UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UM *FRAMEWORK* INTERATIVO PARA VALIDAÇÃO DE RESULTADOS DE PESQUISAS DE OTIMIZAÇÃO BASEADOS EM HEURÍSTICAS APLICADAS A PROBLEMAS DE PROGRAMAÇÃO DE TAREFAS

RONALDO CASTRO DE OLIVEIRA

ORIENTADOR: PROF. DR. ROBERTO FERNANDES TAVARES NETO

São Carlos - SP
Dezembro/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UM *FRAMEWORK* INTERATIVO PARA VALIDAÇÃO DE RESULTADOS DE PESQUISAS DE OTIMIZAÇÃO BASEADOS EM HEURÍSTICAS APLICADAS A PROBLEMAS DE PROGRAMAÇÃO DE TAREFAS

RONALDO CASTRO DE OLIVEIRA

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de São Carlos para obtenção do título de doutor em Engenharia de Produção.

Orientador: Prof. Dr. Roberto F. Tavares Neto

São Carlos - SP
Dezembro/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Tese de Doutorado do candidato Ronaldo Castro de Oliveira, realizada em 16/12/2016:

Prof. Dr. Roberto Fernandes Tavares Neto
UFSCar

Prof. Dr. Marcelo Seido Nagano
USP

Prof. Dr. Paulo Rogerio Politano
UFSCar

Profa. Dra. Roseli Aparecida Francelin Romero
USP

Profa. Dra. Vitoria Maria Miranda Pureza
UFSCar

São Carlos - SP
Dezembro/2016

Dedicatória

Dedico este trabalho a minha esposa Miriam e meus dois filhos Alex e Bruna, que nasceram durante o doutorado, principalmente pela paciência, compreensão e apoio nos momentos difíceis.

AGRADECIMENTO

Primeiramente agradeço a minha esposa Miriam e meus filhos Alex e Bruna, por embarcar nesta empreitada que é o desenvolvimento de um doutorado e por superar todos os percalços durante esta trajetória.

Agradeço a meu orientador Roberto Fernandes Tavares Neto pela paciência, senso de direcionamento e pela capacidade de me transmitir confiança nos momentos de dúvidas.

Agradeço a professora Andrea Lago da Silva por acreditar em mim e pela indicação do professor Roberto Tavares para ser meu orientador.

Agradeço à Faculdade de Computação e à Universidade Federal de Uberlândia por me proporcionarem o afastamento integral de minhas atividades para a realização do doutorado.

Agradeço ao departamento de Engenharia de Produção da UFSCAR, em especial ao Programa de Pós-graduação e seus professores, pela seriedade na tratativa e no compromisso de transformar os egressos em profissionais qualificados.

Agradeço a todos os meus familiares e amigos pela compreensão por estar longe por um longo período e mesmo assim continuarem a fazer parte de minha vida.

Acima de tudo agradeço a esta força maior que, de alguma forma, nos faz continuar sempre em nosso caminho.

Epígrafe

"Faz-se-a sábio aquele que consegue acumular conhecimentos ao longo do tempo

e com sabedoria transforma estes conhecimentos em algo diferente"

Anônimo

RESUMO

Ao avaliar pesquisas que utilizam heurísticas e meta-heurísticas, percebe-se que existe uma carência de padronização na análise e na representação dos resultados de saída. Alguns trabalhos nesta área apresentam somente resultados tabulados com análises baseadas em valores médios e desvio padrão. Outros trabalhos apresentam além das tabelas, gráficos que permitem uma análise visual dos resultados, mas carecendo de uma análise estatística para a confirmação das hipóteses da pesquisa. Poucos trabalhos apresentam uma análise estatística capaz de fornecer informações mais conclusivas, tais como análises paramétricas e não paramétricas. Esta falta de padronização dificulta a comparação de pesquisas correlatas que tratam do mesmo assunto, além de forçar os pesquisadores o uso de abordagens *ad-hoc* para demonstrar vantagens e desvantagens das novas técnicas desenvolvidas. Este trabalho propõe a implementação de um processo de desenvolvimento de pesquisas que utilizam otimização baseadas em heurísticas e meta-heurísticas para solução de problemas de *scheduling* em Engenharia de Produção, com foco na validação estatística dos resultados de saída. A validação deste processo é realizada através do desenvolvimento implementado em um *framework* computacional que fornece apoio para a definição de *benchmarks* de comparação, parametrização das soluções algorítmicas analisadas, execução das soluções algorítmicas baseadas em heurísticas com armazenamento de resultados e subsídio completo para a análise estatística dos resultados de saída.

Palavras-chave: *framework*, otimização, heurística, *scheduling*, análise estatística de dados.

ABSTRACT

When evaluating researches using heuristics and metaheuristics is perceived that there is a lack of standardization in the analysis and representation of the output results. Some works in this area presents only tabulated results with analysis based on mean and standard deviation values. Other works shows also graphs that allow a visual analysis of the results, but lacking a statistical analysis to reach conclusions. In a few works, it can be perceived the concern about in conducting a statistical analysis that can be able to provide more conclusive information, such as parametric and non-parametric analysis. This lack of standardization makes it difficult to compare related researches with the same goals. In addition, researchers are compelled to use ad-hoc approaches to demonstrate advantages and disadvantages of new techniques developed. This study proposes an implementation of a researches development process using optimization based on heuristics and meta-heuristics for solving *scheduling* problems of production engineering, focusing on statistical validation of output results. The validation of this process is achieved by developing implementation of a computational *framework* that provides support to comparison *benchmarks*, parameterization of algorithmic solutions analyzed, execution of the algorithmic solutions based on heuristics with storage of results and full allowance for statistical analysis from the output results.

Key-words: *framework*, optimization, heuristic, *scheduling*, analysis, statistical data validation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACO - *Ant Colony Optimization* - otimização por colônia de formigas

AIS - *Artificial Immune Systems* - sistemas artificialmente imunes

ALG - Algoritmo

ANOVA - *Analysis of Variance* - análise de variância

BCO - *Bee Colony Optimization* - otimização por colônia de abelhas

BnB - *Branch and Bound* - ramificar e limitar

CMMI - *Capability Maturity Model Integration* - modelo de maturidade em capacitação - Integração

CRUD - *Create, Restore, Update and Delete* - cadastro, consulta, atualização e exclusão

DC - *Divide and Conquer* - dividir e conquistar

DEAP - *Distributed Evolutionary Algorithms in Python* - algoritmos evolucionários distribuídos desenvolvidos em Python

DER - Diagrama e Entidade-Relacionamento

DP - *Dynamic Programming* - programação dinâmica

EA - *Evolutionary Algorithms* - algoritmos evolucionários

EasyLocal++ - *Framework* orientado a objetos para projeto flexível de algoritmos de busca local

ECJ - *Java Evolutionary Computation Toolkit* - conjunto de ferramentas Java para computação evolucionária

EOLib - *Evolving Object Library* - biblioteca de computação evolutiva

ESOF - Engenharia de *Software*

EvA2 - *Evolutionary Algorithms framework version 2* - *framework* de algoritmos evolucionários segunda versão

FIT - *Fitness* - solução encontrada

FLSHCOP - *Framework Local Search Heuristic for Combinatorial Optmization Problems* - *framework* de heurísticas de busca local para problemas de otimização combinatória

FOM - *A Framework for Metaheuristic Optmization* - *framework* para otimização de meta-heurísticas

FSHMACF - *Framework* para Sistemas Heurísticos Multiagentes baseados em Algoritmos de Colônia de Formigas

FWER - *Family-Wise Error Rate* - taxa de erro de agrupamento

GA - *Genetic Algorithmic* - algoritmo genético

GAP - Diferença entre o valor real e o valor previsto

GeFEOA - *Grid-enabled Framework for Exact Optimization Algorithms* - *framework* de algoritmos de otimização exatos para ambientes distribuídos em grid

GNU-GPL - General Public License - licença pública geral

GRASP - *Greedy Randomized Adaptive Search Procedure* - procedimento de busca adaptativo randômico e guloso.

H0 - hipótese nula

H1 - hipótese alternativa

HEAL - *Heuristic and Evolutionary Algorithms Laboratory* - laboratório de algoritmos heurísticos e evolucionários

Heuristic Lab - *framework* para algoritmos heurísticos e evolucionários

HH - *Hyper Heuristic* - hiper-heurística

HOSDA - *Heuristic Optimization with Statistical Data Analysis* - *framework* para otimização heurística com análise de dados estatística

Hotframe - *Heuristic Optimization Framework* - *framework* de otimização heurística

HSD - *Honest Significant Difference* - diferença significante honesta

HyFlex - *A Flexible Framework for the Design and Analysis of Hyper-heuristics* - *framework* flexível para projeto e análise de hiper-heurísticas

IG - *Iterated Greedy* - algoritmo guloso iterativo

INST - Instância

IQR - *Inter Quartile Range* - amplitude interquartil

JCLEC - *Java Class Library for Evolutionary Computation* - biblioteca de classes Java para computação evolucionária

jMetal - Java Metaheuristic Algorithmics - algoritmos meta-heurístico em Java

LS - *Local Search* - busca local

LSD - *Least Significant Difference* - menor diferença significante

MA - *Memetic Algorithmics* - algoritmos meméticos

MALLBA - Biblioteca de soluções para otimização combinatória

MetaRaPS - *Metaheuristic for Randomized Priority Search* - meta-heurística para busca randômica priorizada

METSlib - *Metaheuristic Solution Library* - biblioteca de soluções meta-heurísticas

MO - *Multiobjective Metaheuristics* - meta-herísticas multiobjetivos

MOEAT - *Multiobjective Evolutionary Algorithms Tool* - ferramenta para algoritmos evolucionários multi-objetivos

MS - *Multiple Start* - múltiplos inícios

NP-Opt - *Framework* de otimização para problemas NP (Non-deterministic Polynomial time - Tempo polinomial não determinístico)

OAT - *Optimization Algorithm Toolkit* - conjunto de ferramentas para otimização de algoritmos

oMetah - *Open Metaheuristic* - *framework* para meta-heurísticas de código aberto

OO - *Object Oriented* - orientação a objetos

Opt4J - *framework* modular para otimização de meta-heurísticas

ORM - *Object-Relational Mapping* - mapeamento objeto-relacional

ParaDisEO - *Parallel and Distributed Evolutionary Optimization* - *framework* para otimização evolucionária paralelas e distribuídas

PSO - *Particle Swarm Optimization* - otimização por enxame de partículas.

RAD - *Rapid Aplication Development* - desenvolvimento rápido de aplicações

RBD - *Random Block Design* - experimentos com blocos randômicos

RDP - *Relative Deviation Percentage* - porcentagem de variação relativa

RSD - *Relative Standard Deviation* - desvio padrão relativo

SA - *Simulated Annealing* - recozimento simulado

SD - *Steepest Descent/Fill Climbing* - busca por descida mais íngreme

SS - *Scatter Search* - busca por dispersão

TS - *Tabu Search* - busca Tabu

TSF - *Tabu Search Framework* - *framework* de busca Tabu

UC - *Use Case* - caso de uso

UML - *Unified Modeling Language* - linguagem de modelagem unificada

VNS - *Variable Neighborhood Search* - busca de vizinhança variável

Wallace - *Framework* para computação evolucionária

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Considerações iniciais

Realizar pesquisas que tratam da solução de problemas de sistemas de produção, em muitos casos, é um grande desafio. O fluxo natural na busca das soluções para estes problemas costuma envolver a implementação de uma solução algorítmica e sua validação, seja através da comparação com resultados existentes na literatura, soluções fornecidas por especialistas ou através da aplicação de outras técnicas. Um grande desafio deste processo é que os problemas de sistemas de produção são de grande complexidade – inclusive muitas vezes classificados como NP-difíceis (*NP-hard* - Tempo polinomial não determinístico). Como por exemplo, têm-se vários problemas de *scheduling* (sequenciamento e programação da produção) (BRUCKER; KNUST, 2009). Uma característica relevante dos problemas NP-difícil é que a aplicação de métodos exatos pode demandar um conjunto proibitivo de recursos computacionais. Neste sentido, muitas pesquisas acabam utilizando heurísticas e meta-heurísticas para solucionar estes problemas. Conforme levantamento da literatura, várias meta-heurísticas afirmam conseguir chegar a resultados aceitáveis próximos (ou iguais) aos ótimos, com a vantagem de usar um tempo computacional muito menor (BLUM; ROLI, 2003).

Percebe-se que a utilização de meta-heurísticas para solucionar problemas de engenharia de produção é bastante vasta. Como exemplo, encontram-se trabalhos que utilizam da meta-heurística Busca Tabu na solução de problemas de roteirização de veículos (JIN et al., 2012), trabalhos que aplicam a meta-heurística de Algoritmo Genético (GA - *Genetic Algorithm*) na solução de problemas de localização de facilidades (PASANDIDEH; NIAKI, 2010) e na solução de problemas de dimensionamento de lote de produção (TOLEDO et al., 2013), trabalhos que usam

meta-heurística de otimização por colônia de formigas (ACO - Ant Colony Optimization) em problemas de *scheduling* em *flowshop* permutacional (TAVARES NETO; GODINHO FILHO, 2011), ou ainda trabalhos que utilizam as meta-heurísticas GRASP e *Simulated Annealing* (simulação por recozimento) na solução de problemas de *scheduling* em *open-job shop* flexíveis (WITKOWSKI et al., 2010).

Mesmo com o crescimento do uso de meta-heurísticas na solução dos problemas de sistemas de produção, nota-se que cada pesquisador desenvolve sua própria solução utilizando modelos distintos de análises estatísticas de dados, apresentando resultados em muitos casos somente em tabelas comparativas, nem mesmo se preocupando em utilizar gráficos que poderiam fornecer uma visualização mais clara dos resultados. O que se observa nas publicações da área é que, mesmo os trabalhos que utilizam análises gráficas, as mesmas não seguem um padrão que possa facilitar a comparação de pesquisas correlatas. Outro ponto relevante é que na maioria dos trabalhos não existe a preocupação com uma análise estatística através da utilização de métodos paramétricos e não paramétricos que poderiam fornecer informações mais detalhadas sobre os resultados possibilitando assim conclusões mais assertivas e direcionamento de futuras pesquisas.

Neste sentido, o presente trabalho se baseia no fato que o desenvolvimento de pesquisas na área pode ser facilitado se um pesquisador tiver disponível uma ferramenta que forneça um processo detalhado de desenvolvimento de pesquisas de otimizadores baseados em heurísticas e meta-heurísticas para problemas de programação da produção. Tal ferramenta pode ser de extrema importância se for capaz de fornecer um conjunto de análises estatísticas dos resultados encontrados. Este *framework* pode incluir algumas funções importantes, dentre elas:

- Definição de *benchmarks* (grupo de instâncias de referência com valores ótimos ou melhores valores encontrados) que são utilizados na comparação e na parametrização das soluções heurísticas proposta na pesquisa;
- Um processo de parametrização das soluções algorítmicas analisadas para a definição dos parâmetros de execução;
- Execução automática das soluções algorítmicas baseadas em heurísticas com armazenamento de resultados encontrados;
- Análise estatística completa dos resultados de saída, incluindo métodos estatísticos paramétricos e não paramétricos para garantir análises detalhadas e conclusões mais assertivas.

Espera-se que uma ferramenta com tais características consiga reduzir significativamente o tempo de desenvolvimento de pesquisas focadas em algoritmos de otimização. Além disso, como o *framework* aqui proposto implementa um processo de desenvolvimento, sua utilização tem como consequência uma padronização na análise e representação dos resultados de forma a facilitar a organização das conclusões e a comparação de trabalhos semelhantes.

1.2 Problema de pesquisa

Levando-se em conta uma variedade de pesquisas que utilizam heurísticas e meta-heurísticas e da crescente utilização destas técnicas na solução dos problemas de Engenharia de Produção, principalmente na programação de tarefas da produção (*scheduling*), percebe-se que não existe uma padronização relativa ao processo de análise e representação dos resultados de saída. Como resultado, observa-se que cada pesquisador desenvolve muitas vezes uma aplicação específica para sua pesquisa, naturalmente utilizando técnicas e ferramentas com que está mais familiarizado. Além disto, percebe-se uma falta de padronização dos dados resultantes das execuções dos algoritmos, principalmente das análises estatísticas destes dados de saída e suas representações gráficas. Isto dificulta a comparação, análise e validação dos resultados apresentados.

Desta forma, tem-se como problema de pesquisa a seguinte questão:

> "**Como construir um *framework* para análise e validação estatística dos resultados de pesquisas que utilizam heurística na solução de problemas de *scheduling*?"**

1.3 Motivação

Quando se realizam pesquisas de trabalhos na área, observa-se que vários trazem no título ou mesmo no resumo o termo *framework*. Vale ressaltar que o conceito da palavra *framework* é bastante amplo. No estudo de heurísticas e meta-heurísticas para problemas de otimização combinatória, a palavra *framework* pode definir um conjunto de padrões e premissas que direcionam a solução de um determinado problema (SUNDARAMOORTHY; MARAVELIAS, 2011). Já um *framework*, na ótica da engenharia de *software*, é definido como sendo uma aplicação

que reúne inúmeros procedimentos comuns entre vários projetos de *software* provendo uma funcionalidade genérica. Um *framework* pode ser direcionado para executar uma funcionalidade específica, através de configurações, durante a programação de uma aplicação, sendo quem dita o fluxo de controle da aplicação que será desenvolvida (JOHNSON, 1997).

Como exemplo, Keijzer et al. (2002) propõem um *framework* orientado a objetos para computação evolucionária que prove um conjunto flexível de classes para a construção de aplicações de computação evolucionária que podem utilizar de heurísticas e meta-heurísticas. Andreatta et al. (2003) apresentam um *framework* para heurísticas de busca local para problemas de otimização combinatória. O foco está relacionado em definir um conjunto de padrões no processo de desenvolvimento que facilite a comparação de forma sistêmica das diferentes estratégias algorítmicas e parâmetros para um mesmo problema. Sundaramoorthy e Maravelias (2011) propõem um *framework* que trata do problema de *scheduling* em rede ou sequenciais, assim como os subsistemas contínuos, com o foco na formulação de modelos de programação inteira mista (MIP - *mixed-integer programming*) para os referidos problemas e a busca de soluções para as formulações MIP resultantes de forma eficaz. Tavares e Godiinho Filho (2009) tratam de um *framework* computacional que permite a prototipagem de um grande conjunto de variações de heurísticas baseadas em sistemas de formigas, de forma a definir uma metodologia de desenvolvimento de algoritmos com o objetivo de redução significativa no tempo de implementação e facilitar a comparação entre técnicas propostas. Os exemplos citados, apesar de apoiarem o desenvolvimento de pesquisas na área, não tratam especificamente da análise estatística e representação dos resultados de saída das pesquisas.

Alguns poucos *frameworks* se preocupam com o tratamento estatístico dos resultados encontrados disponibilizando recursos ou componentes para tal. O *framework* para otimização de meta-heurísticas (FOM - *Framework for Optimization Meta-heuristic*), proposto por Parejo et al. (2003), o *framework* para o desenvolvimento de aplicações computacionais evolucionárias (JCLEC - *Java Class Library for Evolutionary Computation*), proposto por Ventura et al. (2008) e o conjunto de ferramentas para algoritmos de otimização (OAT *Optimization Algorithm Toolkit*) proposto por Brownlee (2007), até possuem componentes de análise estatística dos resultados usando métodos paramétricos e não paramétricos, mas não é o foco principal do trabalho. Desta forma pode-se observar uma lacuna nos trabalhos

existentes a respeito de recursos de análise estatística e representação dos resultados, definindo assim a motivação para a construção do presente trabalho.

1.4 Justificativa

Como justificativa para desenvolvimento de um *framework* com o foco na análise estatística e representação dos dados de saída de pesquisas pode-se citar a diminuição significativa do tempo de desenvolvimento da pesquisa e a comparação destes resultados com outras pesquisas correlatas, ou seja, permitir que o resultado publicado por um pesquisador possa ser comparado com outros resultados. Outra justificativa se dá em relação à qualidade dos dados apresentados nas conclusões da pesquisa. Quando se analisa alguns trabalhos na área, verifica-se que o pesquisador apresenta os resultados, mas não se preocupa em realizar uma validação destes resultados e, às vezes, nem mesmo em apresentar os resultados de forma gráfica que facilite um melhor entendimento.

Por exemplo, Hecker et al. (2013) tratam do problema de *scheduling* em uma linha de produção de uma padaria. Faz comparações entre duas meta-heurísticas: ACO (*Ant Colony Optimization*) e PSO (*Particle Swarm Optimization*), com o foco em minimização do *makespan* e na soma das datas de término. Neste trabalho os autores não fazem uma análise de dados estatística envolvendo a distribuição dos resultados encontrados, apenas apresentando resultados baseados nos valores mínimos encontrados. Este mesmo padrão pode ser visto nos trabalhos de Yagmahan e Yenisey (2010) e Rabanimotlagh (2011). Além da não adoção de análises estatísticas, Rabanimotlagh (2011) não apresenta gráficos para visualização dos resultados.

Outros exemplos podem ser vistos em Ahmadizar et al. (2010), Cheng et al. (2010), Huang (2010), Udhayakumar e Kumanan (2010), Liao et al. (2011), Ahmadizar (2012), Keskinturk et al. (2012), Cheng et al. (2013), Li et al. (2013a), Rossi e Lanzetta (2013b), dentre outros que fazem apresentação de valores mínimos, máximos e médios sem apresentar uma técnica estatística que poderia garantir a validação dos resultados. Ainda em relação aos trabalhos citados, os mesmos não apresentam os resultados em gráficos, o que também dificulta muito a sua análise e entendimento.

Pode-se observar que os trabalhos que embasaram esta tese não tratam especificamente da definição de padrões de especificação do processo de análise estatística e representação de resultados de saída e muito menos apresentam um

processo ou um *framework* que possa apoiar esta atividade, justificando assim o desenvolvimento desta pesquisa.

1.5 Objetivos gerais e específicos

O objetivo principal deste trabalho é propor a especificação e o desenvolvimento de um *framework* que tenha como foco a análise estatística, representação e validação de resultados de pesquisas que usam sistemas heurísticos e meta-heurísticos, aplicados a problemas de engenharia de produção, mais especificamente a problemas de *scheduling*.

Diante desta proposta, os objetivos específicos são:

I. Fazer um levantamento bibliográfico dos últimos anos de trabalhos que utilizam heurísticas na solução de problemas de engenharia de produção com foco em *scheduling*, com o intuito de identificar padrões baseados nos procedimentos que estão sendo utilizados pelos pesquisadores na análise e representação dos resultados de suas pesquisas.
II. Especificar um processo de desenvolvimento de pesquisas de otimizadores com base em heurísticas, dando ênfase na etapa de análise estatística e representação dos resultados encontrados, fornecendo um conjunto de ferramentas estatísticas para o pesquisador facilitando assim a validação destes resultados.
III. Especificar a arquitetura do *framework*, juntamente com a estrutura de classes e uma estrutura de armazenamento de dados que permita armazenar os dados de entrada, dados de parâmetros de execução e dados de saída de pesquisas na área;
IV. Construir o *framework* proposto;
V. Testar o *framework* com diferentes conjuntos de dados de forma a comprovar sua usabilidade.

1.6 Classificação do método de pesquisa

Este trabalho está baseado em um conjunto de modelos, variáveis de entrada e saída e variáveis de configuração que podem ser aplicados e tratados em um *framework*, com o intuito de serem utilizados na validação dos resultados de pesquisas

de otimizadores de heurística e meta-heurísticas aplicadas a problemas de engenharia de produção, mais especificamente problemas de *scheduling*. Estas variáveis podem ser: tempo de execução, quantidade de esforço, valores de saída, dentre outras, que podem ser comparadas e verificadas em relação ao ambiente e à base de dados já existentes. Pode-se perceber que o presente trabalho trata de modelo abstrato do mundo real.

Para Martins (2012), uma pesquisa pode ser definida como sendo quantitativa ou qualitativa. As pesquisas quantitativas são mais adequadas para apurar opiniões e atitudes explícitas e conscientes de um conjunto de entrevistados, utilizando para isto instrumentos padronizados, tais como questionários. São utilizados quando se sabe exatamente o que deve ser perguntado para atingir os objetivos da pesquisa. As pesquisas qualitativas possuem um caráter mais exploratório, com o intuito de estimular a se pensar sobre o tema de forma livre, fazendo emergir aspectos subjetivos, atingindo motivações não explícitas, ou mesmo não conscientes, de forma espontânea.

Segundo Morabito e Pureza (2012), modelos quantitativos podem ser definidos como modelos abstratos descritos em linguagem matemática e computacional, que utilizam técnicas analíticas (matemáticas e estatísticas) e experimentais (simulação), para calcular valores numéricos do sistema em questão, podendo ser utilizado na análise de diferentes resultados em diferentes situações. Martins (2012) apresenta métodos mais apropriados para pesquisa quantitativa em engenharia de produção, que são: Pesquisa de Avaliação (*survey*), Modelagem/Simulação, Experimento e Quase-experimento. O presente trabalho está situado na Modelagem/Simulação porque tem como foco a manipulação de variáveis de um modelo abstrato.

Segundo Bertrand e Fransoo (2002) um método de Modelagem/Simulação possui duas abordagens principais: a abordagem empírica e a abordagem axiomática. A abordagem empírica foca na elaboração de um modelo baseado na realidade, garantindo que este modelo seja representativo. Já a abordagem axiomática tem como base a produção de conhecimento sobre o comportamento de variáveis do modelo estudado. Ainda segundo Bertrand e Fransoo (2002) as pesquisas axiomáticas quantitativas podem ser classificadas como normativas ou descritivas. A abordagem normativa tem como objetivo o desenvolvimento de normas, políticas, estratégias e ações a fim de melhorar os resultados disponíveis. Já a abordagem

descritiva preocupa-se em analisar os modelos quantitativos, com o foco no entendimento do processo modelado ou na explicação de suas características.

Como o foco do presente trabalho é fornecer um ambiente padrão para validação dos resultados de pesquisas que fazem uso de heurísticas e meta-heurísticas na solução de problemas de engenharia de produção, pode-se classifica-lo como sendo de abordagem de pesquisa axiomática descritiva quantitativa. A Figura 1.1 apresenta uma visão da classificação da metodologia de pesquisa utilizada.

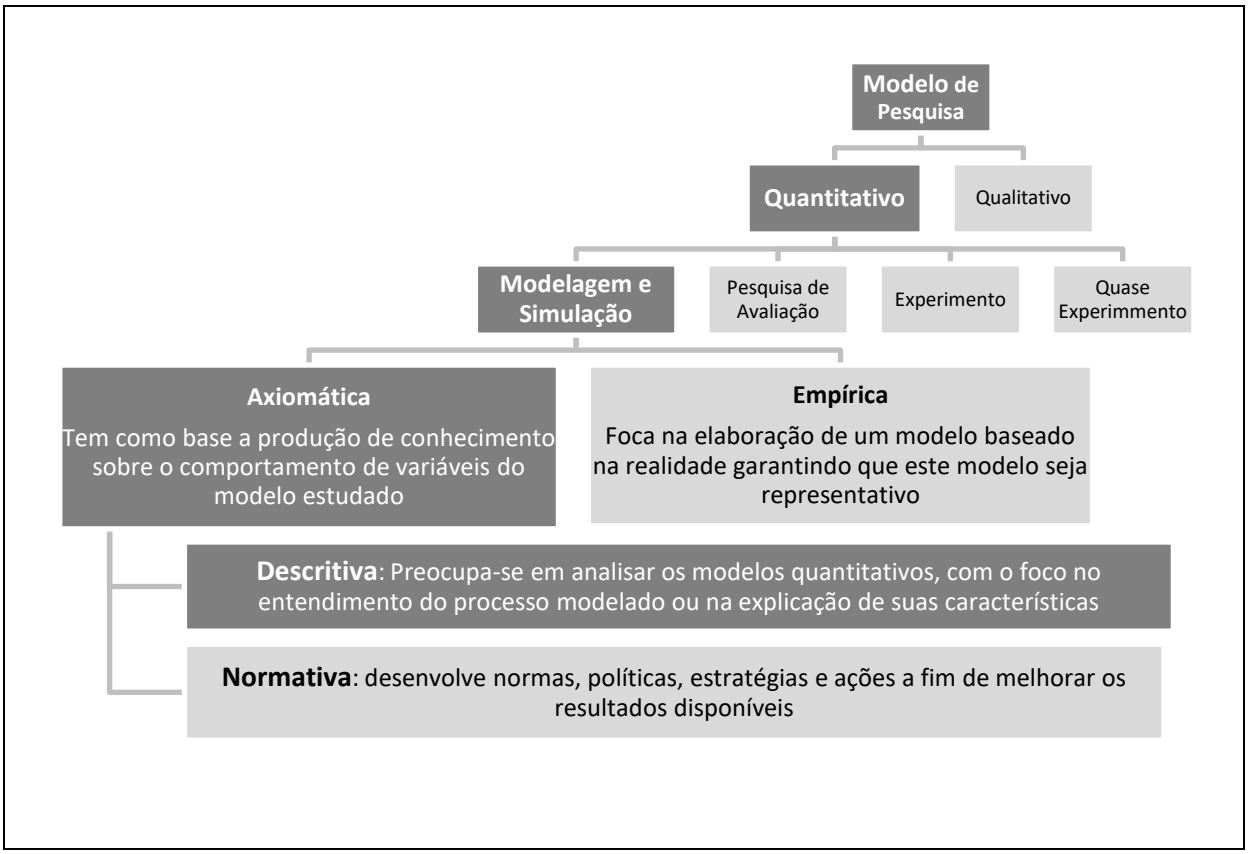


Figura 1.1 - Classificação da metodologia de pesquisa utilizada
Fonte: elaborado pelo autor

1.7 Passos para implementação da pesquisa

Para o desenvolvimento do presente trabalho definiu-se um conjunto de oito passos conforme pode ser visto na Figura 1.2, sendo detalhados como segue:

- **Passo 1 – Definir objetivos e classificação da metodologia de pesquisa utilizada.** Ao fim deste passo tem-se o presente capítulo, que apresenta o problema de pesquisa, os objetivos a serem alcançados, a classificação da

metodologia de pesquisa utilizada e uma visão macro do *framework* desenvolvido.

- **Passo 2 – Realizar revisão bibliográfica de trabalhos relacionados a *framework*.** Nesta etapa foi realizado um levantamento detalhado, destacando os trabalhos na literatura relacionados a *framework* juntamente com as motivações para que levaram a realização deste trabalho.
- **Passo 3 – Realizar revisão bibliográfica de problemas de engenharia de produção com foco em *scheduling*.** Nesta etapa foi detalhado os principais problemas de engenharia de produção com foco em *scheduling*, realizando um levantamento dos principais trabalhos publicados nos últimos anos relacionados ao tema. Ao final dos passos 2 e 3 tem-se o capítulo 2 desta tese.
- **Passo 4 – Especificar os métodos de análise estatística dos resultados de saída.** Nesta etapa foi especificado os métodos de análise de resultados de resultados de pesquisas relacionadas a problemas de engenharia de produção e é apresentado no capítulo 3.
- **Passo 5 – Identificar padrões de análise e representação dos resultados de pesquisas na área**. Foi feito um levantamento bibliográfico de trabalhos que usam heurísticas na solução do problema de scheduling para identificar padrões de análise e representação dos resultados de pesquisas na área. O capítulo 4 desta tese apresenta este levantamento.
- **Passo 6 – Especificar o processo de desenvolvimento de pesquisas de otimizadores baseados em heurísticas.** Neste passo foi detalhado todo o processo de desenvolvimento de pesquisas com heurísticas incluindo a etapa de análise estatística dos resultados. O capítulo 5 desta tese apresenta este processo.

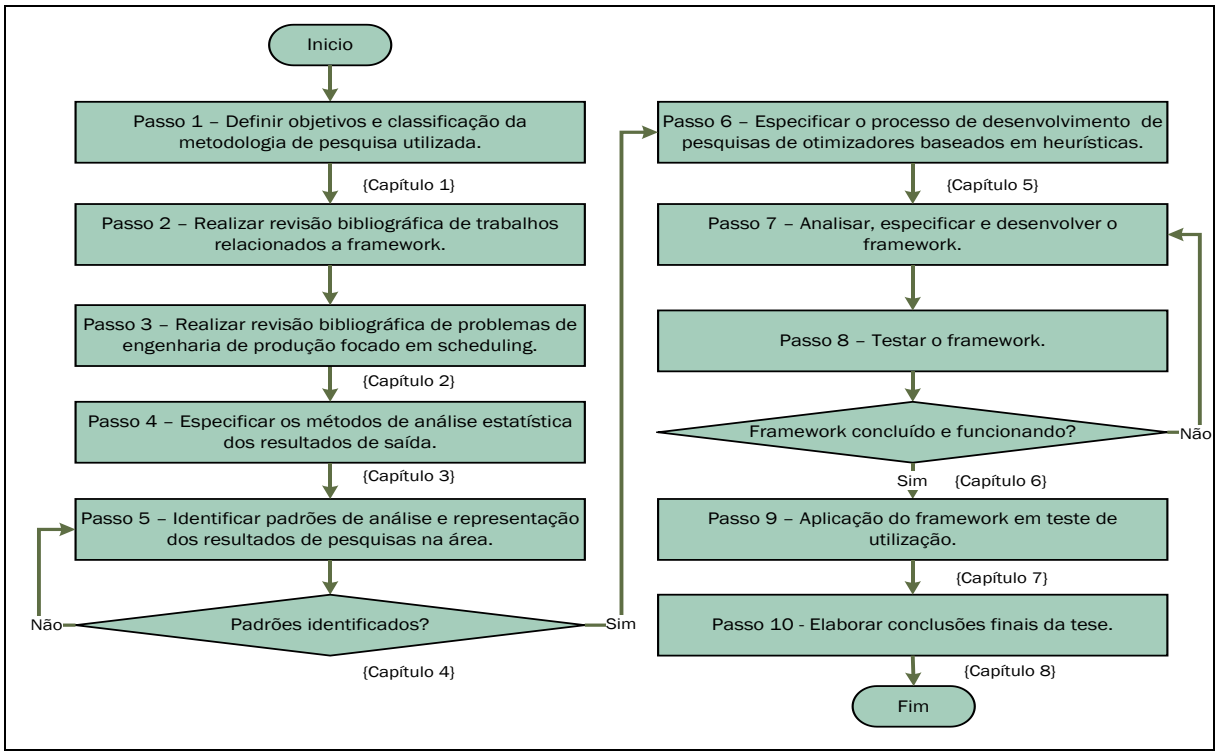


Figura 1.2 - Passos para desenvolvimento da tese
Fonte: elaborado pelo autor

- **Passo 7 – Analisar, especificar e desenvolver o *framework*.** Este passo detalha o processo de desenvolvimento do *framework,* especificando todas as etapas que foram implementadas para que o desenvolvimento do *framework*, incluindo processos de especificação de requisitos, análise, projeto e implementação e testes.
- **Passo 8 – Testar o *framework*.** Neste passo o *framework* desenvolvido foi testado de forma a garantir as funcionalidades definidas e usabilidade de todos os requisitos especificados. O capítulo 6 desta tese inclui os passos sete e oito descritos acima.
- **Passo 9 – Aplicação do *framework* em teste de utilização.** Neste passo foram implementados estudos de caso utilizando o *framework* desenvolvido de forma a validar o processo de análise estatística dos resultados, a usabilidade do sistema e a aplicação do mesmo em uma pesquisa com dados reais, sendo que os resultados são apresentados no capítulo 7 desta tese.
- **Passo 10 - Elaborar conclusões finais da tese.** Neste passo foram descritas as conclusões finais do referido trabalho inclusive com proposições para trabalhos futuros, que estão inclusos no capítulo 8 no final desta tese.

1.8 Desenvolvimento da pesquisa - estrutura do *framework*

Como resultado para este trabalho obteve-se um *framework* com uma arquitetura de implementação bem definida, que agiliza o processo de desenvolvimento de pesquisas da área e permite a validação estatística dos resultados de pesquisa que compara vários algoritmos heurísticos e meta-heurísticos aplicado a problemas de engenharia de produção. O foco da pesquisa foi para aplicação a problemas de *scheduling,* mas da forma como o *framework* foi implementado pode ser aplicado a uma vasta gama de problemas de otimização com base em heurísticas. A Figura 1.3 apresenta uma visão mais detalhada do *framework* desenvolvido.

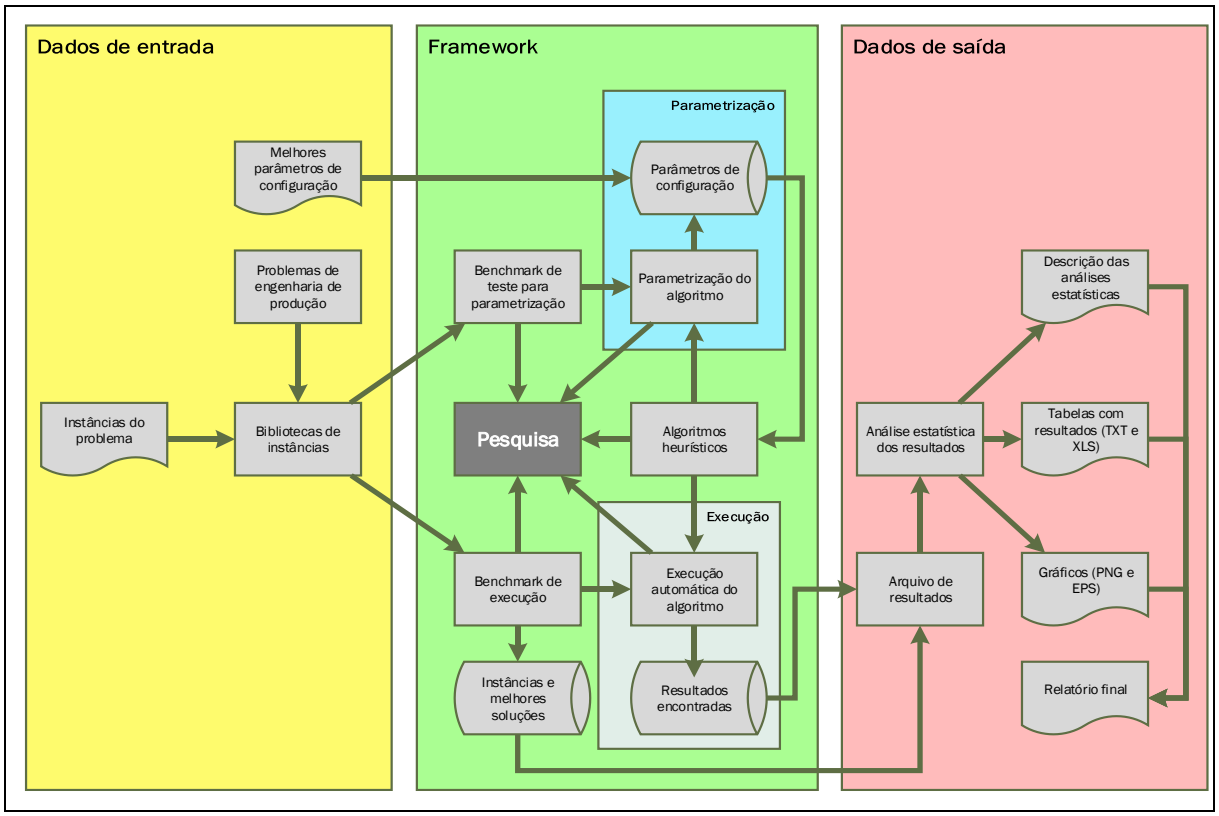


Figura 1.3 - Detalhamento da arquitetura do *framework* desenvolvido
Fonte: elaborado pelo autor

I. Dados de entrada - composto pela especificação do problema de engenharia de produção, pela especificação da biblioteca de instâncias, seja uma biblioteca já existente ou uma criada pelo próprio pesquisador, as instâncias do problema em questão e os melhores parâmetros de configuração dos algoritmos caso estes valores sejam conhecidos previamente.

II. *Framework* - concentra todas as informações sobre a pesquisa incluindo os algoritmos heurísticos comparados, o conjunto de instâncias de execução (*benchmark* de execução) com os valores ótimos e melhores valores conhecidos e o subconjunto de instâncias de teste para parametrização (*benchmark* de teste).

III. Parametrização - responsável pela execução automática dos algoritmos, com grupos de parâmetros de configuração previamente definido pelo pesquisador, para cada instância do subconjunto de *benchmark* de teste. O objetivo é fornecer um apoio ao ajuste fino de um processo externo de parametrização do algoritmo. Após a execução cabe ao pesquisador escolher os melhores parâmetros de cada algoritmo.

IV. Execução dos algoritmos - responsável pela execução automática dos algoritmos, com os parâmetros de configuração previamente definidos, para cada instância do conjunto de *benchmark* de execução. Os valores da solução encontrada e o tempo total de execução são armazenados no *framework* para serem utilizados na análise dos resultados.

V. Dados de Saída - composto pelos arquivos de resultados de saída dos valores processados entre os valores encontrados e os valores definidos como base de cálculo (soluções ótimas, melhores soluções ou outros), para cada instância do *benchmark* de execução. Tem-se ainda a análise estatística destes resultados baseados nos padrões de representação de dados, juntamente com as descrições das análises, as tabelas com os resultados e os gráficos gerados. Todas estas informações irão compor o relatório final da pesquisa que é gerado automaticamente pelo *framework*.

A Figura 1.4 mostra uma especificação dos dados de entrada e saída das execuções automáticas realizadas pelo *framework*.

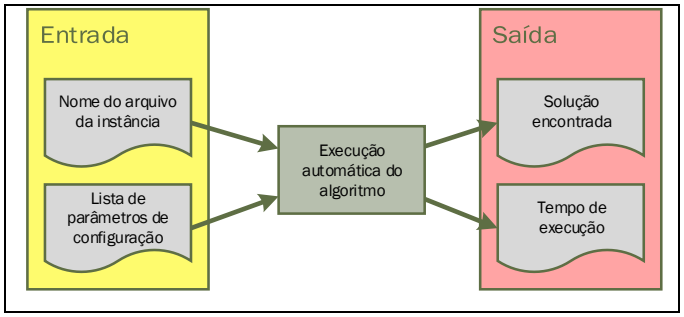


Figura 1.4 - Dados de entrada e saída das execuções automáticas do *framework*
Fonte: elaborado pelo autor

1.9 Organização deste documento

Para a apresentação deste doutoramento, o presente documento está organizado em oito capítulos. Este capítulo 1 que é o capítulo inicial de introdução. No capítulo 2 é apresentado um levantamento bibliográfico de abordagens estruturadas para desenvolvimento de heurísticas além realizar uma revisão de problemas de engenharia de produção com foco em *scheduling* (programação de tarefas). O capítulo 3 apresenta um levantamento sobre os métodos de análise estatística de resultados de pesquisas de problemas de engenharia de produção. No capítulo 4 tem-se a identificação dos padrões de análise e representação de dados de trabalhos baseados em problemas de *scheduling* e utilizando como meta-heurística principal a colônia de formigas (ACO). No capítulo 5 é descrito o processo de desenvolvimento de otimizadores baseados em heurísticas incluindo a etapa de análise estatística dos resultados. No capítulo 6 é apresentado o *framework* propriamente dito com a definição do processo de desenvolvimento, a especificação completa dos requisitos, os artefatos de análise e projeto, as etapas de desenvolvimento e testes de verificação e validação. No capítulo 7 é apresentada uma aplicação do *framework* em testes de validação, usabilidade e aplicação com dados reais de uma pesquisa. Por fim, no capítulo 8 estão descritas as conclusões gerais relativas ao desenvolvimento deste trabalho incluindo propostas de trabalhos futuros.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata da fundamentação teórica dos conceitos abordados na tese. Inicialmente foi feito um levantamento dos principais trabalhos a respeito de *frameworks* que implementam abordagens estruturadas para desenvolvimento de heurísticas. São levantados trabalhos de *framework* para desenvolvimento de heurísticas e trabalhos de *framework* com foco em análise estatística de resultados. É apresenta também uma revisão teórica dos conceitos de problemas de sistemas de produção aprofundando em problemas de programação de tarefas (*scheduling*) com uma revisão de trabalhos relacionados com o tema. O capítulo ainda apresenta os conceitos de experimentos computacionais com heurísticas, e por fim, apresenta conceitos para definição do conjunto de instâncias para problemas de *scheduling*, com um levantamento das principais bibliotecas de instâncias relacionadas a *scheduling* disponíveis.

2.1 Abordagens estruturadas para desenvolvimento de heurísticas

Muitos problemas de otimização não podem ser resolvidos por técnicas de otimização matemática clássica devido a sua complexidade e custo de se analisar o espaço de soluções viáveis. Para alcançar soluções de alta qualidade, são usados frequentemente algoritmos heurísticos para problemas de otimização combinatória. Esses algoritmos não tem a garantia de encontrar soluções ótimas, mas focam em oferecer uma compensação razoável entre tempo de execução e qualidade da solução. Com isso, muitas vezes são preferíveis em aplicações reais. Segundo Wagner et al. (2012), nas últimas décadas, o sucesso das técnicas de otimização heurística aplicados a domínios de problemas diferentes, incentivou o

desenvolvimento de uma ampla variedade de paradigmas de otimização que usam processos naturais como fonte de inspiração (como por exemplo, algoritmos genétivos, redes neurais, otimização de colônias de formigas, otimização por colônia de abelhas dentre outros). Para o desenvolvimento e aplicação de algoritmos de otimização heurística na ciência e na indústria, são necessários sistemas de *software* já validados, flexíveis e fáceis de utilizar. Estes sistemas devem apoiar os pesquisadores no desenvolvimento de novos algoritmos e também deve permitir aos usuários aplicar facilmente diferentes métodos de otimização em problemas específicos.

Os métodos heurísticos têm provado ser uma ferramenta abrangente na solução de problemas de otimização. No entanto, para Parejo et al. (2012), as heurísticas são geralmente baseadas em características específicas do problema em questão, o que torna a sua concepção e desenvolvimento uma tarefa complexa. Para resolver esse inconveniente, as meta-heurísticas aparecem como um avanço significativo que podem ser adaptados para incorporar o conhecimento específico de problemas em diversas áreas.

É muito difícil que exista uma ferramenta ótima universal para solucionar qualquer problema (HO; PEPYNE, 2001). O teorema NFL (*No Free Lunch* - não existe almoço grátis), apresentado por Wolpert e Macready (1997) tem sido usado como um argumento contra o uso de *frameworks* genéricos de otimização por heurísticas e meta-heurísticas. Neste sentido é necessário combinar um problema a ser resolvido e a técnica de otimização utilizada para resolvê-lo, a fim de obter soluções ótimas ou quase ótimas. Neste contexto *frameworks* são ferramentas personalizadas que nos permitem executar esta implementação da melhor forma possível em termos de custo e esforço de implementação (PAREJO et al., 2012). A finalidade destes *frameworks* é a otimização de tais mecanismos de adaptação de uma forma mais reutilizável possível e sem esforço.

Percebe-se, então, que existem diversas ferramentas computacionais ou *frameworks* para auxiliar pesquisadores da área de otimizadores. Entretanto a maioria foi desenvolvida pensando em ambientes específicos. Estes otimizadores foram baseadas em implementações de heurísticas e meta-heurísticas na solução dos problemas, além de estarem fortemente embasados em técnicas e desenvolvimento orientado a objetos e padrões de projeto (PAREJO et al., 2012).

2.1.1 Definição de *framework*

Em pesquisas na área de otimização, observa-se que vários trabalhos utilizam o termo *framework*. Todavia o conceito de *framework* é bastante amplo. Johnson (1992) define *framework* como um projeto reutilizável de um programa ou parte de um programa expresso como um conjunto de classes ou componentes. Desde que *frameworks* são projetos reutilizáveis, não apenas o código, eles possuem uma arquitetura diferenciada em relação à maioria dos *software*s. *Frameworks* são projetados por especialistas em um domínio específico e, em seguida, utilizados por usuários não especialistas, que buscam soluções para problemas específicos. Segundo Parejo et al. (2003) um *framework* é algo mais que uma biblioteca. Um *framework* não é apenas um conjunto de funções, mas também contém um projeto que pode ser usado, adaptado e completado de modo a atender as necessidades do usuário. Um *framework* define uma estrutura inteira para uma aplicação.

Para Johnson (1992) um *framework*, na ótica da engenharia de *software*, é definido como sendo uma aplicação que reúne inúmeros procedimentos comuns entre vários projetos de *software* provendo uma funcionalidade genérica. Um *framework* pode ser direcionado para executar uma funcionalidade específica, através de configurações, durante a programação de uma aplicação, sendo quem dita o fluxo de controle da aplicação desenvolvida. No estudo de heurísticas e meta-heurísticas para problemas de otimização combinatória, a palavra *framework* pode definir um conjunto de padrões e premissas que direcionam a solução de um determinado problema (SUNDARAMOORTHY; MARAVELIAS, 2011).

Um *framework* pode ser bem-sucedido se somente se alguns critérios importantes forem satisfeitos (CAHON et al., 2004):

- Facilidade de utilização: o *framework* tem de ser relativamente fácil de utilizar;
- Flexibilidade/adaptabilidade: os algoritmos evolutivos integrados podem ser adaptados a uma grande variedade de problemas apenas parametrizando-os ou especializando-os;
- Tem que ser aberto: a plataforma deve permitir o projeto e integração de novos algoritmos pela reutilização dos algoritmos ou de partes dele;

- Portabilidade: para satisfazer um grande número de usuários, o *framework* deve suportar diferentes arquiteturas e seus sistemas operacionais associados.
- Desempenho e robustez: como as aplicações de otimização são muitas vezes demoradas o problema de desempenho é crucial. Além disso, a execução dos algoritmos deve ser robusta para garantir a confiabilidade dos resultados.

Parejo et al. (2003) apresenta um conjunto de pontos positivos e negativos da utilização de um *framework*, o que pode ser visto no Quadro 2.1.

Quadro 2.1 - Pontos positivos e negativos do uso de *frameworks*

Pontos positivos	Pontos negativos
Reduz o esforço de implementação e habilita a aplicação de técnicas e variantes diferentes com pouco esforço de desenvolvimento	Curva de aprendizado de utilização do *framework* é maior
Ferramentas adicionais para ajudar na solução de problemas (monitoramento, relatórios, computação paralela e distribuída)	Necessidade de conhecimento avançado para ajuste dos algoritmos e inflexibilidade para adaptar o uso de algumas variantes heurísticas
Implementação otimizada e livre de erros (exceto as extensões e adaptações criadas pelos usuários ou erros não detectados presentes no *framework*)	Induz a complexidade (quando se está realizando a procura por erros e testes) e dependências adicionais
Novos usuários com pouco conhecimento podem usar o *framework* não somente como ferramenta para ambiente de desenvolvimento de aplicações, mas sim como uma metodologia a ser seguida	A escolha do *framework* correto pode ser um problema, já que a mudança para outro tem um custo elevado, eles oferecem suporte a diversas características e não há *benchmark*s comparativos entre *frameworks* na literatura

Fonte: Parejo et al. (2012)

2.1.2 Alguns *frameworks* existentes na literatura

Com o intuito de conhecer os *frameworks* disponíveis na literatura foi feita uma pesquisa de levantamento de trabalhos publicados sobre o assunto tratado nesta tese. O foco principal foi identificar a existência de *frameworks* que possuem recursos para realizar análise estatística e representação dos resultados encontrados em pesquisas de otimização que fazem uso de heurísticas e meta-heurísticas. A pesquisa foi realizada utilizando o portal de busca de publicações *Engineering Village*[1], que tem como base o COMPENDEX (*COMPuterized ENgineering inDEX*). Os termos

[1] https://www.engineeringvillage.com/

utilizados, com filtro no objetivo, título ou resumo (*Subject/Title/Abstract*), foram: "*Optimization Framework*", "*Heuristic Optimization Framework*", "*Metaheuristic Optimization Framework*", "*Otimization Framework Statistical Analysis*" e "*Framework Data Analysis*". Muitos trabalhos não foram considerados no levantamento porque não tinham o foco deste trabalho, ou eram monetizados, ou não estavam disponíveis para *download* do arquivo. Em um segundo momento uma busca secundária foi realizada baseada nos principais trabalhos referenciados. Esta busca resultou em um total de vinte e cinco (25) *frameworks* na área, publicados entre 2002 e 2014, que são apresentados, em ordem crescente de data de publicação e descritos a seguir.

No projeto MALLBA (ALBA et al., 2002) os autores abordam a resolução de problemas de otimização combinatória usando soluções algorítmicas implementados em C++. Oferece três famílias de métodos de resolução de problemas: exatos, heurísticos e híbridos. Além disso, para cada método de resolução, fornece três implementações diferentes: sequencial, paralelo para redes locais (LAN - *Local Area Network*) e paralelo para redes de longa distância (WAN - *Wide Area Network*). As principais características são: integração de todos os soluções sob os mesmos princípios de projeto, facilidade para alternar entre os processos de otimização sequenciais para paralelos e cooperação entre processos para fornecer soluções híbridas mais poderosas. Possui uma arquitetura de *software* flexível e extensível onde novas soluções podem ser facilmente adicionadas e camadas de comunicação alternativas podem ser usadas. Cada método de resolução é encapsulado em uma solução estando disponíveis soluções exatas tais como *Divide e Conquer* (DC) e *Branch and Bound* (BnB), métodos heurísticos tais como Simulated Annealing (SA), Tabu Search (TS), Algoritmos Genéticos (GA) e Algoritmos Meméticos (MA). Além disso, as técnicas híbridas foram implementadas combinando as soluções anteriores, por exemplo, GA+TS, GA+SA, BnB+SA.

O *framework* Hotframe (*Heuristic Optimization Framework*) (FINK; VOB, 2002) é uma ferramenta desenvolvida em C++ que fornece componentes de *software* reutilizáveis no domínio da meta-heurística com ênfase especial nas semelhanças e variabilidades. O modelo resultante constitui a base para o projeto do *framework*. A arquitetura define a colaboração entre os componentes de *software* (em particular no que diz respeito à interface entre componentes genéricos meta-heurísticos e complementos específicos do problema). O *framework* é descrito em relação à sua arquitetura, incluindo componentes, implementação e aplicação. São disponibilizados

as estruturas de classes das meta-heurísticas de busca local iterativa, *Simulated Annealing*, busca Tabu e algoritmos evolutivos. O *framework* pode ser estendido em várias direções com a inclusão de novos componentes heurísticos ou podendo ser adicionados novos componentes padrões específicos de problemas, resultando em um grande conjunto implementados de espaços de solução e componentes correspondentes, o que permite a aplicação em muitos tipos de problemas comuns de forma simples e fácil. O projeto não está mais ativo e não possui mais atualizações disponíveis.

Keijzer et al. (2002) apresentaram a biblioteca de objetos evolutivos (EOLib - *Evolving Object Library*), um *framework* orientado a objetos desenvolvido em C++ para computação evolucionária que prove um conjunto flexível de classes para a construção de aplicações de computação evolucionária que podem utilizar de heurísticas e meta-heurísticas. O objetivo do projeto é ser capaz de evoluir qualquer estrutura de dados (objeto) com o intuito que o valor encontrado tenha significado. O *framework* está centrado nas interfaces onde qualquer objeto pode evoluir se existir uma interface que possa implementar esta evolução. As soluções desenvolvidas com esta biblioteca não estão limitadas aos paradigmas básicos de computação evolucionária, tais como algoritmos genéticos, estratégias evolucionárias, programação evolucionária ou programação genética, seja ao nível da evolução da população da solução ou da aplicação dos operadores de variação. O *framework* possui ainda algumas facilidades para aplicações de computação evolucionária, tais como pontos de observação para parada e reinício da execução da aplicação, disponibilização de múltiplas análises estatísticas, representação gráficas dos resultados, dentre outros.

Andreatta et al. (2003) apresentaram um *framework* para heurísticas de busca local, desenvolvido em C++, para problemas de otimização combinatória, aqui nomeado de FLSHCOP, com o foco no desenvolvimento e comparação, de forma sistemática, de diferentes estratégias algorítmicas e parâmetros para um mesmo problema. Esta comparação esta embasada não somente nas diferentes linguagens de implementação, mas também nas diferentes arquiteturas. O *framework* propõe a utilização de um conjunto de padrões de projeto para modelar diferentes aspectos envolvendo heurísticas de busca local, tais com algoritmos para a construção de soluções iniciais, métodos para geração de vizinhanças e critérios de seleção de movimento. As características de encapsulamento e abstração permitem

comparações imparciais entre as diferentes heurísticas, reuso de código e facilidade para extensões.

O EasyLocal++ (GASPERO; SCHAERF, 2003) é um *framework* para modelagem e resolução de problemas de otimização combinatória através de meta-heurísticas de busca local. A estrutura (atualmente na versão 3.0) é inteiramente escrita em C ++ fazendo uso de modelos de metaprogramação para conseguir uma forte separação de interesses e desempenho. Para resolver um problema, basta implementar métodos para calcular a função de custo do problema especificado e enumerar os movimentos de busca locais aplicados ao problema. O *framework* chama esses métodos para resolver o problema usando uma das heurísticas implementadas (por exemplo, *Simulated Annealing*, busca TABU dentre outros). Ele fornece uma estrutura de modularização baseada em princípios para o projeto de algoritmos de busca local e apresenta várias vantagens no que diz respeito à implementação direta do algoritmo a partir do zero, não apenas em termos de reutilização de código, mas também em metodologia e clareza conceitual. Além disso, é facilmente extensível por meio de novas derivações de classe e composições. Ele também oferece um módulo de teste que apresenta os resultados em gráficos e aplica análises estatísticas.

O *framework* NP-Opt (MENDES et al., 2001; MENDES, 2003) é uma ferramenta orientada a objetos desenvolvida em Java, com foco na utilização de algoritmos genéticos, algoritmos meméticos e uma estratégia de *multiple start*. O objetivo principal é tratar de problemas mono-objetivo tais como: sequenciamento em máquina simples, máquinas paralelas e *flowshop* com famílias de tarefas; *gate matrix layout* e ordenamento de Genes. Os três primeiros problemas pertencem área de sequenciamento da produção. O problema de *gate matrix layout* tem relação com o desenho de circuitos elétricos e o último pertence à área de bioinformática. Para o problema de sequenciamento em máquina simples há ainda uma heurística construtiva chamada ATCS que é utilizada como base para comparação de desempenho. O *framework* implementa algumas características tradicionais para soluções genéticas e meméticas tais como: estrutura populacional, controlador *Fuzzy* para parâmetros evolutivos e múltiplas populações.

O *framework* FOM (*A Framework for Metaheuristic Optmization* - *Framework* para Otimização de Meta-heurísticas) (PAREJO et al., 2003) foi desenvolvido em Java com o foco na construção de hyper-heurísticas que são métodos que buscam a construção de outras heurísticas para a solução de um problema particular. A ideia

básica por trás da estrutura é separar o lado do problema dos algoritmos meta-heurísticos, permitindo que isso reutilize diferentes componentes meta-heurísticos em diferentes problemas. Trabalha com problemas de otimização mono-objetivo e disponibiliza meta-heurísticas tais como: colônia de formigas, algoritmos evolutivos, GRASP, *Simulated Annealing*, VNS, busca TABU, dentre outros. O *framework* tenta evitar implementações prontas para problemas específicos, que reduzem ao máximo o tempo computacional para obter uma solução insignificantemente melhor. No entanto, a ferramenta tenta tornar esses métodos aplicáveis para um número crescente de problemas de uma maneira fácil, simples e rápida, devendo apenas descrever o problema e obter soluções utilizando as meta-heurísticas. Outra característica importante é que a ferramenta apresenta meios de análises estatísticas dos resultados encontrados.

O *framework* ParaDisEO (CAHON et al., 2004) foi desenvolvido em C++ e é dedicado ao projeto de meta-heurísticas para otimização de problemas de natureza contínua, discreta e combinatória. É proposto uma abordagem flexível para implementação de algoritmos evolutivos paralelos e distribuídos que tratam de problemas multi-objetivos dentro deste ambiente. As extensões incluem métodos de busca local (busca descendente, *Simulated Annealing* e busca Tabu para otimização combinatória, busca baseada em gradientes para otimização numérica), mecanismos de hibridização (busca local por acoplamento e busca evolutiva global) e modelos paralelos/distribuídos. O *framework* se encarrega da carga para gerenciar explicitamente a comunicação e a concorrência. Além disso, os modelos podem ser implementados eficientemente em multiprocessadores compartilhados e em redes distribuídas.

O *framework* ECJ (*Java Evolutionary Computation Toolkit*) (LUKE et al., 2004; WHITE, 2012; LUKE, 2015) é uma rica biblioteca de programação evolutiva portátil de computação evolutiva baseada em Java. É altamente flexível, pois quase todas as classes são dinamicamente determinadas em tempo de execução por um arquivo de parâmetro fornecido pelo usuário. Além disso, todas as estruturas são dispostas para serem facilmente modificáveis permitindo execuções eficientes com paralelismo e *multi-threading*. O *framework* está focado em algoritmos genéticos, estratégias evolucionárias, múltiplas populações e espécies, otimizações multi-objetivos NSGA-II e SPEA2, otimização por enxame de partículas, dentre outros. Muitas outras características estão presentes, incluindo possibilidades de pontos de verificação,

interface gráfica com visualização de gráficos dos resultados, gerador de números aleatórios, dentre outras. ECJ é o mais popular sistema de pesquisa de computação evolutiva baseado em Java, que está sendo desenvolvido desde 1998. Possui uma excelente documentação, incluindo o detalhamento necessário ao usuário para desenvolvimento de novas soluções, mas a falta de uma interface gráfica mais amigável o que aumenta a curva de conhecimento na utilização de todos os recursos disponíveis. Para instalar o *framework* é necessário baixar e extrair os arquivos principais do ECJ e todas as bibliotecas de otimização, ajustar vários arquivos de configuração, caminhos de classe e construir as fontes, o que não é uma tarefa muito fácil para usuários inexperientes.

Lau et al. (2005) propuseram uma solução de *software* chamada como *framework* para busca Tabu (TSF - *Tabu Search Framework*), que é um *framework* de *software* C ++ genérico para implementação de pesquisa tabu. O *framework* se diferencia em relação à reutilização de códigos através do uso de um conjunto bem definido de classes genéricas abstratas que definem claramente seus papéis colaborativos no algoritmo. Além disso, o *framework* incorpora um processo centralizado e um mecanismo de controle que melhora a busca com inteligência. Isso resulta em um *framework* genérico que é capaz de resolver uma ampla gama de problemas de otimização combinatória usando várias técnicas de busca tabu e estratégias adaptativas, mostramos ser capaz de obter soluções de qualidade dentro de uma implementação razoável, bem como tempo de computacional reduzido.

O *framework* Open Metaheuristic (oMetah) (DRÉOK et al., 2006) é uma biblioteca voltada para a concepção de metaheurísticas (por exemplo, algoritmos genéticos, busca de TABU, *Simulated Annealing*, algoritmos de colônias de formigas, etc.). É focada na concepção, teste e comparação de algoritmos meta-heurísticos. Usa várias interfaces de comunicação para ligar os algoritmos de otimização aos problemas. É projetado para que se possa facilmente adicionar um problema, ou algoritmo ou interface. Além disso, ele vem com um conjunto de *scripts* para testar os algoritmos e produzir relatórios com gráficos e análises estatísticas. O projeto não está mais ativo e sua última atualização foi em 2006.

O *framework* Open Beagle (GAGNÉ; PARIZEAU, 2006) foi desenvolvido em C++ para trabalhar com computação evolutiva (EC - *Evolutionary Computation*). Fornece um ambiente de *software* de alto nível para desenvolver qualquer tipo de EC,

com suporte para vários recursos e estratégias. A arquitetura segue os princípios de programação orientada a objetos, onde as abstrações são representadas por objetos com poucos acoplamentos onde é comum e fácil reutilizar código. Foi projetado para fornecer um ambiente genérico, fácil de usar, portátil, eficiente, robusto, elegante e gratuito.

O *framework Grid-enabled Framework for Exact Optimization Algorithms* (ZUNINO et al., 2007), aqui nomeado de GeFEOA, é um *framework* de otimização para construção de métodos exatos voltados a problemas mono e multi-objetivos. O ambiente guia a implementação de métodos distribuídos em *grid* utilizando uma arquitetura mestre-escravo para paralelização. O ambiente implementa a solução *Branch & Bound* com os métodos paralelos de otimização exatos para a exploração do espaço de busca (TPM, *e-constrained* e PPM).

O *framework* OAT (*Optimization Algorithm Toolkit*) (BROWNLEE, 2007) é um projeto de *software* de código aberto escrito em Java que fornece um conjunto de domínios de problemas de otimização de inteligência computacional com instâncias de problema, algoritmos clássicos e de última geração, visualização, gráficos e muito mais. Fornece uma série de algoritmos evolutivos juntamente com *benchmark* de instâncias para problemas de otimização de função contínua e uma série de algoritmos de otimização por colônia de formigas aplicados a um grupo pequeno de instâncias do problema do caixeiro viajante (TSP - *Travelling Salesmam Problem*) provenientes do TSPLIB[2]. Possui uma interface de programação de aplicações (API - *Application Programming Interface*) rudimentar para explorar os algoritmos e instâncias de problema, bem como uma interface gráfica básica. Os algoritmos foram projetados para serem auto-contidos de modo que sua implementação pudesse ser adaptada e estendida para necessidades específicas de aplicação. Possui ambiente com intuito de promover as melhores práticas em projeto experimental, execução e análise, fornecendo uma metodologia matricial simples de algoritmos e problema que permite que sejam exportados arquivos com o intuito de promover a reprodutibilidade e compartilhamento. Ainda estão disponíveis rotinas de execução dos algoritmos em lotes e uma série de ferramentas de teste de hipóteses estatísticas para desenvolvimento de relatórios e interpretação dos resultados.

[2] http://comopt.ifi.uni-heidelberg.de/*software*/TSPLIB95/

O *framework* Wallace (TIMPERLEY; STEPNEY, 2008) apresenta uma estrutura de computação evolutiva, que consegue ser fácil de usar e ao mesmo tempo genérica, através de uma linguagem específica de domínio, e simultaneamente consegue eficiência através da meta-programação, bem como suporte o paralelismo. Inclui também um novo modelo de representação múltipla de desenvolvimento individual, suportado pela meta-programação. Emprega uma arquitetura baseada em componentes, onde algoritmos são descritos em termos de uma série de componentes personalizáveis, cada um responsável por implementar alguma parte da evolução incremental empregados pela maioria dos algoritmos evolutivos. O foco principal é atingir tanto a facilidade de uso e alto desempenho. O *framework* explora o poder das linguagens específicas de domínio (DSLs - *Domain Specific Languages*) com o processo de reflexão computacional, com a capacidade de escrever e modificar partes de um programa em tempo de execução. Tal abordagem não só permite ao usuário escrever algoritmos em termos mais naturais, mas também mantém a expressividade da linguagem de programação subjacente. Para manter o alto desempenho utiliza-se de meta-programação que permite escrever código de alto nível, mas as mesmas características de abordagens de baixo nível em termos de desempenho, sendo escolhida a linguagem Julia (BEZANSON et al., 2012) por ser dinâmica de alto nível, relativamente jovem, projetada para computação técnica, baseada em múltiplos despachos e com tipagem forte.

O *framework* JCLEC (VENTURA et al., 2008) é um sistema de *software* Java para o desenvolvimento de aplicações computacionais evolucionárias. Este sistema foi concebido como um *framework*, aplicando padrões de projeto para maximizar a sua reutilização e adaptabilidade a novos paradigmas com um mínimo de esforço de programação. A arquitetura do JCLEC compreende três módulos principais: o núcleo contém todas as definições de tipos abstratos e suas implementações, o executor de experimentos é um ambiente de *scripts* para executar algoritmos em modo batch e uma interface gráfica que permite aos usuários configurar um algoritmo, executá-lo interativamente e visualizar os resultados obtidos. O sistema possui uma interface gráfica que facilita tarefas como a configuração, execução e verificação de resultados, disponibilizando uma variedade de algoritmos evolucionários (algoritmos clássicos, algoritmos multi-objetivos NSGA-II e SPEA2, algoritmos meméticos, dentre outros) e representações já desenvolvidas. Além disso, este sistema pode ser usado por

pesquisadores, porque é fácil de estender e permite definir ambientes de teste das soluções.

O HyFlex (*A Flexible Framework for the Design and Analysis of Hyper-heuristics*) (BURKE et al., 2009) apresenta um *framework*, desenvolvido em Java, inspirado em abordagens hiper-heurísticas, cujo objetivo é automatizar o processo de seleção e combinação de heurísticas mais simples ou geração de novas heurísticas a partir de componentes de heurísticas existentes, a fim de resolver difíceis problemas computacionais de busca. A principal motivação é elevar o nível de generalidade em que as metodologias de busca podem atuar se distinguido de outros algoritmos porque operam em um espaço de busca de heurísticas (ou componentes heurísticos) ao invés de diretamente no espaço de busca de soluções para o problema. O *framework* trabalha em um alto nível de abstração e muitas vezes não tem conhecimento do domínio, tendo somente acesso a um conjunto de heurísticas de baixo nível (estruturas de vizinhança) que podem ser chamadas mesmo não conhecendo como elas funcionam. Uma vez que um algoritmo hiper-heurístico tenha sido desenvolvido, ele pode ser aplicado a um novo problema através da substituição do conjunto de heurísticas de baixo nível e a função de avaliação. O *framework* considera quatro tipos de heurísticas de baixo nível: heurísticas mutacionais ou perturbativas (realizam pequenas mudanças na solução trocando, removendo, adicionando ou excluindo componentes desta solução), heurística *Hill-Climbing* (iterativamente faz pequenas mudanças na solução, aceitando as melhorias da solução), heurística *Ruin & Recreate* (faz a destruição parcial da solução para logo depois reconstruir ou recriar a uma nova solução) e heurística de *Crossover* (combina componentes de duas solução de entrada para produzir uma nova solução). O usuário deve projetar estratégias de alto nível que combinam inteligentemente este conjunto de heurísticas fornecidas na busca das melhores soluções.

O *framework* MOEAT (*Multiobjective Evolutionary Algorithms Tool*) (SAĞ; ÇUNKAŞ, 2009) é uma ferramenta desenvolvida em linguagem C#, usando uma variedade de algoritmos evolutivos para a resolução de problemas multi-objetivos (MOEAs - *Multiobjective Evolutionary Algorithms)*, oferecendo um ambiente poderoso para vários tipos de tarefas de otimização. Apresenta muitos recursos úteis tais como a visualização do progresso de execução e os resultados da otimização em um modo dinâmico ou estático e configurações de variáveis de decisão. Da mesma forma que o jMetal, o MOEAT possui vários algoritmos conhecidos para otimização multi-

objetivo. O *framework* possui uma interface gráfica robusta e um editor próprio que simplifica a inclusão de novas funções e restrições para os problemas em questão. Sua interface gráfica possibilita a configuração de testes de forma simples e rápida, além da visualização da fronteira de Pareto para problemas com dois objetivos. O *framework* pode ser estendido para uso de outros algoritmos não evolucionários tais como *Simulated Annealing* e otimização por enxame de partículas (PSO - *Particle Swarm Optimization*).

Tavares e Godinho Filho (2009) apresentaram um *framework*, aqui chamado de FSHMACF, computacional desenvolvido em Java com padrões de projeto que permite a prototipagem de um grande conjunto de variações de heurísticas baseadas em sistemas de formigas. Foi definido uma metodologia de desenvolvimento de algoritmos com o objetivo de reduzir significativamente o tempo de implementação e facilitar a comparação entre as técnicas propostas. Os algoritmos implementados são Ant-Cycle, AS (*Ant-Quantity e Ant-Desity*), ACS (*Ant Colony System*), MMAS (*Max Min Ant System*) e Múltiplo AS (Homogênio e Heterogêneo). Devido a alta robustez do algoritmo de otimização por colônia de formigas o pesquisador pode focar na modelagem do problema e no estabelecimento de parâmetros de configuração, deixando assim a execução do mesmo a cargo de um algoritmo já previamente implementado. O *framework* proposto é um componente independente de *software* que recebe informações de configuração e um problema de otimização combinatória representado através de um formato conhecido sendo processado a resposta do problema por meio de seus algoritmos internos. Este componente é composto internamente dos seguintes elementos: pelo menos uma colônia que contém pelo menos uma formiga. Possui também um espaço de busca e, caso necessário, pode utilizar um módulo de troca.

