Giuliano Frascati

Otimização com Aprendizado Autônomo para Programação da Produção com Sequências de Instâncias Heterogêneas

São Carlos
2017

Giuliano Frascati

Otimização com Aprendizado Autônomo para Programação da Produção com Sequências de Instâncias Heterogêneas

Tese apresentada ao CCET (Centro de Ciências Exatas e Tecnológicas) da Universidade Federal de São Carlos - UFSCar, para a obtenção de Título de Doutor em Engenharia de Produção, na Área de Planejamento e Controle de Sistemas Produtivos.

Orientador: Roberto Fernandes Tavares Neto

São Carlos
2017

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 Giuliano Frascati, realizada em 17/03/2017:

Prof. Dr. Roberto Fernandes Tavares Neto
UFSCar

Prof. Dr. Carlos Dias Maciel
USP

Prof. Dr. Marcelo Seido Nagano
USP

Prof. Dr. Paulo Rogerio Politano
UFSCar

Prof. Dr. Walther Azzolini Júnior
USP

Dedico este trabalho aos professores que me orientaram para sua execução e também aos familiares e amigos que me apoiaram nesta jornada.

Think Higher!

Everyone pictures to become somebody
You know, I know
One of a few changes grime into honey
Guess how, guess how
Don't believe it's fame and fortune
Don't believe it's karma leading to the end of the game

Think higher!
Believe in your desire
Define that you'll be strong
And nothing can go wrong
Think higher!

Kings have been conquered by bondsmen and servants
You know, I know
All is achieved by your own persuasion
That's shown, that's shown
All you need you'll find inside you
Use it and you'll make your way into
The gain of the game

Think higher!
Believe in your desire
Define that you'll be strong
And nothing can go wrong
Think higher!
Obey your inner fire
Whatever you'll be through
The win belongs to you
Think higher!
The gain belongs to you

Andi Deris

Agradecimentos

O autor gostaria de agradecer a organização brasileira de fomento CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e o CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) – processos 475214/2013-7, 448161/2014-1 and 308047/2014-1 – e a FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) – processos 2016/05673-1 e 2016/05672-5 – por financiar esta pesquisa. Agradecendo também aos departamentos da UFSCar (Universidade Federal de São Carlos) e EESC/USP (Escola de Engenharia de São Carlos / Universidade de São Paulo) pelo apoio para alcançar estes objetivos.

Resumo

Muitos problemas de programação e sequenciamento da produção, mas também em Planejamento e Sequenciamento Avançados (PSA), são NP-Difíceis. Este projeto aborda um problema de *scheduling* em ambiente de máquina única com tempos de preparação dependentes da sequência das tarefas, sendo possível terceirizar parcialmente a demanda e restringindo que não existam atrasos para a entrega das ordens, com objetivo da minimização do custo total para a terceirização. Algoritmos Evolutivos (AE) representam uma estratégia de solução veloz para problemas NP-Difíceis. No entanto, pesquisadores da área de computação evolutiva afirmam que AEs dependem sensivelmente da configuração de seus parâmetros. Este trabalho investiga, na literatura de sequenciamento e PSA, como os autores que desenvolvem AEs determinam a parametrização e avaliação de seus algoritmos, pesquisando ainda quais são as estratégias sugeridas como estado da arte para a parametrização automática de AEs. Esta tese apresenta uma estratégia inovadora para configuração automatizada de AEs, incluindo um novo paradigma para otimização em sequências de instâncias heterogêneas chamado OAA (Otimização com Aprendizado Autônomo) que tem por objetivo solucionar de maneira integrada um problema de otimização e a parametrização do algoritmo configurável com um processo de decisão autônomo para detecção de heterogeneidades dentro da sequência de instâncias.

Palavras-chave: Sequenciamento da Produção, Planejamento e Sequenciamento Avançados, Algoritmos Evolucionários, Parametrização Automática, Otimização com Aprendizado Autônomo.

Abstract

Many scheduling problems, but also in Advanced Planning and Scheduling (APS), are NP-Hard. This project addresses a scheduling problem in a single machine environment with sequence dependent setup times. It is possible to partially outsource the demand restricting that there are no delays for delivering of the orders, aiming to minimize the cost for outsourcing. Evolutionary Algorithms (EA) represent a fast solution strategy for NP-Hard problems. However, researchers in the field of evolutionary computation say that EAs depend significantly on the configuration of their parameters. This work investigates in the scheduling and APS literature, how the authors who develop EAs determines the parameterization and evaluation of their algorithms, also presenting which strategies are suggested as state of the art for automatic tuning of EAs. This thesis states an innovative strategy for automated configuration of EAs, including a new paradigm for optimization in streams of heterogeneous instances called ALO (Autonomous Learning Optimization). This new paradigm aims to solve integratelly an optimization problem and parameterization of the configurable algorithm with an autonomous decision process for detecting heterogeneities within the sequence of instances.

