UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SISTEMA PARA EMPACOTAMENTO DE VOLUMES REGULARES EM CONTÊINERES UTILIZANDO ALGORITMO GENÉTICO

EBERTON MARX

BLUMENAU
2018

EBERTON MARX

PROTÓTIPO DE SISTEMA PARA EMPACOTAMENTO DE VOLUMES REGULARES UTILIZANDO ALGORITMO GENÉTICO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Marcel Hugo, M. Eng. – Orientador

BLUMENAU
2018

PROTÓTIPO DE SISTEMA PARA EMPACOTAMENTO DE VOLUMES REGULARES EM CONTÊINERES UTILIZANDO ALGORITMO GENÉTICO

Por

EBERTON MARX

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Marcel Hugo, M. Eng. – FURB
Membro:	Prof(a). Roberto Heinzle, Dr – FURB
Membro:	Prof(a). Andreza Sartori, Dr. – FURB

Blumenau, 12 de julho de 2018

Dedico este trabalho aos meus pais que não mediram esforços para me proporcionar a oportunidade de estudar em uma universidade de renome como a FURB.

AGRADECIMENTOS

A Deus pela vida, pelas oportunidades, por me cercar de pessoas justas e boas de coração.

À minha família que sempre será base de amor incondicional

Aos meus amigos que fazem a minha viagem pela vida mais tranquila e leve

Aos meus mestres por compartilharem seu conhecimento, sua atenção, amizade, compreensão e disponibilidade

À minha esposa Vanessa pelo companheirismo e amor.

Às minhas filhas Letícia e Isabela pelo amor que recebo pelo simples fato de existirem.

Aos outros, dou o direito de ser como são. A mim, dou o dever de ser cada dia melhor.

[Chico Xavier]

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de sistema para resolver o problema do empacotamento de volumes tridimensionais em contêineres, também conhecido como 3DBPP. A modelagem do problema utiliza algoritmos genéticos como método de busca para a solução. Os algoritmos genéticos fazem parte da metaheurística computacional e foram criados a partir da observação da teoria da evolução das espécies, proposta por Charles Darwin. Como ponto de partida, o sistema vai ter como entrada uma lista de volumes que simulam a necessidade de um embarque virtual, chamada de minuta de transporte. Baseado na quantidade de volumes, nas suas informações dimensionais e em parâmetros restritivos o algoritmo vai montar grupos de volumes empilhados e dispô-los dentro do contêiner. O algoritmo genético vai aplicar funções para varrer o espaço de soluções possíveis e ao final, baseado em uma função de avaliação, determinar e demonstrar uma otimizada forma de realizar o empilhamento. O protótipo foi desenvolvido na linguagem java, utilizando a IDE NetBeans na versão 8.2. O banco de dados utilizado é o MySQL versão 5.5. Para modelagem dos diagramas de casos de uso e de atividades da UML foi utilizado o plugin easyUML da plataforma NetBeans IDE 8.2. Os resultados dos testes da solução desenvolvida se demonstraram muito satisfatórios. Conclui-se que a aplicação de algoritmos genéticos, aliado com modelagem de carregamento em pilhas e dispostas utilizando o conceito de corte de chapas apresenta uma boa solução para o problema do empacotamento de volumes. A modelagem utilizada neste estudo, trouxe bons resultados. Durante os testes, as taxas de ocupação de espaço nos contêineres foram superiores a 97% para cargas completas, ou seja, quando a área dos volumes é maior ou próxima à área disponível do contêiner.

Palavras-chave: Algoritmos genéticos. Heurística. 3DBPP. Empacotamento tridimensional. Problemas de corte.

ABSTRACT

This work presents the development of a prototype system to solve the problem of packing three-dimensional volumes in containers, also known as 3DBPP. The goal is create a model in order to solve the problem using genetic algorithms (GA). The GA´s are part of computational metaheuristic and were created using concepts of the theory of species proposed by Charles Darwin. The system will have a list of volumes that need to be packing in a virtual shipment, called `minuta de transportes`. Based on the quantity of bins, in their information dimensions and in parameters restriction, the prototype will create group of bins, called stacks and dispose them inside the container. The genetic algorithm is going to apply the functions to sweep the space of possibilities and at the end, based on the evaluation function to determine and show the best way of packaging. The prototype was developed in java programming language, NetBeans IDE 8.2 version and MySql 5.5 database. Class diagram and use cases were created by easyUML plugin. The results of the resolution as very satisfactory to conclude, the application of genetic algorithms, allied with modeling of bin stacks and the concept of cutting sheets, presents a good solution to the 3D bin packaging problem. The modeling created on this prototype presents good results with occupation tax greater than 97% in full load cases

Key-words: Genetic algorithms. Heuristic. 3DBPP. Three-dimensional packaging. Cutting problems.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D - Bidimensional

3D - Tridimensional

AG – Algoritmo genético

BPP - Bin Packing Problem

CPU - Central Processing Unit

GB - GigaByte

H1 - Método das prateleiras bi ortogonais

HP - Heurística Permutacional

HSSI - Heurística de Suavização de Superfícies Irregulares

HV - Heurística de Volumes

IDE - Integrated Development Environment

OpenGL - Open Graphics Library

PC – Ponto de Canto

PIB - Produto Interno Bruto

PM – Problema da Mochila

RAM - Random Access Memory

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

SC - Simulação de Carga

UCn - Caso de Uso número n

UML - Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Gomes e Ribeiro (2004, p. 11) afirmam que a palavra logística é originária do vocábulo francês *loger*, que significa "alocar". Novaes (2007, p. 35) descreve logística como sendo o processo de planejar, implementar e controlar de maneira eficiente o fluxo e a armazenagem de produtos, bem como os serviços e informações associados, cobrindo desde o ponto de origem até o ponto de consumo, com o objetivo de atender os requisitos do consumidor.

Menchik (2010, p. 17) aponta que na maior parte das indústrias, a atividade de transporte representa um dos elementos mais importantes na composição do custo logístico. Nas nações desenvolvidas o frete costuma absorver cerca de 60% do gasto logístico total, entre 8% a 10% do Produto Interno Bruto (PIB). Para Morales (2007, p. 235), o custo de transporte tende a ser menor se o aproveitamento do espaço de carga for otimizado, ou seja, quanto mais mercadoria puder ser transportada na mesma carga, menos gasto no frete, maior margem de lucro e maior competitividade do produto. Entretanto, otimizar o empilhamento de volumes tridimensionais regulares é algo desafiador para as empresas.

Segundo Correia (1992, p. 170), a designação do problema de empacotamento 3D (de peças tridimensionais) engloba todos os casos em que se pretende selecionar a disposição de diferentes tipos de objetos rígidos, em várias quantidades, num espaço de dimensões fixas, obtendo uma solução que proporcione não só um aproveitamento volumétrico eficiente do espaço, mas também a satisfação de outros parâmetros que poderão condicionar a aplicabilidade da solução obtida. Uma solução utilizada por algumas empresas para empacotamento de contêiner, consiste em criar "contêineres virtuais", ou seja, espaços físicos nos depósitos delimitados por marcações, que simulam o tamanho dos contêineres. Nestes espaços realizam simulações de empilhamento dos volumes a fim de se ter ideia do tamanho do contêiner necessário para acomodar a mercadoria. (MENCHIK, 2010, p. 17).

Segundo Vendramini (2007), uma das formas de simular o empilhamento de volumes é a utilização de programas de computadores, que executam algoritmos metaheurísticos especializados, os quais são capazes de apresentar boas soluções para problemas reais computacionalmente modelados. Um destes algoritmos metaheurísticos são os algoritmos genéticos.

Para Linden (2012b), o algoritmo genético (AG) é uma metaheurística que se baseia em modelos computacionais de processos naturais de evolução como ferramenta para resolver problemas. A técnica de busca de um AG é extremamente eficiente no seu objetivo de varrer

o espaço de soluções e encontrar melhores resultados. Alguns problemas do mundo real não possuem um método matemático de resolução, ou seja, as soluções são complexas e não são exatas (problemas NP-completos). Desta forma, fica evidenciado que os algoritmos genéticos podem ser utilizados no problema do empacotamento de contêineres, considerando que são capazes de processar soluções ótimas a um custo de processamento aceitável.

Diante disto, este trabalho propõe a criação de um protótipo que utilize algoritmos genéticos para auxiliar no empacotamento de volumes regulares, visando otimizar o carregamento de contêineres, reduzindo custos no processo de transporte de mercadorias.

1.1 OBJETIVOS

O objetivo principal deste trabalho é criar um protótipo de sistema que auxilie o carregamento de contêineres com volumes regulares tridimensionais.

Os objetivos específicos do trabalho são:

a) disponibilizar um mecanismo para definir qual é o contêiner ideal para o carregamento, baseado em uma lista de volumes e contêineres disponíveis com seus respectivos tamanhos e pesos;
b) disponibilizar um mecanismo para demonstrar de que forma os volumes devem ser acomodados para melhor otimização do espaço de acordo com o peso, empilhamento máximo e orientação do volume;
c) fornecer uma nova modelagem computacional capaz de resolver o problema de empacotamento de volumes, com a utilização de algoritmos genéticos;
d) comparar a eficácia das soluções apresentadas pelo protótipo com outros métodos apresentados na literatura.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos, sendo que o primeiro capítulo traz a introdução e os objetivos. O segundo capítulo trata da fundamentação teórica, que serve de base para a fundamentação deste trabalho. O terceiro capítulo apresenta os diagramas, técnicas e metodologias utilizadas no desenvolvimento da aplicação, descreve também os resultados obtidos pelo uso da aplicação. Por fim, é apresentada a conclusão obtida por meio da análise das etapas anteriores.

2 FUNDAMENTAÇÃO TEÓRICA

Como forma de contextualizar o problema que será trabalhado, a seção 2.1 traz uma explanação sobre o processo de transporte de mercadorias. A seção 2.2 aborda três problemas clássicos da pesquisa operacional e normalmente relacionados com problemas de carregamento: Problema da mochila, corte de chapas bidimensional e empacotamento de volumes tridimensionais (3DBPP) A seção 2.3 trata dos algoritmos genéticos e por fim, na seção 2.4, são apresentados os trabalhos correlatos.

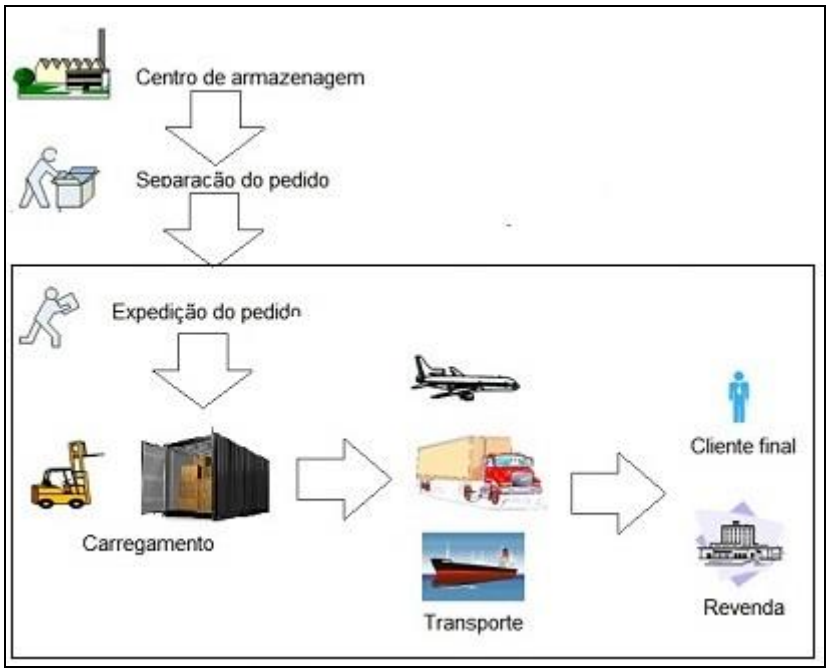
2.1 TRANSPORTE DE MERCADORIAS

Oliveira (2004) destaca a importância de uma estrutura de transportes eficaz para um país com dimensões continentais como o Brasil, com 8.511.965 km^2. A distribuição de cargas deve ser capaz de atender às necessidades de sua economia, alcançando, assim, todas as regiões do país, formando uma logística bem desenhada, que otimize os recursos e meios de transporte, simplificando os processos, visando principalmente três pontos: a redução de prazos de entrega, a redução do estoque e, principalmente, a redução de custos de distribuição, permitindo produtos com preços mais competitivos, o que resultaria no desenvolvimento de toda a economia brasileira

Para Morabito e Arenales (1997), uma escolha adequada do tamanho do contêiner em função da densidade da carga permite que sua capacidade volumétrica seja melhor utilizada. O peso da carga e sua distribuição dentro do contêiner governam as opções de carregamento e casos em que o objetivo é maximizar a utilização do contêiner em termos do volume e do valor da carga carregada. Aspectos de fragilidade e manuseio da carga também podem vir a ser considerados.

Na figura 1 pode se observar um fluxo genérico da logística de transportes de mercadorias. O processo inicia com a separação do pedido de venda no centro de armazenagem. Com a mercadoria separada, inicia o processo de expedição, que consiste no carregamento do contêiner e no transporte por meio terrestre, aéreo ou aquático do mesmo até o cliente final ou revenda. É na expedição da mercadoria que grande parte das empresas faz uso de um documento chamado minuta de transportes.

Figura 1 – Modelo de logística de transportes de mercadorias



Fonte: elaborado pelo autor.

A minuta de transportes é um documento auxiliar que não tem validade fiscal, mas que normalmente acompanha a mercadoria desde o carregamento até o seu destino. A minuta de transportes traz um resumo das informações dos volumes a serem carregados de acordo com o pedido do cliente, além dos dados do transportador.

Figura 2 – Exemplo documento minuta de transporte

MINUTA DE TRANSPORTE

Numero da Minuta: 999999 Cliente: XYZ GLOBAL EXEMPLO DE CLIENTE LTDA

Transportador	Nome Transportador	Data
TRP2BS	TRANSEXEMPLIO TRANSP ENCOMENDAS	01/06/2018

Produtos	Quantidade	Descricao da Embalagem	Peso	Valor mercadoria
MERCADORIA A	1,000	Engradado 50x60x100	41,000	2.238,01
MERCADORIA B	1,000	Caixa Papelao 90x90x50	12,000	3.410,83
MERCADORIA D	1,000	Caixa Papelao 50x50x70	22,000	5.537,02

Eu declaro que recebi os volumes listados na NFF desta minuta de despacho.

Fonte: elaborado pelo autor.

Um dos desafios do setor de expedição é determinar qual ou quais contêiner(es) tem capacidade para carregar os volumes contidos nas minutas de transporte. A seleção do contêiner e a disposição dos volumes em seu interior é o objeto de estudo deste trabalho.

2.2 PROBLEMAS CLÁSSICOS DA PESQUISA OPERACIONAL

Nesta seção serão abortados três problemas clássicos da otimização combinatória, que servem de base para a modelagem do protótipo.

2.2.1 Problema da mochila

Para Lopes (2013, p 88) o problema da mochila (PM) é um problema clássico de otimização combinatória e consiste em organizar n itens com diferentes restrições e lucros dentro de um recipiente limitado (mochila, caixa, contêiner). O objetivo é maximizar o lucro resultante do somatório dos *n* itens carregados no recipiente, respeitando a capacidade máxima.

O PM pode ser classificado de acordo com a dimensão ou número de restrição dos itens que devem compô-lo. Caso o volume a ser acomodado tenha apenas uma restrição tem-se um problema unidimensional. Itens dos quais devemos considerar altura e largura por exemplo, definem um problema bidimensional mais conhecido como problema de corte de chapas. Seguindo esse mesmo raciocínio, ao incluir mais uma dimensão aos volumes, passando estes a terem altura, largura e profundidade, chega-se a um problema tridimensional, no qual podem ser incluídas restrições como a ordem que os volumes devem ser dispostos na mochila ou contêiner, a posição que eles podem ser alocados (em pé, deitado ou de lado), agrupamentos, etc. O PM tridimensional é comumente chamado de problema de empacotamento (navios, contêineres, caminhões ou armazenamento industrial).

2.2.2 – Problema de corte de chapas

Oliveira (2004) explica que problemas de corte são pesquisados desde 1940, embora os estudos com mais relevância tenham surgidos nos anos 60. As pesquisas nesta área apontam para o desenvolvimento de técnicas heurísticas adaptadas para resolução destes problemas da classe Não Polinomial – Complexos. As técnicas matemáticas usadas até o momento atual, não têm capacidade de solução desses tipos de problemas. Gilmore e Gomory publicaram as modelagens e métodos de resolução com maior relevância na literatura.

Para Andrade (2006), problemas de corte bidimensionais consistem basicamente em cortar peças menores a partir de uma peça de dimensões maiores de maneira otimizada. Problemas bidimensionais são comuns em indústrias de aço, papel, alumínio e espuma, onde se busca minimizar a quantidade de material utilizado ou o desperdício de material. Aponta-se ainda como utilidade análoga a atividade de corte, o empacotamento de itens dentro de objetos, como por exemplo o carregamento de contêineres. O empacotamento consiste em

alocar de maneira otimizada peças menores em um recipiente de dimensões maiores. Este último mais especificamente vem a ser objeto de estudo deste trabalho.

2.2.3 Empacotamento de volumes tridimensionais – 3DBPP

O Problema de Empacotamento Tridimensional ou 3D-BPP consiste em empacotar um conjunto de itens retangulares de tamanhos variados no menor número de contêineres possível (ROMAO, 2012). Uma heurística destinada a resolver o problema de empacotamento, deve considerar um conjunto de regras que vise obter um aproveitamento do espaço. Algumas restrições podem ser incorporadas para problemas mais específicos tornando as soluções mais complexas conforme explica Correa et al. (1992).

Silva e Soma (2003) criaram uma representação para o 3D-BPP, a qual pode ser utilizada para modelar o problema proposto neste trabalho.

Como dados de entrada para o 3DBPP temos :

a) um conjunto de n volumes (caixas retangulares), cada um desses itens sendo caracterizados pela largura wj, altura hj e comprimento dj, referenciado por uma tripla ordenada {wj,hj,dj}, ou seja, e.g. {wj,hj,dj} ¹ {hj,wj,dj} (rotações implicam em itens distintos), para todo j Î J = {1, 2, ..., n};
b) os itens não necessitam serem feitos de um mesmo material, muito embora, supõe-se que todos esses possuam uma mesma densidade, são perfeitamente rígidos, além de serem homogêneos;
c) um número limitado de bins (contêineres ou caixas retangulares), onde inicialmente há uma disponibilidade de pelo menos n bins idênticos e tridimensionais, tendo dimensões, internas, comuns: largura W, altura H e comprimento D.