Framework EvA2 (*Evolutionary Algorithms framework version 2*) (KRONFELD et al., 2010) foi implementado em Java e apresenta uma estrutura abrangente de otimização meta-heurística com ênfase em algoritmos evolutivos. Integra vários métodos de otimização tais como estratégias evolutivas, algoritmos genéticos, evolução diferencial, otimização de enxames de partículas, bem como técnicas clássicas como *Nelder-Mead-Simplex* ou *Simulated Annealing*. Devido à estrutura modular, os operadores heurísticos podem ser facilmente trocados entre métodos de otimização e aplicados a qualquer problema. São implementadas técnicas melhoradas para métodos de otimização multimodal e multiobjetivo. Incluem-se funções de

benchmark padrão para valores reais, ruidosos, dinâmicos e multiobjetivos, bem como os conhecidos problemas combinatórios. O *framework* pode ser facilmente estendida por uma classe de problema definida pelo usuário ou interfaceada com funções executáveis de destino, permitindo investigar o desempenho de diferentes algoritmos de otimização ou comparar os efeitos de operadores heurísticos.

No trabalho de Durillo e Nebro (2011) foi apresentado o jMetal, um *framework* de otimização multi-objetivo desenvolvido em linguagem Java. O objetivo principal está na inclusão dos métodos e problemas de otimização multi-objetivos, disponibilizando um modelo a ser seguido no processo de inclusão. O jMetal disponibiliza a implementação de algoritmos bastante conhecidos para problemas multi-objetivos tais como NSGA-II, PAES2, PESA-II, OMOPSO, MOCell, AbYSS, MOEA/D, Densea, CellDE, GDE3, FastPGA, dentre outros. Também disponibiliza um rico conjunto de problemas de teste encontrados na literatura, incluindo problemas clássicos, famílias de problemas e problemas com restrições. Outro ponto é a implementação de indicadores de qualidade conhecidos tais como *Hypervolume*, *Spread*, *Epsilon*, dentre outros. Possui suporte para estudos experimentais incluindo a geração automática das tabelas de resultados em LaTex após a aplicação dos indicadores de qualidade, e análise estatística com a comparação estatística em pares pelo uso de teste de Wilcoxon, teste de normalidade por Kolmogorov-Smirnov, e dependendo da resultado da normalidade executa o teste de Kruskal-Wallis ou teste ANOVA, além de apresentar os resultados em gráficos principalmente o *box-plot* gerados pelo R sumarizando os resultados. Todos estes recursos são apoiados por uma interface gráfica para dar suporte na solução de problemas e na execução de estudos experimentais.

O *framework* Opt4J (LUKASIEWYCZ et al., 2011) está focado em uma estrutura modular para a otimização meta-heurística de tarefas complexas de otimização, decompondo-as em subtarefas que podem ser projetadas e desenvolvidas separadamente. Como essas subtarefas são geralmente correlacionadas têm-se um problema para realizar a otimização de forma separada e a estrutura deve ser capaz de otimizar as subtarefas simultaneamente. Para isso, impõe-se uma distinção entre a representação genética (genótipo) e a representação de uma solução do problema de otimização (fenótipo). Um genótipo composicional e os operadores apropriados permitem a separação do desenvolvimento e dos testes de uma otimização de subtarefas por meio de um desacoplamento restrito. O

framework foi desenvolvido em Java utilizando padrões de projeto de injeção de dependência. Este padrão de projeto separa o comportamento da resolução de dependência e, assim, permite a implementação dos conceitos propostos idealmente com fracos acoplamentos. Como resultado, a configuração das dependências é feita em módulos separados. Uma interface gráfica do usuário permite a seleção e configuração desses módulos e aprimora ainda mais o rápido desenvolvimento e testes.

O *framework* METSlib (MAISCHBERGER, 2011) também foi desenvolvido em C++ que permite a construção de meta-heurísticas de busca local para problemas de otimização mono-objetivos. Possui implementados os métodos baseados em busca local *Randon Restart*, *Simulated Annealing* e busca Tabu, mas permitindo que o usuário crie suas soluções a partir das soluções base. O modelo e os algoritmos são modulares: qualquer algoritmo de pesquisa pode ser aplicado ao mesmo modelo. Por outro lado, nenhuma suposição é feita sobre o modelo, podendo trabalhar em qualquer tipo de problema. Depois de implementado o modelo na estrutura do problema, a biblioteca facilita o teste de diferentes estratégias de busca Tabu ou até mesmo algoritmos diferentes (*Simulated Annealing* ou outros algoritmos baseados na pesquisa local) com poucas linhas de codificação.

O *framework* DEAP (*Distributed Evolutionary Algorithms in Python*) (FORTIN et al., 2012; RAINVILLE et al., 2012) propõe uma nova estrutura de computação evolutiva para a prototipagem rápida e teste de ideias. O objetivo é tornar os algoritmos explícitos e as estruturas de dados transparentes para o usuário. Funciona com o mecanismo de paralelismo e com multiprocessamento de tarefas distribuídas. O *framework* inclui algoritmos genético usando inúmeras representações, estratégias evolucionárias (incluindo CMA-ES), otimização multi-objetivo (NSGA-II, SPEA2, MO-CMA-ES), co-evolução (cooperativa e competitiva) de múltiplas populações, pontos de verificação, *benchmark*s que contém as funções de teste mais comuns, além de algoritmos alternativos tais como otimização de enxames de partículas, evolução diferencial, dentre outros. O DEAP é um *framework* leve que se concentra em fornecer operadores básicos da computação evolutiva e mecanismos gerais para criar facilmente partes personalizadas para implementar algoritmos evolucionários sofisticados. Para isso, utiliza a linguagem de *script*s Python para fornecer o mecanismo essencial para a montagem destas partes de algoritmos evolucionários em sistemas de computação evolucionária coerentes.

O *Heuristic Lab* (WAGNER et al., 2012; ELYASAF; SIPPER, 2014) é um *framework* para algoritmos heurísticos e evolucionários desenvolvido pelos membros do laboratório de algoritmos heurísticos e evolucionários (HEAL - *Heuristic and Evolutionary Algorithms Laboratory*). O principal objetivo é fornecer um sistema abrangente para o desenvolvimento de algoritmos, testes, análise e, em geral, a aplicação de métodos de otimização heurística em problemas complexos. O desenvolvimento foi feito em C# para a plataforma Windows, mas já existem esforços para disponibilizar o sistema para usuários de outros sistemas operacionais. HeuristicLab é muito fácil de instalar e executar, mas possui documentação pouco aprofundada, o que dificulta ao usuário o entendimento de todas as atividades possíveis na ferramenta. O *framework* possui uma interface gráfica confortável e rica em recursos que permite a codificação fácil de novos problemas e a adaptação dos já existentes. Possui vários algoritmos heurísticos e problemas de otimização já estão implementados (algoritmos Genéticos, programação genética, busca Tabu, *Simulated Annealing*, dentre outros). Permite que os usuários possam de forma gráfica visualizar, criar e reutilizar *plugins* para integrar novos recursos e estender as funcionalidades disponíveis. Outro ponto que se destaca é o projeto de experimentos que permite aos usuários definir e executar grandes experimentos selecionando algoritmos, parâmetros e problemas. Os resultados encontrados podem ser visualizados em gráficos estatísticos interativos que podem ser facilmente configurados. Suporta ainda a execução paralela e distribuída de algoritmos em sistemas de mais de um processador e em *clusters*.

2.1.3 Avaliação dos *frameworks* com foco na análise estatística dos resultados

Com o levantamento realizado, e analisando os vinte e cinco (25) *frameworks* encontrados, fica claro a diversidade de trabalhos publicados confirmando o teorema NFL (WOLPERT; MACREADY, 1997) que indica que não existe uma ferramenta ótima universal que resolva todo e qualquer problema. Percebe-se que muitos trabalhos são desenvolvidos especificamente para solucionar uma classe de problemas ou que utiliza somente um tipo de modelagem heurística com variações. Este levantamento teve como objetivo principal identificar quais destes *frameworks* realizavam tratamento

dos resultados encontrados através de análises estatísticas e representação destes resultados, que é o foco da presente tese. Muitos trabalhos não especificam claramente tais recursos ou não possuem implementações disponíveis capazes de realizar estas análises.

Um dos elementos importantes para garantir a validade de qualquer estudo é a capacidade de realizar testes estatísticos nos seus resultados. Portanto, uma das tarefas na solução de problemas de otimização (e em qualquer estudo com um componente empírico) é a análise estatística de dados experimentais e dos resultados encontrados. Existem duas maneiras diferentes de apoiar esta característica. A primeira é fornecer mecanismos de integração com sistemas de análise estatística tais como Projeto R[3], SPSS[4] ou Statistica[5]. A segunda é implementar as funcionalidades e os componentes para análise estatística no próprio *framework*. Uma das desvantagens da primeira abordagem é que o usuário deve importar dados para o sistema de análise estatística, realizar os testes estatísticos, interpretar os resultados e retornar ao *framework* para alterar parâmetros ou implementações, se necessário. Esta abordagem libera o *framework* da implementação dos testes estatísticos dentro do próprio código. Além disso, os sistemas de análise estatística são geralmente mais completos do que implementações de testes integrados em *frameworks*. Por outro lado, o uso da segunda estratégia permite ao *framework* automatizar os testes e a troca de dados associados, mostrando os resultados integrados em sua interface e até mesmo interagindo de forma automática com os resultados dos testes.

O Quadro 2.2 apresenta os *frameworks* que foram encontrados no levantamento realizado, segundo algumas características específicas que foram identificadas, incluindo recursos ou componentes de análise estatística e representação de dados.

Quadro 2.2 - *Frameworks* encontrados de acordo com características específicas

[3] https://www.r-project.org/
[4] https://www-01.ibm.com/*software*/br/analytics/spss/
[5] http://www.statsoft.com.br/

N.	Nome	Autor	Linguagem	Tipo	Interface Gráfica	Execuções em lotes	Análise estatística	Execuções Paralelas	Técnicas de Otimização
1	MALLBA	Alba et al. (2002)	C++	Biblio.				✓	DC, Bnb, DP, SD, SA, TS, EA, PSO, ACO
2	Hotframe	Fink e Voβ (2002)	C++	Biblio.					LS, SA, TS, EA
3	EOLib	Keijzer et al. (2002)	C++	Biblio.			✓		EA
4	FLSHCOP	Andreatta et al. (2003)	C++	Biblio.					LS, SA, TS
5	EasyLocal++	Gaspero e Schaerf (2003)	C++	Biblio.			✓		SD, SA, TS, VNS
6	NP-Opt	Mendes et al. (2001); Mendes (2003)	Java	Aplic.	✓				EA (GA e MA) e MS
7	FOM	Parejo et al. (2003)	Java	Aplic.	✓	✓	✓		SD, SA, TS , GRASP, VNS, EA, ACO, HH
8	ParaDisEO	Cahon et al. (2004)	C++	Biblio.				✓	SD, SA, TS, VNS, EA, PSO, MO
9	ECJ	Luke et al. (2004); White (2012); Luke	Java	Aplic.	✓	✓		✓	SD, EA, PSO, MO
10	TSF	Lau et al. (2005)	C++	Aplic.	✓				TS
11	oMetah	Dréok et al. (2006)	C++	Biblio.	✓	✓	✓		EA (GA), TS, SA, ACO
12	Open Beagle	Gagné e Parizeau (2006)	C++	Biblio.					EA
13	GeFEOA	Zunino et al. (2007)	MPICH-G2	Biblio.				✓	BnB
14	OAT	Brownlee (2007)	Java	Aplic.	✓	✓	✓		SD, EA, AIS, ACO
15	Wallace	Timperley e Stepney (2008)	Julia	Biblio.	✓			✓	EA
16	JCLEC	Ventura et al. (2008)	Java	Aplic.	✓	✓	✓	✓	EA (GA e MA), MO
17	HyFlex	Burke et al. (2009)	Java	Biblio.					HH
18	MOEAT	(Sag e Çunkas (2009)	C#	Aplic.	✓	✓	✓		MO, EA, SA, PSO
19	FSHMACF	Tavares e Godinho Filho (2009)	Java	Biblio.					ACO
20	EvA2	Kronfeld et al. (2010)	Java	Aplic.	✓	✓			SD, SA, EA, PSO, SS, MO
21	jMetal	Durillo e Nebro (2011)	Java	Biblio.	✓	✓	✓		MO, EA (GA), PSO
22	Opt4J	Lukasiewycz et al. (2011)	Java	Aplic.	✓	✓		✓	SA, EA, PSO, MO
23	METSlib	Maischberger (2011)	C++	Biblio.					LS, MS, SA, TS
24	DEAP	Fortin et al. (2012); Rainville et al. (2012)	Python	Biblio.				✓	EA (GA), MO, PSO
25	Heuristic Lab	Wagner et al. (2012); Elyasaf e Sipper (2014)	C#	Aplic.	✓	✓		✓	SD, SA, TS, EA, PSO

Fonte: organizado pelo autor

No quadro apresentado a coluna "Linguagem" especifica a linguagem de desenvolvimento usada no *framework*. A coluna "Tipo" diz respeito a classificação do *framework* se é uma biblioteca de classes ("Biblio.") ou se é uma aplicação completa ("Aplic."). A coluna "Interface gráfica" define se o *framework* possui ou não uma interface gráfica com o usuário. A coluna "Execução em lotes" define se o *framework* é capaz de executar os algoritmos em um processo de lotes automaticamente. A coluna "Análise estatística" define se o *framework* possui algum recurso ou

funcionalidade para realização de alguma análise estatística com os resultados encontrados. A coluna "Execuções paralelas" define se o *framework* é capaz de executar a otimização em ambientes paralelos ou distribuídos. A coluna "Técnicas de otimização" lista quais técnicas são tratadas pelo *framework*, identificadas por: LS (*Local Search*), SD (*Steepest Descent/Fill Climbing*), SA (*Simulated Annealing*), TS (*Tabu Search*), GRASP, VNS (*Variable Neighborhood Search*), EA (*Evolutionary Algorithms*), PSO (*Particle Swarn Optimization*), AIS (*Artificial Immune Systems*), ACO (*Ant Colony Optimization*), SS (*Scatter Search*), MO (*Multiobjective Metaheuristics*), DC (*Divide and Conquer*), BnB (*Branch and Bound*), DP (*Dynamic Programming*), GA (*Genetic Algorithmics*), MA (*Memetic Algorithmics*), MS (*Multiple Start*) e HH (*Hyper Heuristic*).

Dos vinte e cinco (25) *frameworks* levantados, somente oito (8) disponibilizam recursos ou componentes para realização de uma análise estatística capaz de fornecer informações mais detalhadas sobre os resultados encontrados em uma pesquisa, que são: EOlib, EasyLocal++, FOM, oMetah, OAT, JCLEC, MOEAT e JMetal.

O trabalho apresentado por Parejo et al. (2012) realizou um estudo comparativo de estruturas de otimização com meta-heurística. Neste trabalho são feitas análises mais profunda de um conjunto amplo de características, de dez (10) *frameworks* (ECJ, ParadisEO, EvA2, FOM, Jclec, OAT, Opt4j, EasyLocal, HeuristicLab e MALLBA), incluindo a disponibilidade de recursos ou componentes de análise estatística e representação de dados. Analisando os resultados apresentados pode-se perceber que dos dez *frameworks* analisados somente quatro incluem recursos ou componentes de análise estatística. Os *frameworks* FOM, JCLEC e OAT permitem realizar os testes estatísticos paramétricos teste-T (T-student) e teste ANOVA, e os testes não paramétricos teste de Wilcoxon e teste de Mann-Withney. O *framework* OAT faz ainda o teste Kolmogorov-Smirnov que verifica se os dados analisados seguem uma distribuição estatística normal. O *framework* EasyLocal++ permite realizar somente o teste-T e o teste de Mann-Withney.

Outros *frameworks* que foram possíveis perceber a disponibilidade recursos para análise estatística foram EOLib, oMetah, MOEAT e jMetal. O *framework* EOLib permite a realização de múltiplas análises da estatística descritiva, mas não trata de teste mais detalhados da estatística inferencial. O *framework* oMetah também propõe a realização de um conjunto de análises estatísticas, mas o projeto esta inativo desde

2006 e não foi possível identificar quais testes são possíveis de ser realizados. O *framework* MOEAT faz análise da fronteira de Pareto para problemas com dois objetivos. O *framework* JMetal realiza análises em pares pelo teste de Wilcoxon e analisa a característica de normalidade dos dados analisados pelo teste Kolmogorov-Smirnov.

2.1.4 Consideração sobre as abordagens estruturadas para desenvolvimento de heurísticas

Frameworks são ferramentas úteis que podem acelerar o desenvolvimento de projetos que solucionam problemas baseados em otimização, reduzindo o tempo e os custos de desenvolvimento. Eles também podem ser aplicados por usuários não especializados, bem como estender o escopo de aplicações para inúmeras técnicas diferentes de otimização baseada em heurísticas. Muitos dos *frameworks* disponíveis oferecem capacidades e recursos semelhantes, o que significa a existência de certa duplicidade de esforços. Fica claro que existe uma lacuna visível em relação ao suporte a recursos e componentes para realização de análises estatísticas e representação dos resultados. Mesmo sabendo da importância que estas análises trazem para a qualidade do estudo realizado na pesquisa, a maioria dos *frameworks* analisados não possui tais recursos.

2.2 Problema de programação de tarefas (*scheduling problem*)

Segundo Slack et al. (2009) e Arenales et al. (2007) o problema de programação de tarefas (*scheduling*) envolve um conjunto de produtos que devem ser completados através de um conjunto de operações que devem ser executadas. Estas operações necessitam de máquinas e recursos materiais e devem ser programadas seguindo alguma sequência lógica. A programação de tarefas envolve a seleção de uma sequência de operações (ou rotas de processo) que resultará na realização do produto. Envolve também, a determinação dos recursos necessários para executar cada operação da rota, bem como a determinação das datas em que cada operação da rota começará e terminará. Os primeiros modelos de programação de tarefas utilizavam em sua maioria o *makespan* (instante de término da última tarefa) como critério de decisão, mas novos critérios passaram a ser explorados, tendo em vista a

crescente importância para o mercado de fatores ligados a determinação e ao cumprimento das datas de entrega.

Segundo Baker e Trietsch (2009) os problemas de sequenciamento ou programação de tarefas em sistemas de produção são, tradicionalmente, classificados em função do fluxo das operações nas máquinas. Os ambientes de manufatura podem ser classificados em quatro classes principais: de máquina única, máquina paralela, *flowshop* e *jobshop*. Ainda segundo a literatura (ver, por exemplo, Nagano et al. (2004)), existem algumas variações desta classificação que podem ser encontradas nos trabalhos publicados, conforme segue:

- ***Openshop***: não há uma sequência para o processamento das tarefas nas máquinas;
- ***Flowshop* Permutacional**: em cada máquina a sequência das tarefas é a mesma;
- ***Jobshop* com Máquinas Múltiplas**: máquinas paralelas em cada estágio de produção;
- ***Flowshop* com Máquinas Múltiplas**: máquinas paralelas em cada estágio de produção.

A Figura 2.1 ilustra a relação entre as classes dos problemas de programação de operações em máquinas, conforme a classificação dada:

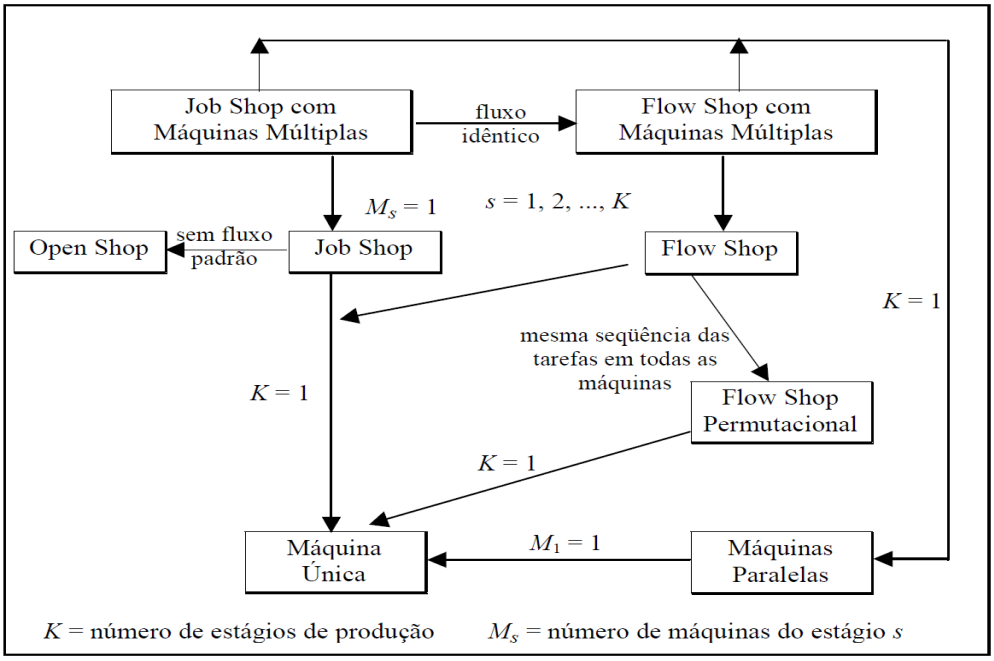


Figura 2.1- Relação entre as classes de problemas de sequenciamento de tarefas
Fonte: (NAGANO et al., 2004)

2.2.1 Ambiente de máquina única

O ambiente de máquina única (*single machine*) consiste em um problema de sequenciamento puro, ou seja, a ordenação das tarefas (*jobs*) já determina completamente a programação da produção (*schedule*). De fato, um problema de máquina única consiste na existência de n tarefas que devem ser processadas por uma determinada máquina, com o objetivo de determinar uma sequência que minimize determinado indicador de desempenho, como por exemplo, *makespan*.

Mesmo que atualmente o número de empresas que trabalham sob esse ambiente seja pequeno, ele ainda possui importância para sistemas reais devido à existência de gargalos na produção, ou seja, equipamentos que limitam a produtividade de uma empresa, que podem ser tratados como um ambiente de máquina única (BAKER; TRIETSCH, 2009). Isto pode ser comprovado analisando trabalhos que têm o foco na resolução deste problema de máquina única, tais como, Wu et al. (2014) que tratam de máquina única com deterioração linear truncada e tempo de finalização, com o foco na minimização do *makespan*, e Wu et al. (2013) que também trata do problema de máquina única para o problema de *scheduling* com dois agentes, com o foco na minimização do tempo total de programação.

2.2.2 Ambiente de máquinas paralelas

Já no caso do ambiente de máquinas paralelas, o problema de *scheduling* é expandido para incluir tanto o sequenciamento das tarefas quanto alocação dessas tarefas nas diversas máquinas (BAKER; TRIETSCH, 2009). Formalmente, um problema de máquinas paralelas consiste em n tarefas (*jobs*) que devem ser processadas em uma das m máquinas, com o objetivo de determinar uma sequência para cada uma das máquinas de forma a minimizar determinado indicador de desempenho. Cada tarefa deve ser processada por apenas uma única máquina, e qualquer máquina pode ser responsável por qualquer tarefa.

Segundo Allahverdi e Ng (2008), um problema de programação paralela com máquinas idênticas envolve o agendamento de tarefas em múltiplas máquinas idênticas em um sistema de produção, em que um trabalho (*job*) pode ser processado por qualquer uma das máquinas livres e cada trabalho finalizado irá liberar a máquina utilizada. Se essas máquinas não são idênticas, então temos um problema de

programação de máquinas não idênticas em paralelo, podendo ser uniformes ou independentes. Um problema de programação de máquina paralela uniforme assume que diferentes máquinas processam o mesmo trabalho com velocidades diferentes. Em outras palavras, os tempos de processamento de um trabalho nas máquinas, podem ser relacionados entre si. Pelo contrário, um problema de programação de máquinas paralelas não relacionadas considera que os tempos de processamento de um trabalho em máquinas diferentes são diferentes e não relacionados, ou não podem ser relacionados por fatores de velocidade.

Vários trabalhos tratam de máquinas paralelas, tais como Liao et al. (2014) que trata deste problema, juntamente com o sequenciamento de caminhões em um processo de transferência de carga (*cross-doking*) com múltiplas plataformas e objetivo de minimização do *makespan*. Já Arnaout et al. (2009) trata do mesmo problema só que com máquinas não relacionadas e tempos de *setup* dependentes e também com minimização do *makespan*.

2.2.3 Ambiente de *flowshop*

Enquanto que nos dois ambientes tratados anteriormente, uma única máquina é capaz de processar completamente uma tarefa, isso não ocorre no ambiente de *flowshop*. Nesse tipo de ambiente, as tarefas são divididas em operações, e o número de operações define por quantas máquinas a tarefa deve passar até que esteja completa. Por exemplo, uma tarefa com três operações, passará por três máquinas. Outra importante característica do *flowshop* é a existência de um fluxo unidirecional, ou seja, uma máquina k será responsável pelo processamento de uma operação k de qualquer tarefa. Esse fato permite que as máquinas sejam organizadas em linha. Um exemplo de um ambiente *flowshop*, é a linha de montagem presente na indústria automobilística (BAKER; TRIETSCH, 2009).

Mesmo que este ambiente trate de um problema de sequenciamento puro, ele apresenta diferenças do ambiente de máquina única, pois como a tarefa deve passar por mais de uma máquina, então há períodos de ociosidade das máquinas. Por exemplo, enquanto a primeira tarefa está sendo processada na primeira máquina as outras permanecem paradas. Sendo assim, uma tarefa com tempo de processamento menor ou maior fará diferença em alguns indicadores de desempenho, o que pode não acontecer no ambiente de máquina única (CHEN et al., 2013).

Quando se efetua um levantamento bibliográfico pode-se facilmente perceber que muitos dos trabalhos que utilizam heurísticas possuem o foco na solução de problemas de *flowshop*. Como por exemplo, Hecker et al. (2014) que trata do problema de *scheduling* com *flowshop* permutacional sem espera, onde em cada máquina a sequência das tarefas é a mesma, avaliando o *makespam* e tempo total ocioso das máquinas. Também Hecker et al. (2013) trata de *scheduling* com *flowshop* em uma linha de produção de uma padaria buscando otimização do *makespan* e otimização do tempo total de finalização. Já Chen et al. (2013) trata de uma nova perspectiva para solução de problemas de *scheduling* em *flowshop* permutacional apresentando um novo esquema de representação de cromossomos com o intuito de melhorar a performance de dois algoritmos existentes de AG e ACO, com o objetivo principal de minimizar o *makespan*.

2.2.4 Ambiente de *jobshop*

O último ambiente a ser descrito é o *jobshop*, que assim como o *flowshop*, possui a característica de que uma tarefa (*job*) deve ser processada por diversas máquinas. No entanto, há uma diferença fundamental: nele, o fluxo não é unidirecional, ou seja, a máquina k pode ser responsável pelo processamento da operação 2 na tarefa i e pelo processamento da operação 10 na tarefa j. Isso implica até mesmo em complexos problemas de definição da organização física dessas máquinas como pode ser visto em Drira et al. (2007). Assim, um problema de *jobshop* consiste em definir uma sequência para cada máquina, respeitando a condição de que a operação n deve ser precedida da operação n-1 para qualquer tarefa, de forma a minimizar/maximizar determinado indicador de desempenho.

A maioria das pesquisas nesta área apenas considera o tempo de finalização e a data de término da janela de tempo como indicador de desempenho. O estudo apresentado por Kuo e Cheng (2013), trata estes dois fatores simultaneamente. Além disso, o tempo de atraso é sempre tratado como o objetivo nos problemas de *jobshop*. No entanto, o tempo de finalização, que está mais próximo da realidade do cliente, pode resultar em menores custos de armazenamento.

Outros exemplos de resolução de problemas de *jobshop* podem ser vistos em trabalhos como o de Ahmadizar e Rabanimotlagh (2013) que trata do problema de *scheduling* definido como *groupshop*, formado por um *jobshop* e um *openshop*, com

o objetivo relacionado à minimização da data de entrega e tempo de processamento. O objetivo principal é minimizar o máximo custo de finalização. Utiliza uma função de limite inferior como modelo de simulação de evento discreto para especificação dos melhores resultados e um ACO é desenvolvido para comparação. Outro trabalho é o de Korytkowski et al. (2013) que trata do problema de *scheduling*, para determinar a alocação sub-ótima de regras de despachando multi-atributo dinâmicas, para maximizar o desempenho do sistema *jobshop*, sendo analisadas as medidas: tempo de fluxo médio, tempo de fluxo máximo, atraso médio e atraso máximo.

2.2.5 Outros ambientes de manufatura

Mesmo com essas quatro classes principais de ambientes de manufatura, algumas pesquisas tratam de desdobramentos mais complexos, por meio da restrições e outras características. Por exemplo, Liang et al. (2013) inclui no ambiente de máquina única a característica de que as tarefas são recebidas ao longo do tempo, ao invés de estarem todas disponíveis em t=0. Outro exemplo é o modelo proposto por Tavares Neto e Godinho Filho (2011) que inclui a terceirização (*outsorcing*) das operações em um ambiente *flowshop*, com o objetivo de minimizar uma soma ponderada entre *makespan* e custo gerado pela terceirização.

2.2.6 Indicadores de desempenho mais comuns

Quando se trabalha com problemas de *scheduling,* é necessário definir qual ou quais serão as funções objetivos que se espera como indicador de desempenho. Arenales et al. (2007) e Baker e Trietsch (2009) listam alguns indicadores comuns na literatura:

I. *Makespan* - duração total da programação das tarefas ou máxima data de finalização de todas as tarefas. Pode ser definido também como sendo somatório das datas de término das tarefas ($\sum C_i$) ou tempo de fluxo ($\sum F_i$), onde C_i é data de término da tarefa *i* e F_i é o tempo de fluxo da tarefa *i*;
II. *Lateness* - atraso total das tarefas, sendo $L_i = C_i - D_i$; onde $\infty < L_i < \infty$; considerando Ci como data de finalização e Di como data de entrega;
III. *Tardiness* - atraso máximo ($T_i = \text{Max} \{L_i, 0\}$)
IV. *Earliness* – adiantamento máximo ($E_i = \text{Max} \{-L_i, 0\}$)

Alguns autores utilizam como indicadores de desempenho outras variáveis diferentes do tempo. No trabalho apresentado por Huang (2010), o mesmo utiliza uma mescla entre variáveis que são tempo de fluxo de trabalho, tempo de inatividade da máquina e tempo total de transporte, ponderados por fatores constantes. Já a pesquisa de Luo et al. (2013) trabalha com a minimização do *makespan* em conjunto com uma variável monetária que é o custo da energia elétrica. Outros indicadores podem ser vistos ainda em Keskinturk et al. (2012) que utilizam o que chama de desequilíbrio relativo total (*total relative imbalance*) e Liao et al. (2011) que usam o tempo de *setup* como indicador. Além desses indicadores isolados, um pesquisador pode interessar-se em formular um modelo que se preocupa com mais de um indicador de desempenho, como por exemplo, o modelo proposto por Yagmahan e Yenisey (2010) que visa minimizar tanto o *makespan* quanto o tempo total de fluxo das tarefas, usando uma soma ponderada entre ambos os indicadores.

De acordo com a literatura analisada nessa pesquisa, a soma ponderada é uma forma muito comum de tratar mais de um indicador. No entanto, outras técnicas podem ser utilizadas, como Liang et al. (2013) que decidiu tratar o problema de minimizar tanto o *makespan* quanto o *tardiness* por meio da fronteira de Pareto (*Pareto front*) que consiste em um conjunto de soluções em que nenhuma delas é melhor que as outras em todos os indicadores tratados, nesse caso, *makespan* e *tardiness*. Isso é chamado conjunto de soluções não dominadas (*non-dominated solutions*) e a obtenção dele facilita o processo de tomada de decisão.

A obtenção da fronteira de Pareto se dá da seguinte forma: ao obter uma nova solução, ela é comparada com as soluções já pertencentes à fronteira de Pareto podendo ocorrer três situações (LIANG et al., 2013):

- A nova solução é dominada por qualquer outra solução pertencente ao conjunto, o que implica em seu descarte;
- A nova solução não é dominada por nenhuma solução pertencente ao conjunto e não domina nenhuma dessas soluções, o que implica na admissão dessa solução ao conjunto;
- A nova solução não é dominada por nenhuma solução pertencente ao conjunto e domina uma ou mais soluções, o que implica na admissão dessa solução ao conjunto e exclusão das soluções dominadas.

Estes problemas de sistema de produção podem ser otimizados utilizando várias técnicas matemáticas e computacionais diferentes, muitas vezes no campo da

otimização combinatória, que tem como objetivo principal maximizar ou minimizar a função objetivo definida sobre certo domínio finito de soluções viáveis. Como visto anteriormente, para um grande conjunto desses problemas, a obtenção da solução ótima não pode ser obtida com o uso de algoritmos exatos, principalmente pela característica não polinomial dos problemas. Devido a isso, tais algoritmos são apropriados somente para problemas de tamanho reduzido ou características especiais. Segundo Barr et al. (1995), é comum utilizar algoritmos baseados em heurísticas e meta-heurísticas para encontrar uma solução aceitável (não necessariamente a solução ótima) para estes problemas. Isso porque uma heurística tem como objetivo permitir uma exploração de uma região reduzida do espaço de soluções possíveis, gerando assim uma solução em um tempo razoavelmente aceitável. Estas soluções aproximadas tornam-se viáveis em função do tamanho do problema que se deseja resolver, ou seja, para problemas de grande porte talvez a única solução seja a utilização de heurísticas e meta-heurísticas.

2.3 Utilização de heurísticas em engenharia da produção

Dentre as diferentes abordagens para a solução de problemas de engenharia de produção, existem inúmeras heurísticas e meta-heurísticas que foram desenvolvidas na procura de melhores soluções. Segundo Blum e Roli (2003) as meta-heurísticas são estratégias de alto nível que guiam a implementação de heurísticas mais específicas, a fim de encontrar soluções viáveis. No desenvolvimento de uma heurística, esta geralmente é dependente do problema específico e exige a necessidade de adaptações para utilização em outros problemas. Ainda Blum e Roli (2003) apresentam algumas abordagens de meta-heurísticas que podem ser utilizada:

- SA - *Simulated Annealing* (otimização por recozimento simulado);
- TABU *Search* (busca TABU);
- GRASP - *Greedy Randomized Adaptive Search Procedure* (procedimento de busca adaptativo randômico ganancioso);
- GA - *Genetic Algorithmics* (algoritmos genéticos);
- ACO – *Ant Colony Optimization* (otimização por colônia de formigas);
- *Iterated greedy* - Algoritmo ganancioso iterativo.

Existem inúmeras outras heurísticas que podem ser utilizadas tais como BCO (*Bees Colony Optimization* - Otimização por colônia de abelhas), VNS (*Variable*

Neighborhood Search - busca por vizinhança variável), PSO (*Particle Swarm Optimization* - otimização por enxame de partículas), dentre outros.

Para entender a importância do uso de heurísticas para solucionar problemas relacionados com sistemas de produção, foi feito um levantamento dos trabalhos publicados. O levantamento relaciona os problemas de sequenciamento de tarefas, tamanho do lote, roteamento de veículos e localização de facilidades com as principais meta-heurísticas propostas por Blum e Roli (2003).

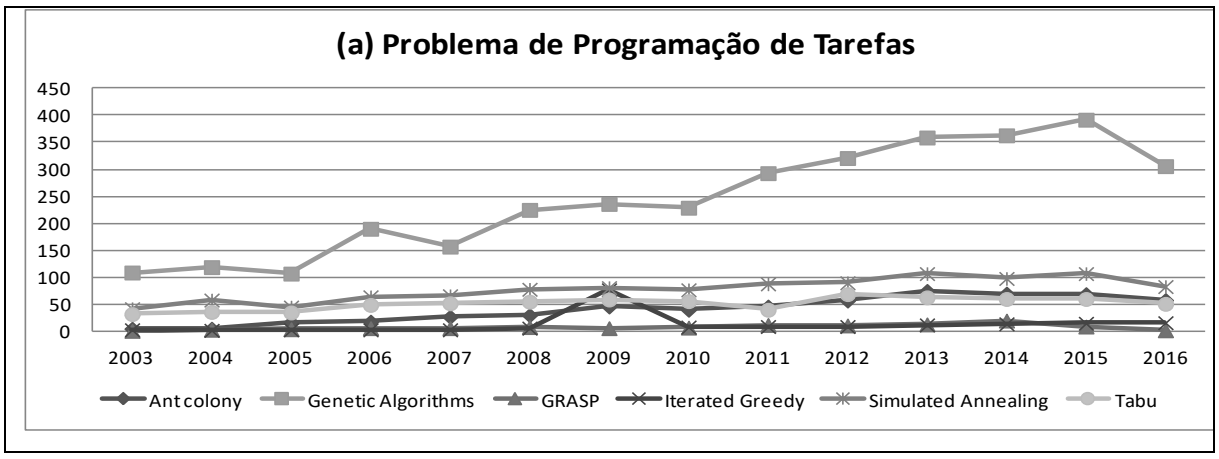
2.3.1 Metodologia de pesquisa do levantamento

O levantamento foi realizado em outubro de 2016, utilizando o portal de busca de publicações Engineering Village[6], que tem como base o COMPENDEX (*COMPuterized ENgineering inDEX*). Parâmetros da pesquisa:

- Na primeira linha de busca foi colocado o nome do problema de sistema de produção limitando somente ao título do trabalho (*Title*);
- Na segunda linha de busca foi colocado o nome da heurística limitado ao objetivo, título e resumo (*Subject/Title/Abstract*);
- Somente artigos de jornais (*Journal article*);
- Somente artigos na língua inglesa;
- Publicados entre 2003 até 2016.

2.3.2 Análise dos resultados

A Figura 2.2 (a) mostra o uso das heurísticas para solucionar o problema de sequenciamento de tarefas (*scheduling*).



[6] https://www.engineeringvillage.com/

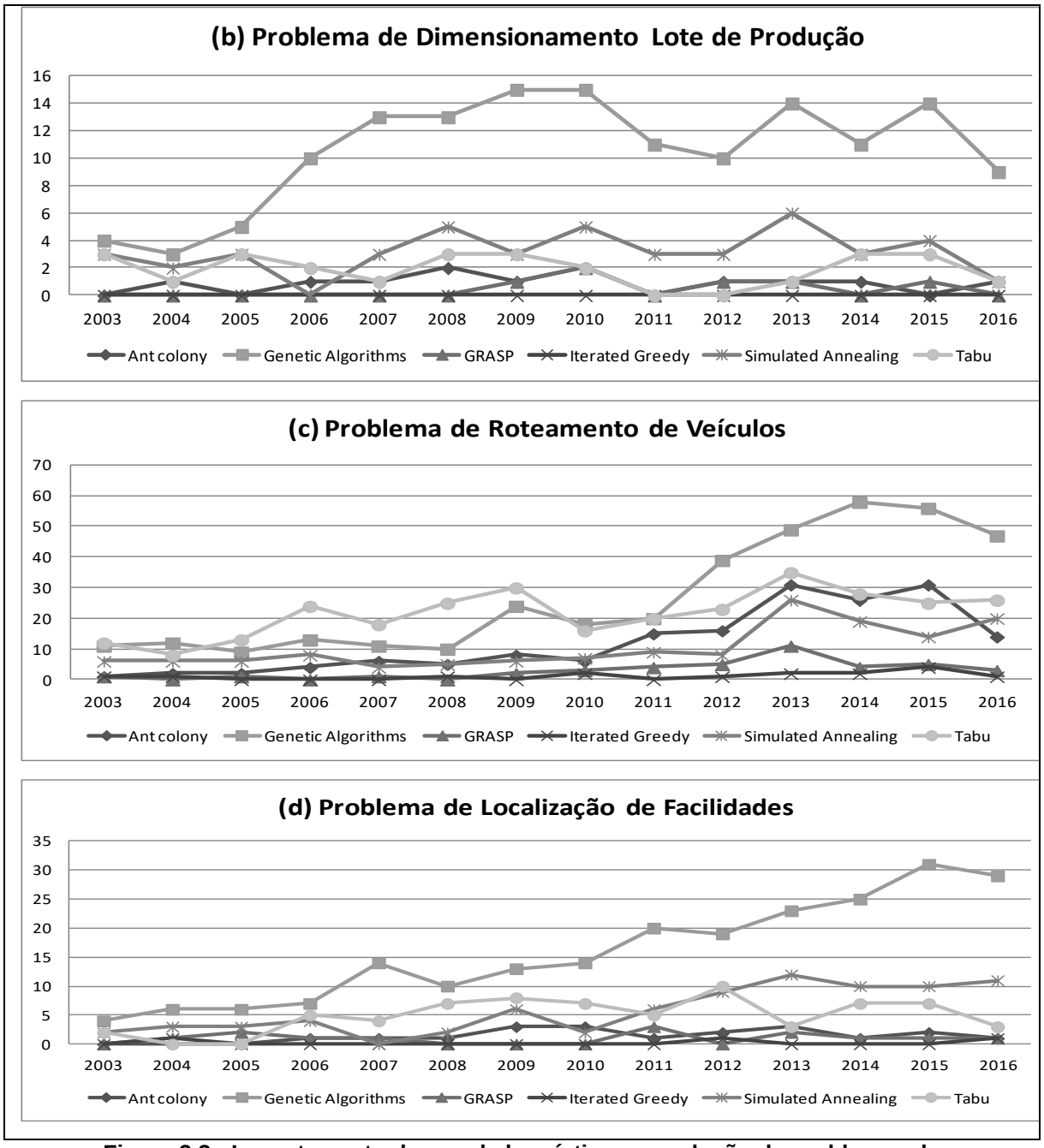


Figura 2.2 - Levantamento de uso de heurísticas na solução de problemas de Engenharia de Produção
Fonte: elaborado pelo autor

Percebe-se um crescimento a partir de 2002, do uso de GA para trabalhar com este problema, mas deve-se levar em conta que nem todos os trabalhos que aparecem na pesquisa são relacionados somente com sistemas de produção, pois o termo utilizado para a busca "*scheduling*" também se aplica a outras áreas como, por exemplo, gerenciamento de projetos. Neste contexto, os problemas de sequenciamento de tarefas, constata-se que a maioria dos trabalhos estão relacionados com *jobshop*, em segundo lugar com *flowshop*, e empatados em terceiro, máquinas simples e máquinas paralelas. Este gráfico mostra ainda que *Simulated*

Annealing e TABU são bastante utilizadas também na solução de programação de tarefas. O grande destaque fica em relação ao crescimento do ACO que em 2003 foram publicados somente três (3) trabalhos e em 2016 já foram vinte e nove (29) publicações, um crescimento bastante expressivo.

A Figura 2.2 (b) mostra o uso das heurísticas para solucionar o problema de tamanho de lote de produção. Para este tipo de problema a utilização de heurísticas não é convencional, o que pode ser observado pela quantidade de trabalhos publicados nesta área. Outro ponto percebido é o crescimento do uso de GA na solução destes problemas a partir de 2004. A Figura 2.2 (c) apresenta o uso das heurísticas para solucionar o problema de roteamento de veículos. Observa-se que de 2003 a 2009 a heurística mais utilizada era TABU e a partir de 2010 o GA acabou sendo a heurística mais utilizada. Outro destaque importante foi o crescimento do uso de ACO na busca por melhores soluções. A Figura 2.2 (d) mostra o uso das heurísticas para solucionar o problema de localização de facilidades. Apesar de não ser muito significativo percebe-se um crescimento do uso de heurísticas com destaque novamente para AG.

Fica claro ao analisar estes gráficos que a utilização de heurísticas para solucionar problemas de sistemas de produção tem crescido bastante nos últimos anos, e mais especificamente para problemas de programação de tarefas (*scheduling*) percebe-se este aumento expressivo a partir de 2003. O uso de soluções baseadas em GA são predominantes, soluções que usam *Simulated Annealing* e TABU ainda são bastante utilizadas por muitos pesquisadores e o grande destaque é o crescimento do uso da ACO principalmente para problemas de programação de tarefas de roteamento de veículos.

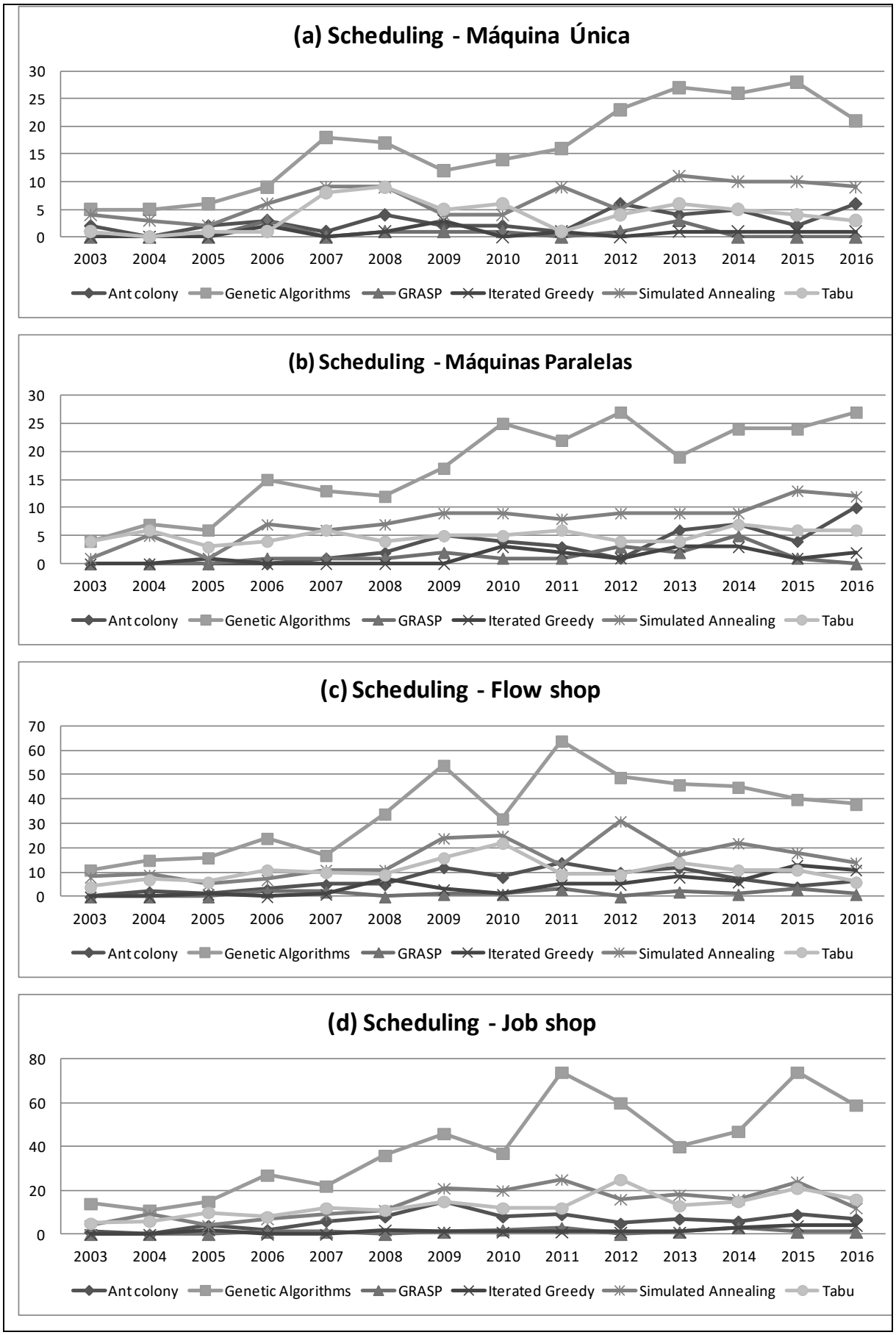


Figura 2.3 - Uso de heurísticas na solução de problemas de *scheduling*
Fonte: elaborado pelo autor

A Figura 2.3 apresenta o resultado da busca por publicações selecionadas de acordo com o sistema de produção. Percebe-se um aumento do uso de heurísticas a partir de 2005, dando os destaques para a utilização de algoritmos genéticos em todos os tipos de problemas e um destaque para o crescimento de ACO na solução de problemas de *flowshop* e *jobshop*.

2.4 Projeto de experimentos computacionais com heurísticas

Um experimento pode ser definido como sendo um conjunto de testes executados sob condições controladas para atingir um propósito ou objetivo específico. Estes testes podem demonstrar uma verdade conhecida, validar uma determinada hipótese sobre os dados ou ainda examinar o desempenho de algo realmente novo, trazendo a tona o conhecimento de um processo particular de forma a medir os efeitos de um ou mais fatores sobre este processo. Quando se fala em experimentos computacionais com a utilização de heurísticas, o que se procura é a resolução de uma série de instâncias, de um determinado problema, usando uma implementação específica da heurística. Neste contexto, algumas decisões que precisam ser tomadas incluem os seguintes fatores (BARR et al., 1995): seleção do problema e das instâncias; implementação do algoritmo; especificação do ambiente computacional; definição das medidas de desempenho; configuração do algoritmo em si; como os resultados serão apresentados. As escolhas definidas pelos pesquisadores dos fatores listados impactam substancialmente nos resultados e na significância do experimento realizado.

No trabalho de Bertrand e Fransoo (2002), os autores apresentam uma metodologia de pesquisa para métodos quantitativos e mais especificamente apresentam algumas etapas para serem aplicadas em pesquisas quantitativas axiomáticas com o uso de simulação, podendo utilizar heurísticas na solução de problemas de engenharia de produção. Já Mason et al. (2003) traz a definição de regras da estatística na experimentação apresentando três fases: fase de planejamento de projeto (com critérios de definição do que deve ser medida, a variação a que está sujeita e a influência dos fatores); fase de projeto experimental (controle das fontes de variação conhecidas que permite estimar o tamanho da variação desconhecida e uma investigação de modelos adequados) e fase de análise

estatística dos dados (faz inferências sobre os fatores de projeto, guia para projetos posteriores, sugere modelos mais adequados).

Kothari (2004) propõe um processo que consiste de uma série de ações ou passos necessários no desenvolvimento de pesquisas que incluem a análise estatística de dados. Os passos sugeridos são:

1. Defina o problema de pesquisa
2. Faça uma revisão da literatura:
    a. Revise conceitos e teorias
    b. Revise resultados de pesquisas anteriores
3. Formule as hipóteses a serem trabalhadas
4. Projete a pesquisa
5. Faça a execução e coleta dos dados
6. Analise os dados
7. Interprete e apresente os resultados

Neste mesmo sentido, Montgomery (2001) propõe sete passos a serem seguidos para implementação de experimentos computacionais. Barr et al. (1995) apresenta uma simplificação destes passos:

1. Defina os objetivos do experimento;
2. Escolha as medidas de desempenho e fatores que serão explorados;
3. Projete e execute o experimento;
4. Analise os dados e tire conclusões;
5. Apresente os resultados do experimento.

2.4.1 Definição dos objetivos do experimento

Um experimento computacional deve ter um propósito ou objetivo claramente especificado. O propósito ajuda a identificar os tipos de resultados a serem alcançados, as hipóteses a serem testadas, os testes a serem executados, os fatores a serem explorados, as medidas a serem usadas e os dados necessários para dar suporte aos itens acima. Apesar de não haver padrões estabelecidos para pesquisa de algoritmos, em geral se aceita que um método heurístico produza contribuições relevantes se possuir as seguintes características (BARR et al., 1995): rápido (produz soluções de alta qualidade mais rapidamente que outras abordagens); preciso

(identifica soluções de maior qualidade que outras abordagens); robusto (menos sensível às diferenças nas características de problemas, dados e parâmetros de ajuste do que outras abordagens); simples (fácil de implementar); alto impacto (resolve um problema novo ou importante mais rapidamente e mais precisamente que outras abordagens); generalizável (possui uma faixa larga de aplicações); inovador (é novo e criativo por si).

Ainda de acordo com Barr et al. (1995), os experimentos computacionais com algoritmos podem ser classificados em:

- **Experimentos comparativos**: comparação do desempenho de diferentes algoritmos para a mesma classe de problemas. Esta comparação pode ser realizada com técnicas competitivas existentes ou mesmo heurísticas populares já publicadas. É preferível que os resultados tanto da heurística proposta quanto dos métodos comparativos sejam processados em um mesmo ambiente computacional para uma melhor comparação. Caso isso seja impraticável, deve-se citar o ambiente computacional em que tais resultados foram obtidos. Se outros métodos não existem, um método mais geral (por exemplo, baseado em programação linear ou inteira) ou uma abordagem gulosa simples devem servir como referência para comparação;
- **Experimentos descritivos**: caracterização ou descrição do desempenho isolado de um algoritmo. O objetivo é ganhar compreensão do comportamento da metodologia e dos fatores que influenciam tal comportamento. Para obter informação dos efeitos dos fatores específicos em um algoritmo, devem-se testar códigos que são idênticos exceto por um único fator e fazer uso de técnicas estatísticas, tal como análise de variância, para testar o efeito do fator nas medidas de desempenho.

2.4.2 Escolha das medidas de desempenho

Um experimento computacional refere-se a um conjunto de variáveis dependentes, que são as medidas de desempenho ou os resultados, sendo afetadas diretamente por um conjunto de variáveis independentes, que são o problema, o algoritmo e os fatores do ambiente de teste. Neste sentido a maioria das pesquisas tem a pretensão de responder as seguintes perguntas (BARR et al., 1995):

- Qual a qualidade da melhor solução encontrada?

- Quanto tempo é requerido para se obter a melhor solução?
- Quão rapidamente o algoritmo encontra boas soluções?
- Quão robusto é o método?
- Quão “distante” a melhor solução está daquelas facilmente obtidas?
- Qual o *trade-off* entre factibilidade e qualidade das soluções?

Todas estas medidas podem ser compiladas em quatro áreas principais de análise (BARR et al., 1995):

a) **Qualidade das soluções**: Ao testar um algoritmo heurístico, a consideração adicional de quão perto a solução heurística encontrada está da solução de referência é de grande importância. Esta distância da solução de referencia é comumente chamada de "*GAP*". Se a solução ótima é conhecida: o desvio percentual relativo (RPD - *Relative Percentage Derivation*) da solução ótima é reportado (RUIZ; STÜTZLE, 2007; RUIZ et al., 2009; HANOUN et al., 2011).

$$RPD = 100 * \frac{\big(A(I) - OPT(I)\big)}{OPT(I)}$$

Onde: A(i) é a solução heurística encontrada, OPT(I) é a solução de referência ou solução ótima conhecida.

Se a solução ótima não é conhecida o desvio percentual de um limitante inferior (ou superior) é reportado. Este limitante inferior pode ser conseguido através de um método de menor caminho ou uma heurística mais simples.

$$DP = 100 * \frac{\big(A(I) - LB(I)\big)}{LB(I)}$$

Onde: A(i) é a solução heurística encontrada, LB(I) é a solução de referência de limitante inferior disponível.

b) **Esforço computacional**: A velocidade de obtenção das soluções é um fator chave para análise do desempenho. É importante reportar:
   - Tempo para obtenção da melhor solução (deve incluir todo o pré-processamento envolvido e operações de suporte);
   - Tempo total de execução;
   - Tempo por fase: importante para determinação de “gargalos” do algoritmo.

c) **Robustez**: Geralmente, a robustez é baseada na habilidade de uma heurística se portar bem em uma faixa larga de instâncias teste, e é obtida através de medidas de variabilidade. As estratégias e os parâmetros da heurística devem permanecer constantes para o conjunto de instâncias ou devem ser automaticamente definidos usando atributos de instâncias individuais.

2.4.3 Fatores a serem explorados

Para Barr, et al (1995), existem ainda três categorias de fatores que afetam o desempenho de um algoritmo em experimentos computacionais. Para cada categoria, é necessário selecionar o que será estudado (tamanho do problema, método heurístico), o que deverá ser fixado em algum nível (classe do problema, regra de parada, ambiente computacional) e o que deve ser ignorado esperando que não influenciará os resultados. As categorias são:

a) **Fatores do problema -** Uma variedade de características do problema tais como dimensões, a estrutura e a distribuição paramétrica podem afetar os resultados observados e são importantes para avaliar a robustez do código produzido. É importante ainda realizar execuções com instâncias grandes ("*stress test*"). Muitos fatores não aparecem em instâncias pequenas e tais execuções podem não gerar previsões precisas para instâncias maiores e mais realistas. Outros pontos que devem ser considerados são as características das instâncias de teste.

b) **Fatores do algoritmo -** Incluem a seleção das heurísticas e dos códigos em experimentos comparativos. Envolvem escolhas específicas do algoritmo sendo testado tais como estratégias e valores de parâmetros (construção da solução inicial, parâmetros da busca heurística com regras bem definidas). Heurísticas robustas que funcionam bem com um único conjunto de valores de parâmetros são geralmente preferíveis a heurísticas que requerem valores individuais para cada instância, a não ser que a heurística seja auto-ajustável, com a correção dos parâmetros do algoritmo em tempo de execução, tais como apresentados pelos trabalhos de Birattari et al. (2009) e Lopez-Ibanez et al. (2011).

c) **Fatores do ambiente -** De modo ideal em uma comparação entre algoritmos, os mesmos deveriam ser programados pelo mesmo especialista, os testes deveriam ser aplicados com um mesmo conjunto de instâncias de teste e deveriam ser executados em um mesmo ambiente computacional. Neste contexto, os resultados em termos de tempo e qualidade das soluções são diretamente comparáveis. Caso contrário é necessário determinar o efeito dos fatores discrepantes nos resultados.

2.4.4 Projeto e execução dos experimentos

O projeto de experimento é o processo de planejamento de um determinado experimento para garantir que os dados apropriados sejam coletados. Um bom projeto de experimento deve ser imparcial, de forma a alcançar os objetivos definidos, demonstrando claramente o desempenho do processo testado e quais as razões para este desempenho, gerando conclusões aceitáveis e podendo ser reproduzível. Todas estas características são particularmente úteis para o ensaio de métodos heurísticos (BARR et al., 1995).

a) **Escolhendo um projeto de experimento** - A melhor forma de alcançar tais resultados é através do uso de conhecidos métodos de projeto estatísticos experimentais (DOE - *Design of Experiments*) (MONTGOMERY, 2001; MASON et al., 2003). Esta é uma abordagem que garante que os dados coletados possam ser analisados por métodos estatísticos para chegar a conclusões válidas e objetivas. Quando os resultados de um determinado experimento variam de acordo com as condições de ensaio (problemas resolvidos, ambiente computacional, definições de parâmetros) a utilização de uma metodologia estatística é a única abordagem objetiva para análise e obtenção de conclusões. Neste sentido, o projeto de experimentos e a análise estatística dos dados estão inter-relacionados. Segundo Barr et al. (1995), vários autores fazem uso de projeto de experimentos para avaliação de otimização em algoritmos heurísticos.

b) **Selecionando as instâncias de teste -** O conjunto de problemas de teste podem ser obtidos a partir de definições práticas ou criados pelo pesquisador com *softwares* de geração de problema. Problemas do mundo

real refletem o propósito final dos métodos heurísticos e são importantes para avaliar a eficácia de uma determinada abordagem. Instâncias representativas são importantes, quando encontradas em um domínio do problema. Se a situação permite, os problemas devem ser coletados com fatores controlados por algum projeto experimental estatístico. Em alguns casos, é importante desenvolver problemas de teste especiais para testar características de desempenho e padrões de resposta particulares do método. Neste sentido, os problemas podem ser criados com um *software* de geração automática de instâncias, que fornece ao usuário controle sobre características do problema, permitindo a criação de instâncias com uma configuração específica de fatores do projeto do experimento. Geradores podem fornecer a solução ótima ou então um valor factível que possa ser utilizado na análise. Se as instâncias são criadas pelo usuário, o processo de geração deve ser claramente descrito e as novas instâncias geradas devem ser armazenadas (ou o processo de geração reproduzível deve ser documentado) de forma que possibilite a utilização por outros pesquisadores. Em geral, espera-se que, quanto maior o número de instâncias avaliadas, mais informativo será o estudo. As instâncias mais interessantes são aquelas que estressam ou testam os limites do *software* ou causam sua falha.

c) **Executando o experimento** - Neste passo do processo são realizadas as execuções da solução proposta e a coleta de dados. O pesquisador deve assegurar que o projeto experimental está sendo seguido e que os detalhes pertinentes do processo, estão documentados. A execução dos testes deve seguir a ordem aleatória que foi planejada e o ambiente de computação deve ser uniforme. Esses elementos ajudam a reduzir a variabilidade e a influência de fatores desconhecidos.

2.4.5 Análise de dados

Segundo Barr et al. (1995), ferramentas de análise de dados: pacotes estatísticos gerais (R, Statistica, SPSS, Minitab, Matlab, etc.), *software*s para projeto de experimentos - DOE (extensão DOE para R, Minitab, SAS, etc.), *softwares* para

visualização de dados: por meio de gráficos que permitem a visualização de todos os dados e consequentemente uma maior compreensão do processo pode ser atingida e métodos manuais e reconhecimento de padrões.

Métodos estatísticos devem ser empregados sempre que possível para indicar a força das relações entre fatores diferentes e as medidas de desempenho, podendo indicar a confiabilidade e validade dos resultados. Uma vez que os dados tenham sido analisados, os resultados são interpretados com uma série de conclusões e inferências deduzidas das evidências coletadas. É papel do pesquisador avaliar a significância estatística destas conclusões, verificando suas implicações em relação as conclusões finais da pesquisa realizada. A experimentação tende a ser um processo iterativo, onde cada etapa não somente expande o conhecimento da solução algorítmica proposta, mas também traz novas questões, e algumas vezes, a necessidade de novas medidas de desempenho.

2.4.6 Apresentação dos resultados do experimento

Após a coleta e análise dos dados o pesquisador deve fazer inferências sobre os resultados e consequentemente obter conclusões sobre a pesquisa realizada. Este processo deve ser feito com cautela para que não existam conclusões enviesadas, não representando a realidade. Neste sentido, é importante que o pesquisador construa um relatório da pesquisa com todas as análises estatísticas e as inferências realizadas, de forma que fique disponível para futuras consultas. Kothari (2004) apresenta um modelo para interpretação e escrita de relatórios estatístico da pesquisa. Além disso, Barr et al (1995) propõem uma série de orientações para reportar os experimentos computacionais:

- Reprodutibilidade deve ser essencial, ou seja, faça o relatório de forma a permitir a reprodução substancial dos resultados;
- Especifique todos os detalhes dos fatores que podem influenciar nos resultados, tais como heurística, código, parâmetros, números pseudo-aletórios, dados de entrada, estruturas de dados não triviais e ambiente computacional;
- Seja preciso quanto a medidas de tempo;
- Mostre como alguns parâmetros do algoritmo são definidos;
- Use técnicas estatísticas de projeto de experimentos;

- Compare a heurística com outros métodos;
- Reduza a variabilidade dos resultados;
- Produza um relatório que abranja os resultados obtidos.

2.5 Conjunto de instâncias para problemas de *scheduling*

Quando se refere ao tamanho da amostra (número de instâncias de problemas usadas nas execuções dos algoritmos), há dois aspectos principais a serem determinado. Em primeiro lugar, a amostra mínima considerada aceitável para as necessidades de cada teste. Não existe acordo estabelecido sobre esta especificação, mas existem trabalhos que relacionam o tamanho da amostra com o poder do teste estatístico, tal como apresentado por Morse (1999). Neste trabalho Morse (1999) especifica um *software* que calcula o tamanho da amostra a ser analisada dependo de parâmetros estatísticos definidos e do teste estatístico realizado. A priori, quanto maior for à amostra, maior a potência do teste. Em segundo lugar, tem-se que avaliar como os resultados variam quando forem provenientes de um número maior de amostras. Em todos os testes estatísticos usados para comparar duas ou mais amostras, o aumento do tamanho da amostra melhora a potência do teste (GARCÍA et al., 2009).

Segundo Bartz-Beielstein et al. (2010), a maioria dos modelos científicos requerem o uso de dados para a avaliação e validação. As três fontes principais que os pesquisadores utilizam para obter estes dados são o mundo real, bancos de dados de biblioteca e procedimentos de geração aleatória. Além destas três fontes Rardin e Uzsoy (2001), propuseram ainda uma quarta forma que são variações aleatórias do conjunto de dados do mundo real. Estes modelos são apresentados a seguir.

2.5.1 Conjunto de dados do mundo real

A escolha natural como fonte de instâncias de problemas é a utilização de dados do mundo real. Como vantagens pode-se destacar a relevância, o que permite estimar a eventual utilidade prática dos resultados de uma experiência, o aumento da credibilidade do experimento e que estes tipos de dados são geralmente imparciais. No entanto, pode haver sérias dificuldades com o uso de dados do mundo real, principalmente porque podem não estar disponíveis, ou então não existirem

quantidades suficientes para a realização dos experimentos. Outro problema, é que a maioria dos dados reais para uma dada aplicação são gerados pelo mesmo processo físico, limitando assim o tamanho e a variabilidade das amostras observadas (BARTZ-BEIELSTEIN et al., 2010).

2.5.2 Biblioteca de instâncias

Dependendo do problema tratado na pesquisa e devido à impossibilidade de coletar dados reais ou mesmo pela baixa quantidade de instâncias reais, o pesquisador pode optar pela utilização de bibliotecas de instâncias de problemas que já estão publicadas e disponíveis para serem utilizadas. Entre as vantagens destas bibliotecas de instâncias são que algumas propriedades do problema, tal como as soluções ótimas e resultados de outras pesquisas sobre o conjunto de instâncias, são conhecidos. Uma preocupação é que os conjuntos de dados das bibliotecas podem ser influenciados pela inclusão de problemas para os quais uma determinada experiência fornece bons resultados. Um problema que pode ser levantado é que a comparação de procedimentos baseados em um mesmo conjunto de instâncias de uma biblioteca leva à seleção de procedimentos que funcionam de forma eficiente para esse conjunto. Estes mesmos procedimentos podem não fornecer bons resultados para os tipos mais comuns de instâncias (BARTZ-BEIELSTEIN et al., 2010).

Existem inúmeras bibliotecas disponíveis na internet, que possuem instâncias para várias classes de problemas diferentes, incluindo para os problemas de *scheduling*. O Quadro 2.3 apresenta as principais bibliotecas.

Quadro 2.3 - *Benchmark* de instâncias para problemas de *scheduling*

Referência	Descrição
(BEASLEY, 1990) [7]	OR-Library - apresenta uma grande variedade com mais cinquenta conjuntos de problemas para diferentes áreas da pesquisa operacional, inclusive para *scheduling*, incluindo *flowshop*, *jobshop*, *openshop* dentre outros, cada qual com um grande conjunto de instâncias de teste.
(TAILLARD, 1993) [8]	Propõe 260 problemas de *scheduling* gerados aleatoriamente com o objetivo de minimização do *makespan*, cujo tamanho das instâncias é maior do que exemplos já publicados e

[7] http://www.brunel.ac.uk/~mastjjb/jeb/info.html
[8] http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html

	correspondendo às dimensões reais de problemas industriais. Os tipos de problemas de *scheduling* propostos são: *flow shop* permutacional, *job shop* e *open shop*.
(KOLISCH; SPRECHER, 1997) [9]	PSPLIB - Esta biblioteca contém diferentes conjuntos de problemas para vários tipos de problemas de *scheduling com* recursos limitados, assim como as melhores soluções e soluções baseadas em heurísticas.
(DEMIRKOL et al., 1998) [10]	Conjuntos extensos de problemas de teste gerados aleatoriamente para os problemas de *scheduling* com minimização de *makespan* Cmax e máximo atraso Lmax em problemas *flow shop* e *job shop*. São 600 problemas que incluem três tipos diferentes de rotas, quatro configurações de data de vencimento diferentes e uma variedade de tamanhos de problema.
(VALLADA et al., 2015) [11]	Apresenta um novo *benchmark* de instâncias difíceis (grandes) para problemas de *flow-shop scheduling* com minimização de *makespan*. O *benchmark* consiste de 240 instâncias grandes e 240 instâncias pequenas com até 800 tarefas e 60 máquinas. O objetivo é gerar um ponto de referência que satisfaça as características desejadas de qualquer *benchmark*, sendo abrangente, passível de análise estatística e discriminante quando vários algoritmos venham a ser comparados.

Fonte: elaborado pelo autor

A grande vantagem destas bibliotecas de instâncias é que novos problemas e novos resultados encontrados podem ser incluídos junto ao conjunto de instancias existentes. Inclusive é uma prática comum dos pesquisadores no intuito de disponibilizar seus resultados para outros pesquisadores na realização de pesquisas futuras.

2.5.3 Instâncias geradas randomicamente

Quando não for possível ou desejável a utilização de dados do mundo real ou a utilização de uma biblioteca de instâncias, a alternativa é a geração aleatória das instâncias. Estes tipos de dados também são conhecidos como dados sintéticos. Geralmente, os dados sintéticos são gerados por um programa de computador usando um gerador de números pseudo-aleatórios. As vezes, os dados sintéticos podem ser gerados por variações aleatórias dos dados do mundo real ou de uma biblioteca. No entanto, as vantagens substanciais de dados sintéticos incluem a velocidade e a facilidade de produção, bem como a quantidade, variedade, e relevância dos dados que podem ser obtidos (BARTZ-BEIELSTEIN et al., 2010).

[9] http://www.om-db.wi.tum.de/psplib/main.html
[10] http://optimizizer.com/DMU.php
[11] http://www.webofinstances.com/

Se as instâncias são geradas pelo pesquisador, então o processo de geração deve ser claramente descrito, para ser utilizado por outros pesquisadores. O maior problema da geração randômica de instâncias é a obtenção do valor ótimo ou um valor factível para realizar comparações. Como problemas de *scheduling* são NP-difícil, dependendo do tamanho do problema não se consegue obter o valor ótimo com um procedimento exato. Neste sentido o pesquisador deve utilizar de recursos tais como:

- Limitar o tempo de execução de um procedimento exato com o intuito de obter um resultado factível de comparação;
- Utilizar uma abordagem gulosa simples, ou seja, desenvolver uma solução capaz de fornecer uma solução factível para comparação;
- Se não for possível a definição de uma solução factível para comparação, o pesquisador pode adotar a estratégia de definir o melhor resultado encontrado entre todas as soluções algorítmicas, para cada instância, como sendo o valor base de comparação.

2.6 Consideração a respeito da fundamentação teórica

Este capítulo apresentou a fundamentação teórica da referida tese de doutorado. Com o levantamento sobre abordagens estruturadas para desenvolvimento de heurísticas, percebe-se que uma lacuna em trabalho com o foco na definição de um *framework* para validação estatística de resultados de pesquisa para problemas de *scheduling*, que é o objetivo principal deste trabalho. A especificação dos problemas de engenharia de produção com os ambientes de manufatura para problemas de programação de tarefas (*scheduling*), permitiu verificar o crescimento da utilização de procedimentos heurísticos e meta-heurísticos para solucionar estes tipos de problemas. Neste sentido, o entendimento dos conceitos sobre projeto de experimentos computacionais com heurísticas, se fez necessário como base de conhecimento para a especificação e desenvolvimento do *framework*. Por fim é apresenta a conceituação sobre o conjunto de instâncias para problemas de *scheduling*, juntamente com um levantamento sobre as principais bibliotecas de instâncias disponíveis.

Capítulo 3

ANÁLISE ESTATÍSTICA DE EXPERIMENTOS COMPUTACIONAIS

Este capítulo apresenta um levantamento sobre os conceitos e métodos estatísticos para a análise e representação de resultados de experimentos computacionais heurísticos de problemas de engenharia de produção com foco em *scheduling*.

3.1 Introdução

Segundo Flexer (1996) a realização de experimentos computacionais é uma tarefa difícil principalmente por causa dos métodos utilizados e os resultados que devem ser analisados, que são muitos complexos para um tratamento formal. Para um experimento comparativo de análise de resultados não se tem instrumentos formais para decidir qual dos métodos é o melhor. Existe uma vasta literatura estatística, teorias sobre aprendizado computacional e muito mais que podem ajudar na tomada de decisão. Mas a decisão final é sempre baseada em uma análise empírica dos resultados. Estas mesmas características são percebidas em sistemas por otimização através de heurísticas e meta-heurísticas.