Keywords: Scheduling, Advanced Planning and Scheduling, Evolutionary Algorithms, Automatic Tuning, Autonomous Learning Optimization.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ABC – *Artificial Bee Colony*

AC – Algoritmo de Corrida

ACO – *Ant Colony Optimization*

AE – Algoritmo Evolutivo

AEMO – Algoritmo Evolutivo Multi Objetivo

AG – Algoritmo Genético

AGA – *Adaptative Genetic Algorithm*

ALO – *Autonomous Learning Optimization*

AM – Aprendizado de Máquina

ANOVA – *Analysis Of Variance*

APS – *Advanced Planning and Scheduling*

BAPT – *Bilevel Automated Parameter Tuning*

B&B – *Branch-and-Bound*

BI – *Bucket Indexed*

BN – *Bayesian Networks*

BT – Busca Tabu

CACO – *Combinational Ant Colony Optimization*

CATS – *Combinatorial Auctions*

CBR – *Case-Based Reasoning*

CP – *Constraint Programming*

CPLEX – *IBM ILOG CPLEX Optimization Studio*

CRE – *Calibration and Relevance Estimation*

CRP – *Capacity Requirement Planning*

CRS4EAs – *Chess Rating System for Evolutionary Algorithms*

DE – *Differential Evolution*

DT – *Darwin Toolkit*

EA – *Evolutionary Algorithms*

ECJ – *Evolutionary Computation toolkit*

EDA – *Estimation of Distribution Algorithm*

EDD – *Earliest Due Date*

ELS – *Estimation-based Local Search*

ERP – *Enterprise Resourse Planning*

ES – *Evolution Strategies*

FIFO – *First In First Out*

FMS – *Flexible Manufacturing Systems*

FPC – *Fitness-Probability Clouds*

FPTAS – *Fully Polynomial-Time Approximation Scheme*

FSM – *Finite State Machines*

G-MLLP – *Goal - Mixed Linear Logic Programming*

GP – *Goal Programming*

HBMO – *Honey Bee Mating Optimization*

IRACE ou I/F-RACE – *Iterated Race for Automatic Algorithm Configuration*

ICA – *Imperialist Competitive Algorithm*

ILS – *Iterated Local Search*

LP – *Linear Programming*

LOOCV – *Leave-One-Out Cross Validation*

LPT – *Longest Processing Time*

M3 SCP – *Lawson M3 Supply Chain Planning*

MEA – *Memetic Evolutionary Algorithm*

MIP – *Mixed Integer Programming*

MILP – *Mixed Integer Linear Programming*

MKP – *Binary Multiple Knapsack*

MMP – *Multi-site Master Planning*

moGA – *multistage operation-based Genetic Algorithm*

MOGA – *Multi-Objetive Genetic Algorithm*

MOSA – *Multi-Objective Simulated Annealing*

MRP – *Material Requirements Planning*

MRP II – *Manufacturing Resource Planning*

MSP – *Model Selection Problems*

MTO – *Make-To-Order*

MTZ – *Miller–Tucker–Zemlin*

NSGA – *Non-dominated Sorting Genetic Algorithm*

OAA – Otimização com Aprendizado Autônomo

PCA – *Principal Component Analysis*

PID – *Proportional Integral Derivative*

PSA – Planejamento e Sequenciamento Avançado

PSO – *Particle Swarm Optimization*

QAP – *Quadratic Assignment Problem*

QCP – *Quasi-group Completion Problem*

QIEA – *Quantum-Inspired Evolutionary Algorithm*

RBFN – *Radial Basis Function Networks*

RD – Razão de Desempenho

REVAC – *Relevance Estimation and Value Calibration*

RFID – *Radio-Frequency IDentification*

RIP – *Root Identification Problem*

RHIM – *Random Heterogeneous Island Model*

SA – *Simulated Annealing*

SAPS – *Scaling And Probabilistic Smoothing*

SCOP – *Stochastic Combinatorial Optimization Problems*

SEE – *Simulated Economic Environment*

SM – *Surrogate Modelling*

SPEA – *Strength Pareto Evolutionary Algorithm*

SPEAR – *Fast Bit-vector Arithmetic Theorem Prover*

SPO – *Sequential Parameter Optimization*

SPT – *Shortest Processing Time*

SSGA – *Steady-State Genetic Algorithm*

SVM – *Support Vector Machine*

SW-GCP – *Graph Colouring Problem based on the Small World*

TI – *Time-Indexed*

TS – *Tabu Search*

TSP – *Traveling Salesman Problem*

TT – *Timetabling Problem*

UIO – *Unique Input Output*

VG – *Variable Greedy*

VNS – *Variable Neighborhood Search*

VRP-SD – *Vehicle Routing Problem with Stochastic Demands*

XML – *eXtensible Markup Language*

Sumário

Capítulo 1

Introdução

Segundo Moon et al. (2006), organizações manufatureiras estão evoluindo para cadeias globais com múltiplas plantas de produção e consistindo de facilidades fornecedoras, produtoras e montadoras. Por essa razão, o processo de planejamento e sequenciamento é complexo, devendo ser tratado ao longo de toda a cadeia de suprimentos buscando atingir alta qualidade dos produtos, mas minimizando custos e estoques, enquanto maximizando o desempenho do sistema produtivo.