A resolução do 3D-BPP consiste em se empacotar ortogonalmente todos os n itens, utilizando a menor quantidade possível de bins. Outro fator abordado pelos autores é a importância do equilíbrio dos volumes carregados.

2.3 ALGORITMOS GENÉTICOS (AG)

Para falar sobre algoritmos genéticos, é preciso primeiramente sintetizar a teoria da evolução, de Charles Darvin. Para Charles Davin o mecanismo de evolução é uma competição que seleciona os indivíduos mais bem adaptados em seu ambiente podendo assegurar descendentes, transmitindo as características que permitiram sua sobrevivência (LINDEN, 2012b).

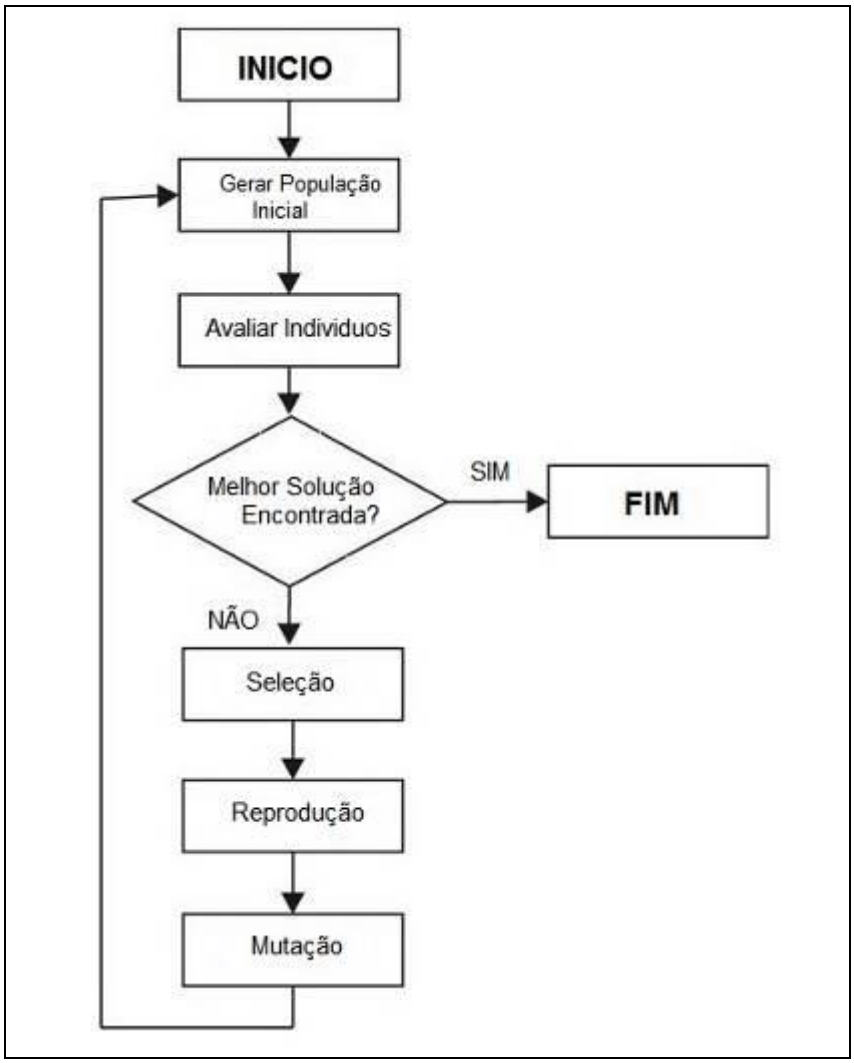
Silva e Soma (2002) descrevem que, baseado na teoria da evolução, John Holland elaborou um método de busca computacional que daria origem ao algoritmo genético. O algoritmo genético é um procedimento que parte de uma população de indivíduos (configuração inicial do problema), faz a avaliação de cada um, de acordo com uma função objetivo, seleciona os melhores e promove manipulações genéticas. As manipulações genéticas como o cruzamento e a mutação, tem por objetivo a criação de uma nova população. As populações encontradas nessas interações, fazem parte do universo de soluções do problema e são avaliadas até que se encontre uma solução considerada ótima.

Para Linden (2012b), algoritmos genéticos em geral são programas simples que necessitam somente de informações locais, não necessitando de derivadas ou qualquer outra informação adicional. O processamento da solução é feito sem a necessidade de interação humana. Os algoritmos genéticos fazem parte do grupo de técnicas conhecidas como metaheurísticas, que são técnicas não determinísticas e vêm sendo usadas com sucesso para encontrar boas soluções para uma ampla variedade de problemas de otimização, desde a sua concepção.

Para melhor entendimento sobre o funcionamento de um algoritmo genético básico, a seguir tem-se uma sequência de etapas (esquematicamente na figura 3):

a) gerar população inicial;
b) avaliar cada indivíduo da população;
c) enquanto critério de parada não for satisfeito faça:
    - selecionar indivíduos mais aptos;
    - reproduzir novos indivíduos aplicando os operadores crossover e mutação;
    - armazenar novos indivíduos em uma nova população;
    - avaliar cada indivíduo da nova população.

Figura 3 - Fluxograma Algoritmo Genético



Fonte: elaborado pelo autor.

Conforme explica Linden (2006a) a etapa 3.1 realiza a seleção de indivíduos dentro da população. Esta seleção deve ser feita de tal forma que os indivíduos mais aptos (isto é, aqueles que têm uma função de avaliação maior) sejam selecionados mais frequentemente do que aqueles menos aptos, de forma que as boas características daqueles passem a predominar dentro da nova população de soluções.

Para Vendramini (2007) os algoritmos genéticos demonstram-se altamente eficazes para resolução de problemas como o empacotamento. Para que seja aplicada uma solução heurística é necessário que o responsável pelo projeto desenvolva uma modelagem que faça com que o problema a ser resolvido forneça dados e se adapte com as características de um esquema de algoritmo genético.

2.4 TRABALHOS CORRELATOS

Neste tópico são apresentados três trabalhos relacionados ao tema de empacotamento de volumes regulares. A seção 2.4.1 apresenta o trabalho desenvolvido por Vendramini

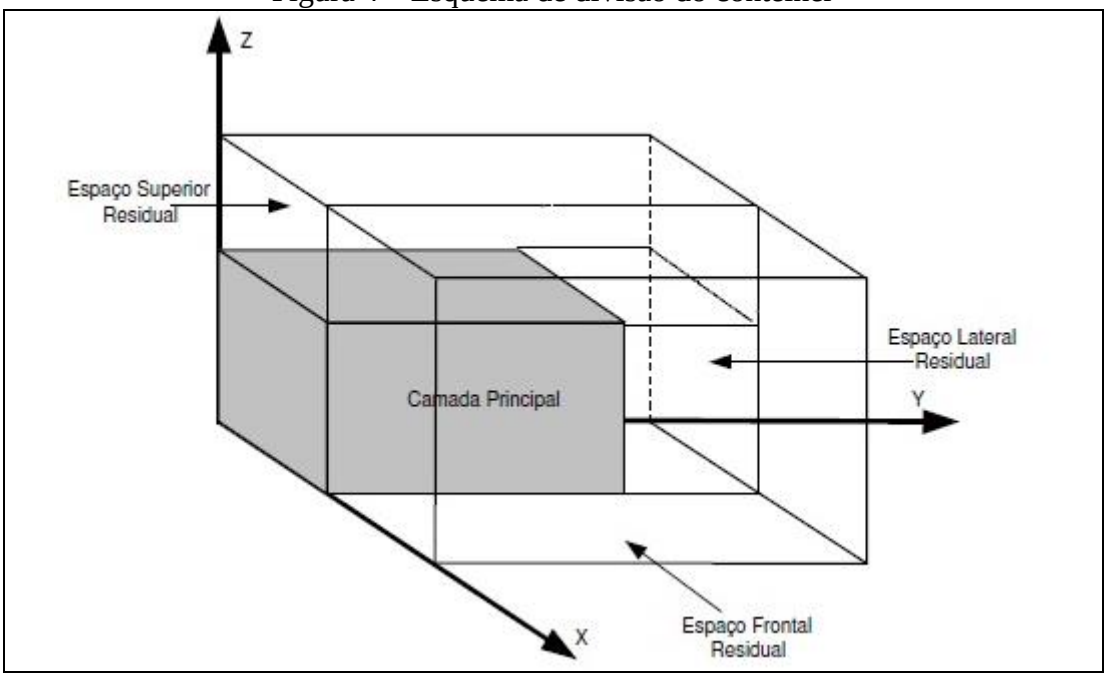
(2007) em sua dissertação de Pós-Graduação, que apresenta uma solução baseada no algoritmo genético Chu-Beasley. A seção 2.4.2 descreve o trabalho desenvolvido por Silva e Soma (2002) que traz uma proposta baseada na Heurística de Volumes (HV). Por fim, a seção 2.4.3 aborda, a aplicação BP3D Interativo (LIBERALINO et al, 2008), que possibilita inputs do usuário no resultado gerado pela heurística do sistema.

2.4.1 Otimização do 3D-BPP utilizando metaheuristica eficiente

Vendramini (2007) apresenta uma proposta de solução para o 3D-BPP, baseado em uma metaheurística especializada conhecida como algoritmo genético Chu-Beasley, tendo como base um algoritmo heurístico e o outro metaheurístico. Ele propõe uma solução para o carregamento de contêineres com caixas regulares de tamanhos e quantidades variados. Ao final é realizada uma comparação entre a solução encontrada por cada algoritmo, visto que ambos trabalham de forma diferente.

No algoritmo Heurístico, o contêiner é supostamente dividido em quatro partes: camada principal, espaço lateral residual, espaço superior residual e espaço frontal residual. O empilhamento da carga do contêiner segue uma ordem, partindo da lateral esquerda para a direita, preenchendo o espaço horizontal, logo após segue o preenchimento na vertical de baixo para cima e em seguida do fundo do contêiner para frente, conforme mostra a Figura 4.

Figura 4 – Esquema de divisão do contêiner

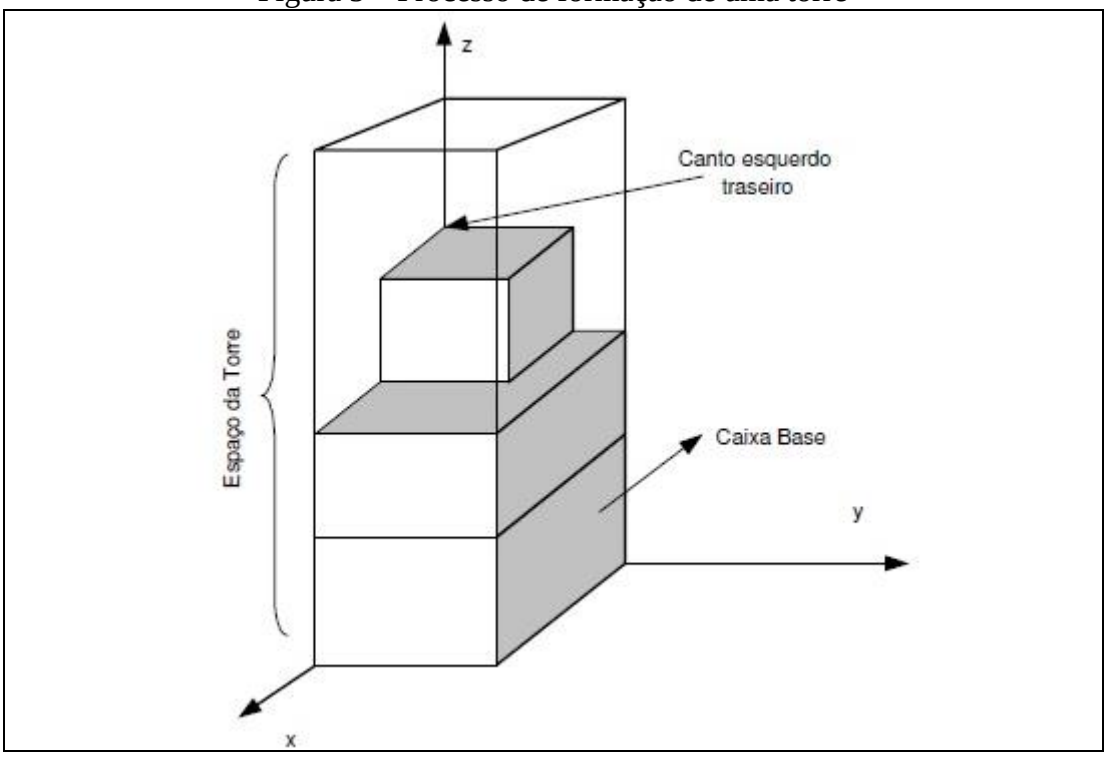


Fonte: Vendramini (2007).

O segundo algoritmo estudado é Chu-Beasley. Este sugere que o carregamento seja feito dividindo em torres formadas pelas caixas. Após esse empilhamento inicial, as torres são

dispostas no contêiner de maneira sistemática seguindo regras de decomposição do espaço a ser alocado, conforme mostra a Figura 5.

Figura 5 – Processo de formação de uma torre



Fonte: Vendramini (2007).

Segundo Vendramini (2007), o algoritmo genético de Chu-Beasley resolveu de forma satisfatória o problema do 3D-BPP, sendo que o resultado para um conjunto de volumes mais homogêneos foi melhor do que quando aplicado para o carregamento de volumes heterogêneos. A aplicação foi desenvolvida na linguagem Fortran 4.0. Nele não foram definidas regras de orientação do volume dentro da carga (rotação), carga máxima suportada pelo volume, equilíbrio das caixas dentro da carga e não existe interação com o usuário na resolução do problema.

2.4.2 Implementação de algoritmo genético no problema do 3D-BPP

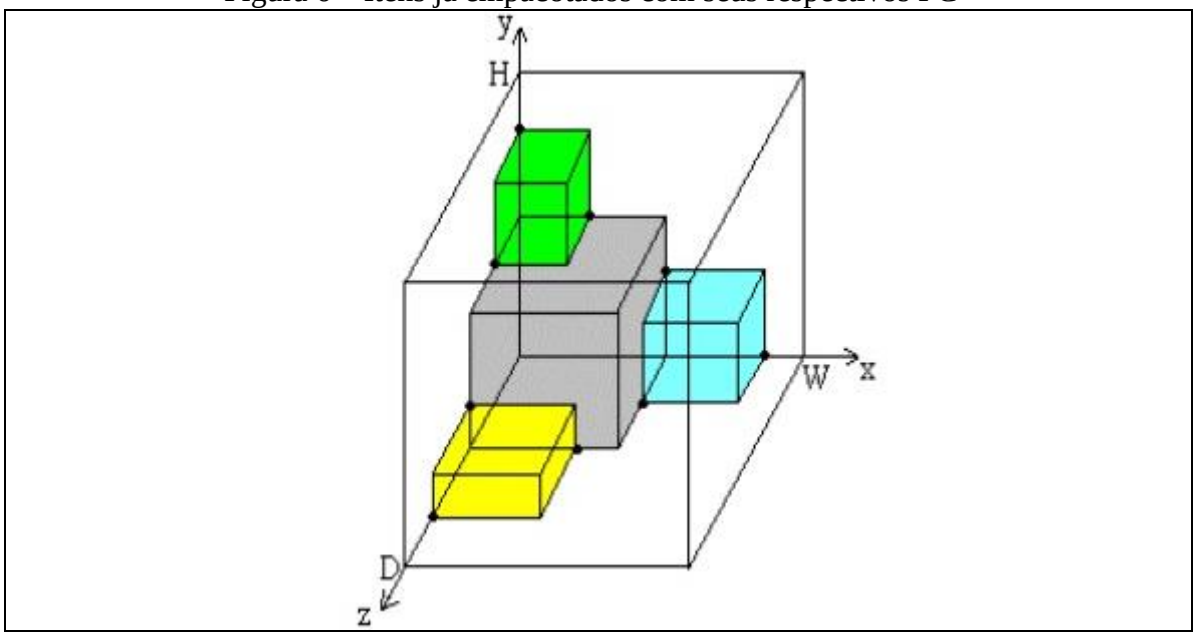
Silva e Soma (2002) descreveram a utilização de algoritmo genético para solucionar o 3D-BPP. Eles fazem um apanhado sobre a capacidade dos algoritmos genéticos resolverem problemas complexos bem como descrevem as etapas e um modelo de AG baseado em adaptações.

O AG desenvolvido por Silva e Soma (2002) utiliza a heurística gulosa chamada de Heurística de Volumes (HV). Ela funciona como operador de mutação, alterando a ordem dos itens na solução, podendo melhorá-la ou não, intensificando a busca por uma boa solução. O objetivo desse método é evitar uma convergência prematura do algoritmo.

Na solução proposta por Silva e Soma (2002), a população não é gerada aleatoriamente, com isso evita que seja gerada uma população com características idênticas a uma criada anteriormente no processo de busca de solução. Outra característica diz respeito ao operador de mutação que não preza pela conservação das características dos pais, fazendo uso dos estudos genéticos que mostram que um indivíduo geneticamente perfeito não é criado pelo processo de mutação, mas sim pelo cruzamento de diversos tipos de indivíduos.

A Figura 6 demonstra o esquema de alocação de itens no contêiner proposto por Silva e Soma (2002). A alocação tem por base a definição de pontos de canto (PC) ou pontos extremos, na imagem representados pelos círculos mais escuros, chamados aqui de "*p*", os quais indicam que nenhum outro volume pode ser inserido no contêiner tendo alguma intersecção à direita de *p*, acima de *p*, ou defronte a *p*, utilizando o esquema de coordenadas cartesianas, representadas por (X_p,Y_p, Z_p).

Figura 6 – Itens já empacotados com seus respectivos PC



Fonte: Silva e Soma (2002).