A estrutura básica para análise destes tipos de experimentos é semelhante. A questão a ser analisada é se existem efeitos da variação das variáveis independentes (características dos parâmetros utilizados e características do conjunto de dados em questão) em relação as variáveis dependentes (medições de desempenho). As observações (em termos de variáveis dependentes) realizadas durante os experimentos são somente uma parte do todo. Existem pelo menos três argumentos que motivam esta premissa. Primeiro que os dados disponíveis para a experimentação são comumente definidos como uma parte de conjunto maior de

dados. Segundo, com as amostras de dados são definidos subconjuntos para teste e parametrização. Terceiro que existe certa influência sobre os valores randômicos utilizado nas soluções (FLEXER, 1996).

Siegel (1957) aponta para validade e representatividade dos eventos que estão sendo observados e indica a utilização de procedimentos de estatística inferencial para fornecer apoio a obtenção de conclusões. Somente através de testes estatísticos pode-se garantir que os efeitos observados nas variáveis dependentes sejam influenciados pelas variações das variáveis independentes e não por mero acaso.

No mesmo sentido, García et al. (2009) citam que ao analisar as publicações de artigos em revistas especializadas, percebe-se que a maioria dos artigos fazem comparações dos resultados com base em valores médios do conjunto de execuções. Poucos trabalhos usam métodos estatísticos para comparar os resultados, embora esta necessidade esteja crescendo e recentemente está sendo sugerida como uma premissa por muitos revisores de trabalhos a serem publicados. Quando se encontram trabalhos que fazem uso de técnicas estatísticas, geralmente, são baseados na média e variância. Percebe-se que procedimentos com testes estatísticos paramétricos e não paramétricos estão sendo considerados para serem utilizados na análise dos resultados.

Os métodos estatísticos na pesquisa funcionam como uma ferramenta no planejamento da pesquisa, analisando os dados e apresentando resultados que permitem deduzir conclusões mais assertivas. A maioria dos estudos apresenta um grande volume de dados brutos que devem ser adequadamente reduzidos, de modo que, os mesmos, possam ser lidos facilmente e possam ser utilizados para uma análise posterior. Segundo Kothari (2004) a classificação e tabulação dos dados alcançam este objetivo até certo ponto, mas é necessário dar um passo adiante no sentido de desenvolver certos índices ou medidas para resumir os dados. Somente após isso é possível adotar o processo de generalização a partir de pequenos grupos (ou seja, amostras) da população.

A estatística é dividida em duas grandes áreas: a estatística descritiva e estatística inferencial. A estatística descritiva diz respeito ao desenvolvimento de certos índices a partir dos dados brutos, ao passo que a estatística inferencial se preocupa com o processo de generalização. A estatística inferencial é também conhecida como estatística de amostragens e está preocupada principalmente com dois grandes tipos de problemas: a estimativa de parâmetros populacionais, e os

testes de hipóteses (FLEXER, 1996; KOTHARI, 2004)(KOTHARI, 2004). Mais especificamente a análise estatística de dados faz uso da análise descritiva unidimensional (medidas de tendência central, medidas de dispersão, medidas de assimetria e curtoses) e análise estatística inferencial (testes paramétricos, teste não paramétricos e análise *post-hoc*) que serão detalhados mais adiante.

3.2 Princípios básicos dos testes estatísticos

Para que haja uma compreensão e entendimento das análises descritiva e inferencial citadas anteriormente é importante que sejam apresentados os conceitos básicos que envolvem tais análises estatísticas. Ressaltando que estas análises serão utilizadas no *framework*, proposta da presente pesquisa.

3.2.1 Testes de hipóteses

Um procedimento de teste de hipóteses permite avaliar a validade (ou não) de uma afirmação sobre uma determinada característica da população, usando para isto, os dados de amostra retirada desta população. Essa característica é representada por uma variável aleatória X, cujo comportamento probabilístico é expresso por sua função de densidade de probabilidade f. Admite-se que f depende de um parâmetro θ, cujo valor é desconhecido. Deseja-se, através do experimento envolvendo a coleta de dados, decidir entre duas afirmações mutuamente excludentes a respeito do valor de θ: a hipótese nula (H_0) e a hipótese alternativa (H_1) (PINHEIRO et al., 2009). Na estatística inferencial, o teste de hipóteses pode ser aplicado para inferências sobre uma ou mais populações de amostras. A hipótese nula é a especificação do não efeito ou não diferença e a hipótese alternativa representa a presença de um efeito ou de uma diferença entre as amostras. Quando se aplica um procedimento estatístico para rejeitar a hipótese nula, um nível de significância α é usado para definir em que valor a hipótese nula deva ser rejeitada (DERRAC et al., 2011).

Seja X_1, X_2, ..., X_n uma amostra aleatória proveniente da população, que foram levantados experimentalmente. Deseja-se estabelecer um critério de decisão que leve a aceitar H_0 ou rejeitar H_0 (em favor de H_1). Usualmente, a escolha desse critério passa por eleger uma função t de X_1, X_2, ..., X_n chamada de estatística do teste e dividir o conjunto de valores possíveis de t em duas partes: região de aceitação A e região de

rejeição R, também denominado região crítica. Assim, conforme o valor de t, calculado a partir dos dados, pertença a A ou pertença a R, aceita-se H_0 ou rejeita-se H_0 (PINHEIRO et al., 2009).

3.2.2 Intervalo de confiança e níveis de confiança

Ao combinar a dimensão da amostra com a sua variabilidade, gera-se um intervalo de confiança para a média da população. Se for assumido que a amostra é aleatoriamente selecionada de uma população que segue uma distribuição normal, então, com 95% de confiança, pode-se dizer que o intervalo de confiança inclui a média da população (LARSON; FARBER, 2004). A maioria dos intervalos de confiança é calculada para 95%, mas também podem ser utilizados outros níveis de confiança conforme for a necessidade. Se for intenção obter um nível de confiança maior, ou seja, se almeja ter mais confiança de que um intervalo contenha o parâmetro real, então o intervalo tem de ser maior. Assim sendo, os intervalos de confiança de 99% são mais amplos que os de 95% e os de 90% são mais estreitos que os de 95% (LARSON; FARBER, 2004).

Um p-valor de confiança 0,01 seria estatisticamente mais significante do que um p-valor de 0,005, mas isto não é correto. A partir do momento em que é definido o nível de confiança, todos os resultados são ‘estatisticamente significantes’ ou ‘estatisticamente não significantes’. No entanto, há autores que fazem uma distinção do grau de significância e utilizam símbolos para este propósito, mas estas denominações e notações não são padronizadas. O Quadro 3.1 apresenta um exemplo de níveis de significância que também são utilizados nos resultados de muitos *softwar*es estatísticos disponíveis.

Quadro 3.1 - Níveis de significância

P-valor	**Descrição**	**Notação**
>0.05	Não significante	ns
0.01 até 0.05	Significante	*
0.001 até 0.01	Muito significante	**
< 0.001	Extremamente significante	***

Fonte: (LARSON; FARBER, 2004)

3.2.3 Valor de p (p-valor) ou nível crítico

O nível crítico ou p-valor é o menor valor de α (nível de significância do teste) para o qual, ao usar o procedimento de teste e trabalhar com os valores observados

X_1, X_2, ..., X_n ainda seria rejeitado H_0. O nível crítico costuma ser representado pelo símbolo $\hat{\alpha}$ (PINHEIRO et al., 2009). Embora esta definição seja simples, o conceito de nível crítico é, ao mesmo tempo um dos mais importantes, por ser muito usado em publicações científicas para reportar resultados de um teste de hipótese e um dos mais sutis, uma vez que normalmente é necessário algum tempo de amadurecimento e de familiarização com este assunto para que seja alcançado uma compreensão maior do seu significado.

3.2.4 Tipos de erros

Ao aplicarmos um teste de hipótese, é necessária a tomada de decisão com base em informações apenas parcial (a que está nos dados amostrais) sobre a realidade. Portanto há dois tipos possíveis de erro de decisão que poderão vir a ser cometidos, e que seria conveniente evitar (PINHEIRO et al., 2009):

- α = P (erro do tipo I) = P (rejeitar H_0 | H_0 verdadeiro)
- β = P (erro do tipo II) = P (aceitar H_0 | H_0 falso)

Dois conceitos importantes também são:

- Tamanho do teste ou nível de significância = α = P (erro do tipo I)
- Potência do teste = 1- β = 1 - P (erro do tipo II) = P (rejeitar H_0 | H_1 verdadeiro)

O Quadro 3.2 apresenta os tipos de erros para o teste de hipótese.

Quadro 3.2 - Tipos de erros de decisão

Decisão	Situação Real	
	H_0 é verdadeira	**H_0 é falsa**
H_0 é aceita	Decisão correta	Erro tipo II
H_0 é rejeitada	Erro tipo I	Decisão correta

Fonte: (PINHEIRO et al., 2009)

Os testes estatísticos procuram confirmar se uma determinada hipótese (H_0) pode ser rejeitada. Os testes podem ser bilaterais ou unilaterais (BUSSAB; MORETTIN, 2004):

- Nos testes bilaterais:
  - A hipótese nula (H_0) inclui sempre o sinal de igual
  - A hipótese alternativa (H_1) inclui sempre o sinal desigual
- Nos testes unilaterais:
  - A hipótese nula (H_0) inclui o sinal de maior ou igual
  - A hipótese alternativa (H_1) inclui o sinal oposto

O p-valor nos testes unilaterais é duas vezes o p-valor dos testes bilaterais. Os testes unilaterais são apropriados quando se pode afirmar com certeza (mesmo antes da coleta de dados) que não haverá diferença entre as médias, ou a diferença é no sentido do que se especifica. Caso contrário, deve-se usar testes bilaterais.

3.2.5 Testes estatísticos

Algumas premissas devem ser especificadas para que sejam analisados somente um subconjunto destes testes estatísticos. São elas:

- Todos os valores obtidos nas amostras que são analisados são ordinais;
- O presente trabalho trata de análises comparativas das diferenças entre diversas amostras, ou seja, entre duas ou mais amostras;
- Os dados das amostras são obrigatoriamente dependentes, ou seja, as soluções heurísticas comparadas são executadas com um mesmo conjunto de instâncias de problema;
- As amostras coletadas podem seguir uma distribuição normal ou não.

Os testes estatísticos podem ser divididos em teste paramétricos e não paramétricos. Testes paramétricos possuem uma variedade de características fortes (tais como normalidade dos dados) e são mais poderosos que os métodos não paramétricos. Se um teste paramétrico estiver sendo usado e a característica de normalidade não puder ser garantida, o que acontece é que as instâncias que estão sendo julgadas como não significantes poderiam ser significantes, mas não vice e versa. Se isto acontecer, deve-se partir para utilização de testes não paramétricos, comumente chamados de distribuição livre. No trabalho de Flexer (1996), são avaliados 61 trabalhos de redes neurais entre os anos de 1994 e 1995, onde somente 4,9% apresentam testes estatísticos mais elaborados e 27,7% apresentam valores de média, variância e intervalo de confiança. A grande maioria dos trabalhos realizam testes empíricos nos resultados. Isto comprova a necessidade de utilização de testes estatísticos na análise de resultados de experimentos de forma a tentar melhorar a qualidade dos trabalhos.

Como a premissa principal para a aplicação dos testes paramétricos é que a amostra siga uma distribuição normal, se faz necessário a verificação da normalidade das amostras analisadas. Existem vários procedimentos para realizar este teste de

normalidade, dentre eles pode-se citar o teste de Kolmogorov-Smirnov com correção de Lilliefors (LILLIEFORS, 1967) e o teste de Shapiro-Wilks (SHAPIRO; WILK, 1965).

Quando o teste de validação dos dados da pesquisa verifica que a amostra obtida é proveniente de uma população não normal, o desejo para justificar o uso de testes mais poderosos faz com que o pesquisador tente alterar a distribuição dos dados. Isto pode ser feito por meio da aplicação de métodos matemáticos sobre as dados originais, que irão "transformá-los" para que a suposição de normalidade torne-se sustentável. A questão que deve ser levantada com esta tentativa é se este processo de "normalização" da distribuição, alterando os valores numéricos dos dados podem causar uma distorção do efeito experimental investigado. Este é um questionamento que o pesquisador pode ou não ser capaz de responder. Se o processo de transformação dos dados tem o efeito de diminuir o efeito experimental, o pesquisador fica sujeito a uma situação paradoxal. Este tipo de ação pode fazer com que o teste paramétrico escolhido reduza a sensitividade das medidas e perca poder, ou seja, a hipótese nula H_0 poderá ser rejeitada quando na realidade não deveria ser (SIEGEL, 1957).

Quando a pesquisa envolve a comparação dos dados entre duas ou mais amostras e quando a verificação da normalidade dos dados e o teste de igualdade das variâncias (homocedasticidade) forem conferidos, é comum que seja escolhido um teste paramétrico na análise dos dados. Mas se as suposições do teste não forem satisfeitas, então é difícil, se não impossível, dizer qual será realmente o poder do teste paramétrico. Neste sentido a escolha mais adequada é a utilização de testes não paramétricos (SIEGEL, 1957).

Vários autores tratam especificamente da tabela de decisão para a aplicação dos testes estatísticos a partir de um conjunto de parâmetros específicos. A partir destes trabalhos e analisando os parâmetros quanto a quantidade de amostra analisadas, a questão de normalidade das amostras e a dependência das amostras são listados todos os testes estatísticos utilizados para uma análise. O Quadro 3.3 apresenta a tabela de decisão para os testes paramétricos e não paramétricos.

Quadro 3.3 - Tabela de decisão para testes estatísticos inferenciais

Número de amostras	Normalidade das amostras	Dependências das amostras	Teste estatístico	Referências
1 amostra	Normal	Uma amostra somente	Teste-t de amostra única	(MARUSTER; BACAREA, 2010)
	Não Normal	Uma amostra somente	Teste Wilcoxon de amostra única	(MARUSTER; BACAREA, 2010) (SHESKIN, 2003)

			Teste de Kolmogorov-Smirnov	(SIEGEL, 1957) (SHESKIN, 2003)
			Teste de execuções	(SIEGEL, 1957)
			Teste da mediana para amostra única	(SHESKIN, 2003)
2 amostras	Normal	Independente	Teste-t amostras independentes	(MARUSTER; BACAREA, 2010) (HOWELL, 2013) (DU-PREL et al., 2010)
		Dependente	Teste-t pareado	(MARUSTER; BACAREA, 2010) (HOWELL, 2013) (DU-PREL et al., 2010)
	Não Normal	Independente	Teste de Mann-Whitney	(MARUSTER; BACAREA, 2010) (SHESKIN, 2003) (HOWELL, 2013) (DU-PREL et al., 2010)
			Teste de Festinger	(SIEGEL, 1957)
			Teste de Kolmogorov-Smirnov	(SIEGEL, 1957) (SHESKIN, 2003)
			Teste das medianas	(SIEGEL, 1957) (SHESKIN, 2003)
			Teste de Moses de reações extremas	(SIEGEL, 1957)
			Teste de execuções de Wald-wolfowitz	(SIEGEL, 1957)
			Teste de White	(SIEGEL, 1957)
			Teste de Wilcoxon	(SIEGEL, 1957)
		Dependente	Teste de sinais	(SIEGEL, 1957) (SHESKIN, 2003)
			Teste de Wilcoxon pareado	(MARUSTER; BACAREA, 2010) (SIEGEL, 1957) (SHESKIN, 2003) (HOWELL, 2013) (DU-PREL et al., 2010)
K amostras	Normal	Independente	Teste ANOVA simples	(MARUSTER; BACAREA, 2010) (HOWELL, 2013) (DU-PREL et al., 2010)
		Dependente	Teste ANOVA com medições repetidas	(MARUSTER; BACAREA, 2010) (DU-PREL et al., 2010)
	Não Normal	Independente	Teste Kruskal-Wallis	(MARUSTER; BACAREA, 2010) (SIEGEL, 1957) (SHESKIN, 2003) (HOWELL, 2013) (DU-PREL et al., 2010)
			Extensões do teste das medianas	(SIEGEL, 1957) (SHESKIN, 2003)
			Teste de Jonckheere	(SIEGEL, 1957)
			Teste de execuções Mood	(SIEGEL, 1957)
			Teste de Mosteller slippage	(SIEGEL, 1957)
			Teste de extensão de Whitney	(SIEGEL, 1957)
		Dependente	Teste de Friedman	(MARUSTER; BACAREA, 2010) (SIEGEL, 1957) (SHESKIN, 2003) (HOWELL, 2013) (DU-PREL et al., 2010)
			Teste de Wilson	(SIEGEL, 1957)

Fonte: adaptado de pelo autor baseado nas referências listadas

Em uma análise comparativa entre mais que duas soluções algorítmicas, se o teste aplicado, seja paramétrico ou não paramétrico, for confirmada a hipótese alternativa que pelo menos uma das soluções testada é diferente das outras, é necessário que o pesquisador faça uma análise *post-hoc* com o intuito de identificar

qual dentre as soluções analisadas é a melhor. Uma análise *post-hoc* é um procedimento muito importante, sem a qual o teste de hipóteses multivariado poderia apresentar restrições quanto ao uso, tornando a possibilidade de descobrir falsos positivos inaceitavelmente elevados.

Baseado nas informações do Quadro 3.3 e realizando uma simplificação na escolha dos testes estatísticos, principalmente baseados nos trabalhos apresentados por Maruster e Bacarea (2010), Siegel (1957), Sheskin (2003), Howell (2013) e Du Prel et al. (2010), é apresentado o roteiro para escolha de testes inferenciais estatísticos. A Figura 3.1 apresenta este roteiro. Vale ressaltar que os testes em destaque (cor mais escura) são os objetivos do presente trabalho, que são detalhados mais adiante.

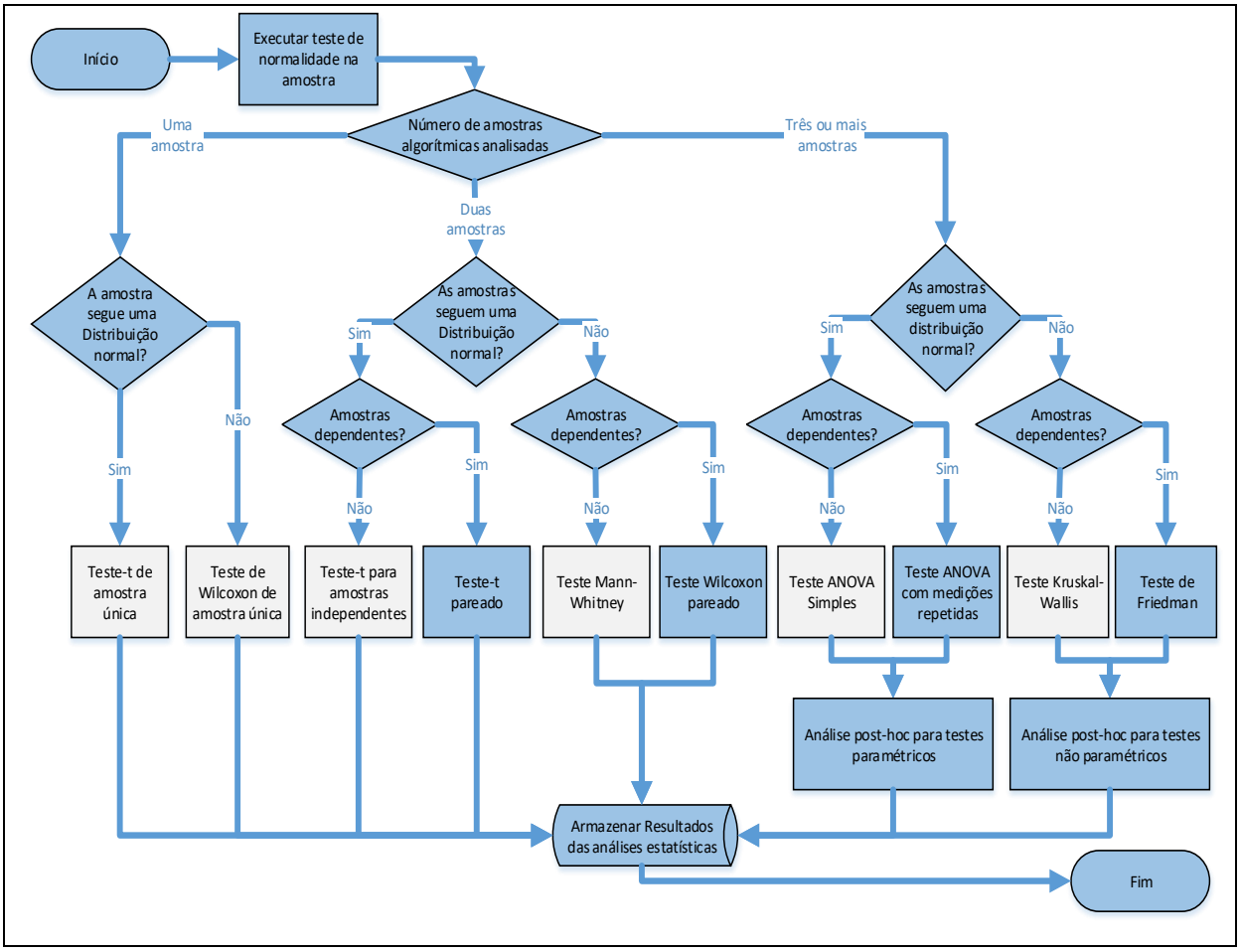


Figura 3.1 - Síntese das análises estatísticas de resultados
Fonte: elaborado pelo autor

3.2.6 Poder do teste

Quando vários testes estatísticos alternativos estão disponíveis para tratar os dados de uma pesquisa, é muito importante que o pesquisador faça a escolha correta

de qual teste utilizar. Como critério de escolha o pesquisador poderá usar o conceito de poder do teste. Segundo Siegel (1957) o poder do teste é definido como a probabilidade que o teste venha a rejeitar a hipótese nula quando de fato é falsa e deve ser rejeitada, ou seja: Poder do teste = 1 - probabilidade do erro do tipo II. Assim, a estatística do teste é considerada boa se tem uma pequena probabilidade de rejeitar a hipótese nula H_0, quando H_0 é verdadeiro, mas tem uma grande probabilidade de rejeitar H_0, quando H_0 for falso.

Contudo, existem considerações sobre o poder do teste que devem ser levadas em consideração na escolha do teste estatístico. Deve-se considerar a natureza da população da qual a amostra foi retirada e o tipo de medição que foi utilizado nas definições das variáveis operacionais da pesquisa. Estas questões também determinam qual teste estatístico é ideal para analisar um determinado conjunto de dados. A escolha entre os testes estatísticos que podem ser usados com uma determinada pesquisa deve ser baseada em três critérios (SIEGEL, 1957):

- O modelo estatístico do teste deve se adequar às condições da pesquisa;
- As exigências de medição do teste devem ser cumpridas pelas medidas utilizadas na pesquisa;
- De todos os testes com modelos estatísticos adequados e requisitos de medição apropriados, qual teste deve ser escolhido que tenha o maior poder-eficiência.

Pelo critério de generalidade, os testes não paramétricos são preferíveis em relação aos paramétrico. Pelo critério de poder do teste, no entanto, os testes paramétricos são superiores, precisamente por causa da força de seus pressupostos (SIEGEL, 1957).

3.3 Estatística descritiva dos dados

O objetivo principal da estatística descritiva é se preocupar com a organização, apresentação e sintetização de dados, tornando-os mais fáceis de serem entendidos descrevendo tendências, médias e variações (LARSON; FARBER, 2004). São utilizados valores de medidas descritivas, gráficos e tabelas para realizar as análises. Tanto Bruni (2010) como Bussab e Morettin (2004) apresentam os valores das medidas descritivas dos dados classificados em medidas de tendência central e medidas de dispersão, que busca identificar o afastamento absoluto ou relativo dos

dados. Quanto maior a dispersão, menor a informação contida na medida de posição central. Outra classificação diz respeito a medidas de ordenamento e forma dos dados.

As medidas de tendência central são aquelas que produzem valores em torno da distribuição dos dados analisados, e que visam definir em um único número o conjunto de dados (FREUND, 2009). As principais medidas de tendência central são são apresentadas no Quadro 3.4 abaixo.

Quadro 3.4 - Medidas de tendência central

Nome	Descrição
Média	É a divisão da soma dos valores da amostra pelo número de observações (média simples) ou os valores são ponderados pelas frequências com que ocorrem, somados e depois dividindo pelo total das frequências (média ponderada). Possui propriedades de unicidade e simplicidade sendo somente indicada para distribuições simétricas ou levemente assimétricas. Quando todos os dados são analisados diz-se tratar de média populacional (**μ**) e quando os dados de uma amostra são processados diz-se tratar de uma média amostral ($\overline{X}$).
Mediana	Valor que ocupa a posição central de um conjunto de valores ordenados, ou seja, a mediana (**Md**) divide a distribuição de valores em duas partes iguais: 50% acima e 50% abaixo do seu valor. Quando o conjunto possui quantidade par de valores, há dois valores centrais, neste caso, a mediana é o valor médio dos dois valores centrais do conjunto de dados ordenados. Possui as mesmas propriedades da média mas não é tão afetada pelos valores extremos, sendo uma medida mais robusta.
Moda	É o valor que ocorre com maior frequência não dependendo do nível de mensuração da variável, sendo aplicada tanto a fenômenos qualitativos quanto quantitativos. Se todos os valores forem diferentes não há moda (**Mo**), por outro lado, um conjunto pode ter mais do que uma moda: bimodal, trimodal ou multimodal. A moda possui algumas vantagens tais como: a modificação de um valor da amostra não necessariamente altera o valor da moda e não é influenciada por valores extremos.

Fonte: (BUSSAB; MORETTIN, 2004; LARSON; FARBER, 2004; BRUNI, 2010)

A dispersão do conjunto de dados é a variabilidade que os dados apresentam entre si. Se todos os valores forem iguais não existe dispersão e a dispersão é pequena quando os valores são próximos uns dos outros. Se os valores são muito diferentes entre si, a dispersão é grande, assim, as medidas de dispersão apresentam o grau de agregação dos dados (FREUND, 2009). As medidas descritivas mais comuns para quantificar a dispersão são apresentadas no Quadro 3.5 abaixo.

Quadro 3.5 - Medidas de dispersão

Nome	Descrição
Amplitude	É a diferença entre as entradas máximas e mínimas de um conjunto de observações. A utilidade da amplitude total como medida de dispersão é muito limitada, pois depende apenas dos valores extremos. A maior vantagem em usá-la é a simplicidade do seu cálculo.
Desvio Médio Absoluto	Uma vez que se deseja medir a dispersão ou grau de concentração dos valores em torno da média nada mais interessante do que analisar o comportamento dos desvios de cada valor em relação à média, ou seja, o desvio médio absoluto (**DMA**). Porém,

	para qualquer conjunto de dados, a soma de todos os desvios é igual a zero. Neste caso, considera-se o módulo de cada desvio evitando com isso que a somatória dos desvios seja igual a zero.
Variância	Embora o desvio médio seja uma medida melhor do que a amplitude, ainda não é uma medida ideal, principalmente porque não diferencia os pequenos dos grandes afastamentos em relação à média. Como forma de amenizar os problemas computacionais associados à extração dos módulos das diferenças para o cálculo do desvio médio absoluto, outra alternativa é elevar os afastamentos ao quadrado, eliminando assim o problema dos sinais como também potencializando os afastamentos, enfatizando os grandes desvios em relação às observações mais próximas da média. Seu resultado define a medida de variação, denominada de variância (**S^2 ou σ^2**). Esta estatística isolada tem difícil interpretação por apresentar unidade de medida igual ao quadrado da unidade da medida dos dados. A maior desvantagem diz respeito à impossibilidade de comparação entre a variância e a média.
Desvio Padrão	Devido à dificuldade de interpretação da variância, por ter sua unidade de medida ao quadrado, na prática usa-se o desvio padrão que é a raiz quadrada da variância. Embora apresente um cálculo razoavelmente extenso, o desvio padrão (**S ou σ**) tem uma praticidade e empregabilidade muito grande na estatística, sendo a medida de dispersão mais empregada.
Erro padrão da média	Diferentes amostras retiradas de uma mesma população podem apresentar médias diferentes. A variação existente entre este conjunto de médias é estimada através do erro padrão (**S_x**), que corresponde ao desvio padrão das médias.
Coeficiente de Variação	Uma questão a ser analisada é se um desvio-padrão é grande ou pequeno, questão esta relevante na avaliação da precisão do método. Um desvio padrão pode ser considerado grande ou pequeno dependendo da ordem de grandeza da variável. O coeficiente de variação (**CV**) é uma medida relativa de dispersão, utilizada para comparar, em termos relativos, o grau de concentração em torno da média.

Fonte: (MONTGOMERY; RUNGER, 2002; LARSON; FARBER, 2004; BRUNI, 2010)

As medidas de ordenamento fornecem uma ideia sobre a distribuição dos dados ordenados. Apresentam a vantagem de não serem afetadas pela forma da distribuição dos dados analisados ou por valores extremos. Um exemplo de medida de posição é a mediana que representa o ponto central de uma série ordenada de dados. Muitas vezes é preciso aumentar a informação da análise feita, surgindo assim à necessidade de outras medidas. Embora as medidas de posição e de variação possibilitem descrever estatisticamente um conjunto de dados, é necessário verificar como a distribuição dos dados se comporta de forma geral. Além das medidas de ordenamento, é necessária a identificação da forma da distribuição dos dados. O Quadro 3.6 apresenta as medidas de ordenamento e forma.

Quadro 3.6 - Medidas de ordenamento e forma

Nome	Descrição
Quartis, decis e percentis	Dividem a série ordenada em quatro, dez e cem partes, respectivamente, necessárias para aumentar qualidade da informação da análise realizada.
Assimetria (*skewness*)	É o grau de assimetria (**As**) de uma distribuição. É importante que a distribuição seja em forma de sino (simétrica), sendo que a metade da esquerda do seu histograma é aproximadamente a imagem-espelho da metade direita. As distribuições assimétricas apresentam caudas em uma das extremidades. Quando está à direita, é positivamente assimétrica, e se está à esquerda, é negativamente assimétrica.

Grau de concentração (kurtosis)	É o grau de achatamento de uma distribuição (**K**) em relação a uma distribuição padrão, denominada de curva normal. A curva normal, que é a base referencial, recebe o nome de mesocúrtica. Já, uma distribuição que apresentar uma curva de frequência mais estreita do que a normal é denominada de leptocúrtica e a que apresentar uma curva de frequência mais aberta recebe o nome de platicúrtica
Amplitude Interquartil	A amplitude interquartil (**IQR**) é a diferença entre o terceiro e o primeiro quartil.

Fonte: (LARSON; FARBER, 2004; BRUNI, 2010)

3.4 Análise gráfica dos dados

Os gráficos apresentam uma forma simples de transmissão de informação contidas em diferentes conjuntos de dados, permitindo compreender de maneira simples e eficiente aspectos e relações numéricas. A representação de uma série de dados através de gráficos permite, ao mesmo tempo, uma visão ampla e algumas visões particulares de um conjunto de dados analisados (BRUNI, 2010). O Quadro 3.7 apresenta os tipos de gráficos mais comuns para análise dos dados.

Quadro 3.7 - Tipos de gráficos para análise

Nome	Descrição
Histograma	É a representação gráfica de uma distribuição de frequência por meio de retângulos justapostos, contendo as classes de valores na abscissa e as frequências, absolutas ou relativas, nas ordenadas, centradas nos pontos médios. Os histogramas tem caráter preliminar sendo um importante indicador da distribuição de dados.
Gráfico de colunas ou barras	Consiste em construir retângulos, em que uma das dimensões é proporcional à magnitude a ser representada, sendo a outra arbitrária porém, igual para todas as colunas (ou barras).
Gráfico de linha	Exibe informações com uma série de pontos de dados chamados de marcadores ligados por uns segmentos de linha reta. Gráficos de linhas mostram como algumas alterações de dados específicos em intervalos de tempo são iguais, sendo, também muitas vezes, usado para visualizar uma tendência nos dados em intervalos de tempo de uma série temporal
Box Plot	Representa graficamente os dados da distribuição de uma variável quantitativa em função de seus parâmetros, que são: o menor valor (x_1), os quartis (Q_1, Q_2 - mediana e Q_3) e o maior valor (x_n). Este gráfico é importante para visualizar a posição, dispersão e assimetria da distribuição dos dados, além de fornecer informações importantes sobre o comportamento do conjunto de dados como simetria e variabilidade. Deve-se tratar as observações atípicas (*outliers*) porque podem alterar as médias e variabilidade dos grupos distorcendo as conclusões obtidas.
Diagrama de Dispersão	É uma representação de pontos de dados em um gráfico X-Y. O eixo y é utilizado para representar a variável dependente e o eixo x é para representar uma variável independente que pode ser controlada. Dependendo das variáveis consideradas, a relação entre elas pode ser fortemente linear, não linear ou mesmo inexistente.

Fonte: (LARSON; FARBER, 2004; BRUNI, 2010)

3.5 Testes de normalidade

A utilização de um teste paramétrico necessita da premissa principal que as amostras dos dados sigam uma distribuição normal. Um teste de normalidade aplicada sobre estas amostra pode indicar ou não a presença desta característica nos

dados observados. Existem vários testes de normalidade disponíveis dentre os quais são detalhados os seguintes:

1. **Teste de Shapiro-Wilks** (SHAPIRO; WILK, 1965) - Analisa os dados observados para calcular o nível de simetria e curtose (forma da curva), a fim de calcular a diferença em relação a uma distribuição gaussiana. Em seguida obtendo-se o p-valor a partir da soma dos quadrados dessas discrepâncias.
2. **Teste de Kolmogorov-Smirnov com correção de Lilliefors** (LILLIEFORS, 1967) - Primeiro são calculadas a média e a variância da amostra. Em seguida são calculadas as discrepâncias máximas entre a função de distribuição empírica e a função de distribuição cumulativa da distribuição normal com a média e a variância estimada. Finalmente, são avaliadas se a discrepância máxima é suficientemente grande para ser estatisticamente significativa, exigindo assim a rejeição da hipótese nula.

   Os dois testes acima possuem as seguintes formulações de hipótese:

$$\begin{cases} H_0: x_i \in N(\bar{x}, \sigma_x^2) \\ H_1: x_i \notin N(\bar{x}, \sigma_x^2) \end{cases}$$

A hipótese nula (H_0) verifica se a amostra analisada segue a distribuição normal e a hipótese alternativa o contrário. Se o p-valor calculado pelos testes for menor que o valor de significância, por exemplo 0,05 (95% de confiança), a hipótese nula é rejeitada, indicando que a amostra não segue uma distribuição normal.

3.6 Testes paramétricos

Segundo Sheskin (2003), a distinção entre testes paramétricos e não paramétricos leva em consideração o nível de medições representadas pelos dados que serão analisados. Desta forma, um teste paramétrico utiliza dados compostos por valores reais, isto não quer dizer que sempre será utilizado um teste paramétrico. Há outras premissas iniciais para uma utilização segura destes tipos de testes. O não cumprimento destas condições pode fazer com que o resultado da análise estatística perca credibilidade.

Os testes paramétricos (teste-t e teste-F e variações) impõem as seguintes premissas que devem ser consideradas (SIEGEL, 1957; SHESKIN, 2003; GARCÍA et al., 2009):

- Os eventos devem ser independentes significando que o que ocorre com uma não modifica a probabilidade da outra ocorrência;
- As observações da população devem seguir uma distribuição normal ou distribuição Gaussiana com um determinado valor de média μ e variância σ;
- As observações da população devem possuir a propriedade de heteroscedasticidade, que significa possuir as mesmas variâncias ou em casos especiais, deve ser conhecida a taxa de variação entre as variâncias;
- No caso de análise de variância, as médias destas populações normais devem ser combinações lineares dos efeitos devidos às colunas e/ou linhas, ou seja, os efeitos devem ser aditivos. Além disso, a natureza destes testes também impõe uma exigência de medição.

No presente trabalho, fica clara a independência dos eventos, dado que eles são execuções de algoritmos com diferentes características ou mesmo com sementes iniciais sendo geradas aleatoriamente. Quanto à normalidade da amostra ela deve ser verificada com mais detalhes, de acordo com os métodos apresentados anteriormente. Na maioria dos trabalhos da área utilizam-se funções estocásticas baseadas em números aleatórios, sendo que os resultados obtidos com as execuções dos algoritmos dificilmente vão seguir uma distribuição normal, e mais ainda, não terão variâncias iguais.

Derrac et al. (2011) comenta que os testes paramétricos tem sido comumente utilizados em pesquisas com análise de experimentos em inteligência computacional. Como exemplo ele cita a utilização do teste-t pareado para realizar a comparação entre as diferenças entre dois algoritmos, com o intuito de verificar se a diferença média no desempenho ao longo dos problemas é significativamente diferente de zero. Na comparação de múltiplos algoritmos, o método estatístico mais comum para testar as diferenças é o ANOVA com medidas repetidas. Vale ressaltar que se as condições para utilização dos testes paramétricos não forem atendidas, pode-se ter problemas com o poder do teste ou mais ainda com falsos positivos. Neste caso o melhor seria utilizar testes não paramétricos como visto anteriormente.

3.6.1 Teste-t pareado

O teste-t pareado é utilizado para determinar se as médias de dois grupos ou amostras são iguais ou dependentes, ou seja, os valores obtidos são provenientes de

um mesmo conjunto de instâncias e considera-se que esta comparação seja influenciada pelo efeito destas instâncias. As premissas para aplicação deste teste é que ambos os grupos sigam uma distribuição normal com variâncias iguais, mas podendo ser desconhecidas (SHESKIN, 2003).

Sejam (X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n), pares de variáveis aleatórias e suas diferenças $D_i = X_i - Y_i$, para todo i. Admite-se que D_1, D_2, ..., D_n sejam também variáveis aleatórias com distribuição normal, ou seja, $D_i \in N(\mu_D, \sigma_D^2)$. As hipóteses nula e alternativa da análise a serem testadas no teste-t são (PINHEIRO et al., 2009):

$$\begin{cases} H_0: \mu_D = 0 \\ H_1: \mu_D \neq 0 \end{cases}; \quad \begin{cases} H_0: \mu_D = 0 \\ H_1: \mu_D > 0 \end{cases}; \quad \begin{cases} H_0: \mu_D = 0 \\ H_1: \mu_D < 0 \end{cases}$$

A hipótese nula (H_0) verifica se as médias das diferenças entre as amostras são iguais, ou seja, igual à zero. Existem três possibilidades para a hipótese alternativa (H_1): se a média das diferenças é diferente de zero indicando que é um teste bilateral e os teste unilaterais verificando se a diferença das médias é maior ou menor que zero. Se o p-valor calculado pelos testes for menor que o valor de significância, por exemplo 0,05 (95% de confiança), a hipótese nula é rejeitada, indicando que a média das diferenças da amostra é diferente de zero, e consequentemente, não são iguais. Para obter uma informação mais precisa, no caso, qual solução é melhor, basta avaliar o sinal do valor do teste estatístico T que também é calculado. Por exemplo, uma verificação entre duas amostras A e B, se o p-valor for menor que a significância definida e o teste estatístico T for positivo, por se tratar de uma operação de diferença entre as amostras, significa que a amostra B é melhor que a amostra A. Caso o valor do teste estatístico T for negativo, significa que a amostra A é melhor que a amostra B.

3.6.2 Teste ANOVA com medições repetidas

Os procedimentos da análise de variância (ANOVA) são utilizados para avaliar se existe uma diferença entre pelo menos duas médias de um conjunto de dados para a qual duas ou mais médias possam ser calculadas. O teste estatístico calculado para uma análise de variância é baseada na distribuição F (de Fisher e Snedecor), a qual é uma distribuição de probabilidade teórica contínua. Um valor F computado, comumente chamado de proporção F (F *ratio*) sempre vai estar dentro do intervalo $0 \leq F \leq \infty$. Existe um número infinito de distribuições F - cada uma em função do número

de graus de liberdade aplicados na análise (função tanto do número de amostras quanto do número de indivíduos em cada amostra). A análise de variância com medições repetidas, também chamada de análise de variância de um fator e com indivíduos (blocos), executa um teste de hipótese envolvendo k amostras dependentes. Quando o número de amostras for igual a dois (k = 2) o resultado deste teste ANOVA deverá ser equivalente ao test-t pareado para comparação de duas amostras dependentes (SHESKIN, 2003).

A análise da variância permite que vários grupos sejam comparados ao mesmo tempo. Em outras palavras, a análise de variância é utilizada quando se quer decidir se as diferenças amostrais observadas são reais (causadas por diferenças significativas nas populações observadas) ou casuais (decorrentes da mera variabilidade amostral). Portanto, essa análise parte do pressuposto que o acaso só produz pequenos desvios, sendo as grandes diferenças geradas por causas reais. Na análise da variância, a variabilidade observada nas amostras divide-se em duas componentes: variabilidade das observações dentro de um grupo em torno da média (dentro dos grupos) e variabilidade entre as médias dos grupos (entre grupos). Se a variabilidade entre grupos for suficientemente grande face à variabilidade dentro dos grupos, rejeita-se a hipótese nula, que afirma que todas as médias da população são iguais (PINHEIRO et al., 2009).

Na execução do teste, a média de cada amostra k é utilizada para estimar o valor da média da população que a amostra representa. Se a estatística de teste for significante, isso indica que há uma diferença significativa entre pelo menos duas médias das amostras do conjunto de k médias. Como resultado desta análise, o pesquisador pode concluir que existe uma alta probabilidade de que, pelo menos, duas das amostras representam populações com diferentes valores de média. Para o cálculo da estatística de teste, são contrastados dois componentes da variabilidade (que são partes da variabilidade total): variabilidade entre as amostras, que é variância das médias das k condições experimentais; e a variabilidade residual, que representa a quantidade de variabilidade dentro das k pontuações de cada um dos n indivíduos que não podem ser explicados com base no efeito de tratamento, ou seja, é uma variabilidade que resulta de fatores externos não controlados pelo pesquisador e muitas vezes referido como erro experimental. A proporção F, que é a estatística de teste, é obtida dividindo a variabilidade das amostras pela variabilidade residual (SHESKIN, 2003).

A formulação das hipóteses a serem testadas na análise de variância são (SHESKIN, 2003):

$$\begin{cases} H_0: \mu_1 = \mu_2 = \cdots = \mu_n \\ H_1: \ \mu_i \neq \mu_{i'}, \exists \ par \ (i, i'), onde \ i \neq \ i' \end{cases}$$

Na hipótese nula H_0 todas as médias populacionais das amostras são iguais, indicando que consequentemente as amostras são também iguais. Na hipótese alternativa H_1 pelo menos uma das médias das amostras é diferente das demais. Se os p-valores calculado pelo ANOVA, tanto para o fator em questão que é o algoritmo, como para os blocos que são as instâncias, forem menores que o valor de significância, por exemplo 0,05 (95% de confiança), a hipótese nula é rejeitada, indicando que pelo menos uma das amostras é diferente das outras. Se a hipótese alternativa for confirmada é necessário à execução de uma análise *post-hoc* para identificar com mais clareza as diferenças entre as amostras.

3.7 Testes não paramétricos

Os testes não paramétricos não são baseados em modelos estatísticos que especificam um conjunto de condições restritivas. Além de assumir que as observações são independentes, alguns testes não paramétricos assumem que as variáveis de estudo tenham continuidades fundamentais. A exigência de medição de testes não paramétricos é mais fraca, ou seja, a maioria dos testes não paramétricos não possuem restrições, e, portanto, as conclusões baseadas em inferência não paramétrica são mais gerais (SIEGEL, 1957).

Os testes não paramétricos, além da sua definição original para lidar com dados nominais ou ordinais, podem ser também aplicados a dados contínuos através da realização de transformações com base em ranques, ajustando os dados de entrada para os requisitos de teste. Eles podem realizar duas classes de análise: comparações de pares e comparações múltiplas. Procedimentos estatísticos de pares realizam comparações individuais entre dois algoritmos, obtendo em cada aplicação um p-valor independente do outro. Por conseguinte, a fim de realizar uma comparação, que envolve mais do que dois algoritmos, devem ser utilizados os testes de comparações múltiplas. Em comparações 1 x N, um método de controle é destacado (o algoritmo de melhor desempenho) por meio da aplicação do teste. Em seguida, todas as hipóteses de igualdade entre o método de controle e o resto podem ser testadas pela

aplicação de um conjunto de procedimentos *post-hoc*. Comparações N × N, considerando as hipóteses de igualdade entre todos os pares de algoritmos existentes, também são possíveis, com a inclusão de procedimentos específicos *post-hoc* para esta tarefa (DERRAC et al., 2011).

Derrac et al (2011) especifica ainda um conjunto de teste não paramétricos para comparação entre pares e múltiplas comparações. Para comparação entre pares o melhor método é o teste de Wilcoxon (WILCOXON, 1945), que é um teste não paramétrico simples, seguro e robusto. Ainda segundo Derrac et al (2011), para comparações múltiplas, o teste de Friedman (FRIEDMAN, 1937, 1940) é o melhor procedimento conhecido para testar as diferenças entre mais de duas amostras. Siegel (1957) apresenta um conjunto de vantagens para utilização dos testes não paramétricos:

- As declarações de probabilidade obtidas na maioria dos testes estatísticos não paramétricos são probabilidades exatas (exceto no caso de grandes amostras, para os quais excelentes aproximações são obtidas), independentemente da forma da distribuição da população da qual a amostra aleatória foi desenhada;
- Se forem utilizadas amostras pequenas, não existe alternativa a não ser à utilização de um teste estatístico não paramétrico;
- Existem testes estatísticos não paramétricos adequados para o tratamento de amostras compostas de observações de inúmeras populações diferentes;
- Testes estatísticos não paramétricos estão disponíveis para tratar o ranqueamento dos dados;
- Os métodos não paramétricos conseguem tratar dados classificatórios. Nenhuma técnica paramétrica aplica-se a tais dados;
- Testes estatísticos não paramétricos são geralmente muito mais fáceis de aprender e serem aplicados do que são testes paramétricos.

Embora o teste de Wilcoxon e o teste de Friedman utilizados para comparações múltiplas pertencerem aos testes estatísticos não paramétricos, os mesmos operam de maneiras diferentes. A principal diferença encontra-se no cálculo da classificação. O cálculo do teste de Wilcoxon está baseado na classificação das diferenças entre as

funções de forma independente, ao passo que Friedman e os procedimentos derivados calculam o ranqueamento entre os algoritmos (GARCÍA et al., 2009).

3.7.1 Teste de Wilcoxon pareado

O teste de Wilcoxon é usado para responder se duas amostras representam duas populações diferentes. Trata-se de um procedimento não paramétrico empregado em uma situação de teste de hipóteses envolvendo a avaliação com duas amostras. É o análogo do teste t pareado em procedimentos estatísticos não paramétricos, portanto, é um teste de pares que visa detectar diferenças significativas entre o comportamento dos dois algoritmos (GARCÍA et al., 2009).

Seja d_i a diferença entre os escores de desempenho dos dois algoritmos. As diferenças são classificadas de acordo com seus valores absolutos. Em caso de empates, pode-se aplicar um dos métodos disponíveis existentes, tais como ignorar os empates, atribuir o ranque mais alto entre outros (GIBBONS; CHAKRABORTI, 2014), embora seja recomendável a utilização da média dos ranques para o tratamento destes empates. Seja R^+ a soma dos ranques para as funções em que o segundo algoritmo superou o primeiro, e R^- A soma dos ranques para o oposto. Os ranques de $d_i = 0$ são divididos igualmente entre as somas, e se houver um número ímpar deles, um deles é ignorado (DERRAC et al., 2011):

$$\begin{cases} R^+ = \sum_{d_i>0} ranque\,(d_i) + \frac{1}{2}\sum_{d_i=0} ranque\,(d_i) \\ R^- = \sum_{d_i<0} ranque\,(d_i) + \frac{1}{2}\sum_{d_i=0} ranque\,(d_i) \end{cases}$$

Seja T a menor das somas, T = min (R^+, R^-). Se T for inferior ou igual ao valor da distribuição de Wilcoxon para N graus de liberdade, de acordo com Sheskin (2003), a hipótese nula é rejeitada, indicando que um algoritmo foi melhor que o outro, com o p-valor associado.

As formulações hipótese para o teste de Wilcoxon são (SHESKIN, 2003):

$$\begin{cases} H_0: \theta_D = 0 \\ H_1: \theta_D \neq 0 \end{cases}; \quad \begin{cases} H_0: \theta_D = 0 \\ H_1: \theta_D > 0 \end{cases}; \quad \begin{cases} H_0: \theta_D = 0 \\ H_1: \theta_D < 0 \end{cases}$$

Na hipótese nula (H_0) a mediana da diferença dos ranques é igual a zero. Existem três possibilidades para a hipótese alternativa (H_1). Se a mediana das diferenças dos ranques é diferente de zero indicando que é um teste bilateral e os testes unilaterais verificando se a diferença das medianas é maior ou menor que zero.

Se o p-valor calculado pelos testes for menor que o valor de significância, por exemplo 0,05 (95% de confiança), a hipótese nula é rejeitada, indicando que a mediana das diferenças dos ranques das amostras é diferente de zero, e consequentemente, não são iguais. Para obter uma informação mais precisa, no caso qual solução é melhor, basta avaliar o intervalo de confiança calculado no teste. Por exemplo, uma verificação entre duas amostras A e B, se o p-valor for menor que a significância definida e os valores do intervalo de confiança forem positivos, por se tratar de uma operação de diferença, significa que a amostra B é melhor que a amostra A. Caso os valores do intervalo de confiança forem negativos, significa que a amostra A é melhor que a amostra B.

3.7.2 Teste de Friedman

Na análise entre pares, ao se tentar extrair uma conclusão envolvendo mais de uma comparação pareada, pode-se obter um erro acumulado vindo de sua combinação. Em termos estatísticos, perde-se o controle sobre a taxa de erro de agrupamento (FWER - *Family-Wise Error Rate*), definida como a probabilidade de fazer uma ou mais falsas descobertas entre todas as hipóteses quando se realizam vários testes em pares. Portanto, um teste de comparação aos pares, tal como o teste de Wilcoxon, não deve ser utilizado para conduzir várias comparações envolvendo um conjunto de algoritmos porque o FWER não pode ser controlado. Neste caso de comparação entre vários algoritmos, deve ser utilizado um teste capaz de fazer este controle da taxa de erro de agrupamento, tal qual o teste de Friedman (DERRAC et al., 2011).

O teste de Friedman (FRIEDMAN, 1937, 1940) é um teste não paramétrico análogo ao teste paramétrico ANOVA. Enquanto o teste ANOVA necessita que os dados estejam em uma distribuição normal e que possuam variâncias iguais, o teste de Friedman é livre para estas restrições. Este teste utiliza os ranques dos dados ao invés de seus valores brutos para o cálculo da estatística de teste. Como o teste de Friedman não faz suposições sobre a distribuição, ele não é tão poderoso quanto o teste padrão se as populações forem realmente normais (SHESKIN, 2003).

Em seu trabalho, Zimmerman e Zumbo (1993) examinam o efeito de alguns testes não paramétricos variando as amostras com distribuições normal, uniforme, *mixed*-normal, exponencial, laplace e Cauchy. O autor conclui que o teste de Friedman

é mais poderoso que o teste ANOVA para distribuições muito assimétricas (com calda longa), como a distribuição Cauchy. Isto acontece porque estes métodos são menos sensíveis a *outliers* e oferecem mais poder que os testes paramétricos, isto se as premissas dos modelos paramétricos não forem atendidas.

O teste de Friedman é usado para responder se em um conjunto de k amostras (onde k ≥ 2), pelo menos duas destas amostras representam populações com diferentes medianas. É um teste de comparação múltipla que visa detectar diferenças significativas entre o comportamento de dois ou mais algoritmos. O primeiro passo no cálculo da estatística de teste é converter os resultados originais em ranques. Eles são calculados utilizando o seguinte procedimento (GARCÍA et al., 2009):

- Reunir resultados observados para cada par de algoritmos/problema;
- Para cada problema i, utilizar valores ranqueados de 1 (melhor resultado) até k (pior resultado). Definir estes ranques como r_i^j (1 ≤ j ≤ k);
- Para cada algoritmo j, calcular a média dos ranques obtidos em todos os problemas para obter o ranque final $R_j = \frac{1}{n}\sum_i r_i^j$.

Assim, os algoritmos são ranqueados para cada problema separadamente. O algoritmo de melhor desempenho deve ter a classificação de 1, o segundo melhor 2 e assim por diante. Mais uma vez, em caso de empate, recomenda-se o cálculo das médias dos ranques. Sob a hipótese nula, que afirma que todos os algoritmos devem possuir comportamentos semelhantes (por conseguinte, os seus ranques R_j devem ser iguais) a estatística do teste de Friedman F_f pode ser calculada como apresentado na equação (12) (GARCÍA et al., 2009).

$$F_f = \frac{12n}{k(k+1)}\left[\sum_j R_i^j - \frac{k(k+1)^2}{4}\right] \quad (1)$$

A estatística do teste de Friedman é distribuída de acordo com a distribuição χ^2 (qui-quadrado) com k - 1 graus de liberdade, quando n e k são muito grandes (como regra, n > 10 e k > 5). Para um número pequeno de algoritmos e problemas, os valores críticos exatos devem ser computados (DERRAC et al., 2011).

As formulações de hipótese para o teste de Friedman são:

$$\begin{cases} H_0: \theta_1 = \theta_2 = \cdots = \theta_k \\ H_1: \ \theta_i \neq \theta_{i'}, \exists \ par \ (i, i'), onde \ i \neq \ i' \end{cases}$$

A hipótese nula para teste de Friedman (H_0) afirma a igualdade entre as medianas das populações. A hipótese alternativa (H_1) é definida como a negação da hipótese de nula, ou seja, que pelo uma das amostras analisadas é diferente das

outras (GARCÍA et al., 2009). Se o p-valor calculado pelo teste de Friedman for menor que o valor de significância, por exemplo, 0,05 (95% de confiança), a hipótese nula é rejeitada, indicando que existe pelo menos uma das amostras que é diferente das outras. Se a hipótese alternativa for confirmada é necessário à execução de uma análise *post-hoc* para identificar as diferenças entre as amostras.

3.8 Análise *post-hoc*

O objetivo principal dos testes de hipótese para testes paramétricos e não paramétricos, aplicados a múltiplas comparações, é verificar se existem diferenças entre as amostras analisadas. Se a hipótese nula (H_0) for rejeitada, fica confirmado que pelo menos uma amostra analisada é diferente das outras. Neste caso é necessária a aplicação de uma análise *post-hoc* que pode ser capaz de fornecer informações adicionais para a identificação de qual amostra se destaca, ou seja, qual dos algoritmos analisados é melhor.

Sheskin (2003) apresenta um conjunto de procedimentos de comparação para análise *post-hoc* após aplicação do teste ANOVA. Estes testes são: múltiplos testes-t, teste de Fisher´s LSD (*Least Significant Difference* - Menor Diferença Significante) (HAYTER, 1986), teste de Bonferroni-Dunn (DUNN, 1961), teste Tukey´s HSD (*Honest Significant Difference* - Diferença Significante Honesta) (TUKEY, 1949), teste de Newman-Keuls, teste Scheffd e o teste de Dunnett. A única diferença na aplicação dos procedimentos acima mencionados para a comparação de análise de variância são as diferentes medições para a variabilidade do erro que são utilizadas. Ainda segundo Sheskin (2003), para o teste de Friedman pode ser utilizado um procedimento de comparação, que é essencialmente a aplicação do método de Bonferroni-Dunn.

Jaccard et al. (1984) apresenta um estudo comparativo entre métodos para procedimentos de múltiplas comparações entre pares. Em seu estudo, vários trabalhos na área são avaliados e discutidos apresentando as análises com os inúmeros testes realizados. Em uma das conclusões apresentadas, o teste Tukey´s HSD, apesar de apresentar um poder de teste inferior a outros, mas principalmente devido à simplicidade computacional e as características estatísticas, pode ser fortemente recomendado. Outra conclusão apresentada é que o teste de Bonferroni-Dunn não é recomendável por ser um teste inferior ao teste HSD de Tukey.

No trabalho de Howell (2013), são detalhados dois procedimentos principais de múltiplas comparações para aplicação no ANOVA. Estes procedimentos são o teste de Fisher´s LSD (HAYTER, 1986) e o teste de Bonferroni-Dunn (DUNN, 1961). Segundo o autor o procedimento de Tukey´s HSD (TUKEY, 1949) é o mais conhecido, mas não apresenta detalhes na especificação do mesmo. Hollander et al (2013) apresenta o teste de Nemeny (Wilcoxon–Nemenyi–McDonald–Thompson) (NEMENYI, 1963) como alternativa para análise post-hoc no procedimento de comparação múltipla com todos os tratamentos para o teste de Friedman com soma dos ranqueamentos.

Demsar (2006), García et al. (2010) e Derrac et al. (2011) explicam o teste de Friedman e alguns procedimentos *post-hoc* com p-valor ajustados, tais como o procedimento de Bonferroni−Dunn (DUNN, 1961), procedimento de Holm (HOLM, 1979), procedimento de Holland (HOLLAND; COPENHAVER, 1987), procedimento de Finner (FINNER, 1993), procedimento de Hochberg (HOCHBERG, 1988), procedimento de Hommel (HOMMEL, 1988), procedimento de Rom (ROM, 1990) e procedimento de Li (LI J., 2008).

Pereira et al. (2014) define que os testes de Fisher´s LSD (HAYTER, 1986) e Tukey HSD (TUKEY, 1949) que trabalham com ranqueamentos, como sendo os dois mais liberais e mais poderosos teste *post-hoc* para serem utilizados juntamente com o teste de Friedman.

Percebe-se que existem inúmeros procedimentos para a análise post-hoc tanto para teste paramétricos como para os testes não paramétricos. Cada procedimento possui suas próprias características, incluindo principalmente o controle da taxa de erro do agrupamento (FWER), sendo utilizado de acordo com os resultados que se espera alcançar. O foco deste trabalho não é descrever e detalhar o funcionamento de cada procedimento separadamente, mas sim mostrar a importância da aplicação destes testes na obtenção de informações mais detalhadas sobre o processo de comparação entre várias soluções heurísticas, de forma a identificar qual é a melhor solução.

3.9 Consideração a respeito da análise estatística de experimentos computacionais

Este capítulo apresentou um levantamento dos métodos de análise estatística de experimentos computacionais com foco em problemas de *scheduling*. Como dito anteriormente, quando se avalia trabalhos da área não é comum que os resultados apresentados tenham sido validados por métodos estatísticos. Neste sentido, muitas vezes a análise destes resultados fica comprometida não sendo possível a comparação com outros trabalhos da mesma área. Foram definidos os conceitos estatísticos necessários para o conhecimento na aplicação de pesquisas que tratam deste assunto e foram apresentados os principais testes estatísticos paramétricos, não paramétricos e análise *post-hoc* que podem auxiliar na análise de representação de resultados. Os testes estatísticos apresentados fazem parte da análise estatística realizada pelo *framework*.

Capítulo 4

IDENTIFICAÇÃO DE PADRÕES DE ANÁLISE E REPRESENTAÇÃO DOS RESULTADOS DE PESQUISA

Este capítulo apresenta uma proposta de classificação de resultados de pesquisas que utilizam heurísticas e meta-heurísticas na solução de problemas de engenharia de produção, com o objetivo principal de identificar padrões em relação a como os pesquisadores estão analisando e representando estes resultados.

4.1 Considerações iniciais

Como visto, uma das áreas mais ativas na pesquisa em métodos quantitativos aplicados à engenharia de produção é o desenvolvimento e aplicação de heurísticas e meta-heurísticas para a resolução de problemas de planejamento da produção e distribuição (CHEN; VAIRAKTARAKIS, 2005; TAVARES NETO; GODINHO FILHO, 2009). Percebe-se uma diversidade de problemas, desde planejamento da produção a curto, médio e longo prazo, do planejamento de redes logísticas, até a roteirização de veículos. Arenales et al. (2007) apresentam mais de 15 problemas clássicos distintos abordados pela Engenharia de Produção. Durante a pesquisa de técnicas para otimizar tal categoria de projeto é comum encontrar métodos exatos (como por exemplo programação matemática, *Branch and Bound* e Programação Dinâmica) e métodos que se baseiam em heurísticas e meta-heurísticas. As meta-heurísticas, normalmente correspondem a algoritmos pseudo-probabilísticos, cuja análise recai em um conjunto de técnicas estatísticas multivariadas (ARENALES et al., 2007).

A análise preliminar da literatura mostrou que existe um movimento de se realizar, cada vez mais, análises estatísticas mais complexas para a determinação da qualidade da solução da meta-heurísticas, como por exemplo, testes paramétricos tais

como teste de hipótese baseado em ANOVA (*analysis of variance* - análise de variância), teste-t e teste-F (HUANG et al., 2013), ANOVA com análise *post-hoc* (LI et al., 2011) e Teste-t (SALMASI et al., 2011; TZENG et al., 2011 e ARNAOUT et al., 2012). Alguns trabalhos fazem uso de testes não paramétricos, tais como Wilcoxon (LIANG et al., 2013), Kruskal-Wallis (BERRICHI; YALAOUI, 2013). Porém, não se percebe a adoção de padrões, sendo encontradas desde análises baseadas em média e desvio padrão até análises estatísticas mais detalhadas. Esta falta de padronização dos dados gera uma dificuldade de comparação de resultados entre pesquisas que envolvem a mesma temática.

O presente capítulo, então, tem como objetivo geral analisar um conjunto de trabalhos recentes publicados que tratam do uso de meta-heurísticas em problemas de engenharia de produção (dando o enfoque em problemas discretos de programação da produção), e analisá-los buscando identificar os principais métodos utilizados para a análise e representação dos resultados.

4.2 Metodologia de pesquisa utilizada

O desenvolvimento desta etapa da pesquisa dividiu-se em dois estágios: um estágio exploratório e um estágio quantitativo descritivo (LAKATOS; MARCONI, 2005). No estágio exploratório, buscou-se aumentar/familiarizar o conhecimento sobre o assunto, inclusive com o intuito de esclarecer conceitos. Nesta etapa foi desenvolvida a classificação que fornece suporte para o estágio quantitativo descritivo. O estágio quantitativo descritivo, por sua vez, teve como objetivo a estruturação das informações coletadas de forma a gerar análises que permitam conclusões acerca do tema estudado.

Devido ao grande conjunto de heurísticas, meta-heurísticas e demais técnicas utilizadas para se resolver problemas de programação da produção, foi necessário limitar a pesquisa a uma técnica específica. Escolheu-se a meta-heurística ACO (*Ant Colony Optimization* - Otimização por Colônia de Formigas) (BAUER; BULLNHEIMER, 2000; STÜTZLE; HOOS, 2000). Esta meta-heurística foi escolhida porque é uma técnica probabilística, que nos permite analisar resultados com componente estocástico e se mostrou muito pesquisada e relevante para o problema de *scheduling* em engenharia de produção.

4.3 Classificação proposta

A classificação proposta dos resultados de pesquisa está dividida em quatro grandes caracterizações com suas respectivas subdivisões de classificação que são apresentadas abaixo:

- Caracterização do processamento: tipo do problema de sistema de produção, o fluxo do processo e unidade de grandeza adotada;
- Caracterização da meta-heurística utilizada: heurística ou meta-heurística, *benchmark* de comparação de dados e processamento dos dados;
- Caracterização da representação dos dados: análise de dados utilizada e representação gráfica;
- Caracterização do tempo de execução.

O Quadro 4.1 apresenta uma visão sumarizada de toda a classificação proposta.

4.3.1 Caracterização do processamento

A classificação proposta inicia com a caracterização do processamento que define o tipo do problema de sistema de produção presente no trabalho analisado, qual o fluxo do processo correspondente à organização da produção e a unidade de grandeza utilizada como objetivo principal.

4.3.1.1 Tipo de problema

O tipo do problema representa qual a classificação do problema que está sendo tratado no trabalho em questão. Esta classificação dos tipos de problemas foi descrita no capítulo anterior Slack et al. (2009) e Arenales et al. (2007). Na classificação proposta foi utilizada a letra 'T' para identificar o tipo de problema, que são apresentados no Quadro 4.2, juntamente com sua codificação no modelo proposto.

Vale ressaltar que o presente trabalho está focado no levantamento de problemas de *scheduling*. Esta escolha está baseada na pesquisa apresentada no capítulo 2, que mostra que a quantidade de trabalhos publicados na solução de problemas de *scheduling* é bastante superior aos outros tipos de problemas listados.

Quadro 4.1 - Sumarização da classificação proposta

Caracterização do Processamento

Sigla	Descrição	Subcaracterística
T	Tipo de Problema	1 - Dimensionamento de Lotes de Produção (*Lot Sizing Problem*) 2 - Programação da Produção (*Scheduling Problem*) 3 - Localização de Facilidades (*Location Facility Problem*) 4 - Roteamento de Veículos (*Vehicle Routing Problem*)
F	Fluxo do Processo	1 - Estágio único com apenas uma máquina 2 - Estágio único com máquinas idênticas em paralelo 3 - Estágio único com máquinas não idênticas em paralelo 4 - Processo multiestágios unidirecional (*flow shop*) 5 - Processo multiestágios unidirecional, que permite que estágios sejam pulados (*overflow*) 6 - Processo multiestágios unidirecional, com máquinas idênticas em paralelo 7 - Processo multiestágios unidirecional, com máquinas idênticas em paralelo, com *overflow* 8 - Processo multiestágios unidirecional, com máquinas não idênticas em paralelo 9 - Processo multiestágios unidirecional, com máquinas não idênticas em paralelo, com *overflow* 10 - Processo multiestágios multidirecional, e.g., *job shop* 11 - Processo multiestágios multidirecional, com máquinas idênticas em paralelo 12 - Processo multiestágios multidirecional, com máquinas não idênticas em paralelo
U	Unidade de Grandeza	T - Tempo: 1 - Processamento (*Makespan*, Tempo de Finalização - $\sum$Ci, Temp. de Fluxo - $\sum$Fi ou $\bar{F}$) 2 - *Lateness* (Li = Ci – Di; onde $\infty < Li < \infty$; Ci - data de finalização, Di - data de entrega) 3 - *Tradiness* - Atraso (Ti = Max {Li, 0}) 4 - *Earliness* – Adiantamento (Ei = Max {-Li, 0}) $ - Valores monetários # - Numérico M – Utilização de Multi-objetivo para análise: S { U1, U2, ..., Un } - Soma ponderada, ex: S { T1, T2 }; onde U1, U2, ...,Un são elementos da unidade de grandeza P { U1, U2, ..., Un } – Pareto, ex: P { $, T2 }; onde U1, U2, ...,Un são elementos da unidade de grandeza
		Caracterização da Meta-heurística Utilizada
H	Heurística	1 - SA - *Simulated Annealing* 2 - TABU *Search* 3 - GRASP 4 - AG - Algoritmo Genético 5 - *Iterated Greedy* 6 - ACO - *Ant Colony Optimization* 7 - BCO - *Bees Colony Optimization* 8 - VNS - *Variable Neighborhood Search* 9 - *Estimation of distribution algorithm* 10 - PSO - *Particle Swarm Optimization* 11 - MetaRaPS (*Metaheuristic for Randomized Priority Search*)
B	*Benchmark* de Comparação dos Dados	E - Externo R - Real A - Aleatório: 1 - Distribuição Binomial; 2 - Distribuição de Poisson; 3 - Distribuição Normal; 4 – Distribuição Uniforme; 5 - Distribuição exponencial
P	Processamento	G - GAP (diferença entre os dados encontrados) A – Absoluto O - Outros
		Caracterização da Representação dos Dados
A	Análise estatística univariada dos dados	1 - Estatística descritiva a - Mínimo b - Máximo c - Média d - Mediana e - Desvio Padrão f - Variância g - Moda h - Desvio Médio 2 - Teste de hipótese paramétrico a - Teste - t b - Teste - p c - Teste - F d - ANOVA (Analysis of Variance) 3 - Teste de hipótese não paramétricos a - Sinal b - Mann-Whitney c - Wilcoxon d - Kruskal-Wallis e - Friedman 4 - Análise *post-hoc* a - Tukey HSD b - Fisher's LSD c - Bonferroni d - Duncan's MRT e - Nemenyi Test f - Conover-Inman g - Dwass-Steel-Critchlow-Fligner 5 - Outras variáveis de resposta a - N-vezes que atingiu a melhor solução conhecida b - Percentual (%) de Falha c - Métrica de Cobertura
G	Gráficos	1 – Barras ou Colunas 2 – Gráfico de Linha 3 – Diagrama Retangular 4 – Gráfico Circular ou de Setores 5 – Gráfico Polar 6 – Histograma 7 – Gráfico de Dispersão 8 – *Boxplot*
		Caracterização da Representação dos Dados
E	Execução	# - Limitado por número máximo de ciclos V – Limitado por valor específico T – Limitado por tempo

Fonte: elaborado pelo autor

Quadro 4.2 - Dimensões relacionadas ao tipo do problema

Tipo do Problema - T	
1	Dimensionamento de Lotes de Produção (*Lot Sizing Problem*)
2	Programação da Produção (*Scheduling Problem*)
3	Localização de Facilidades (*Facility Location Problem*)
4	Roteamento de Veículos (*Vehicle Routing Problem*)

Fonte: elaborado pelo autor

4.3.1.2 Fluxo de caracterização do processo

Maccarthy e Fernandes (2000) apresentam uma classificação multidimensional dos sistemas de produção, com uma visão estendida com o intuito de facilitar o entendimento dos sistemas produtivos. Foi utilizada a classificação dos tipos de fluxo, dentro da dimensão descrição do processamento. Para esta classificação foi utilizada a letra 'F' para identificar os tipos de fluxo, que são apresentados no Quadro 4.3, juntamente com sua codificação.

Quadro 4.3 - Dimensões relacionadas ao fluxo de materiais do problema

Fluxo de Caracterização do Processo - F	
1	Estágio único com apenas uma máquina
2	Estágio único com máquina idênticas em paralelo
3	Estágio único com máquinas não idênticas em paralelo
4	Processo multiestágios unidirecional (*flow shop*)
5	Processo multiestágios unidirecional, que permite que estágios sejam pulados (*overflow*)
6	Processo multiestágios unidirecional, com máquinas idênticas em paralelo
7	Processo multiestágios unidirecional, com máquinas idênticas em paralelo, com *overflow*
8	Processo multiestágios unidirecional, com máquinas não idênticas em paralelo
9	Processo multiestágios unidirecional, com máquinas não idênticas em paralelo, com *overflow*
10	Processo multiestágios multidirecional, ex. *job shop*
11	Processo multiestágios multidirecional, com máquinas idênticas em paralelo
12	Processo multiestágios multidirecional, com máquinas não idênticas em paralelo

Fonte: elaborado pelo autor

4.3.1.3 Unidade de grandeza

A unidade de grandeza representa qual é a função objetivo a ser alcançada na análise do problema. Dependendo do que se espera avaliar, o pesquisador pode utilizar várias unidades de grandeza em um mesmo trabalho. Na classificação proposta foi utilizada a letra 'U' para identificar as unidades de grandeza, que são apresentados no Quadro 4.4 juntamente com a codificação no modelo proposto.

Quadro 4.4 - Dimensões relacionadas as grandezas consideradas na função objetivo

Grandezas consideradas na função objetivo - U			
T	Unidade de Tempo	1	Tempo de Processamento (*Makespan*, Tempo de Finalização ou Tempo de Fluxo)
		2	*Lateness* (Li = Ci - Di; onde Ci - data de finalização, Di - data de entrega)
		3	*Tardiness* - Atraso (Ti = max {Li, 0})
		4	*Earliness* – Adiantamento (Ei = Max {-Li, 0})
$	Valores monetários		
#	Valores numéricos		
M	Utilização de multi-objetivo para análise	S { U1, U2, ..., Un } - Soma ponderada, ex: S { T1, T2 }	
		P { U1, U2, ..., Un } - Pareto, ex: P { $, T2 }	
		onde U1, U2, ...,Un, são elementos da unidade de grandeza.	