Giacon e Mesquita (2011) argumentam que o principal objetivo de um sistema para a programação da produção é gerar um plano que atenda as exigências dos clientes de acordo com a capacidade da função produtiva. Indicadores de desempenho como: tempo de fluxo das ordens; tempo final de processamento (*makespan*); atraso total; estoque em processo; e utilização representam objetivos muitas vezes conflitantes e devem ser minimizados ou maximizados para obter-se um plano de produção coerente com os objetivos do sistema produtivo.

Em um *survey* com as empresas filiadas à Fiesp, no estado de São Paulo, Giacon e Mesquita (2011) constatam que a programação detalhada da produção é uma atividade complexa especialmente em empresas que operam com estratégia de produção contra pedido, ou *Make-To-Order* (MTO). Em sua pesquisa Giacon e Mesquita (2011) avalia-

ram 94 empresas, sendo 48% de pequeno porte (faturamento até R$ 50 milhões), 21% de médio porte (entre R$ 50 milhões e R$ 250 milhões) e 30% de grande porte (acima de R$ 250 milhões), 69% dos respondentes eram diretores e gestores da área de planejamento da produção e logística, concluindo que planejadores valem-se principalmente de estratégias manuais para o apontamento de ordens, ainda são raras as aplicações de otimização nas indústrias paulistas.

Shobrys e White (2000) classificam os sistemas para gestão de produção em três categorias temporais: a função de planejamento como a coordenação do sistema no longo prazo, semanas ou meses, os sistemas para sequenciamento compõem atividades que devem ser executadas em uma escala de dias ou semanas e os sistemas de controle em um horizonte de minutos ou segundos. Shobrys e White (2000) afirmam que, empresas como Exxon Chemicals, Dupont e McDonald atingiram bons resultados para a redução de custos e estoque focando-se agressivamente em melhorias na integração dessas funções.

Dentro das atividades de curto prazo, a programação de operações é um processo que toma a decisão que ocorre diretamente após serem decididos: quais serão os produtos produzidos; em qual escala eles serão fabricados e quais recursos serão alocados para sua produção. As decisões de sequenciamento assumem as respostas para estas perguntas e se tornam importantes em situações onde a capacidade dos recursos é fixa e associada a comprometimentos de investimentos no longo prazo Baker (1974).

De acordo com van Eck (2003), a evolução dos sistemas que dão suporte ao planejamento e controle para organizações parte do MRP (*Material Requirements Planning*) e CRP (*Capacity Requirement Planning*) evoluíram para sistemas chamados de MRP II (*Manufacturing Resource Planning*). Apesar das funções essenciais apresentadas por esses sistemas que permitem integrar os fluxo de informações dentro das organizações surgem os sistemas ERP (*Enterprise Resourse Planning*) que podem ser vistos, majoritariamente, como bases de dados providas com inúmeros tipos de aplicações.

Segundo Santa-Eulalia et al. (2011) as capacidades de planejamento em um sis-

tema ERP, apesar de fundamentais, são limitadas quando não potencializadas com a utilização de sistemas para Planejamento e Sequenciamento Avançado (PSA). De acordo com Hvolby e Steger-Jensen (2010), sistemas PSA não substituem o ERP, mas suplementam a funcionalidade de planejamento e sequenciamento presente em sistemas ERP, principalmente pela sua habilidade em simular diferentes cenários. As metas de um sistema PSA são informatizar e automatizar o processo de planejamento através do uso de simulação e otimização, permanecendo a tomada de decisão nas mãos dos planejadores que possuem experiência sobre a cadeia de suprimentos.

Para Musselman et al. (2002), o PSA consiste de um sistema capaz de planejar e replanejar rapidamente de maneira eficiente as operações de um sistema produtivo. Gruat-La-Forme et al. (2005) afirmam que o gerenciamento da cadeia de suprimentos tem ganho importância quando competitividade, responsividade e satisfação do cliente são palavras chave para o sucesso em um área de negócios. Segundo Hvolby e Steger-Jensen (2010), organizações que falharem em explorar suas bases de dados, criando e utilizando modelos computacionais, enfrentaram sérias desvantagens competitivas. Nanvala (2011) argumenta que uma boa função de sequenciamento deve tomar a decisão certa no momento correto, de acordo com as condições do sistema de manufatura.

Santa-Eulalia et al. (2011) argumentam que, apesar dos avanços evidenciados na literatura sobre a implementação e aplicação de sistemas PSA, ainda existem pontos onde melhorias são necessárias. Para Santa-Eulalia et al. (2011), dois aspectos principais devem ser considerados em um sistema PSA: a arquitetura (como o sistema é organizado, incluindo a "hierarquia" e o "planejamento integral") e o motor (que determina como o PSA realiza as suas funções de otimização).

A grande maioria das estratégias de solução aplicadas em sistemas PSA se focam em problemas NP-Difíceis, assim como reportado na literatura que trata apenas formulações para programação de tarefas. Nesse contexto, o presente trabalho aborda o problema NP-Difícil de *scheduling* em ambiente de máquina única com tempos de *setup* dependentes da

sequência e terceirização permitida, concentrando-se no enorme potencial da aplicação de meta-heurísticas para otimização em Engenharia de Produção.

