n\n\n\nSKUs_JUNTADOS\n")
        print(skus_juntados)

        print("\n\n########## DebugOt END ##########\n\n")

########## FUNCOES ##########
def CriarPasta(pasta):
    '''
    Cria nova pasta no mesmo diretorio do codigo
    CALL: nome_da_pasta
    OUT : nada
```

```
    '''
    current_directory = os.getcwd()
    final_directory = os.path.join(current_directory, pasta)
    if not os.path.exists(final_directory):
       os.makedirs(final_directory)


def ImportarCSV(arq):
    '''
    Importar os arquivos .csv para matrizes do numpy
    CALL: arq - nome do arquivo .csv que sera importado
    OUT : matriz vinda do arquivo .csv transformada em numpy array
    '''
    #BackupCSV(arq)
    #CRIANDO TABELA
    arquivo = open(arq, 'r')
    with arquivo as ReadFile:
        ReadFile = list(csv.reader(arquivo, delimiter=';'))

        #TROCANDO VIRGULA POR PONTO
        for linha in ReadFile:
            for item in range(len(linha)):
                linha[item] = linha[item].replace(',', '.')
    ReadFile = np.array(ReadFile)
    return ReadFile

def ExportarCSV(M, caminho, arq):
    '''
    Exportar as numpy arrys para arquivos .csv
    CALL : M    - nome da numpy array que sera exportada
           arq - nome do arquivo .csv que sera criado e exportado
    OUT  : nada
    '''
    #ESCREVENDO NOVA TABELA EM .csv
    CriarPasta(caminho)
    if len(caminho)==0:
        pass
    elif caminho[-1]!='/':
        caminho+='/'
    arquivo = open(caminho+arq, 'w', newline='')
    with arquivo as WriteFile:
        write = csv.writer(WriteFile, dialect='csvPadrao')
        write.writerows(M)


def ExibirMatriz(M):
    '''
    Imprimi uma matriz
    CALL   : M - nome da matriz numpy array que será utilizada
    OUTPUT : Nada
```

```
        '''
        for linha in M:
            print(linha)


    ########## PREPARAR AS MATRIZES ##########
    def JuntarSKUs(pos):
        '''
        - Juntar duas SKUs que estao na mesma posicao e atribuir um novo sku, que representa esses dois produtos.
        - Somar o fluxo desses dois SKUs, na fileira e na coluna, pois o fluxo esta associado a posicao do SKU.
        - Juntar as SKUs da mesma posicao na tabela de fluxo, colunas e linhas
        - CALL: pos - matriz pos
        - OUT:  dicionario dos skus juntados {"endereco": [SKUs]}. O primeiro SKU repetido NAO esta na lista
        '''
        skus_juntados = {}
        lista_values_repetidos = []
        flag = True
        i=1
        while(i<len(pos)):
            pos_atu = pos[i][0]
            pos_ant = pos[i-1][0]
            if(pos_atu == pos_ant):
                if flag:
                    flag = False
                    key_pos = pos[i-1][0]
                    value_sku = pos[i-1][1]
                    lista_values_repetidos.append(value_sku)
                    skus_juntados[key_pos] = lista_values_repetidos
                key_pos = pos[i][0]
                value_sku = pos[i][1]
                lista_values_repetidos.append(value_sku)
                skus_juntados[key_pos] = lista_values_repetidos
                pos = np.delete(pos, i, 0)
                i-=1
            else:
                lista_values_repetidos = []
                flag = True
            i+=1

        return pos, skus_juntados


    def CorrigirSKUsJuntados(pos, skus_juntados):
        for key in skus_juntados:
            lista_skus = skus_juntados[key]
            for i in range(1, len(lista_skus)):
                lista_where = np.where(lista_skus[i] == pos)[0]
```