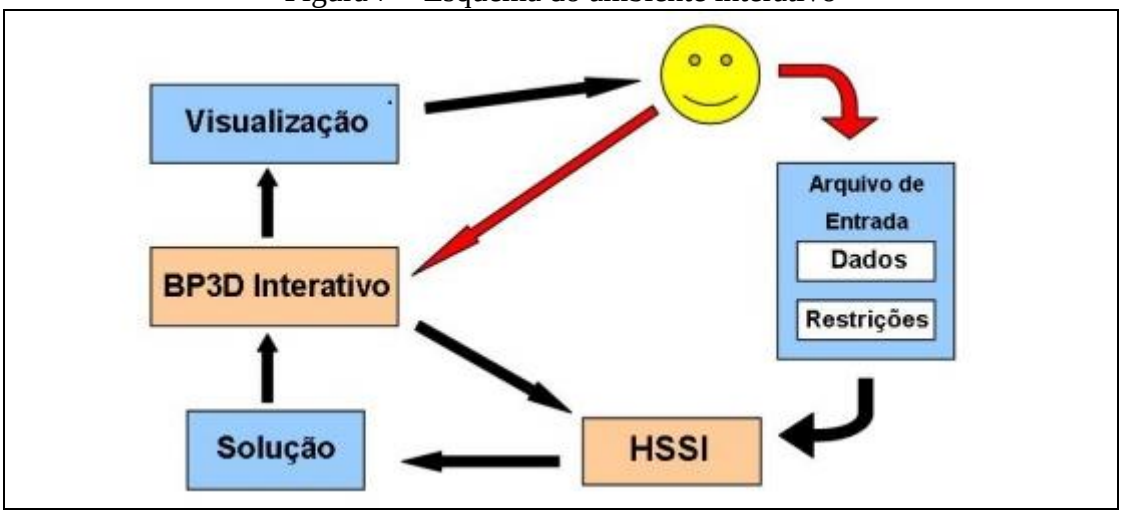
Silva e Soma (2002) demonstraram através de testes uma comparação entre o AG, heurística permutacional (HP), Heurística de Volumes (HV) e H1 (que tem como base a construção de camadas, conhecido como método das prateleiras bi ortogonais). Os resultados demonstram que a Heurística de Volumes (HV) apresenta melhores soluções se comparada com a H1. Sendo assim Silva e Soma (2002) concluem que a HV pode ser aplicada para resolução do 3D-BPP, gerando bons resultados. Neste trabalho não foi abordada a questão da estabilidade do empacotamento, regras de orientações de volumes e carga máxima do volume.

2.4.3 BP3D interativo para resolver o 3DBPP com user hints.

Liberalino et al. (2008) desenvolveram uma aplicação chamada BP3D INTERATIVO para resolver o problema do Bin Packing Tridimensional em contêineres utilizando uma heurística híbrida. A conhecida Heurística de Suavização de Superfícies Irregulares (HSSI) foi utilizada em conjunto com um processo de interação usuário-máquina. Desta forma, a solução apresentada pela HSSI recebe entradas do usuário (*User Hints*). O HSSI recebe as dicas e altera o resultado da solução, aproveitando o conhecimento humano lógico fornecido na interação. A aplicação não leva em conta a questão do equilíbrio no empacotamento e nem utiliza regras de orientação do volume e carga máxima suportada no empilhamento.

Na Figura 7 é possível observar o esquema interativo sugerido por Liberalino et al. (2008). No fluxo, inicialmente o sistema recebe uma parametrização das restrições e a entrada de dados do problema a ser resolvido. Na sequência essas informações são trabalhadas pela Heurística HSSI que propõe uma solução prévia, a qual é visualizada pelo usuário. Neste momento a interface inicia o processo do BP3D Interativo, no qual o usuário faz a inserção de dicas que visam melhorar a solução apresentada. Novamente o sistema inicia o processo de busca de solução pelo HSSI. Este processo se repete até que uma solução considerada ótima seja apresentada.

Figura 7 – Esquema do ambiente interativo



Fonte: Liberalino et al. (2008).

O sistema foi desenvolvido na linguagem C++ com apoio das bibliotecas gráficas OpenGL. A aplicação BP3D interativo surgiu como uma opção de solução para o problema do empacotamento tridimensional. Os ganhos com a interação homem-máquina foram considerados pequenos mas observou uma melhoria nas soluções.

3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo refere-se às etapas de desenvolvimento do protótipo. Na seção 3.1 é tratado sobre os requisitos funcionais e não funcionais. A seção 3.2 apresenta a especificação por meio de diagramas da Unified Modeling Language (UML). A seção 3.3 apresenta detalhes da implementação, destacando as técnicas e ferramentas utilizadas, além da usabilidade do sistema por meio de imagens capturadas durante os testes. Por fim a seção 3.4 apresenta os resultados dos testes obtidos.

3.1 REQUISITOS

O protótipo do sistema de carregamento de volumes regulares em contêineres deverá:

a) permitir o cadastro de parâmetros do algoritmo genético (Requisito Funcional - RF);
b) permitir o cadastro de contêineres (Requisito Funcional - RF);
c) permitir o cadastro de volumes (RF);
d) permitir o cadastro de dados da simulação de carga (SC) com seus respectivos volumes e quantidades (RF);
e) sugerir qual contêiner deve ser utilizado na carga (RF);
f) mostrar em tela a sequência de empilhamento dos volumes dentro do contêiner (RF);
g) mostrar o volume ocupado, o peso total e o volume disponível do contêiner para cada simulação de carga (RF);
h) utilizar o banco de dados MYSQL (Requisito Não Funcional - RNF);
i) ser desenvolvido na Linguagem de programação Java (RNF);
j) utilizar algoritmos genéticos como mecanismo de busca de uma melhor solução (RNF).

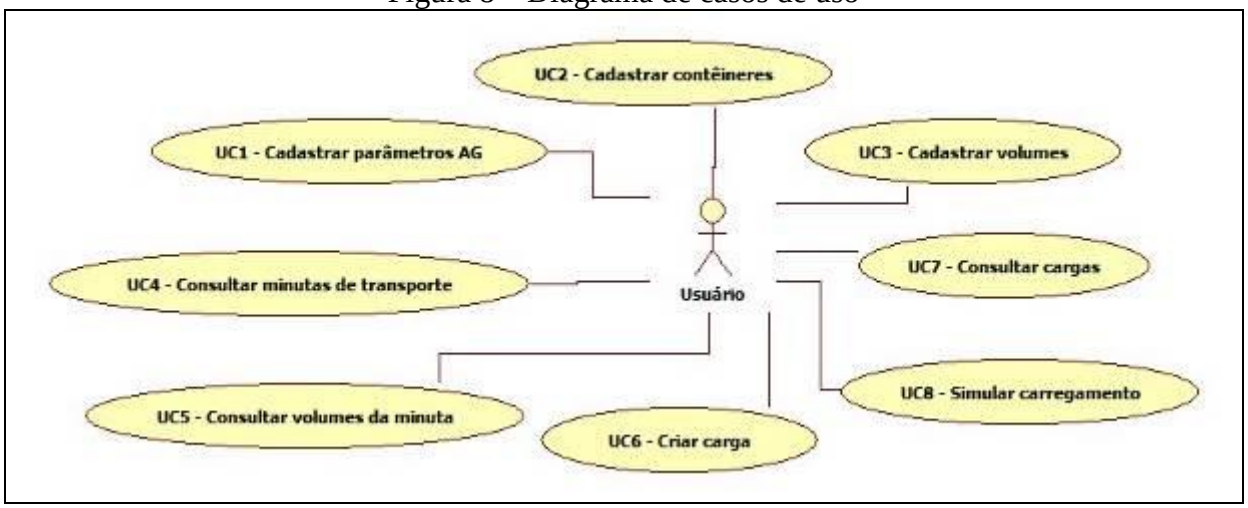
3.2 ESPECIFICAÇÃO

Na especificação do protótipo foram utilizados os diagramas de casos de uso e de atividades da UML, modelados utilizando o plugin easyUML, versão 1.4 na plataforma Netbeans IDE 8.2.

3.2.1 Diagrama de casos de uso

De acordo com o levantamento de requisitos foi criado o diagrama de casos de uso representado na figura 8. Ele representa as ações e funcionalidades disponíveis para o ator chamado Usuário.

Figura 8 – Diagrama de casos de uso



Fonte: elaborado pelo autor.

O caso de uso UC1 – Cadastrar parâmetros AG refere-se à funcionalidade disponível para cadastrar os parâmetros do algoritmo genético tais como:

a) `Número Máximo de Populações`: Um dos critérios de parada do algoritmo do protótipo é o limite de populações definido neste parâmetro.
b) `Tempo Máximo Execução`: O tempo total de processamento em busca da solução também faz parte dos critérios de parada do AG. Este é definido pela unidade milissegundos e é reiniciado para cada novo contêiner necessário para a solução.
c) `Número máximo indivíduos`: Determina o número máximo de indivíduos por população.
d) `Percentual seleção natural`: Este parâmetro determina o percentual de indivíduos que devem ser eliminados durante a criação de uma nova população.
e) `Percentual de Aptos`: Determina o número de indivíduos ou soluções que devem ser mantidos ao se criar uma nova população de soluções.

O caso de uso UC2 – Cadastrar contêineres refere-se à funcionalidade disponível para cadastrar as informações dos contêineres utilizados para o empacotamento dos volumes.

O caso de uso UC3 – Cadastrar volumes refere-se à funcionalidade disponível para cadastrar as informações dos volumes.

O caso de uso UC4 – Consultar minutas de transportes refere-se a funcionalidade disponível para visualizar as minutas de transportes. O caso de uso UC5 – Consultar volumes da minuta refere-se a funcionalidade disponível para visualizar a listagem de volumes e quantidades que compõem uma minuta previamente selecionada. O caso de uso UC6 – Criar carga refere-se à funcionalidade disponível para criar uma carga baseado na listagem de minutas de transportes selecionadas.

O caso de uso UC7 – Consultar cargas refere-se à funcionalidade disponível para consultas as cargas previamente criadas no UC6. As cargas disponíveis nesta consulta podem ser utilizadas no caso de uso UC8 – Simular carregamento, o qual baseado na listagem de volumes e contêineres da carga selecionada, faz a disposição dos volumes dentro do(s) contêiner(es).

3.2.2 Diagrama de atividades

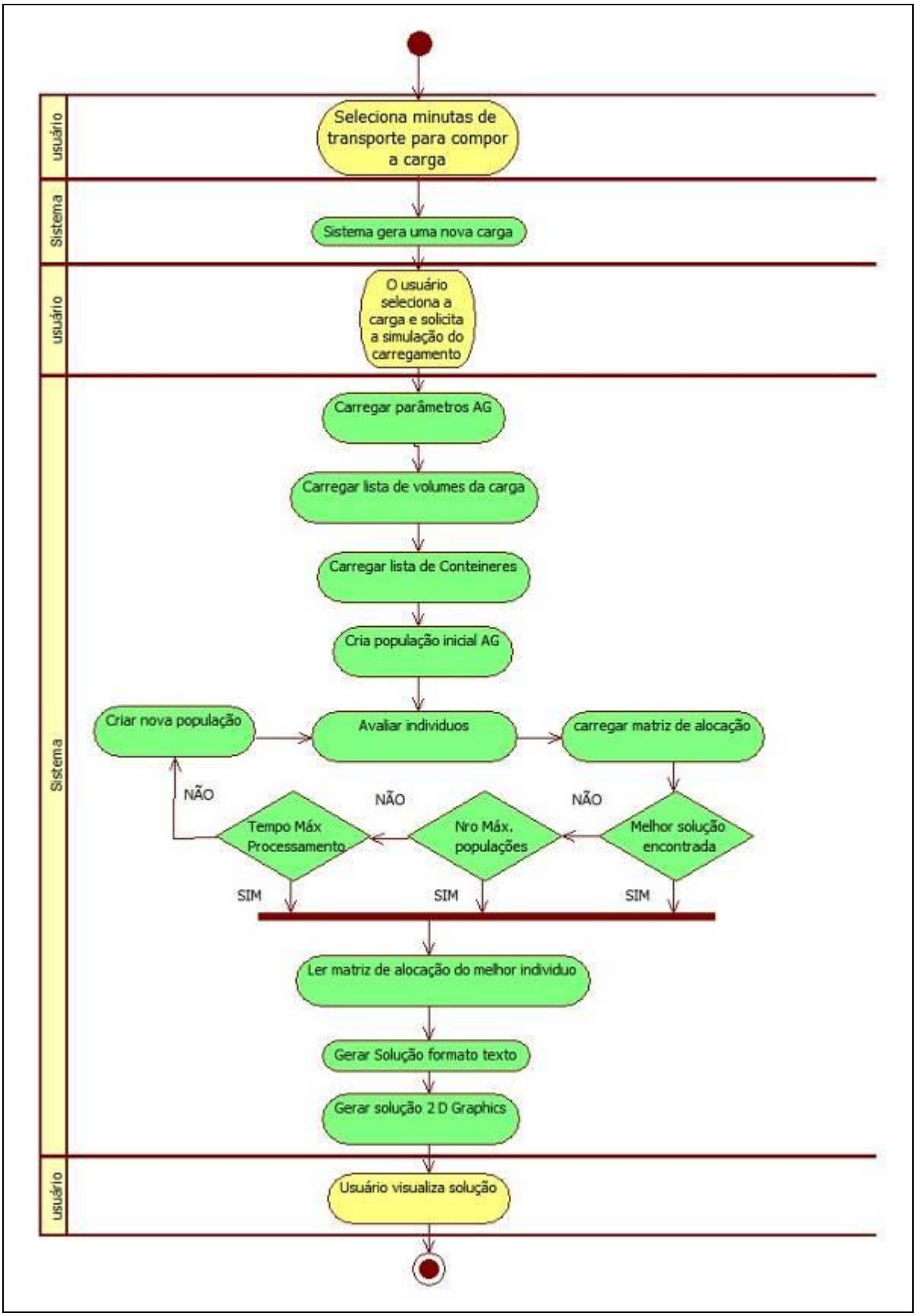
O diagrama de atividades está representado na figura 9. O processo inicia quando o usuário seleciona as minutas de transportes pendentes de processamento para montar uma carga. Com a listagem selecionada pelo usuário, o sistema cria uma nova carga na qual inclui todos os volumes a serem carregados. O usuário após receber a numeração da carga, solicita que seja feito a simulação do carregamento do contêiner. O sistema inicia neste momento a carga dos parâmetros do algoritmo genético. A próxima etapa consiste em carregar os volumes e as informações das dimensões dos mesmos. Feito isso o sistema busca a listagem de contêineres cadastrados. Conforme demonstrado no diagrama e como parte da estrutura básica de um algoritmo, a próxima etapa é fazer a criação da população inicial do algoritmo genético. A partir deste momento o sistema entra em iteração até que alguma das condições de parada seja verdadeira. Etapa seguinte à criação da população inicial é a avaliação dos indivíduos (passo A). O melhor indivíduo na avaliação tem sua matriz de alocação carregada e disponível caso o laço finalize. Três são os critérios de parada definidos no processo conforme segue: tempo máximo de processamento, número máximo de populações e melhor solução encontrada. Se nenhum dos critérios for positivo, o sistema efetua a criação de uma nova população do AG e inicia novamente o passo A.

A melhor solução consiste em um percentual de volume residual próximo de zero. Espaço residual é todo local não ocupado dentro do contêiner, cuja posição esteja à frente da última pilha carregada no contêiner, ou seja, da pilha posicionada mais próximo ao final do contêiner. O cálculo para se chegar ao valor percentual considera o somatório dos espaços

sem ocupação e divide pelo tamanho do contêiner. Durante os testes, devido à variedade de volumes o menor percentual residual encontrado foi 0,075% cuja solução foi considerada ideal. Considera-se então que um parâmetro para determinar o critério de parada baseado na melhor solução seria algo próximo de 0,05% e que diferente disso o sistema deve avaliar os demais critérios de parada.

Ainda sobre os critérios de parada, caso um deles seja positivo o sistema busca o melhor indivíduo avaliado e lê a matriz de alocação que determina de que maneira os volumes devem ser dispostos no contêiner ou contêineres. A próxima etapa é exibir os dados para o usuário no formato texto e no formato gráfico.

Figura 9 – Diagrama de atividades



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas na implementação do protótipo para carregamento de contêineres.

3.3.1 Técnicas e ferramentas utilizadas

O protótipo foi desenvolvido na linguagem Java, utilizando a IDE NetBeans na versão 8.2. Os cadastros básicos são gravados em banco de dados MySQL versão 5.5. Foi utilizado o MySQL Workbench como sistema de gerenciamento e design de banco de dados. Para apresentar a solução no formato gráfico foi utilizado a classe Graphics do pacote java.awt.

O problema do empacotamento de volumes no contêiner foi modelado a partir da criação de pilhas de volumes. Inicialmente se trabalha com um modelo 3D, ou seja, as dimensões base, largura e altura dos volumes e do contêiner são consideradas. Uma vez criadas todas as pilhas, deixa-se a dimensão altura de lado. Desta forma, passa a se considerar uma modelagem 2D, ou seja, analisar a área base das pilhas e a área base dos contêineres. Isto permite utilizar um modelo de solução semelhante ao corte de chapas. O detalhamento dessa modelagem é feito a seguir.

3.3.1.1 Modelagem em pilhas de volumes

A criação de pilhas de volumes faz parte da solução apresentada neste trabalho. Esta abordagem em pilhas foi utilizada por Vendramini (2007), a qual chamou as chamou de torres.

O protótipo faz também a criação das pilhas iniciando pela lista dos volumes que compõe a carga e ordena-os em ordem decrescente da área (base * largura), garantindo que o próximo volume a ser empilhado é menor que o anterior, mantendo o equilíbrio (físico) da pilha. No quadro 1 pode-se observar os comandos e validações para alocar um volume na pilha - alguns trechos de código foram omitidos para melhorar a visualização. Antes de incluir um novo volume na pilha, faz se a validação se a altura do volume somado à altura da pilha, não ultrapassa a altura do contêiner. Além disso, verifica-se o peso suportado por cada elemento da pilha para não haver sobrecarga e danificar os volumes do nível mais baixo. Esse comando está representado na linha 13.

Se as duas condições forem satisfeitas adiciona-se o volume na pilha e recalcula-se a altura total da pilha e o peso máximo permitido da próxima caixa. Na linha 17 é adicionado o volume na lista de volumes da pilha. A linha 19 soma a altura do volume na altura total da

pilha. Na linha 21 é feito a contabilização da altura do volume na altura total da pilha. Caso contrário dá início à criação de uma nova pilha.

Quadro 1 – Código de formação de pilhas

```
1   while (itVol.hasNext()) //lOOP LISTA DE VOLUMES DA CARGA
2   {
3       vol_pilha = itVol.next(); //LER O PROXIMO VOLUME
4       (...)
5
6       (...)
7       //BUSCAR O PESO MAXIMO SUPORTADO PELO VOLUME PARA DETERMINAR A CARGA MÁXIMA DA PILHA
8       if (pesoMax > vol_pilha.getCargamax())
9           pesoMax = vol_pilha.getCargamax();
10
11      (...)
12      // VALIDA O LIMITE DE ALTURA DA PILHA E O PESO MAXIMO SUPORTADO PELO VOLUME ATUAL
13      if (( (altura_pilha+vol_pilha.getAltura())<= alturaConteinere ) && (pesoMax >= SomaPeso))
14      {
15          (...)
16          //ADICIONA O VOLUME DA PILHA
17          clsPilha.setVolumes(vol_pilha);
18          //SOMA A ALTURA DO VOLUME NA ALTURA TOTAL DA PILHA
19          altura_pilha+=vol_pilha.getAltura();
20          // SOMAR O PESO DO VOLUME NO PESO TOTAL DA PILLHA PILHA
21          SomaPeso+= vol_pilha.getPeso();
22          (...)
23      }
24
25      (...)
26  }//FIM DO LOOP
```

Fonte: elaborado pelo autor.