Os autores em Hvolby e Steger-Jensen (2010), descrevem que em meados de 2000, os maiores fornecedores de ERP do mercado começaram a integrar sistemas PSA aos seus produtos. SAP e Oracle foram alguns dos primeiros a incluir essa funcionalidade, ambos baseados no sistema de otimização CPLEX da IBM (*IBM ILOG CPLEX Optimization Studio*).

Segundo Shen (2002), problemas de programação da produção são tipicamente NP-Difíceis e encontrar soluções ótimas é impossível sem um algoritmo essencialmente enumerativo onde, os tempos computacionais crescem exponencialmente em relação ao tamanho do problema.

Para problemas NP-Difíceis, estratégias tradicionais de resolução não são capazes de comprovar respostas ótimas em problemas de média e larga escala, devido ao alto consumo de recursos computacionais (tempo para execução e/ou memória para armazenagem de dados). Kung e Chern (2009) argumentam que, apesar de populares, modelos de programação inteira mista (MIP – *Mixed Integer Programming* – e MILP – *Mixed Integer Linear Programming*) são intratáveis para problemas complexos como o planejamento e sequenciamento em uma industria com múltiplas estruturas de produtos e múltiplos estágios de produção com tempos de preparação dependentes da sequência das tarefas. Mullen et al. (2009) revisam a literatura sugerindo o algoritmo baseado em colônias de formigas (ou ACO – *Ant Colony Optimization*) como uma boa alternativa para a obtenção de soluções precisas e em um tempo significativamente menor para problemas de otimização combinatória NP-Difíceis. Outros autores também observam que, para cenários com complexidade NP-Difícil, a meta-heurística ACO tem mostrado resultados promissores (por exemplo, Merkle e Middendorf (2003), Madureira et al. (2012), Tavares Neto (2012), Frascati (2014), Tavares Neto et al. (2015), M'Hallah e Alhajraf (2016)).

O ACO é um algoritmo que se baseia em trilhas de feromônios e no comportamento de formigas reais que as utilizam como um meio de comunicação. O desempenho desse tipo de algoritmo está fortemente ligado aos parâmetros utilizados para sua configuração, sendo que a descoberta de tais parâmetros não é uma tarefa trivial (Gambardella e Dorigo, 1996; Eiben e Smit, 2011a).

Essa forte dependência dos parâmetros também é notada em outras técnicas como Algoritmos Genéticos (AG), otimização por exames de partículas (PSO – *Particle Swarm Optimization*) e algoritmos estocásticos em geral (Eiben et al., 2007).

Nesses casos, a obtenção de soluções ótimas, ou quase ótimas, depende da escolha dos parâmetros que definem o funcionamento do algoritmo e das características da instância do problema sendo resolvida. Por exemplo, no caso de um AG simples com representação binária diversos parâmetros devem ser determinados para sua operação. Podem ser aplicadas diferentes estratégias de evolução, como mutação, seleção e recombinação, sendo ainda necessário definir parâmetros como tamanho da população, probabilidade da ocorrência de mutações, entre outros (Eiben e Smit, 2011b).

Este trabalho observa uma lacuna da literatura, muitos trabalhos não validam propriamente o desempenho de algoritmos com natureza estocástica, apresentam experimentos muito simplificados sobre conjuntos com números irrisórios de amostras. Observou-se ainda a inexistência de trabalhos que descreveram a aplicação de parametrização automática para AEs aplicados à solução de problemas complexos em Engenharia de Produção. O foco deste projeto é o estudo de técnicas para automatizar o processo de parametrização que prometem oferecer maior precisão para configuração de AEs com alto desempenho.

Para Birattari (2009), é senso comum aos praticantes da área que as meta-heurísticas são em geral sensíveis aos valores de seus parâmetros e sua afinação cuidadosa melhora significativamente o desempenho desses algoritmos. Tradicionalmente, o processo de parametrização é conduzido com experimentos de tentativa e erro. Tais procedimentos

consomem tempo e recursos humanos, sendo ainda sujeitos a erros que levam à um mau balanceamento dos parâmetros do algoritmo. Segundo Stützle et al. (2010) e Eiben e Smit (2011a), técnicas para a configuração automática são recentes e a literatura demonstra que esta é uma área de intensas pesquisas atualmente.

Birattari (2009) define o problema da parametrização de uma maneira prática que fornece os meios para modelar e resolver o problema da afinação de uma meta-heurística. Segundo Hutter et al. (2009), automatizar o processo da configuração de algoritmos é altamente relevante em diversos contextos.