```
                if len(lista_where) > 0:
                    for indice in lista_where:
                        pos[indice][1] = lista_skus[0]
            return pos


        def JuntarFluxo(fluxo, skus_juntados):
            for i in range(2):
                fluxo = fluxo.transpose()
                for key in skus_juntados:
                    header_fluxo = fluxo.transpose()[0]
                    lista_skus_juntados = np.array(skus_juntados[key])
                    lista_skus_juntados = lista_skus_juntados.astype('int32')
                    header = lista_skus_juntados[0]
                    lista_skus_juntados = np.sort(lista_skus_juntados)[::-1]
                    lista_skus_juntados = lista_skus_juntados.astype('str')
                    lista_para_somar = np.array([np.zeros(len(fluxo[0]))])
                    for sku in lista_skus_juntados:
                        indice_fluxo = np.where(sku == header_fluxo)    [0][0]
                        lista_para_somar                =                np.concatenate((lista_para_somar,
np.array([fluxo[indice_fluxo]])), axis=0)
                        fluxo = np.delete(fluxo, indice_fluxo, axis=0)
                    lista_para_somar = lista_para_somar[1:]
                    lista_para_somar = lista_para_somar.astype('int32')
                    soma = np.sum(lista_para_somar,axis=0)
                    soma[0] = header
                    soma = soma.astype('str')
                    soma = np.array([soma])
                    fluxo = np.append(fluxo, soma, axis=0)
            fluxo[0][0] = '0'
            fluxo = OrdenarMatriz(fluxo, 0, 0) #ordena linha
            fluxo = OrdenarMatriz(fluxo, 1, 0) #ordena coluna
            fluxo[0][0] = ''
            return fluxo


        def OrdenarMatriz(M, axis, indice):
            '''
            Ordena uma matriz 2D pela coluna ou pela linha
            Para ordenar fluxo, precisamos converter o '' para '0'
            Para ordenar dist , nao precisamos converter nada
            CALL: M      - matriz a ser ordenada
                  axis   - 0 ordena mudando as linhas e 1 ordena mudando as colunas
                  indice - o indice da matriz que sera a referencia para ordenar
            OUT : M      - Matriz ordenada de acordo com os argumentos dados
            '''
            try:
                M = M.astype('int32')
            except ValueError:
                pass
```

```
    if axis == 1:
        M = M.transpose()

    ind = np.argsort(M[:,indice])
    M = M[ind]

    if axis == 1:
        M = M.transpose()

    M = M.astype('str')
    return M


########## SIMULATED ANNEALING ##########
def EncontrarPos(pos, sku):
    '''
    CALL: sku           - valor do SKU a ser pesquisado
    OUT : lista_pos_sku - todas as posicoes que o SKU pesquisado se encontra.
                          Mesmo que seja apenas uma posicao, retorna como lista
    '''
    sku = str(sku)
    lista_pos_sku = np.array([])
    lista_temp = np.where(sku == pos)[0]
    for indice in lista_temp:
        lista_pos_sku = np.append(lista_pos_sku, pos[indice][0])
    return lista_pos_sku

def EncontrarSKU(posicao, pos):
    '''
    CALL: posicao do sku que sera encontrado
          exemplo: "OA001"
          matriz pos
    OUT:  sku naquela posicao
    '''
    return pos[np.where(str(posicao)==pos)[0][0]][1]


#Essa funcao pega a distancia minima entre todas as posicoes dos skus(inicial e final)
def DistMinimaSKU(pos,dist,sku_in,sku_fin,output=False):
    '''
    OUTPUT: se output=True retorna (posInicialMin,posFinalMin,distMin)\n
    posInicialMin - posicao do sku inicial com menor caminhada\n
    posFinalMin - posicao do sku final com menor caminhada\n
    distMin - distancia minima entre esses skus
    '''
    listaPosInicial = EncontrarPos(pos,sku_in)

    posInicialMin = ""
    posFinalMin = ""
```

```
            distMin = np.inf

            for posicao in listaPosInicial:
               auxDist,auxPos = DistMinimaPosSKU(pos,dist,posicao,sku_fin)
               if(auxDist<distMin):
                  distMin = auxDist
                  posInicialMin = posicao
                  posFinalMin = auxPos
            if output:
               return posInicialMin,posFinalMin,distMin
            if not(output):