Fonte: elaborado pelo autor

4.3.2 Caracterização da meta-heurística utilizada

A próxima caracterização da classificação proposta está relacionada com características da meta-heurística que foi utilizada no trabalho, definindo especificamente a heurística, os dados de comparação de referência (*benchmark*) e o processamento utilizado na tratativa dos dados.

4.3.2.1 Heurística

A heurística representa a solução algorítmica desenvolvida que serve como base para o trabalho. Na classificação proposta foi utilizada a letra 'H' para identificar as heurísticas, que são apresentados no Quadro 4.5, juntamente com sua codificação no modelo proposto.

Quadro 4.5 - Algoritmo base heurístico utilizado

Algoritmo base utilizado - H	
1	SA - *Simulated Annealing*
2	TABU *Search*
3	GRASP
4	AG - *Genetic Algorithm*
5	IG - *Iterated Greedy*
6	ACO - *Ant Colony Optimization*
7	BCO - *Bees Colony Optimization*
8	VNS - *Variable Neighborhood Search*
9	*Estimation of distribution algorithm*
10	PSO - *Particle Swarm Optimization*
11	MetaRaPS (*Metaheuristic for Randomized Priority Search*)

Fonte: elaborado pelo autor

4.3.2.2 *Benchmark* de comparação dos dados

O *benchmark* de comparação de dados representa a base de dados utilizada para comparação entre trabalhos. Esta classificação é importante porque toda a análise de dados e a apresentação dos resultados adotam estes valores como referência. Existem inúmeras distribuições estatísticas na literatura, entretanto, na classificação proposta, foram listadas as mais utilizadas nos trabalhos de Sweeney et

al. (2007) e Bruni (2010). Na classificação proposta foi utilizada a letra 'B' para identificar os *benchmarks* utilizados, apresentados no Quadro 4.6, juntamente com sua codificação no modelo proposto.

Quadro 4.6 - *Benchmark* de comparação de dados adotado no trabalho

Benchmark de comparação adotado - B			
1	Externo		
2	Real		
3	Aleatório	1	Distribuição Binomial
		2	Distribuição de Poisson
		3	Distribuição Normal
		4	Distribuição Uniforme
		5	Distribuição Exponencial

Fonte: elaborado pelo autor

4.3.2.3 Processamento

O processamento define como o trabalho efetua as comparações com os valores de *benchmark* especificados. Na classificação proposta foi utilizada a letra 'P' para identificar os processamentos utilizados, apresentados no Quadro 4.7, juntamente com sua codificação no modelo proposto.

Quadro 4.7 - Técnica de processamento utilizada no tratamento de dados

Processamento dos dados - P	
G	GAP (distância entre os resultados encontrados e os valores de referência)
A	Valor Absoluto da informação
O	Outros

Fonte: elaborado pelo autor

4.3.3 Caracterização da representação dos dados

A próxima caracterização da classificação proposta está relacionada com a forma de análise e representação dos resultados utilizados no trabalho, definindo especificamente como a análise de dados foi feita e quais foram as representações gráficas. Bruni (2010) e Sweeney et al. (2007) apresentam os principais conceitos estatísticos e também apresentam os principais gráficos que podem ser utilizados na representação de dados.

4.3.3.1 Análise estatística dos dados

A análise de dados define como o trabalho efetua o tratamento estatístico dos resultados encontrados. Na classificação proposta foi utilizada a letra 'A' para identificar as análises de dados utilizadas, apresentados no Quadro 4.8, juntamente com sua codificação no modelo proposto.

Quadro 4.8 - Técnica utilizada para análise de dados

Análise estatística dos dados - A

1	Estatística descritiva	a	Mínimo	e	Desvio Médio
		b	Máximo	f	Desvio Padrão
		c	Média	g	Variância
		d	Mediana	h	Moda
2	Teste de hipótese paramétrico	a	Teste - t	c	Teste - F
		b	Teste - p	d	ANOVA (*Analysis of Variance*)
3	Teste de hipótese não paramétrico	a	Sinal	d	Kruskal-Wallis
		b	Mann-Whitney	e	Friedman
		c	Wilcoxon		
4	Análise *post-hoc*	a	Tukey HSD	e	Nemenyi Test
		b	Fisher's LSD	f	Conover-Inman
		c	Bonferroni	g	Dwass-Steel-Critchlow-Fligner
		d	Duncan's MRT		
5	Outras variáveis de resposta	a	N-vezes que atingiu a melhor solução conhecida		
		b	Percentual (%) de Falha		
		c	Métrica de cobertura		

Fonte: elaborado pelo autor

4.3.3.2 Representação gráfica

A representação gráfica especifica como os resultados encontrados são representados. Na classificação proposta foi utilizada a letra ‘G’ para identificar as representações gráficas utilizadas, apresentados no Quadro 4.9, juntamente com sua codificação no modelo proposto.

Quadro 4.9 - Representação gráfica utilizada

Representação gráfica utilizada - G	
1	Gráfico de barras ou colunas
2	Gráfico de linha
3	Diagrama retangular
4	Gráfico circular ou de setores
5	Gráfico polar
6	Gráfico de histograma
7	Gráfico de dispersão
8	Gráfico *boxplot*

Fonte: elaborado pelo autor

4.3.4 Caracterização do tempo de execução

O tempo de execução especifica a limitação das execuções do(s) algoritmo(s) heurísticos na busca da solução do problema. Na classificação proposta foi utilizada a letra ‘E’ para identificar os tempos de execução utilizados, apresentados no Quadro 4.10, juntamente com sua codificação no modelo proposto.

Quadro 4.10 - Critério de parada adotado na solução

Critério de parada adotado - E

#	Limitado por número máximo de ciclos de execução
V	Limitado por valor específico encontrado
T	Limitado por tempo máximo de execução

Fonte: elaborado pelo autor

4.3.5 Aplicação da classificação proposta

A Figura 4.1 apresenta uma visão geral da aplicação da classificação sugerida, com todas as possibilidades de subclassificações a serem analisadas. Esta classificação segue as seguintes regras:

- Cada característica levantada é identificada pela respectiva letra seguindo as indicações definidas nos itens anteriores, assim como a classificação da sub-característica correspondente ao que está sendo analisado;
- As características são separadas pelo símbolo ‘/’;
- Se houver mais de uma sub-característica de mesmo tipo estas devem ser representadas com a colocação do símbolo ‘-‘, por exemplo, "H4-H6", indicando que o trabalho utilizou duas heurísticas como algoritmo base;
- A ausência de determinada característica é identificada pela letra correspondente de acordo com a classificação, seguida do símbolo '-'. Por exemplo, "G-" indicando que não tem nenhuma representação gráfica.

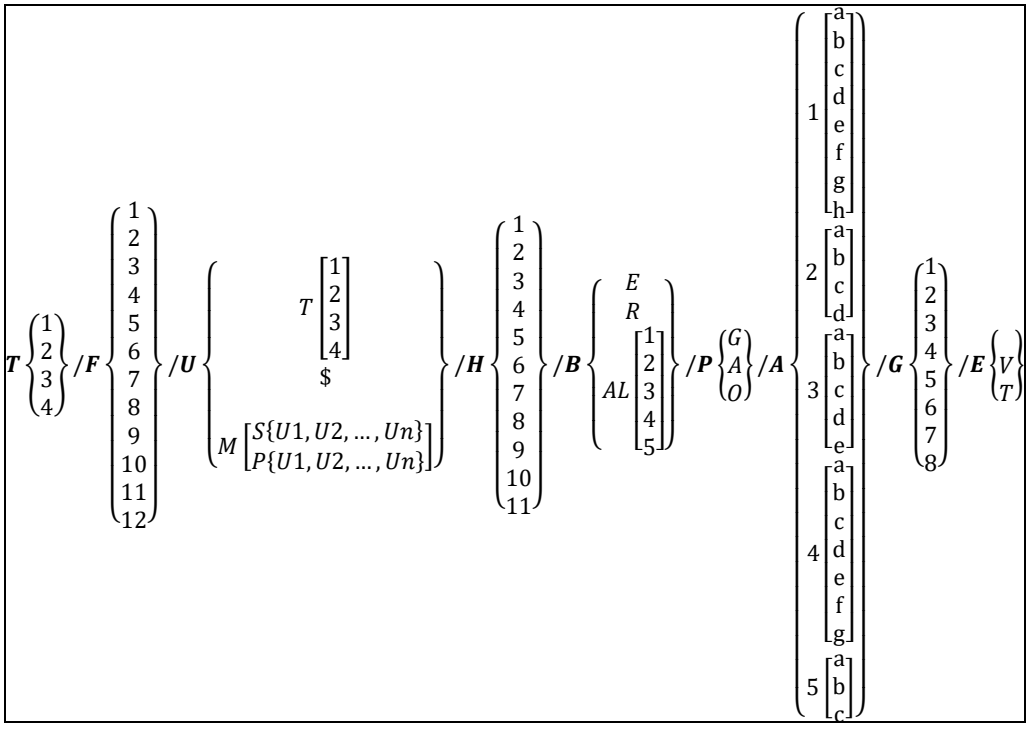


Figura 4.1 - Sumarização da classificação proposta
Fonte: elaborado pelo autor

O Quadro 4.11 apresenta um exemplo de como a classificação proposta deve ser aplicada.

Quadro 4.11 – Exemplo de aplicação da classificação proposta

Exemplo: **T2/F4/UT1–U$-UMP{T1,$}/H6/BA/PG/A1a–A1e–A3e/G1–G8/E#**	
Análise da classificação:	
T2	Problema de Programação de Tarefas - *Scheduling*
F4	Processo multiestágios unidirecional – *Flow shop*
UT1	Unidade de grandeza utilizada - Tempo - *Makespan*
U$	Unidade de grandeza utilizada – Valor Monetário
UMP {T1, $}	Unidade de grandeza multi-objetivo usando Pareto para análise com
H6	*Makespan* e Valor monetário
BA	Meta-heurística ACO - Otimização por Colônia de Formigas
PG	*Benchmark* de comparação foi aleatório
A1a	Processamento utilizado foi o GAP
A1e	Análise utilizando valor mínimo
A3e	Análise utilizando desvio padrão
G1	Análise de Teste não paramétrico de Friedman
G8	Gráfico de Barras
E#	Gráfico *Boxplot*
	Limitado pelo número máximo de ciclos de execução

Fonte: elaborado pelo autor

4.4 Parâmetros de pesquisa

A levantamento foi realizado entre os meses de junho e agosto de 2013 sendo atualizada em junho de 2014, por meio de busca de periódicos utilizando o portal de busca de publicações *Engineering Village*[12]. Este tem como base o COMPENDEX (*COMPuterized ENgineering inDEX*), que concentra periódicos na área de engenharia de produção e pesquisa operacional.

Os parâmetros de pesquisas adotados foram:

I. Combinação de palavras chaves entre os termos de problemas de produção (*single machine*, *parallel machine*, *jobshop*, *job shop*, *flowshop* e *flow shop)* e a meta-heurística de colônia de formigas *(*ACO, *ant colony)*;
II. Definição dos tipos de campos de busca: somente título (*Title*) e Assunto/Título/Resumo (*Subject/Title/Abstract*);
III. Somente artigos de *journal* (*Journal article*);
IV. Somente artigos em inglês (*English only*);
V. A busca foi limitada aos anos entre 2010 a 2014.
VI. Foram excluídas as palavras chaves “*project*”, ”*crew*”, ”*grid*”, ”*yard*”, “*thermoeletric*”, ”*eletricity*”, ”*power*”, ”*cloud*”, “*wafer*” e “*highway*”, por não

[12] www.engineeringvillage.com

serem relacionadas com o problema de *scheduling* na engenharia de produção.

Na Tabela 4.1 são apresentos os resultados obtidos pela busca assim como os filtros utilizados e a quantidade de artigos encontrados.

Tabela 4.1 - Resultados encontrados na busca de trabalhos

Nro	Filtro Utilizado no *Engineering Village*	Quantidade de artigos
1	(((ACO) WN TI) AND ((single machine) WN TI))	2
2	(((ACO) WN TI) AND ((parallel machine) WN TI))	2
3	(((ACO) WN TI) AND ((*jobshop*) WN TI))	0
4	(((ACO) WN TI) AND ((job shop) WN TI))	0
5	(((ACO) WN TI) AND ((*flowshop*) WN TI))	0
6	(((ACO) WN TI) AND ((flow shop) WN TI))	0
7	(((ant colony) WN TI) AND ((single machine) WN TI))	4
8	(((ant colony) WN TI) AND ((parallel machine) WN TI))	5
9	(((ant colony) WN TI) AND ((*jobshop*) WN TI))	0
10	(((ant colony) WN TI) AND ((job shop) WN TI))	8
11	(((ant colony) WN TI) AND ((*flowshop*) WN TI))	3
12	(((ant colony) WN TI) AND ((flow shop) WN TI))	4
13	(((ACO) WN KY) AND ((single machine) WN KY))	14
14	(((ACO) WN KY) AND ((parallel machine) WN KY))	22
15	(((ACO) WN KY) AND ((*jobshop*) WN KY))	0
16	(((ACO) WN KY) AND ((job shop) WN KY))	18
17	(((ACO) WN KY) AND ((*flowshop*) WN KY))	6
18	(((ACO) WN KY) AND ((flow shop) WN KY))	15
19	(((Ant colony) WN KY) AND ((single machine) WN KY))	32
20	(((Ant colony) WN KY) AND ((parallel machine) WN KY))	32
21	(((Ant colony) WN KY) AND ((*jobshop*) WN KY))	0
22	(((Ant colony) WN KY) AND ((job shop) WN KY))	34
23	(((Ant colony) WN KY) AND ((*flowshop*) WN KY))	13
24	(((Ant colony) WN KY) AND ((flow shop) WN KY))	33
	Total de Artigos Encontrados	**247**

Fonte: elaborado pelo autor

4.5 Artigos encontrados

Como resultado foram identificados 247 artigos. Quando aplicados os filtros de leitura como análise de título, resumo e palavras chave, foram selecionados 46, que foram utilizados na integra para o levantamento de dados. Essa diferença de resultados ocorreu porque muitos artigos foram contados mais de uma vez por aparecerem no resultado de diferentes filtros (duplicidade) ou porque muitos foram descartados por não serem adequados à classificação e ao levantamento bibliográfico proposto ou ainda por não estarem disponíveis para consulta.

Podem ser apontados quatro causas principais para a desclassificação de trabalhos. Primeiramente, mesmo passando pelos filtros, foram encontrados artigos que não tratavam do problema de *scheduling*, como por exemplo, Hirsch et al. (2012) que propõe uma solução para um problema de roteirização, ou Vendramin et al. (2012) que utiliza a técnica de otimização por colônia de formigas para resolver problemas relacionados à rede de Internet. Artigos que tratavam do problema de *scheduling*, mas usaram outra meta-heurística para a resolução do problema, como Li et al. (2013a) que empregou o problema de *scheduling* com rotas flexíveis de processamento utilizando uma técnica baseada em feromônio, também foram descartados.

Além disso, foram excluídos artigos que utilizaram a meta-heurística de otimização por colônia de formigas apenas como *benchmark* para confirmar a superioridade de outra técnica, como Belaid et al. (2012) que procuraram minimizar o tempo de *setup* total e o atraso máximo no problema de programar a ordem dos lotes de produto em tanques que servem de estoque intermediário antes de serem embalados. Várias técnicas de resolução foram propostas, entre elas a otimização por colônia de formigas, mas foi possível observar que o foco do artigo era provar a superioridade da chamada “heurística dedicada”. Por fim, a última causa para exclusão de artigos foi o fato de não adotarem o problema de *scheduling* puro, como Berrichi et al. (2010) que realizou um estudo combinando decisões de *scheduling* de produção com o de manutenção, visando minimizar o *makespan* e a indisponibilidade do sistema.

A Figura 4.2 apresenta a quantidade de artigos publicados por ano, segundo os filtros apresentados e a quantidade de artigos classificados.

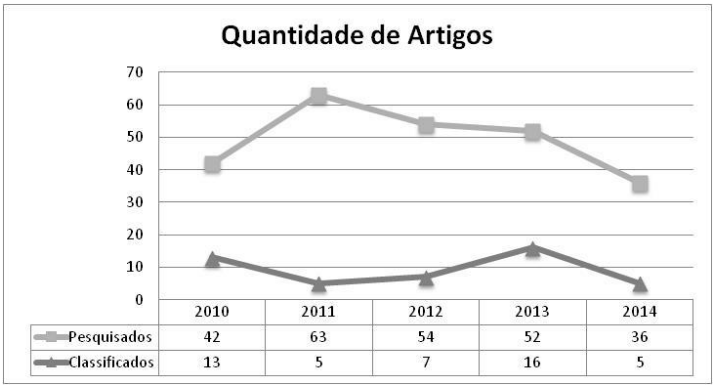


Figura 4.2 - Quantidade de artigos pesquisados e classificados
Fonte: elaborado pelo autor

4.6 Análise da classificação

Tendo com base os 46 artigos selecionados, estes foram analisados e classificados de acordo com o Quadro 4.12.

Quadro 4.12 - Classificação dos artigos encontrados segundo proposta

N	Referência	Classificação
1	(LIAO et al., 2014)	T2/F8/UT1/H1-H6/BAL3/PA/A1c-A1e/G2/ET
2	(WU et al., 2014)	T2/F1/UT1/H2-H6/BAL4/PG/A1b-A1c-A1h/G-/E#-EV
3	(ARNAOUT et al., 2012)	T2/F8/UT1/H2-H6-H11/BAL4/PG/A1h-A2a/G1-G2/E#
4	(HECKER et al., 2014)	T2/F8/UT1/H4-H6/BE/PA/A2d/G2/E#-ET
5	(AHMADIZAR; RABANIMOTLAGH, 2013)	T2/F10/UT1/H6/BE-BAL4/PA/A1a-A1c/G-/E#
6	(HECKER et al., 2013)	T2/F9/UT1/H6-H10/BE/PG/A1a/G2/EV
7	(CHEN et al., 2013)	T2/F8/UT1/H4-H6/BE-BAL4/PG/A1c-A1h/G2/E#
8	(WU et al., 2013)	T2/F1/UT1/H4-H6/BAL4/PG/A1b-A1c-A1h/G2/E#-EV
9	(CHENG et al., 2013)	T2/F2/UT1/H6/BAL4/PG/A1a-A1b-A1c/G-/E#
10	(KORYTKOWSKI et al., 2013)	T2/F10/UT1-UT3/H6/BAL4/PA/A1c/G2/E#
11	(LIN et al., 2013)	T2/F8/UT3/H6/BAL4/PG/A1c-A5a/G-/E#
12	(HUANG; YU, 2013)	T2/F6/UMS{UT1-UT3-UT4}/H6/BAL4/PG-PA/A1c-A1b-A1c-A1f/G-/E#
13	(THIRUVADY et al., 2013)	T2/F1/UT3/H6/BE-BAL4/PG-PA/A1a-A1c-A1e-A5b/G6-G8/E#-ET
14	(LI et al., 2013)	T2/F12/UT1-UT3/H6/BAL3/PG/A1b-A1c/G1-G2/E-
15	(BERRICHI; YALAOUI, 2013)	T2/F6/UT3/H4-H6-H10/BAL4/PG-PO/A1a-A1b-A1c-A3d/G2-G8/E#
16	(HUANG et al., 2013)	T2/F11/UMS{UT3-UT4}/H6/BAL4/PG-PA/A1a-A1b-A1c-A1e-A2d/G-/E#
17	(LUO et al., 2013)	T2/F6/UT1-U$/H6/BAL4/PG/A1a-A1b-A1c-A5c/G7/E#
18	(KUO; CHENG, 2013)	T2/F10/UMS{UT3,UT4}/H6-H10/BAL4/PA/A1a-A1b-A1c-A1h-A2c/G2/E#
19	(LIANG et al., 2013)	T2/F2/UMP{UT1-UT2}/H6-H8/BR/PO/A1c-A3c/G2-G7-G8/E#
20	(ROSSI; LANZETTA, 2013a)	T2/F4/UT1/H6/BE/PG/A1c-A1h/G2/E#
21	(ROSSI; LANZETTA, 2013b)	T2/F4/UT1/H6/BE/PA/A1a-A1c/G-/E#
22	(KRISHNARAJ et al., 2012)	T2/F4/UT1/H6/BE/PG-PA/A1a-A1c-A2a/G-/E#
23	(AHMADIZAR, 2012)	T2/F4/UT1/H6/BE-BAL4/PA/A1a-A1c/G-/E#
24	(LIANG et al., 2012)	T2/F1/UT1/H6/BAL4-BAL5/PG/A1c/G-/E#
25	(KESKINTURK et al., 2012)	T2/F2/UT1/H3-H6/BAL4/PA/A1a-A1c/G8/E#
26	(TZENG et al., 2011)	T2/F4/UT1/H6-H9/BE-BAL4/PG/A1c-A2d-A2a/G-/ET
27	(AHMADIZAR; HOSSEINI, 2012)	T2/F1/UMS{UT1(*makespan*),UT1 (tempo total de finalização)}/H6/BAL4/PA/A1a-A1c/G-/E#
28	(XU et al., 2012)	T2/F1/UT1/H6/BAL4/PG-PA/A1a-A1b-A1c-A1e/G8/E#-EV
29	(LI et al., 2011)	T2/F10/U$/H6-H1/BAL4/PA/A1a-A1c-A1e-A2d-A5a-A4b/G-/E#
30	(LIAO et al., 2011)	T2/F4/UT1/H6/BAL4/PG-PA/A1a-A1c/G-/E#-ET
31	(TAVARES NETO; GODINHO FILHO, 2011)	T2/F4/UMS{UT1-U$}/H6/BAL4/PG/A1c-A1e/G8/E#
32	(RABANIMOTLAGH, 2011)	T2/F4/UMS{UT1 (*makespan*),UT1(flow time)}/H6/BE/PG/A1a/G-/E#
33	(SALMASI et al., 2011)	T2/F4/UT1/H6/BAL4/PG/A1c-A2a/G-/ET
34	(XING et al., 2010)	T2/F10/UT1/H6/BE-BAL4/PA/A1a-A1c-A1e/G-/E#
35	(YAGMAHAN; YENISEY, 2010)	T2/F4/UMS{UT1 (flowtime) ,UT1 (*makespan*)}/H6/BE/PG/A1a/G1/E#
36	(ARNAOUT et al., 2009)	T2/F3/UT1/H6/BAL4/PG-PA/A1a-A1b-A1c/G2-G7/E#
37	(GATICA et al., 2010)	T2/F2/UT2/H6/BAL4/PG-PA/A1c-A2d-A5a/G1/E-
38	(SEO; KIM, 2010)	T2/F10/UT1/H6/BE/PG-PA/A1a-A1c-A1e/G-/E#
39	(MIRABI, 2010)	T2/F4/UT1/H6/BAL4/PG/A1a-A1b-A1c-A2a/G2/E#
40	(BEHNAMIAN; ZANDIEH; et al., 2010)	T2/F2/UMS{UT1,UT3,UT4}/H1-H6-H8/BAL4/PG-PO/A1c-A2d/G2-G7/ET
41	(BEHNAMIAN; FATEMI GHOMI; et al., 2010)	T2/F7/UMS{UT3,UT4}/H1-H6-H8/BAL4/PG/A1c-A2d/G2-G8/ET
42	(ALMEDER; MÖNCH, 2010)	T2/F2/UT3/H6-H8/BAL4/PG/A1c-A3c/G-/ET
43	(CHEN et al., 2010)	T2/F2/UT1/H3/H6/PG/A1b-A1c/G2-G6/EV-ET
44	(HUANG, 2010)	T2/F10/UT1-U#/H6/BAL4/PG-PA/A1a-A1b-A1c/G-/E#
45	(CHENG et al., 2010)	T2/F1/UT1/H6/BAL4/PA/A1a-A1b-A1c/G2/E#
46	(UDHAYAKUMAR; KUMANAN, 2010)	T2/F10/UT1/H6/BE/PG-PA/A1a-A1c/G2/E#-EV

Fonte: elaborado pelo autor

A partir da classificação realizada é possível avaliar os dados de forma a verificar como os pesquisadores estão analisando e representando os resultados de suas pesquisas. Na Figura 4.3 é apresentada uma visão de quais foram os meios de publicação dos trabalhos classificados. Percebe-se que a grande maioria dos artigos foram publicados em jornais relacionados a engenharia de produção e pesquisa operacional. Vale destacar também a quantidade de artigos publicados em jornais relacionados à computação e inteligência artificial.

Figura 4.3 - Meio de publicação dos artigos analisados
Fonte: elaborado pelo autor

Figura 4.4 - País de origem dos artigos analisados
Fonte: elaborado pelo autor

Já a Figura 4.4 apresenta os países de origem dos autores dos trabalhos, destacando os países de Taiwan, China e Irã. Observam-se trabalhos realizados em pesquisas colaborativas, tais como Almeder e Monch (2010) desenvolvido por pesquisadores da Áustria e da Alemanha e artigos feito em conjunto entre pesquisadores lotados nos Estados Unidos tais como Liao et al. (2014), Liang et al. (2012) e Keskinturk et al. (2012).

A Figura 4.5 apresenta a classificação dos artigos levantados quanto ao tipo de problema de engenharia de produção. Destacam-se os trabalhos direcionados para o problema de *flowshop* (total de dezesseis), seguido de trabalhos com máquinas paralelas (total de doze).

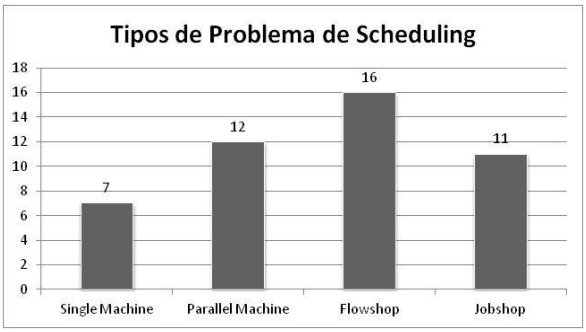


Figura 4.5 - Classificação dos artigos analisados por tipo de problema
Fonte: elaborado pelo autor

Quanto à classificação multidimensional de problemas de sistemas de produção, incluindo uma classificação do fluxo de processo, a Figura 4.6 apresenta os artigos analisados. Novamente percebe-se uma aglomeração de trabalhos voltados para problemas de *flowshop* e máquinas paralelas.

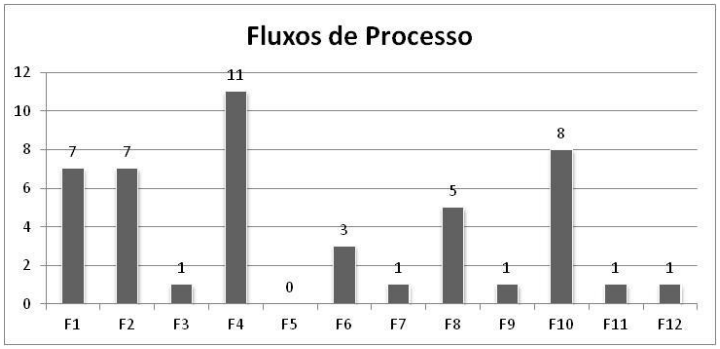


Figura 4.6 - Classificação dos artigos analisados por fluxo de processo
Fonte: elaborado pelo autor

Dentre os trabalhos analisados, 53,62% das publicações, adotam como função objetivo o tempo, tais como minimização do *makespan*, tempo de finalização ou tempo

de fluxo. Sendo que 15,94% das publicações, adotam a função objetivo baseada na minimização do atraso total do processo produtivo. Vale destacar as publicações que utilizam de função multiobjetivo como, por exemplo, a realização da soma ponderada de uma função de tempo (*makespan*) e valores monetários (LUO et al., 2013). A Figura 4.7 apresenta a classificação dos artigos analisados por unidades de grandeza.

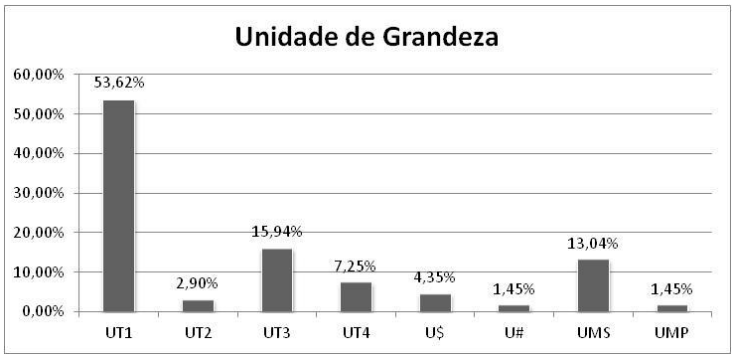


Figura 4.7 - Classificação dos artigos analisados por unidade de grandeza
Fonte: elaborado pelo autor

Como definido anteriormente, o foco deste levantamento foi dado para pesquisas que utilizam de otimização por colônia de formigas (ACO) para solução de problemas de *scheduling* na engenharia de produção. Consequentemente, observa-se na Figura 4.8 que todos os trabalhos analisados lidam com a meta-heurística ACO. Demais meta-heurísticas citadas, tais como SA (*simulated annealing*), AG (*genetic algorithm*) e VNS (*variable neighborhood search*), ou foram desenvolvidas como soluções híbridas com ACO ou foram desenvolvidas em conjunto para obtenção de soluções para comparações de *benchmark* com o ACO.

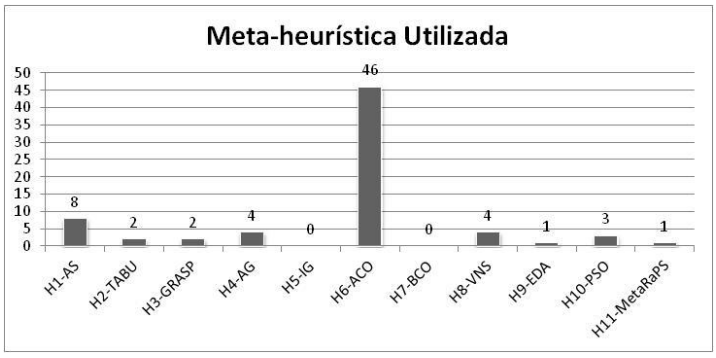


Figura 4.8 - Classificação dos artigos analisados por meta-heurística utilizada
Fonte: elaborado pelo autor

Quando se utilizam heurísticas e meta-heurísticas na busca de soluções, corre-se o risco de não obter valores ótimos, que poderiam ser encontrados através de métodos exatos. Desta forma, os pesquisadores fazem uso, muitas vezes, de

bibliotecas de problemas prontos, tal como OR-Library[13], ou ainda utilizam procedimentos pré-definidos para a criação dos *benchmarks* de comparação, tal como Taillard (1993). A Figura 4.9 apresenta a classificação dos *benchmarks* de comparação dos trabalhos analisados. Vale destacar que 28,30% dos trabalhos utilizaram um *benchmark* externo baseado em bibliotecas de problemas disponíveis. A grande parte dos trabalhos, 64,15%, obtiveram os *benchmarks* de forma aleatória utilizando uma distribuição estatística uniforme em decorrência as características dos problemas de *scheduling*.

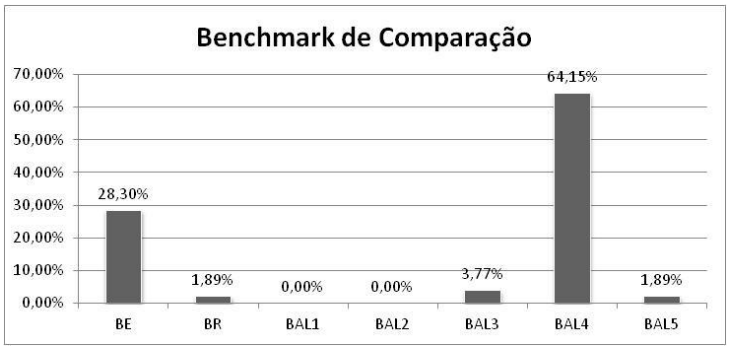


Figura 4.9 - Classificação dos artigos analisados por *benchmark* de comparação
Fonte: elaborado pelo autor

A Figura 4.10 apresenta o tipo de processamento aplicado aos dados dos trabalhos analisados. Percebe-se que 43,48% dos trabalhos utilizam como tipo de processamento o GAP, ou seja, o cálculo da diferença da solução ótima existente em relação à solução encontrada. Outros 26,09% fazem uso de valores absolutos, dando ênfase somente à solução encontrada no trabalho. Vale destacar a quantidade de artigos, 23,91%, que utilizam um processamento baseado em uma junção do GAP e de valores absolutos obtidos.

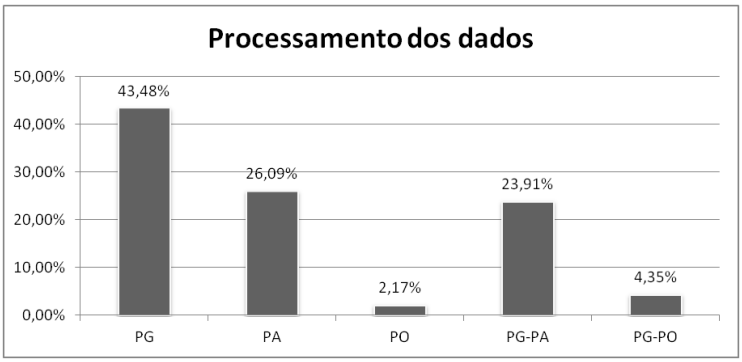


Figura 4.10 - Classificação dos artigos analisados pelo processamento
Fonte: elaborado pelo autor

[13] https://files.nyu.edu/jeb21/public/jeb/info.html

O foco deste levantamento foi identificar os tipos de análises de dados e a representações gráficas utilizadas nos artigos avaliados, de forma a identificar possíveis padrões utilizados pelos pesquisadores. Assim na Figura 4.11 são apresentados em valores numéricos os tipos de análise de dados utilizados.

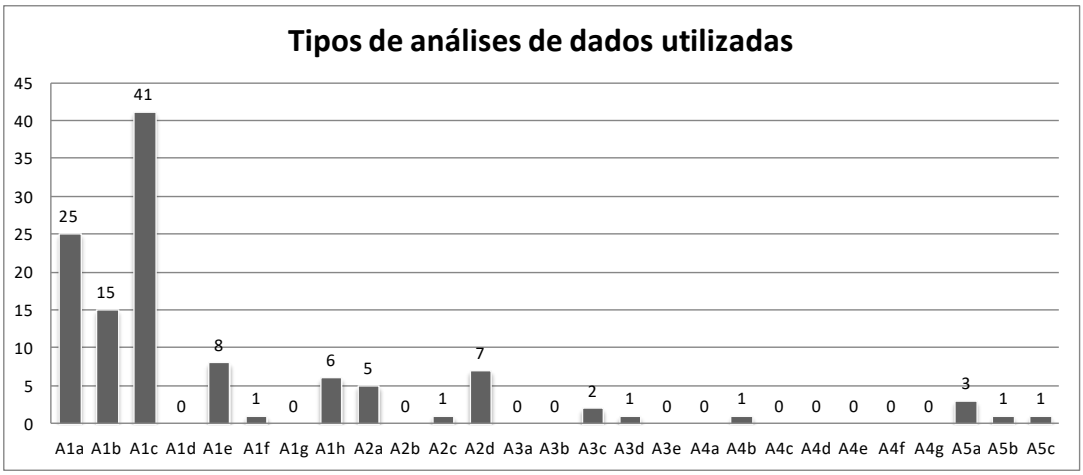


Figura 4.11 - Classificação dos artigos analisados pelo tipo de análise de dados
Fonte: elaborado pelo autor

Pode-se observar que existe uma concentração de análise de dados simples aplicada aos trabalhos avaliados. A grande maioria dos artigos faz uso somente de valores mínimos (A1a), máximos (A1b) e médios (A1c). Quando se avalia um trabalho somente por este tipo de análise, fica muito difícil saber se os resultados apresentados e discutidos são de qualidade ou não, ou seja, muitas vezes o pesquisador define que o seu método apresenta bons resultados, mas é impossível comprovar estas afirmações. Mesmo se no trabalho forem apresentados valores de desvio padrão (A1e) ou desvio médio (A1h) a confirmação da qualidade destas informações não poderá ser garantida.

Para uma análise mais detalhada dos dados levantados, foi realizada uma subclassificação, de forma a categorizar os tipos de análise de dados utilizada. Se a pesquisa utilizou para análise de dados as características de A1a até A1h (mínimo, máximo, média, mediana, desvio padrão, variância, moda e desvio padrão) e A5a até A5c (N-vezes que atingiu a melhor solução conhecida, % de falhas e métrica de cobertura), foi classificada como sendo de "Análise Estatística Simples". Se a pesquisa utilizou para análise de dados as características A2 (teste de hipótese paramétrico), A3 (teste de hipótese não paramétrico) e A4 (análise *post-hoc*), foi classificada como sendo de "Análise Estatística Elaborada". A quantidade de artigos

analisados aplicando esta nova classificação pode ser vista na Figura 4.12, onde observa-se que em 29 dos artigos avaliados, que representa 63%, os pesquisadores aplicaram análises estatísticas simples.

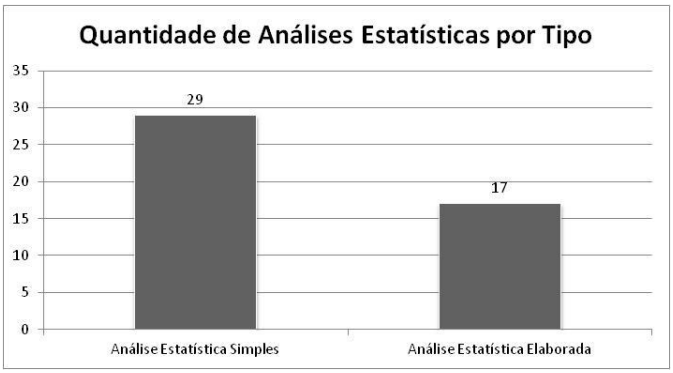


Figura 4.12 - Classificação dos artigos analisados pela quantidade de análises estatísticas por tipo
Fonte: elaborado pelo autor

Na Figura 4.13 é apresentada a quantidade de tipos de gráficos utilizados nos artigos analisados. Um destaque, 35,71%, é justamente a não utilização de representações gráficas na visualização dos resultados das pesquisas. Outros destaques podem ser dados na utilização de gráficos de linha, 33,93%, e na utilização do gráfico *boxplot*, 12,5%, que possui uma visualização mais clara dos resultados obtidos.

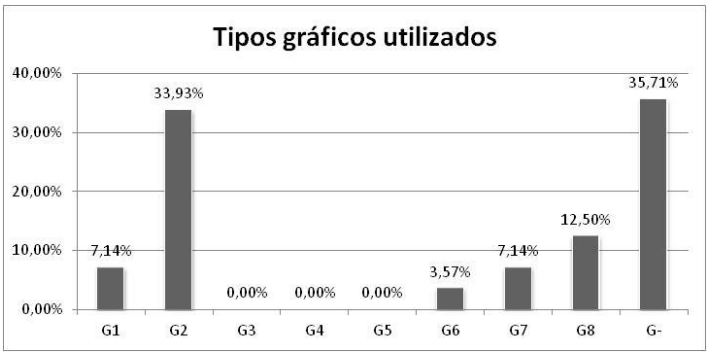


Figura 4.13 - Classificação dos artigos analisados por tipo de gráfico utilizado
Fonte: elaborado pelo autor

Por fim, a Figura 4.14 apresenta a porcentagem dos trabalhos de acordo com a condição de parada da solução apresentada. Observa-se que 53,70% dos trabalhos utiliza como condição de parada a limitação pelo número máximo de ciclos da solução proposta. Isto ocorre devido à característica da meta-heurística de otimização por colônia de formigas, que precisa da definição da quantidade de ciclos em que o algoritmo é executado, de forma a encontrar uma solução factível. Observa-se

também que 18,52% dos trabalhos utilizam como condição de parada a limitação pelo tempo de execução.

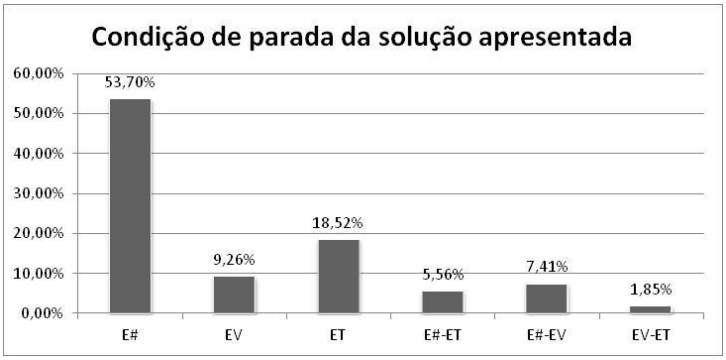


Figura 4.14 - Classificação dos artigos analisados pela condição de parada da solução apresentada
Fonte: elaborado pelo autor

4.7 Discussões dos resultados

O levantamento e classificação dos dados dos 46 artigos encontrados permitiu um estudo mais detalhado sobre como os pesquisadores realizam as análises e representações gráficas dos seus resultados de pesquisa. Vale ressaltar que os valores referentes ao ano de 2014 podem estar inferiores a realidade, devido ao fato da atualização da pesquisa ter sido realizada em junho de 2014. Neste estudo foi adotada a subclassificação definida anteriormente para "Análise Estatística Simples" e "Análise Estatística Elaborada" para expor as análises e representações gráficas utilizada nos trabalhos.

Na Figura 4.15 mostra uma tendência dos pesquisadores utilizarem gráficos de linha para apresentarem seus resultados. Mesmo que nos anos de 2011 e 2012 não tenha sido analisado nenhum trabalho com a apresentação de gráficos de linha observa-se uma elevação de utilização deste tipo de gráfico no ano de 2013. Outro destaque fica para o uso de gráficos do tipo *boxplot* que apresenta uma análise mais detalhada dos dados encontrados (mediana, primeiro e terceiro quartil, valores máximo e mínimo), sendo utilizado para avaliar a distribuição empírica dos dados. Apesar de não ser muito expressivo, é possível acompanhar um aumento no uso deste gráfico entre os anos de 2010 a 2013. Um fato relevante neste levantamento é justamente a não utilização de representações gráficas para avaliação dos resultados de pesquisa. No gráfico observa-se um total de 20 trabalhos nesta situação e com um leve crescimento nos anos de 2012 e 2013.

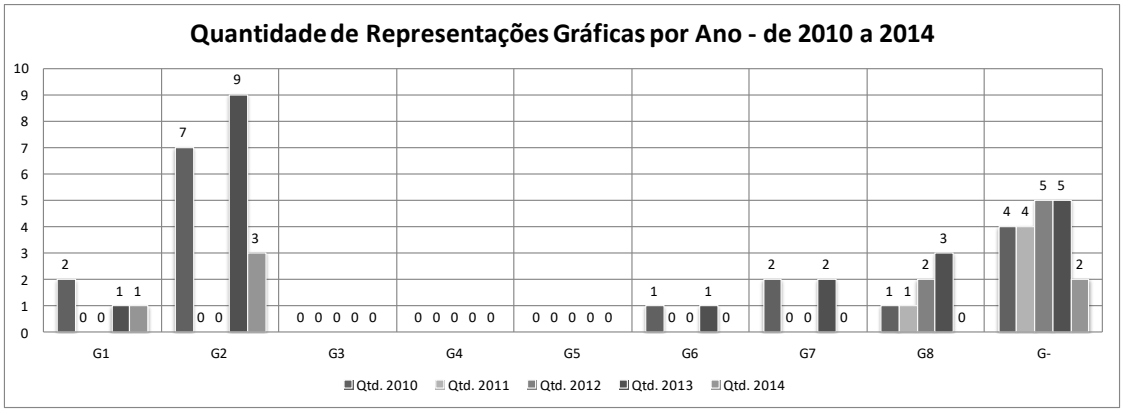


Figura 4.15 - Quantidade de representações gráficas por ano - de 2010 a 2014
Fonte: elaborado pelo autor

Quando se avalia a representação gráfica junto com a análise estatística elaborada, conforme pode ser visto na Figura 4.16, o que observa-se é uma redução da não utilização de gráficos (G-) e uma diminuição da utilização de gráficos de linha (G2). Outro ponto é o aumento da utilização de gráficos de dispersão (G7) e gráfico *boxplot* (G8), possivelmente devido à necessidade de uma melhor representação dos dados estatísticos identificados. Ao se fazer esta mesma avaliação para análise estatística simples, observa-se que a maioria das pesquisas ou usam representação de gráfico de linha ou nem mesmo usam gráficos para representar os resultados. Estes valores podem ser vistos.

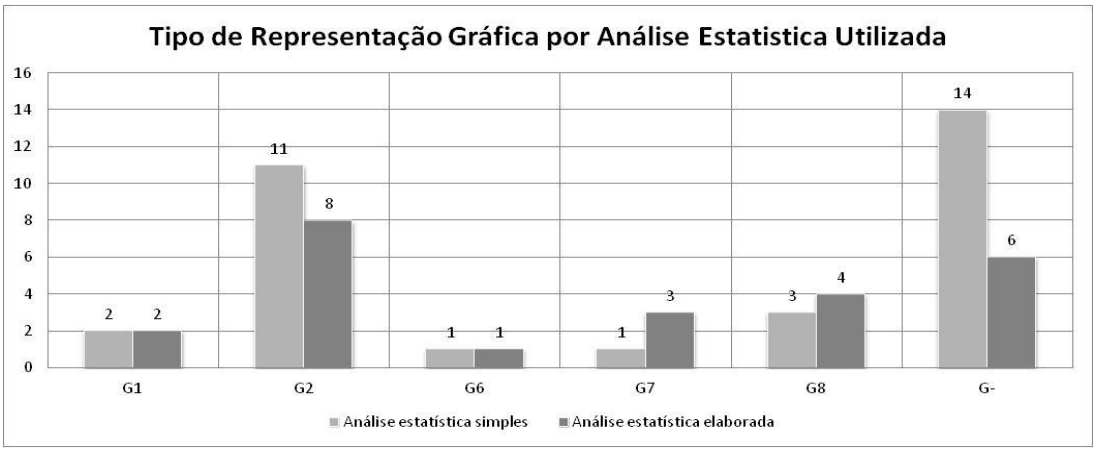


Figura 4.16 - Tipo de representação gráfica por análise estatística utilizada
Fonte: elaborado pelo autor

A Figura 4.17 apresenta o agrupamento das análises estatísticas classificadas por ano. Analisando visualmente o gráfico, fica difícil identificar padrões, mas observa-se algumas aglomerações de valores, principalmente nas análises de mínimo (A1a), máximo (A1b), média (A1c), desvio padrão (A1e) e desvio médio (A1h), todos

classificados como análise estatística simples. Para análises estatísticas elaboradas observa-se as aglomerações nos teste paramétricos Teste-t (A2a) e no ANOVA (A2d). Vale ainda destacar a utilização de métodos não paramétricos utilizados para a garantia da qualidade dos resultados tais como Wilcoxon (A3c) e Kruskal-Wallis (A3d).

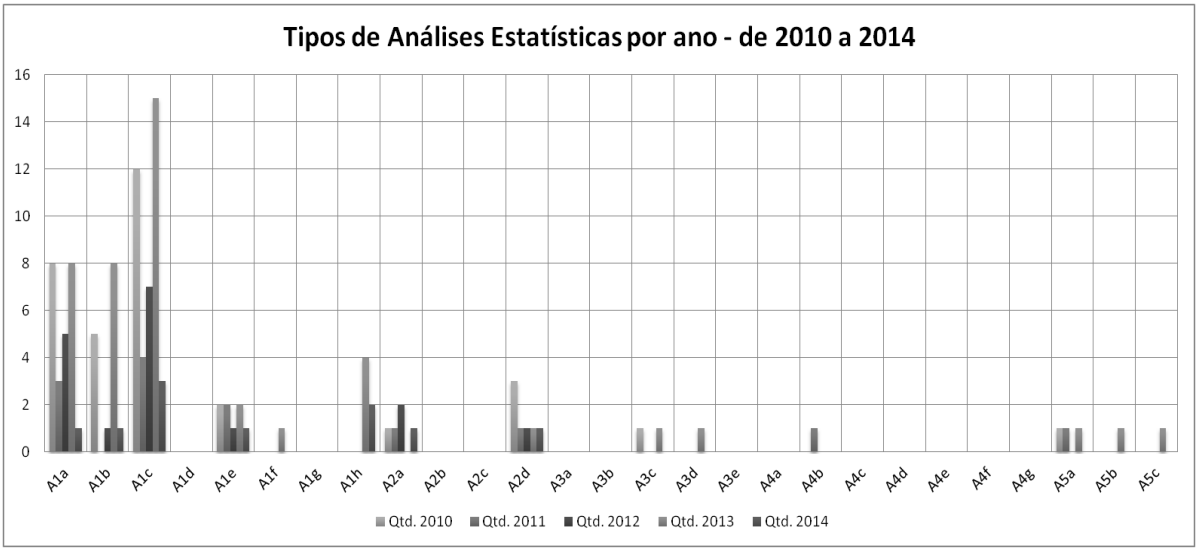


Figura 4.17 - Tipos de análises estatísticas por ano - de 2010 a 2014
Fonte: elaborado pelo autor

A Figura 4.18 apresenta todos os testes estatísticos levantados e relaciona estes de acordo com a utilização dos respectivos anos de publicação. Desta forma observam-se duas concentrações ou picos importantes que são a utilização dos testes paramétricos Teste-t (A2a) e ANOVA (A2d). Lembrando que estes testes são adotados para garantir que os resultados da pesquisa possam ser validados por alguma técnica estatística.

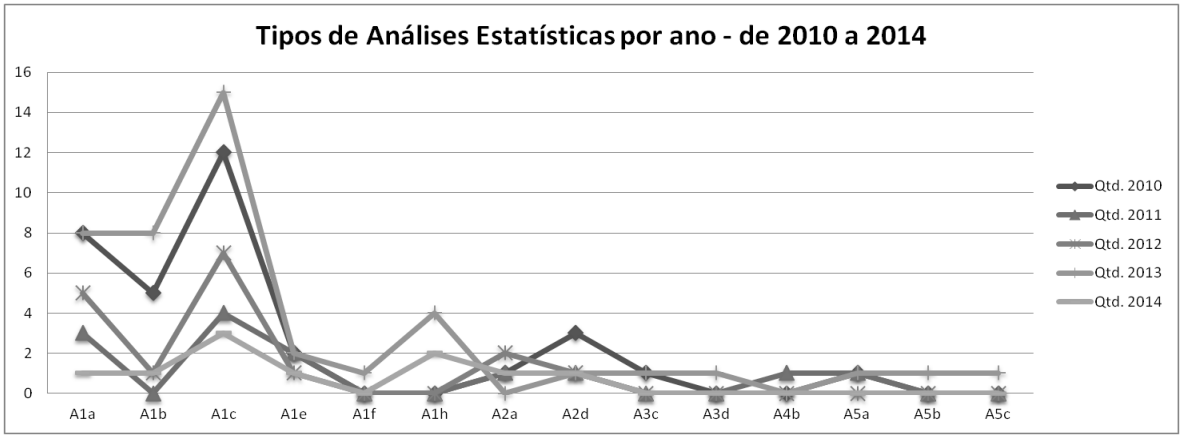


Figura 4.18 - Sumarização dos tipos de análises estatísticas por ano - de 2010 a 2014
Fonte: elaborado pelo autor

Por fim, na Figura 4.19 pode-se observar como os pesquisadores aplicam análise de dados estatísticos de acordo como tipo de problema estudado. Como destaque, têm-se os de problemas de *flowshop* e máquinas paralelas que fazem uso de análises estatísticas elaboradas. Isto se deve ao fato de que nestes tipos de problemas, muitas vezes, os resultados são próximos de outros trabalhos já publicados e o pesquisador deve apresentar seus resultados garantindo que são comparativamente melhores que outros.

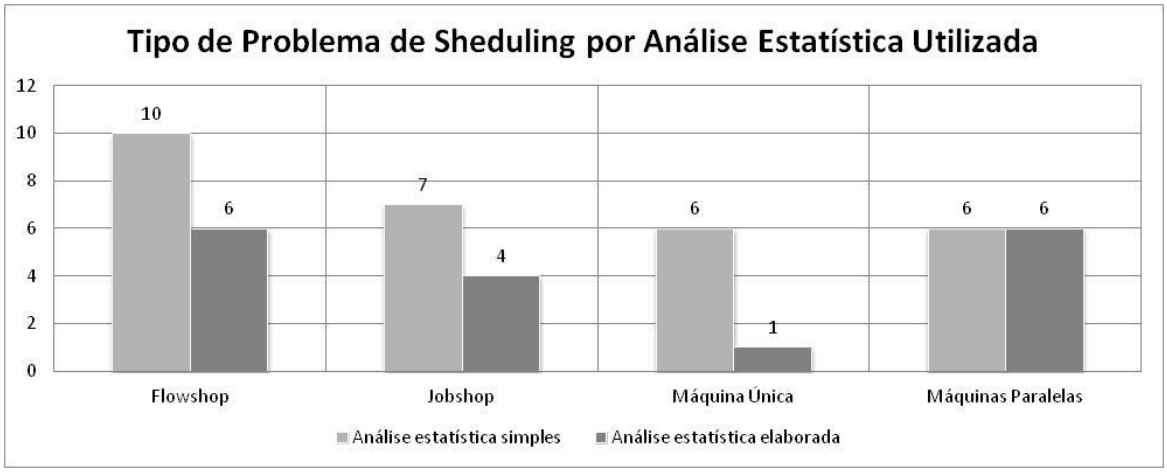


Figura 4.19 - Tipo de problemas de *scheduling* por análise estatística utilizada
Fonte: elaborado pelo autor

4.8 Padrões identificados

Observa-se na pesquisa realizada que ainda não existe uma preocupação, por parte de vários pesquisadores, na análise estatística e representação dos resultados de saída. A pesquisa mostrou que muitos trabalhos são publicados sem apresentar os resultados em gráficos e nem mesmo realizam uma análise estatística mais detalhada. Considerando que a análise dos resultados de saída utilizando técnicas paramétricas e não paramétricas pode esclarecer dúvidas sobre as hipóteses definidas pelos pesquisadores. Este é o caminho para que pesquisas na área possam ser melhores avaliadas e comparadas com trabalhos correlatos.

Apesar da pesquisa ser focada em problemas de *scheduling* com aplicação de meta-heurística de colônia de formigas (ACO) e limitado aos anos entre 2010 e 2014, consegue-se verificar quais são as principais ferramentas estatísticas e gráficos utilizados na análise e representação de resultados. Abaixo são consolidados alguns padrões que foram identificados como usuais em pesquisa desta área.

- **Análise estatística dos resultados de pesquisa:**
  I. Deve ser calculado os valores da estatística descritiva tais como mínimo, máximo, médio, desvio padrão e mediana dos resultados encontrados fornecendo assim uma análise prévia;
  II. Os resultados da pesquisa devem ser validados através de um método estatístico, garantindo assim a qualidade dos resultados apresentados, podendo ser utilizado os testes paramétricos tais como Teste-t e ANOVA ou teste não paramétricos tais como Wilcoxon e Kruskal-Wallis;
  III. Apesar da pesquisa não ser conclusiva em relação a utilização de uma análise *post-hoc*, tal qual desenvolvida por Li et al. (2011), quando se compara mais de dois métodos ou algoritmos, somente realizar um teste estatístico não fornece a informação necessária para o pesquisador. O teste vai informar se os métodos ou algoritmos são iguais ou diferentes. Neste caso é necessário utilizar uma análise *post-hoc*, tais como Tukey HSD, Fisher´s LSD e teste de Nemenyi, que é capaz de fornecer uma informação mais precisa dos resultados.
- **Representação gráfica dos resultados:**
  I. Utilização de gráficos de linha para representação contínua dos dados obtidos na pesquisa, podendo ser, por exemplo, para apresentar o comportamento de cada algoritmo testado em relação às instâncias ou então para representar a convergência destes algoritmos ao valor ótimo, dentre outros;
  II. Utilização de gráficos *boxplot* para representação de resultados estatísticos calculados fornecendo assim uma ferramenta para comparação;
  III. Utilização de gráficos de dispersão que também podem ser utilizados na análise do comportamento dos algoritmos frente aos valores calculados para cada instância do problema.

4.9 Consideração a respeito da identificação de padrões de análise e representação dos resultados de pesquisa

Este capítulo apresentou uma proposta de classificação de pesquisas que usam otimizações baseadas em heurísticas para solução de problemas de sistemas de produção. Foram definidas como premissas desta pesquisa a utilização da meta-heurística de colônia de formigas (ACO) e aplicadas a problemas de programação da produção (*scheduling*), limitado aos anos entre 2010 e 2014. O objetivo principal desta classificação foi o levantamento e a especificação de padrões de análise e representação de resultados de pesquisas na área.

A classificação proposta descreve a categorização das publicações relacionadas ao tema. Todas estas informações e as teorias estatísticas disponíveis foram compiladas de forma a construir um modelo de classificação com o intuito de facilitar a identificação de padrões utilizados. Esta classificação foi aplicada a um conjunto de 46 (quarenta e seis) trabalhos publicados nos últimos anos (de 2010 a 2014). Pode-se comprovar a viabilidade de uso da classificação mostrando a facilidade de análise e comparação das características permitindo assim o estabelecimento de padrões utilizados.

A pesquisa propiciou a identificação de alguns padrões que dizem respeito à análise e representação de dados que se aplicados, poderão facilitar o processo de compreensão, análise e comparação entre trabalhos correlatos. A consolidação dos padrões estabelecidos são utilizados para a especificação e construção do *framework* base de validação de trabalhos que usam heurísticas na solução de problemas de sistemas de produção, que é o objetivo principal desta tese.

Capítulo 5

PROCESSO DE DESENVOLVIMENTO DE PESQUISAS DE OTIMIZAÇÃO BASEADAS EM HEURÍSTICAS

Este capítulo apresenta um processo de desenvolvimento de pesquisas de otimizadores baseados em heurísticas. É exposto o fluxo de processo principal com todas as descrições e o detalhamento de cada etapa deste processo fornecendo assim base para o desenvolvimento do *framework*.

5.1 Processo de desenvolvimento

Considerando o capítulo 2 e os trabalhos de Barr et al. (1995), Montgomry (2001) e Kothari (2004) é especificado o processo de pesquisas comparativas em otimização utilizando algoritmos heurísticos e meta-heurísticos aplicados a problemas de *scheduling* em engenharia de produção. Apesar de ser especificado para problemas de *scheduling*, o processo modelado pode ser aplicado a outros tipos de problemas de engenharia de produção.

Foram definidos seis processos para o método proposto, sendo:

1. **Definir requisitos da pesquisa e dos algoritmos** - especifica todas as informações sobre a pesquisa que será realizada, qual o problema de engenharia de produção que será tratado na pesquisa e quais são os algoritmos que serão comparados;
2. **Especificar *benchmarks* e instâncias de comparação** - especifica a biblioteca de instâncias que será utilizada e qual o conjunto de instâncias definidas no *benchmark* de execução dos algoritmos e qual o subconjunto de instâncias para o *benchmark* de teste de parametrização;

3. **Desenvolver a solução do algoritmo (análise, projeto, codificação e teste)** - propõe um processo de desenvolvimento da solução algorítmica baseado em três etapas;
4. **Realizar a parametrização de configuração do algoritmo heurístico** - especifica um conjunto de grupos de parâmetros de teste e executa os algoritmos com estes grupos e com o *benchmark* de teste para a definição dos melhores parâmetros de execução do algoritmo;
5. **Executar o algoritmo** - executa o algoritmo com os parâmetros de configuração e com as instâncias do *benchmark* de execução, armazenado os resultados encontrados;
6. **Realizar análise estatística e reportar os resultados** - especifica um conjunto de padrões de análises estatísticas e representação dos resultados encontrados na pesquisa, exibindo estas análises em forma de relatório.

O objetivo principal deste método é realizar pesquisas comparativas entre algoritmos heurísticos e neste sentido é necessário definir um conjunto de algoritmos que serão utilizados para comparação. Os processos 3, 4 e 5 são executados para cada um destes algoritmos definidos. A Figura 5.1 apresenta o método proposto.

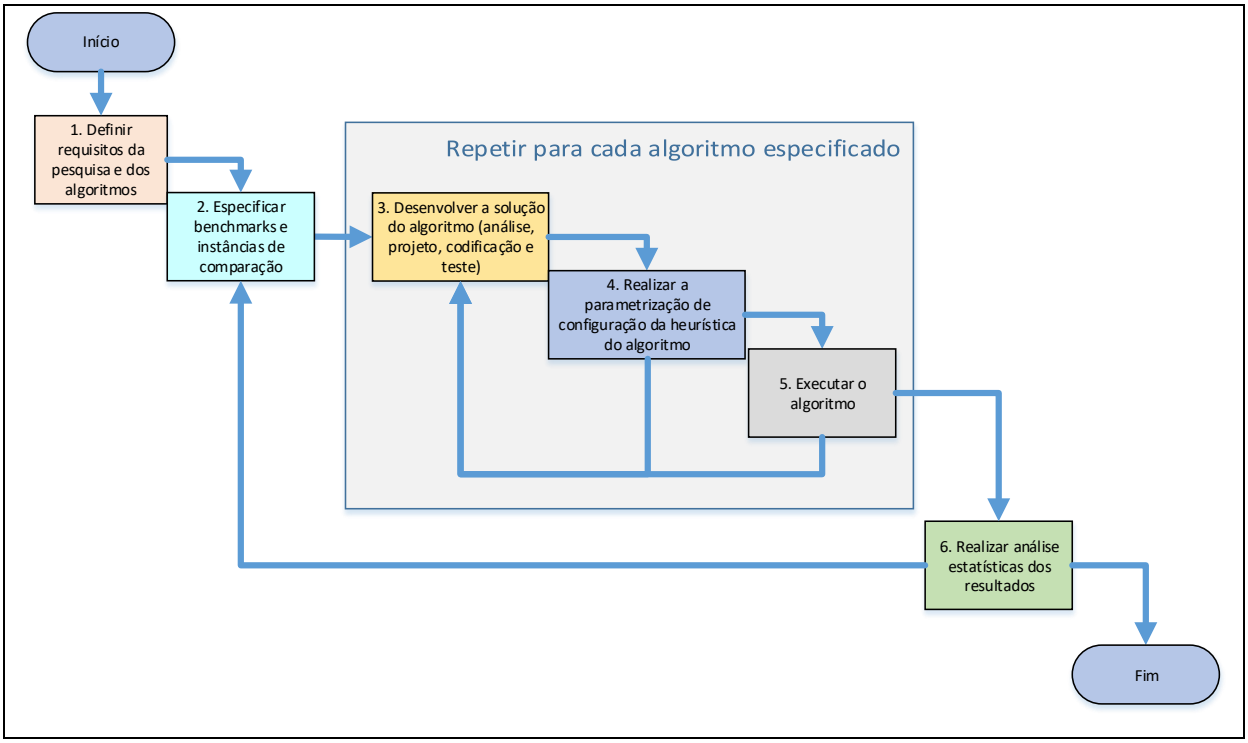


Figura 5.1 - Processo de desenvolvimento de pesquisa de otimizadores heurísticos
Fonte: elaborado pelo autor

5.2 Definir requisitos da pesquisa e dos algoritmos

A primeira etapa da pesquisa é justamente definir os requisitos completos do que será pesquisado. Esta etapa especifica as seguintes ações:

1. Definir a pesquisa - especifica por completo todas as informações necessárias para o entendimento da pesquisa que será realizada;
2. Definir o problema de engenharia de produção - identificar o problema de engenharia de produção tratado pela pesquisa e as características tais como: tipo do problema, fluxo de processo e função objetivo;
3. Especificar os algoritmos a serem comparados - especifica as informações necessárias para o entendimento dos algoritmos heurísticos que serão comparados na pesquisa.

A Figura 5.2 apresenta o processo de definição dos requisitos da pesquisa e dos algoritmos.

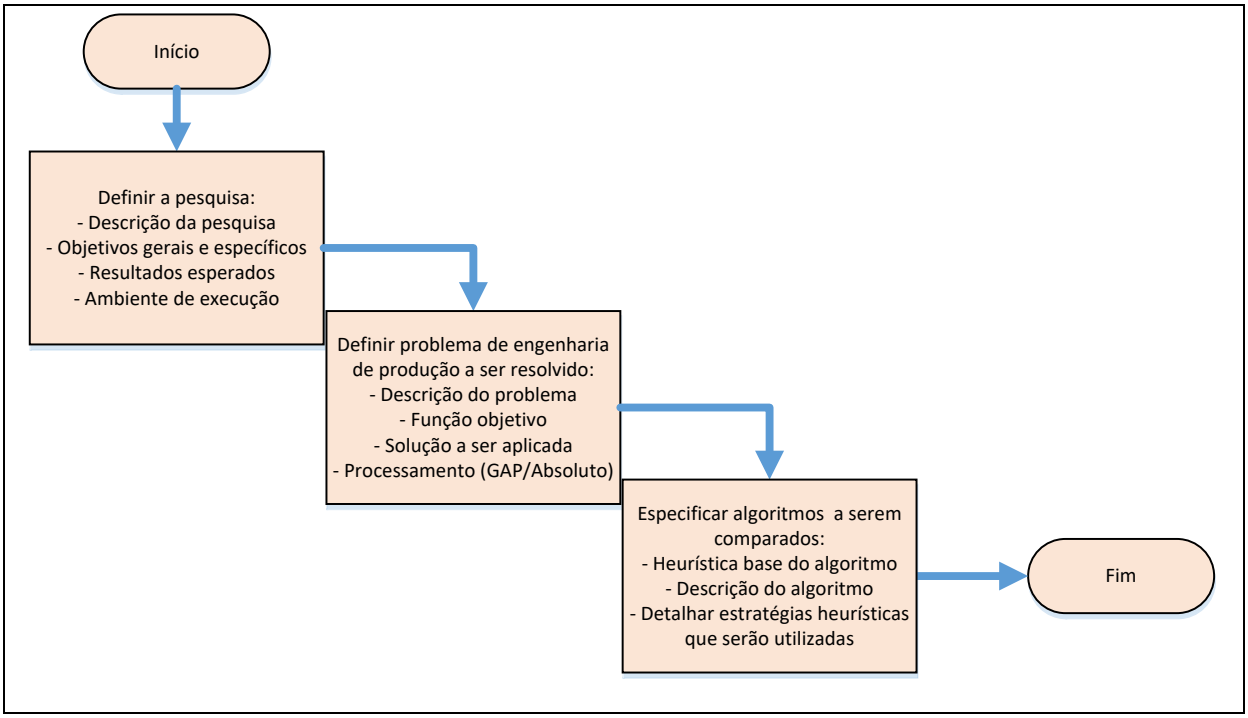


Figura 5.2 - Processo de definição dos requistos de pesquisa e dos algoritmos
Fonte: elaborado pelo autor

5.3 Especificar *benchmarks* e instâncias de comparação

Após as definições propostas no item anterior é necessário que o pesquisador especifique quais são as instâncias utilizadas na parametrização e na execução dos

algoritmos. O processo permite que seja selecionada uma biblioteca de instâncias publicada por outros autores como Taillard (1993), Demirkol et al. (1998) e Kolisch e Sprecher (1997) que definem *benchmarks* para problemas de *scheduling* com variações e mais recentemente, o trabalho de Vallada et al. (2015) que especifica grandes instâncias de *benchmark* para problemas de *flow-shop scheduling* com minimização de *makespan*. A grande vantagem de utilizar uma biblioteca já existente é que se tem acesso a valores ótimos obtidos para as instâncias, ou então, melhores valores factíveis obtidos por outras pesquisas. Isto facilita muito a comparação com os resultados que serão obtidos nas soluções heurísticas propostas.

O pesquisador pode ainda cadastrar sua própria biblioteca e seu conjunto de instâncias, por exemplo, gerada randomicamente a partir de especificações da pesquisa. Se for esta a opção do pesquisador, o mesmo deverá definir e especificar como os valores de comparação serão obtidos, através de um processo exato usando programação inteira ou programação inteira mista. Muitos trabalhos optam por esta solução mas, sempre esbarram no problema de obtenção de valores ótimos para as instâncias principalmente pela característica exponencial dos problemas de *scheduling*. Para pequenas instâncias a obtenção de valores ótimos é viável em termos de tempo de execução. Entretanto, para instâncias maiores os recursos computacionais e o tempo de processamento são proibitivos, sendo muitas vezes inviável. Em vários trabalhos da área, como por exemplo Rodriguez et al. (2013), o pesquisador executa um modelo de programação inteira, limitado por tempo de execução, para obter uma solução factível para a comparação. Pode-se ainda definir que o melhor valor obtido pelos algoritmos para cada instância é a referência de comparação com os outros. Outro procedimento é aplicar as soluções comparadas ao conjunto de instâncias criados obtendo os resultados de cada solução. Este procedimento pode ser visto nos trabalhos de Liao et al. (2014) e Arnaout et al. (2012).

Depois que a biblioteca de instâncias e as instâncias estão cadastradas o pesquisador deve definir o conjunto de instâncias que farão parte do *benchmark* de execução e definir o subconjunto de instâncias do *benchmark* de teste para ser usado na parametrização dos algoritmos. A Figura 5.3 apresenta o processo de especificação dos *benchmarks* e instâncias de comparação.

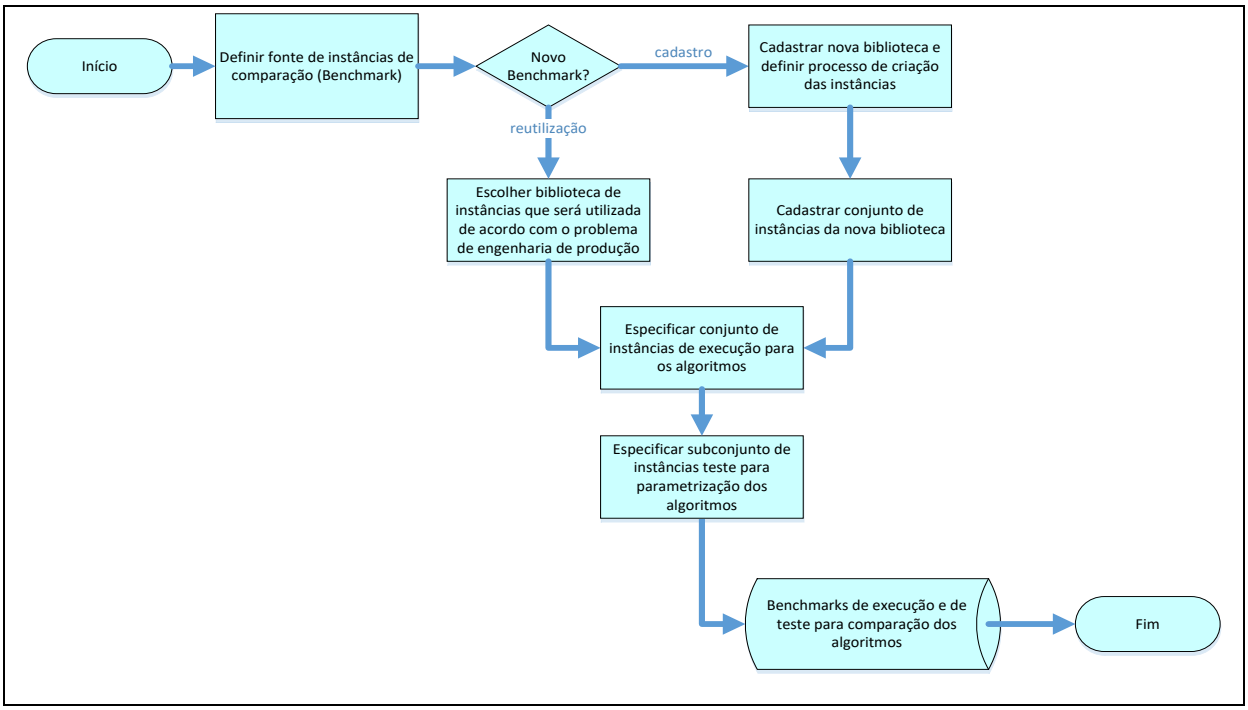


Figura 5.3 - Processo de especificação dos *benchmarks* e instâncias de comparação
Fonte: elaborado pelo autor

5.4 Desenvolver a solução do algoritmo (análise, projeto, codificação e teste)

A Figura 5.4 apresenta a proposta do processo de desenvolvimento da solução do algoritmo heurístico. Como o desenvolvimento de pesquisas na área muitas vezes acaba sendo um desenvolvimento solo, ou seja, realizado por um único pesquisador, foi pensado em um processo de desenvolvimento com entregas evolutivas, onde na solução proposta são agregadas novas funcionalidades. Este processo está subdividido em 3 fases principais, sendo:

- **Fase 1** - Construção do protótipo inicial contendo principalmente todas as funções básicas necessárias e a codificação da heurística encolhida como base. Nesta fase são desenvolvidas as funções básicas de alocação de memória, manipulação de vetores e matrizes, acesso a arquivos externos, tratamento de dados dentre outras e também é desenvolvida a estrutura de funcionamento da heurística base;
- **Fase 2** - Desenvolvimento das estratégias heurísticas especificadas contendo a especificação de todas as estratégias pensadas para agregar valor à heurística base, como por exemplo, funções de busca diferenciais ou inovadoras, novas estratégias evolucionárias dentre outras;

- **Fase 3** - Refinamento das estratégias heurísticas responsáveis por avaliar as respostas das estratégias heurísticas desenvolvidas através de testes mais aprofundados de forma a garantir a qualidade da solução desenvolvida.