Esse conceito de pilhas simplifica a modelagem do sistema. Uma vez criadas as pilhas, o modelo não precisa se preocupar verticalmente com o carregamento, pois o que importa para o algoritmo é o volume base da pilha e sua área e orientação.

3.3.1.2 Criação de pilhas por contêiner

Um dos diferenciais deste trabalho diz respeito a aleatoriedade dos contêineres a serem carregados. Isso significa que o volume dos itens a serem carregados vai determinar, em tempo de execução, os contêineres que melhor acomodam esses volumes. Isso é feito baseado no volume das caixas bases das pilhas e no volume da área do contêiner. As caixas bases são as primeiras caixas ou volumes da pilha. Elas estão no nível mais baixo e recebem acima delas os demais volumes que formam a pilha. A parte do código responsável por determinar quais pilhas cabem no contêiner ou qual o melhor contêiner está representado no quadro 2, sendo que alguns trechos foram subtraídos para facilitar a visualização.

Inicia-se o processamento de escolha de contêiner lendo a lista de volumes da carga, conforme linha 3. Para cada nova pilha lida (linha 3) o sistema busca a caixa base (linha11). Na linha 15 o sistema passa a área da caixa base e o total das áreas já processadas para um método que retorna o contêiner chamado de `retornaConteinere`. Esse método está especificado nas linhas 62 a 99 e inicia com uma leitura via result set conforme linha 71. O

resultado dessa leitura é adicionado a uma estrutura de lista conforme linha 78. Na linha 87 percorre-se a lista de contêineres a fim de encontrar um que suporte a área das pilhas. Essa verificação é feita na linha 94. Caso nenhum contêiner satisfaça a condição retorna o de maior capacidade, conforme linha 96. Com o contêiner retornado no método a verificação na linha 18 valida se a área total da pilha mais o somatório das pilhas já processadas são menores que a capacidade suportada pelo contêiner. Caso a condição seja verdadeira, na linha 23 a pilha é adicionada na lista de pilhas a serem carregadas e na linha 25 a pilha é excluída da listagem geral de pilhas. Se a condição não for verdadeira (linha 23), cria-se um novo contêiner conforme linha 36, percorre-se a lista de pilhas conforme linha 44. A pilha é adicionada ao contêiner na linha 50.

Quadro 2 – Código escolha contêiner

```
            (...)
            //LOOP DE PILHAS CRIADAS COM TODOS OS VOLUMES DA CARGA
            while (itListPilhas.hasNext() )
            {
                ClsPilha clspilha = new ClsPilha();

                //LER DADOS DA PILHA
                clspilha = itListPilhas.next();

                //BUSCAR A CAIXA BASE DA PILHA
                clsvolumes = clspilha.getFirstVolume();

                //SOLICITA CONTEINER COM CAPACIDADE DE ARMAZENAR A PILHA ATUAL
                // A VARIAVEL AreaTotPilha É O SOMATORIO DAS AREAS DA CAIXA BASE DAS PILHAS JA PROCESSADAS
                clsconteineres = retornaConteinere(AreaTotPilha+clsvolumes.getArea());

                //VERIFICAR SE  O VOLUME ATUAL SOMADO AO TOTAL DA AREA DA PILHA É MENOR QUE A ÁREA DO CONTEINER
                if (clsconteineres.getCargamax() >= (AreaTotPilha+clsvolumes.getArea()))
                {
                    //SOMAR A AREA DO VOLUME AO TOTAL DA AREA DA PILHA
                    AreaTotPilha+= clsvolumes.getArea();
                    //ADICIONA A PILHA NA LISTA DE PILHAS A SEREM INSERIDAS NO CONTEIINER
                    listPilhaCont.add(clspilha);
                    //REMOVE A PILHA DA LISTA TOTAL DE PILHAS
                    itListPilhas.remove();
                }
                else
                {
                    //AQUI ATINGIU O MAXIMO DA CAPACIDADE DO CONTEINER, DEVE CRIAR UM NOVO OBJETO DE  PILHAS POR CONTEINER
                    // PROCESSAR O AG PARA CARREGAMENTO

                    /INCREMENTAR A SEQUENCIA DO CONTEINER
                    seqconteiner++;

                    //CRIAR NOVO OBJETO CONTEINER
                    clsPilhaConteinere = new ClsPilhaConteinere();
                    (...)

                    //PERCORRER A LISTA DE PILHAS E ADICIONAR NO NOVO CONTEINER

                    Iterator<ClsPilha> itclsPilha = listPilhaCont.iterator();

                    //LOOP DE PILHAS A SEREM INSERIDAS
                    while(itclsPilha.hasNext())
                    {
                        ClsPilha clsPilha = new ClsPilha();
                        //LER DADOS DA PILHA
                        clsPilha = itclsPilha.next();
                        //ADICIONAR A PILHA AO CONTEINER
                        clsPilhaConteinere.setPilhaCont(clsPilha);
                        //REMOVE PILHA DA LISTA
                        itclsPilha.remove();
                    }
                    (...)
                    //PROCESSAR AG DO CONTEINER
                    lisGenes = processaPilhasAG(clsPilhaConteinere);
                    (...)
                }
            }

        public ClsConteineres retornaConteinere(double areaPilhas)
        {
            ClsConteineres conteiner = null;

            (...)
            //RESULT SET DE CONTEINERS ORDENADOS PELO VOLUME MÁXIMO DE CARGA
            //PROCEDIMENTO REALIZADO UMA UNICA VEZ, NA PROXIMAS CHAMADAS
            //VAI UTILIZAR O LIST DE CONTEINERS

                rs_conteinere = daoConteinere.conteinere_retorna_resultset("select * from tb_conteinere order by volume_max asc");

                //PERCORRER O RESULT SET DE CONTEINERES
                while (rs_conteinere.next())
                {
                    (...)
                    //ADICIONAR OS CONTEINERS A UMA LISTA
                    ListaConteiner.add(conteiner);
                    (...)
                }

            //CRIAR ITERATOR COM A LISTA DE CONTEINERS
            Iterator<ClsConteineres> itCont = ListaConteiner.iterator();


            //PERCORRER LISTA DE CONTEINERES
            while ( itCont.hasNext() )
            {
                (...)
                //LER DADOS DO CONTEINER
                conteiner = itCont.next();

                //SE A AREA DAS PILHAS FOR MENOR QUE A CAPACIDADE DO CONTEINER RETORNA O CONTEINER
                if ( areaPilhas <= (conteiner.getCargamax()) )
                    return(conteiner);
            }
            // SE TODA A LISTA DE CONTEINER FOI PERCORRIDA RETORNA
            //O ULTIMO CONTEINER , COM MAIOR CAPACIDADE, PARA CARREGAMENTO
            return(conteiner);
        }
```

Fonte: elaborado pelo autor.

3.3.1.3 Genes, indivíduos e populações

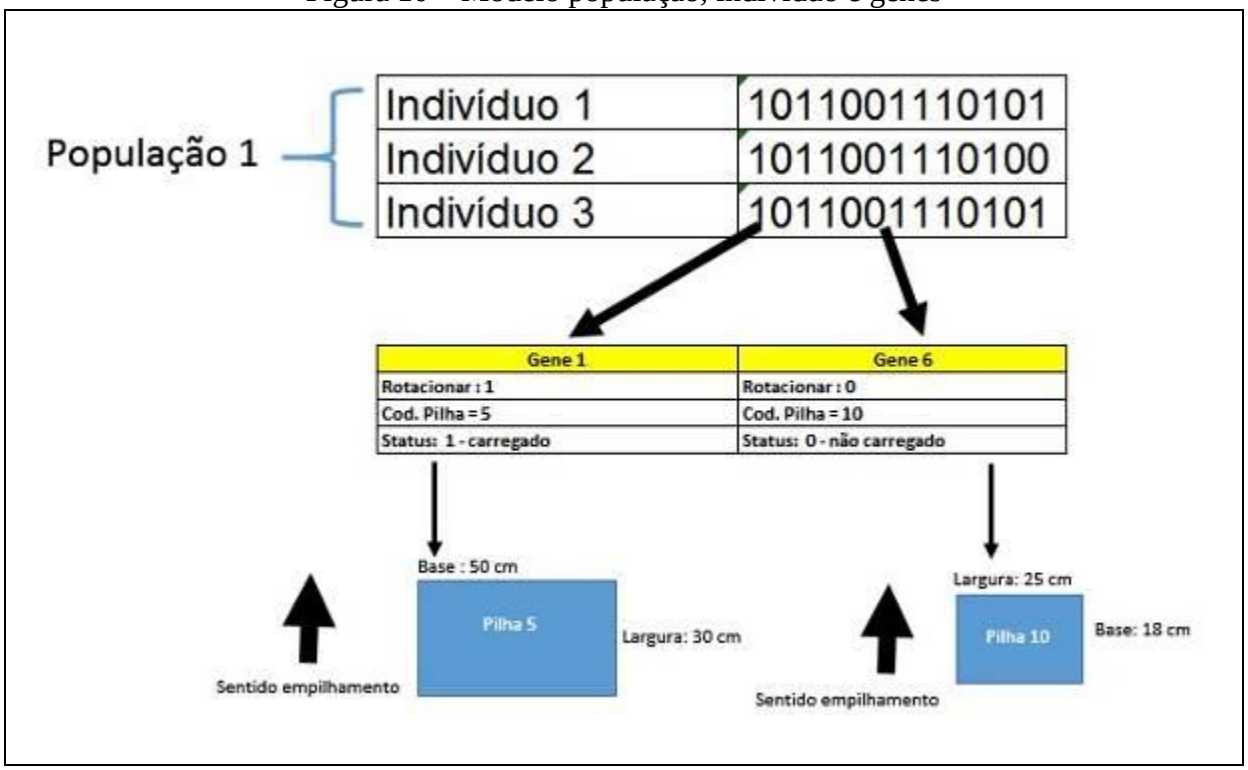
Nesta seção é feito a definição das estruturas que fazem parte do algoritmo genético e uma explanação sobre os operadores genéticos.

Abaixo fica demonstrado a definição dos termos do AG que será implementado neste trabalho:

a) Genes ou cromossomos – serão representados pelas pilhas de volumes.
b) Indivíduos – São compostos por uma lista de genes. O indivíduo pode ser interpretado como uma solução para o carregamento.
c) População: é um conjunto de indivíduos processadas pelo algoritmo genético a cada nova geração.

Por padrão computacional os genes são representados por valores binários, inteiros ou reais. Neste trabalho, será relacionado cada volume base da pilha a um gene e atribuido a ele o valor de 0 ou 1, sendo que 0 (zero) indica que a pilha não será rotacionada e 1 (um) indica que deve rotacionar no momento do carregamento. Como convenção, foi definido que o valor 0, indica que a pilha segue o padrão base x largura e será carregado neste sentido. O valor 1, indica que a pilha será empilhada baseado na orientação largura x base. Cada gene contém um vetor de informações, sendo elas a orientação, o código da pilha e o campo de status (0 – não coube no contêiner e 1 coube no contêiner) conforme figura 10. A criação dos indivíduos parte da listagem de genes a serem alocadas no contêiner. Para determinar a posição do gene dentro dos indivíduos, o sistema utiliza uma função randômica entre 0 (zero) e o número total de genes, de acordo com o retorno dessa função os genes são incluídos na listagem do indivíduo. Os parâmetros do AG definem a quantidade de indivíduos a serem criados. A população por sua vez vai ser criada a cada iteração do AG enquanto não se encontre uma solução para o problema ou satisfaça os critérios de parada. O número máximo de populações também é definido por parâmetro. Uma população representa um conjunto de soluções para resolver a carga e nela temos indivíduos(soluções) com mais aptidão ou menos aptidão para a resolução.

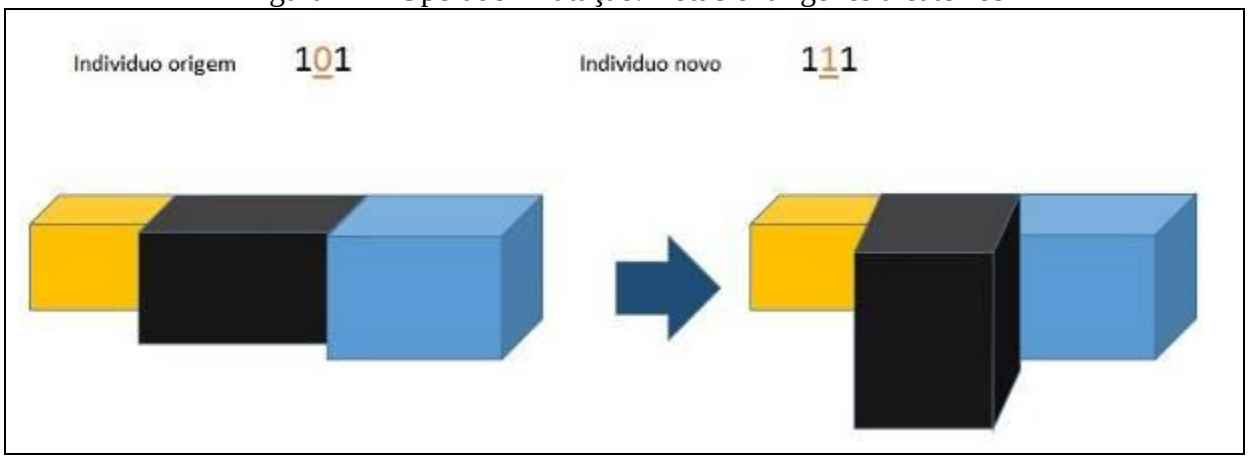
Figura 10 – Modelo população, indivíduo e genes



Fonte: elaborado pelo autor.

Durante o processo de evolução das populações, utiliza-se dois tipos de operadores genéticos do algoritmo genético: `mutação` e `crossover`. Mutação consiste em alterar o valor da cadeia de genes do indivíduo. Utiliza-se dois operadores de mutação, um deles consiste em alterar a rotação de pilhas de forma aleatória conforme figura 11.

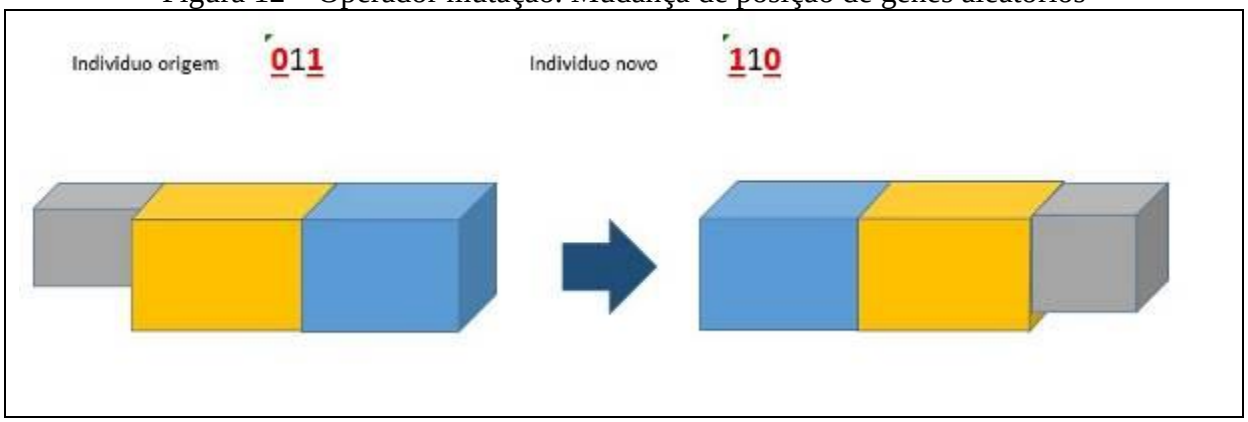
Figura 11 – Operador mutação. Rotacionar genes aleatórios



Fonte: elaborado pelo autor.

O segundo operador de mutação é executado a partir da troca de genes (pilhas) dentro do mesmo indivíduo. Para isso seleciona-se de forma randômica dois genes e eles trocam de posição dentro do indivíduo conforme figura 12.

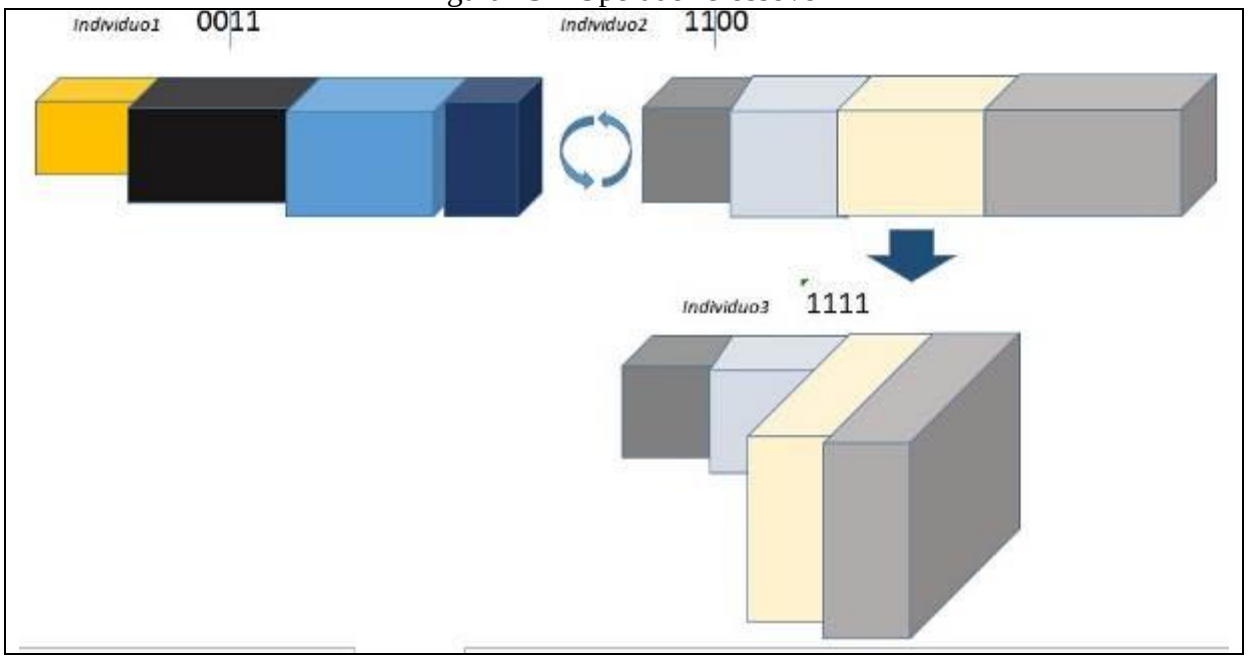
Figura 12 – Operador mutação. Mudança de posição de genes aleatórios



Fonte: elaborado pelo autor.