               return distMin

         # Essa funcao pega a distancia minima entre uma posicao inicial e uma serie de posicoes
de um sku
         def DistMinimaPosSKU(pos,dist,pos_in,sku_fin):
            '''
            OUT: (distMin,posMin)
            Retorna a distancia minima em float e a posicao em str
            '''
            indexInicial = np.where(dist[0]==pos_in)[0]
            listaPos = EncontrarPos(pos,sku_fin)
            #TODO: se o sku procurado em EncontrarPos() nao estiver na planilha pos, da erro.
Por enquanto devolve a mesma pos e a dist = 0
            if(len(listaPos)==0):
               return 0,pos_in
            #fim do migue
            listaDist = np.empty((1,0),dtype = "float32")
            for posicao in listaPos:
               indexAtual = np.where(dist[0]==posicao)[0]
               listaDist = np.append(listaDist,dist[indexInicial][0][indexAtual][0])
            listaDist = listaDist.astype("float")
            distMin = np.min(listaDist)
            posMin = listaPos[np.where(listaDist==distMin)[0]][0]
            return distMin,posMin


         def EncontrarDistMinima(pos, dist, lista_pos_sku):
            '''

            CALL: lista_pos_sku - lista de todas as posicoes do sku que esta sendo
                        somado no SomarTroca() e SomarCaminhada()
            OUT : distancias    - lista de todas as distancias do sku pesquisado,
                        retornando apenas a menor distancia para cada posicao.
                        A matriz distancias esta ordenada pelos SKUs, mas ela eh
                        de uma dimensao, apenas com as distancias minimas
            '''
            lista_dist_sku = np.array([dist[0]])
            lista_temp = np.zeros([1,len(dist[0])])
```

```
    for endereco in lista_pos_sku:
        for linha in dist:
            if linha[0] == endereco:
                linha = np.reshape(linha, [1,len(linha)])
                lista_temp = np.append(lista_temp, linha, axis=0)
                lista_temp = np.delete(lista_temp, 0, 0)
                lista_dist_sku = np.append(lista_dist_sku, lista_temp, axis=0)
                lista_temp = np.zeros([1,len(dist[0])])

    cabecalho = lista_dist_sku[0][1:]

    for i in range(len(cabecalho)):
        for linha in pos:
            if linha[0] == cabecalho[i]:
                cabecalho[i] = linha[1]

    distancias = np.delete(lista_dist_sku, [0], axis=0)
    distancias = np.delete(distancias, [0], axis=1)
    distancias = distancias.astype('float64')
    lista_dist_min = np.amin(distancias, axis=0, keepdims=True)
    distancias = np.insert(lista_dist_min, 0, cabecalho, axis=0)


    distancias = distancias.transpose()
    ind=np.argsort(distancias[:,0])
    distancias = distancias[ind]

    i=1
    while i<len(distancias):
        sku_ant = distancias[i-1][0]
        sku_atu = distancias[i][0]
        if sku_ant == sku_atu:
            dist_ant = distancias[i-1][1]
            dist_atu = distancias[i][1]
            if dist_ant < dist_atu:
                distancias = np.delete(distancias, i, 0)
            else:
                distancias = np.delete(distancias, i-1, 0)
            i-=1
        i+=1


    distancias = distancias.transpose()
    distancias = np.delete(distancias, 0, axis=0)
    distancias = np.reshape(distancias, (len(distancias[0])))
    return distancias


def EncontrarFluxo(fluxo, sku):
    '''
```

```
        Retorna a linha de fluxo do SKU selecionado, SEM o cabecalho, ja convertido para
'int32'
        CALL: sku       - valor do SKU a ser procurado
              fluxo     - matriz fluxo
        OUT : linha_sku - lista de 'int32' do fluxo encontrado, sem o cabecalho
        '''
        sku = str(sku)
        fluxo_sku = fluxo[np.where(sku==fluxo)[0][1]]
        fluxo_sku = np.delete(fluxo_sku, 0)
        return fluxo_sku


    def SomarCaminhada(pos, dist, fluxo):
        '''
        Calcula a caminhada
        CALL   : pos   - matriz posição
                 dist  - matriz distância
                 fluxo - matriz fluxo
        OUTPUT : soma  - somatório da distância percorrida calculada
        '''
        soma = 0
        for i in range(1, len(fluxo)):
            sku = fluxo[i][0]
            fluxo_sku = EncontrarFluxo(fluxo,sku)