Em cada fase do processo é seguido o fluxo de desenvolvimento de sistemas acompanhando as principais etapas de paradigmas de desenvolvimento de *software* (PRESSMAN, 2011; SOMMERVILLE, 2011). Percebe-se que dentro de cada fase existe um ciclo, que se for necessário, são realizadas especificações de novas funcionalidades até que a fase esteja concluída. Ao final de cada ciclo, todas as funcionalidades desenvolvidas são verificadas se estão de acordo com o que foi especificado e se estão funcionando perfeitamente. Somente após esta verificação é que se passa para a próxima fase. As etapas propostas para cada fase de desenvolvimento são:

- **Análise de requisitos** - engloba todas as tarefas que lidam com investigação, definição e escopo do sistema ou suas alterações, identificando assim as necessidades do que será desenvolvido;
- **Especificação do projeto** - se encarrega de transformar os resultados da análise de requisitos em um documento ou conjunto de documentos capazes de serem interpretados diretamente pelo programador na codificação do sistema. Para atingir este objetivo, o pesquisador deve mapear as estruturas e funcionalidades identificadas dentro do contexto e das restrições da arquitetura de forma a tornar possível a construção do *software*;
- **Implementação** - transformação das especificações de projeto elaborando e preparando os módulos necessários, em um código executável;
- **Teste de verificação** - nesta etapa o código desenvolvido é testado verificando assim a consistência, completitude, corretitude e confiabilidade do mesmo. São realizados testes de unidade de cada função ou módulo e teste de interação, ou seja, integração das funções ou módulos dos requisitos especificados. Se todos os requisitos definidos na fase estiverem completos e funcionando corretamente é passado para nova fase do processo. Se ainda existirem funcionalidades definidas nos requisitos que ainda não foram desenvolvidas o ciclo continua.

Para que a implementação da solução heurística seja realizada é necessário que seja definido pelo menos uma instância do problema e um conjunto de parâmetros base de configuração do algoritmo para execução dos testes de verificação.

Ao fim da Fase 3 de refinamento das estratégias heurísticas é realizado um teste de validação que tenta assegurar que o sistema final desenvolvido corresponda a todos os requisitos especificados na solução heurística. Se necessário, e a solução não estiver de acordo com a necessidade da pesquisa, o processo retorna para a fase de especificação das estratégias heurísticas de forma a implementar novas funcionalidades ou modificar as existentes.

Conforme a Figura 5.4, pode ser que o algoritmo definido como integrante do processo de comparação já tinha sido desenvolvido em pesquisas anteriores e o mesmo pode estar nas bases de dados do *framework* proposto ou deverá ser inserido.

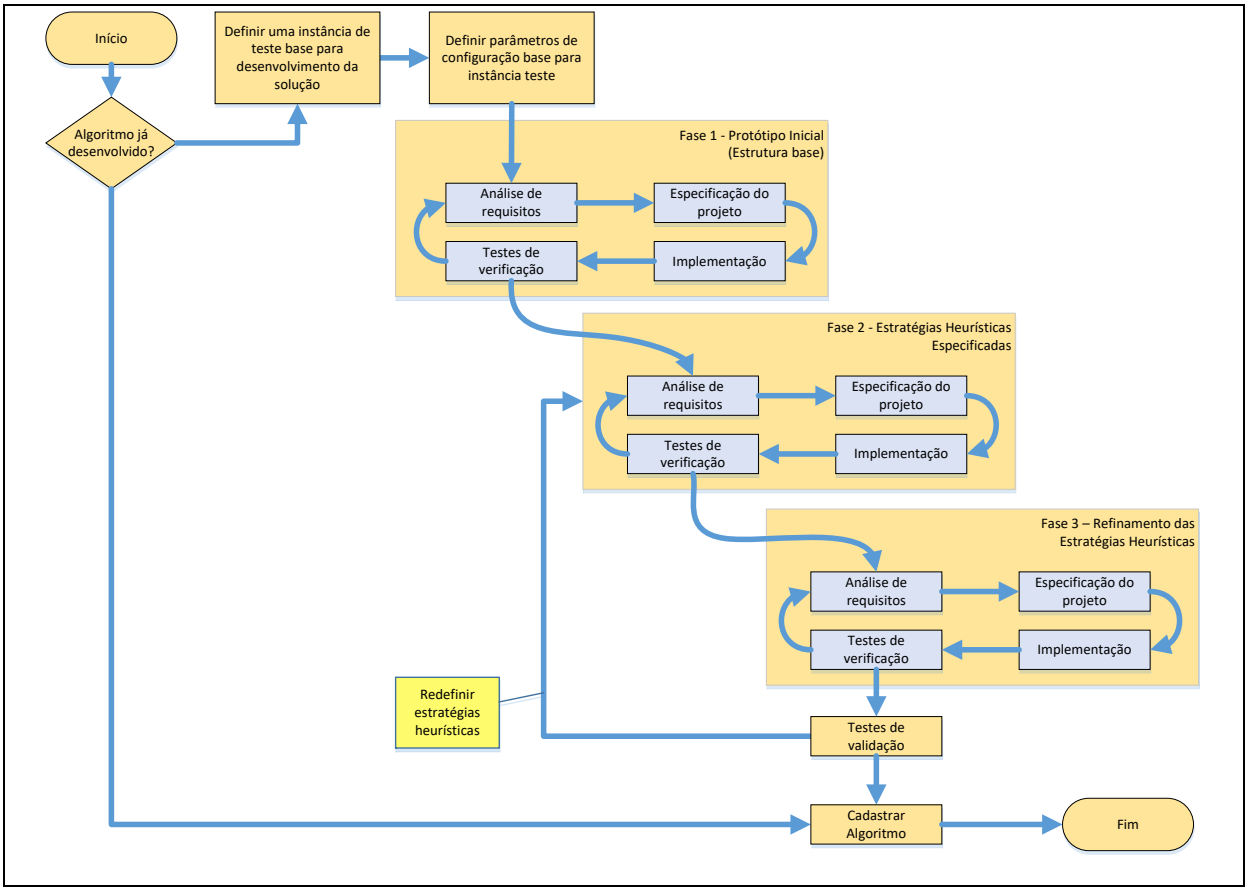


Figura 5.4 - Processo de desenvolvimento da solução do algoritmo
Fonte: elaborado pelo autor

5.5 Realizar a parametrização de configuração do algoritmo heurístico

O objetivo proposto nesta tese é fornecer um apoio ao pesquisador a um processo externo de parametrização, não sendo o foco definir um processo de parametrização de configuração automática conforme trabalhos proposto por Birattari et al. (2009) e Lopez-Ibanez et al. (2011). Neste sentido o pesquisador deve inicialmente cadastrar todos os parâmetros de configuração da solução heurística. Se os melhores valores de parâmetros já são conhecidos do pesquisador o mesmo efetua o registro para cada parâmetro cadastrado.

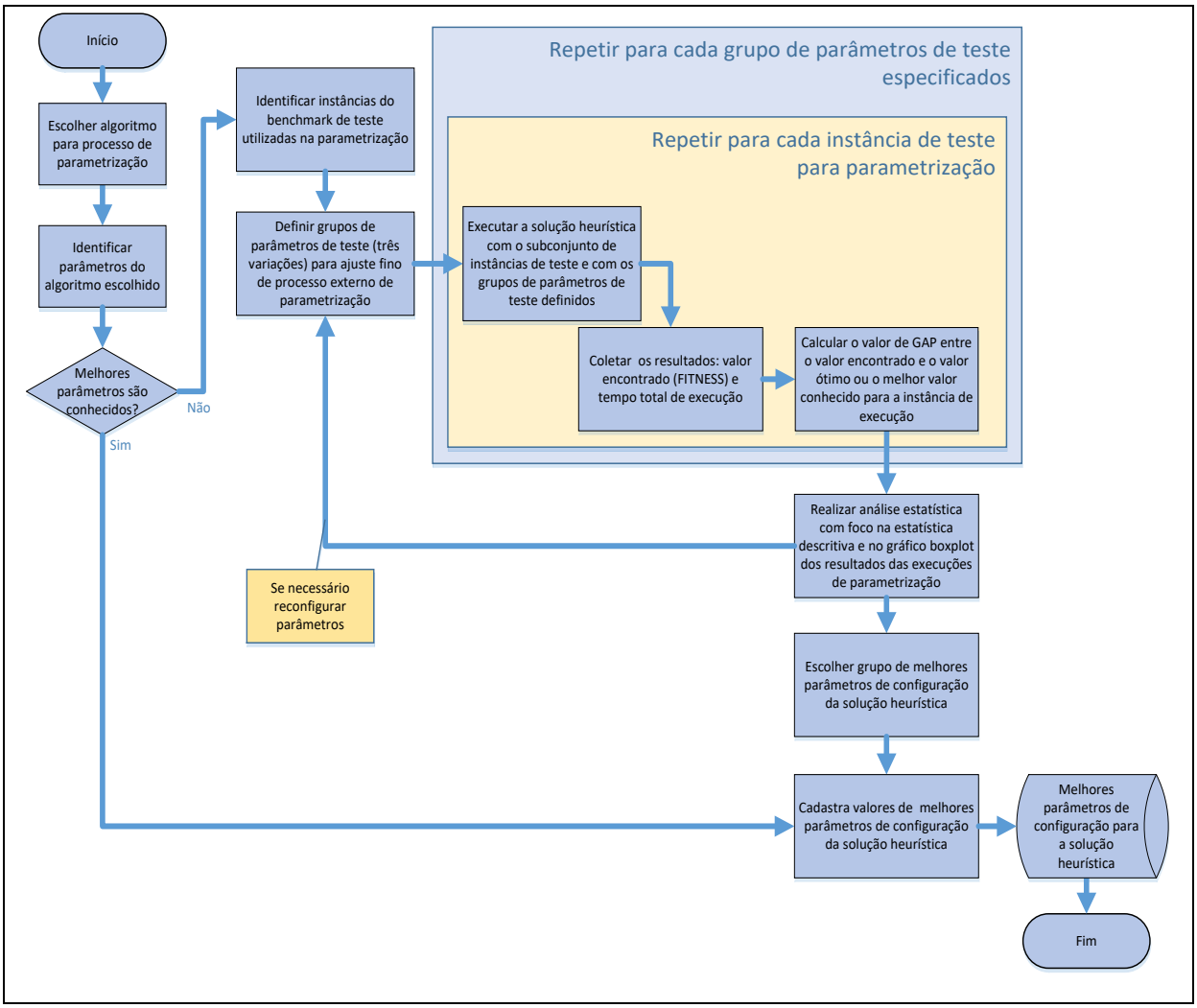


Figura 5.5 - Processo de parametrização de configuração do algoritmo heurístico
Fonte: elaborado pelo autor

O pesquisador pode ainda optar por fazer um processo simplificado de apoio à parametrização dos algoritmos. Com valores variados dos parâmetros obtidos por um

procedimento externo, o pesquisador define estes grupos de parâmetros no *framework* e solicita a parametrização. O *framework* executa o algoritmo para cada grupo de parâmetros definido e para cada instância do *benchmark* de teste especificado, calculando os valores de GAP ou absoluto com os melhores valores conhecidos para as instâncias. Após todos os valores das execuções serem obtidos, o *framework* realiza a análise estatística com os resultados obtidos com foco na estatística descritiva (média, desvio padrão, erro padrão médio, intervalo de interquartil, coeficiente variabilidade, *skewness*, *kurtosis*, moda, frequência da moda, mínimo, primeiro quartil, mediana, terceiro quartil e máximo) e no gráfico *boxplot*. Com estas informações o pesquisador seleciona qual grupo de parâmetros foi o melhor e registra os valores para cada parâmetro cadastrado. A Figura 5.5 apresenta o processo proposto.

5.6 Executar o algoritmo

Para que o processo de execução automática seja realizado são necessárias três etapas principais:

1. O arquivo executável referente ao algoritmo deve estar disponível para o *framework*;
2. O processo de parametrização estar concluído com as definições dos melhores valores dos parâmetros de execução;
3. As instâncias do *benchmark* de execução devem estar definidas.

Cumprida estas três etapas o pesquisador pode solicitar a execução dos algoritmos.

O *framework* seleciona um algoritmo dos que serão comparados e para cada instância do *benchmark* de execução especifica uma lista de argumentos de entrada de execução do algoritmo contendo o nome do arquivo da instância e todos os valores de parâmetros de configuração cadastrados e executa o algoritmo coletando e armazenando ao final os resultados obtidos (*fitness*) e o tempo total de execução. Com os valores de referência (valor ótimo ou melhor valor conhecido) da instância e os resultados obtidos são calculados os valores de GAP/RDP valores absolutos, sendo armazenado no *framework*. A Figura 5.6 apresenta o processo proposto.

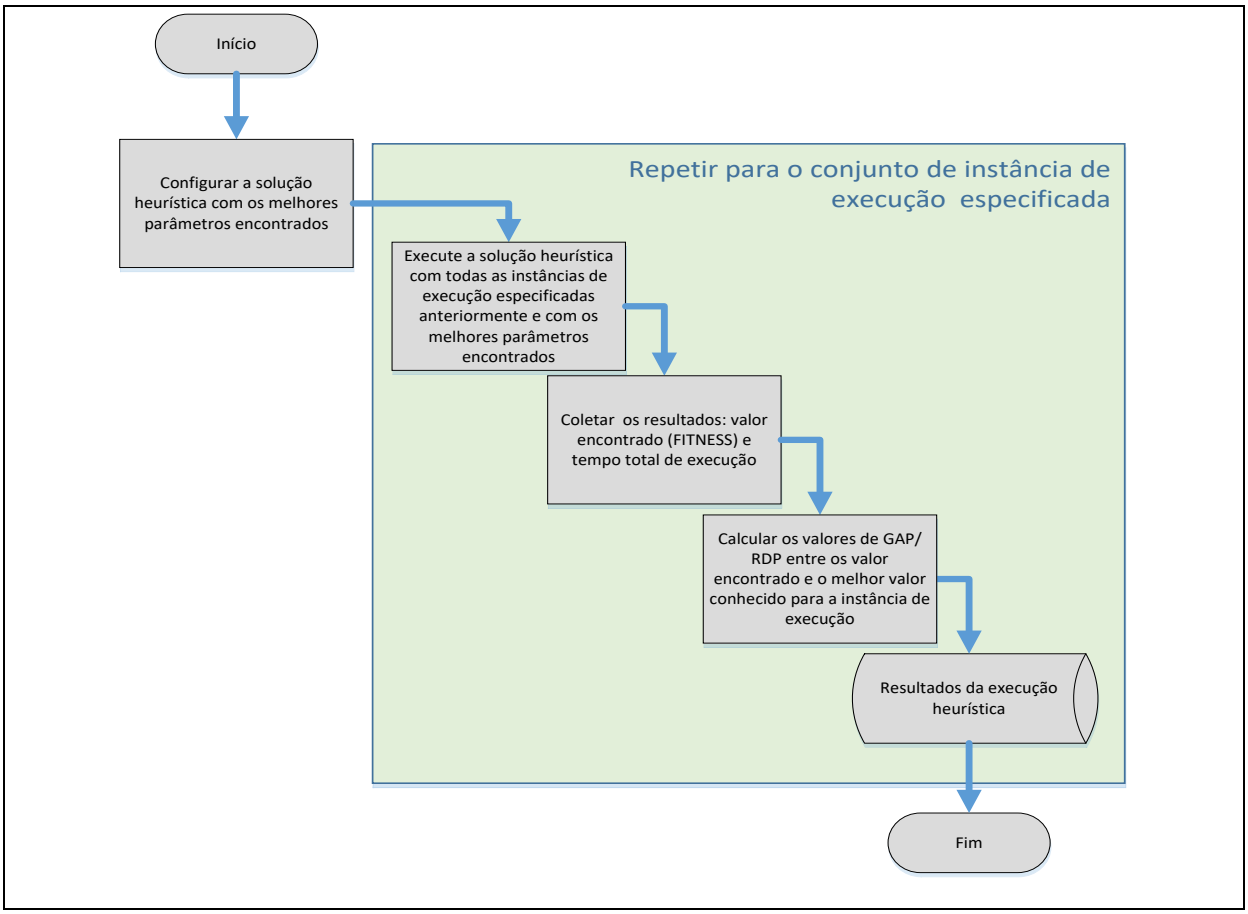


Figura 5.6 - Processo de execução do algoritmo e coleta dos resultados
Fonte: elaborado pelo autor

5.7 Realizar análise estatística e reportar os resultados

O objeto de estudo desta tese considera as seguintes restrições em relação à escolha dos testes estatísticos aplicados na análise.

- As análises estatísticas são baseadas em estudos comparativos entre resultados obtidos através da execução das soluções heurística em um conjunto de instâncias do problema;
- Todos os valores obtidos através das medições são ordinais;
- Considera-se que as amostras são dependentes, ou seja, todas as soluções heurísticas utilizadas na comparação são executadas com um mesmo conjunto de instâncias do problema.

Neste sentido o processo relativo à realização da análise estatística juntamente com a elaboração do relatório de resultados foi definido em 5 etapas principais, sendo:

1. Realizar análise da estatística descritiva das amostras;
2. Realizar análise gráfica das amostras;

3. Realizar análise de normalidade das amostras;
4. Realizar análise comparativa das amostras;
5. Executar análise da estatística inferencial das amostras.

A Figura 5.7 apresenta o processo proposto já com as restrições listadas acima.

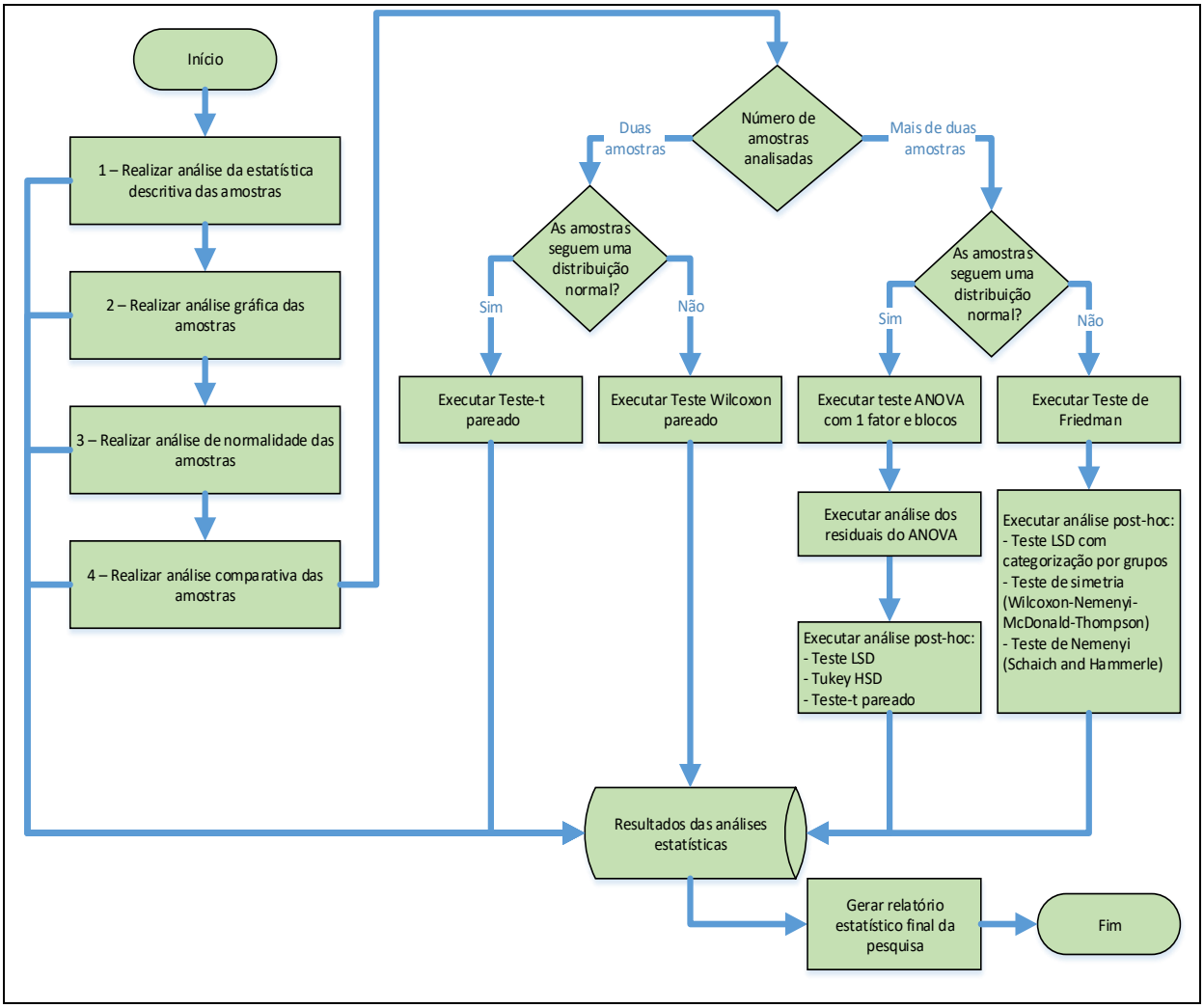


Figura 5.7 - Processo de análise estatística e elaboração do relatório de resultados
Fonte: elaborado pelo autor

5.7.1 Realizar análise da estatística descritiva das amostras

Nesta etapa são calculados os indicadores relativos a estatística descritiva tais como: média, desvio padrão (sd), erro padrão (média), intervalo de interquartil, coeficiente variabilidade (média/sd), *skewness* (simetria dos dados), *kurtosis* (grau de concentração dos dados em relação a média), moda, frequência da moda, valor mínimo, primeiro quartil, mediana, terceiro quartil, valor máximo. Recomenda-se que os valores calculados para cada um dos algoritmos tratados na pesquisa, sejam

apresentados em forma de tabela para facilitar a visualização e comparação dos resultados.

5.7.2 Realizar análise gráfica das amostras

Uma das formas de avaliar os resultados da análise estatística de dados é por meio de gráficos. Neste sentido a proposta é a construção dos seguintes gráficos:

- ***Boxplot* das amostras com e sem *outliers*** - permite uma comparação da variabilidade dos dados e valores atípicos que podem influenciar o cálculo de outras medidas;
- **Gráfico de moda e frequência da moda** - ajuda o pesquisador na identificação visual do valor que ocorre com maior frequência em um conjunto de dados. O objetivo é verificar se existe um domínio da concentração da variável estudada no valor modal para cada amostra;
- **Gráfico dos valores de *fitness* por instâncias** - fornece uma visualização do comportamento de cada solução algorítmica em relação ao conjunto de instâncias utilizadas;
- **Gráfico de coordenadas paralelas** - usado para comparar os dados de tipos ou magnitudes diferentes em uma única visualização. Um gráfico de coordenadas paralelas considera que cada linha da tabela de dados é uma linha ou perfil que é representado no gráfico. Como informações adicionais podem ser apresentadas os valores de média e mediana dos dados;
- **Gráfico de dispersão com limites de erro padrão e média** - apresenta em cada coluna uma visão da dispersão dos valores de *fitness* de cada algoritmo com os valores limitantes do erro padrão e a média dos valores.

5.7.3 Realizar análise de normalidade das amostras

Os testes de normalidade são usados para determinar se um conjunto de dados de uma dada variável aleatória pode ser representado por uma distribuição normal ou não. O teste calcula a probabilidade da variável aleatória subjacente estar normalmente distribuída.

- **Calcular os testes de normalidade** - são os testes de Shapiro-Wilk (SHAPIRO; WILK, 1965) e Lilliefors-Kolmogorov-Smirnov (LILLIEFORS,

1967). É interessante que os valores calculados para os testes de normalidade para cada um dos algoritmos tratados na pesquisa sejam apresentados em forma de tabela para facilitar a visualização e comparação dos resultados;

- **Gráficos de histogramas das amostras** - é uma representação gráfica, de um conjunto de dados previamente tabulado e dividido em classes uniformes. A construção de histogramas é de caráter preliminar em qualquer estudo sendo um importante indicador da distribuição de dados em relação ao comportamento dos dados;
- **Gráficos de comparação de quartil (QQplot) das amostras** - também pode ser utilizado para determinar se um conjunto de dados pertence a distribuição normal de probabilidade sendo mais apropriado quando forem poucos dados analisados. Em tais gráficos, os pontos são formados pelos quartis amostrais e se no resultado os pontos (dados da amostra) alinham-se em relação à reta apresentada a distribuição da amostra pode ser consideradas como sendo normal.

5.7.4 Realizar análise comparativa das amostras

Como o processo de análise inferencial de comparação entre as amostras é realizada entre pares, ter uma informação contendo as diferenças entre as soluções analisadas pode facilitar conclusões da pesquisa. A proposta é apresentar um gráfico *boxplot* para as diferenças entre todos os pares das amostras analisadas, permitindo uma análise mais detalhada da comparação. O *boxplot* das diferenças deve ser apresentado com e sem *outliers* permitindo uma análise destes valores extremos nos dados.

5.7.5 Executar análise da estatística inferencial das amostras

Seguindo as restrições do presente trabalho, onde a análise se refere a pesquisas comparativas entre duas ou mais amostras e que os dados das amostras são dependentes por serem obtidos a partir de um mesmo conjunto de instâncias de teste, a análise estatística inferencial das amostras fica dividida entre dois grupos principais: (i) análise da estatística inferencial para duas amostras e (ii) análise da estatística inferencial para três ou mais amostras.

Se a pesquisa que está sendo realizada com somente duas amostras, independentemente se o teste de normalidade indicar que as amostras seguem uma distribuição normal, podem ser executados os testes: (i) teste paramétrico Teste-t pareado para amostras independentes e (ii) testes não paramétrico de Wilcoxon pareado para amostras independentes, dependendo somente da definição do parâmetro de normalidade feito pelo pesquisador. Deve ficar claro que se o pesquisador tiver definido que os dados de entrada seguem uma distribuição normal será executado o Test-t com as amostras. Se o teste de normalidade falhar os resultados podem ficar enviesados, o mesmo acontecendo para o teste de Wilcoxon. Tanto para o Teste-t como para o Wilcoxon, as hipóteses alternativas são testadas para valores iguais, maiores e menores que zero fornecendo uma escolha de análise para o pesquisador.

Se a pesquisa que está sendo realizada tiver mais de duas amostras (limitada a 10 amostras), independentemente se o teste de normalidade indicar que as amostras seguem uma distribuição normal, podem ser executados os testes: (i) teste paramétrico ANOVA de 1 fator com blocos e (ii) teste não paramétrico de Friedman, dependendo somente da definição do parâmetro de normalidade feito pelo pesquisador. Para o teste ANOVA são calculados os resultados gerais que apresentam principalmente os p-valores encontrados e são gerados os gráficos dos valores residuais para análise. Para o teste de Friedman são calculados os resultados gerais que contém o p-valor do teste.

Tanto para o teste ANOVA como o teste de Friedman, se a hipótese nula for rejeitada, conclui-se que pelo menos uma das amostras analisadas na comparação é diferente das outras. Para obter informações mais detalhadas, como por exemplo, qual amostra apresenta melhores resultados, é necessário fazer uma análise *post-hoc*. De acordo com a disponibilidade de pacotes da linguagem R disponíveis para utilização foram definidos os seguintes testes de análise *post-hoc*:

- Teste ANOVA com um fator e blocos
    I. **Teste de Fisher´s LSD com categorização por grupos** - calcula a menor diferença significante (LSD) entre duas médias como se estas médias fossem as únicas a serem comparadas (como no teste-t) e declara significante qualquer diferença maior que o LSD. As amostras são categorizadas em grupos a partir destas diferenças com o valor LSD calculadas;

II. **Teste de Tukey´s HSD** - cria um conjunto de intervalos de confiança sobre as diferenças entre as médias dos valores dos níveis de um fator com a probabilidade de cobertura do agrupamento especificado. Os intervalos são baseados na estatística de faixa que acompanha a distribuição t-Student e no método proposto por Tukey´s (*Honest Significant Difference* - Diferença Significante Honesta);

III. **Teste-t pareado** - executa inúmeros testes-t entre os pares e utiliza um método de correção do erro que neste caso é o método de correção de Bonferroni.

- Teste de Friedman

I. **Teste de Fisher´s LSD com categorização por grupos** - o funcionamento é o mesmo descrito para o Fisher´s LSD para o teste ANOVA;

II. **Teste de simetria (Wilcoxon-Nemenyi-McDonald-Thompson)** - executa testes utilizando uma abordagem baseada em permutações nas comparações entre os pares. Tem como objetivo central o controle da taxa de erro de agrupamento (FWER - F*amilyWise Error Rate*). O resultado final é conseguido através da comparação de cada teste estatístico observado com o valor máximo das estatísticas permutadas sobre todas as variáveis. Em outras palavras, o p-valor reflete a chance de se conseguir uma estatística de teste que apresente valores maiores quanto a observada, dado que foram realizados inúmeros testes;

III. **Teste de Nemenyi (Schaich and Hammerle)** - executa testes entre pares para múltiplas comparações das somas dos ranques das médias para dados em blocos não replicados. A estatística do teste refere-se ao quartil superior da estatística de faixa que acompanha a distribuição de t-Student.

5.7.6 Gerar relatório estatístico final da pesquisa

Após a execução de todos os testes estatísticos descritos acima o *script* constrói o relatório final. O Quadro 5.1 apresenta a estrutura deste relatório que é gerado automaticamente, utilizando o padrão RTF (*Reatch Text Format*) criando o

arquivo de saída "ResultadosGerais-<<Rótulo da Pesquisa>>.doc" na pasta de resultados da pesquisa.

Quadro 5.1 - Estrutura do relatório estatístico gerado automaticamente

Cabeçalho com informações sobre a pesquisa
1. Resultados da estatística descritiva dos dados
2. Análise gráfica das amostras
   - 2.1. *Boxplot* das amostras com e sem *outliers*
   - 2.2. Gráfico da moda e da frequência da moda
   - 2.3. Gráfico dos valores de *fitness* por instâncias de cada algoritmo
   - 2.4. Gráfico de coordenadas paralelas das amostras com média e mediana
   - 2.5. Gráfico de dispersão com limites de erro padrão e média
3. Análise de normalidade das amostras
   - 3.1. Teste de normalidade das amostras
   - 3.2. Histogramas das amostras
   - 3.3. Gráfico de comparação quartil-quartil das amostras
4. Análise gráfica comparativa entre as amostras

Para a definição de uma pesquisa que efetua comparações com dois algoritmos, tem-se:
5. Teste-t pareado
6. Teste Wilcoxon pareado
   - 6.1. Teste de sinal bilateral
   - 6.2. Teste Wilcoxon pareado com somatório dos ranqueamentos

Para a definição de uma pesquisa que efetua comparações entre três ou mais algoritmos, tem-se:
5. Teste ANOVA com um fator e blocos e análise *post-hoc*
   - 5.1. Resultado geral do ANOVA com um fator e blocos
   - 5.2. Análise dos residuais do ANOVA com um fator
   - 5.3. Análise *Post-hoc* utilizando teste Fisher´s LSD
   - 5.4. Análise *Post-hoc* utilizando Tukey HSD
   - 5.5. Análise *Post-hoc* utilizando Teste-t pareado
6. Teste de Friedman com análise *post-hoc*
   - 6.1. Resultado geral do teste de Friedman
   - 6.2. Análise *Post-hoc* utilizando Teste LSD
   - 6.3. Análise *Post-hoc* utilizando teste de simetria
   - 6.4. Análise *Post-hoc* utilizando teste de Nemenyi

Referências

Fonte: elaborado pelo autor

Complementando o relatório final, todas as tabelas de resultados dos testes estatísticos são gravadas em arquivos no formato de planilha XLSX (arquivos de planilha do *software* Microsoft Excel) e em formato TXT (arquivo texto). Todos os gráficos criados são também gravados nos formatos PNG (*Portable Network Graphics* - arquivo de dados de imagens) e EPS (*Encapsulated Postscript* - arquivos do tipo *Postscript* em formato vetorial padrão para editoração eletrônica) para futuras consultas, facilitando assim o processo de formatação e utilização em trabalhos científicos. As imagens multigráficos, tais como os gráficos de histograma e QQplot, também são gravadas individualmente em pastas respectivas tanto no formato PNG como EPS.

5.8 Consideração a respeito do processo de desenvolvimento de pesquisa de otimização baseado em heurísticas

Este capítulo apresentou uma proposta para um processo de desenvolvimento de pesquisas de otimizadores baseados em heurísticas. Foram definidos todos os processos necessários para que uma pesquisa nesta área possa ser executada, desde a definição dos requisitos da pesquisa, a especificação dos *benchmarks* de instâncias de comparação, o desenvolvimento das soluções heurísticas comparadas, a parametrização destas soluções, a execução e coleta e resultados e principalmente a análise estatística dos resultados. O ponto principal do processo é a análise estatística dos resultados de pesquisa que especifica um conjunto de etapas a serem seguidas juntamente com os procedimentos estatísticos a serem calculados direcionando na padronização de um relatório estatístico final capaz de apoiar as pesquisas da área.

Capítulo 6

O FRAMEWORK DESENVOLVIDO

Este capítulo apresenta o processo de desenvolvimento que foi utilizado no *framework* com todas as especificações dos requisitos de desenvolvimento, a análise do sistema, as iterações de construção implementadas, incluindo a especificação do *script* de análise estatística dos resultados encontrados, o gerenciamento dos cadastros, as execuções automáticas, o processo de parametrização simplificado dos parâmetros de execução, os testes de integração e de sistemas e os testes de aplicação do *framework* com dados reais.

6.1 Processo de desenvolvimento do *framework*

O processo de engenharia de *software* contribui em muito para a estrutura do sistema que será desenvolvido fornecendo um roteiro eficaz para o desenvolvedor. Vários autores apresentam modelos de processo, muitos deles baseados em modelos mais tradicionais, tais como modelo cascata, modelo de prototipação, modelo espiral e muitos outros. Existem também metodologias que vão além de somente um processo, chegando a especificar um modelo completo ou um *framework* de desenvolvimento, tais como CMMI (*Capability Maturity Model* - Integration ou Modelo de Maturidade em Capacitação - Integração) ou RAD (*Rapid Aplication Development* - Desenvolvimento Rápido de Aplicações) dentre outros. O interessante é que o desenvolvimento de um sistema em ambientes reais dificilmente conseguirá seguir um modelo pré-definido, devido às próprias características do sistema, fazendo com que muitas vezes um desenvolvedor, uma equipe ou mesmo uma empresa faça adaptações de modelos disponíveis no mercado de forma a ajusta-lo à sua realidade (PRESSMAN, 2011).

O modelo cascata, definido na Figura 6.1, é o modelo mais tradicional disponíveis na literatura, sugerindo uma abordagem sequencial e sistemática, na qual cada etapa do processo somente inicia quando a etapa anterior for concluída. Este modelo apresneta problemas para ser seguido na íntegra principalmente porque os requisitos de um sistema dificilmente são conhecidos por completo no início do projeto. Neste sentido, muitas variações deste modelo foram propostas, como exemplo o modelo em V, que descreve a relação das etapas do processo cascata com ações de garantia de qualidade, tais como testes sistemáticos (PRESSMAN, 2011).

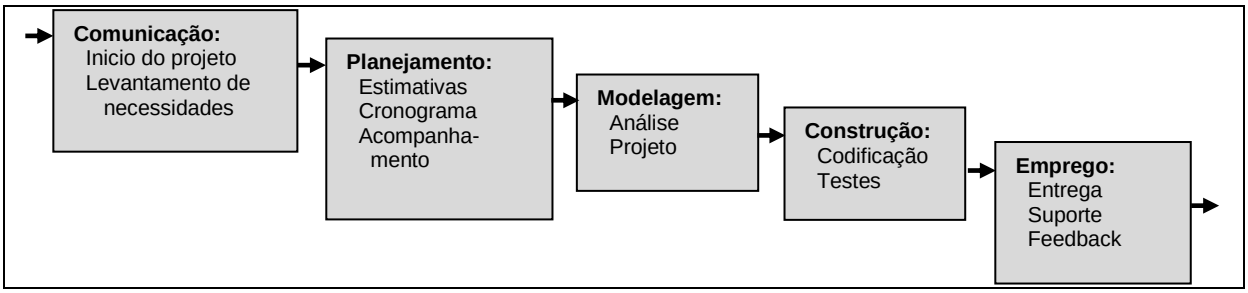


Figura 6.1 - Modelo cascata de desenvolvimento de *software*
Fonte: (PRESSMAN, 2011)

Uma das variações propostas tanto por Sommerville (2011) e Pressman (2011) é o processo iterativo incremental na qual os requisitos iniciais do projeto não são completamente conhecidos e a utilização de um processo linear não seria muito bem aceita. Pode ser necessária a apresentação rápida de um determinado conjunto de funcionalidades para somente depois ser desenvolvido o refinamento e expansão desta funcionalidade em versões posteriores da aplicação. Neste caso um modelo que desenvolva o sistema de forma incremental seria mais bem aceito. O modelo incremental é capaz de combinar elementos do fluxo de processos lineares e paralelos.

A ideia definida no modelo é que, a cada novo incremento, mais requisitos do sistema tenham sido implementados. Cada incremento irá depender do conjunto de requisitos que devem ser desenvolvidos e da necessidade de já ter um protótipo para validação. A Figura 6.2 apresenta a estrutura do modelo incremental. Neste modelo a cada incremento é seguido um ciclo de desenvolvimento cascata completo, desde a etapa de comunicação até a etapa de emprego. Novamente, o que se tem no mercado muitas vezes são adaptações deste modelo para adequação da realidade do ambiente de desenvolvimento.

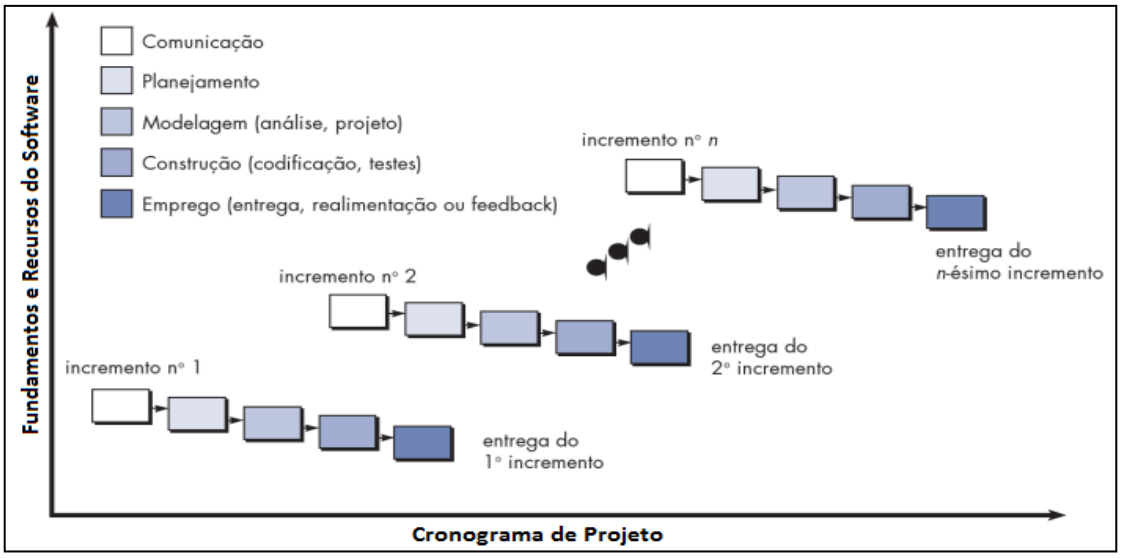


Figura 6.2 - Modelo incremental de processo de *software*
Fonte: (PRESSMAN, 2011)

Com base no modelo incremental visto acima, e pelo *framework* proposto por esta tese ser desenvolvido por somente um pesquisador, definiu-se um modelo de desenvolvimento que melhor se adequa à realidade de construção de uma tese de doutorado. Neste sentido, têm-se as seguintes etapas:

I. Levantamento e especificação dos requisitos gerais e regras de negócios completas do *framework;*
II. Análise do sistema - especificação dos requisitos gerais com detalhamento do que será desenvolvido. Nesta etapa serão desenvolvidos os documentos de casos de uso na especificação dos requisitos, o modelo conceitual (diagrama de classe simplificado) e os diagramas de sequência com foco na troca de mensagens entre os objetos identificados;
III. Análise (A) - enfatiza a investigação do problema e de novos requisitos da iteração assim como a solução esperada;
IV. Projeto (P) - enfatiza a solução conceitual que resolve por completo os requisitos definidos na iteração. Nesta etapa serão desenvolvidos o diagrama de classe completo, diagrama de pacotes, refinamento dos diagramas de sequência e diagrama de entidade e relacionamento;
V. Codificação (C) - efetua a transformação das especificações desenvolvidas na análise e no projeto em um código executável;
VI. Teste unitário (T) - realiza os testes de cada unidade desenvolvida, podendo ser um requisito completo ou parte do mesmo, garantindo que esta unidade esteja funcionando perfeitamente;

VII. Teste de integração e de sistema do *framework* - realiza testes de forma integrada, incluindo todas as funcionalidades do sistema que foram desenvolvidas, garantindo que o sistema esteja pronto e funcionando perfeitamente;

VIII. Aplicação do *framework* - utilização do *framework* desenvolvido com soluções heurísticas de forma a legitimar a utilização do mesmo na validação dos resultados de pesquisas, conforme já definido anteriormente.

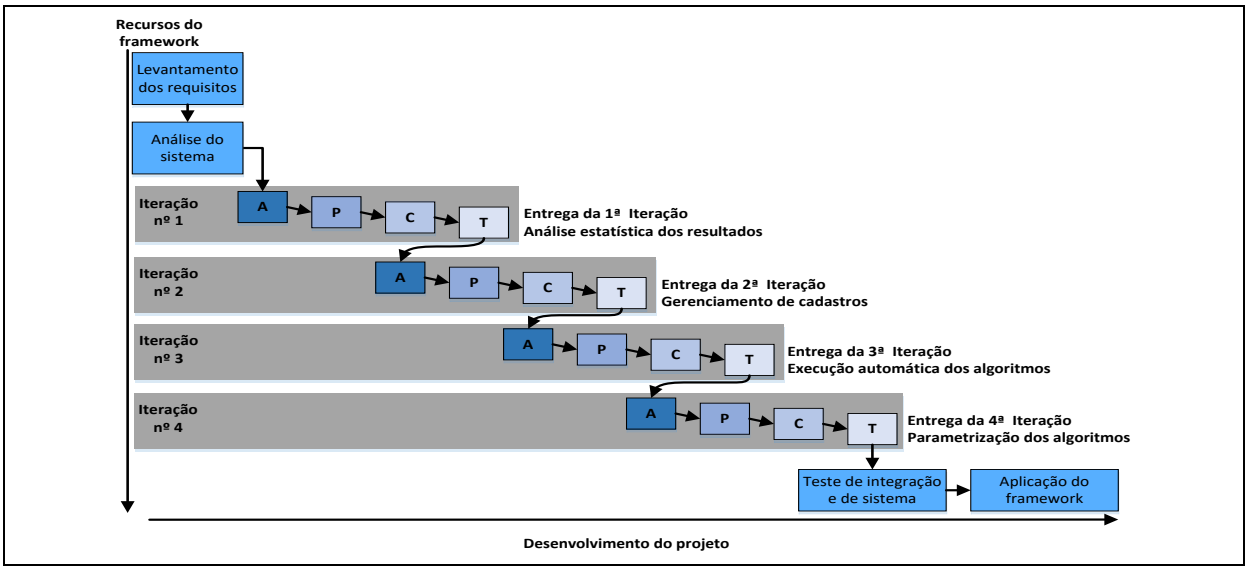


Figura 6.3 - Processo de desenvolvimento para o *framework*
Fonte: adaptado pelo autor baseado em (PRESSMAN, 2011) e (SOMMERVILLE, 2011)

A Figura 6.3 apresenta o processo de desenvolvimento para o *framework* deste trabalho. Conforme dito anteriormente, o processo está baseado no modelo iterativo incremental e foi dividido em quatro (4) iterações, baseadas na estrutura do *framework* apresentado na Figura 1.3. como segue:

- **Iteração 1** - Desenvolvimento do *script* de análise estatística dos resultados utilizando programação em linguagem R (R DEVELOPMENT CORE TEAM, 2014). O *script* roda a partir dos resultados de pesquisa formatados em um arquivo texto de entrada. São passados ainda como parâmetros o número de algoritmos comprados, uma especificação se os resultados encontrados seguem uma distribuição normal e uma especificação se os dados são dependentes entre si. Ao final da execução o *script* gera o relatório estatístico com todas as análises já definidas;
- **Iteração 2** - Desenvolvimento dos gerenciamentos dos cadastros, ou seja, o cadastro de todos os dados necessários para o funcionamento do

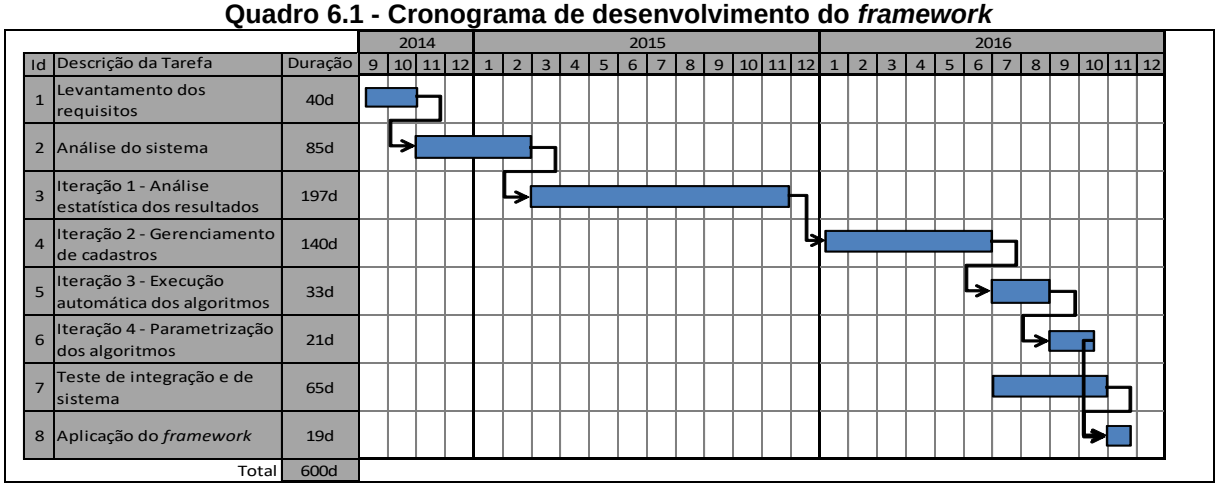
framework. Foram desenvolvidos nesta iteração o controle de cadastro gerenciamento de pesquisas, problemas de engenharia de produção, algoritmos que serão comparados na pesquisa, os parâmetros de execução dos algoritmos, usuários, bibliotecas de instâncias e instâncias. Para cada cadastro são implementadas as operações básicas de criação, consulta, atualização e deleção de manipulação dos objetos;

- **Iteração 3** - Desenvolvimento da execução automática dos algoritmos, ou seja, é a execução de todos os algoritmos cadastrados na pesquisa com todas as instâncias de execução definidas. Todos os resultados encontrados são armazenados no *framework* e ao final da execução é gerado o arquivo de resultados encontrados para cada instância, que é utilizado na análise estatística desenvolvida na iteração 1. Outro arquivo gerado ao final da execução é de tempo total de execução de cada algoritmo por cada instância que também pode ser uma informação importante da pesquisa a ser analisada. Nesta iteração também foi desenvolvido a chamada do *script* para realização da análise estatística de resultados;
- **Iteração 4** - Desenvolvimento do processo simplificado de parametrização com definição de grupos de teste de parâmetros como apoio a um processo externo de parametrização. Esta iteração executa um algoritmo escolhido com cada uma das instâncias definidas no *benchmark* de teste.

O cronograma completo de desenvolvimento do *framework* com todas as etapas e iterações definidas na Figura 6.3 é apresentado no Quadro 6.1 abaixo. O quadro apresenta a descrição da tarefa, a duração total em dias além de apresentar uma estimativa do início e término de cada tarefa. Percebe-se que a tarefa 3 (Iteração 1 - Análises Estatística dos Resultados) e a tarefa 4 (Iteração 2 - Gerenciamento dos Cadastros) foram as que gastaram mais dias de desenvolvimento, sendo cento e noventa e sete (197) dias e cento e quarenta (140) dias respectivamente. Isto ocorreu porque nestas tarefas grande parte do tempo foi gasto com o aprendizado da plataforma tecnológica envolvida. No total, por estimativa, foram gastos seiscentos (600) dias para o desenvolvimento do *framework*. Considerando que os dias trabalhos no desenvolvimento foram de segunda a sexta-feira, descontados feriados e recessos, que a quantidade de horas por dia dedicadas exclusivamente ao desenvolvimento em média foi de três (3) horas por dia e que o *framework* foi desenvolvido por somente

um recurso, pode-se chegar ao esforço de desenvolvimento que foi de um mil e oitocentas (1800) horas de desenvolvimento.

Quadro 6.1 - Cronograma de desenvolvimento do *framework*



Fonte: elaborado pelo autor

6.2 Tecnologia de desenvolvimento utilizadas

Tomou-se como princípio nas decisões de projeto de desenvolvimento do *framework* a utilização de ferramentas e aplicativos gratuitos e de código aberto sob a licença publica geral GNU-GPL (*General Public License*) (LICENSE, 1989), principalmente para não gerar custos nesta etapa e com o intuito de disponibilizar para utilização da comunidade de pesquisadores da área.

O *framework* foi desenvolvido usando a linguagem de programação PYTHON 2.6[14]. O Python é uma linguagem orientação a objetos (MCLAUGHLIN et al., 2007), livre de código aberto sobre a licença Python *Software* Foundation License, poderosa, multiplataforma, de alto nível, de desenvolvimento rápido, que segue técnicas de não repetição de códigos e com o intuito de manter a codificação simples (PYTHON *SOFTWARE* FOUNDATION, 2013).

Como o intuito era desenvolver o *framework* para ser utilizado na web, sendo disponibilizado para outros pesquisadores, foi utilizado o *framework* web DJANGO 1.6[15]. O Django é o principal *framework web* em Python, livre, de código aberto, com uma comunidade ativa e poderosa que fornece soluções e componentes diversos e um suporte completo apoiado pela comunidade desenvolvedora. Possui mapeamento objeto-relacional (ORM - *Object-relational mapping*) que agiliza muito o

[14] https://www.python.org/
[15] https://www.djangoproject.com/

desenvolvimento a partir do modelo relacional do sistema e fornece uma excelente estrutura base para projetos, podendo organizar com aplicativos facilitando a integração entre projetos e aplicativos de terceiros (DJANGO *SOFTWARE* FOUNDATION, 2013). Todo o desenvolvimento em Python/Django foi codificada no projeto Eclipse[16] mantido pela Oracle, com plugin PyDev[17] para desenvolvimento em Python. Os templates em Html do *framework* foram desenvolvidos usando o Notepad++[18].

Para desenvolvimento do banco de dados do *framework* foi escolhido o MYSQL Server 5.6[19] com o MySql Workbench 6.2[20] para administração, modelagem dos diagramas, criação e manutenção da base de dados. O MySql também é um *software* livre e de código aberto muito utilizado por grandes corporações, portável, com compatibilidade com diversas linguagens inclusive o Python, com excelente desempenho e estabilidade e que exigente poucos recursos para execução.

O *script* de análise estatística dos resultados foi desenvolvido na linguagem R[21] (R DEVELOPMENT CORE TEAM, 2014) principalmente pela facilidade de codificação e pela integração com outras plataformas. O R também é um *software* livre de código aberto sob a licença publica geral GNU-GPL (*General Public License*) (LICENSE, 1989). O R fornece um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos diversos, sendo capaz de executar métodos estatísticos complexos da mesma forma que ferramentas proprietárias disponíveis no mercado. Toda a codificação desenvolvida em R foi feita no editor Tinn-R[22].

Para a especificação dos documentos de análise e projeto foi utilizado UML (*Unifield Modeling Language* - Linguagem Unificada de Modelagem) (BOOCH et al., 2006 e ERIKSSON et al., 2003) com o *framework* StarUml[23].

Foi dado ao *framework* o nome de HOSDA (*Heuristic Optimization with Statistical Data Analysis - Framework* para otimização heurística com análise de dados estatística) e o mesmo foi registrado[24] com o endereço web www.hosda.com.br.

O *framework* foi implantado em uma máquina com seguintes especificações:

[16] https://eclipse.org/
[17] http://www.pydev.org/
[18] https://notepad-plus-plus.org/
[19] https://www.mysql.com/
[20] https://www.mysql.com/products/workbench/
[21] https://www.r-project.org/
[22] http://nbcgib.uesc.br/lec/*software*/editores/tinn-r/pt
[23] http://staruml.io/
[24] http://registro.br/

- Sistema operacional Windows 7 - Profissional com Service Pack 1 - 64 bits;
- Processador Intel Core I5-3470 com 3.20 GHz de *clock*;
- Memória RAM de 8 *Gbytes*;
- Disco rígido de 500 *Gbytes*.

Esta máquina possui um número IP (*internet protocol* - protocolo de internet) fixo como número 200.136.228.66, e está rodando um servidor web montado pelo próprio Django na porta 80.

6.3 Levantamento dos requisitos do *framework*

A Figura 6.4 apresenta o diagrama de caso de uso com todas as funcionalidades que são implementadas no *framework*. Salienta-se a recorrência da denominação dos casos de uso que iniciam com o termo "Gerenciar", que neste caso especifica o controle para manipulação dos objetos, ou seja, as ações de criação, recuperação, alteração e remoção de objetos pertencentes a uma determinada classe. Estas ações são comumente chamadas no meio de tecnologia de CRUD (*Create*, *Restore*, *Update*, *Delete* - criar, recuperar, atualizar e apagar) correspondendo a estas operações básicas.

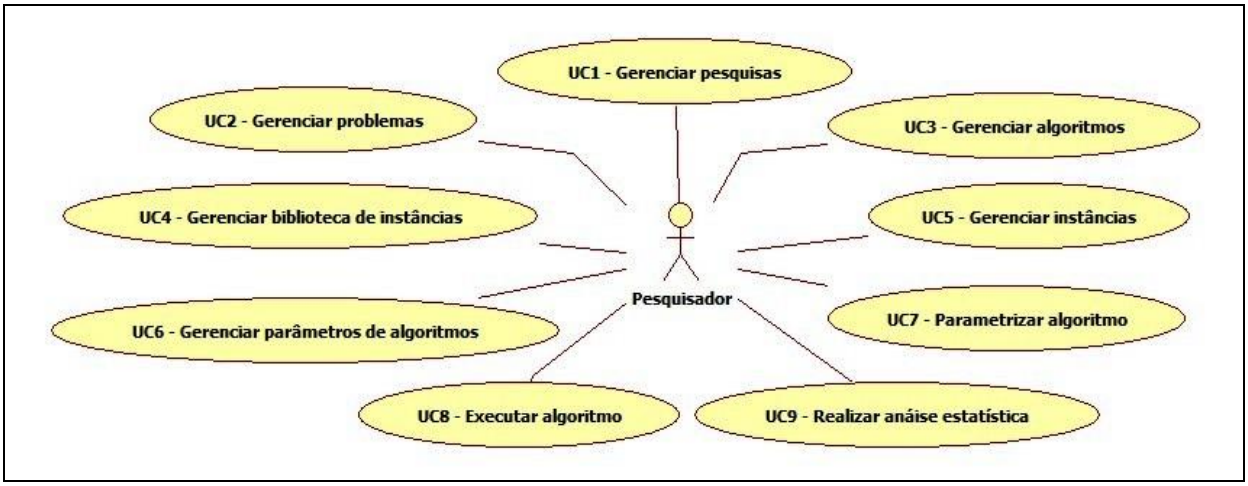


Figura 6.4 - Diagrama de caso de uso do *framework*
Fonte: elaborado pelo autor

A seguir são apresentadas as especificações de todos os casos de uso conforme Figura 6.4. Nesta especificação é utilizado um modelo de documentação para a definição dos requisitos do sistema conforme proposto por Eriksson et al. (2003). Nos quadros 6.2 até 6.10 são apresentados estas especificações.

Quadro 6.2 - Caso de uso gerenciar pesquisa

Especificação de caso de uso	
Nome:	UC1 - Gerenciar pesquisas

Descrição:	Este caso de uso controla os cadastros das pesquisas realizadas.
Atores:	Pesquisador
Pré-condição:	Devem estar cadastrados no sistema o problema que será resolvido, a biblioteca de instâncias, todas as instâncias desta biblioteca e os algoritmos que serão comparados na pesquisa.
Fluxo básico:	1. Listar todas as pesquisas cadastradas no sistema; 2. Para um novo cadastro deve ser especificado o nome, o rótulo da pesquisa, a descrição, objetivos, resultados esperados, equipe de pesquisa, ambiente de execução e a localização do diretório onde os arquivos estarão disponíveis; 3. Criar diretório com o rótulo da pesquisa na pasta Pesquisas; 4. Escolher o método de processamento da pesquisa na lista (Processamento); 5. Listar todos os problemas cadastrados no sistema; 6. Escolher um problema para ser resolvido pela pesquisa; 7. Se não existir um problema executar UC2 - Gerenciar problemas; 8. Listar todas as bibliotecas de instâncias cadastradas no sistema; 9. Escolher uma biblioteca de instância para ser utilizada; 10. Se não existir biblioteca executar UC4 - Gerenciar biblioteca de instâncias; 11. Listar todas as instâncias da biblioteca de instâncias definida; 12. Definir o conjunto das instâncias do *benchmark* de execução; 13. Definir o subconjunto das instâncias do *benchmark* de teste; 14. Listar todos os algoritmos cadastrados no sistema; 15. Escolher os algoritmos para serem comparados pela pesquisa; 16. Se não existir um algoritmo executar UC3 - Gerenciar algoritmo.
Fluxo alternativo:	• Deve ser permitido que uma determinada pesquisa possa ser alterada; • Deve ser permitido que uma determinada pesquisa possa ser consultada; • Deve ser permitido que uma determinada pesquisa possa ser excluída: a. Excluir todos os objetos da classe Execução relacionada a pesquisa; b. Apagar o diretório e os arquivos com o apelido da pesquisa na pasta Pesquisas controlado pelo *framework;* c. Apagar o objeto da pesquisa no *framework.*

Fonte: elaborado pelo autor

Quadro 6.3 - Caso de uso gerenciar problemas

Especificação de caso de uso	
Nome:	UC2 - Gerenciar problemas
Descrição:	Este caso de uso controla os cadastros dos problemas tratados nas pesquisas.
Atores:	Pesquisador
Pré-condição:	Não possui.
Fluxo básico:	1. Listar todos os problemas cadastrados no sistema; 2. Para um novo cadastro especificar o nome e a descrição do problema; 3. Escolher qual o tipo do problema utilizado na pesquisa a partir de lista definida (TipoProblema); 4. Escolher qual o fluxo de processamento do problema utilizado na pesquisa a partir de lista definida (FluxoDeProcesso); 5. Escolher qual a função objetivo do problema utilizado na pesquisa a partir de lista definida (TipoProblema).
Fluxo alternativo:	1. Deve ser permitido que um determinado problema possa ser alterado; 2. Deve ser permitido que um determinado problema possa ser consultado; 3. Não deve ser permitido que um determinado problema possa ser excluído.

Fonte: elaborado pelo autor

Quadro 6.4 - Caso de uso gerenciar algoritmo

Especificação de caso de uso	
Nome:	UC3 - Gerenciar algoritmos
Descrição:	Este caso de uso controla os cadastros dos algoritmos que são comparados na pesquisa.
Atores:	Pesquisador
Pré-condição:	O arquivo executável do algoritmo deve estar disponível.
Fluxo básico:	1. Listar todos os algoritmos cadastrados no sistema; 2. Para um novo cadastro deve ser especificado o nome e o rótulo do algoritmo, descrição, estratégias heurísticas, referências, linguagem e ambiente de desenvolvimento e parâmetros de execução; 3. Criar diretório com o rótulo do algoritmo na pasta Algoritmos controlada pelo *framework;* 4. Escolher qual o tipo de execução do algoritmo a partir de lista definida (TipoExecução); 5. Escolher qual o tipo de heurística utilizado no algoritmo a partir de lista definida (TipoHeurística); 6. Realizar escolha do arquivo executável do algoritmo definindo o nome do arquivo do algoritmo e o diretório do algoritmo; 7. Copiar o arquivo executável para a pasta do algoritmo.
Fluxo alternativo:	1. Deve ser permitido que um determinado algoritmo possa ser alterado; 2. Deve ser permitido que um determinado algoritmo possa ser consultado; 3. Deve ser permitido que um determinado algoritmo possa ser excluído: a. Excluir todos os objetos da execução relacionada ao algoritmo; b. Excluir todos os objetos dos parâmetros relacionados ao algoritmo; c. Apagar a diretório e os arquivos com o apelido do algoritmo na pasta Pesquisas controlado pelo *framework;* d. Apagar o objeto do algoritmo no *framework.*

Fonte: elaborado pelo autor

Quadro 6.5 - Caso de uso gerenciar biblioteca de instâncias

Especificação de caso de uso	
Nome:	UC4 - Gerenciar biblioteca de instâncias
Descrição:	Este caso de uso controla os cadastros das bibliotecas de instâncias que são comparados na pesquisa.
Atores:	Pesquisador
Pré-condição:	O problema ao qual a biblioteca de instância está relacionada deve estar cadastrado no sistema.
Fluxo básico:	1. Listar todas as bibliotecas de instâncias cadastradas no sistema; 2. Para um novo cadastro deve ser especificado o nome, a descrição, o processo de criação e a referência externa da biblioteca de instâncias e diretório de arquivos; 3. Criar diretório com o nome da biblioteca de instâncias na pasta Bibliotecas controlada pelo *framework;* 4. Escolher qual o tipo de *benchmark* da biblioteca a partir de lista definida (Tipo*Benchmark*); 5. Escolher qual o problema ao qual a biblioteca está relacionada a partir de problemas cadastrados no sistema (Problema).
Fluxo alternativo:	1. Deve ser permitido que uma determinada biblioteca de instâncias possa ser alterada; 2. Deve ser permitido que uma determinada biblioteca de instâncias possa ser consultada; 3. Não deve ser permitido que uma determinada biblioteca de instâncias possa ser excluída.

Fonte: elaborado pelo autor

Quadro 6.6 - Caso de uso gerenciar instâncias

Especificação de caso de uso	
Nome:	UC5 - Gerenciar instâncias
Descrição:	Este caso de uso controla os cadastros das instâncias que fazem parte de uma biblioteca de instâncias.
Atores:	Pesquisador
Pré-condição:	A biblioteca de instância ao qual a instância faz parte deve estar cadastrada.
Fluxo básico:	1. Listar todas as bibliotecas de instâncias cadastradas no sistema; 2. Escolher uma biblioteca de instâncias da lista; 3. Listar todas as instâncias da biblioteca de instâncias escolhida; 4. Para um novo cadastro deve ser especificado o nome, descrição, o valor da solução ótima, o valor da melhor solução e a data quando a melhor solução foi encontrada; 5. Para cadastro de conjunto de novas instâncias de uma biblioteca deve ser especificado qual biblioteca estas instâncias pertencem e selecionado o diretório de origem das instâncias, que deve conter os arquivos de cada instância, um arquivo texto com os solução ótima de cada instância separados por linha e um arquivo texto com as melhores soluções e a data quando a melhor solução foi encontrada também separados por linhas; 6. Copiar o(s) arquivo(s) da(s) instância(s) para o diretório com o nome da biblioteca de instâncias especificada na pasta Bibliotecas no *framework*; 7. Criar um objeto no *framework* para cada instância.
Fluxo alternativo:	1. Deve ser permitido que uma determinada instância possa ser alterada; 2. Deve ser permitido que uma determinada instância possa ser consultada; 3. Não deve ser permitido que uma determinada instância possa ser excluída. 4. Deve ser permitido que um grupo de instâncias de uma biblioteca possa ser cadastrado de uma só vez: a. Listar todas as bibliotecas cadastradas no sistema; b. Escolher uma biblioteca de instância na lista; c. Escolher os arquivos de instâncias que serão cadastrados. Junto com os arquivos de instâncias deve ter um arquivo texto com valores ótimos e um arquivo com os melhores valores e datas de obtenção; d. Fazer *upload* de todos os arquivos e gravar na pasta da biblioteca de instância especificada; e. Repetir para cada instância da lista: i. Identificar o valor ótimo e o melhor valor para esta instância; ii. Cadastrar esta instância no sistema para a biblioteca de instância definida.

Fonte: elaborado pelo autor

Quadro 6.7 - Caso de uso gerenciar parâmetros do algoritmo

Especificação de caso de uso	
Nome:	UC6 - Gerenciar parâmetros dos algoritmos
Descrição:	Este caso de uso controla os cadastros dos parâmetros de execução dos algoritmos que são comparados na pesquisa.
Atores:	Pesquisador
Pré-condição:	O algoritmo deve estar cadastrado
Fluxo básico:	1. Listar todos os parâmetros cadastrados no sistema; 2. Para um novo parâmetro especificar o nome, descrição, melhor valor do parâmetro se for conhecido e a ordem na linha de comando ao qual o parâmetro deve aparecer; 3. Listar todos os algoritmos cadastrados no sistema; 4. Escolher um algoritmo ao qual o parâmetro faz parte.
Fluxo alternativo:	1. Deve ser permitido que um determinado parâmetro possa ser alterado; 2. Deve ser permitido que um determinado parâmetro possa ser consultado; 3. Deve ser permitido que um determinado parâmetro possa ser excluído: a. Apagar o objeto de parâmetro do *framework.*

Fonte: elaborado pelo autor

Quadro 6.8 - Caso de uso parametrizar algoritmo

Especificação de caso de uso	
Nome:	UC7 - Parametrizar algoritmo
Descrição:	Este caso de uso efetua a parametrização de cada algoritmo comparado na pesquisa com os grupos de teste de parâmetros cadastrados e com cada instância de teste definida pelo pesquisador.
Atores:	Pesquisador
Pré-condição:	- A pesquisa deve estar cadastrada; - O algoritmo deve estar cadastrado; - Os parâmetros dos algoritmos devem estar cadastrados; - O *benchmark* de teste deve estar definido.
Fluxo básico:	1. Listar todas os algoritmos cadastradas no sistema; 2. Escolher um algoritmo para fazer a parametrização; 3. Listar todos os parâmetros cadastrados para o algoritmo já na ordem definida; 4. Especificar três (3) grupos de teste de parâmetros diferentes para realizar a parametrização. Cada grupo deve conter todos os valores de cada parâmetros cadastrado para o algoritmo separados por espaços; 5. Listar todos as pesquisas cadastrados que estão relacionadas ao algoritmo; 6. Escolher a pesquisa que será usada para realizar a parametrização a partir da definição do *benchmark* de teste especificado; 7. Escolher qual será a base de cálculo do processamento dos dados definido na pesquisa podendo ser valor ótimo, melhor valor, valor mínimo encontrado ou valor médio encontrado; 8. Identificar todos os parâmetros do algoritmo escolhido; 9. Identificar os valores dos grupos de teste de parâmetros para o algoritmo; 10. Repetir para cada grupo de teste de parametrização: 10.1. Repetir para cada instância do *benchmark* de teste: 10.1.1. Montar linha de comando de execução do algoritmo com argumentos de entrada contendo nome arquivo da instância de teste, o número de parâmetros e a lista com os valores de parâmetros de configuração do grupo de teste especificado; 10.1.2. Executar o algoritmo com lista de argumentos de entrada; 10.1.3. Guardar os valores de retorno da execução do algoritmo: valor FIT encontrado e o tempo total de execução. 10.2. Montar e gravar na pasta do algoritmo as tabelas com os resultados por grupo de teste de parâmetros por instâncias de teste com os dados dos valores FIT encontrados ("<<Rótulo da Pesquisa>>FIT.txt") e dos tempos totais de execução ("<<Rótulo da Pesquisa>>TIME.txt"); 10.3. Montar e gravar na pasta do algoritmo a tabela com o processamento dos resultados calculados a partir da base de cálculo definida ("<<Rótulo da Pesquisa>>.txt"); 10.4. Realizar a análise estatística com os valores já processados apresentando relatório com foco nas estatísticas descritivas e gráfico *boxplot* de forma a permitir que o pesquisador identifique qual o melhor grupo de teste de parâmetro definido. 11. Fazer *download* de todos os resultados do processo de parametrização; 12. Gravar na pasta da pesquisa as informações sobre a parametrização realizada.
Fluxo alternativo:	Não possui

Fonte: elaborado pelo autor

Quadro 6.9 - Caso de uso executar algoritmo

Especificação de caso de uso	
Nome:	UC8 - Executar algoritmo
Descrição:	Este caso de uso efetua a execução de cada algoritmo da pesquisa com os parâmetros definidos e com cada instância de execução especificada.
Atores:	Pesquisador
Pré-condição:	- A pesquisa deve estar cadastrada; - O algoritmo deve estar cadastrado; - Os parâmetros dos algoritmos devem estar cadastrados e definidos os melhores valores; - O *benchmark* de execução deve estar definido.
Fluxo básico:	1. Listar todas as pesquisas cadastradas no sistema; 2. Escolher uma pesquisa; 3. Listar os dados cadastrados da pesquisa escolhida; 4. Escolher qual será a base de cálculo do processamento dos dados definido na pesquisa podendo ser valor ótimo, melhor valor, valor mínimo encontrado, valor médio encontrado ou a definição de um algoritmo como referência para o cálculo; 5. Listar todos os algoritmos cadastrados e relacionados com a pesquisa; 6. Se a base de cálculo do processamento for um algoritmo como referência permitir que o pesquisador escolha qual será o algoritmo de referência; 7. Repetir para cada algoritmo definido: 7.1. Identificar todos os parâmetros do algoritmo escolhido com os melhores valores e a ordem definida; 7.2. Repetir para cada instância do *benchmark* de execução: 7.2.1. Montar linha de comando de execução do algoritmo com argumentos de entrada contendo nome arquivo da instância de execução, o número de parâmetros e a lista com os valores de parâmetros de configuração; 7.2.2. Executar o algoritmo com lista de argumentos de entrada; 7.2.3. Guardar os valores de retorno da execução do algoritmo: valor FIT encontrado e o tempo total de execução. 8. Montar e gravar na pasta da pesquisa as tabelas com os resultados por grupo de teste de parâmetros por instâncias de teste com os dados dos valores FIT encontrados ("<<Rótulo da Pesquisa>>FIT.txt") e dos tempos totais de execução ("<<Rótulo da Pesquisa>>TIME.txt") separadas por tabulações; 9. Montar e gravar na pasta da pesquisa a tabela com o os valores dos processamento dos resultados calculados a partir da base de calculo definida ("<<Rótulo da Pesquisa>>.txt") separados por tabulações; 10. Fazer *download* de todos os resultados do processo de execução; 11. Gravar na pasta da pesquisa as informações sobre a execução realizada.
Fluxo alternativo:	Não possui

Fonte: elaborado pelo autor

Quadro 6.10 - Caso de uso realizar análise estatística

Especificação de caso de uso	
Nome:	UC9 - Realizar análise estatística
Descrição:	Este caso de uso realiza a análise estatística com os resultados das execuções dos algoritmos para cada instância de execução especificada pelo pesquisador.
Atores:	Pesquisador
Pré-condição:	- A pesquisa deve estar cadastrada - Deve estar disponível o arquivo "<<Rótulo da Pesquisa>>.txt", no diretório da pesquisa, com os valores processados nas execuções dos algoritmos.
Fluxo básico:	1. Listar todas as pesquisas cadastradas no sistema; 2. Escolher uma pesquisa; 3. Definir se os resultados encontrados seguem uma distribuição estatística normal e se são dependentes entre si; 4. Conferir a existência do arquivo com resultados das execuções; 5. Identificar a quantidade de algoritmos comparados na pesquisa;

Continuação do Quadro 6.10 - Caso de uso realizar análise estatística

Fluxo básico:	6. Montar lista de argumentos de entrada contendo o apelido da pesquisa, o número de algoritmos comparado na pesquisa, a normalidade (True/False) e a dependência (True/False); 7. Chamar *o script* para análise estatística com argumentos de entrada definidos: 7.1. Calcular resultados da estatística descritiva (média, desvio padrão, erro padrão, intervalo interquartil, coeficiente de variabilidade, simetria dos dados, grau de concentração dos dados em relação a média, moda, frequência da moda, valor mínimo, primeiro quartil, mediana, terceiro quartil e valor máximo); 7.2. Realizar análise gráfica das amostras: 7.2.1. Apresentar gráficos *boxplot* com e sem *outliers*; 7.2.2. Apresentar gráfico da moda e frequência da moda; 7.2.3. Apresentar gráficos dos valores encontrados (*fitness*) pelas instâncias; 7.2.4. Apresentar gráfico de coordenadas paralelas das amostras com valores da média e da mediana; 7.2.5. Apresentar gráfico de dispersão com limites de erro padrão e média. 7.3. Realizar teste de normalidade para os valores encontrados: 7.3.1. Calcular e apresentar tabela com os valores dos testes de Shapiro-Wilk (número de amostras menor ou igual a 4000) e Lilliefors (número de amostras maior que 4000); 7.3.2. Apresentar gráficos de histograma de cada algoritmo; 7.3.3. Apresentar gráficos de comparação quartil-quartil de cada algoritmo. 7.4. Realizar análise gráfica comparativa das amostras apresentando gráficos *boxplot* das diferenças entre os algoritmos; 7.5. Se número de algoritmos for igual a dois (2): 7.5.1. Se os resultados seguirem uma distribuição normal: • Executar teste-t pareado apresentando resultados; 7.5.2. Se os resultados não seguirem uma distribuição normal: • Executar teste de sinal bilateral com apresentação de gráfico das diferenças entre os algoritmos; • Executar teste Wilcoxon pareado apresentando resultados; 7.6. Se o número de algoritmos comparados for maior que dois (2): 7.6.1. Se os resultados seguirem uma distribuição normal: • Executar teste ANOVA com um fator e blocos apresentando resultados gerais; • Apresentar gráficos para análise dos residuais do ANOVA; • Executar análise *post-hoc* usando teste Fisher´s LSD (*Least Significant Difference* - Menor Diferença Significante); • Executar análise *post-hoc* utilizando Tukey HSD (*Honest Significant Difference* - Diferença Significante Honesta); • Executar análise *post-hoc* utilizando teste-t pareado; 7.6.2. Se os resultados não seguirem uma distribuição normal: • Executar teste de Friedman apresentando resultados gerais; • Executar análise *post-hoc* usando teste Fisher´s LSD (*Least Significant Difference* - Menor diferença significante); • Executar análise *post-hoc* utilizando teste de simetria; • Executar análise *post-hoc* utilizando teste de Nemenyi; 7.7. Gerar arquivos PNG e EPS de todos os gráficos apresentados; 7.8. Gerar arquivos TXT e XLS de todas as tabelas com resultados apresentados; 8. Gerar arquivo "Resultadosgerais-<<Apelido pesquisa>>.DOC" contendo relatório estatístico final com todos os resultados acima; 9. Fazer *download* de todos os resultados da análise estatística.
Fluxo alternativo:	Não possui

Fonte: elaborado pelo autor

6.4 Análise e especificação do *framework*

A Figura 6.5 apresenta o diagrama de classe simplificado com a especificação de todas as classes do sistema, juntamente com as propriedades definidas para cada classe e os relacionamentos necessários para a representação das regras de negócio do *framework* proposto. Este diagrama de classe tem como foco a classe Pesquisa que concentra todas as informações necessárias para o controle e execução do *framework*. Uma restrição é que o campo RotuloPesq não pode ter mais do que 12 caracteres, para não ter problemas de formatação do relatório estatístico que é gerado automaticamente.