O operador de crossover consiste em realizar a troca de genes entre indivíduos. Nesta a abordagem não será o gene por completo, apenas ocorre a troca de informação genética sobre a orientação das pilhas entre dois indivíduos, criando a partir disso um terceiro. Para iniciar o processo, será feito a seleção de dois indivíduos, `individuo1` e `individuo2`. Depois é copiado todo o conteúdo genético do `indivíduo2` para o novo `indivduo3`, dessa forma a cadeia de genes ou pilhas do `individuo2` servirá de base para o `indivíduo3`. Na próxima etapa é feito a segmentação do `indivíduo1` e `indivíduo3` ao meio. Para a primeira metade de pilhas mantem-se as informações do `individuo3`, a outra metade terá mudança no seu material genético, recebendo dados do `individuo1`. Este processo está demonstrado na figura 13.

Figura 13 – Operador crossover



Fonte: elaborado pelo autor.

Pode-se perceber no exemplo da figura 13 que o novo individuo formado (`indivíduo3`) herda características do `indivíduo1` e `indivíduo2`.

3.3.1.4 Alocação das pilhas no contêiner

A cada iteração do algoritmo genético, ou seja, a cada novo indivíduo criado, o sistema faz a avaliação da capacidade do mesmo como possível solução do problema. Um indivíduo com uma boa avaliação indica que sua cadeia de pilhas pode ser perfeitamente condicionada no contêiner, de maneira que o espaço residual, se existir, é menor que todas as outras soluções encontradas. Para realizar o teste de alocação do indivíduo dentro do contêiner é utilizado uma matriz bidimensional. O tamanho da matriz de alocação, é determinado pela base do contêiner e pela largura do contêiner. Sendo que a largura em centímetros vai determinar as colunas da matriz e a base em centímetros determina as linhas. Todo o espaço do contêiner fica representado em um plano cartesiano bidimensional, por este motivo referenciamos o problema do corte de chapas anteriormente.

Cada posição da matriz de alocação pode receber dois valores 0 (zero) ou 1 (um). Todo valor zero encontrado na matriz indica que nenhuma pilha está ocupando aquele espaço. Quando se observa o valor 1 (um) em uma determinada posição, significa que aquela posição já está sendo utilizada por alguma pilha e nesta posição não se pode alocar nenhuma outra pilha.

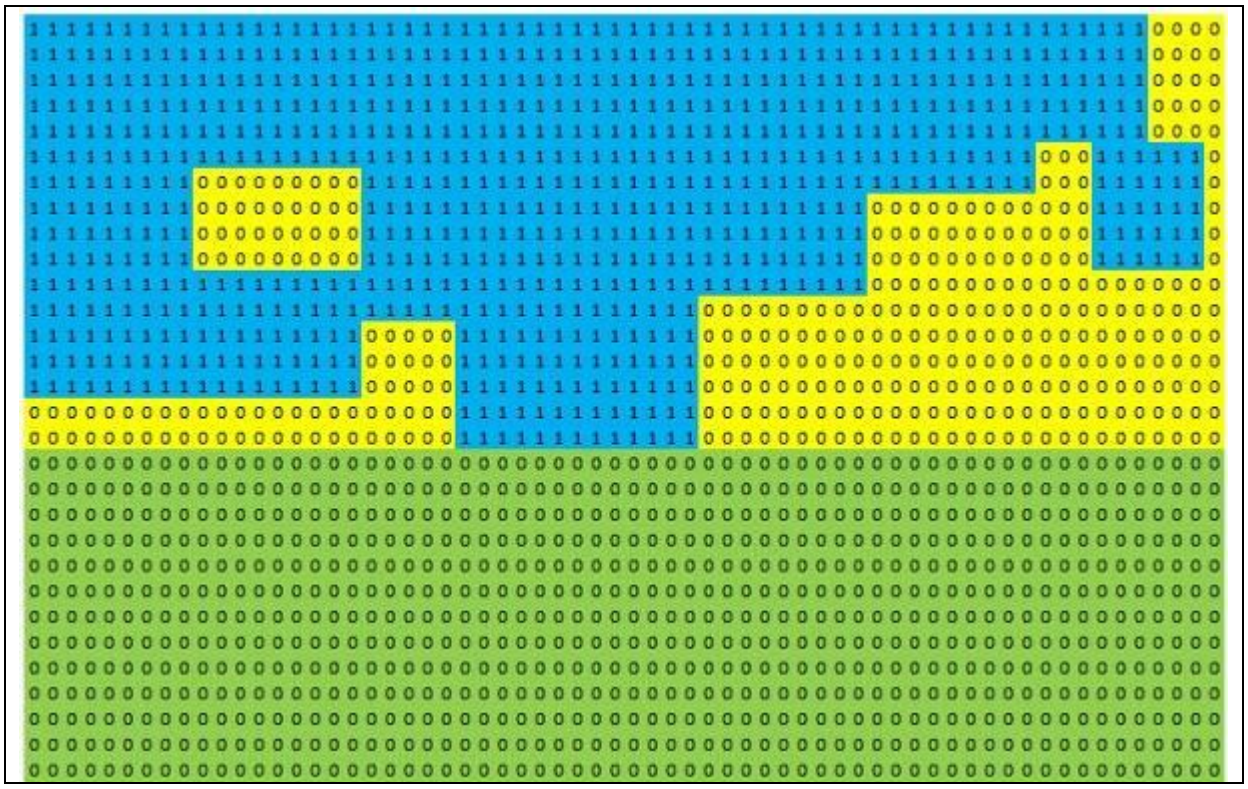
Inicialmente a matriz de alocação é criada e inicializada com zero para todas as posições, indicando que o contêiner está vazio, conforme figura 14.

Figura 14 – Matriz de alocação de um contêiner vazio

Fonte: elaborado pelo autor.

Para cada gene do indivíduo que precisa ser carregado no contêiner percorre-se as colunas da matriz de alocação da esquerda para a direita e as linhas do início ao final da matriz até encontrar uma posição não carregada, ou seja, com o valor zero. Ao encontrar essa posição o algoritmo realiza um teste para verificar se o gene cabe na matriz a partir dessa posição. Para isso ele percorre as posições (colunas e linhas) para o tamanho da base e largura do contêiner em centímetros, validando se todas estão disponíveis. Caso esse intervalo percorrido apresente alguma posição com 1, ou seja, já ocupado, o sistema dispara nova busca, partindo dessa nova posição. Este processo se repete até que todas as condições sejam satisfeitas, ou seja, até que haja um espaço disponível para carregamento ou que todas as posições da matriz de alocação sejam percorridas indicando que aquele gene não pode ser carregado. A figura 15 representa uma matriz de alocação ou contêiner em fase de carregamento.

Figura 15 – Matriz de alocação de um contêiner carregado



Fonte: elaborado pelo autor.

Na figura 15, há três áreas destacadas na matriz de alocação. A área em azul, com valores 1, representa o espaço do contêiner ocupado pelas pilhas de volumes. As regiões em verde e amarelo, indicam as áreas livres dentro do contêiner, ou seja, onde não foram incluídas nenhuma pilha ou volumes. A área em amarelo foi destacada para demonstrar o que o sistema considera de espaço residual para fins de avaliação do indivíduo. Para cada indivíduo que passa por esse processo de análise de alocação, o sistema atribui uma nota baseado no percentual de espaço residual. Quanto menor o espaço residual mais apto é o indivíduo ou solução encontrada. A tabela 1 representa uma população de indivíduos avaliados. O exemplo de cálculo de percentual residual está demonstrado no quadro 3.

Quadro 3 – Calculo percentual residual

Considerando as dimensões de um contêiner de 10 pés conforme abaixo:

`Base` = 2.99 m

`Largura` = 2.50 m

Inicialmente, é feita a conversão da unidade metros para centímetros. Após é feito o cálculo da `área total` do contêiner:

```
area total = base * largura
```

Substituindo os valores tem-se:

`area total` = 299 * 250 = 74750 cm^2.

Suponha para fins de exemplificação que após o carregamento do contêiner encontram-se 150 posições destacadas como `espaço residual`. O cálculo do `percentual residual` utilizado para avaliação do indivíduo vai ser:

`percentual residual` = (`espaço residual` * 100) / (`area total`);

Substituindo os valores:

`espaço residual` = 150

`area total` = 74750

`percentual residual` = (150 * 100) / (74750) = 0,2007 %.

O valor de 0,2007% será a nota que o indivíduo irá receber como avaliação.

Fonte: elaborado pelo autor.

Tabela 1 – Indivíduos e percentuais de espaço residual

1	Individuo Código	Percentual residual
2	9	0,07528
3	1	0,08993
4	2	0,09763
5	18	0,12554
6	15	0,15554
7	14	0,15772
8	16	0,18554
9	19	0,19554
10	20	0,22554
11	17	0,25554
12	3	1,37623
13	12	1,58273
14	13	2,53726
15	10	2,61864
16	4	2,65482
17	5	3,93341
18	11	4,31256
19	6	5,21201
20	7	6,49066
21	8	7,76928

Fonte: elaborado pelo autor.

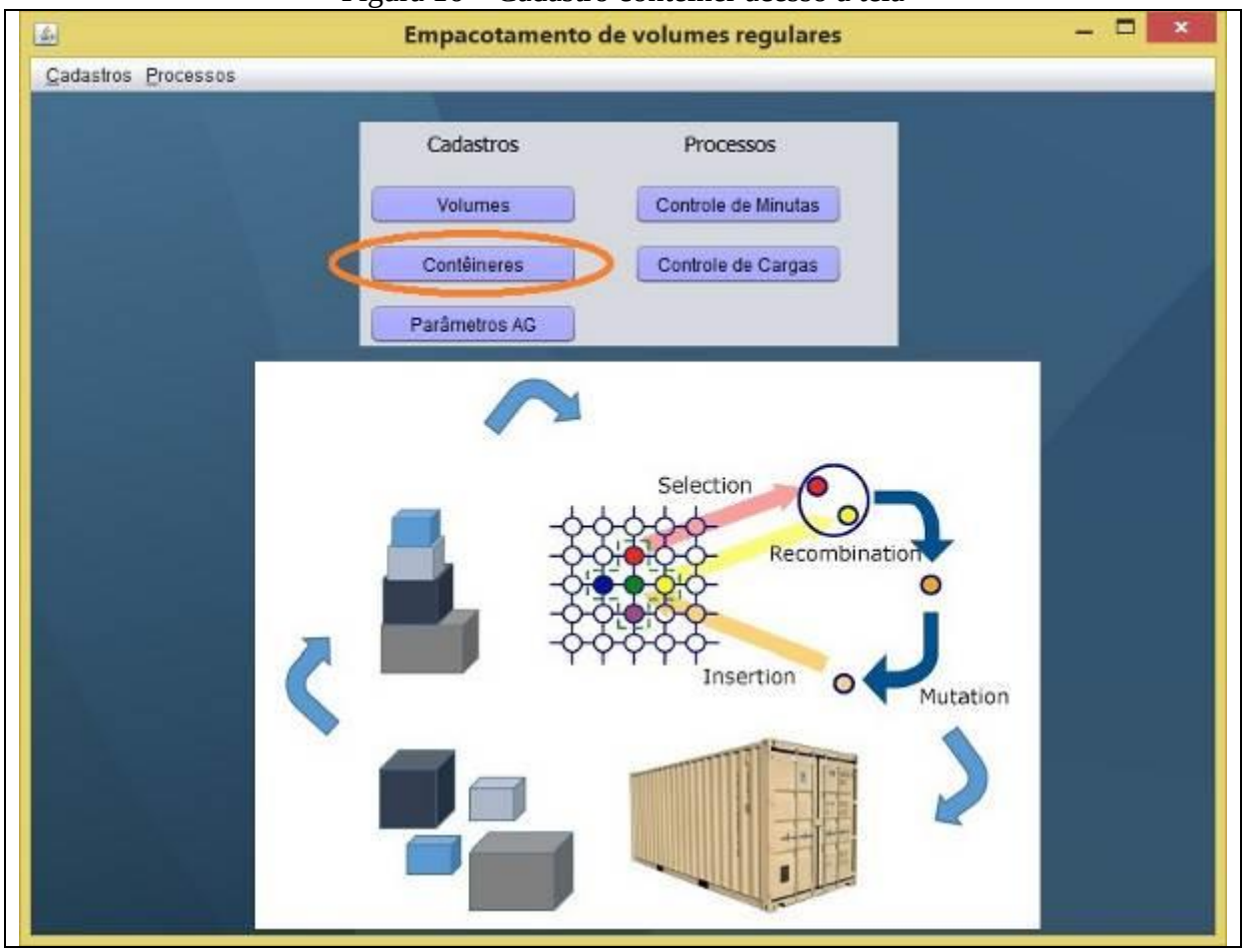
A cada população criada no AG é feito a ordenação dos indivíduos por ordem crescente de aptidão. A linha 1 da figura 18 destacada em verde representa o indivíduo ou

solução mais bem avaliado para resolver o carregamento. A linha 21 destacada em vermelho representa a pior solução encontrada pelo AG.

3.3.2 Operacionalidade da implementação

Nesta seção é apresentada a operacionalidade e funcionamento do protótipo. Será utilizado um exemplo no qual o usuário realiza o cadastro de um contêiner e seleciona algumas minutas de transportes para criação de uma carga. Na sequência o usuário realiza a simulação da carga e o sistema exibe o resultado obtido. Nas figuras 17, 18, 19 e 20 é apresentado o passo a passo para cadastramento de um contêiner. Inicialmente o usuário acessa o menu principal e clica no botão `Contêineres`, conforme a figura 16.

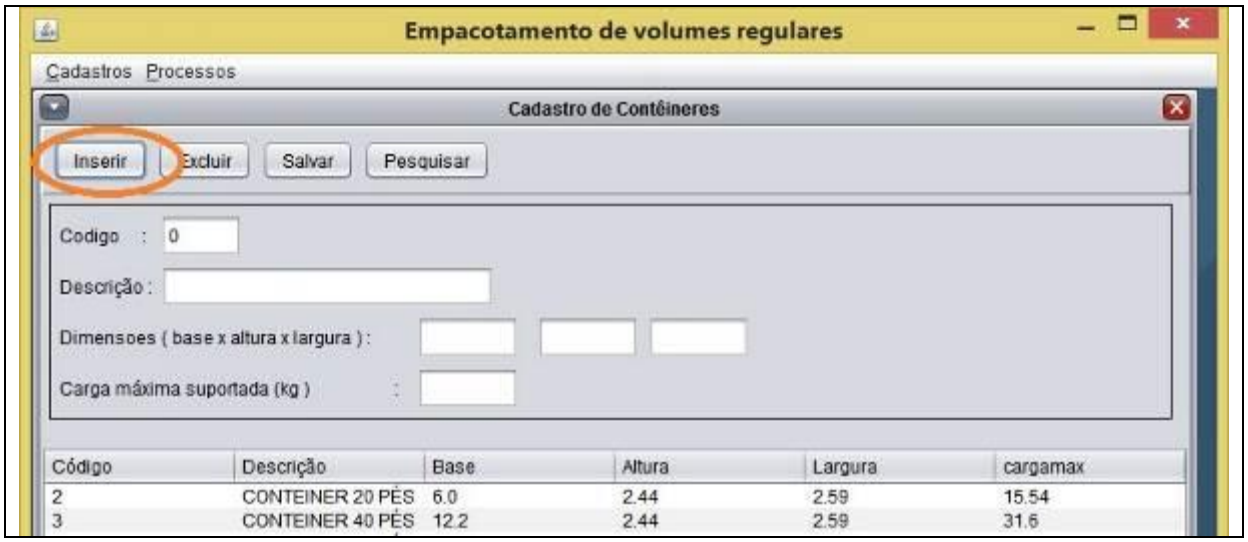
Figura 16 – Cadastro contêiner acesso a tela



Fonte: elaborado pelo autor.

Na figura 17 pode-se observar a tela de cadastro de contêineres no momento que o usuário clica no botão inserir.