            lista_pos_sku = EncontrarPos(pos, sku)
            dist_sku = EncontrarDistMinima(pos, dist, lista_pos_sku)

            dist_sku  = dist_sku.astype('float64')
            fluxo_sku = fluxo_sku.astype('float64')
            soma = soma + np.sum(dist_sku * fluxo_sku)
        return soma


    def TempoExecucao(tempo_total):
        if tempo_total<60:
            return ("%i s" %tempo_total)

        elif 60<=tempo_total<3600:
            m = tempo_total//60
            resto = tempo_total%60
            s = resto
            return ("%0.2imin %0.2is" %(m,s))

        elif 3600<=tempo_total<86400:
            h = tempo_total//3600
            resto = tempo_total%3600
            m = resto//60
            resto = tempo_total%60
            s = resto
```

```
            return ("%ih %0.2imin %0.2is" %(h,m,s))


        def CaminhadaPedido(pedido,pos,dist,pegaDeixa = False):
            '''
            pedido - pedido que tera sua caminhada calculada, precisa estar no formato (1,0)\n
            pegaDeixa (bool) - se True, comeca e termina na posicao padrao de coleta/retorno do palete (P0001)\n\n
            OUTPUT: caminhada total (float)
            '''
            caminhada = 0
            if pegaDeixa:
                pedido = np.insert(pedido,0,pos[np.where(pos[:,0]=="P0001")[0]][0][1])
                pedido = np.insert(pedido,len(pedido),pos[np.where(pos[:,0]=="P0001")[0]][0][1])
            if len(pedido) == 2:
                #return isso
                caminhada =  DistMinimaSKU(pos,dist,pedido[0],pedido[1])
            else:
                anteriorPos,atualPos,distMin = DistMinimaSKU(pos,dist,pedido[0],pedido[1],True)
                caminhada+=distMin
                for i in range(2,len(pedido)):
                    #TODO: precisa decidir como tratar os skus que estao na OCP e nao aparecem na pos, por enquanto fiz que a dist eh igual a 0 e mantive a pos atual igual
                    distMin,atualPos = DistMinimaPosSKU(pos,dist,atualPos,pedido[i])
                    caminhada+=distMin
            return caminhada
        ########## MAIN ##########
        def NovaOcp(pos,dist,ocp,t_execucao=False,r=0.5):
            '''
            Gera a PickList
            CALL   : ocp       - .csv com todas as informações dos CD em um mês
                     t_execucao - se o tempo de execução será mostrado
            OUTPUT : nova_ocp,caminhada
            '''
            ti = time.time()
            ocp = np.delete(ocp,0,axis = 0)
            auxOcp = np.copy(ocp)
            novaOcp = np.empty((0,1),dtype = "<U22")

            i = 1
            while(len(auxOcp)>0):
                try:
                    atual = auxOcp[i][0]
                except:
                    pedido = [auxOcp[i-1][1]]
                    pedido = np.insert(pedido,0,pos[np.where(pos[:,0]=="P0001")[0]][0][1])
                    pedido = np.append(pedido,pos[np.where(pos[:,0]=="P0001")[0]][0][1])
                    pedido = pedido.reshape(-1,1)
```

```
                novaOcp = np.append(novaOcp,pedido)
                break
            anterior = auxOcp[i-1][0]
            if(anterior != atual):
                pedido = np.array(auxOcp[:,1][0:i])
                if(len(pedido)>1):
                    #otimizar aqui antes do append
                    # pedido = Otimizador(pos,dist,len(pedido),r,pedido)
                    #depois de otimizar, insere onde ele pega e deixa o palete
                    pedido = np.insert(pedido,0,pos[np.where(pos[:,0]=="P0001")[0]][0][1])
                    pedido = pedido.reshape(-1,1)
                    novaOcp = np.append(novaOcp,pedido, axis= 0)
                    auxOcp = np.copy(auxOcp[i:])
                    i=0
                elif(len(pedido)==1):
                    pedido = np.insert(pedido,0,pos[np.where(pos[:,0]=="P0001")[0]][0][1])
                    pedido = pedido.reshape(-1,1)
                    novaOcp = np.append(novaOcp,pedido, axis= 0)
                    auxOcp = np.copy(auxOcp[i:])
                    i=0
            i+=1
        tf = time.time()
        if t_execucao:
            print("\n\n##########\nTEMPO EXECUCAO PL:", TempoExecucao(tf-ti),
end='\n\n')
            print(novaOcp)
        ExportarCSV(novaOcp,", "pick_list.csv")
        return novaOcp.reshape(-1,1)