- *Desenvolvimento de algoritmos complexos*: Configuração manual é uma tarefa que requer trabalho intensivo, podendo consumir grande parte do tempo total para o desenvolvimento do algoritmo, o processo automático pode reduzir tempo e fornecer melhores resultados que procedimentos manuais ou a incorporação de parâmetros definidos na literatura para um algoritmo semelhante;

- *Estudos empíricos e comparação de algoritmos*: Comparar duas estratégias de solução representa uma questão central para definir se determinado algoritmo é fundamentalmente superior a outro ou se ele está apenas melhor configurado. Processos automáticos podem mitigar comparações injustas entre algoritmos;

- *Aplicações práticas*: O desempenho de um algoritmo complexo normalmente depende dos parâmetros que definem sua operação, usuários finais possuem pouco conhecimento do impacto dos parâmetros sobre o desempenho dos algoritmos e normalmente aplicam as configurações padrões. Mesmo que cuidadosamente parametrizado para um determinado conjunto de instâncias o algoritmo pode obter resultados ruins para casos heterogêneos aos da fase de treinamento, o processo automático pode ser aplicado de maneira conveniente para melhorar o desempenho nesses casos.

Alguns exemplos de esforços para a criação de mecanismos que automatizam o

processo de parametrização automática são os trabalhos de Yuan e Gallagher (2007), Balaprakash et al. (2007), Nannen et al. (2008), Birattari (2009), Smit e Eiben (2010a), Eiben e Smit (2011a), Kattan e Arif (2012), Riff e Montero (2013), Ugolotti e Cagnoni (2014), Sinha et al. (2014) e Vecek et al. (2016). As estratégias desenvolvidas por esses autores visam, de modo geral, minimizar o tempo necessário para encontrar valores relativamente melhores para os parâmetros de maneira a conferir alto nível de desempenho ao algoritmo sendo configurado dado um conjunto de instâncias para treinamento.

Segundo Nannen e Eiben (2006), para problemas de sequenciamento, pode ser difícil determinar valores de parâmetros que garantam alta qualidade de solução para todas as possíveis instâncias do problema. Portanto, um AE com conjunto de parâmetros fixos pode precisar ser reconfigurado se as características das instâncias que o algoritmo resolve forem modificadas em relação aos problemas utilizados na fase de treinamento, ou afinação (*tuning*).

Segundo Smit e Eiben (2009), encontrar bons parâmetros para a operação de AEs representa um problema complexo que está além das capacidades humanas para sua solução manual, com uma função de otimização não linear, interação entre as variáveis, múltiplos ótimos locais, ruído (relacionado a característica estocástica natural desse tipo de algoritmo) e ainda a falta de pacotes para solução analítica.

Birattari (2009) define os problemas sequenciados que simulam a operação de um cenário prático com um horizonte de tempo para a aplicação de um algoritmo otimizador como uma "sequência" (*stream*) de instâncias. Dessa forma, o horizonte de operação para um cenário produtivo deve ser representado por uma sequência temporal de problemas cujos dados correspondam a operação do sistema por um determinado período.

A necessidade de reconfiguração do algoritmo de otimização pode se mostrar uma barreira para sua aplicação em casos reais, mesmo aplicando uma técnica automatizada é necessário conhecimento para detectar quando a reconfiguração é necessária. Configurar uma meta-heurística ou um algoritmo genético exige treinamento especializado para

realizar o processo de parametrização. Os softwares existentes são recentes e ainda carentes de interfaces amigáveis para utilização, configuração e interpretação dos seus resultados.

Observando que de acordo com a literatura avaliada até o momento não existe a descrição um algoritmo similar, este trabalho propõe um novo paradigma, chamado OAA (Otimização com Aprendizado Autônomo), capaz de tratar uma sequência de instâncias heterogêneas fornecendo respostas de alta qualidade em termos da velocidade de execução e dos objetivos sendo otimizados. Um sistema OAA é composto basicamente por um algoritmo para Aprendizado de Máquina (AM) e outro para a parametrização automática de AEs, além do próprio AE e uma estratégia de controle para guiar o aprendizado de máquina. Este paradigma define uma nova camada de implementação para resolver um problema NP-Difícil, esta conta com um algoritmo de otimização veloz e configurável e sua parametrização automática utilizando um algoritmo baseado em ACO para configurar o otimizador e uma estratégia baseada em AM para gerenciar o conhecimento acumulado com operação do algoritmo otimizador.

Os resultados apresentados a seguir, mostram que analisar AEs com diferentes configurações fornece um modelo mais robusto para a otimização de problemas combinatórios altamente complexos, principalmente com a ocorrência de instâncias heterogêneas e para cenários onde é imperativo obter respostas com alta velocidade.

Com um sistema responsável por parametrizar um algoritmo de busca configurável e outro capaz de detectar instâncias heterogêneas é possível automatizar não só a obtenção de valores para os parâmetros ajustáveis do algoritmo, mas também a tomada de decisão que desencadeia tal processo. A camada OAA tem como objetivo facilitar a aplicação de AEs para a solução de problemas combinatórios complexos, evitando a necessidade de configuração prévia e ainda tornando o processo de parametrização invisível ao usuário final, mas minimizando a necessidade de configuração do algoritmo otimizador para potencializar seu desempenho.

Focando-se em um problema de *scheduling* e aproveitando algoritmos previamente implementados em Frascati (2014) e Frascati et al. (2014), esta pesquisa explora profundamente as oportunidades de otimização proporcionadas por algoritmos evolutivos como o ACO.