Foi definido uma limitação de navegabilidade entre as várias classes relacionadas à Pesquisa tais como Problema, BibliotecaInst e Instância com as definições de *benchmark* de execução e *benchmark* de teste. O motivo disto é que um objeto de Pesquisa precisa identificar os objetos destas classes, mas não necessariamente o contrário. Esta ação permite a criação de vetores de objetos relacionados somente dentro da classe Pesquisa.

Todas as classificações identificadas na pesquisa realizada no Capítulo 5 estão representadas no diagrama de classe através de classes do tipo <<enumeration>>. Estas enumerações são utilizadas na especificação de várias classes tais como Pesquisa com a enumeração de TipoCalculo, na classe Problema com as enumerações do TipoProblema, FluxoDeProcesso e FuncaoObjetivo, na classe Instância com a enumeração do Tipo*Benchmark* e na classe Algoritmo com as enumerações do TipoHeurística e TipoExecucao.

Outra classe importante do diagrama é a classe Algoritmo, que representa uma das soluções heurísticas utilizadas no processo de comparação realizada por uma determinada pesquisa. Uma restrição é que o campo RotuloAlg não pode ter mais do que 6 caracteres, para não ter problemas de formatação do relatório estatístico que é gerado automaticamente. Esta classe está relacionada à classe Parametros que especifica cada um dos parâmetros utilizados na configuração de execução do algoritmo.

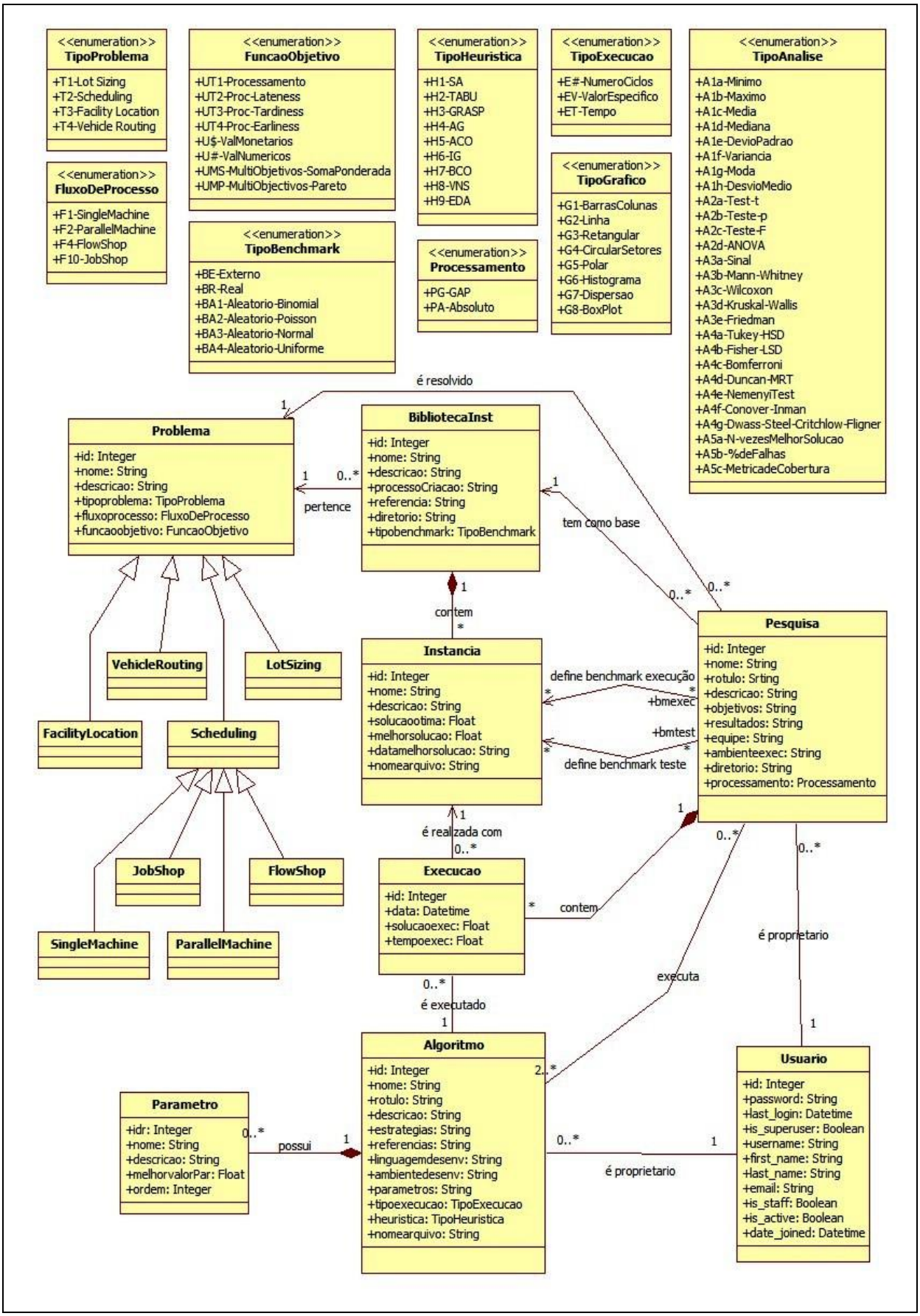


Figura 6.5 - Diagrama de classe simplificado do *framework*
Fonte: elaborado pelo autor

Como parte do processo de desenvolvimento de pesquisas proposto no Capítulo 5, têm-se ainda duas etapas que são controladas pelo *framework*. Uma etapa é a parametrização simplificada, que como descrito, permite um ajuste fino em um processo de parametrização externo. Após a definição dos grupos de teste de parâmetros o *framework* executa cada algoritmo, com cada grupo de teste de parâmetro e com cada instância de teste especificada. Após todas as execuções o *framework* processa a análise estatística dos resultados fornecendo assim ao pesquisador uma ferramenta para tomada de decisão com relação aos melhores parâmetros de execução para o algoritmo. A outra etapa é a execução do algoritmo com os parâmetros de execução identificados e com cada instância do *benchmark* de execução definido. Todos os resultados encontrados são formatados e armazenados em um arquivo texto externo, que servirá como dados de entrada para a análise estatística dos resultados sendo executada pelo *script* em linguagem R.

A Figura 6.6 apresenta o diagrama de entidades e relacionamentos implementado no banco de dados MySql, juntamente com todas as tabelas mapeadas a partir das classes definidas, os campos de cada tabela com identificação dos tipos de dados, as chaves primárias de e todos os relacionamentos identificados a partir do diagrama de classe. Todos os relacionamentos muitos para muitos foram resolvidos no modelo de dados. No relacionamento entre Pesquisa e Instância foram definidas as tabelas TB_*Benchmark*Execucao e TB_BenchamrkTeste, que armazenam quais são as instâncias de execução e quais são as de teste. No relacionamento entre Pesquisa e Algoritmo foi definido a tabela TB_PesquisaAlgoritmo que referencia quais são os algoritmos que são comparados na pesquisa correspondente.

Para todas as classes do tipo <<enumeration>> do diagrama de classes foram criadas tabelas para o armazenamento das opções de escolha de forma a facilitar a administração e expansão do sistema.

O *framework* possui ainda um controle de usuários e acessos. Para que o pesquisador possa trabalhar no *framework* ele deve estar cadastrado e logado no sistema. Neste sentido, se o pesquisador não for cadastrado, o mesmo deve efetuar o cadastro de um novo usuário no sistema. Será enviado um e-mail para o administrador do *framework* solicitando a liberação do cadastro deste novo usuário. Somente após a liberação do usuário é que o pesquisador poderá cadastrar suas pesquisas. Destacado na Figura 6.6, pode-se ver as tabelas que foram criadas

automaticamente pelo Django para gerenciamento de usuários, grupos de usuários, permissões e controle de acesso.

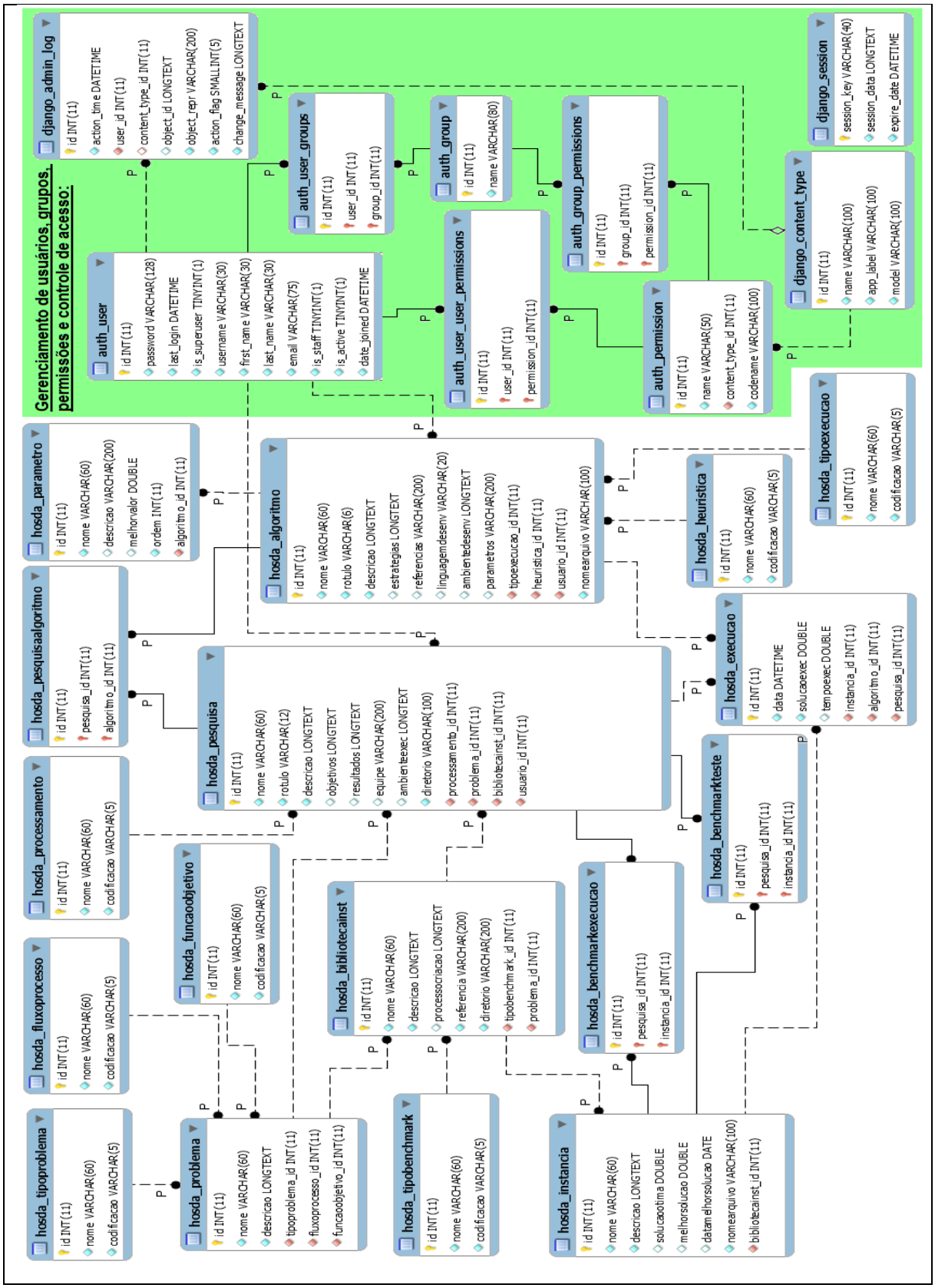


Figura 6.6 - Diagrama de entidades e relacionamentos (DER) do *framework*
Fonte: elaborado pelo autor

6.5 Iteração 1 - Desenvolvimento da análise estatística dos resultados

Como parte do processo de desenvolvimento do *framework* proposto, a iteração 1 especifica a construção da análise estatística dos resultados. Esta iteração foi desenvolvida em linguagem R (R DEVELOPMENT CORE TEAM, 2014) principalmente por ser uma linguagem específica na implementação de análises estatísticas. O apêndice B apresenta um quadro com todos os pacotes com as respectivas referências que foram utilizadas no *script*. O *script* foi desenvolvido utilizando como base o processo Realizar Análise Estatística dos Resultados proposto no método de pesquisas comparativas em otimização utilizando algoritmos heurísticos proposto no capítulo 5. As etapas previstas na execução do *script* e os resultados reportados no relatório das análises estatísticas de saída seguem a estrutura proposta anteriormente.

Para o funcionamento do *script* são necessários alguns parâmetros de entrada. O primeiro parâmetro é o arquivo de dados com os resultados da pesquisa que efetua a comparação entre algoritmos. Este arquivo é um arquivo texto com os dados separados por tabulações. Na primeira linha devem aparecer os rótulos (apelidos) dos algoritmos e as próximas linhas são os resultados encontrados nas execuções de cada um dos algoritmos com cada uma das instâncias definidas. O Quadro 6.11 apresenta o modelo de formatação dos dados de entrada necessário para o *script* funcionar.

Quadro 6.11 - Modelo de formatação do arquivo de dados de entrada

	Alg1	Alg2	Alg3	...	AlgN
1	X_{11}	X_{12}	X_{13}	...	X_{1N}
2	X_{21}	X_{22}	X_{23}	...	X_{2N}
3	X_{31}	X_{32}	X_{33}	...	X_{3N}
4	X_{41}	X_{42}	X_{43}	...	X_{4N}
...	...	...	...	...	...
N	X_{N1}	X_{N2}	X_{N3}	...	X_{NN}

Fonte: elaborado pelo autor

O *script* precisa ainda de mais três parâmetros para escolha de qual teste estatístico inferencial será executado. O primeiro parâmetro é a quantidade de soluções algorítmicas comparadas na pesquisa, podendo ser no mínimo dois (2) e máximo (10). O segundo é um valor booleano (lógico - verdadeiro ou falso) que define se os resultados presentes no arquivo seguem uma distribuição estatística normal para cada algoritmo. Como a maioria das soluções heurísticas possuem uma base

estocástica, ou seja, fazem uso de números aleatórios, pode-se dizer que não seguem uma distribuição estatística normal, mas foi deixado como escolha do pesquisador a definição desta característica. O terceiro valor também é um booleano que especifica a dependência dos dados das amostras comparadas. O *framework* faz comparações entre soluções algorítmicas que são executadas com um mesmo conjunto de instâncias, consequentemente concluí-se que existem dependências entre as amostras analisadas. Mesmo o *framework* não tratando de conjunto de dados independentes o *script* já está pronto para estas análises. O Quadro 6.12 apresenta todos os testes estatísticos inferenciais que são executados pelo *script* desenvolvido.

Quadro 6.12 - Testes estatísticos executados dependendo dos parâmetros de entrada

Nro.de Soluções	Normalidade	Dependência	Teste Estatístico Aplicado
2	Sim	Não	Teste-t para amostras independentes
2	Sim	Sim	Teste-t pareado
2	Não	Não	Teste Mann-Whitney
2	Não	Sim	Teste Wilcoxson pareado
> 2	Sim	Não	Teste ANOVA simples
> 2	Sim	Sim	Teste ANOVA com um fator e blocos
> 2	Não	Não	Teste Kruskal-Wallis
> 2	Não	Sim	Teste Friedman

Fonte: elaborado pelo autor

O objetivo principal do relatório estatístico gerado é fornecer um conjunto de padrões de análises estatísticas que podem ser utilizados na validação dos resultados gerados, melhorando assim a qualidade da pesquisa realizada. Vale ressaltar que o pesquisador deve escolher no relatório quais foram os resultados estatísticos que agregam mais valor a sua pesquisa, ou seja, utilizar somente aquelas informações necessárias. Todo o código desenvolvido para o *script* em linguagem R está disponível para *download* no *framework*[25].

6.6 Iteração 2 - Gerenciamento de cadastros

Esta iteração aborda o gerenciamento dos cadastros de todos os dados necessários para o funcionamento do *framework*. Como dito anteriormente, para cada cadastro são implementadas as operações básicas de criação, consulta, atualização e deleção de objetos das classes pesquisa, problema, algoritmo, parâmetros de execução do algoritmo, usuário, biblioteca de instâncias e instâncias. A Figura 6.7 apresenta a tela principal do *framework* Hosda.

[25] http://www.hosda.com.br/analise/*downloadscheduling*

Figura 6.7 - Tela principal do *framework* Hosda
Fonte: elaborado pelo autor

Os itens do menu de opções "Requisitos Pesquisa" e "*Benchmark*" executam o gerenciamento dos cadastros do *framework*. A Figura 6.8 apresenta o gerenciamento de cadastro de uma pesquisa. Para acessar esta página, o usuário deve clicar no item do menu "Requisitos Pesquisa" e no subitem "Nova Pesquisa". A página é aberta com todos os campos que devem ser cadastrados para uma nova pesquisa. Se este for o caso, o usuário preenche os campos da nova pesquisa, incluindo a escolha dos algoritmos que serão comparados, a biblioteca de instâncias, os *benchmarks* de execução e teste, necessitando que estes dados estejam previamente cadastrados e clica no botão "Salvar". Se os dados cadastrados estivem corretos será criado um novo objeto de pesquisa no sistema. Ao abrir esta tela de cadastro também são listadas todas as pesquisas já cadastradas, permitindo que o usuário, através de botões de ações disponíveis, possa alterar ou excluir um objeto pesquisa do sistema.

Se o usuário clicar um "Listar Pesquisas" será aberta uma página com a listagem de todas as pesquisas cadastradas. Além dos botões de ação de alteração e exclusão, esta disponível um botão de detalhes da pesquisa que fornece um relatório completo das informações da pesquisa cadastrada, dos algoritmos comparados e dos *benchmark*s definidos permitindo que este relatório possa ser impresso.

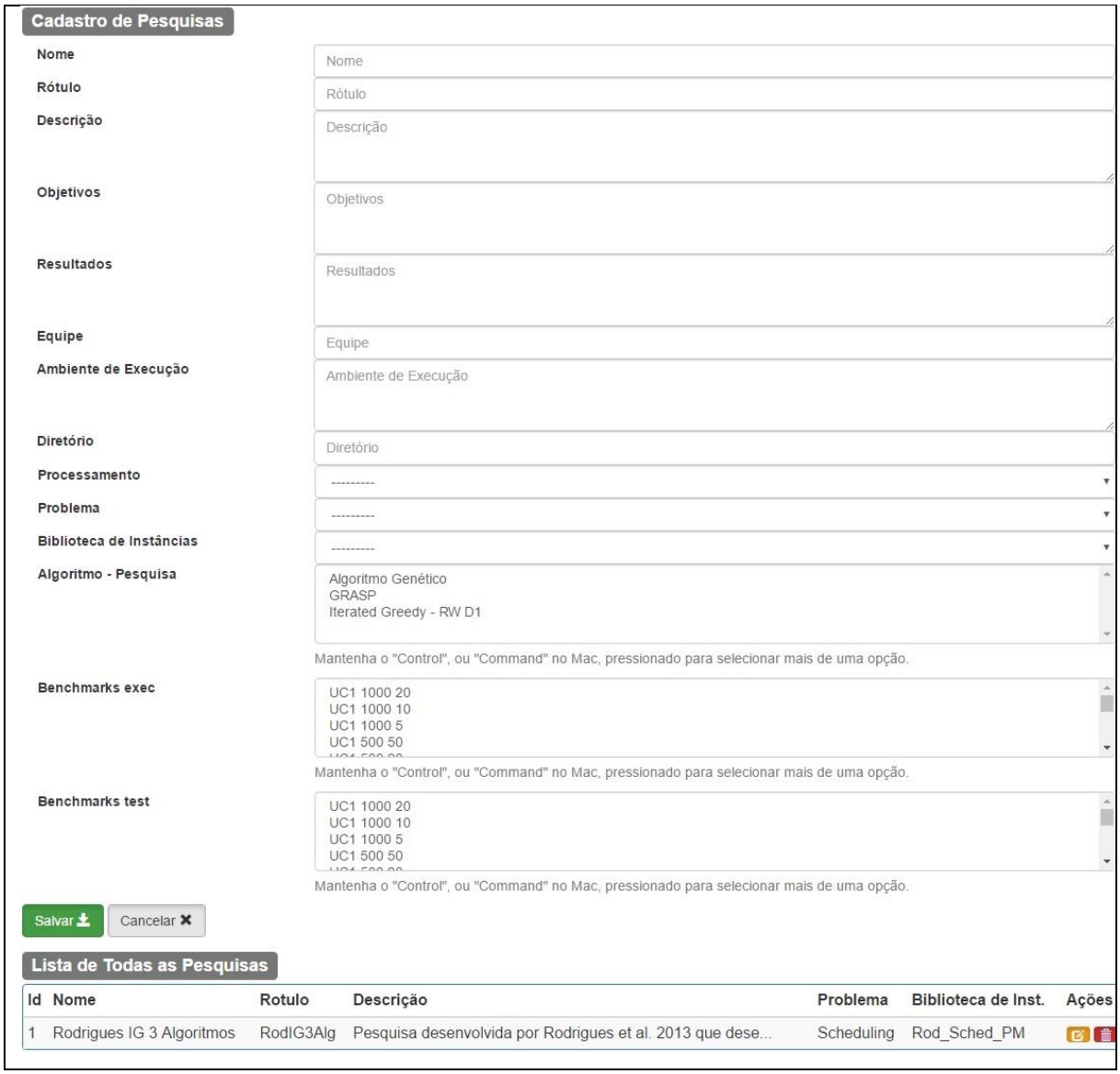


Figura 6.8 - Tela de cadastro de pesquisas
Fonte: elaborado pelo autor

O gerenciamento dos cadastros de problemas de engenharia de produção, algoritmos comparados, parâmetros de execução, bibliotecas de instâncias e instâncias seguem o mesmo padrão de execução descrito acima. O gerenciamento de cadastro de usuários é um pouco diferente. Quando for solicitado o cadastro de um novo usuário, será enviado um e-mail para o administrador do sistema solicitando o aceite e liberação do usuário para utilização do *framework*. Somente após a liberação do administrador é que o usuário poderá utilizar o sistema. O usuário somente terá acesso aos dados cadastrados por ele não permitindo acesso a outras pesquisas. Para que este acesso seja liberado é necessário que o usuário faça o *login* no sistema. A única exceção é a listagem de pesquisas já cadastradas com relatório de detalhes da pesquisa que pode ser solicitado por qualquer pessoa mesmo não estando cadastrado no sistema.

6.7 Iteração 3 - Desenvolvimento da execução automática dos algoritmos

Esta iteração trata da execução automática dos algoritmos disponibilizando os resultados encontardos para o *script* de análise estatística. A Figura 6.9 apresenta a tela da execução automática dos algoritmos. Para acessar esta página o usuário deve clicar no item do menu "Execução" e no subitem "Executar Pesquisa". O usuário escolhe um item da lista de pesquisas apresentada solicitando a execução. São apresentados os dados principais da pesquisa e informações sobre a existência de valores ótimos e melhores valores nas instâncias definidas no *benchmark* de execução. O *framework* permite que o pesquisador escolha a base de cálculo que será utilizada no processamento dos dados, podendo ser: valor ótimo, melhor valor, valor mínimo encontrado, valor médio encontrado ou valor de referência a um algoritmo cadastrado. A escolha do valor ótimo ou melhor valor somente será possível se estes valores estivem disponíveis nos cadastros das instâncias de execução. Se a escolha for como valor de referência a um algoritmo, o usuário deverá escolher qual dos algoritmos cadastrado será utilizado, sendo que este algoritmo não fará parte do arquivo de resultados processados pelo *framework*. Após todas as escolhas definidas o usuário deve clicar no botão "Realizar Execução". O *framework* irá processar a execução de todos os algoritmos cadastrados na pesquisa com os parâmetros de execução especificados e com todas as instâncias definidas no *benchmark* de execução.

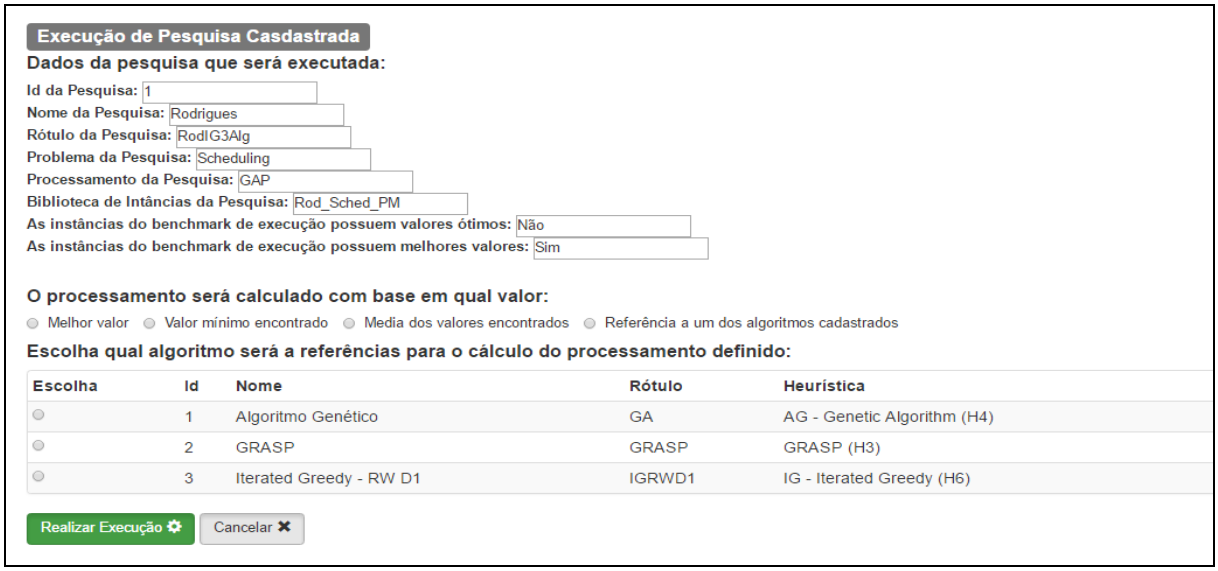


Figura 6.9 - Tela do processo de execução de pesquisa cadastrada
Fonte: elaborado pelo autor

Os resultados encontrados são armazenados no *framework* e ao final da execução são gerados os arquivos contendo os resultados encontrados sendo: arquivo texto com os resultados das execuções ("<<Rótulo da Pesquisa>>Fit.txt"), arquivo texto com os tempos das execuções ("<<Rótulo da Pesquisa>>Time.txt") e um arquivo texto com os dados processados a partir da base de cálculo definida e o tipo de processamento cadastrado na pesquisa ("<<Rótulo da Pesquisa>>.txt"). Outro arquivo gerado é o "DadosExecução.txt" que contém informações sobre esta última execução incluindo, além de informações da pesquisa e dos algoritmos executados, data e hora da execução e base de cálculo escolhida. O *framework* faz ainda o *download* de todos os resultados encontrados. Se já existirem resultados de execuções anteriores estes dados e as análises estatísticas correspondentes a estes resultados serão apagados do sistema.

É possível para o pesquisador recuperar todos os resultados da última execução realizada na pesquisa bastando clicar no subitem "Resultados da Execução da Pesquisa" do item do menu "Execução". O usuário deve escolher uma pesquisa da lista apresentada e se existir alguma execução recente o *framework* faz o *download* dos resultados disponíveis.

Para que o *framework* possa chamar as soluções algorítmicas cadastradas o pesquisador deve, no cadastro de cada algoritmo, ter feito o *upload* do arquivo executável do algoritmo, que segue o padrão especificado no *framework* para os argumentos de entrada e o formato de saída dos dados da execução. Os argumentos de entrada para a linha de comando de execução de um algoritmo devem ter:

1. O caminho completo para o nome do arquivo do algoritmo;
2. O caminho completo para o nome da instância que está sendo executada;
3. A quantidade de parâmetros de execução (exemplo 3);
4. O valor do parâmetro 1;
5. O valor do parâmetro 2;
6. O valor do parâmetro 3.

Como exemplo tem-se:

c:/.../algoritmos/algAAA.exe c:/.../instancias/inst01.txt 3 34.0 56.4 21.7

O algoritmo deve, após a execução, gravar um arquivo texto de saída com o nome "resultados.txt" contendo:

- Primeira linha: o FIT,ou seja, resultado da execução do algoritmo com os parâmetros e com a instância;

- Segunda linha: o TEMPO, ou seja, quanto tempo foi gasto na execução do algoritmo.

Após a execução da pesquisa o usuário pode solicitar que a análise estatística dos resultados seja processada. A Figura 6.10 apresenta a tela de chamada deste processo. Para acessar esta página, o usuário deve clicar no item do menu "Análise Estatística" e no subitem "Realizar Análise Estatística de Pesquisa Executada". O usuário escolhe da lista apresentada qual pesquisa que se quer analisar e especifica se os dados dos resultados encontrados seguem uma distribuição normal ou não. Novamente vale lembrar que se as soluções algorítmicas propostas fazem uso de valores aleatórios dificilmente os dados dos resultados seguirão uma distribuição normal, mas foi deixado como decisão de análise para o pesquisador. O *framework* faz ainda o *download* de todos os resultados encontrados.

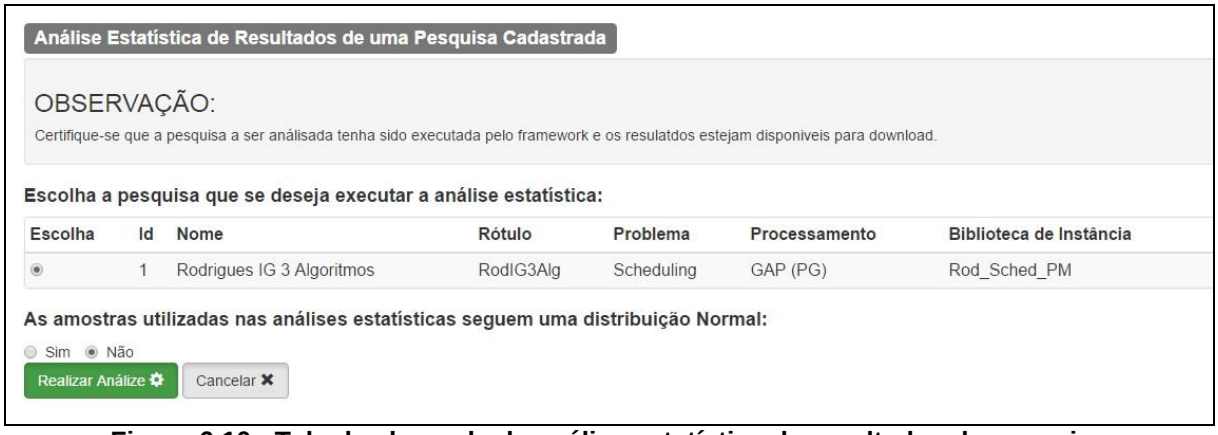


Figura 6.10 - Tela de chamada da análise estatística de resultados de pesquisa
Fonte: elaborado pelo autor

Quando solicitado o processamento da análise estatística, o *framework* apaga todos os dados anteriores gerados por outro processamento e executa o *script* a partir dos parâmetros de entrada sendo: o nome do arquivo de resultados processados pela pesquisa ("<<Rótulo da Pesquisa>>.txt"), a quantidade de soluções algorítmicas comparadas na pesquisa, a definição de normalidade dos dados definida pelo pesquisador e a definição de dependência, que no caso do *framework* é executado como dependente. O Quadro 6.13 apresenta as possibilidades de testes estatísticos inferenciais que podem aparecer no relatório final gerado pelo *script* dependendo dos parâmetros de entrada.

Quadro 6.13 - Testes estatísticos executados pelo *framework*

Nro.de Soluções	Normalidade	Teste Estatístico Aplicado
2	Sim	Teste-t pareado
2	Não	Teste Wilcoxon pareado
> 2	Sim	Teste ANOVA com um fator e blocos

> 2	Não	Teste Friedman

Fonte: elaborado pelo autor

É possível para o pesquisador recuperar todo os resultados do último processamento da análise estatística, bastando clicar no subitem "Resultados das Análises Estatísticas" do item do menu "Análise Estatística". O usuário deve escolher uma pesquisa da lista apresentada e se existir algum processamento recente, o *framework* faz o *download* de todos os resultados disponíveis.

Outra opção disponibilizada no *framework* foi a possibilidade do processamento da análise estatística com *upload* de resultados sem a necessidade de cadastro completo de uma pesquisa, com seus algoritmos e o *benchmark* de execução. A Figura 6.11 apresenta a tela de chamada deste processo. Para acessar esta página, o usuário deve clicar no item do menu "Análise Estatística" e no subitem "Realizar Análise Estatística com *Upload* de Arquivo". Se o pesquisador tiver os resultados de uma pesquisa em mãos, o mesmo deve ser formatado conforme os dados de entrada definidos anteriormente. Deve ser definido um nome para a pesquisa, o número de soluções algorítmicas comparadas, a característica de normalidade e a liberação da característica de dependência que possibilita a execução de outras análises estatísticas. Além destes dados o pesquisador deve escolher o arquivo de resultados já formatado no padrão definido no Quadro 6.11.

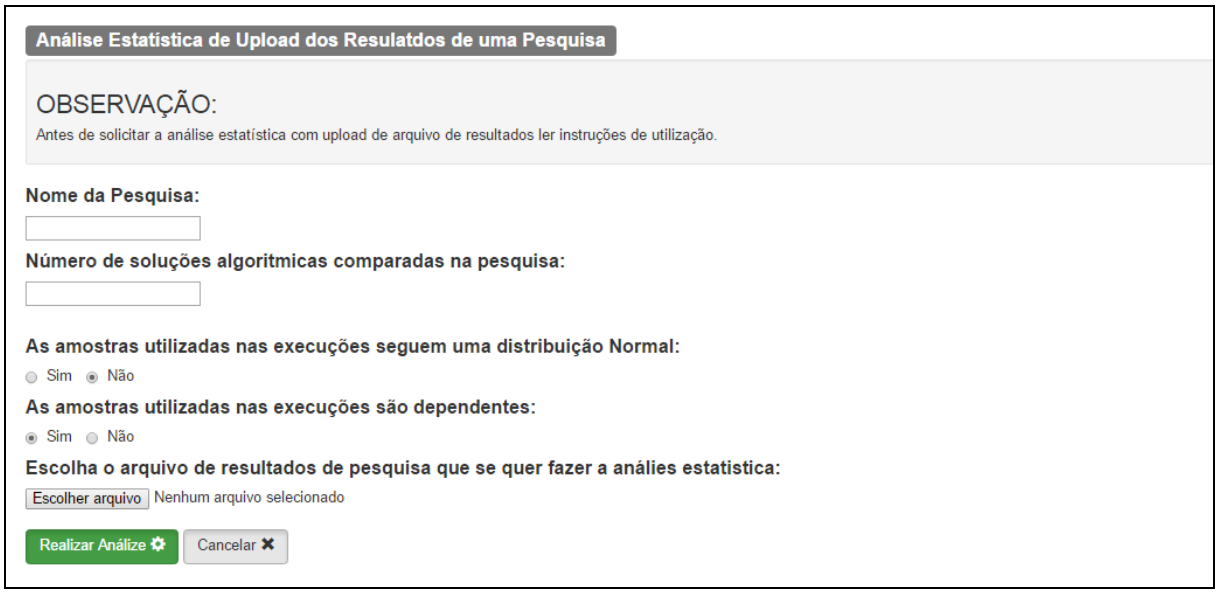


Figura 6.11 - Tela de chamada da análise estatística com *upload* de resultados
Fonte: elaborado pelo autor

O *script* de análise estatística é executado com os dados carregados por *upload* gerando o conjunto de análises estatísticas. Da mesma forma o *framework* faz o *download* de todos os resultados encontrados.

Todo o *script* codificado em linguagem R desenvolvido para o *framework* está disponível para *download* [26]. Se o pesquisador necessitar pode fazer o *download* do *script* para rodar em sua própria máquina. Como pré-requisito para execução do *script* deve ser instalado do *software* R versão 3.2.1[27]. Todas as bibliotecas necessárias para que o *script* seja executado são instaladas e carregadas na execução.

6.8 Iteração 4 - Desenvolvimento do processo simplificado de parametrização com grupos de teste de parâmetros

Esta última iteração do processo de implementação do *framework* diz respeito ao processo de parametrização de um algoritmo. O foco desta tese não é fazer um processo de parametrização completo. Neste sentido esta etapa desenvolve um processo simplificado que pode dar apoio a um processo de parametrização externo realizado pelo pesquisador. A Figura 6.12 apresenta a tela de chamada deste processo. Para acessar esta página, o usuário deve clicar no item do menu "Parametrização" e no subitem "Executar Parametrização de Algoritmo". O usuário escolhe um item da lista de algoritmos disponíveis e solicita a parametrização. São apresentados os dados principais do algoritmo sendo solicitado que o usuário defina três (3) diferentes grupos de valores para realizar a parametrização. O *framework* apresenta os nomes dos parâmetros cadastrados para o algoritmo já na ordem que devem ser inseridos na execução do algoritmo. Os grupos de valores definidos devem possuir valores numéricos inteiros ou reais separados por espaços. O usuário deve ainda escolher uma pesquisa cadastrada para definição do *benchmark* de teste e a base de cálculo que será utilizada no processamento dos dados podendo ser: valor ótimo, melhor valor, valor mínimo encontrado e valor médio encontrado. A escolha do valor ótimo ou melhor valor somente será possível se estes valores estivem disponíveis nos cadastros das instâncias de teste. Após todas as escolhas definidas o usuário deve clicar no botão "Executar Parametrização do Algoritmo".

[26] http://www.hosda.com.br/analise/*downloadscheduling*
[27] https://cran.r-project.org/bin/windows/base/old/3.2.1/

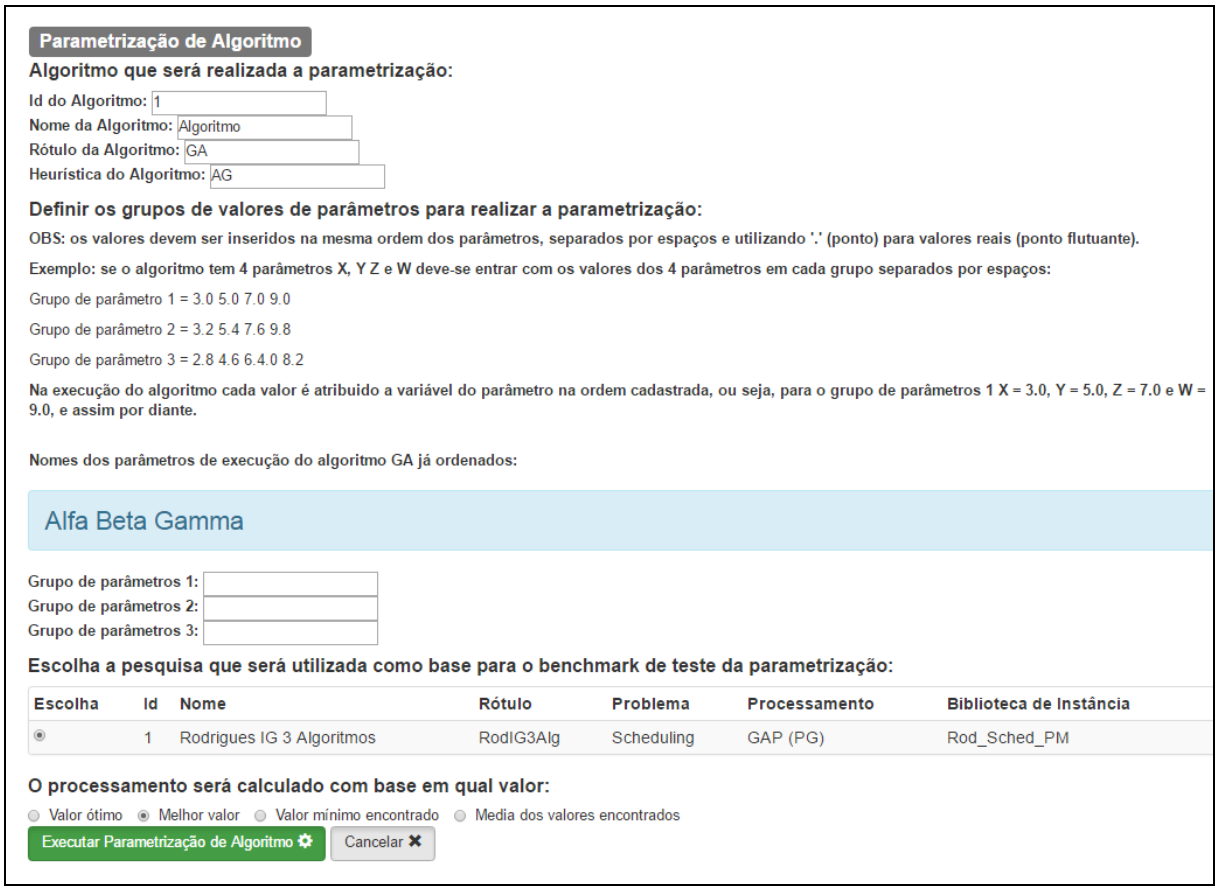


Figura 6.12 - Tela de chamada do processo de parametrização de um algoritmo
Fonte: elaborado pelo autor

O *framework* irá processar a parametrização do algoritmo cadastrado com os três grupos de parâmetros de execução definidos e com todas as instâncias do *benchmark* de teste da pesquisa escolhida. O *framework* gera os arquivos de resultados encontrados e gera um outro arquivo que é o "DadosParametrizacao.txt" que contém informações sobre esta última parametrização realizada para o algoritmo. Após a obtenção dos resultados o *framework* executa automaticamente o processo de análise estatística. O *framework* faz ainda o *download* de todos os resultados encontrados. Se já existirem resultados de parametrizações anteriores estes dados e as análises estatísticas correspondentes a estes resultados serão apagados do sistema.

É possível para o pesquisador recuperar todos os resultados da última parametrização para um algoritmo bastando clicar no subitem "Resultados da Última Parametrização" do item do menu "Parametrização". O usuário deve escolher um algoritmo da lista apresentada e se existir algum parametrização recente, o *framework* faz o *download* dos resultados disponíveis.

6.9 Consideração a respeito do *framework* desenvolvido

Este capítulo apresentou a especificação, o desenvolvimento e o resultado da implementação do *framework* para validação de pesquisas de otimização baseadas em heurísticas para problemas de *scheduling*. O processo de desenvolvimento do *framework* foi definido como iterativo e incremental sendo especificado, além das etapas de definição de requisitos e análise do sistema, quatro iterações principais de desenvolvimento. A primeira iteração, a análises estatísticas dos resultados, foi conseguida através da codificação em linguagem R. O *script* desenvolvido é capaz de: (i) receber os resultados das execuções dos algoritmos para as instâncias especificadas em um *benchmark* de execução, (ii) efetuar todas as análises estatísticas definidas do método de pesquisa e (iii) gerar automaticamente um relatório estatístico com todas as informações tabuladas e um conjunto até quinze (15) gráficos. Com este relatório objetiva-se facilitar a análise de resultados de pesquisa fornecendo assim um conjunto de padrões para a representação de dados.

A segunda iteração preocupou-se com o desenvolvimento da base de funcionamento do *framework*, ou seja, o gerenciamento de todos os cadastros necessários para que uma pesquisa pudesse ser executada e analisada estatisticamente. Nesta etapa foram desenvolvidos as funcionalidades de criação, consulta, alteração e exclusão de objetos das classes de pesquisa, problema, algoritmo, parâmetro, biblioteca de instância, instâncias e usuários. Como decisões de projeto foram utilizados somente *software* livre de código aberto e foi definido que o *framework* seria implantado em um servidor web para acesso irrestrito a outros pesquisadores.

A terceira iteração implementou a execução automática de uma pesquisa, com a chamada dos algoritmos cadastrados junto com seus parâmetros de execução e com as instâncias de execução previamente definidas. Os resultados das execuções são armazenados e processados deixando prontos para a chamada do *script* que processa as análises estatísticas.

A quarta iteração implementou a parametrização simplificada de um algoritmo como um processo de apoio a uma parametrização externa. O *framework* executa o algoritmo com os grupos de parâmetros de teste definidos e com as instâncias de teste especificadas. Os resultados destas execuções são processados e tabulados sendo aplicados automaticamente ao *script* que processa as análises estatísticas,

permitindo que o pesquisador a análise e tomada de decisão e qual grupo de parâmetros é melhor para o algoritmo.

O *framework* foi desenvolvido utilizando *software*s livre de código aberto sendo implantado em um servidor web de forma a disponibilizar suas funcionalidades para outros pesquisadores. Os processos de execução automática dos algoritmos com a tabulação dos resultados, a parametrização simplificada de um algoritmo e a geração de análises estatísticas dos resultados obtidos fornecem ferramentas que podem facilitar o trabalho de um pesquisador no desenvolvimento de pesquisas de otimizadores heurísticos para problemas de engenharia de produção. O foco da tese foi para problemas de programação de tarefas (*scheduling*), mas pode-se perceber que o *framework* pode ser aplicado a uma vasta gama de problemas de otimização. Outro ponto importante é que as informações do relatório estatístico gerado podem ser utilizadas na análise e representação dos resultados obtidos em uma pesquisa, melhorando assim a qualidade das conclusões no trabalho e facilitando a comparação de pesquisas correlatas.

Capítulo 7

TESTES E APLICAÇÃO DO *FRAMEWORK*

Este capítulo apresenta todos os testes de utilização e aplicação do *framework* realizados com o intuito de validar o perfeito funcionamento de todas as funcionalidades descritas anteriormente. Inicialmente foram realizados testes de validação do *script* desenvolvido para a análise estatística de dados. Em seguida foram realizados os testes de integração do sistema onde todas as funcionalidades do *framework* foram verificadas. Finalmente o *framework* é aplicado a um conjunto de dados reais de uma pesquisa desenvolvida na área.

7.1 Validação da análise estatística dos resultados

Para validar o *script* de análise estatística dos resultados desenvolvido na iteração 1 conforme apresentado no capítulo anterior, foram utilizados os conjuntos de dados apresentados nos trabalhos de Rodriguez et al. (2013) e Tavares Neto e Oliveira (2015).

7.1.1 Conjunto de dados de Rodriguez et al. (2013)

O trabalho de Rodriguez et al. (2013) trata o problema de *scheduling* com máquinas paralelas com a função objetivo de minimização do tempo total de finalização, utilizando a meta-heurística *Iterated Greedy (IG)*. Esta meta-heurística realiza um conjunto de iterações com uma heurística construtiva precedida de uma fase de destruição[28]. Segundo os autores, o uso de IG tem solucionado um grande número de problemas, principalmente por ser fácil de implementar e por apresentar

[28] Maiores informações sobre IG podem ser encontrados em Ruiz e Stutzle (2007).

excelente performance. O objetivo do trabalho é aplicar o IG para solucionar o problema de *scheduling* com máquinas paralelas com instâncias de larga escala.

Os autores definem sete conjuntos (UC1, UC2, UC3, UC4, UC5, JC e MC), com dez instâncias do problema, geradas randomicamente através de diferentes maneiras da escolha do tempo de processamento da tarefa j na máquina i (p_{ij}, i = 1,..., n e j = 1,..., m) e seu peso (w_j), totalizando 133 instâncias. Para fazer a comparação, as instâncias definidas são executadas utilizando método exato CPLEX com limitação de tempo de execução em duas horas. Devido a esta limitação, não se tem a solução ótima, mas tem-se uma solução factível para utilização na comparação. Utiliza-se o método não paramétrico de Friedman (FRIEDMAN, 1940) para análise dos dados, com dois métodos alternativos de análise *post-hoc*. O primeiro é a aplicação do teste de Iman e Davenport (IMAN; DAVENPORT, 1980) com o método de Holm´s (HOLM, 1979) e o segundo a utilização do teste de Wilcoxon pareado com ranqueamento de sinais (WILCOXON, 1945).

A meta-heurística IG-RW-D1 proposta por Rodriguez et al. (2013) é comparada com outras quatro abordagens: um método iterativo de multi partida (MultiS), uma busca Tabu (Tabu), um algoritmo genético (GA) e um GRASP com religamento e caminhos (GRASP). Com os valores encontrados para cada instância são calculados as derivações percentuais relativas (RPD - *Relative Percentage Deviation*) que é obtido com o valor médio encontrado da meta-heurística, menos o melhor valor conhecido (CPLEX), dividido pelo melhor valor conhecido e multiplicado por 100. O artigo faz a análise estatística dos resultados e chega à conclusão que o método IG proposto (IG-RW-D1) é o melhor entre todos os comparados, tanto para instâncias pequenas como instâncias grandes.

A partir dos dados apresentados por Rodriguez et al. (2013) para as cinco meta-heurísticas, com as 133 instâncias, estes são aplicados ao *script* em linguagem R de análise estatística proposta nesta tese. Ao adotar o trabalho de Rodriguez et al. (2013) como teste de validação da iteração 1 do desenvolvimento do *framework*, tem-se a intenção de comparar resultados estatísticos validados pela comunidade acadêmica, o que não seria possível com dados gerados exclusivamente para este fim. A proposta do *script* desenvolvido é oferecer aos pesquisadores uma ferramenta capaz de reunir em um só local um conjunto de gráficos e análises estatísticas para fomentar as conclusões baseados nos dados coletados.

O Apêndice A apresenta o relatório gerado automaticamente com os dados das cinco soluções de meta-heurísticas de Rodriguez et al. (2013). Conforme pode ser visto esse relatório apresenta os principais resultados estatísticos e gráficos agrupados em um único documento. No que diz respeito às análises presentes nesse documento, temos (obs.: as referências a figuras e tabelas são relativas ao Apêndice A):

I. Analisando os valores calculados da estatística descritiva nas Tabelas 1 e 2, pode-se observar que os algoritmos MultiS e Tabu são muitos discrepantes, apresentando valores médios e desvio padrão muito elevados em comparação com os outros. A análise neste ponto pode ficar mais focada nos algoritmos GA, GRASP e IGRWD1. A menor média e o menor desvio padrão são apresentados pelo IGRWD1, mas somente com estes valores não é possível definir qual é o melhor, principalmente quando se avalia o valor da kurtosis (grau de concentração dos dados a respeito da média) que possui um valor relativamente baixo indicando uma dispersão dos valores em relação ao valor médio e o valor do erro padrão médio que se aproxima muito do valor obtido para o GA;

II. Analisando o gráfico *boxplot* da Figura 1, primeiro observa-se que o MultS, o Tabu e o GRASP possuem muitos *outliers*, que são pontos fora dos limites, ao contrário do GA e do IGRWD1. Ao observar o mesmo gráfico sem os *outliers* na Figura 2 confirma-se a discrepância dos algoritmos MultS e Tabu, mas fica a existência de dúvidas na comparação com os outros três algoritmos;

III. O gráfico da Figura 3 mostra que o algoritmo IGRWD1 apresenta moda igual a zero com uma maior frequência entre os três melhores. Ao observar somente os valores das médias, inicialmente o GA ganharia do GRASP. Entretanto ao analisar este gráfico os papéis se invertem, pois o GRASP apresenta um valor de moda próximo a zero com uma frequência muito maior que o GA;

IV. O gráfico da Figura 5 apresenta os valores de *fitness* pelas instâncias. Observa-se que o GA e o IGRWD1 apresentam respostas mais comportadas em relação as instâncias e que o GRASP, apesar de ter vários valores próximos a zero, em algumas instâncias, apresenta valores discrepantes. Analisando os dados de entrada chega-se a conclusão que

para instâncias muito grandes a solução do GRASP parece não fornecer boas respostas;

V. Tanto o gráfico de coordenadas paralelas na Figura 7 como gráfico de dispersão na Figura 8 podem levar o pesquisador a tirar conclusões precipitadas. Neste caso, definindo os melhores algoritmos como sendo GA e IGRWD1 e que o IGRWD1 ganharia do GA. Mas é necessária uma análise mais detalhada para confirmar estas hipóteses;

VI. Analisando os testes de normalidade, os gráficos de histograma e os gráficos de comparação quartil-quartil das amostras (item 3 do relatório), percebe-se claramente que as amostras dos dados dos algoritmos comparados não seguem uma distribuição normal, enviesando assim a aplicação de uma análise paramétrica dos dados como ANOVA. Neste sentido, o melhor seria partir para uma análise não paramétrica usando Friedman conforme definido no método de pesquisa;

VII. Os Gráficos 10 e 11 apresentam o *boxplot* das diferenças entre os algoritmos. Quando se avalia as diferenças entre MultS e Tabu com os outros três algoritmos (MultS-GA, MultS-GRASP, MultS-IGRWD1, Tabu-GA, Tabu-GRASP e Tabu-IGRWD1) verifica-se que apesar da mediana estar próxima do zero existe uma grande variabilidade para os valores, indicando que são algoritmos com respostas bem diferentes. A diferença de Tabu-MultS possui valores muito próximos de zero e com baixa variabilidade indicando que estes dois algoritmos são praticamente iguais. Nas diferenças entre GRASP-GA, IGRWD1-GA e IGRWD1-GRASP observa-se certa semelhança entre os algoritmos, tendendo para uma igualdade maior entre IGRWD1 e GA, mas ainda não é possível confirmar uma conclusão definitiva;

VIII. Mesmo indicando que os dados dos algoritmos não seguem uma distribuição normal, o teste ANOVA com um fator e blocos com análise *post-hoc* foi executado. O resultado geral do ANOVA (item 5.1) comprova que pelo menos um dos algoritmos é diferente através da análise do p-valor menor que 2e-16. Neste exemplo a análise dos residuais do ANOVA não contribui muito para a análise. Após verificado que existem diferenças entre os algoritmos a análise *post-hoc* é necessária para tentar identificar qual seria o melhor. A primeira técnica utilizada é o teste LSD que neste caso

calcula um valor LSD e classifica os algoritmos em grupos distintos em termos do somatório dos ranqueamentos obtidos. Esta classificação em grupos permite uma análise mais detalhada para identificar qual seria o melhor algoritmo. A Figura 15 apresenta o gráfico de barras com os agrupamentos dos algoritmos. Percebe-se que o Mults e Tabu foram classificados em um grupo denominado de "a", indicando que são parecidos e possuem somatório dos ranques bem elevado. Os outros três algoritmos, GRASP, GA e IGRWD1 foram classificados em outro grupo denominado de "b" indicando que o comportamento dos algoritmos são próximos e que o menor somatório de ranque apresentado seria para o IGRWD1, mas com valor muito próximo ao GA. As análises posteriores de Tukey HSD e teste-t pareado somente serviriam para a confirmação destes valores e para uma análise mais criteriosa. O problema desta análise é que o teste ANOVA requer premissas que não são satisfeitas com os dados analisados, tal qual o critério de normalidade das amostras. Por este motivo considerar os resultados do teste ANOVA poderia levar a um erro na análise final e portanto deve ser descartado;

IX. Como os dados das amostras não seguem uma distribuição normal, o correto é aplicar um teste não paramétrico que neste caso é o teste de Friedman com análise *post-hoc* (item 6 do relatório). O resultado geral do teste de Friedman (item 6.1) comprova que pelo menos um dos algoritmos é diferente através da análise do p-valor menor que 2.2e-16. Após verificado que existem diferenças entre os algoritmos a análise *post-hoc* é necessária para tentar identificar qual seria o melhor. A primeira técnica utilizada também é o teste LSD que classifica os algoritmos em grupos distintos em termos do somatório dos ranqueamentos obtidos. A Figura 16 apresenta o gráfico de barras com os agrupamentos dos algoritmos. Percebe-se que o Mults, Tabu e GA foram classificados em um grupo denominado de "a", indicando que são parecidos e possuem somatório dos ranques bem elevado. O GRASP foi classificado em outro grupo denominado de "b" e o IGRWD1 foi classificado em outro grupo denominado de "c". Isto indica que o menor somatório de ranque apresentado seria para o IGRWD1 e consequentemente poderia ser considerado o melhor algoritmo entre os cinco avaliados na pesquisa. As análises *post-hoc* posteriores de teste de

Simetria e teste de Nemenyi apenas serviriam para confirmar estes resultados.

Os resultados apresentados no relatório estatístico do Apêndice A confirmam as conclusões do trabalho de Rodriguez et. al (2013), que consideram a solução heurística IG-RW-D1 desenvolvida pelo autor é melhor que as outras quatro soluções utilizadas na comparação, sendo comprovadas as conclusões finais apresentadas no artigo, conforme Tabela 7.1 abaixo.

Tabela 7.1 - Tabela comparativa dos resultados obtidos

Meta-heurística	(RODRIGUEZ et al., 2013)		***Framework* proposto**	
	Ranqueamento geral utilizando teste de Holm´s Obs.: (+) significa que existe diferença significativa		Somatório do ranqueamento geral utilizando Fisher´s LSD Alpha = 0,05 t-Student = 1,964467 LSD = 41,19823	
	Valor	Classificação	Valor	Classificação
IG-RW-D1	1,64	Ganhador	218,0	C
GRASP	2,51	+	334,0	B
GA	3,49	+	471,5	A
MultS	3,65	+	482,0	A
Tabu	3,71	+	489,5	A

Fonte: elaborado pelo autor com base no trabalho de (RODRIGUEZ et al., 2013)

Tal resultado confirma a validade do *script* desenvolvido para o *framework* já que a análise dos resultados das execuções dos algoritmos alcançou os mesmos resultados, ou seja, a solução IG-RW-D1 é considerada a melhor solução entre todas as comparadas.

7.1.2 Conjunto de dados do trabalho Tavares Neto e Oliveira (2015)

O trabalho de Tavares Neto e Oliveira (2015) trata do problema integrado de programação de tarefas (*scheduling*) e roteirização de veículos com o objetivo de redução do tempo total de fluxo em sistemas integrados produção-distribuição, ou seja, busca minimizar o tempo de entrega do produto para o cliente. A descrição do problema estudado indica dois dos subsistemas de produção (um ambiente de máquina única) e distribuição (um único veículo capacitado capaz de múltiplas rotas). É apresentado um algoritmo construtivo baseado no NEH (NAWAZ et al., 1983) com um conjunto de oito procedimentos de inicialização. Esses oito algoritmos são aplicados em um conjunto de 2.430 instâncias de teste contendo cinco ordens de serviço e comparados com a solução ótima. Em instâncias maiores, os oito algoritmos

de inicialização têm seu comportamento avaliado comparativamente com o uso de 12.150 instâncias de teste de ordem maior ou igual.

Este trabalho propõe um algoritmo com dois estágios para a solução do problema. Os estágios são:

I. Fase de ordenação - Nessa fase, o conjunto de trabalhos a serem processados é colocado em uma lista L ordenada segundo uma regra específica. Foram utilizadas sete regras, gerando assim sete algoritmos distintos, a saber:
   - FIFO – *First In*, *First Out* (primeiro a chegar é o primeiro a sair): essa regra não faz nenhum tipo de ordenação;
   - SPT – *Shortest Processing Time* (menor tempo de processamento): ordena as tarefas conforme seu tempo de processamento, em ordem crescente;
   - LPT – *Longest Processing Time* (maior tempo de processamento): ordena as tarefas conforme seu tempo de processamento, em ordem decrescente;
   - SIZE: ordena as tarefas por tamanho, em ordem crescente;
   - SIZEDEC: ordena as tarefas por tamanho, em ordem decrescente;
   - NNSETUP: ordena as tarefas através de um algoritmo padrão de caminhos mínimos (onde as distâncias entre duas ordens são dadas pelos tempos de *setup* respectivos);
   - NNDIST: ordena as tarefas através de um algoritmo padrão de caminhos mínimos.

II. Fase de inserção - A fase de inserção da heurística proposta é muito semelhante ao NEH: inicialmente, o primeiro trabalho da lista ordenada é inserido na sequência produtiva final, sendo inserida em uma rota vazia. Depois, cada tarefa é inserida na sequência de produção (como o NEH) e, para cada posição, testa-se cada posição de cada rota na sequência de distribuição. Além disso, também são testadas novas rotas no início, entre as rotas e após todas as rotas. Como o NEH, cada tarefa é alocada nas posições que melhor satisfazem a função objetivo do problema.

Para análise da resposta dos algoritmos, um conjunto extensivo de 12.150 instâncias foi criado da seguinte forma:

- O número de tarefas para cada instância é n = [5, 10, 20, 40, 80];
- Uma função randômica uniforme foi usada para determinar os valores para as variáveis do problema definido:
    - Produção:
        - Tempo de Produção: ρ_i = [1, 100];
        - Tempo de *setup*: é criado um espaço θ_s x θ_s, $\theta_s \in$ [10, 50,100] e as ordens são indexadas em uma posição aleatória nesse espaço. O tempo de *setup* κ_{ij} entre duas ordens é dado pela distância euclidiana entre as ordens;
    - Distribuição:
        - Distância entre os pontos de entrega de duas ordens δ_{ij}: é obtida usando o espaço θ_s x θ_s, $\theta_s \in$ [5, 10, 30]. A distância entre as ordens e a origem é multiplicada por um fator $\theta_g \in$ [1, 10, 30];
        - O tamanho de cada ordem: σ_i = [5, 10, 20];
        - A capacidade de cada caminhão: ψ = { max(σ_i), 1 . θ_s }.

A análise dos algoritmos teve que ser realizada em duas etapas devido ao fato de não ser possível obter valores ótimos para instâncias com 10 ou mais ordens de serviço. Na primeira etapa foram utilizados 2.430 instâncias de 5 ordens de serviço onde obteve-se a resposta ótima através de um modelo de programação linear mista. Na segunda etapa não foi possível a obtenção de valores ótimos para instâncias maiores, então os diferentes algoritmos foram comparados entre si com todo o conjunto de instâncias definidos. Em ambas as análises, para cada uma das instâncias, calculou-se o valor do desvio percentual entre a melhor resposta obtida e a resposta do algoritmo. Esse desvio é indicado por *gap = ((F - F*)/F**) x 100, onde *F* é a solução obtida pelo algoritmo em questão e *F** a melhor solução encontrada para a instância analisada.

Foram realizados ainda dois tipos de análises dos resultados sendo a primeira com o tratamento dos dados obtidos de forma agregada utilizando para isto os resultados apresentados no relatório estatístico gerado pelo *script* desenvolvido. A segunda, uma análise gráfica baseada em gráficos *box-plot* com agrupamento dos dados em função de cada parâmetro de geração das instâncias.

Os resultados encontrados das execuções dos algoritmos para cinco ordens de serviço foram tabulados e foram submetidos ao *script* de análise estatística

desenvolvido onde também foi definido que as amostras não seguiam uma distribuição estatística normal. O relatório estatístico apresentou as análises do teste de Friedman juntamente com análise post-hoc com o teste de Nemenyi (Tukey) apresentado na Tabela 7.2. O teste de Friedman informa que existem diferenças significativas entre as soluções e com a análise post-hoc pode-se identificar qual a melhor solução. Nesta situação, com 5 ordens de serviço, foi a solução SPT.

Tabela 7.2 - Resultado da análise *post-hoc* com teste de Nemenyi (Tukey) com 5 ordens de serviço (n=5)

	FIFO	LPT	NNDIST	NNSETUP	SIZE	SIZEDEC
LPT	0,999					
NNDIST	0,006	0,003				
NNSETUP	0,669	0,532	0,555			
SIZE	0,513	0,379	0,709	0,999		
SIZEDEC	0,997	0,988	0,063	0,965	0,905	
SPT	1,58E-11	1,1E-11	0,008	2,82E-06	4,47E-06	8,16E-10

Fonte: gerado automaticamente pelo *script* de análise estatística de resultados

Realizando a análise gráfica dos resultados percebe-se que, no caso da variação da distribuição do tempo de *setup*, apesar dos resultados serem muito próximos, com os valores médios, o algoritmo SPT obteve melhores resultados nos três conjuntos, conforme pode ser visto na Figura 7.1.

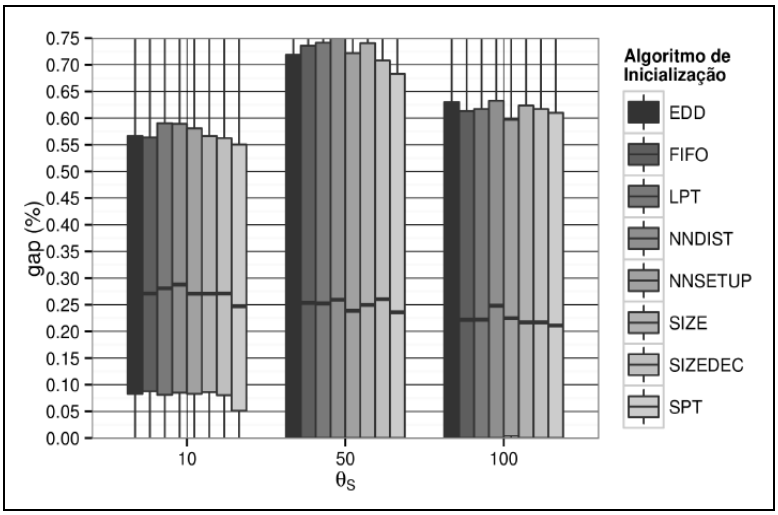


Figura 7.1 - Gap para problemas de tamanho n=5, agrupados conforme *θs* (valores de referência encontrados através de modelo de programação inteira mista)
Fonte: (TAVARES NETO; OLIVEIRA, 2015)

Para instâncias com 10 ou mais ordens de serviço, como não seria possível obter os valores ótimos, optou-se por realizar uma análise comparativa incluindo apenas as heurísticas desenvolvidas, com o intuito de identificar qual das estratégias de inicialização elencadas se destaca frente às demais. Procedendo da mesma forma como descrito anteriormente, os resultados encontrados das execuções dos algoritmos foram tabulados e submetidos ao *script* de análise estatística desenvolvido,

onde também foi definido que as amostras não seguiam uma distribuição estatística normal. O relatório estatístico apresentou as análises do teste de Friedman juntamente com análise post-hoc com o teste de Nemenyi (Tukey) apresentado na Tabela 7.3. Ao contrário do que ocorreu na Tabela 7.2, percebe-se nesse caso que o algoritmo NNSETUP é, de forma geral, melhor que o algoritmo SPT. A Figura 7.2 permite analisar a relação entre o *gap* e o algoritmo, também conforme o tempo de *setup*, onde se percebe que o algoritmo NNSETUP é melhor que as outras soluções.