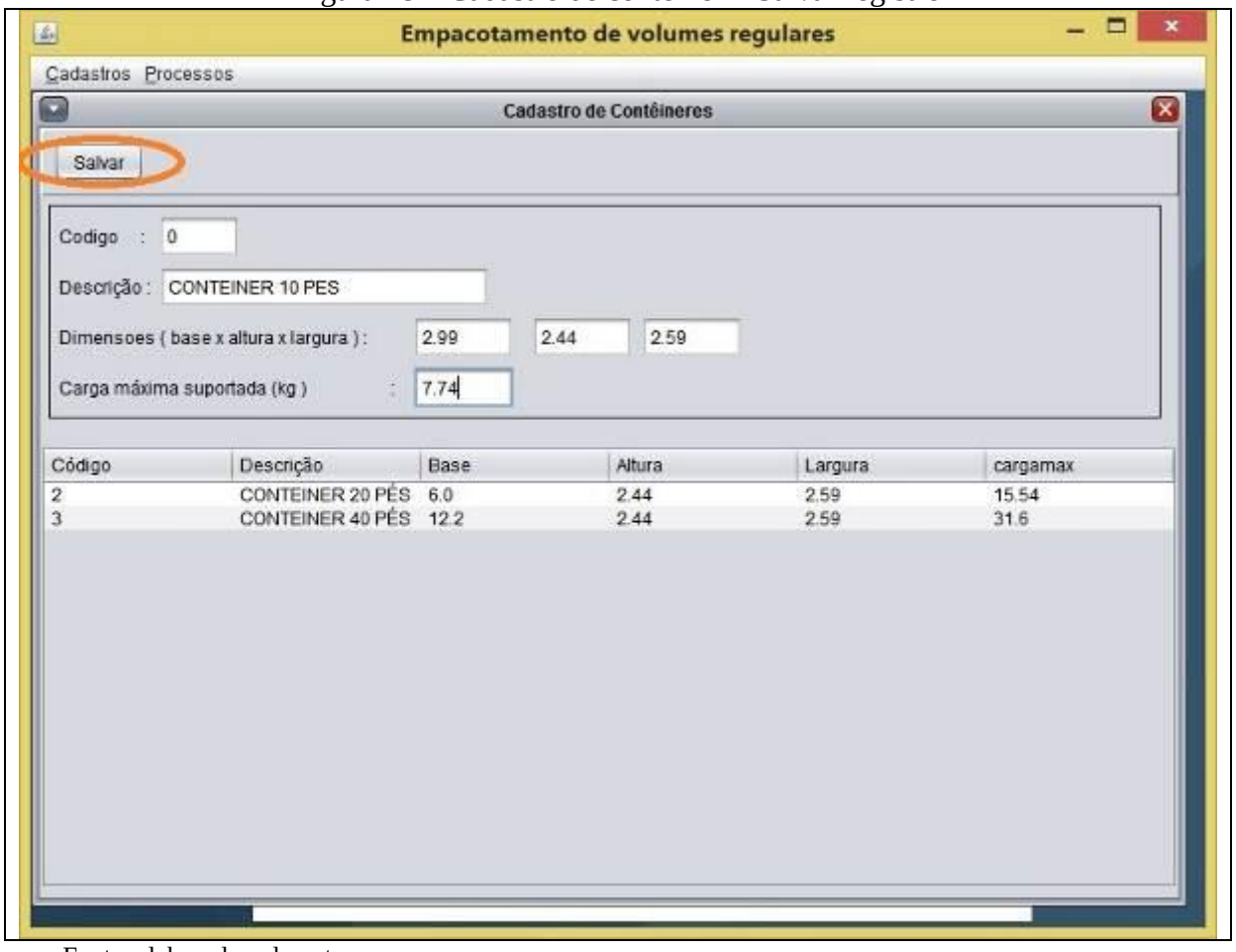
Figura 17 – Cadastro de contêiner inclusão



Fonte: elaborado pelo autor.

A figura 18 demonstra a tela de cadastro de contêineres com os campos referentes ao cadastro de contêineres tais como descrição, dimensões e carga máxima preenchidos pelo usuário e a ação de salvar o registro, clicando no botão salvar.

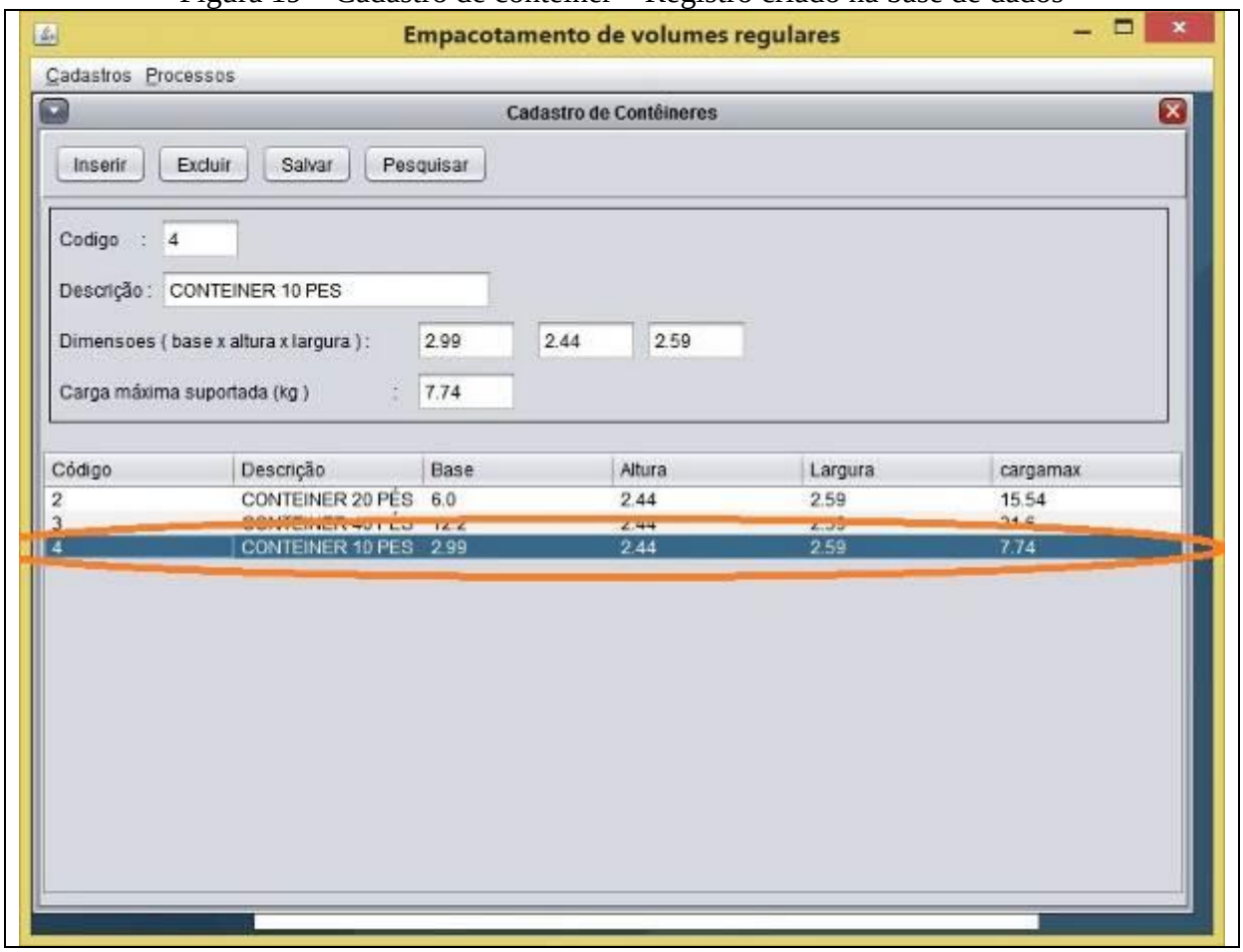
Figura 18 – Cadastro de contêiner – Salvar registro



Fonte: elaborado pelo autor.

A figura 19 mostra o registro salvo na base de dados.

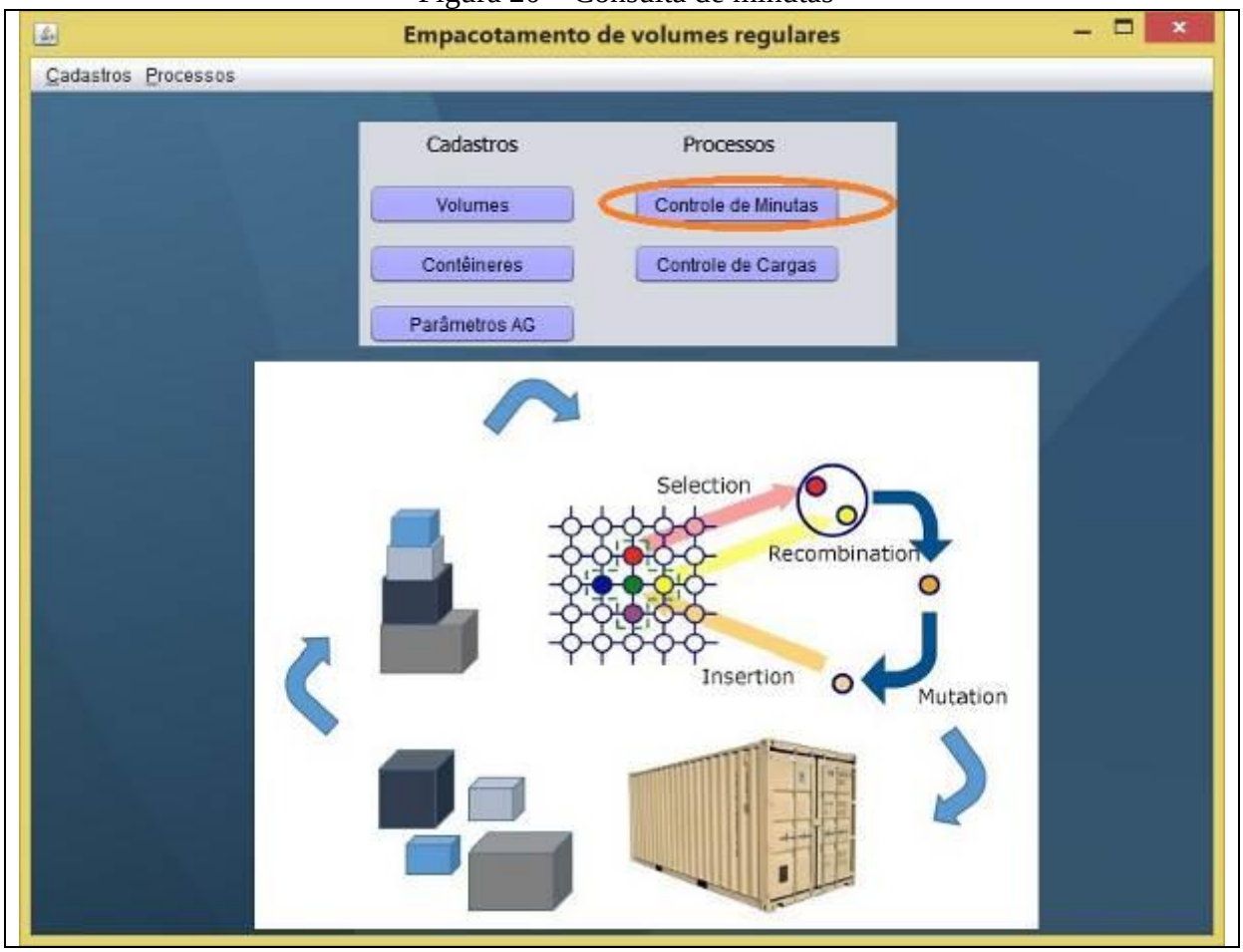
Figura 19 – Cadastro de contêiner – Registro criado na base de dados



Fonte: elaborado pelo autor.

A seguir é demonstrado o comportamento do sistema durante a criação de uma carga e a simulação de carregamento desta. Na figura 20 temos o momento em que o usuário seleciona o botão “controle de minutas” no menu inicial do sistema.

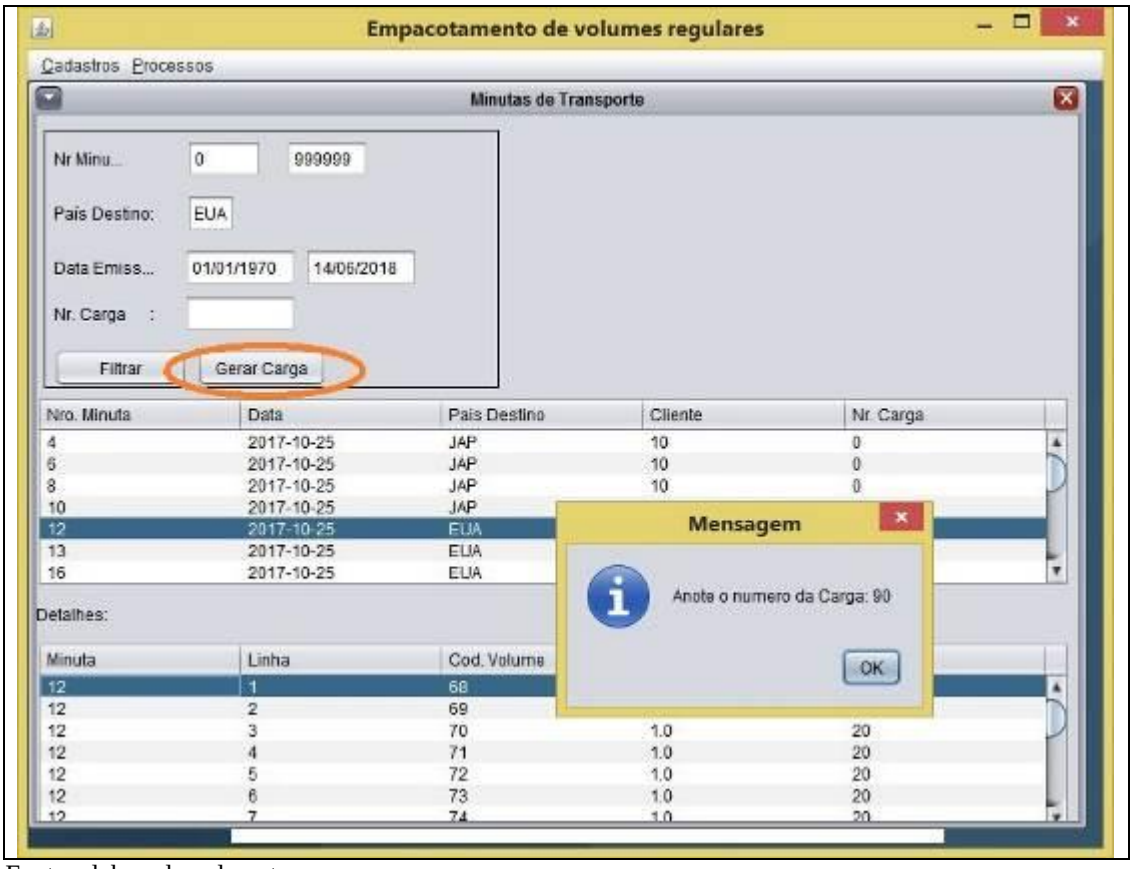
Figura 20 – Consulta de minutas



Fonte: elaborado pelo autor.

Na figura 21 temos a representação da criação de uma carga a partir da seleção de uma minuta de transporte e da seleção do botão “gerar carga”. Podemos ver a exibição de uma caixa de diálogo com o número da carga.

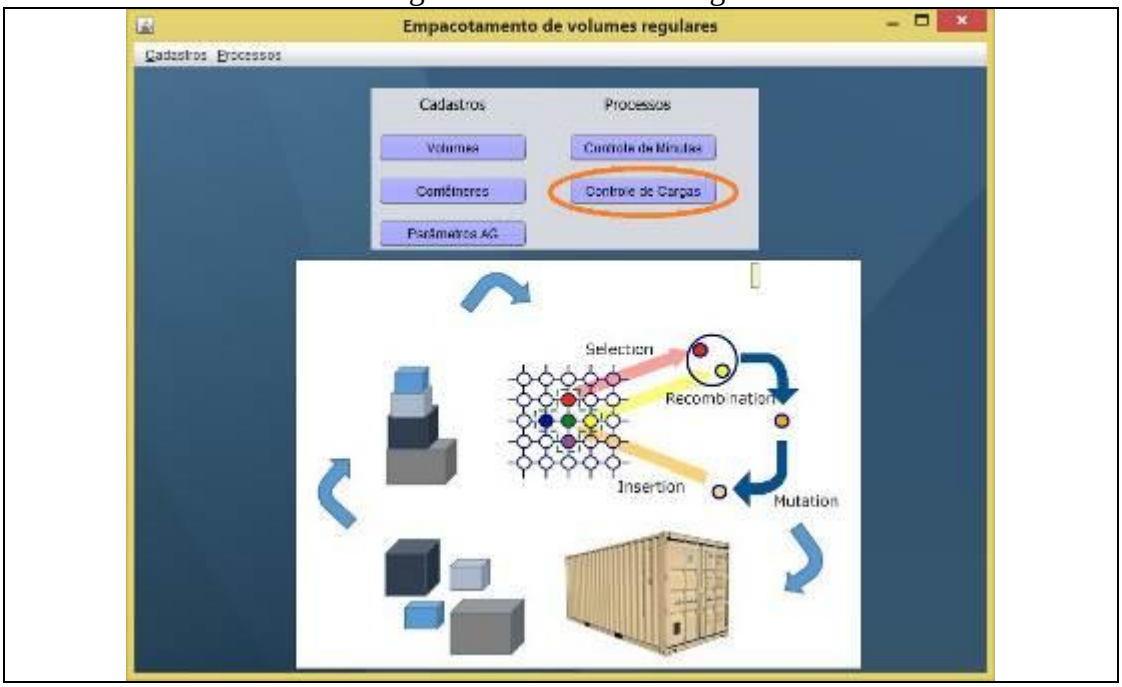
Figura 21 – Gerar carga



Fonte: elaborado pelo autor.

Para consultar os dados da carga criada, o usuário acessa via menu principal do sistema o botão “controle de cargas”. Essa ação está sendo representada na figura 22.

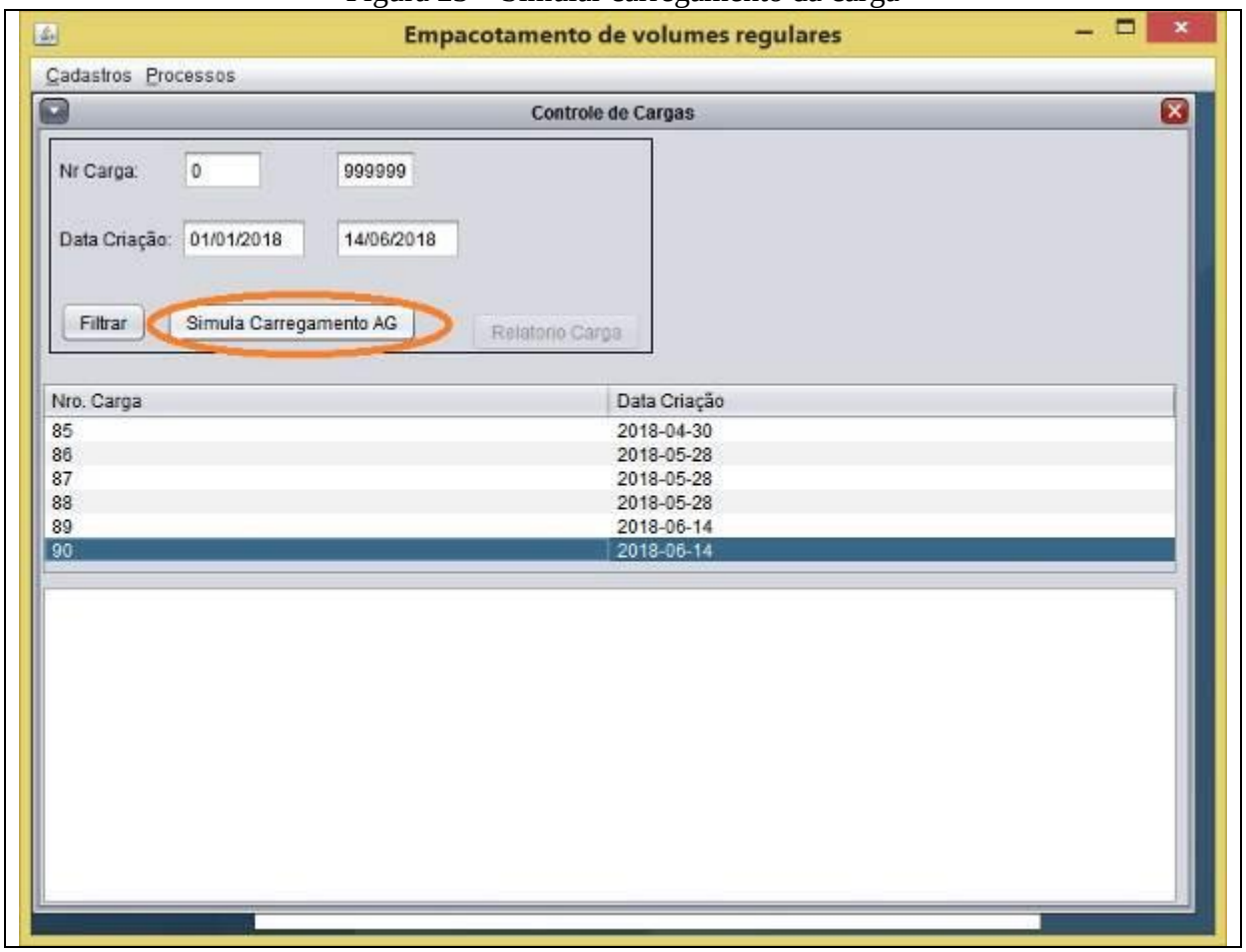
Figura 22 – Consultar carga



Fonte: elaborado pelo autor.