Esta pesquisa destina-se a um problema em Engenharia de Produção. No entanto, é importante destacar que o foco deste projeto está em *otimizar o algoritmo otimizador* aplicado ao problema. Portanto, a mesma estratégia de otimização aqui apresentada pode ser aplicada a outros elos da cadeia de suprimentos que originam problemas combinatórios complexos, bem como para formulações que integram mais de um componente a ser otimizado, como o problema integrado que relaciona o sequenciamento das tarefas e a distribuição dos itens finais, ou ainda em outras áreas da ciência que apresentam problemas NP-Difíceis e demandam soluções rápidas para sequências de instâncias heterogêneas.

Tanto Frascati (2014) e Frascati et al. (2014) como Tavares Neto et al. (2015), Tavares Neto e Godinho Filho (2013), Tavares Neto (2012), Tavares Neto e Godinho Filho (2012), Tavares Neto e Godinho Filho (2011b), Tavares Neto e Godinho Filho (2011a) e Tavares Neto (2010) representam esforços para o desenvolvimento de meta-heurísticas baseadas em ACO para problemas de sequenciamento, apontando a necessidade do aprofundamento das pesquisa sobre a configuração automática de algoritmos. Observa-se também que a parametrização é um tema recorrente e atual na área de AEs (Eiben et al., 1999; Birattari et al., 2002; Balaprakash et al., 2007; Yuan e Gallagher, 2007; Birattari, 2009; Stützle et al., 2010; Eiben e Smit, 2011a; Pellegrini et al., 2012; Yeguas et al., 2014; Rabanal et al., 2015; Aleti e Moser, 2016).

Nesse contexto, este projeto é apresentado da seguinte forma: o capítulo 2 constrói uma revisão de literatura destacando as práticas para configuração de AEs encontradas em trabalhos que propõem sua aplicação para otimização em problemas de *scheduling* e PSA, enquanto o capítulo 3 trata propriamente a parametrização de algoritmos. O

capítulo 4 descreve a proposta de solução para o problema apresentado, compreendido pela integração de dois problemas, um representado pela otimização de uma função objetivo e o segundo da definição dos melhores parâmetros para a configuração do algoritmo de otimização, possibilitando encontrar respostas com alta qualidade e de maneira veloz para instâncias desconhecidas. Os experimentos conduzidos para avaliar o sistema desenvolvido e a análise dos resultados estão descritos no capítulo 5. Por fim, o capítulo 6 faz as considerações finais pertinentes ao projeto e direciona potenciais pesquisas futuras.

Capítulo 2

Otimização e Aplicações de AEs em Engenharia de Produção

Este capítulo concentra-se na análise dos trabalhos relacionados com a otimização para problemas de programação e sequenciamento da produção em ambiente de máquina única, mas também aplicações em modelos para PSA. Focando-se no processo de desenvolvimento dos algoritmos evolutivos implementados, mas principalmente nos meios utilizados para configuração desses algoritmos estocásticos e sobretudo na validação do seu desempenho.

Apesar da difusão de estratégias evolutivas para solução de problemas combinatórios, muitos autores aplicam algoritmos exatos para casos com complexidade NP-Difícil, onde estratégias tradicionalmente enumerativas como *Branch-and-Bound* (B&B) não são capazes de comprovar respostas ótimas para problemas de larga escala caso existam restrições para o tempo de execução e/ou recursos de memória para armazenar as informações do algoritmo. Davari et al. (2015) aplica uma estratégia B&B para o problema da minimização do atraso total ponderado, quando as tarefas possuem instantes de liberação distintos, concluindo que método é eficaz para instâncias com até 50 tarefas. Por exemplo, Wang et al. (2015a) comparam um algoritmo de programação dinâmica

e um B&B para o problema de sequenciamento que envolve dois agentes de máquina única onde deseja-se minimizar o atraso total em uma das máquinas enquanto para a outra máquina o atraso máximo não deve exceder um valor determinado.

Desenvolvendo um B&B para minimização do tempo total de fluxo para um problema de máquina única com restrições de precedência, Pereira (2016) implica que o algoritmo é capaz de resolver instâncias entre 25 e 40 tarefas com até uma hora de processamento, mas apenas para instâncias com características específicas, ou seja, determinados grupos de instâncias apresentaram maior dificuldade para a estratégia de otimização aplicada.

Keshavarz et al. (2015) investigam a complexidade do problema de sequenciamento com tempos de preparação dependentes da sequencia onde as tarefas são sequenciadas em grupos, também em ambiente de máquina única. Esses autores mostram que o problema tratado é NP-Difícil e implementam um B&B para solucioná-lo, avaliando o algoritmo em instâncias geradas aleatoriamente contendo entre 10 e 25 tarefas.