    def GerarPL(pos, ocp, t_execucao=False):
        '''
        Gera a PickList
        CALL   : pos       - .csv com as posições dos produtos
                 ocp       - .csv com todas as informações dos CD em um mês
                 t_execucao - se o tempo de execução será mostrado
        OUTPUT : ocp
        '''
        ti = time.time()
        ocp = np.delete(ocp,0,axis = 0)
        linha = 1
        while linha < len(ocp):
            if ocp[linha][0] != ocp[linha-1][0]: # mudou o palete
                ocp = np.insert(ocp, linha, DeixarPalete(ocp, pos, linha-1), axis=0)
                linha += 1

            linha += 1

        ocp = np.insert(ocp, len(ocp), DeixarPalete(ocp, pos, linha-1), axis=0)
```

```
            ocp = ocp[:,1]
            ocp = np.reshape(ocp, (len(ocp), 1))

            ocp = np.insert(ocp, 0, ['99991'], axis=0)

            tf = time.time()
            if t_execucao:
                print("\n\n##########\nTEMPO EXECUCAO PL:", TempoExecucao(tf-ti),
end='\n\n')
                print(ocp)
            ExportarCSV(ocp,", "pick_list.csv")
            return ocp

        def GerarFluxo(pos, pick_list, t_execucao=False):
            '''
            Gera o fluxo
            CALL   : pos        - matriz posição
                     pick_list  - matriz pick_list
                     t_execucao - se haverá tempo de execução limite
            OUTPUT : fluxo_zero -
            '''
            ti = time.time()
            ########## IMPORTACAO DAS MATRIZES ##########
            pick_list = pick_list.astype('int32')

            header = np.unique(pos[:,1]).astype('int32')
            header = np.sort(header).astype('str')
            header = np.insert(header, 0, '')
            fluxo_zero = np.zeros((len(header),len(header))).astype('int32').astype('str')
            fluxo_zero[:,0] = header
            fluxo_zero[0,:] = header
            pick_list = np.reshape(pick_list, len(pick_list)).astype('str')
            ########## CONTADOR DO FLUXO ##########
            for i in range(len(pick_list) - 1):
                inicial = pick_list[i]
                final   = pick_list[i+1]

                linha_prod_in = np.where(fluxo_zero == inicial)
                if linha_prod_in[0].size == 0:
                    continue
                linha_prod_in = linha_prod_in[1][0]
                linha_prod_fin   = np.where(fluxo_zero == final)
                if linha_prod_fin[0].size == 0:
                    continue
                linha_prod_fin   = linha_prod_fin[1][0]

                fluxo_zero[linha_prod_in][linha_prod_fin]                                    =
int(fluxo_zero[linha_prod_in][linha_prod_fin]) + 1

            ########## EXPORTACAO ##########
```

```
        tf = time.time()
        if t_execucao:
            print("\n\n##########\nTEMPO EXECUCAO FL:", TempoExecucao(tf-ti),
end='\n\n')
            print(fluxo_zero)

        ExportarCSV(fluxo_zero,", "fluxo.csv")
        return fluxo_zero

    def DeixarPalete(ocp, pos, linha_anterior):

        palete = ocp[linha_anterior][0]
        palete = palete[:3]
        palete = palete[0] + '0' + palete[1:]
        where_palete = np.where(palete == pos)
        if where_palete[0].size == 0:
            return pos[np.where(pos[:,0]=="P0001")[0]]
        else:
            return pos[where_palete[0][0]]