A figura 23 indica o momento em que o usuário seleciona a carga e clica no botão “Simula carregamento AG”.

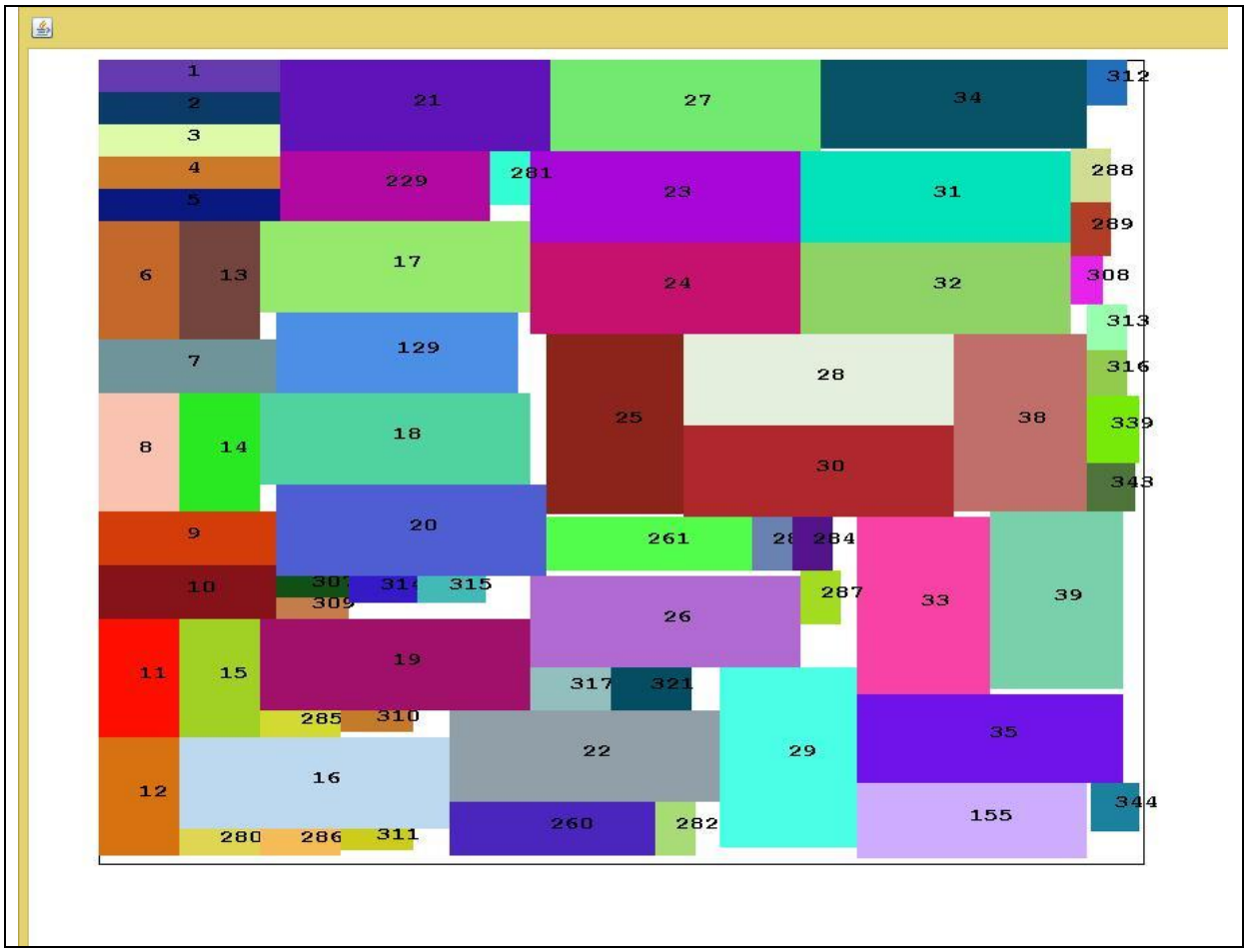
Figura 23 – Simular carregamento da carga



Fonte: elaborado pelo autor.

O resultado gráfico desse processamento está representado na figura 24, na qual podemos ver a área do contêiner e as pilhas dispostas em seu interior em uma representação 2D.

Figura 24 – Resultado disposição de pilhas no contêiner após simulação de carga



Fonte: elaborado pelo autor.

Além do relatório 2D com a posição e os códigos das pilhas a simulação de carregamento tem como saída um arquivo texto com os volumes que fazem parte de cada pilha. Na figura 25 temos a representação de uma das pilhas e os volumes contidos nela.

Figura 25 – Volumes da pilha

```
.....
.....
                          Pilhas do Conteinere
.....
.....
PILHA : 1
Rotacionar: SIM
Posicao na carga: 0 - 0
Volumes:
         33 - EMBALAGEM VOL0033 - Peso do Volume: 33.0 -  Peso Max Suportado: 330.0
         33 - EMBALAGEM VOL0033 - Peso do Volume: 33.0 -  Peso Max Suportado: 330.0
         33 - EMBALAGEM VOL0033 - Peso do Volume: 33.0 -  Peso Max Suportado: 330.0
```

Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

O protótipo apresenta boas soluções considerando as restrições de dimensão, peso e a carga máxima suportada pelo volume ou caixa. Outro detalhe importante é que a solução faz a seleção do contêiner adequado para suportar a carga, possibilitando dessa forma facilitar o planejamento do carregamento e a contratação do veículo/contêiner ideal para o transporte.

Na seção 3.4.1 são expostos métodos de testes utilizados. Na seção 3.4.2 são apresentados os resultados para cada categoria de teste. Na seção 3.4.3 é feito o comparativo entre o protótipo desenvolvido com os trabalhos correlatos.

3.4.1 Métodos e parâmetros aplicados nos testes

Para os testes foram utilizadas duas categorias de problema, categoria A e categoria B. Na categoria A, foram utilizados 980 volumes fortemente heterogêneos. Na categoria B, o teste utiliza 1679 volumes ou caixas. Os testes foram realizados em um computador com processador Intel Core™ i7-5500U CPU 2.40 GHz e 8 GB de memória RAM e sistema operacional Windows 8.1 Pro.

Os testes aplicados têm como base a lista de volumes a serem carregados, suas respectivas dimensões e restrições de empilhamento e a composição dos parâmetros do AG como número máximo de indivíduos, número máximo de populações, percentual de aptos e percentual de seleção natural.

Estão cadastrados no sistema 3 modelos de contêineres disponíveis conforme tabela 2.

Tabela 2 – Contêineres disponíveis para carregamento

Contêiner	Base	Largura	Altura	Área Total (Base*Largura)	Volume Total (Base*Largura*Altura)
Contêiner 1	2.99m	2.59m	2.44m	7,74m^2	18,89m^3
Contêiner 2	6.00m	2.59m	2.44m	15,54m^2	37,91m^3
Contêiner 3	4.00m	2.59m	2.44m	10,36m^2	25,27m^3

Fonte: elaborado pelo autor.

Na tabela 3 está a relação de volumes da categoria A. Nesta relação há a quantidade de volumes por carga, dados de dimensão (base, largura e altura) além da carga máxima suportada durante o empilhamento, ou seja, a caixa tem um limite de sobrecarga definida previamente, com o objetivo de evitar danos aos volumes. Além disso, na relação há o peso em kg de cada volume

Tabela 3 – Volumes categoria A

Volume	Quantidade	Base	Largura	Altura	Carga máx suportada	Peso
Caixa 1	30.00	0.05m	0.10m	0.10m	101.00 kg	10.10 kg
Caixa 2	60.00	0.20m	0.30m	0.10m	100.00 kg	10.00 kg
Caixa 3	50.00	0.20m	0.40m	0.11m	330.00 kg	33.00 kg
Caixa 4	50.00	0.12m	0.18m	0.12m	120.00 kg	12.00 kg
Caixa 5	50.00	0.13m	0.25m	0.13m	390.00 kg	39.00 kg
Caixa 6	150.00	0.26m	0.33m	0.14m	140.00 kg	14.00 kg
Caixa 7	50.00	0.19m	0.45m	0.80m	150.00 kg	15.00 kg
Caixa 8	50.00	0.16m	0.20m	0.16m	160.00 kg	16.00 kg
Caixa 9	50.00	0.17m	0.10m	0.17m	170.00 kg	17.00 kg

Caixa 10	50.00	0.18m	0.08m	0.18m	180.00 kg	18.00 kg
Caixa 11	50.00	0.15m	0.40m	0.80m	570.00 kg	57.00 kg
Caixa 12	40.00	0.10m	0.20m	0.20m	200.00 kg	20.00 kg
Caixa 13	50.00	0.25m	0.70m	0.22m	250.00 kg	25.00 kg
Caixa 14	150.00	0.10m	0.20m	0.45m	100.00 kg	10.00 kg
Caixa 15	50.00	0.45m	0.12m	0.10m	470.00 kg	47.00 kg
Caixa 16	50.00	0.20m	0.10m	0.44m	200.00 kg	20.00 kg

Fonte: elaborado pelo autor.

Considerando as dimensões `base`, `largura` e `altura` dos volumes categoria A, temos que calcular a `area total` das caixas, sendo que a `area total` é o somatório das `areas cúbicas` de todos os volumes da carga. A `area cúbica` é obtida pela multiplicação da `base`, `largura` e `altura` dos volumes. Como exemplo, vamos calcular a `area cúbica` da `Caixa 16`, conforme abaixo:

`area cúbica caixa 16` = `base` * `largura` * `altura`;

`area cúbica caixa 16` = 0,20 * 0,10 * 0,44

`area cúbica caixa 16` = 0,0088 metros cúbicos.

Aplicando o cálculo para todas as caixas e efetuando o somatório temos que o volume total da categoria A é de 13,45 metros cúbicos. Considerando os contêineres disponíveis para carregamento podemos supor que os volumes poderiam ser alocados no `contêiner 1`, cuja área cúbica é de 18,89 metros cúbicos. Lembrando que deve ser levado em consideração a capacidade de peso suportado por cada caixa durante o empilhamento, isso pode gerar a necessidade de alocação de mais um contêiner para acomodar as caixas.

A listagem de volumes da categoria B está apresentada na tabela 4.

Tabela 4 – Volumes categoria B

Volume	Quantidade	Base	Largura	Altura	Carga máxima suportada	Peso
Caixa 1	150.00	0.16m	0.31m	0.31m	310.00 kg	31.00 kg
Caixa 2	250.00	0.16m	0.32m	0.32m	320.00 kg	32.00 kg
Caixa 3	150.00	0.17m	0.33m	0.33m	330.00 kg	33.00 kg
Caixa 4	225.00	0.17m	0.34m	0.34m	340.00 kg	34.00 kg
Caixa 5	25.00	0.20m	0.39m	0.39m	390.00 kg	39.00 kg
Caixa 6	250.00	0.20m	0.40m	0.40m	400.00 kg	40.00 kg
Caixa 7	180.00	0.21m	0.41m	0.41m	410.00 kg	41.00 kg
Caixa 8	105.00	0.21m	0.42m	0.42m	420.00 kg	42.00 kg
Caixa 9	25.00	0.22m	0.43m	0.43m	430.00 kg	43.00 kg
Caixa 10	100.00	0.22m	0.44m	0.44m	440.00 kg	44.00 kg
Caixa 11	125.00	0.45m	0.40m	0.45m	1350.00 kg	135.00 kg

Caixa 12	150.00	0.10m	0.20m	0.45m	100.00 kg	10.00 kg
Caixa 13	25.00	0.45m	0.12m	0.10m	470.00 kg	47.00 kg

Fonte: elaborado pelo autor.

Considerando as dimensões base, largura e altura dos volumes `categoria B`, temos que calcular a `area total` das caixas para termos uma estimativa do resultado que o protótipo deve apresentar. Aplicando o somatório das `áreas cúbicas` para todas as caixas da `categoria B` temos que a `área total` da `categoria B` é de 49,49 metros cúbicos. Considerando os contêineres disponíveis para carregamento, da mesma forma que fizemos para a `categoria A`, podemos supor que os volumes para serem carregados necessitariam de dois contêineres , um `contêiner 2` com capacidade de 37,91 metros cúbicos e um `contêiner 1`, cujo volume total é de 18,89 metros cúbicos. Da mesma forma que a `categoria A`, temos que ressalvar que deve ser levado em consideração a capacidade de peso suportado por cada caixa durante o empilhamento, pois isso pode gerar a necessidade de mais espaço para realizar o carregamento.

Ainda detalhando o teste que iremos realizar, vamos criar uma tabela de categorias de parâmetros para o AG. Para cada categoria, determinamos critérios de parada ou de execução do AG. O `número máximo de indivíduos` determina quantos indivíduos devem ser criados para cada população do AG. O `número máximo de populações` determina quantas populações devem ser criadas para cada simulação de carga. O `percentual de seleção natural` indica quantos indivíduos ou soluções devem ser descartados à medida que o AG vai criando as populações. De maneira contrária, o `percentual de aptos` indica quantos indivíduos devem ser mantidos para as populações futuras. Esta tabela de parametrização é apresentada na tabela 5.

Tabela 5 – Categorias de parâmetros do AG

Categoria	Nro máx. Individuos	Nro máx. populações	% seleção natural	% de aptos
Tipo 1	<= 50	>50	<= 10	<= 5%
Tipo 2	<= 50	>50	> 10	> 5%
Tipo 3	<= 50	>50	<= 10	<= 5%
Tipo 4	>50	<=50	> 10	> 5%
Tipo 5	>50	<=50	<= 10	<= 5%
Tipo 6	>50	<=50	>10	> 5%

Fonte: elaborado pelo autor.

3.4.2 Resultados dos testes da categoria A e categoria B

Para ambas as categorias, realizamos um teste para analisar a capacidade do protótipo de realizar o carregamento e ajudar a identificar qual tipo de parametrização apresenta melhor performance de processamento.

Na `categoria A`, o `tempo de processamento` e o `percentual residual` está representado na tabela 6. Podemos notar que o contêiner escolhido está de acordo com a previsão que tínhamos baseada na área das caixas da carga.

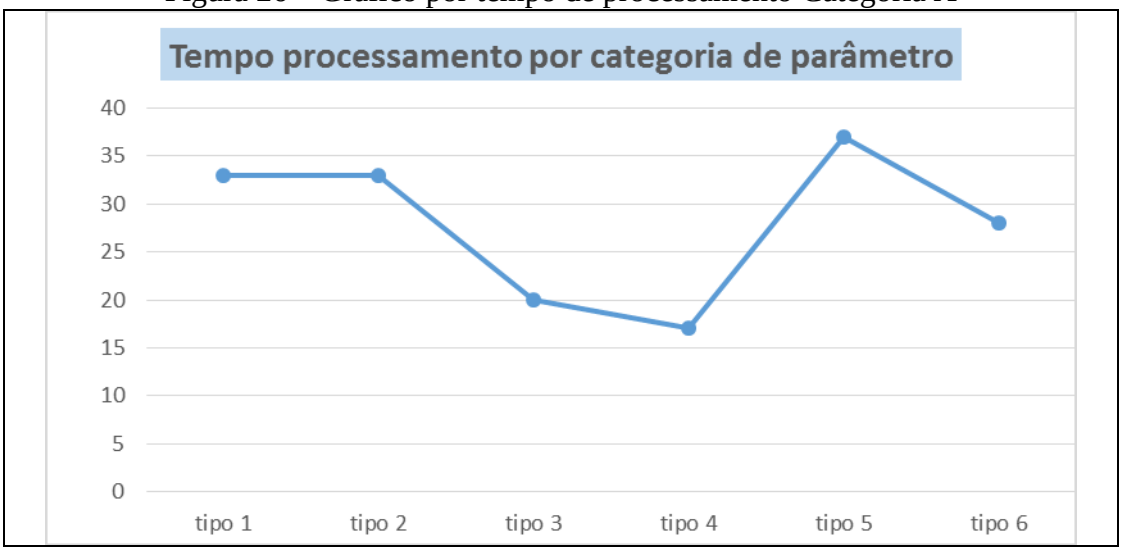
Tabela 6 – Resultado dos testes Categoria A

Categoria A			
Categoria de parâmetros	**Tempo Processamento (s)**	**% residual**	**Contêiner(es) utilizado(s)**
tipo 1	33	5,34%	Contêiner 1
tipo 2	33	4,66%	Contêiner 1
tipo 3	20	5,10%	Contêiner 1
tipo 4	17	4,68%	Contêiner 1
tipo 5	37	4,95%	Contêiner 1
tipo 6	28	5,24%	Contêiner 1

Fonte: elaborado pelo autor.

O gráfico da figura 26 representa o teste realizado para a `categoria A` baseado no `tempo de processamento`. Conforme podemos analisar, a categoria de parâmetros `tipo 4` é a que apresenta menor `tempo de processamento`.