Boland et al. (2016) desenvolvem um novo modelo MILP para uma variação do problema com tempo indexado (TI – *Time-Indexed*) chamada *Bucket Indexed* (BI) e comparam os dois modelos para 125 instâncias de 30 tarefas. Para os experimentos, Boland et al. (2016) definem uma restrição de 1800 segundos para o término das execuções que aplicam o *solver* CPLEX 12.3, sendo capaz, dentro do tempo limite, de comprovar as respostas ótimas para 82% dos casos de teste e ainda falhado em encontrar respostas factíveis para instâncias específicas.

Outros autores, como Yin et al. (2016a) e Zhao (2016) apresentaram soluções baseadas em técnicas matemáticas exatas, um modelo MILP e um algoritmo de programação dinâmica, respectivamente. Esses autores afirmam que soluções baseadas em métodos exatos que buscam comprovar respostas ótimas, podem apresentar dificuldades para instâncias de médio e grande porte caso o tempo execução do algoritmo seja restrito.

Considerando que cada uma das tarefas pertence a uma família de produtos com

tempos de preparação dependentes da ordem das famílias sequenciadas, Herr e Goel (2016) afirmam que, para problemas com mais de 10 tarefas, o modelo MIP implementado encontra dificuldades em comprovar respostas ótimas, ou seja, o algoritmo se complica enquanto tenta aproximar o intervalo entre os limitantes máximo e mínimo para a solução do problema. Para contornar a complexidade da estratégia MIP, Herr e Goel (2016) apresentam uma heurística polinomial que supera os resultados da formulação MIP.

Um volume considerável de publicações apresentam estratégias heurísticas, baseadas em algoritmos construtivos, que são capazes de fornecer soluções razoáveis e velozes. No entanto, a qualidade das respostas geradas depende do conhecimento de métodos eficientes para as especificações dos problemas de otimização tratados.

Han et al. (2015) trabalham com 10 diferentes modelos de sequenciamento *on-line* para ambientes de máquina única e máquinas paralelas. Em um problema de sequenciamento *on-line* as informações de cada tarefa não são conhecidas antes do momento de sua chegada e devem ser programadas imediatamente. Han et al. (2015) comparam suas heurísticas para diferentes formulações do problema com outras tradicionais, afirmando que, os algoritmos desenvolvidos são velozes e aplicáveis em problemas de larga escala. Porém, apenas para quatro das 10 formulações apresentadas foi possível desenvolver, comprovando matematicamente, uma heurística polinomial baseada nas peculiaridades da formulação do problema que fosse capaz de determinar respostas ótimas para todas as instâncias possíveis.

Ji et al. (2015) apresentam um problema de programação em ambiente de máquina única com formação de um número limitado de lotes e deterioração das tarefas, características presentes na indústrias de componentes metálicos produzidos por extrusão. Ji et al. (2015) provam que o problema é NP-Difícil, mas que é possível desenvolver um algoritmo polinomial de aproximações (FPTAS – *Fully Polynomial-Time Approximation Scheme*) para o modelo. No entanto, sem a restrição de um número limitado de

lotes o problema se torna mais complexo e um algoritmo de aproximações não pode ser definido.

Laalaoui e M'Hallah (2016) tratam do problema combinatório descrito como múltiplas mochilas binárias (*binary multiple knapsack* – MKP) para programar ordens em um sistema produtivo que envolve paradas para manutenção com objetivo minimizar a penalidade para ordens com atraso, onde cada tarefa tem um peso diferente para a penalidade do descumprimento dos prazos de entrega. Esses autores exploram as vantagem da estrutura de dados em formato de lista encadeada para a implementação de métodos para exploração de vizinhança a partir do algoritmo denominado VNS (*Variable Neighborhood Search*).

Outros autores que buscam solucionar problemas complexos de programação da produção, como Ji e Li (2015), Yuan et al. (2015), Zhao e Tang (2015),Fang e Lu (2016), Gu et al. (2016), Lazarev et al. (2016), Li e Lu (2016), Herr e Goel (2016), Lu e Zhang (2016), Ou et al. (2016), Shioura et al. (2016), Wu e Cheng (2016), Wu et al. (2016), Xingong e Yong (2016), Yin et al. (2016b), Zhao (2016) e Zhao et al. (2016), descrevem heurísticas com complexidade polinomial para variadas configurações de problemas em ambiente de máquina única. Neste ponto, é importante destacar que algoritmos polinomiais baseiam-se nas características específicas das formulações do problemas, dificilmente podem ser generalizados para outros casos e muitas vezes respostas ótimas podem ser encontradas apenas para grupos com determinadas características de instâncias.

Nanvala (2011) revisa a aplicação de algoritmos genéticos para solucionar problemas de sequenciamento em sistemas de manufatura flexíveis (FMS – *Flexible Manufacturing Systems*). O autor encontra evidências de que estratégias híbridas entre AG e procedimentos de busca local ou mesmo outras meta-heurísticas é comum para aplicações em FMS e a literatura é dividida entre autores que abordam problemas multi critério e outros que tratam objetivos singulares de otimização.