Tabela 7.3 - Resultado da análise *post-hoc* com teste de Nemenyi (Tukey) para todas as instâncias

	FIFO	LPT	NNDIST	NNSETUP	SIZE	SIZEDEC
LPT	0,198					
NNDIST	0,999	0,165				
NNSETUP	0,000	0,000	0,000			
SIZE	0,992	0,021	0,996	0,000		
SIZEDEC	0,831	0,002	0,869	0,000	0,998	
SPT	1,166E-08	0,000	1,302E-08	0,000	1,67E-06	7,105E05

Fonte: gerado automaticamente pelo *script* de análise estatística de resultados

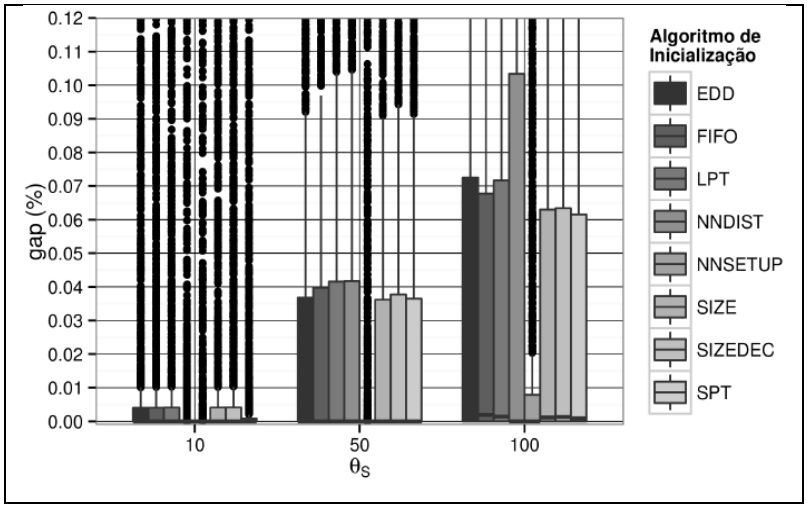


Figura 7.2 - Gap por algoritmo de inicialização, agrupados conforme *θs* (valores de referência obtidos apenas pelos resultados dos 8 algoritmos)
Fonte: (TAVARES NETO; OLIVEIRA, 2015)

De acordo com as análises, o algoritmo inicializado através da regra NNSETUP se mostrou mais adequando que os demais para a resolução do problema proposto, mesmo que para uma limitação de cinco (5) ordens de serviço o algoritmo SPT tenha sido melhor. Os resultados estatísticos apresentados no relatório final gerado pelo *script* desenvolvido auxiliaram as conclusões do trabalho.

7.2 Teste de integração do *framework*

Para validar o funcionamento do *framework* fazendo um teste de integração do sistema, foi utilizado o mesmo conjunto de dados apresentados no trabalho de Rodriguez et al. (2013). Como as instâncias de execução e os algoritmos não estavam disponíveis, forma criados artifícios para que o *framework* pudesse ser executado:

- O trabalho faz a comparação da solução IGRWD1 com mais quatro (4) outras soluções, sendo que analisando o relatório apresentado no item anterior os algoritmos MultS e Tabu são muito inferiores as outras soluções. Neste sentido no teste de integração foram definidos os algoritmos GA, GRASP e a solução IGRWD1 proposta pelo autor;
- Dos sete conjuntos de instâncias, totalizando cento e trinta e três (133), foram escolhidos dois conjuntos o UC1 e JC, que totalizam trinta e oito (38) instâncias, que foram cadastradas no *benchmark* de execução da pesquisa;
- Para cada uma das trinta e oito (38) instância escolhidas foram criados arquivos texto contendo o melhor valor, calculado pelo método CPLEX, e os valores encontrados na pesquisa para os algoritmos GA, GRASP e IGRWD1, separados por linhas;
- Das trinta e oito (38) instâncias foram selecionadas dez (10) para serem definidas como *benchmark* de teste da pesquisa;
- Foram criados arquivos executáveis para cada algoritmo que liam o arquivo da instância especificada na linha de comando, um tempo aleatório qualquer era gasto simulando o tempo de execução da solução e retornando ao final o valor encontrado para o algoritmo e o tempo total de execução.

Todos os dados da pesquisa de Rodriguez et al. (2013) foram cadastrados no *framework* e podem ser consultados[29]. A partir deste cadastro as principais funcionalidades do *framework* podem ser executadas.

Para parametrizar um algoritmo pode-se escolher um algoritmo já cadastrado (GA, GRASP ou IGRWD1), definir os três grupos de teste de parâmetros, escolher qual a pesquisa será usada para o *benchmark* de teste (RodIG3Alg), definir qual será a base de cálculo de processamento e solicitar a parametrização. Quando a

[29] http://www.hosda.com.br/pesquisa/detail/1/

parametrização é solicitada o algoritmo é executado com cada grupo de teste de parâmetro e com cada uma das dez instâncias de teste definidas. Como as execuções do algoritmo são simuladas a variação dos parâmetros de teste não altera os resultados encontrados, mas o *framework* consegue executar todo o processo retornando os resultados encontrados e o relatório estatístico dos resultados.

Para a execução da pesquisa primeiro escolhe-se a pesquisa em questão (RodIG3Alg) e depois qual será a base de cálculo do processamento dos resultados. Quando a execução é solicitada cada algoritmo cadastrado na pesquisa juntamente com os seus parâmetros de execução é executado com as instâncias definidas no *benchmark* de execução, retornando os resultados encontrados. Nesta simulação o *framework* consegue retornar os resultados exatamente como apresentados no trabalho de Rodriguez et al. (2013).

Após a execução pode-se solicitar o processamento da análise estatística de resultados da pesquisa. Primeiro deve ser escolhido a pesquisa que se deseja realizar a análise e depois o pesquisador deve definir se os resultados encontrados seguem uma distribuição estatística normal. Se a pesquisa não tiver sido executada ainda, não existem os arquivos de resultados e, portanto, a análise não será realizada. Se houver uma execução recente da pesquisa, o arquivo de resultados processados é submetido ao *script* que gera o relatório estatístico completo destes resultados.

No exemplo definido, foi realizada a análise estatística da pesquisa de Rodriguez et al. (2013) que já estava cadastrada no *framework*, sendo definido que o arquivo de resultados (RodIg3Alg.txt) não seguia uma distribuição estatística normal. Sendo assim o *script* de análise estatística executou todos os testes definidos e o teste de Friedman com as análises post-hoc utilizando Teste LSD (*Least Significant Difference* - Menor diferença significante), teste de simetria e teste de Nemenyi. A Figura 7.3 apresenta o gráfico com os agrupamentos que foram obtidos através das múltiplas comparações dos tratamentos do teste LSD para o teste de Friedman. Este gráfico foi retirado do relatório estatístico gerado automaticamente pelo *script* e confirma as hipóteses do autor que indicando que a solução proposta (IGRWD1) é a melhor na comparação com as outras soluções.

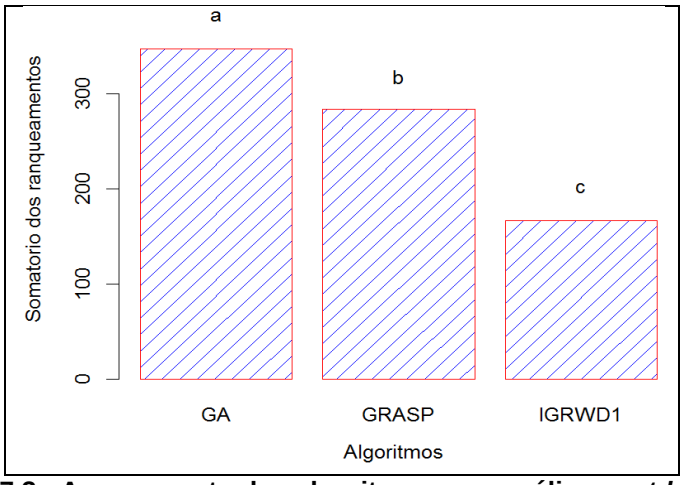


Figura 7.3 - Agrupamento dos algoritmos com análise *post-hoc* LSD
Fonte: gerado automaticamente pelo *script* de análise estatística de resultados

A massa de dados criada com base no trabalho de Rodriguez et al. (2013) serviu para que o *framework* fosse testado integralmente, com todas as suas funcionalidades.

7.3 Teste de aplicação do *framework* com dados reais de uma pesquisa

Para que se possa garantir que o *framework* funcione com dados reais foi realizado um teste de aplicação. Para tal, foram analisadas heurísticas criadas para a resolução do problema de planejamento integrado entre produção por múltiplas máquinas paralelas e distribuição por único veículo capacitado (*Integrating Production by Multiple Parallel Machines and Distribution by a Single Capacitaded Vehicle: Models and Algorithms*). Essa pesquisa está em fase de análise em periódico internacional.

7.3.1 Descrição do trabalho

O trabalho em questão aborda o problema da integração da programação de tarefas da produção, estoques e distribuição (*Integrated Scheduling of Production, Inventory and Distribution Problems* - ISPIDP) onde a produção é representada por um ambiente de máquinas paralelas e a distribuição é realizada por um único veículo capacitado com várias rotas. Analisando as questões práticas comumente encontradas em cenários do mundo real foi considerado que os tempos de preparação no local de produção são dependentes da sequência. Buscou-se minimizar o

makespan de todo o sistema. Quatro abordagens são utilizadas neste trabalho: um modelo de programação inteira-mista (MIP), uma heurística construtiva (inspirada no NEH), um procedimento de Algoritmo Genético (GA) e duas heurísticas baseadas em *Iterated Greedy* (IG e IGR) propostas e implementadas pelos autores.

O algoritmo IG é uma implementação do *Iterated Greedy* onde a fase de destruição remove um número fixo de tarefas randômicas da solução corrente. O Algoritmo IGR seleciona uma rota randômica e remove todas as tarefas da rota. Este passo continua até que o número pretendido de tarefas a serem removidas seja alcançado. Estes algoritmos requerem a especificação de três parâmetros:

- O procedimento de ordenamento usado pela inicialização do algoritmo: as possibilidades consideradas foram as regras FIFO, SPT, LPT, NNSETUP, NNDIST, SIZE e SIZEDEC, conforme descrito no trabalho de Tavares Neto e Oliveira (2015) apresentado no item 7.1.2;
- O número de tarefas a serem destruídas na fase de destruição. As possibilidades são 20%, 40%, 60%, 80% e 100% do número total de pedidos;
- O número de ciclo de destruição-reconstrução executado pelo algoritmo. No caso foram considerados {1; 5; 10; 20} x número de pedidos.

Para achar a combinação mais adequada para os parâmetros de configuração, a pesquisa em questão utilizou do pacote IRACE (LOPEZ-IBANEZ et al., 2011) para a identificação da melhor combinação para estes parâmetros. O resultado deste processo automatizado de parametrização definiu a inicialização NNSETUP como sendo a melhor, removendo 20% dos pedidos e executando o número de ciclos igual ao número de pedidos da instância analisada. Este conjunto de parâmetros foi obtido para os dois algoritmos propostos IG e IGR.

7.3.2 *Benchmark* utilizado

Para analisar o desempenho dos algoritmos foi adotado o conjunto de doze mil cento e cinquenta (12.150) instâncias geradas aleatoriamente (TAVARES NETO; OLIVEIRA, 2015). O modelo de programação inteira-mista (MIP) obteve a soluções ótimas para o subconjunto de duas mil quatrocentos e trinta (2430) instâncias, geradas com cinco (5) tarefas, não sendo possível a obtenção destes valores para os outros subconjuntos.

7.3.3 Execução da pesquisa

Para a execução da pesquisa foram selecionadas quatro abordagens para serem comparadas, sendo a primeira a heurística construtiva (NEH) com inicialização do método NNSETUP e as outras são heurísticas baseadas em *Iterated Greedy* (IG) com os métodos de inicialização NNSETUP, NNDIST e SPT. Como base de cálculo do processamento dos dados gerado pelo *framework* foi definido o valor ótimo de cada instância, que estão disponíveis para o subconjunto de duas mil quatrocentos e trinta (2430) instâncias gerado a partir de cinco tarefas. Deste subconjunto foram escolhidos um mil e oitenta (1080) instâncias para o *benchmark* de execução, limitando o tempo de *setup* a [50, 100] e distância entre os pontos de entrega a [10, 30]. Para o *benchmark* de teste utilizado no processo de parametrização simplificado foi definido um subconjunto de trinta e seis (36) instâncias limitado a variação aleatória de final quinze (15) de cada grupo de instâncias geradas.

Os parâmetros de execução das heurísticas comparadas são a função objetivo (*fit* = *makespan*), o número de máquinas (1) e o número de veículos (1). As variações da solução IG (IG-NNSETUP, IG-NNDIST e IG-SPT) possuem ainda mais dois parâmetros que são o número de ciclos de execução (*nOfCycles*) do IG e número de destruições (*numberToDestroy*) das soluções. Foi realizada uma parametrização de cada uma das abordagens IG com o intuito de descobrir qual os melhores valores para estes dois últimos parâmetros. Foram definidos três grupos de parâmetros de teste:

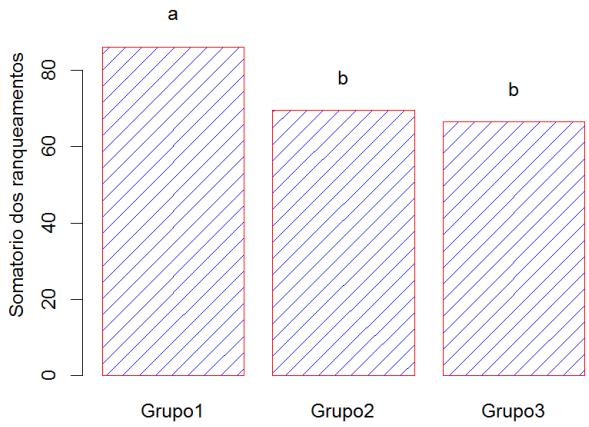
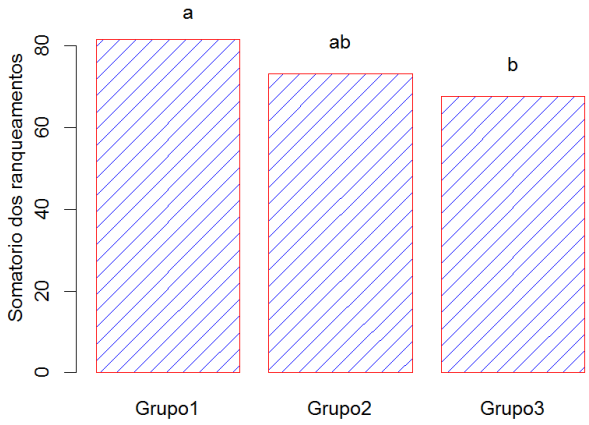
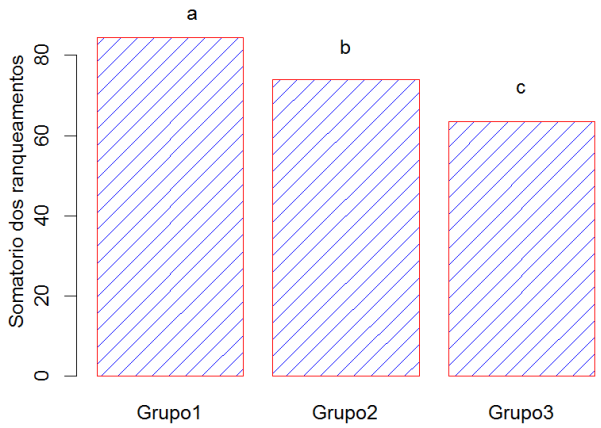
- Grupo 1 de teste: *nOfCycles* = 10 e *numberToDestroy* = 1;
- Grupo 2 de teste: *nOfCycles* = 25 e *numberToDestroy* = 2;
- Grupo 3 de teste: *nOfCycles* = 50 e *numberToDestroy* = 5;

Cada uma das abordagens IG foram executadas com cada grupo de parâmetros e com cada uma das 36 instâncias definidas no benchmark de teste. Os resultados encontrados destas execuções foram aplicados ao *script* de análise estatística que gerou os relatórios estatísticos. Todos os resultados do processo de parametrização estão disponíveis no *framework* para consultas[30]. O Quadro 7.1 apresenta uma sumarização dos dados estatísticos gerados pela execução do *script* para cada uma das abordagens com a execução do teste de Friedman com a análise post-hoc utilizando o teste LSD. Percebe-se que os p-valores encontrados de todas

[30] http://www.hosda.com.br/parametrizacao/resultados

as abordagens são menores que o nível de significância estatística definido (0,05). Isto significa que pelo teste de Friedman, existem diferenças significativas entre as abordagens.

Quadro 7.1 - Resultados da parametrização simplificada das abordagens IG

Solução	Teste de Friedman p-valor	Valor LSD	Grupo	Soma de ranques	M	Resultado Gráfico
IG - NNDIST	0.0001477	8,82	1	86,0	a	
			2	69,5	b	
			3	66,5	b	
IG - NNSETUP	0.01869	9.55	1	81,5	a	
			2	73,0	A b	
			3	67,5	b	
IG - SPT	0.0005673	9,81	1	84,5	a	
			2	74,0	b	
			3	63,5	c	

Fonte: elaborado pelo autor com informações extraídas dos relatórios estatísticos

A análise post-hoc com o teste de LSD classifica cada uma das abordagens usando letras de acordo com o somatório dos ranques e com o próprio valor LSD calculado. Percebe-se que para as abordagens IG-NNSETUP e IG-SPT o grupo de parâmetros 3 foi o melhor. Para o IG-NNDIST a diferença do somatório dos ranques entre o grupo 2 e o grupo 3 é menor que o valor LSD calculado. Por este motivo, os dois grupos foram colocados na mesma classificação (b), o que significa que são equivalentes. Este processo permitiu definir os valores de parâmetros de execução das soluções propostas, que foi o grupo 3 com os valores de parâmetros *nOfCycles* = 50 e *numberToDestroy* = 5.

As quatro abordagens definidas foram então executadas com os parâmetros de execução encontrados e com cada uma das instâncias definidas no *benchmark* de execução. Todos os resultados do processo de execução estão disponíveis no *framework* para consultas[31]. Os resultados foram aplicados ao *script* de análise estatística que gerou o relatório final da análise de comparação entre abordagens. A execução do teste de Friedman forneceu um p-valor menor que 2.2e-16, o que significa a existência de diferenças significativas entre as soluções comparadas. Isto principalmente ocorreu porque as soluções IG foram comparadas com uma solução heurística de menos eficiente que é o NEH. Na análise post-hoc com o teste LSD o valor LSD encontrado foi de 43.143, o que coloca as soluções IG-SPT, IG-NNDIST e IG-NNSETUP em uma mesma classificação (b). Todos os resultados da análise estatística processo de execução estão disponíveis no *framework* para consultas[32]. A Tabela 7.4 apresenta a classificação de cada uma das abordagens de acordo com o somatório dos ranques usando análise *post-hoc* com teste LSD, o que também pode ser visto na Figura 7.4.

As conclusões destes resultados encontrados são:

1. As variações da abordagem IG são superiores em termos de desempenho que a heurística construtiva NEH;
2. Para o IG-NNDIST, de acordo com o resultado da parametrização, poder-se-ia ter utilizado os valores de parâmetros de *nOfCycles* = 25 e *numberToDestroy* = 2, reduzindo assim o esforço computacional;
3. As abordagens IG-NNDIST, IG-NNSETUP e IG-SPT são equivalentes, ou seja, pela análise estatística com os *benchmarks* definidos não se pode

[31] http://www.hosda.com.br/execucao/resultados
[32] http://www.hosda.com.br/analise/estatisticas

especificar qual é a melhor abordagem, mesmo que o somatório dos ranques tenha sido levemente menor para o IG-NNSETUP.

Tabela 7.4 - Classificação das abordagens pela análise *post-hoc* LSD

	Abordagem	Soma dos ranques	Classificação
1	NEHTP	4253.0	a
2	IG -SPT	2199.0	b
3	IG-NNDIST	2174.5	b
4	IG-NNSETUP	2173.5	b

Fonte: gerado automaticamente pelo *script* de análise estatística de resultados

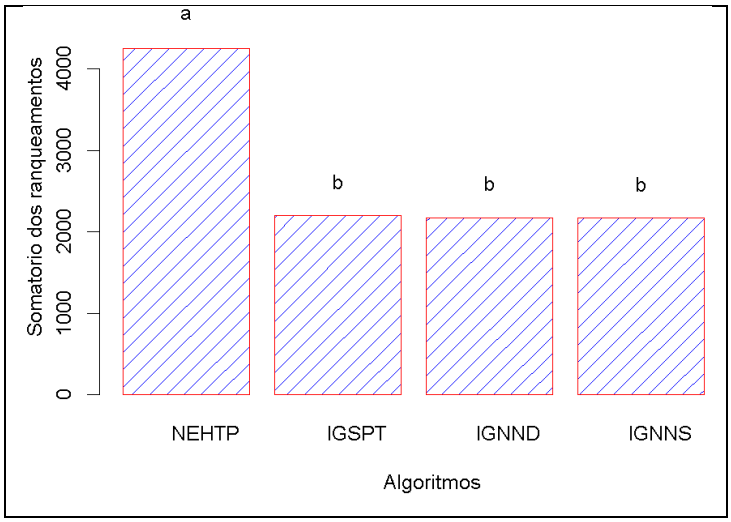


Figura 7.4 - Classificação das abordagens pela análise *post-hoc* LSD
Fonte: Gerado automaticamente no relatório estatístico

7.4 Consideração sobre o teste de utilização do *framework*

O *framework* foi desenvolvido em iterações como mostrado no Capítulo 6, onde na primeira iteração foi desenvolvido o *script* que realiza a análise estatística dos resultados e nas outras três iterações o *framework* foi codificado integralmente. Os testes de utilização seguiram a mesma sequência do desenvolvimento, e neste sentido foi definida a realização de três etapas para os testes completos.

Na primeira etapa foram realizados testes com o *script* de análise estatística. Para tanto foram utilizados dois trabalhos já publicados. No primeiro trabalho (RODRIGUEZ et al., 2013), os resultados apresentados foram aplicados ao *script* de forma a validar as decisões de análise e as conclusões alcançadas pelos autores. No segundo trabalho (TAVARES NETO; OLIVEIRA, 2015) o relatório estatístico gerado pelo *script* foi utilizado na análise dos resultados fornecendo informações importantes para as conclusões alcançadas.

Na segunda etapa, com o desenvolvimento do *framework*, foram realizados testes de utilização de cada iteração desenvolvida. Ao término da implementação do *framework*, foram realizados testes de integração do sistema com o intuito de testar cada uma das funcionalidades especificadas. Para isto foi necessário a utilização de um conjunto de dados disponíveis onde fossem conhecidas todas as análises e conclusões finais. Foram utilizados os resultados do trabalho apresentado por Rodriguez et al. (2013), onde a pesquisa desenvolvida pelos autores foi cadastrada no sistema. Utilizou-se alguns artifícios para simular as instâncias do problema e os algoritmos que seriam executados, já que os mesmos não estavam disponíveis. Este conjunto de resultados permitiu que todas as funcionalidades do *framework* fossem testadas integralmente.

Na terceira etapa foi necessária a aplicação do *framework* com um conjunto de dados reais. Para tal, foram utilizadas informações e heurísticas provenientes de um trabalho em fase de publicação do grupo de pesquisa. Foram definidas quatro abordagens algorítmicas e um conjunto de instâncias de teste. Todos os dados da pesquisa foram cadastrados no framework, realizando o processo de parametrização simplificado, a execução das abordagens com as instâncias e geração automática do relatório estatístico dos resultados encontrados, permitindo validar a aplicação do framework em uma situação real de pesquisa.

Os testes realizados comprovam a disponibilidade, a facilidade e a usabilidade do framework desenvolvido. Os relatórios estatísticos gerados pelo *script* foram capazes de apoiar nas análises dos resultados e construção das conclusões. Com os dados da pesquisa cadastrados no framework foi possível fazer vários testes de parametrização e execução das soluções, inclusive sendo embasado pelas análises estatísticas dos resultados.

Capítulo 8

CONSIDERAÇÕES FINAIS

Pesquisas que tratam de problemas de sistemas de produção enfrentam desafios na busca por soluções otimizadas. Muitos problemas não podem ser resolvidos por técnicas de otimização matemática clássica devido a sua complexidade e o custo de se analisar o espaço de soluções viáveis. Devido a isto, se percebe um crescimento na utilização de heurísticas e meta-heurísticas para solucionar tais problemas com intuito de alcançar soluções de alta qualidade sem garantia de soluções ótimas mas, que oferecem uma compensação razoável entre tempo de execução e qualidade da solução encontrada.

De acordo com a literatura, a utilização de heurísticas e meta-heurísticas para solucionar tais problemas é bastante vasta. Entretanto, muitos trabalhos são desenvolvidos especificamente para solucionar uma única classe de problemas ou que utiliza somente um tipo de modelagem heurística com variações. Isto ocorre por que é muito difícil que exista uma ferramenta ou um *framework* universal que consiga solucionar qualquer classe de problema. Isto leva ao desenvolvimento de soluções específicas e direcionadas para a pesquisa em questão. Muitas vezes faz com o pesquisador fique focado somente no método heurístico na busca de soluções para o problema estudado deixando de lado o processo de análise e representação dos resultados, principalmente a comparação de resultados com a utilização de métodos estatísticos.

Alguns *frameworks* que utilizam heurísticas e meta-heurísticas são bibliotecas de classes utilizadas para a extensão de soluções ou métodos previamente definidos. Outros *frameworks* são implementações de *software* completas que agregam inúmeras funcionalidades e componentes heurísticos. Observa-se que poucos possuem recursos ou implementações com foco na análise estatística dos resultados, mas existe um movimento na direção de se realizar, cada vez mais, análises

estatísticas mais complexas. A qualidade das conclusões da pesquisa somente poderá ser alcançada com métodos estatísticos inferenciais paramétricos e não paramétricos.

No levantamento realizado de *frameworks* percebe-se que existem muitas iniciativas de vários pesquisadores na busca pelo desenvolvimento de uma ferramenta que realmente possa facilitar o processo. Mas fica claro que estes trabalhos estão focados ou nos problemas que solucionam ou nos métodos heurísticos que implementam, deixando de lado a análise estatística dos dados.

Como visto, existem muitos frameworks que desenvolvem soluções heurísticas e meta-heurísticas na solução de problemas de otimização, mas poucos destes possuem recursos de análise estatística. Desta forma pode-se observar uma lacuna nos trabalhos existentes na literatura, que disponibilizam recursos de análise estatística e representação dos resultados, justificando assim a construção desta tese. Neste sentido, o objetivo principal foi especificar um processo de desenvolvimento de pesquisas que buscasse desenvolver otimizadores baseados em heurísticas e meta-heurísticas para problemas de programação da produção (*scheduling*), sendo validado através de um *framework* computacional que apoia todas as etapas da pesquisa e principalmente fornecendo subsídio completo para a análise estatística dos resultados de saída utilizando métodos paramétricos e não paramétricos.

Para atingir este objetivo foram realizados alguns passos que conduziram o desenvolvimento desta tese. Para que um processo comparativo de análise estatística fosse desenvolvido, foi necessário o levantamento e o entendimento destes métodos estatísticos, o que foi realizado no capítulo três. Foram especificados os valores calculados da estatística descritiva, definidos os gráficos que fornecem informações sobre as amostras analisadas e especificados os métodos da estatística inferencial paramétrica e não paramétrica. A definição de qual método deveria ser calculado ficou baseado em três parâmetros, sendo o número de amostras comparadas, as dependências entre as amostras e a normalidade das amostras analisadas. O *script* desenvolvido em linguagem R é capaz de realizar todos os métodos a partir dos parâmetros, mas foi limitado no *framework*. Como todas as soluções algorítmicas são executadas como mesmo conjunto de instâncias, definiu-se que as amostras deveriam ser dependentes. O *script* desenvolvido gera relatórios estatísticos com todas as análises o que permite ao pesquisador fazer conclusões mais assertivas.

Outro passo no desenvolvimento deste trabalho, foi a identificação de padrões de análise e representação dos resultados, realizado no capítulo quatro. Foi especificado uma classificação de trabalhos de otimização por heurísticas sendo realizado um levantamento de trabalhos recentes com foco em problemas de *scheduling* e otimização por colônia de formigas. Todos os trabalhos encontrados foram categorizados com o intuito de identificar os padrões. Observou-se que a maioria dos trabalhos não possuíam análises estatísticas dos resultados, nem mesmo gráficos destes resultados e que as conclusões muitas vezes eram baseadas em valores médios e desvio padrão. Este levantamento contribuiu fornecendo uma base para a especificação de quais artifícios e recursos estatísticos deveriam estar disponíveis no *framework*.

Baseado nas definições de projeto de experimentos computacionais e dos levantamentos realizados anteriormente foi especificado o processo de desenvolvimento de pesquisas de otimização com heurísticas, que pode ser visto no capítulo cinco. Foram definidos todos os processos necessários para que uma pesquisa nesta área possa ser executada, desde a definição dos requisitos da pesquisa, a especificação dos *benchmarks* de instâncias de comparação, o desenvolvimento das soluções heurísticas comparadas, a parametrização destas soluções, a execução e coleta e resultados e principalmente a análise estatística dos resultados. Este processo além de ser a base para o desenvolvimento do *framework* fornece um guia para pesquisadores que trabalham na área.

O processo de desenvolvimento do *framework* foi estabelecido para um único recurso sendo definido um modelo iterativo-incremental. Todos os requisitos necessários ao desenvolvimento foram identificados e descritos, sendo realizado o processo de análise de sistemas que especificou o diagrama de classes e o modelo de dados. O desenvolvimento foi dividido em quatro iterações. A primeira iteração, a análise estatística dos resultados, foi conseguida através da codificação de um *script* em linguagem R, capaz de receber os resultados das execuções dos algoritmos, efetuar todas as análises estatísticas definidas gerando automaticamente um relatório estatístico. Com este relatório objetivou-se facilitar a análise de resultados de pesquisa fornecendo assim um conjunto de padrões para a representação de dados.

A segunda iteração preocupou-se com o desenvolvimento da base de funcionamento do *framework*, ou seja, o gerenciamento de todos os cadastros necessários para que uma pesquisa pudesse ser executada. Como decisões de

projeto foram utilizados somente *software* livre de código aberto e foi definido que o *framework* seria implantado em um servidor *web* para acesso irrestrito a outros pesquisadores. A terceira iteração implementou a execução automática de uma pesquisa. Os resultados das execuções são armazenados e processados deixando prontos para a chamada do *script* que processa as análises estatísticas. A quarta iteração implementou a parametrização simplificada de um algoritmo permitindo que o pesquisador faça a análise e tomada de decisão de quais são os melhores parâmetros de execução.

Por fim, foram realizados testes com o framework para validar as funcionalidades desenvolvidas. Foram feitos testes com o *script* que gera o relatório estatístico, testes de integração do sistema validando as funcionalidades e teste de aplicação com dados reais. Os testes realizados comprovaram a disponibilidade, a facilidade e a usabilidade do *framework* desenvolvido. Os relatórios estatísticos gerados pelo *script* foram capazes de apoiar as análises dos resultados e construção das conclusões. Com os dados da pesquisa cadastrados no *framework* foi possível fazer testes de parametrização e execução das soluções, inclusive sendo embasado pelas análises estatísticas dos resultados.

O framework desenvolvido mostrou ser uma ferramenta capaz de apoiar pesquisas de otimização baseada em heurísticas. Os principais resultados alcançados pela pesquisa são:

- Capacidade de armazenar e disponibilizar todas as informações de uma pesquisa para futuras consultas;
- Pode ser usado como repositório de bibliotecas de instâncias para especificação de *benchmarks* de execução e de teste para outras pesquisas;
- Permite realizar um processo simplificado de parametrização para dar apoio a identificação dos melhores parâmetros de execução das soluções algorítmicas comparadas;
- Permite realizar a execução automática das soluções com controle dos resultados obtidos;
- Gera relatório estatísticos amplos, a partir dos resultados das execuções da pesquisa incluindo tabelas, gráficos e teste estatísticos, capazes de

fornecer embasamento para que o pesquisador possa extrair conclusões mais assertivas e permitindo a comparação de pesquisas correlatas.

Além destes resultados podem-se especificar as seguintes contribuições deste trabalho:

- O *framework* é uma ferramenta de apoio a pesquisas e um guia de processo a ser usado como base no desenvolvimento de otimizadores baseados em heurísticas;
- Colabora na diminuição do tempo de desenvolvimento de pesquisas por disponibilizar recursos automatizados;
- Fornece aos pesquisadores uma ferramenta capaz de garantir que os resultados de trabalhos que estão sendo analisados possam ser comprovados;
- Especifica um conjunto de resultados estatísticos incluindo dados da estatística descritiva, gráficos e aplicação de métodos paramétricos e não paramétricos da estatística inferencial, servindo como padrão de análise e representação de resultados para futuras pesquisas;
- Facilita a comparação dos resultados encontrados com outras pesquisas correlatas e melhora a qualidade dos dados apresentados nas conclusões da pesquisa;
- Apesar de ser desenvolvido para problemas de *scheduling*, por limitação do foco da pesquisa, o *framework* é capaz de ser aplicado a outros tipos de problemas com possibilidade de utilização de diferentes métodos de otimização heurística e meta-heurística;
- Disponibiliza para outros pesquisadores o processo de análise estatística desenvolvido em R para execução, consultas e adequações;
- Permite a reprodutividade de uma pesquisa que pode ser executada mais de uma vez no *framework,* se necessário.

O presente trabalho apresenta limitações que são decorrentes de decisões do pesquisador na identificação dos caminhos seguidos, nas variações destes caminhos em função do desenvolvimento da pesquisa e dos objetivos definidos. A pesquisa realizada na identificação de padrões de análise e representação de resultados foi limitada a problemas de *scheduling* com utilização de otimização por colônia de formigas (ACO). Esta escolha foi em função da heurística possuir características

evolucionárias parecidas com outras heurísticas e apresentar um crescimento significativo de utilização nos últimos anos. Outra limitação é o amplo leque de métodos estatísticos para análise de dados. O *framework* está limitado a análises inferenciais paramétricas e não paramétricas, sendo necessária uma análise mais profunda na busca de outros métodos que poderiam contribuir para as conclusões de pesquisa na área tal como teste de clusterização por agrupamentos. Outro ponto é que o *framework* permite que o usuário especifique se os resultados analisados seguem ou não uma distribuição normal. Esta escolha define o método estatístico executado podendo enviesar as análises estatísticas. Neste sentido seria necessário realizar testes mais detalhados em relação ao processo de transformação dos resultados em uma distribuição normal assim como a confirmação de qual seria a distribuição dos resultados se forem utilizados valores aleatórios em uma distribuição normal.

Por fim, tem-se algumas limitações tecnológicas relativas as decisões do projeto durante o desenvolvimento do *framework*. A execução das soluções algorítmicas comparadas pelo *framework* estão limitada a somente arquivos executáveis (com extensões .EXE ou .BAT). O *framework* está implantado em um ambiente de servidor Windows por necessitar de alguns componentes deste sistema operacional para gerar o relatório estatístico final. Outra limitação, é que o servidor ao qual o framework está implantado, apesar de ser dedicado, não possui especificações para execução de problemas que necessitam de alto desempenho.

Percebe-se que o *framework* desenvolvido é capaz de apoiar pesquisas na área de otimização com aplicação de heurísticas e meta-heurísticas. A delimitação da pesquisa para problemas de programação de tarefas (*scheduling*) não exclui a possibilidade de utilização do *framework* para um amplo conjunto de problemas e métodos de otimização heurísticos. Espera-se que o *framework* seja aplicado a outras realidades em maior escala para que se comprove a sua validade.

O trabalho realizado abre novas perspectivas ainda a serem exploradas. Seria interessante a criação de um grupo de pesquisa com foco na especificação de padrões de análises estatística e representação de resultados bem definidos e o desenvolvimento de novas funcionalidades e aprimoramento do *framework*, implementando um repositório de pesquisas, *benchmarks* de instâncias e soluções heurísticas de forma que estas pesquisas possam ser comparadas. Outro ponto seria o desenvolvimento de recursos e componentes capazes de executar pesquisas em

ambientes paralelos e distribuídos com testes abrangentes de carga de utilização, devido ao fato de que pesquisas na área necessitam de muito poder de processamento. Outra ideia seria a especificação de um procedimento automatizado de parametrização com foco no refinamento e definição dos melhores parâmetros de execução das soluções algorítmicas analisadas. Por fim, uma linha de extensão deste trabalho seria um estudo aprofundado da relação existente entre o valor do GAP obtido das execuções em relação ao tempo de execução da solução algorítmica de forma a relacionar estatísticamente estas variáveis com o intuito de obter análises mais precisas.

REFERÊNCIAS

AHMADIZAR, F. A new ant colony algorithm for makespan minimization in permutation flow shops. **Computers & Industrial Engineering**, v. 63, n. 2, p. 355–361, 2012. Elsevier Ltd.

AHMADIZAR, F.; GHAZANFARI, M.; FATEMI GHOMI, S. M. T. Group shops scheduling with makespan criterion subject to random release dates and processing times. **Computers & Operations Research**, v. 37, n. 1, p. 152–162, 2010. Elsevier.

AHMADIZAR, F.; HOSSEINI, L. Bi-criteria single machine scheduling with a time-dependent learning effect and release times. **Applied Mathematical Modelling**, v. 36, n. 12, p. 6203–6214, 2012. Elsevier Inc.

AHMADIZAR, F.; RABANIMOTLAGH, A. Group shop scheduling with uncertain data and a general cost objective. **The International Journal of Advanced Manufacturing Technology**, v. 70, n. 5–8, p. 1313–1322, 2013.

ALBA, E.; ALMEIDA, F.; BLESA, M.; et al. MALLBA - A library of skeletons for combinatorial optimisation. **Euro-Par 2002 Parallel Processing**, v. 754, p. 63–73, 2002.

ALLAHVERDI, A.; NG, C. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research**, v. 187, p. 985–1032, 2008.

ALMEDER, C.; MÖNCH, L. Metaheuristics for scheduling jobs with incompatible families on parallel batching machines. **Journal of the Operational Research Society**, v. 62, n. 12, p. 2083–2096, 2010.

ANDREATTA, A.; CARVALHO, S.; RIBEIRO, C. A framework for local search heuristics for combinatorial optimization problems. **Optimization Software Class Libraries**, 2003. New York: Kluwer Academic Publishers.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional**. Rio de Janeiro: Campus, 2007.

ARNAOUT, J.-P.; MUSA, R.; RABADI, G. A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: enhancements and experimentations. **Journal of Intelligent Manufacturing**, v. 25, n. 1, p. 43–53, 2012.

ARNAOUT, J.-P.; RABADI, G.; MUSA, R. A two-stage Ant Colony Optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. **Journal of Intelligent Manufacturing**, v. 21, n. 6, p. 693–701, 2009.

BAKER, K. R.; TRIETSCH, D. **Principles of Sequencing and Scheduling**. John Wiley & Sons, 2009.

BARR, R. S.; GOLDEN, B. L.; KELLY, J. P.; RESENDE, M. G. C.; STEWART, W. R. Designing and Reporting on Computacional Experiments with Heuristic Methods. **Journal of Heuristics**, v. 1, n. 1, p. 9–32, 1995.

BARTZ-BEIELSTEIN, T.; CHIARANDINI, M.; PAQUETE, L.; PREUSS, M. **Experimental Methods for the Analysis of Optimization Algorithms**. Springer Science & Business Media, 2010.

BAUER, A.; BULLNHEIMER, B. Minimizing total tardiness on a single machine using ant colony optimization. **Central European Journal of Operational Research**, p. 1–20, 2000.

BEASLEY, J. E. OR-Library: Distributing Test Problems by Electronic Mail. **Journal of the Operational Research Society**, v. 41, n. 11, p. 1069–1072, 1990.

BEHNAMIAN, J.; FATEMI GHOMI, S. M. T.; ZANDIEH, M. Development of a hybrid metaheuristic to minimise earliness and tardiness in a hybrid flowshop with sequence-dependent setup times. **International Journal of Production Research**, v. 48, n. 5, p. 1415–1438, 2010.

BEHNAMIAN, J.; ZANDIEH, M.; FATEMI GHOMI, S. M. T. Bi-objective parallel machines scheduling with sequence-dependent setup times using hybrid metaheuristics and weighted min–max technique. **Soft Computing**, v. 15, n. 7, p. 1313–1331, 2010.

BELAID, R.; T'KINDT, V.; ESSWEIN, C. Scheduling batches in flowshop with limited buffers in the shampoo industry. **European Journal of Operational Research**, v. 223, n. 2, p. 560–572, 2012. Elsevier B.V.

BERRICHI, A.; YALAOUI, F. Efficient bi-objective ant colony approach to minimize total tardiness and system unavailability for a parallel machine scheduling problem. **The International Journal of Advanced Manufacturing Technology**, 2013.

BERRICHI, A.; YALAOUI, F.; AMODEO, L.; MEZGHICHE, M. Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling. **Computers & Operations Research**, v. 37, n. 9, p. 1584–1596, 2010. Elsevier.

BERTRAND, J. W. M.; FRANSOO, J. C. Operations management research methodologies using quantitative modeling. **International Journal of Operations & Production Management**, v. 22, n. 2, p. 241–264, 2002.

BEZANSON, J.; KARPINSKI, S.; SHAH, V. B.; EDELMAN, A. Julia: A Fast Dynamic Language for Technical Computing. **arXiv:1209.5145**, p. 1–27, 2012.

BIRATTARI, M.; YUAN, Z.; BALAPRAKASH, P.; STÜTZLE, T. Automated Algorithm Tuning using {F}-races: Recent Developments. **Proceedings of MIC 2009, the 8th Metaheuristics International Conference**, p. Proceedings on CD-ROM, 10 pages, 2009.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Computing Surveys (CSUR)**, v. 35, n. 3, p. 268–308, 2003.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: guia do usuário**. Rio de Janeiro: CAMPUS - RJ, 2006.

BROWNLEE, J. OAT : The Optimization Algorithm Toolkit. , , n. December, p. 1–6, 2007.

BRUCKER, P.; KNUST, S. Complexity results for scheduling problems. Disponível em: <http://www.informatik.uni-osnabrueck.de/knust/class/>. Acesso em: 25/7/2014.

BRUNI, A. L. **ESTATISTICA APLICADA A GESTAO EMPRESARIAL**. 2 ediçao ed. São Paulo: ATLAS, 2010.

BURKE, E.; CURTOIS, T.; HYDE, M. HyFlex: A flexible framework for the

design and analysis of hyper-heuristics. **Multidisciplinary International Scheduling Conference (MISTA 2009), …**, , n. August, p. 790--797, 2009.

BUSSAB, W. O.; MORETTIN, P. A. **Estatística básica**. 5a edição ed. São Paulo: Editora Saraiva, 2004.

CAHON, S.; MELAB, N.; TALBI, E.; SCHOENAUER, M. ParaDisEO-based design of parallel and distributed evolutionary algorithms. **Artificial Evolution**, p. 216–228, 2004.

CHEN, C.-F.; WU, M.-C.; LI, Y.-H.; TAI, P.-H.; CHIOU, C.-W. A comparison of two chromosome representation schemes used in solving a family-based scheduling problem. **Robotics and Computer-Integrated Manufacturing**, v. 29, n. 3, p. 21–30, 2013. Elsevier.

CHEN, H.; DU, B.; HUANG, G. Q. Metaheuristics to minimise makespan on parallel batch processing machines with dynamic job arrivals. **International Journal of Computer Integrated Manufacturing**, v. 23, n. 10, p. 942–956, 2010.

CHEN, Z.-L.; VAIRAKTARAKIS, G. L. Integrated Scheduling of Production and Distribution Operations. **Management Science**, v. 51, n. 4, p. 614–628, 2005.

CHENG, B.; LI, K.; CHEN, B. Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization. **Journal of Manufacturing Systems**, v. 29, n. 1, p. 29–34, 2010. Elsevier Ltd.

CHENG, B.; WANG, Q.; YANG, S.; HU, X. An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes. **Applied Soft Computing**, v. 13, n. 2, p. 765–772, 2013. Elsevier B.V.

DEMIRKOL, E.; MEHTA, S.; UZSOY, R. Benchmarks for shop scheduling problems. **European Journal of Operational Research**, v. 109, p. 137–141, 1998.

DEMŠAR, J. Statistical Comparisons of Classifiers over Multiple Data Sets. **The Journal of Machine Learning Research**, v. 7, p. 1–30, 2006.

DERRAC, J.; GARCÍA, S.; MOLINA, D.; HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v. 1, n. 1, p. 3–18, 2011.

DJANGO SOFTWARE FOUNDATION. Django: The Web framework for perfectionists with deadlines. Disponível em: <https://www.djangoproject.com/>. .

DRAGULESCU, A. A. xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files. , 2014.

DRÉOK, J.; TFAILIK, W.; AUMASSONK, J.-P. OMETAH - Open Metaheuristic. Disponível em: <http://freecode.com/projects/ometah>. Acesso em: 5/11/2016.

DRIRA, A.; PIERREVAL, H.; HAJRI-GABOUJ, S. Facility layout problems: A survey. **Annual Reviews in Control**, v. 31, n. 2, p. 255–267, 2007.

DUNN, O. J. Multiple Comparisons Among Means. **Journal of the American Statistical Association**, v. 56, n. 293, p. 52–64, 1961.

DU-PREL, J.-B.; RÖHRIG, B.; HOMMEL, G.; BLETTNER, M. Choosing statistical tests: part 12 of a series on evaluation of scientific publications. **Deutsches Arzteblatt international**, v. 107, n. 19, p. 343–348, 2010.

DURILLO, J. J.; NEBRO, A. J. JMetal: A Java framework for multi-objective optimization. **Advances in Engineering Software**, v. 42, n. 10, p. 760–771, 2011.

ELYASAF, A.; SIPPER, M. Software review: The HeuristicLab framework. **Genetic Programming and Evolvable Machines**, v. 15, n. 2, p. 215–218, 2014.

ERIKSSON, H.-E.; PENKER, M.; LYONS, B.; FADO, D. **UML 2 Toolkit**. John Wiley & Sons, 2003.

FINK, A.; VOB, S. HotFrame: A heuristic optimization framework. **Optimization software class libraries**, p. 81–154, 2002.

FINNER, H. On a Monotonicity Problem in Step-Down Multiple Test Procedures. **Journal of the American Statistical Association**, v. 88, n. 423, p. 920–923, 1993.

FLEXER, A. Statistical Evaluation of Neural Network Experiments: Minimum Requirements and Current Practice. **Cybernetics and Systems Research**, v. 2, p. 1005–1008, 1996.

FORTIN, F.-A.; RAINVILLE, F.-M. DE; GARDNER, M.-A.; PARIZEAU, M.; GAGN, C. DEAP : Evolutionary Algorithms Made Easy. **Journal of Machine Learning Research**, v. 13, p. 2171–2175, 2012.

FOX, J. RcmdrMisc: R Commander Miscellaneous Functions. , 2015.

FREUND, J. E. **Estatística Aplicada - Economia, Administração e Contabilidade**. Bookman, 2009.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the American Statistical Association**, v. 32, n. 200, p. 675–701, 1937.

FRIEDMAN, M. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. **The Annals of Mathematical Statistics**, v. 11, n. 1, p. 86–92, 1940. Institute of Mathematical Statistics.

GAGNÉ, C.; PARIZEAU, M. Genericity in Evolutionary Computation Software Tools: Principles and Case-Study. **International Journal on Artificial Intelligence Tools**, v. 15, p. 173–194, 2006.

GAGOLEWSKI, M.; TARTANUS, B. R package stringi: Character string processing facilities. , 2015.

GARCÍA, S.; FERNÁNDEZ, A.; LUENGO, J.; HERRERA, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. **Information Sciences**, v. 180, n. 10, p. 2044–2064, 2010. Elsevier Inc.

GARCÍA, S.; MOLINA, D.; LOZANO, M.; HERRERA, F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 Special Session on Real Parameter Optimization. **Journal of Heuristics**, v. 15, n. 6, p. 617–644, 2009.

GASPERO, L. DI; SCHAERF, A. EasyLocal++: An Object-Oriented Framework for Exible Design of Local Search Algorithms. **Software - Practice and Experience**, v. 33, n. 8, p. 733–765, 2003.

GATICA, C.; ESQUIVEL, S.; LEGUIZAMON, G. An ACO approach for the parallel machines scheduling problem. **Inteligencia Artificial**, v. v 14, n. n 46, p. 84–

95, 2010.

GIBBONS, J. D.; CHAKRABORTI, S. **Nonparametric Statistical Inference, Fourth Edition: Revised and Expanded**. 2014.

GROSS, J.; LIGGES, U. nortest: Tests for Normality. , 2015.

HANOUN, S.; NAHAVANDI, S.; KULL, H. Pareto Archived Simulated Annealing for Single Machine Job Shop Scheduling with Multiple Objectives. ICCGI 2011 : The Sixth International Multi-Conference on Computing in the Global Information Technology. **Anais...** , 2011.

HAYTER, A. The maximum familywise error rate of Fisher's least significant difference test. **Journal of the American Statistical Association**, v. 81, n. 396, p. 1000–1004, 1986.

HECKER, F. T.; HUSSEIN, W. B.; PAQUET-DURAND, O.; HUSSEIN, M. A.; BECKER, T. A case study on using evolutionary algorithms to optimize bakery production planning. **Expert Systems with Applications**, v. 40, n. 17, p. 6837–6847, 2013. Elsevier Ltd.

HECKER, F. T.; STANKE, M.; BECKER, T.; HITZMANN, B. Application of a modified GA, ACO and a random search procedure to solve the production scheduling of a case study bakery. **Expert Systems with Applications**, v. 41, n. 13, p. 5882–5891, 2014. Elsevier Ltd.

HIRSCH, P.; PALFI, A.; GRONALT, M. Solving a time constrained two-crane routing problem for material handling with an ant colony optimisation approach: an application in the roof-tile industry. **International Journal of Production Research**, v. 50, n. 20, p. 6005–6021, 2012.

HO, Y.-C. H. Y.-C.; PEPYNE, D. L. Simple explanation of the no free lunch theorem of optimization. **40th IEEE Conference on Decision and Control**, 2001.

HOCHBERG, Y. A sharper bonferroni procedure for multiple tests of significance. **Biometrika**, 1988.

HOLLAND, B. S.; COPENHAVER, M. D. An Improved Sequentially Rejective Bonferroni Test Procedure. **Biometrics**, v. 43, n. 2, p. 417–423, 1987.

HOLLANDER, M.; WOLFE, D. A.; CHICKEN, E. **Nonparametric Statistical Methods**. 3 ed ed. John Wiley & Sons, 2013.

HOLM, S. A simple sequentially rejective multiple test procedure. **Scandinavian Journal of Statistics**, v. 6, p. 65–70, 1979.

HOMMEL, G. A stagewise rejective multiple test procedure based on a modified Bonferroni test. **Biometrika**, v. 75, n. 2, p. 383–386, 1988.

HOTHORN, T.; BRETZ, F.; WESTFALL, P. Simultaneous Inference in General Parametric Models. **Biometrical Journal**, v. 50, n. 3, p. 346–363, 2008.

HOTHORN, T.; HORNIK, K.; WIEL, M. A. VAN DE; ZEILEIS, A. A Lego System for Conditional Inference. **The American Statistician**, v. 60, n. 3, p. 257–263, 2006. Taylor & Francis.

HOWELL, D. **Fundamental Statistics for the Behavioral Sciences**. Cengage Learning, 2013.

HUANG, R.-H. Multi-objective job-shop scheduling with lot-splitting production. **International Journal of Production Economics**, v. 124, n. 1, p. 206–213, 2010.

Elsevier.

HUANG, R.-H.; YANG, C.-L.; CHENG, W.-C. Flexible job shop scheduling with due window—a two-pheromone ant colony approach. **International Journal of Production Economics**, v. 141, n. 2, p. 685–697, 2013. Elsevier.

HUANG, R.-H.; YU, S.-C. Clarifying Cutting and Sewing Processes with Due Windows Using an Effective Ant Colony Optimization. **Mathematical Problems in Engineering**, v. 2013, p. 1–12, 2013.

IHAKA, R.; MURRELL, P.; HORNIK, K.; FISHE, J. C.; ZEILEIS, A. colorspace: Color Space Manipulation. , 2015.

IMAN, R. L.; DAVENPORT, J. M. Approximations of the critical region of the Friedman statistic. **Communications in Statistics - Theory and Methods**, v. A9, n. 6, p. 571–595, 1980.

JACCARD, J.; BECKER, M. A.; WOOD, G. Pairwise multiple comparison procedures: A review. **Psychological Bulletin**, v. 96, n. 3, p. 589–596, 1984.

JIN, J.; CRAINIC, T. G.; LØKKETANGEN, A. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. **European Journal of Operational Research**, v. 222, n. 3, p. 441–451, 2012. Elsevier B.V.

JOHNSON, R. Components, frameworks, patterns. **ACM SIGSOFT Software Engineering Notes**, , n. 217, p. 1–23, 1997.

JOHNSON, R. E. Documenting frameworks using patterns. **ACM SIGPLAN Notices**, v. 27, p. 63–76, 1992.

KEIJZER, M.; MERELO, J.; ROMERO, G.; SCHOENAUER, M. Evolving objects: A general purpose evolutionary computation library. **Artificial Evolution**, 2002.

KESKINTURK, T.; YILDIRIM, M. B.; BARUT, M. An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times. **Computers & Operations Research**, v. 39, n. 6, p. 1225–1235, 2012. Elsevier.

KOLISCH, R.; SPRECHER, A. PSPLIB - A project scheduling problem library. **European Journal of Operational Research**, 1997.

KORYTKOWSKI, P.; RYMASZEWSKI, S.; WIŚNIEWSKI, T. Ant colony optimization for job shop scheduling using multi-attribute dispatching rules. **The International Journal of Advanced Manufacturing Technology**, v. 67, n. 1–4, p. 231–241, 2013.

KOTHARI, C. **Research methodology: methods and techniques**. 2004.

KRISHNARAJ, J.; PUGAZHENDHI, S.; RAJENDRAN, C.; THIAGARAJAN, S. A modified ant-colony optimisation algorithm to minimise the completion time variance of jobs in flowshops. **International Journal of Production Research**, v. 50, n. 20, p. 5698–5706, 2012.

KRONFELD, M.; PLANATSCHER, H.; ZELL, A. The EvA2 optimization framework. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 6073 LNCS, p. 247–250, 2010.

KUO, R. J.; CHENG, W. C. Hybrid meta-heuristic algorithm for job shop scheduling with due date time window and release time. **The International Journal of**

Advanced Manufacturing Technology, v. 67, n. 1–4, p. 59–71, 2013.

LAKATOS, E. M.; MARCONI, M. D. A. M. **Fundamentos de metodologia científica**. 6 ediçao ed. São Paulo: Atlas, 2005.

LARSON, R.; FARBER, B. **Estatística aplicada**. Prentice Hall, 2004.

LAU, H. C.; JIA, X.; WAN, W. C. A Generic Object-Oriented Tabu Search Framework. **Metaheuristics: Progress as Real Problem Solvers**, p. 203–226, 2005.

LI, D.; WANG, Y.; XIAO, G.; TANG, J. Dynamic parts scheduling in multiple job shop cells considering intercell moves and flexible routes. **Computers & Operations Research**, v. 40, n. 5, p. 1207–1223, 2013. Elsevier.

LI, J.; SUN, S.; HUANG, Y. Adaptive Hybrid ant colony optimization for solving Dual Resource Constrained Job Shop Scheduling Problem. **Journal of Software**, v. 6, n. 4, p. 584–594, 2011.

LI (DAVID), J. A two-step rejection procedure for testing multiple hypotheses. **Journal of Statistical Planning and Inference**, v. 138, p. 1521–1527, 2008.

LIANG, Y.-C.; HSIAO, Y.-M.; TIEN, C.-Y. Metaheuristics for drilling operation scheduling in Taiwan PCB industries. **International Journal of Production Economics**, v. 141, n. 1, p. 189–198, 2013. Elsevier.

LIANG, Y.-C.; LEE, Z.-H.; CHEN, Y.-S. A novel ant colony optimization approach for on-line scheduling and due date determination. **Journal of Heuristics**, v. 18, n. 4, p. 571–591, 2012.

LIAO, C.-J.; TSAI, Y.-L.; CHAO, C.-W. An ant colony optimization algorithm for setup coordination in a two-stage production system. **Applied Soft Computing**, v. 11, n. 8, p. 4521–4529, 2011. Elsevier B.V.

LIAO, T. W.; CHANG, P. C.; KUO, R. J.; LIAO, C.-J. A comparison of five hybrid metaheuristic algorithms for unrelated parallel-machine scheduling and inbound trucks sequencing in multi-door cross docking systems. **Applied Soft Computing**, v. 21, p. 180–193, 2014. Elsevier B.V.

LICENSE, G. N. U. G. P. GNU General Public License. **Distribution**, , n. June, p. 1–9, 1989.

LILLIEFORS, H. W. on the Kolmogorov-Smirnov Test for Normality With Mean and Variance Unknown. **Journal of the American Statistical Association**, v. 62, n. 318, p. 399–402, 1967.

LIN, C.-W.; LIN, Y.-K.; HSIEH, H.-T. Ant colony optimization for unrelated parallel machine scheduling. **The International Journal of Advanced Manufacturing Technology**, v. 67, n. 1–4, p. 35–45, 2013.

LOPEZ-IBANEZ, M.; DUBOIS-LACOSTE, J.; STÜTZLE, T.; BIRATTARI, M. **The irace Package : Iterated Racing for Automatic Algorithm Configuration**. 2011.

LUKASIEWYCZ, M.; GLAS, M.; REIMANN, F.; TEICH, J. Opt4J - A Modular Framework for Meta-heuristic Optimization. **Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011)**, p. 1723–1730, 2011.

LUKE, S. The ECJ Owner ' s Manual. , 2015. User Manual, .

LUKE, S.; PANAIT, L.; BALAN, G.; PAUS, S. ECJ - A Java-based Evolutionary Computation Research System. Disponível em:

<https://cs.gmu.edu/~eclab/projects/ecj/>. Acesso em: 6/11/2016.

LUO, H.; DU, B.; HUANG, G. Q.; CHEN, H.; LI, X. Hybrid flow shop scheduling considering machine electricity consumption cost. **International Journal of Production Economics**, v. 146, n. 2, p. 423–439, 2013. Elsevier.

MACCARTHY, B. L.; FERNANDES, F. C. F. A multi-dimensional classification of production systems for the design and selection of production planning and control systems. **Production Planning & Control**, v. 11, n. 5, p. 481–496, 2000.

MAISCHBERGER, M. METSlib | An Open Source Tabu Search Metaheuristic framework in modern C++ (cpp). Disponível em: <https://projects.coin-or.org/metslib>. Acesso em: 5/11/2016.

MAISCHBERGER, M. COIN-OR METSlib a Metaheuristics Framework in Modern C++. **Framework**, 2011.

MARTINS, R. A. Abordagens Quantitativas e Qualitativas. **Metodologia de Pesquisa em Egenharia de Produçao e Gestão de Operações**, 2012. Rio de Janeiro: Elsevier Brasil.

MARUSTER, M.; BACAREA, V. Comparing groups for statistical differences: how to choose the right statistical test? **Biochemia Medica**, v. 20, n. 1, p. 15–32, 2010.

MASON, R. L.; GUNST, R. F.; HESS, J. L. **Statistical Design and Analysis of Experiments**. 2003.

MCLAUGHLIN, B.; POLLICE, G.; WEST, D. **Head First Object-Oriented Analysis and Design: A Brain Friendly Guide to OOA&D**. "O'Reilly Media, Inc.," 2007.

MENDES, A. DE S. **O Framework NP-Opt e suas Aplicações a Problemas de Otimização**, 2003.

MENDES, A.; FRANÇA, P.; MOSCATO, P. NP-Opt: an optimization framework for NP problems. **Proceedings of the IV SIMPOI/POMS 2001**, p. 11–14, 2001.

MENDIBURU, F. DE. agricolae: Statistical Procedures for Agricultural Research. , 2015.

MIRABI, M. Ant colony optimization technique for the sequence-dependent flowshop scheduling problem. **The International Journal of Advanced Manufacturing Technology**, v. 55, n. 1–4, p. 317–326, 2010.

MONTGOMERY, D. **Design and analysis of experiments**. 5th Editio ed. New York: John Wiley & Sons, 2001.

MONTGOMERY, D. D. C.; RUNGER, G. C. G. G. C. G. **Applied statistics and probability for engineers**. Third Edit ed. New York: John Wiley & Sons, Inc., 2002.

MORABITO, R.; PUREZA, V. Modelagem e Simulação. **Metodologia de Pesquisa em Egenharia de Produçao e Gestão de Operações**, 2012. Rio de Janeiro: Elsevier Brasil.

MORSE, D. T. MINSIZE2: A Computer Program for Determining Efect Size and Minimum Sample Size for Statistical Significance for Univariate, Multivariate and Nonparamétric Tests. **Educational and Psychological Measurement**, v. 59, n. 3, p. 518–531, 1999.

NAGANO, M. S.; ANTONIO, L.; LORENA, N.; INPE, L. A. C. Programaçao da Produçao Flow Shop Permutacional com Minimizaçao do Tempo Médio de Fluxo.

XXXVI Simpósio Brasileiro de Pesquisa Operacional. **Anais...** , 2004. São João Del-Rei - MG.

NAWAZ, M.; ENSCORE, E. E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. **Omega**, v. 11, n. 1, p. 91–95, 1983. Pergamon.

NEMENYI, P. **Distribution-free Multiple Comparisons**, 1963. Princeton University.

PAREJO, J. A.; RACERO, J.; GUERRERO, F.; KWOK, T.; SMITH, K. A. FOM: A Framework for Metaheuristic Optimization. **Computational Science — ICCS 2003**. p.886–895, 2003. Springer Berlin Heidelberg.

PAREJO, J. A.; RUIZ-CORTÉS, A.; LOZANO, S.; FERNANDEZ, P. Metaheuristic optimization frameworks: A survey and benchmarking. **Soft Computing**, v. 16, n. 3, p. 527–561, 2012.

PASANDIDEH, S. H. R.; NIAKI, S. T. A. Genetic application in a facility location problem with random demand within queuing framework. **Journal of Intelligent Manufacturing**, v. 23, n. 3, p. 651–659, 2010.

PEREIRA, D. G.; AFONSO, A.; MEDEIROS, F. M. Overview of Friedman's test and post-hoc analysis. **Communications in Statistics-Simulation and Computation**, , n. December, p. 931–971, 2014.

PINHEIRO, J.; GOMES, G.; CUNHA, S.; CARVAJAL, S. **Estatística Básica A Arte de Trabalhar Com Dados**. São Paulo: Elsevier Editora Ltda, 2009.

POHLERT, T. PMCMR: Calculate Pairwise Multiple Comparisons of Mean Rank Sums. , 2015.

PRESSMAN, R. S. **Engenharia de Software**. McGraw Hill Brasil, 2011.

PYTHON SOFTWARE FOUNDATION. Python 2.6 license. Disponível em: <http://www.python.org/download/releases/2.6/license/>. .

RABANIMOTLAGH, A. An efficient ant colony optimization algorithm for multiobjective flow shop scheduling problem. **World Academy of Science, Engineering and Technology**, p. 127–133, 2011.

R DEVELOPMENT CORE TEAM, R. R: A Language and Environment for Statistical Computing. **R Foundation for Statistical Computing**, 2014.

RAINVILLE, F. DE; FORTIN, F.; GARDNER, M.; PARIZEAU, M.; GAGNÉ, C. DEAP : A Python Framework for Evolutionary Algorithms. **Companion proc. of the Genetic and Evolutionary Computation Conference**, p. 85–92, 2012.

RARDIN, R.; UZSOY, R. Experimental evaluation of heuristic optimization algorithms: A tutorial. **Journal of Heuristics**, v. 304, p. 261–304, 2001.

RMETRICS, T. C.; WUERTZ, AND D.; SETZ, AND T.; CHALABI, Y. fBasics: Rmetrics - Markets and Basic Statistics. , 2014.

RODRIGUEZ, F. J.; LOZANO, M.; BLUM, C.; GARCÍA-MARTÍNEZ, C. An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. **Computers & Operations Research**, v. 40, n. 7, p. 1829–1841, 2013.

ROM, D. M. A sequentially rejective test procedure based on a modified Bonferroni inequality. **Biometrika**, v. 77, n. 3, p. 663–665, 1990.

ROSSI, A.; LANZETTA, M. Nonpermutation flow line scheduling by ant colony optimization. **Artificial Intelligence for Engineering Design, Analysis and**

Manufacturing, v. 27, n. 4, p. 349–357, 2013a.

ROSSI, A.; LANZETTA, M. Scheduling flow lines with buffers by ant colony digraph. **Expert Systems with Applications**, v. 40, n. 9, p. 3328–3340, 2013b.

RUIZ, R.; STÜTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, v. 177, p. 2033–2049, 2007.

RUIZ, R.; VALLADA, E.; FERNÁNDEZ-MARTÍNEZ, C. Scheduling in flowshops with no-idle machines. **Studies in Computational Intelligence**, v. 230, p. 21–51, 2009.

SAĞ, T.; ÇUNKAŞ, M. A tool for multiobjective evolutionary algorithms. **Advances in Engineering Software**, v. 40, n. 9, p. 902–912, 2009.

SALMASI, N.; LOGENDRAN, R.; SKANDARI, M. R. Makespan minimization of a flowshop sequence-dependent group scheduling problem. **The International Journal of Advanced Manufacturing Technology**, v. 56, n. 5–8, p. 699–710, 2011.

SARKAR, D. Lattice multivariate data visualization with R. **Springer**, 2008.

SCHAFFER, M. E. rtf: Rich Text Format (RTF) Output. , 2013.

SEO, M.; KIM, D. Ant colony optimisation with parameterised search space for the job shop scheduling problem. **International Journal of Production Research**, v. 48, n. 4, p. 1143–1154, 2010.

SHAPIRO, S. S.; WILK, M. B. An Analysis of Variance Test for Normality (Complete Samples). **Biometrika**, v. 52, n. 3/4, p. 591–611, 1965.

SHESKIN, D. J. **Handbook of Parametric and Nonparametric Statistical Procedures: Third Edition**. CRC Press, 2003.

SIEGEL, S. Nonparametric statistics. **The American Statistician**, v. 11, n. 3, p. 13–19, 1957.

SIGNORELL, A. DescTools: Tools for Descriptive Statistics. , 2015.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. **Administração da produção**. Atlas, 2009.

SOMMERVILLE, I. **ENGENHARIA DE SOFTWARE**. PEARSON BRASIL, 2011.

STÜTZLE, T.; HOOS, H. MAX–MIN ant system. **Future generation computer systems**, , n. November, p. 1–39, 2000.

SUNDARAMOORTHY, A.; MARAVELIAS, C. A general framework for process scheduling. **Aiche Journal**, 2011.

SWEENEY, D. J.; ANDERSON, D. R.; WILLIAMS, T. A. **Estatística aplicada à administração e economia**. 2 edição ed. São Paulo: Cengage Learning, 2007.

TAILLARD, E. Benchmarks for basic scheduling problems. **European Journal of Operational Research**, v. 64, n. 2, p. 278–285, 1993.

TAVARES NETO, R. F.; GODINHO FILHO, M. Proposta de um framework para prototipagem de sistemas heurísticos multiagentes baseados em algorítmos de colônia de formigas. Pesquisa Operacional. **Anais...** . v. 29, p.643–668, 2009. SOBRAPO.

TAVARES NETO, R. F.; GODINHO FILHO, M. An ant colony optimization

approach to a permutational flowshop scheduling problem with outsourcing allowed. **Computers & Operations Research**, v. 38, n. 9, p. 1286–1293, 2011. Elsevier.

TAVARES NETO, R. F.; OLIVEIRA, R. C. DE. A Redução Do Tempo Total De Fluxo Em Sistemas Integrados Produção-Distribuição. XXII SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO - SIMPEP. **Anais...** , 2015. Bauru - SP.

THIRUVADY, D.; SINGH, G.; ERNST, A. T.; MEYER, B. Constraint-based ACO for a shared resource constrained scheduling problem. **International Journal of Production Economics**, v. 141, n. 1, p. 230–242, 2013. Elsevier.

TIMPERLEY, C. S.; STEPNEY, S. Wallace : An efficient generic evolutionary framework. , p. 365–372, 2008.

TOLEDO, C. F. M.; OLIVEIRA, R. R. R. DE; MORELATO FRANÇA, P. A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. **Computers & Operations Research**, v. 40, n. 4, p. 910–919, 2013.

TUKEY, J. W. Comparing individual means in the analysis of variance. **Biometrics**, v. 5, n. 2, p. 99, 1949.

TZENG, Y.-R.; CHEN, C.-L.; CHEN, C.-L. A hybrid EDA with ACS for solving permutation flow shop scheduling. **The International Journal of Advanced Manufacturing Technology**, v. 60, n. 9–12, p. 1139–1147, 2011.

UDHAYAKUMAR, P.; KUMANAN, S. Sequencing and scheduling of job and tool in a flexible manufacturing system using ant colony optimization algorithm. **The International Journal of Advanced Manufacturing Technology**, v. 50, n. 9–12, p. 1075–1084, 2010.

VALLADA, E.; RUIZ, R.; FRAMINAN, J. M. New hard benchmark for flowshop scheduling problems minimising makespan. **European Journal of Operational Research**, v. 240, n. 3, p. 666–677, 2015. Elsevier B.V.

VENDRAMIN, A. C. K.; MUNARETTO, A.; DELGADO, M. R.; VIANA, A. C. GrAnt: Inferring best forwarders from complex networks' dynamics through a greedy Ant Colony Optimization. **Computer Networks**, v. 56, n. 3, p. 997–1015, 2012. Elsevier B.V.

VENTURA, S.; ROMERO, C.; ZAFRA, A.; DELGADO, J. A.; HERVÁS, C. JCLEC: A Java framework for evolutionary computation. **Soft Computing**, v. 12, n. 4, p. 381–392, 2008.

WAGNER, S.; KRONBERGER, G.; BEHAM, A.; et al. Architecture and Design of the HeuristicLab Optimization Environment. **First Australian Conference on the Applications of Systems Engineering, ACASE**, v. 6, p. 197–261, 2012.

WHITE, D. R. Software review: The ECJ toolkit. **Genetic Programming and Evolvable Machines**, v. 13, n. 1, p. 65–67, 2012.

WICKHAM, H. stringr: Simple, Consistent Wrappers for Common String Operations. , 2015.

WILCOXON, F. Individual Comparisons by Ranking Methods. **Biometrics Bulletin**, v. 1, n. 6, p. 80–83, 1945.

WITKOWSKI, T.; ANTCZAK, P.; ANTCZAK, A. Solving the Flexible Open-Job Shop Scheduling Problem with GRASP and Simulated Annealing. **2010 International Conference on Artificial Intelligence and Computational Intelligence**, p. 437–442,

2010. Ieee.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67–82, 1997.

WU, C.-C.; WU, W.-H.; CHEN, J.-C.; YIN, Y.; WU, W.-H. A study of the single-machine two-agent scheduling problem with release times. **Applied Soft Computing**, v. 13, n. 2, p. 998–1006, 2013. Elsevier B.V.

WU, C.-C.; WU, W.-H.; WU, W.-H.; et al. A single-machine scheduling with a truncated linear deterioration and ready times. **Information Sciences**, v. 256, p. 109–125, 2014. Elsevier Inc.

XING, L.-N.; CHEN, Y.-W.; WANG, P.; ZHAO, Q.-S.; XIONG, J. A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. **Applied Soft Computing**, v. 10, n. 3, p. 888–896, 2010. Elsevier B.V.

XU, R.; CHEN, H.; LI, X. Makespan minimization on single batch-processing machine via ant colony optimization. **Computers & Operations Research**, v. 39, n. 3, p. 582–593, 2012. Elsevier.

YAGMAHAN, B.; YENISEY, M. M. A multi-objective ant colony system algorithm for flow shop scheduling problem. **Expert Systems with Applications**, v. 37, n. 2, p. 1361–1368, 2010. Elsevier Ltd.

ZIMMERMAN, D. W.; ZUMBO, B. D. Relative Power of the Wilcoxon Test, the Friedman Test, and Repeated-Measures ANOVA on Ranks. **Journal of Experimental Education**, v. 62, n. 1, p. 11, 1993.

ZUNINO, I.; MELAB, N.; TALBI, E.-G. A grid-enabled framework for exact optimization algorithms. **21st European Conference on Modelling and Simulation: Simulations in United Europe, ECMS 2007**, 2007.