Figura 26 – Gráfico por tempo de processamento Categoria A

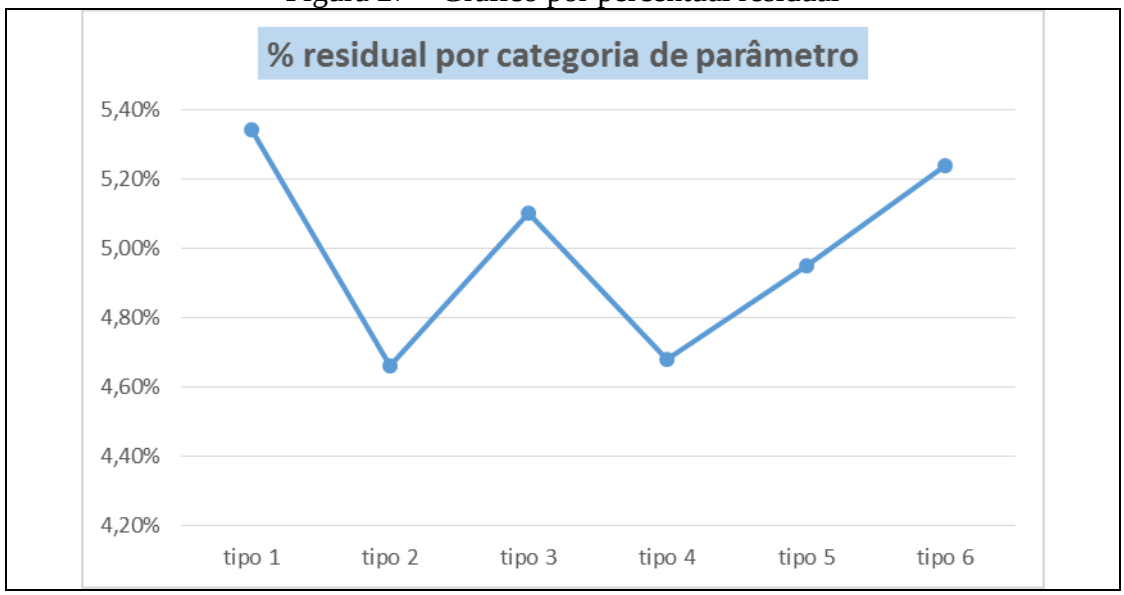


Fonte: elaborado pelo autor.

Ainda para a `categoria A`, trazemos um gráfico (figura 27) no qual temos a avaliação da solução por categoria de parâmetro, baseada no `percentual residual` da solução, lembrando que o `percentual residual` é todo espaço desperdiçado entre as pilhas, ou seja,

quanto mais `espaço residual` menos espaço disponível no contêiner e menos pilhas carregadas. Neste caso temos duas categorias de parâmetros que apresentam as melhores soluções, as categorias `tipo 2` e `tipo 4`.

Figura 27 – Gráfico por percentual residual



Fonte: elaborado pelo autor.

Para análise dos resultados obtidos nos testes da `categoria B`, o `tempo de processamento` e o `percentual residual` está representado na tabela 7. Além disso temos a informação dos contêineres necessários para acomodar as caixas, o que está de acordo com a previsão feita na seção 3.4.1

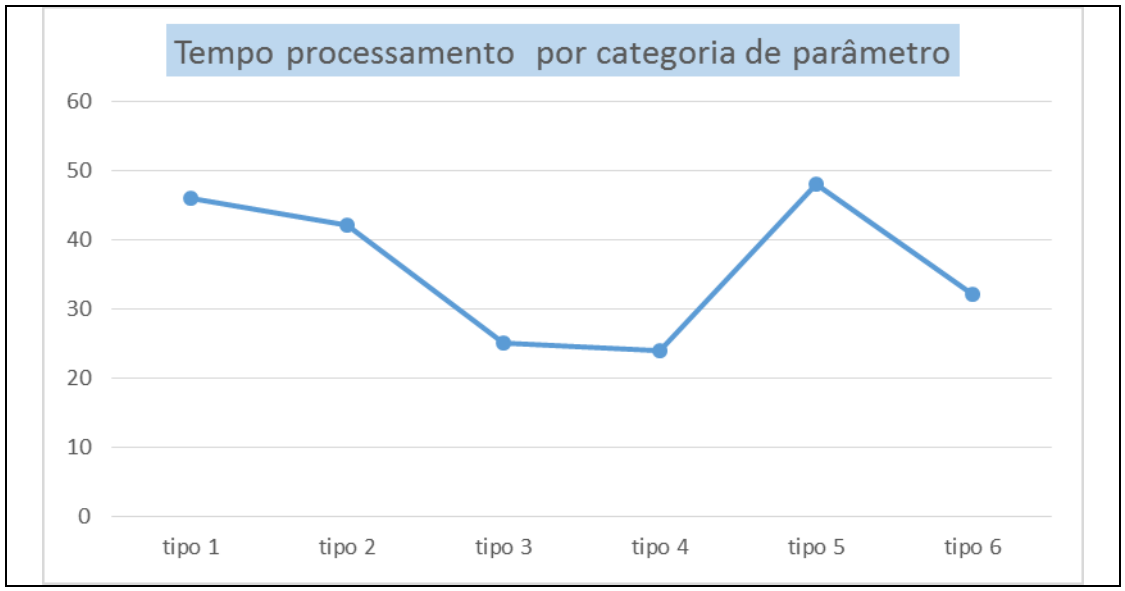
Tabela 7 – Resultado dos testes Categoria B

Categoria B			
Categoria de parâmetros	**Tempo processamento (s)**	**% residual**	**Contêiner(es) utilizado(s)**
tipo 1	46	2,56%	Contêiner 2 Contêiner 1
tipo 2	42	2,77%	Contêiner 2 Contêiner 1
tipo 3	25	2,46%	Contêiner 2 Contêiner 1
tipo 4	24	2,79%	Contêiner 2 Contêiner 1
tipo 5	48	2,92%	Contêiner 2 Contêiner 1
tipo 6	32	2,66%	Contêiner 2 Contêiner 1

Fonte: elaborado pelo autor.

O gráfico da figura 28 representa o teste realizado para a `categoria B` baseado no `tempo de processamento`. Conforme podemos analisar, a categoria de parâmetros `tipo 4` é a que apresenta menor `tempo de processamento`.

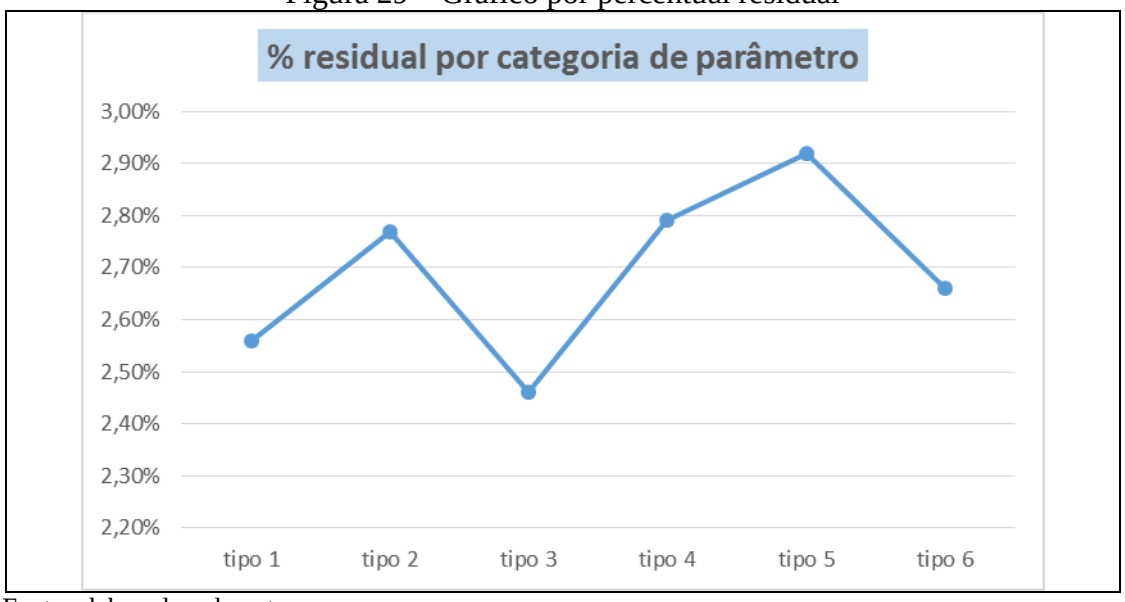
Figura 28 – Gráfico por tempo de processamento Categoria B



Fonte: elaborado pelo autor.

Na figura 29 está representado o gráfico da avaliação da solução baseada no `percentual residual` para cada categoria de parâmetro. Neste caso a categoria `tipo 3` obteve o menor `espaço residual` (espaço desperdiçado entre as pilhas alocadas no contêiner), ou seja, o espaço do contêiner foi melhor aproveitado. A categoria `tipo 4` que teve boa avaliação no `tempo de processamento` não teve uma boa avaliação do `percentual residual`, ficando na penúltima posição dentre as seis categorias avaliadas.

Figura 29 – Gráfico por percentual residual



Fonte: elaborado pelo autor.

Podemos concluir a partir dos experimentos realizados que o protótipo apresenta melhores soluções para os problemas `categoria A` quando aplicados parâmetros `tipo 4`. Ou seja, que tenha o `número máximo de indivíduos` das populações criados pelo AG maior que 50, o `número máximo de populações` seja menor que 50. Além disso o `percentual de`

`seleção natural` deve ser maior que 10% e o `percentual de aptos` maior que 5%. Para a `categoria B`, a categoria de parâmetros que melhor resolveu o problema foi a `tipo 3`, a qual tem como características um `número máximo de indivíduos` menor ou igual a 50, o `número máximo de populações` menor ou igual a 50. Além disso o `percentual de seleção natural` deve ser menor ou igual a 10% e o `percentual de aptos` menor ou igual a 5%.

A formulação do trabalho através da criação de pilhas de volumes se demonstrou bastante eficaz quando não se tem dentro da lista de volumes da carga, um número pequeno de volumes, com dimensões muito maiores que os demais, pois neste caso, a formação da pilha prejudica a otimização do espaço de carga.

3.4.3 Comparativo com correlatos

Nenhum dos trabalhos correlatos tem como característica a capacidade de respeitar todas as restrições apresentadas neste protótipo e todos trabalhavam com contêineres previamente definidos, sendo assim não podemos fazer um comparativo numérico com o presente trabalho, apenas relacionar a capacidade de solução entre o protótipo desenvolvido e os trabalhos correlatos, de acordo com o quadro 4.

Quadro 4 – Comparativo entre os trabalhos correlatos e o protótipo

Correlatos / Características	Vendramini (2007)	Silva e Soma (2002)	Liberalino et al. (2008)	Protótipo desenvolvido
Linguagem	FORTRAN 4.0	Ansi C	C++	Java
Equilíbrio implementado	Não	Não	Não	Sim
Método Utilizado	Chu-Beasley	Heurística de Volumes	HSSI + User Hints	Algoritmo genético
Interação com usuário	Não	Não	Sim	Não
Regras de Orientação do Volume	Não	Não	Não	Sim
Carga máxima do volume	Não	Não	Não	Sim
Visualização gráfica do carregamento	Não	Não	Não	Sim

Fonte: elaborado pelo autor.

A partir do Quadro 4 pode-se concluir que o único sistema que possui interação com o usuário é o de Liberalino et al. (2008). Nenhum dos trabalhos apresenta solução para o

equilíbrio dos volumes no momento do carregamento e nem utilizam regras de orientação dos volumes (se permite rotacionar em todos os sentidos) e carga máxima de empilhamento.

Todas as soluções apresentadas trabalham com contêineres previamente definidos, o que varia é a quantidade de volumes e seus respectivos tamanhos. Uma das funcionalidades implementada no protótipo proposto é que, partindo de uma lista de volumes com tamanhos e quantidades variadas, o sistema define o(s) melhor(res) contêiner(es) para que seja realizado o carregamento. Além disso, ele apresenta a disposição física dos volumes dentro do contêiner, visando otimizar a carga e respeitando regras como a carga máxima de empilhamento, rotação/orientação e equilíbrio dos volumes. Esse pacote de funções não está disponível em nenhuma das soluções apresentadas.

Desta forma, pode-se concluir que o presente trabalho trouxe significativas melhorias nos processos desenvolvidos até o momento. Considerando-se que disponibiliza funcionalidades que melhoram o planejamento de cargas e que visam reduzir custos de operação de logística, como mão de obra, transportes, locação de contêineres, espaço de armazenamento, além do aprofundamento do estudo referente aos algoritmos genéticos.

4 CONCLUSÕES

O protótipo desenvolvido cumpre seu papel no que diz respeito ao empacotamento de volumes regulares em contêineres para os parâmetros de testes realizados. A formulação da solução teve êxito em seu objetivo inicial utilizando uma modelagem mista de empacotamento 3D com o corte de chapas durante a simulação do carregamento.

Os tempos de processamento do AG para solucionar as categorias de testes apresentados são factíveis e a solução não demanda de muita capacidade computacional para ser alcançada.O protótipo atendeu ao pré-requisito de buscar uma solução, sem que o usuário necessite pré-selecionar um ou mais contêineres. O desenvolvimento do protótipo foi feito com a IDE NetBeans 8.2, linguagem Java e banco de dados MySql e todas essas ferramentas apresentaram-se eficazes durante a implementação.

Os trabalhos correlatos tiveram papel fundamental na elaboração deste protótipo e se demonstraram eficazes na solução do problema do empacotamento tridimensional. Como é preciso compará-los com o protótipo desenvolvido a fim de fazer uma crítica sobre a sua operacionalidade, pode se destacar que nenhum deles tem detalhamento sobre a capacidade de carga dos volumes e não apresentam uma representação gráfica e prática de como os volumes devem ser alocados no contêiner. Além disso trabalham com contêineres previamente selecionados, desta forma pode ocorrer que que solução contenha volumes que não podem ser carregados. O protótipo desenvolvido faz um cálculo dinâmico do melhor contêiner para o carregamento e indica quantos contêineres são necessários para que nenhum volume fique sem alocação.

4.1 EXTENSÕES

A partir do trabalho desenvolvido pode-se sugerir as seguintes extensões:

a) implementar um segundo AG para otimizar a formação das pilhas. Isso se faz necessário para diminuir os espaços ociosos na pilha, quando as dimensões dos volumes tem um grande variação;
b) criar uma visualização tridimensional das pilhas e do contêiner.

REFERÊNCIAS

ANDRADE, Mariana Silva Faleiro de. **Algoritmos evolutivos mono e multiobjectivos par problemas bidimensionais de corte.** 2009. 14 f. Dissertação (Mestrado em modelagem matemática e computacional) - Departamento de pesquisa e pós graduação, Centro federal de educação tecnológica de Minas Gerais. CEFET-MG, Minas Gerais.

CORREA, Maria H. et al. Problemas de empacotamento tridimensional. **Investigação Operacional**, Porto, v**.** 12, n. 2, p.169-180**,** dez.1992.Disponível em: <https://www.researchgate.net/publication/274636564_Problemas_de_empacotamento_tridimensional>. Acesso em 29 mar.2017.

GOMES, Carlos. F. S.; RIBEIRO, Priscilla. C. C. **Gestão da cadeia de suprimentos integrada à tecnologia da informação**. São Paulo/SP: Pioneira Thonsom Learning, 2004.

HOLLAND, John H. **Adaptation in natural artificial systems.** Michigan/EUA: University of Michigan Press, 1975.

LIBERALINO, Carlos H. P. et al. Uso de Dicas do Usuário na Otimização do Problema Bin Packing Tridimensional. In: SBPO - A PESQUISA OPERACIONAL E O USO RACIONAL DE RECURSOS HIDRICOS, XL, 2008, João Pessoa/PB. **Anais eletrônicos...** João Pessoa/PB: 2008.p.1942-1950. Disponível em: < http://www.din.uem.br/sbpo/sbpo2008/pdf/arq0196.pdf>. Acesso em 25 mar. 2017.

LINDEN, Ricardo. **Algoritmos genéticos:** uma importante ferramenta da inteligência computacional.v-2. Rio de Janeiro/RJ: Brasport, 2006a.

LINDEN, Ricardo. **Algoritmos genéticos:** uma importante ferramenta da inteligência computacional, v-3. Rio de Janeiro/RJ: Editora Ciência Moderna, 2012b.

LUDOVICO, Nelson. **Logística internacional:** um enfoque em comercio exterior. São Paulo/SP: Editora Saraiva, 2007.

MENCHIK, Carlos R. **Gestão estratégica de transportes e distribuição**. Curitiba /PR: IESD Brasil S.A, 2010.

MOSQUERA, Gabriela. P. **Contribuições para o Problema de Corte de Estoque Bidimensional na Industria Moveleira.** 2007. 115 f. Dissertação (Mestrado Profissional) - Departamento de Matemática , Universidade do Estado de São Paulo – UNESP, São Paulo.

MORABITO, Rodrigo.; ARENALES, Marcos N. Abordagens para o problema do carregamento de contêineres. **Pesquisa Operacional**, v. 17, n. 1, p. 2, 1997.

MORALES, Silvia R. **Otimização do carregamento de produtos paletizados em caminhões. Gestão & Produção**, Uberlandia/MG, v. 4, p.234-252, ago. 2007.
NOVAES, Antônio G. **Logística e gerenciamento da cadeia de distribuição:** estratégica, operação e avaliação. Rio de Janeiro/RJ: Elsevier, 2007.

OLIVEIRA, Simone. **Sistema de transportes no Brasil – O multimodalismo como opção logística.** 2004. 108 f. Monografia (MBA) –Depoartamento de Pós graduação e atividades complementares CEPAC, Universidade Gama Filho - UGF, Rio de Janeiro.

ROMAO, Oberlan C; SANTOS, André G. Apoio à decisão no carregamento de veículos com empacotamento tridimensional. In SIMPOSIO BRASILEIRO DE PESQUISA OPERACIONAL,XLIV, 2012 Rio de Janeiro/RJ. **Anais eletrônicos.** Rio de Janeiro/RJ. 2012 p.1844-1855.

SILVA, José. L. C.; SOMA, Ney Y. Um algoritmo genético aplicado ao problema de empacotamento de bins tridimensionais.In XXII Encontro Nacional de Engenharia de Produção, 2002, 8 p. Curitba/PR.**Anais eletrônicos**.Curitiba/PR. 2002.

SILVA, José. L. C.; SOMA, Ney Y. Um algoritmo polinomial para o problema de empacotamento de contêineres com estabilidade estática da carga.**Pesquisa Operacional**, v.23 n.1, p.8, 2003

VENDRAMINI, Eliane. **Otimização do problema de carregamento de container usando uma metaheurística eficiente**. 2007. 115 f. Dissertação (Mestrado em Engenharia Elétrica) – Departamento Engenharia Elétrica, Universidade Estadual Paulista - UNESP, Ilha Solteira.