    def Otimizador(pos, dist, l, R,pedido,
                   report=False, t_execucao=False, debug=False):
        '''
        Função principal do FSimulated, utiliza o simulated annealing para otimizar a
resposta\n\n
        CALL   : pos           - matriz posição\n
                 pos_bloqueadas - matriz posição\n
                 fluxo         - matriz fluxo\n
                 dist          - matriz distâncias\n
                 l             - Numero total de atividades\n
                 R             - Cooling rate\n
                 nome_caminho   - caminho no qual arquivo final será salvo\n
                 nome_arquivo   - nome do arquivo final\n
                 lista_emails   - lista de emails que receberão os resultados\n
                 report        - \n
                 t_execucao     - \n
                 debug         - \n
        OUTPUT : novo_pedido
        '''
        # DebugOt(debug, pos, dist, fluxo, skus_juntados)
        ########## PARAMETROS ##########
        n     = l     #Numero total de atividades
        N     = l*n   #Num de iter por temperatura -> l * n (N)
        Ti    = 100   #Temp inic (T0)
        Tf    = 1     #Temp final (TF)
        Ta    = Ti    #Temp atual (T)
        delta = 0     #Resultado atu menos anterior
        ########## OUTPUT ##########
        novo_pedido = pedido
        while(Ta > Tf):
```

```
            #Permutacao
            for i in range(1, N):
                #sorteia os 2 produtos
                skus = np.random.randint(len(pedido),size = 2)
                while(skus[0]==skus[1]):
                    skus = np.random.randint(len(pedido),size = 2)
                #calcula caminhada anterior (soma_ant)
                soma_ant = CaminhadaPedido(pedido,pos,dist,True)
                #troca os produtos
                aux = pedido[skus[1]]
                pedido[skus[1]] = pedido[skus[0]]
                pedido[skus[0]] = aux
                #calcula caminhada atual (soma_atu)
                soma_atu = CaminhadaPedido(pedido,pos,dist,True)
                #calcula a diferenca entre as caminhadas
                delta = soma_atu - soma_ant
                if delta <= 0: #soma apos a troca eh MENOR que a antes da troca ---> MELHORIA
                    soma_ant = soma_atu
                    novo_pedido = np.copy(pedido)
                else:
                    aleatorio = np.random.random() #numero aleatorio entre 0 e 1
                    exponencial = exp(-delta/Ta) #e^-delta/Ta
                    if aleatorio < exponencial: # Aceitar a piora
                        soma_ant = soma_atu #nao faz nada
                        novo_pedido = np.copy(pedido)
                    else:
                        #desfaz a troca
                        aux = pedido[skus[1]]
                        pedido[skus[1]] = pedido[skus[0]]
                        pedido[skus[0]] = aux


                soma_atu = 0

            ########## FIM DA TEMPERATURA ##########
            Ta = R * Ta

        return novo_pedido
```

APENDICE D – ALGORITMO PARA EXECUÇÃO DAS FUNÇÕES

```
#importacao de biblios
import FSimulated as fs
import numpy as np
import sys

#importando matrizes
importarPL = False

dist = fs.ImportarCSV('dist.csv')
pos = fs.ImportarCSV('pos.csv')
pos = np.delete(pos,2,1)
ocp = fs.ImportarCSV('ocp.csv')
if importarPL:
    pick_list = fs.ImportarCSV("pick_list.csv")
else:
    print("Gerando PL")
    pick_list = fs.GerarPL(pos,ocp)
#otimizador
pos_original = np.copy(pos)
pos, skus_juntados = fs.JuntarSKUs(pos)
pos = fs.CorrigirSKUsJuntados(pos, skus_juntados)
print("Calculando caminhada inicial")
caminhada_inicial = fs.CaminhadaPedido(pick_list.reshape(1,len(pick_list))[0],pos,dist)
print("Otimizando")
nova_ocp = fs.NovaOcp(pos_original,dist,ocp,r = 0.9)
caminhada_final = fs.CaminhadaPedido(nova_ocp.reshape(1,len(nova_ocp))[0],pos,dist)
fs.ExportarCSV(nova_ocp,"","novaOCP.csv")
sys.stdout = open("resultados.txt", 'w')
print("Caminhada inicial: %.2f\nCaminhada final:
%.2f"%(caminhada_inicial,caminhada_final
```