APÊNDICE A - RELATÓRIO ESTATÍSTICO COM 5 ALGORITMOS

RELATÓRIO DAS ANÁLISES ESTATÍSTICAS

Nome da pesquisa: **RodIG5Alg**
Quantidade de algoritmos testados na análise estatística: **5**
Nome do arquivo de dados de entrada: **RodIG5Alg.txt**
As amostras seguem uma distribuição estatística NORMAL: **NÃO**
As amostras são dependentes: **SIM**
Nível de significância estatística utilizada: **0.05**
Nível de confiança estatística utilizada: **95%**
Testes estatísticos executados:

Teste ANOVA com um fator e blocos
Teste Friedman

1 - Resultados da estatística descritiva dos dados

A estatística descritiva é a primeira etapa da análise estatística utilizada para descrever e resumir o conjunto de dados analisados, fornecendo informações simples sobre os dados amostrados e sobre as observações realizadas. O objetivo é fornecer um entendimento inicial dos dados como parte de uma análise estatística mais detalhada. São apresentados para cada amostra: média, desvio padrão, erro padrão da média, intervalo de interquartil, coeficiente de variabilidade (média/desvio padrão), skewness (simetria dos dados), kurtosis (grau de concentração dos dados a respeito da média), moda, frequência da moda, valor mínimo, primeiro quartil, mediana, terceiro quartil e o valor máximo.

Tabela 1 - Resultados da estatística descritiva dos dados

	mean	sd	se.mean	IQR	cv	skewness	kurtosis
GA	1.41729	4.24619	0.36819	1.3	2.99598	1.61867	7.3394
GRASP	17.03398	76.85599	6.66426	2.59	4.51192	6.57648	46.99291
IGRWD1	-0.54218	2.68763	0.23305	0.25	-	-2.75479	12.00008
MultiS	67.5191	155.99646	13.52661	42.79	2.31041	3.51244	14.39776
Tabu	67.08226	154.88015	13.42981	42.77	2.30881	3.51376	14.44037

Tabela 2 - Resultados dos quartis dos dados

	moda	freq.mod	min	quart.1	median	quart.3	max
GA	0.33	3	-11.34	0.26	0.62	1.56	23.95
GRASP	0.01	14	-13.62	0.01	0.13	2.6	651.21
IGRWD1	0	26	-13.92	-0.13	0	0.12	9.56
MultiS	0.02	5	-12.65	0.07	1.32	42.86	984.61
Tabu	0.03	5	-12.65	0.09	1.48	42.86	975.99

Legenda da tabela de resultados da estatística descritiva:

mean = média
sd = desvio padrão
se(mean) = erro padrão (média)
IQR = intervalo de interquartil
cv = coeficiente variabilidade = média/sd
skewness = simetria dos dados
kurtosis = grau de concentração dos dados a respeito da média
mod = moda
freqMod = frequência da moda
quantiles = quartis
- min = valor mínimo
- quart-1 = primeiro quartil
- median = mediana
- quart-3 = terceiro quartil
- max = valor máximo

2 - Análise gráfica das amostras

2.1 - *Boxplot* das amostras

O *boxplot* (gráfico de caixa) é um gráfico utilizado para avaliar a distribuição empírica dos dados. O *boxplot* é formado pelo primeiro e terceiro quartil e pela mediana. Os extremos do gráfico indicam os valores mínimo e máximo, a menos que valores discrepantes (*outliers*) estejam presentes, nesse caso o gráfico se estende ao máximo de 1.5 vezes da distância interquartil. Estes pontos fora destes limites representam pontos fora de uma distribuição. O *boxplot* pode também ser utilizado para uma comparação visual entre dois ou mais grupos, onde duas ou mais caixas são colocadas lado a lado verificando a variabilidade entre elas, a mediana e assim por diante. Outro ponto importante é a diferença entre os quartis (Q3 - Q1) que é uma medida da variabilidade dos dados. A Figura 1 apresenta o *boxplot* com a presença de *outliers* e a Figura 2 apresenta o *boxplot* considerando todos os pontos, mas excluindo os *outliers*.

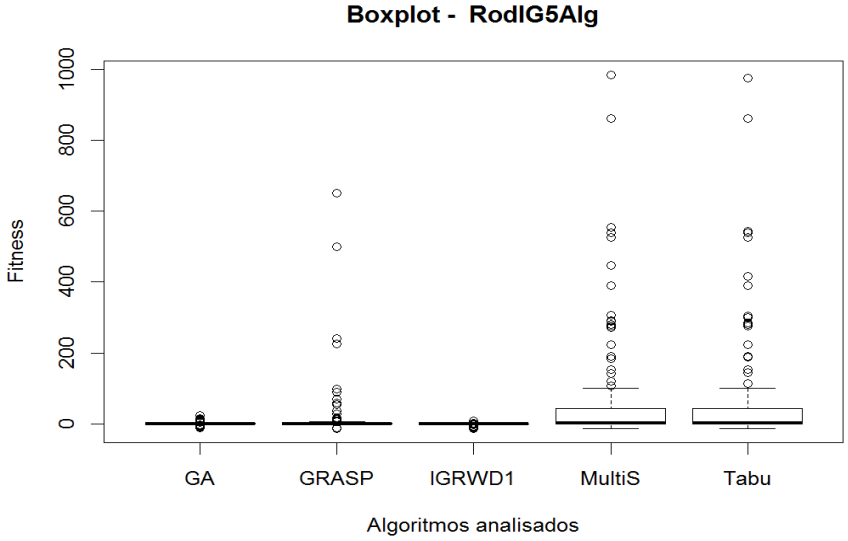


Figura 1 - Gráfico *boxplot* das amostras analisadas com *outliers*.

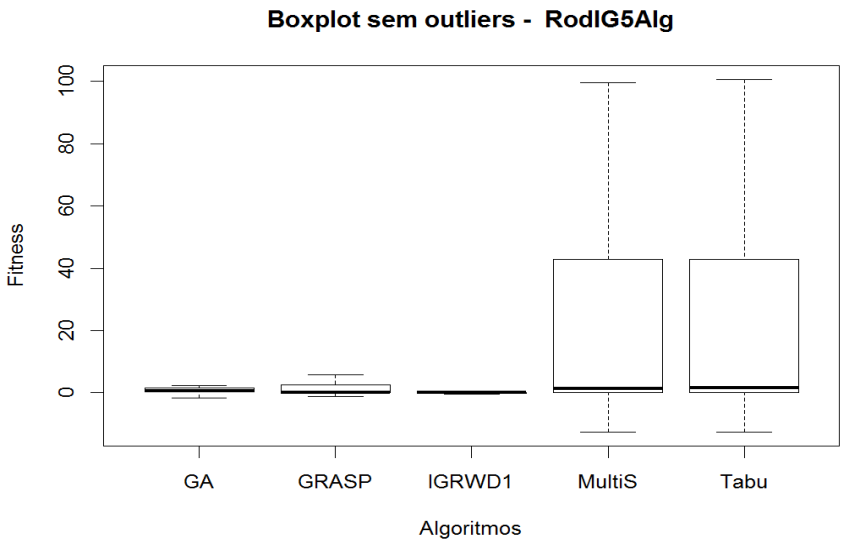


Figura 2 - Gráfico *boxplot* das amostras analisadas sem *outliers*.

2.2 - Gráfico da moda e da frequência da moda

Este gráfico mostra visualmente os valores da moda e da frequência da moda para cada amostra. O objetivo é verificar se existe um domínio da concentração da variável estudada no valor modal para cada amostra. A moda refere-se ao elemento da amostra que possui mais repetições e a frequência da moda indica quantas vezes este elemento aparece na amostra. Como os valores podem ter escalas diferentes o gráfico apresenta duas escalas para o eixo y, sendo que o eixo y a esquerda correspondendo a moda e o eixo y a direita correspondendo a frequência da moda.

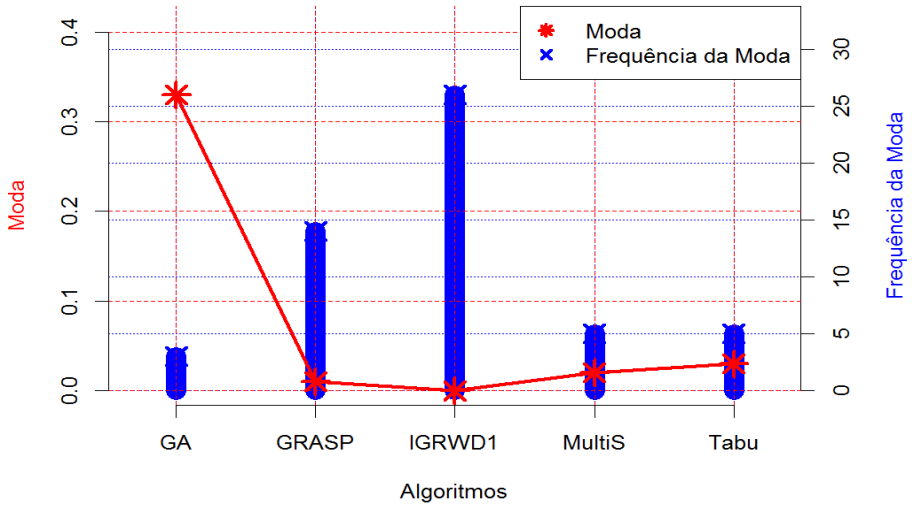


Figura 3 - Gráfico de moda e da frequência da moda.

2.3 - Gráfico dos valores de *fitness* por instâncias de cada algoritmo

Este gráfico mostra visualmente os valores de *fitness* de cada algoritmo pelas instâncias. O objetivo é visualizar como os algoritmos analisados se comportam em relação as instâncias testadas. A figura 4 apresenta os valores de *fitness* sobrepostos em um único gráfico. A figura 5 apresenta os gráficos separados dos valores de *fitness* de cada algoritmo.

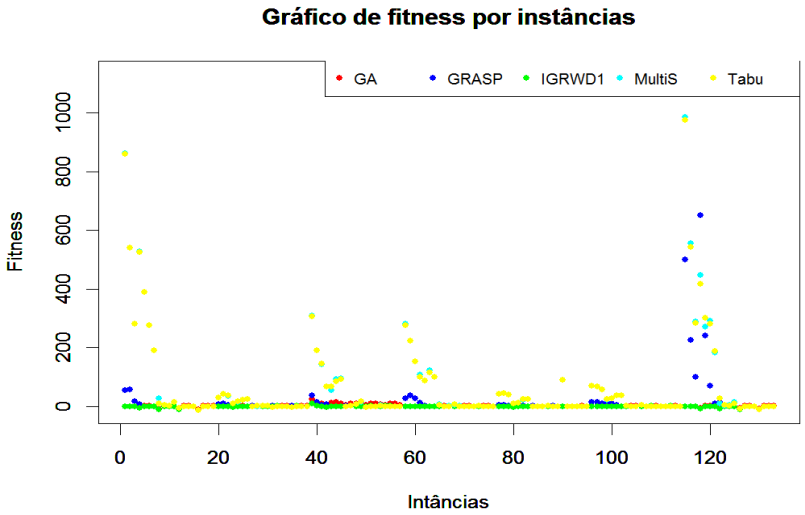


Figura 4 - Gráfico unificado de *fitness* por instâncias.

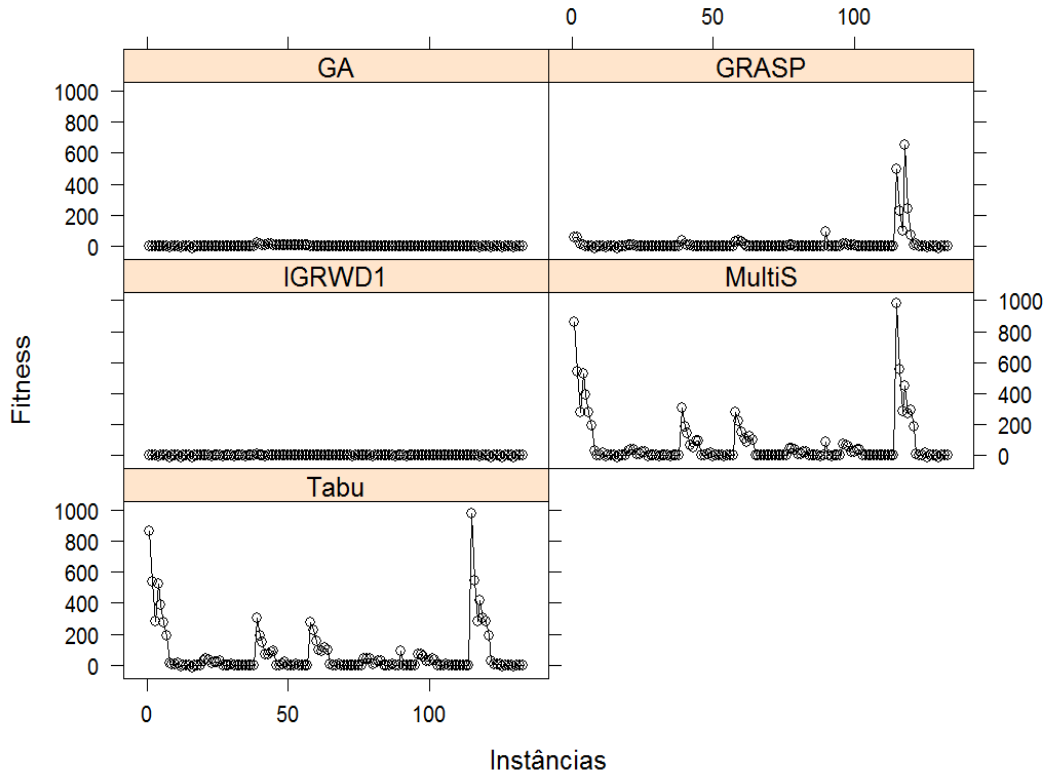


Figura 5 - Gráficos separados de *fitness* por instâncias.

2.4 - Gráfico de coordenadas paralelas das amostras com média e mediana

Um gráfico de coordenadas paralelas é usado para comparar os valores de dados que são de tipos ou magnitudes diferentes em uma única visualização. Os valores podem ser ajustados e, em seguida, apresentados como pontos em uma linha possibilitando a identificação de padrões entre os dados das amostras. Um gráfico de coordenadas paralelas considera que cada linha da tabela de dados é uma linha ou perfil que aparece no gráfico. Cada valor de uma linha é representada por um ponto na linha plotada. A Figura 6 apresenta o gráfico de coordenadas paralelas com valores não ajustados, principalmente porque também são apresentados os valores da média e da mediana de cada amostra.

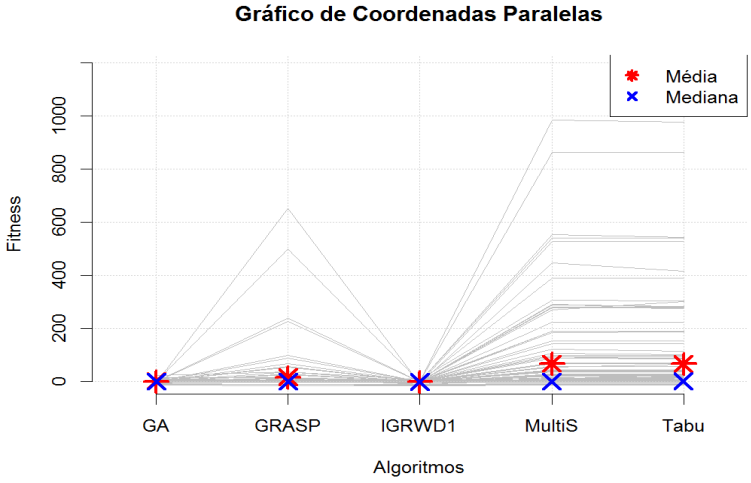


Figura 6 - Gráfico de coordenadas paralelas com média e mediana.

2.5 - Gráfico de dispersão com limites de erro padrão e média

Este gráfico apresenta em cada coluna uma visão da dispersão dos valores de *fitness* de cada algoritmo, com os valores limitantes do erro padrão e a média dos valores, permitindo uma análise mais detalhada da comparação entre as amostras.

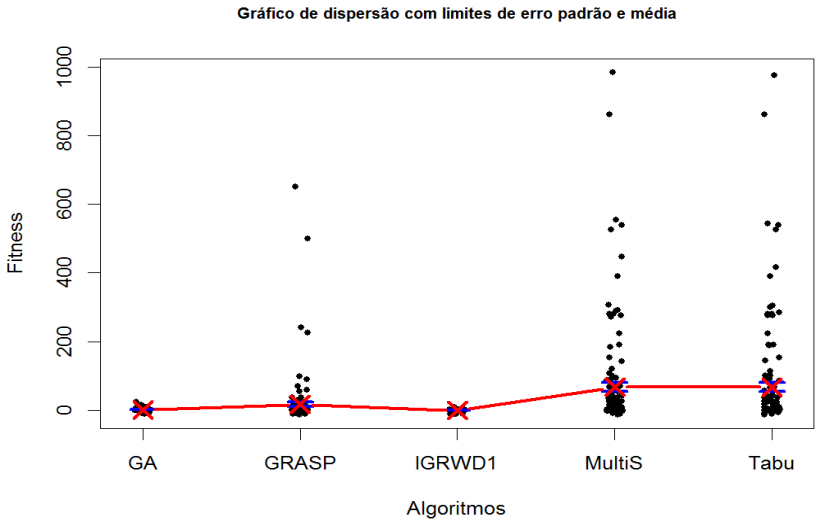


Figura 7 - Gráfico de dispersão com limites de erro padrão e média.

3 - Análise de normalidade das amostras

3.1 - Teste de normalidade das amostras

Os testes de normalidade são usados para determinar se um conjunto de dados de uma dada variável aleatória pode ser representada por uma distribuição normal ou não. O teste calcula a probabilidade da variável aleatória subjacente estar normalmente distribuída. Para os teste de normalidade foram utilizados os testes de Shapiro-Wilk (SW), para amostra de 0 a 4000 elementos e o teste de Lilliefors-Kolmogorov-Smirnov (LL) para amostras maiores que 4000 elementos. Em ambos os testes tem-se:
- Hipótese nula: a amostra avaliada é descrita por uma distribuição normal (p-valor é maior ou igual ao valor de significância estatística).
- Hipótese alternativa: a amostra avaliada não segue uma distribuição normal (p-valor é menor que o valor de significância estatística).

Tabela 3 - Teste de normalidade das amostra por Shapiro-Wilk (SW) e Lilliefors (Kolmogorov-Smirnov) (LL).

	Estatística	SW-p.value	Estatística	LL-p.value	Result. do teste
GA	W : 0.75189976	0	-	-	não normal
GRASP	W : 0.24513706	0	-	-	não normal
IGRWD1	W : 0.49307045	0	-	-	não normal
MultiS	W : 0.50609348	0	-	-	não normal
Tabu	W : 0.50631195	0	-	-	não normal

3.2 - Histogramas das amostras

O gráfico de histograma, também conhecido como distribuição de frequências ou diagrama das frequências, é uma representação gráfica, em colunas (retângulos), de um conjunto de dados previamente tabulado e dividido em classes uniformes. A base de cada retângulo representa uma classe e a altura de cada retângulo representa a quantidade ou frequência com que o valor dessa classe ocorreu no conjunto de dados. A construção de histogramas é de caráter preliminar

em qualquer estudo sendo um importante indicador da distribuição de dados em relação ao comportamento dos dados.

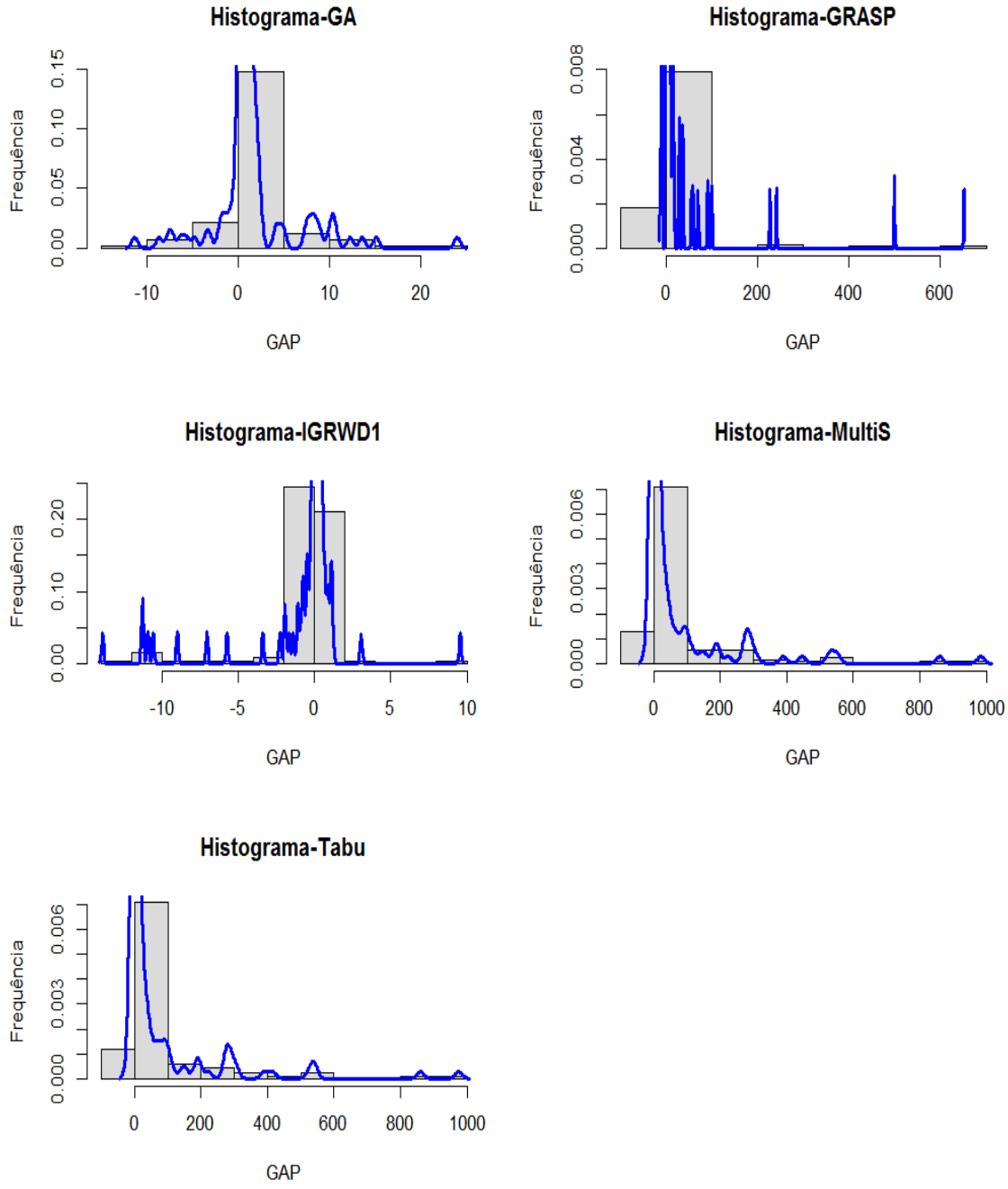


Figura 8 - Histograma das amostras analisadas.
Obs.: os gráficos individuais dos histogramas para cada amostra estão disponíveis na pasta '3-Histogramas-RodIG5Alg.

3.3 - Gráfico de comparação quartil-quartil das amostras

O gráfico de comparação de quartil (quartil-quartil plot ou q-q plot) também pode ser utilizado para determinar se um conjuntos de dados pertence a distribuição normal de probabilidade, sendo mais apropriado quando forem poucos dados analisados. Em tais gráficos os pontos são formados pelos quartis amostrais e se no resultado os pontos (dados da amostra) alinham-se em relação à reta apresentada, a distribuição da amostra pode ser consideradas como sendo normal.

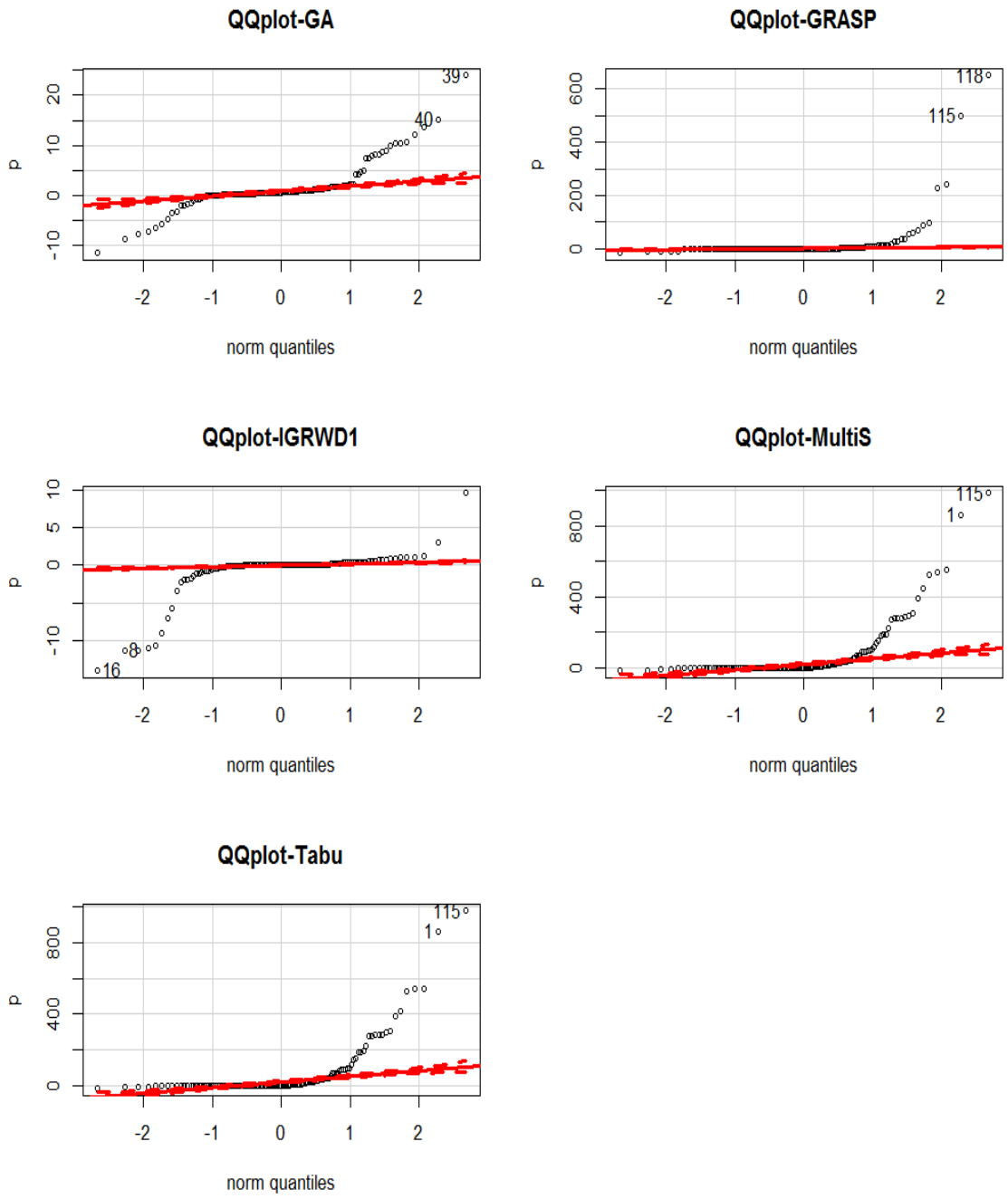


Figura 9 - Gráfico de comparação quartil-quartil das amostras analisadas.
Obs.: os gráficos individuais do q-q plot para cada amostra estão disponíveis na pasta '3-CompQuartil-RodIG5Alg.

4 - Análise gráfica comparativa entre as amostras

Gráfico *boxplot* das diferenças entre as amostras

Este gráfico apresenta um *boxplot* para as diferenças entre todos os pares das amostras analisadas, permitindo uma análise mais detalhada da comparação entre as amostras e fornecendo mais informações para uma análise gráfica mais detalhada. A Figura 10 apresenta

o *boxplot* das diferenças considerando os *outliers* dos dados e a Figura 11 apresenta o mesmo gráfico desconsiderando os *outliers* dos dados.

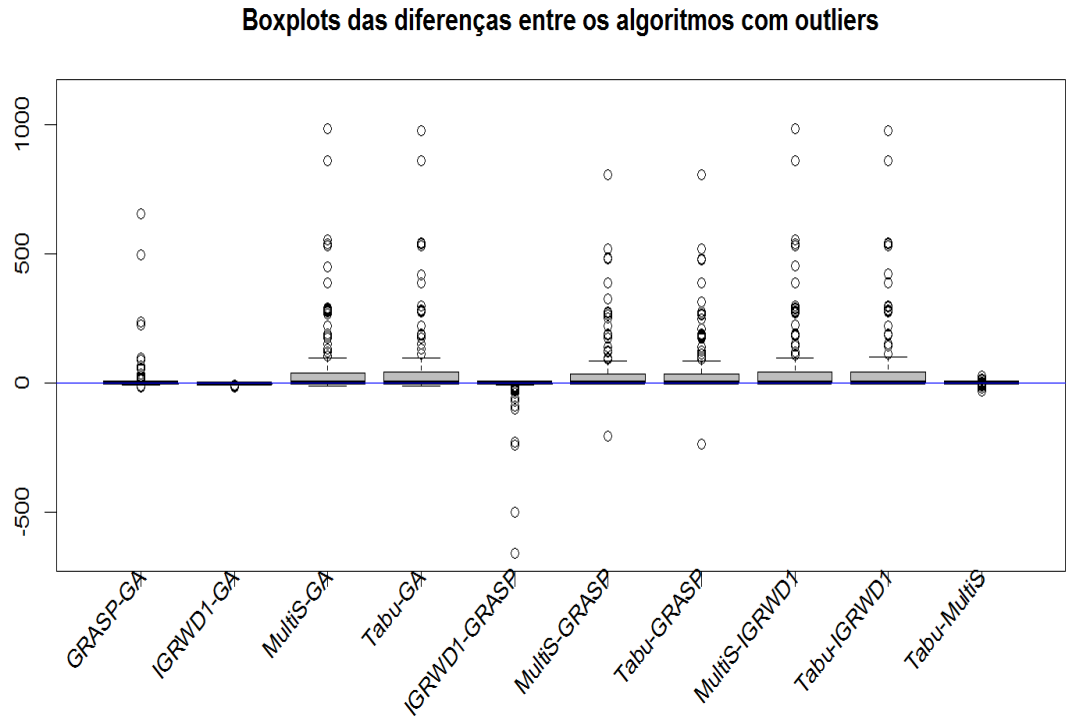


Figura 10 - Gráfico *boxplot* das diferenças entre as amostras com *outliers*.

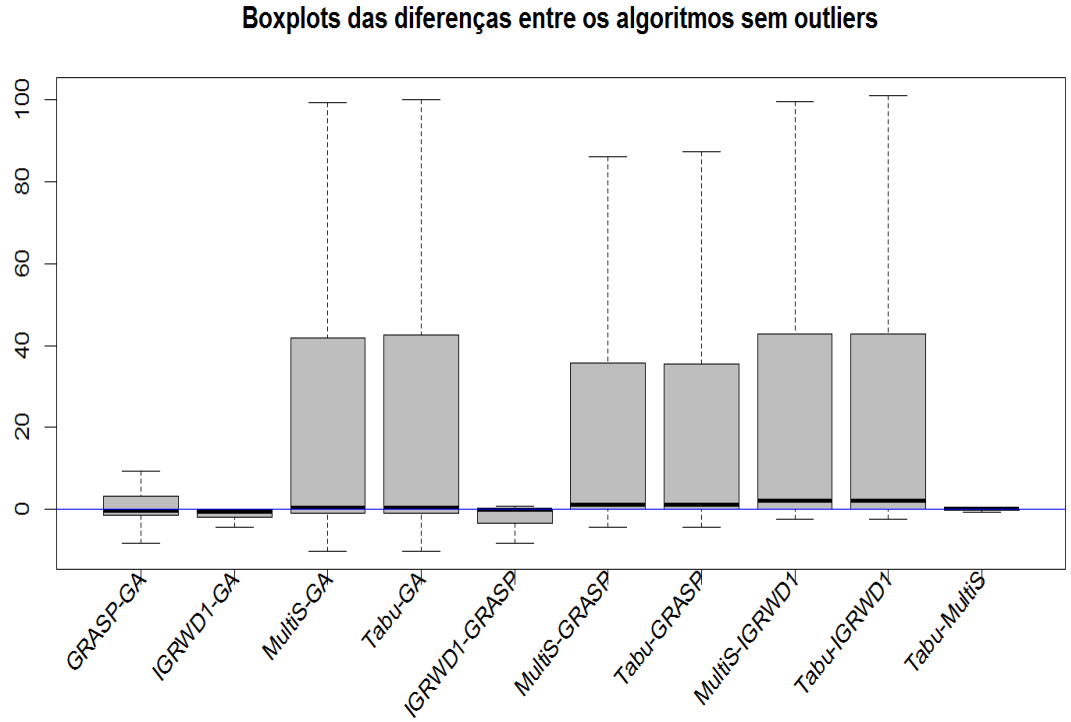


Figura 11 - Gráfico *boxplot* das diferenças entre as amostras sem *outliers*.

5 - Teste ANOVA com um fator e blocos e análise *post-hoc*

A análise de variância (ANOVA) é um teste paramétrico que visa, fundamentalmente, verificar a existência de uma diferença significativa entre as médias das amostras e detectar a influência dos fatores especificados. A análise compara médias de diferentes populações para verificar se essas populações possuem médias iguais ou não. Assim, essa técnica permite que vários grupos sejam comparados ao mesmo tempo. Em outras palavras, a análise de variância é utilizada quando se quer decidir se as diferenças amostrais observadas são reais (causadas por diferenças significativas nas populações observadas) ou casuais (decorrentes da mera variabilidade amostral). Portanto, essa análise parte do pressuposto que o acaso só produz pequenos desvios, sendo as grandes diferenças geradas por causas reais. Os pressupostos básicos da análise de variância são: as amostras devem ser aleatórias e independentes, as populações devem seguir uma distribuição normal e as variâncias populacionais são iguais.
As hipóteses nula e alternativa da análise a serem testadas na análise de variância são:
- Hipótese nula (H0): as médias populacionais são iguais.
- Hipótese alternativa (H1): as médias populacionais são diferentes, ou seja, pelo menos uma das médias é diferente das demais.

Se a hipótese alternativa for confirmada é necessário a execução de uma análise *post-hoc* para identificar as diferenças entre as amostras.
O modelo seguido pelo ANOVA é: µ + ßi + ?i (intercepto, efeito do algoritmo ou fator 1 efeito das instâncias ou bloco).
A análise de variância analisa vários grupos de observação classificados através de um fator e um bloco, ou seja, analisa os resultados encontrados classificados pelos algoritmos testados, que é um fator, comparando com outro fator que neste caso são as instâncias sendo denominados de bloco. Se os grupos forem pré-definidos no início tem-se um experimento com efeitos fixos, que é o caso aplicado na comparação entre vários algoritmos. Vale ressaltar que a análise de dois fatores é necessária porque as instâncias analisadas neste caso são dependentes, ou seja, como se está efetuando uma comparação entre algoritmos, um mesmo conjunto de instâncias é aplicado a cada algoritmo.

5.1 - Resultado geral do ANOVA com um fator e blocos.

Os resultados obtidos com a execução do teste ANOVA com um fator e blocos são:

```
           Df  Sum Sq Mean Sq F value Pr(>F)
alg         4  625024  156256  22.568 <2e-16 ***
inst      132 3505886   26560   3.836 <2e-16 ***
Residuals 528 3655755    6924
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

5.2 - Análise dos residuais do ANOVA com um fator

Estes gráficos mostram visualmente os valores dos residuais que foram calculados pelo ANOVA. Na Figura 12 são apresentados os gráficos de comparação de quartil (QQplot) dos residuais, o histograma dos residuais, o gráfico dos residuais por algoritmo em formato de *boxplot* e o gráfico dos residuais pelo valores de *fitness* dos algoritmos. A Figura 13 apresenta os valores dos residuais por instância em um único gráfico e na Figura 14 apresenta estes gráficos separadamente.

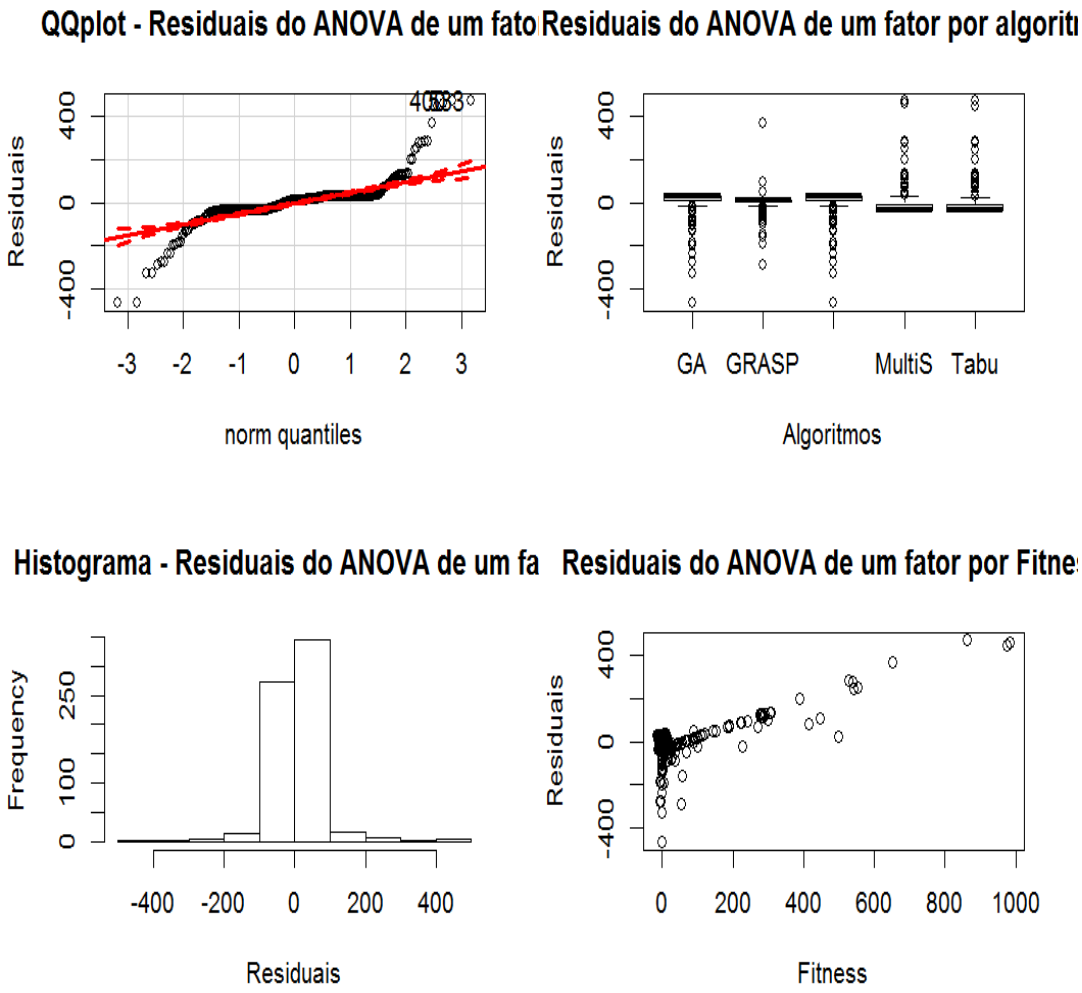


Figura 12 - Gráficos de residuais do ANOVA com um fator

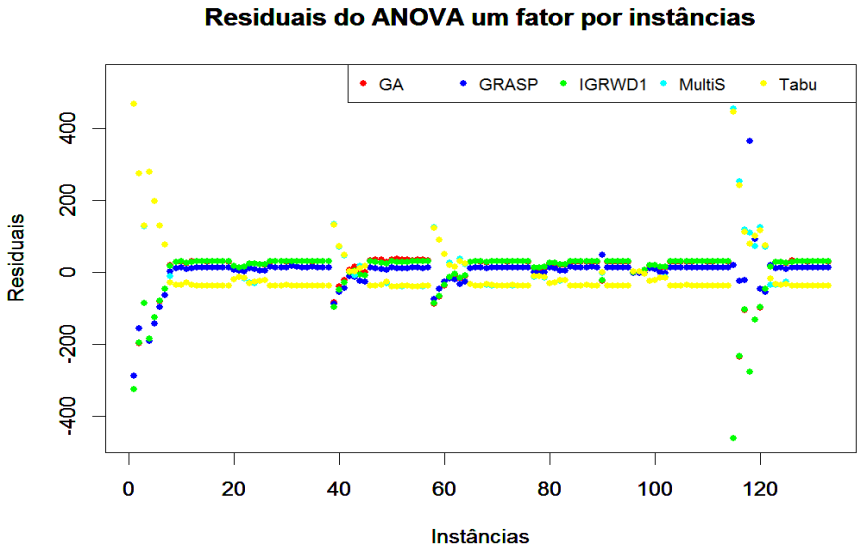


Figura 13 - Gráfico unificado dos residuais do ANOVA com um fator por instâncias.

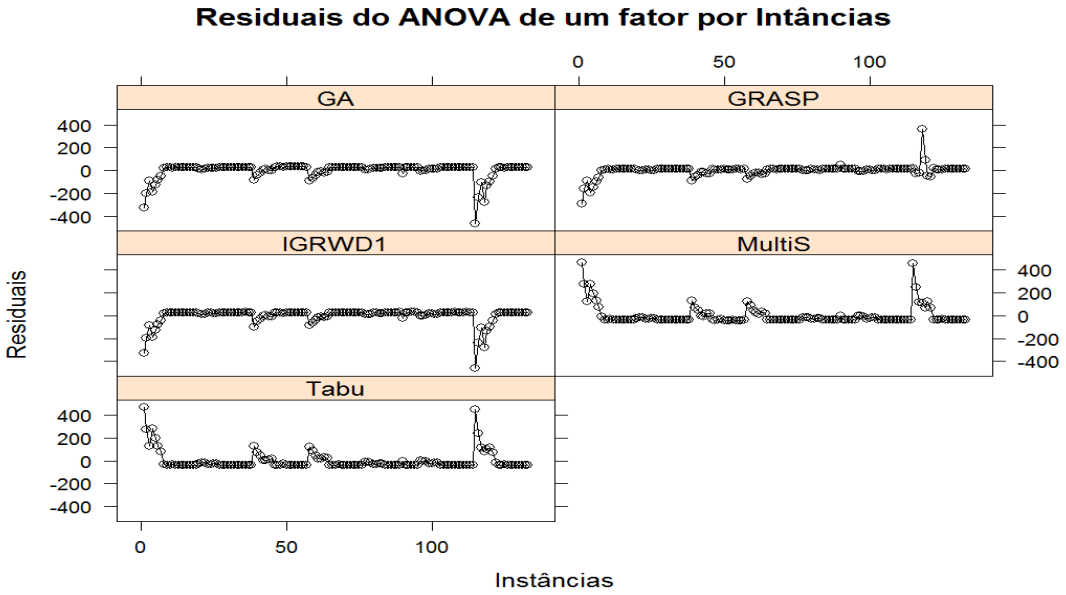


Figura 14 - Gráficos separados dos residuais do ANOVA com um fator por instâncias.

5.3 - Análise *Post-hoc* utilizando teste LSD

Esta análise *post-hoc* está baseada no teste LSD (Least Significant Difference - Menor diferença significante) (função LSD.test do pacote Agricolae, que implementa o teste de LSD e uma análise *post-hoc* através de múltiplas comparações dos tratamentos. A função apresenta no final uma tabela contendo a somatória do ranquemaneto e uma classificação destes valores em grupos específicos, que são definidos através do valor calculado da LSD.

Os resultados obtidos com a execução da análise *post-hoc* do teste ANOVA com um fator e blocos com o pacote Agricolae utilizando a função LSD.test(), são:

Study: dados.anova1fb ~ "alg"

LSD t Test for fit
P value adjustment method: bonferroni

Mean Square Error: 6923.779

alg, means and individual (95 %) CI

	fit	std	r	LCL	UCL	Min	Max
GA	1.4172932	4.246186	133	-12.756644	15.59123	-11.34	23.95
GRASP	17.0339850	76.855995	133	2.860047	31.20792	-13.62	651.21
IGRWD1	-0.5421805	2.687635	133	-14.716118	13.63176	-13.92	9.56
MultiS	67.5190977	155.996463	133	53.345160	81.69304	-12.65	984.61
Tabu	67.0822556	154.880149	133	52.908318	81.25619	-12.65	975.99

alpha: 0.05 ; Df Error: 528
Critical Value of t: 2.818882
Least Significant Difference 28.76323
Means with the same letter are not significantly different.

Groups, Treatments and means

a	MultiS	67.52
a	Tabu	67.08
b	GRASP	17.03
b	GA	1.417
b	IGRWD1	-0.5422

A Tabela 4 mostra os valores referentes as estatística do teste LSd para o ANOVA com um fator e blocos. São apresentados os valores da média (mean), do coeficiente de variabilidade (CV), o erro quadrático médio (MSerror) e o valor da menor diferença significativa que foi encontrada (LSD).

Tabela 4 - Estatística do teste *post-hoc* LSD para a ANOVA com um fator e blocos.

Mean	CV	MSerror	LSD
30.50209	272.7985	6923.77852	28.76323

A Tabela 5 mostra os valores referentes aos parâmetros utilizados no teste LSd para o ANOVA com um fator e blocos. São apresentados os valores dos graus de liberdade do erro experimental (Df), o número de tratamentos do experimento (ntr), que neste caso é o número de algoritmos utilizados na comparação e o valor do método de correção do p-valor definido pelo método de bonferroni.

Tabela 5 - Parâmetros do teste *post-hoc* LSD para a ANOVA com um fator e blocos.

Df	ntr	bonferroni
528	5	2.81888

A Tabela 6 mostra os valores referentes aos valores médios do teste LSd para o ANOVA com um fator e blocos. São apresentados os valores da média da amostra (fit), o desvio padrão (std), o tamanho da amostra analisada (r), limite de controle inferior (LCL), limite de controle superior (UCL), o valor mínimo da amostra (Min) e o valor máximo da amostra (Max).

Tabela 6 - Médias do teste *post-hoc* LSD para a ANOVA com um fator e blocos.

	fit	std	r	LCL	UCL	Min	Max
GA	1.41729	4.24619	133	-12.75664	15.59123	-11.34	23.95
GRASP	17.03398	76.85599	133	2.86005	31.20792	-13.62	651.21
IGRWD1	-0.54218	2.68763	133	-14.71612	13.63176	-13.92	9.56
MultiS	67.5191	155.99646	133	53.34516	81.69304	-12.65	984.61
Tabu	67.08226	154.88015	133	52.90832	81.25619	-12.65	975.99

A Tabela 7 mostra os valores referentes aos valores de agrupamento obtidos com o teste LSd para o ANOVA com um fator e blocos. Para cada algoritmo analisado (ou tratamento) são apresentados os valores do somatório dos ranques da média das amostras e o agrupamento que este algoritmo foi classificado. Esta classificação é importante porque pode identificar dentre os algoritmos comparados quem se destaca dos outros. Esta classificação é definida basicamente pelo valor do LSD encontrado, ou seja, o grupo é formado pelos valores médios dos algoritmos que estão com uma diferença de no máximo o valor LSD.

Tabela 7 - Agrupamentos do teste *post-hoc* LSD para a ANOVA com um fator e blocos.

	trt	means	M
1	MultiS	67.5190977443609	a
2	Tabu	67.0822556390977	a
3	GRASP	17.033984962406	b
4	GA	1.41729323308271	b
5	IGRWD1	-0.54218045112782	b

A Figura 15 apresenta os agrupamentos que foram obtidos com o teste LSd para o ANOVA com um fator e blocos.

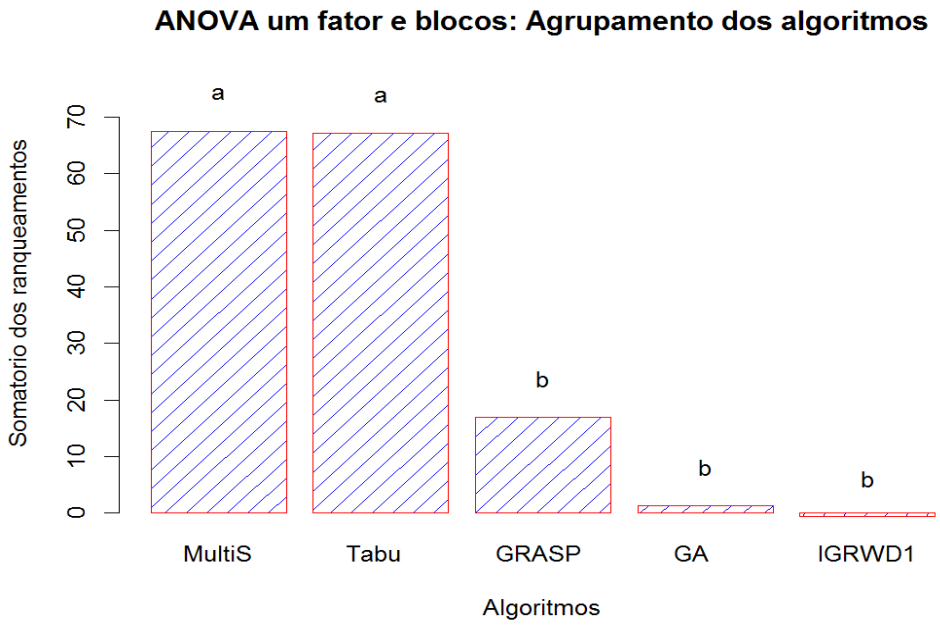


Figura 15 - Gráfico de barras com agrupamentos dos algoritmos após análise *post-hoc* do ANOVA com um fator e blocos utilizando teste LSD.

5.4 - Análise *Post-hoc* utilizando Tukey HSD

A análise *post-hoc* utilizando Tukey HSD cria um conjunto de intervalos de confiança sobre as diferenças entre as médias dos valores dos níveis de um fator com a probabilidade de cobertura dos grupos especificada. Os intervalos são baseados na faixa estatística t-Student e no método proposto por Tukey´s (Honest Significant Difference - Diferença Significante Honesta)
Os resultados obtidos com a execução da análise *post-hoc* do teste ANOVA com um fator e blocos utilizando teste de Tukey-HSD são:

Tukey multiple comparisons of means
95% family-wise confidence level
Fit: formula for ANOVA = fit ~ alg + inst, data = dadosColuna

Tabela 8 - Resultados do *post-hoc* do ANOVA com um fator e blocos utilizando teste Tukey HSD.

	diff	**lwr**	**upr**	**p adj**
GRASP-GA	15.61669	-12.31336	43.54674	0.54305
IGRWD1-GA	-1.95947	-29.88952	25.97058	0.9997
MultiS-GA	66.1018	38.17175	94.03185	0
Tabu-GA	65.66496	37.73491	93.59501	0
IGRWD1-GRASP	-17.57617	-45.50622	10.35389	0.42094
MultiS-GRASP	50.48511	22.55506	78.41516	1e-05
Tabu-GRASP	50.04827	22.11822	77.97832	1e-05
MultiS-IGRWD1	68.06128	40.13123	95.99133	0
Tabu-IGRWD1	67.62444	39.69439	95.55449	0
Tabu-MultiS	-0.43684	-28.36689	27.49321	1

Obs.: Os dados deste *post-hoc* também estão disponíveis no arquivo do Excel '5-PostHocAnova1fbTukeyHSD-RodIG5Alg.xlsx' na pasta de resultados.

5.5 - Análise *Post-hoc* utilizando Teste-t pareado

A análise *post-hoc* utilizando Teste-t pareado calcula a comparação entre os pares e os níveis de grupo com correção para testes múltiplos.
Os resultados obtidos com a execução da análise *post-hoc* do teste ANOVA com um fator e blocos utilizando teste-t pareado são:

```
        Pairwise comparisons using paired t tests
    Data: fit and alg
    P value adjustment method: bonferroni
```

Tabela 9 - Resultados do *post-hoc* do ANOVA com um fator e blocos utilizando teste-t pareado.

	GA	**GRASP**	**IGRWD1**	**MultiS**
GRASP	0.21418	-	-	-
IGRWD1	0	0.09628	-	-
MultiS	3e-05	5e-05	2e-05	-
Tabu	3e-05	6e-05	2e-05	1

Obs.: Os dados deste *post-hoc* também estão disponíveis no arquivo do Excel '5-PostHocAnova1fbTesteTPareado-RodIG5Alg.xlsx' na pasta de resultados.

6 - Teste de Friedman com análise *post-hoc*

O teste de Friedman é um teste não paramétrico equivalente aos experimentos em blocos randômicos (RBD - Randon Blocks Design) na análise de variância regular (ANOVA). Enquanto o teste ANOVA necessita que os dados estejam em uma distribuição normal e que possua variâncias iguais, o teste de Friedman é livre para estas restrições. Este teste utiliza os ranques dos dados ao invés de seus valores brutos para o cálculo da estatística de teste. Como o teste de Friedman não faz suposições sobre a distribuição, não é tão poderoso quanto o teste padrão se as populações forem realmente normais.
As hipóteses de análise do teste de Friedman para comparações através de medidas repetidas são:
- Hipótese nula (H0): As distribuições (amostras) são iguais em relação às medidas repetidas.
- Hipótese alternativa (H1): As distribuições(amostras) são diferentes.
O teste estatístico para Friedman é o teste chi-quadrado com [(número de repetições)-1] graus de liberdade.

6.1 - Resultado geral do teste de Friedman.

Os resultados obtidos com a execução do teste de Friedman são:

```
        Asymptotic Friedman Test
data:  fit by
        alg (GA, GRASP, IGRWD1, MultiS, Tabu)
        stratified by inst
chi-squared = 175.83, df = 4, p-value < 2.2e-16
```

6.2 - Análise *Post-hoc* utilizando Teste LSD

Esta análise *post-hoc* está baseada no teste LSD (Least Significant Difference - Menor diferença significante) (função friedman() do pacote Agricolae), que implementa o teste de Friedman e uma análise *post-hoc* através de múltiplas comparações dos tratamentos. Os dados consistem de 'v' variáveis aleatórias k-variadas mutuamente independentes chamados 'b' blocos. A variável aleatória é de um bloco e está associada com o tratamento. Um primeiro resultado é obtido por teste de Friedman disponível no R. A função apresenta no final uma tabela contendo a somatória do ranquemaneto e uma classificação destes valores em grupos específicos, que são definidos

através do valor calculado da LSD.
Referencias: Conover, W.J.; Practical Nonparametrics Statistics, Wiley, 1999.

Os resultados obtidos com a execução análise *post-hoc* do teste de Friedman, com o pacote Agricolae utilizando a função friedman(), são:

Study: Dados da pesquisa << RodIG5Alg >> em formato de Coluna

alg, Sum of the ranks

	fit	r
GA	471.5	133
GRASP	334.0	133
IGRWD1	218.0	133
MultiS	482.0	133
Tabu	489.5	133

Friedman's Test
==========
Adjusted for ties
Value: 175.8328
Pvalue chisq : 0
F value : 65.16584
Pvalue F: 0

Alpha : 0.05
t-Student : 1.964467
LSD : 41.19823

Means with the same letter are not significantly different.
GroupTreatment and Sum of the ranks

a	Tabu	489.5
a	MultiS	482
a	GA	471.5
b	GRASP	334
c	IGRWD1	218

A Tabela 10 mostra os valores referentes a estatística do teste LSd para o Friedman. São apresentados os valores do qui-quadrado (Chisq), do p-valor do qui-quadrado (p.chisq), o valor do teste-F (F), o p-valor do tetse-F (p,F) e o valor da menor diferença significativa que foi encontrada (LSD).

Tabela 10 - Estatística do teste *post-hoc* LSD para o Friedman.

Chisq	p.chisq	F	p.F	LSD
175.83282	0	65.16584	0	41.19823

A Tabela 11 mostra os valores referentes aos parâmetros utilizados no teste LSd para o Friedman. São apresentados os valores dos graus de liberdade do erro experimental (Df), o número de tratamentos do experimento (ntr), que neste caso é o número de algoritmos utilizados na comparação e o valor do t-student (t.value).

Tabela 11 - Parâmetros do teste *post-hoc* LSD para o Friedman.

	Df	ntr	t.value
	4	5	1.96447

A Tabela 12 mostra os valores referentes aos valores médios do teste LSd para o Friedman. São apresentados os valores da média da amostra (fit), o desvio padrão (std), o tamanho da amostra analisada (r), o valor mínimo da amostra (Min) e o valor máximo da amostra (Max).

Tabela 12 - Médias do teste *post-hoc* LSD para o Friedman.

	fit	std	r	Min	Max
GA	1.41729	4.24619	133	-11.34	23.95
GRASP	17.03398	76.85599	133	-13.62	651.21
IGRWD1	-0.54218	2.68763	133	-13.92	9.56
MultiS	67.5191	155.99646	133	-12.65	984.61
Tabu	67.08226	154.88015	133	-12.65	975.99

A Tabela 13 mostra os valores referentes aos valores de agrupamento obtidos com o teste LSD para o Friedman. Para cada algoritmo analisado (ou tratamento) são apresentados os valores do somatório dos ranques das médias das amostras e os agrupamentos que este algoritmo foi classificado. Esta classificação é importante porque pode identificar dentre os algoritmos comparados quem se destaca dos outros. Esta classificação é definida basicamente pelo valor do LSD encontrado, ou seja, o grupo é formado pelos valores médios dos algoritmos que estão com uma diferença de no máximo o valor LSD.

Tabela 13 - Agrupamentos do teste *post-hoc* LSD para o Friedman.

	trt	Sum of ranks	M
1	Tabu	489.5	a
2	MultiS	482	a
3	GA	471.5	a
4	GRASP	334	b
5	IGRWD1	218	c

A Figura 16 apresenta os agrupamentos que foram obtidos através das múltiplas comparações dos tratamentos com teste LSD para o teste de Friedman.

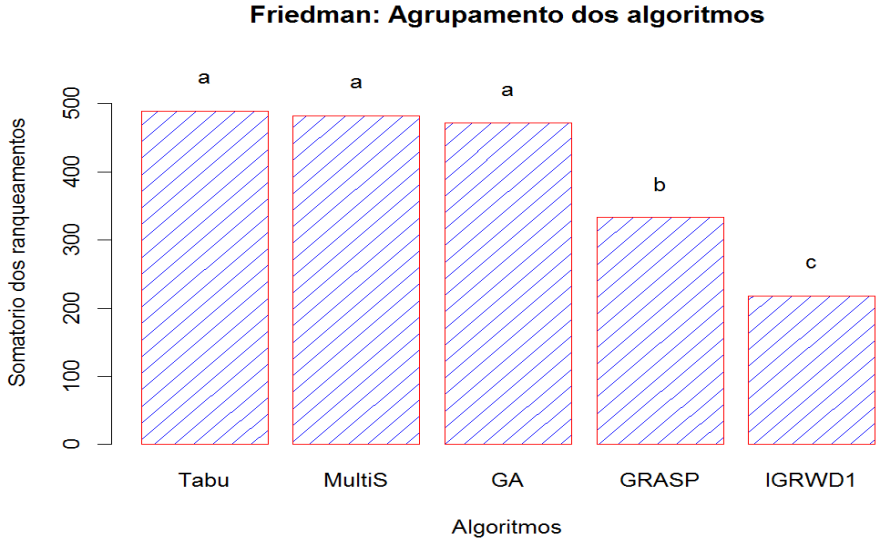


Figura 16 - Gráfico de barras com agrupamentos dos algoritmos após análise *post-hoc* do Friedman.

6.3 - Análise *Post-hoc* utilizando teste de simetria

Esta análise *post-hoc* está baseada no teste de simetria (função *post-hoc*-analysis-for-friedmans-test), que implementa a análise *post-hoc* proposta pelo teste de Wilcoxon-Nemenyi-McDonald-Thompson. A função executa testes utilizando uma abordagem baseada em permutações nas comparações em pares através da chamada da função symmetry_test() do pacote coin. Esta função permite escolher diferentes testes estatísticos, sendo utilizado o teste Tmax, que tem como objetivo central o controle da taxa de erro (FWER - familywise error rate) que significa a probabilidade de identificar um ou mais falsas descobertas, ou erro do tipo I (erro alfa), ao longo de todas as hipóteses quando se executa múltiplos testes de hipóteses. Se for considerado que foram realizadas 1000 permutações, para cada variável de interesse, então pode-se encontrar não somente os valores empíricos de p-valor (p-value) para cada variável, mas também um valor que representa o fato de que foram testadas todas essas variáveis ao mesmo tempo. O resultado final é conseguido através da comparação de cada teste estatístico observado em comparação ao valor máximo das estatísticas permutadas sobre todas as variáveis. Em outras palavras, este p-valor reflete a chance de se ver uma estatística de teste tão grande quanto a observada, dado que foram realizados inúmeros testes. A função apresenta uma tabela com os valores p-value das diferenças entre as amostras analisadas. Através destes valores pode-se identificar as diferenças entre as amostras possibilitando a identificação das melhores.
Os resultados obtidos com a execução da análise *post-hoc* para o teste de Friedman utilizando o teste de simetria são:
$Friedman.Test

Asymptotic General Symmetry Test

data: fit by
alg (GA, GRASP, IGRWD1, MultiS, Tabu)
stratified by inst
maxT = 10.633, p-value < 2.2e-16
alternative hypothesis: two.sided

Tabela 14 - P-value do *post-hoc* de Friedman utilizando teste de simetria.

	pvalue
GRASP - GA	5.08e-07
IGRWD1 - GA	0
MultiS - GA	0.994010647
Tabu - GA	0.955439045
IGRWD1 - GRASP	5.4499e-05
MultiS - GRASP	4.7e-08
Tabu - GRASP	2.9e-08
MultiS - IGRWD1	0
Tabu - IGRWD1	0
Tabu - MultiS	0.99837993

Obs.: Os dados deste *post-hoc* estão também disponíveis no arquivo do Excel '6-PostHocFriedmanSimetria-RodIG5Alg.xlsx' na pasta de resultados.

6.4 - Análise *Post-hoc* utilizando teste de Nemenyi

Esta análise *post-hoc* está baseada na função posthoc.friedman.nemenyi.test pertencente ao pacote PMCMR, que implementa a análise *post-hoc* proposta por Nemenyi test, executando o teste *post-hoc* em pares para múltiplas comparações das somas dos ranques das médias para dados em blocos não replicados. Este teste executa o teste *post-hoc* após confirmação do teste de Friedman que existe uma diferença significativa entre as amostras. A estatística do teste refere-se ao quartil superior da distribuição studentized range (Tukey). A função apresenta uma tabela cruzada com os valores p-value das diferenças entre as amostras analisadas. Através

destes valores pode-se identificar as diferenças entre as amostras possibilitando a identificação das melhores.

Os resultados obtidos com a execução da análise *post-hoc* para o teste de Friedman utilizando o teste de Nemenyi são:

Tabela 15 - P-value do *post-hoc* de Friedman utilizando teste de Nemenyi.

	GA	**GRASP**	**IGRWD1**	**MultiS**
GRASP	0	-	-	-
IGRWD1	0	7e-05	-	-
MultiS	0.99423	0	0	-
Tabu	0.95698	0	0	0.99844

Obs.: Os dados deste *post-hoc* estão também disponíveis no arquivo do Excel '6-PostHocFriedmanNemenyi-RodIG5Alg.xlsx' na pasta de resultados.

Referências de pacotes utilizados disponíveis no site do projeto R

R Core Team, ***R: A Language and Environment for Statistical Computing***, R Foundation for Statistical Computing, Vienna, Austria, 2015.
Disponível em: http://www.R-project.org/

Adrian A. Dragulescu, ***xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files***, , , 2014.
R package version 0.5.7. Disponível em: http://CRAN.R-project.org/package=xlsx

Hadley Wickham <hadley@rstudio.com> [aut, cre, cph], ***stringr: Simple, Consistent Wrappers for Common String Operations***, , , 2015.
R package version 1.0.0. Disponível em: http://CRAN.R-project.org/package=stringr

John Fox <jfox@mcmaster.ca> [aut, cre], ***RcmdrMisc: R Commander Miscellaneous Functions***, , , 2015.
R package version 1.0-3. Disponível em: http://CRAN.R-project.org/package=RcmdrMisc

Thorsten Pohlert, ***PMCMR: Calculate Pairwise Multiple Comparisons of Mean Rank Sums***, , , 2015.
R package version 1.2. Disponível em: http://CRAN.R-project.org/package=PMCMR

Michael E. Schaffer <mschaff@gmail.com> [aut, cre, cph], ***rtf: Rich Text Format (RTF) Output***, , , 2013.
R package version 0.4-11. Disponível em: http://CRAN.R-project.org/package=rtf

Deepayan Sarkar, ***Lattice: Multivariate Data Visualization with R***, , New York, 2008.
ISBN 978-0-387-75968-5. Disponível em: http://lmdvr.r-forge.r-project.org

Felipe de Mendiburu, ***agricolae: Statistical Procedures for Agricultural Research***, , , 2015.
R package version 1.2-2. Disponível em: http://CRAN.R-project.org/package=agricolae

Marek Gagolewski, ***R package stringi: Character string processing facilities***, , , 2015.
. Disponível em: http://stringi.rexamine.com/

Andri Signorell et mult. al., ***DescTools: Tools for Descripive Statistics***, , , 2015.
R package version 0.99.13. Disponível em: http://CRAN.R-project.org/package=DescTools

Rmetrics Core Team, ***fBasics: Rmetrics - Markets and Basic Statistics***, , , 2014.
R package version 3011.87. Disponível em: http://CRAN.R-project.org/package=fBasics

Juergen Gross <gross@statistik.tu-dortmund.de> [aut], ***nortest: Tests for Normality***, , , 2015.
R package version 1.0-4. Disponível em: http://CRAN.R-project.org/package=nortest

Referências de pacotes utilizados provenientes de artigos

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, Achim Zeileis. ***A Lego System for Conditional Inference***, The American Statistician, vol: 60/3, pag: 257--263, 2006.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, Achim Zeileis. ***Implementing a Class of Permutation Tests: The coin Package***, Journal of Statistical *Software*, vol: 28/8, pag: 1--23, 2008.
Disponível em: http://www.jstatsoft.org/v28/i08/

Torsten Hothorn, Frank Bretz, Peter Westfall. ***Simultaneous Inference in General Parametric Models***, Biometrical Journal, vol: 50/3, pag: 346--363, 2008.
Disponível em:

Ross Ihaka, Paul Murrell, Kurt Hornik, Jason C. Fisher, Achim Zeileis. ***colorspace: Color Space Manipulation***, , vol: /, pag: , 2015.
Disponível em: http://CRAN.R-project.org/package=colorspace

Achim Zeileis, Kurt Hornik, Paul Murrell. ***Escaping RGBland: Selecting Colors for Statistical Graphics***, Computational Statistics Data Analysis, vol: 53/, pag: 3259--3270, 2009.
OBS: As referencias listadas acima estão disponíveis no formato latex (bibiTex) no arquivo txt de resultados gerais ('ResultadosGerais-<<Nome da Pesquisa>>.txt') na pasta de resultados.

Session Information

R version 3.2.1 (2015-06-18)

Platform: *x86_64-w64-mingw32*

Locale:
LC_COLLATE=Portuguese_Brazil.1252;LC_CTYPE=Portuguese_Brazil.1252;LC_MONETARY=Portuguese_Brazil.1252;LC_NUMERIC=C;LC_TIME=Portuguese_Brazil.1252

Packages

Base: *base, datasets, graphics, grDevices, grid, methods, stats, utils*

Other: *agricolae (v1.2-2), car (v2.1-0), coin (v1.1-0), colorspace (v1.2-6), DescTools (v0.99.13), fBasics (v3011.87), formatR (v1.2), lattice (v0.20-33), manipulate (v1.0.1), multcomp (v1.4-1), mvtnorm (v1.0-3), nortest (v1.0-4), PMCMR (v1.2), RcmdrMisc (v1.0-3), rJava (v0.9-7), rtf (v0.4-11), sandwich (v2.3-4), stringi (v0.5-5), stringr (v1.0.0), survival (v2.38-3), svSocket (v0.9-57), TH.data (v1.0-6), timeDate (v3012.100), timeSeries (v3012.99), TinnRcom (v1.0.18), xlsx (v0.5.7), xlsxjars (v0.6.1)*

Loaded (not attached): *abind (v1.4-3), acepack (v1.3-3.3), AlgDesign (v1.1-7.3), boot (v1.3-17), class (v7.3-14), cluster (v2.0.3), coda (v0.17-1), codetools (v0.2-14), combinat (v0.0-8), deldir (v0.1-9), digest (v0.6.8), e1071 (v1.6-7), foreign (v0.8-66), Formula (v1.2-1), ggplot2 (v1.0.1), gridExtra (v2.0.0), gtable (v0.1.2), Hmisc (v3.17-0), klaR (v0.6-12), latticeExtra (v0.6-26), LearnBayes (v2.15), lme4 (v1.1-9), magrittr (v1.5), MASS (v7.3-44), Matrix (v1.2-2), MatrixModels (v0.4-1), mgcv (v1.8-7), minqa (v1.2.4), modeltools (v0.2-21), munsell (v0.4.2), nlme (v3.1-122), nloptr (v1.0.4), nnet (v7.3-11), parallel (v3.2.1), pbkrtest (v0.4-2), plyr (v1.8.3), proto (v0.3-10), quantreg (v5.19), R.methodsS3 (v1.7.0), R.oo (v1.19.0), RColorBrewer (v1.1-2), Rcpp (v0.12.1), readxl (v0.1.0), reshape2 (v1.4.1), rpart (v4.1-10), scales (v0.3.0), sp (v1.2-0), SparseM (v1.7), spdep (v0.5-88), splines (v3.2.1), stats4 (v3.2.1), svMisc (v0.9-70), tcltk (v3.2.1), tools (v3.2.1), zoo (v1.7-12)*

Session Details

Working directory: *C:/Users/Ronaldo/Dropbox/Doutorado - UFSCAR/Tese/Qualificação Tese/Análise Estatística de Dados/RodIG5Alg/Resultados-RodIG5Alg*

Output file: *ResultadosGerais-RodIG5Alg.doc*

APÊNDICE B - LISTA DE PACOTES UTILIZADOS NO DESENVOLVIMENTO DO *SCRIPT* EM LINGUAGEM R

O quadro abaixo mostra todos os pacotes com suas respectivas referências que foram utilizados no *script* desenvolvido em linguagem R do *framework* proposto.

Nome do Pacote	**Descrição**	**Referência**
coin	Implementação de procedimentos de inferência condicional comumente conhecidos como testes de permutação	(HOTHORN et al., 2006)
multcomp	Funções para inferência simultânea para modelos paramétricos gerais	(HOTHORN et al., 2008)
colorspace	Funções para controle e manipulação de cores e gráficos	(IHAKA et al., 2015)
xlsx	Coleção de funções para gravação, leitura e formatação de arquivos XLS do Microsoft Excel	(DRAGULESCU, 2014)
stringr	Coleção de funções para manipulação de *strings*	(WICKHAM, 2015)
RcmdrMisc	Conjunto de funções para manipulação do R comander	(FOX, 2015)
PMCMR	Contém funcionalidades para efetuar a comparação múltipla em pares do ranqueamento das médias	(POHLERT, 2015)
rtf	Conjunto de funções R para arquivos de saída Rich Text Format (RTF) contendo tabelas e gráficos de alta resolução que podem ser editados com um processador de texto padrão como o Microsoft Word.	(SCHAFFER, 2013)
lattice	Apresenta um conjunto de funções poderosas e elegantes na para visualização de dados suficiente para a maioria das necessidades gráficas de uma análise de dados.	(SARKAR, 2008)
agricolae	Contém a funcionalidade para a análise estatística dos projetos experimentais aplicados especialmente para experimentos de campo na agricultura e melhoramento de plantas.	(MENDIBURU, 2015)
stringi	Coleção de funções para manipulação de *strings* e caracteres.	(GAGOLEWSKI; TARTANUS, 2015)
DescTools	Coleção de funções estatísticas básicas e definições de conveniência para descrever dados de forma eficiente.	(SIGNORELL, 2015)
fBasics	Coleção de funções para explorar e investigar as propriedades básicas de retornos financeiros e quantidades relacionados.	(RMETRICS et al., 2014)
nortest	Coleção de funções para teste de normalidade.	(GROSS; LIGGES, 2015)

Fonte: elaborado pelo autor.