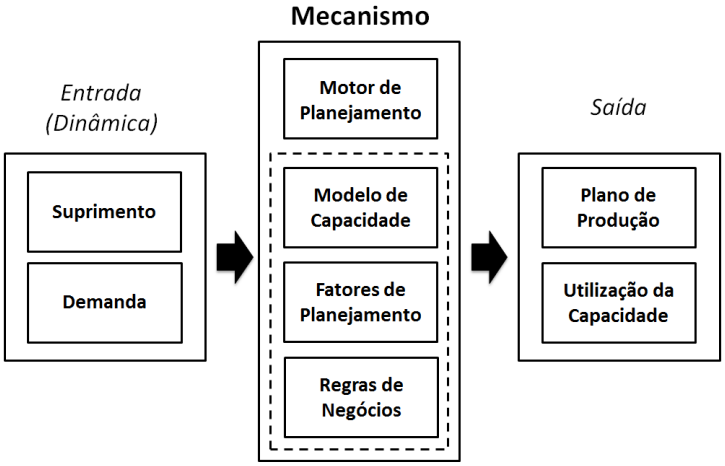


Figura 2.1: Arquitetura básica de um sistema PSA. Alterações nos dados dinâmico forçam o replanejamento. Fonte: adaptado de Lin et al. (2007)

A abordagem do sistema PSA considera um conjunto de dados estático e outro dinâmico. O dados estáticos contém o modelo de capacidade com a estrutura de produtos, lista de materiais, rotas de produção, etc. Mas também dados sobre fatores de planejamento como os tempos de ciclo e ainda regras de negócios. Já os dados dinâmicos são compostos por suprimento, seja de produtos finais ou estoque em processo, e a demanda. Como resultado o sistema é capaz de oferecer um plano de produção e a utilização dos recursos para tal.

A figura 2.1 ilustra o funcionamento do sistema. Os dados de entrada tem características dinâmicas, ou seja, a mudança dessas variáveis desatualiza planos anteriores ou em processamento (Lin et al., 2007).

Um sistema PSA proporciona a oportunidade de avaliar todas as informações em uma cadeia de suprimentos em tempo real. Segundo Fleischmann et al. (2004), o planejamento da produção é realizado de forma muito mais precisa com o uso do PSA, trazendo diversos benefícios como por exemplo a redução do chamado "efeito chicote", ocasionado pela lentidão no trafego de informações na cadeia. Diferentemente dos sistemas ERP, a tecnologia PSA procura por planos factíveis e potencialmente ótimos

ao longo da cadeia de suprimentos considerando explicitamente os gargalos do sistema produtivo.

Segundo van Eck (2003), PSA são sistemas de planejamento revolucionários que baseiam-se em uma larga gama de restrições (disponibilidade de material, capacidade de recursos, prazos de entrega, gerenciamento de estoque, custo, distribuição, sequenciamento, etc.) para produzir um plano otimizado para produção.

Para Fleischmann et al. (2004) e Santa-Eulalia et al. (2011), sistemas PSA possuem três características principais:

1. *Planejamento integral*: planejamento completo da cadeia de suprimentos;

2. *Otimização Real*: com a utilização de métodos exatos ou heurísticos, os sistemas APS buscam obter soluções ótimas ou quase ótimas para os problemas de decisão;

3. *Sistemas hierárquicos de planejamento*: planejamento ótimo para toda a cadeia de suprimentos é impraticável, no entanto, planejamento hierárquico representa o compromisso entre o praticável e a dependência entre as etapas de planejamento.

No estudo de caso tratado por Rudberg e Thulin (2009), é aplicado o sistema PSA *Lawson M3 Supply Chain Planning* (M3 SCP), para MMP (*Multi-site Master Planning*), para o qual os autores analisam as implicações da sua aplicação para o planejamento tático, no médio prazo. O M3 SCP aplica programação linear (*Linear Programming* - LP) e programação inteira mista (MIP) para minimizar os custos de produção.

Shen (2002) abordam um mecanismo de leilões para gerenciar recursos e argumentam que, em um ambiente de manufatura, os planos dificilmente ocorrem como esperados. Ordens serão canceladas e novas também podem ser inseridas, recursos podem se tornar indisponíveis ou serem adquiridos. Para Musselman et al. (2002), a simulação é capaz de oferecer planos para programação de operações altamente realistas, no entanto, afirma que a utilidade de um plano para sequenciamento detalhado degenera rapidamente com

o tempo, uma vez que paradas podem acontecer no chão de fábrica ou mesmo mudanças como ordens urgentes ou cancelamentos de pedidos.

A degeneração dos dados dinâmicos implica que a solução de otimização aplicada à esse tipo de problema deve ser veloz o suficiente para evitar que mudanças no chão de fábrica ocorram antes do processamento para otimização do último cenário.

Santa-Eulalia et al. (2011) argumentam que, apesar dos avanços evidenciados na literatura sobre a implementação e aplicação de sistemas PSA, ainda existem pontos onde melhorias são necessárias. Segundo Santa-Eulalia et al. (2011), em um sistema PSA dois aspectos principais devem ser considerados: a arquitetura (como o sistema é organizado, incluindo a "hierarquia" e o "planejamento integral") e o motor (que determina como e quão veloz o PSA realiza as suas funções de otimização).

Ivert (2012) analisa como o sistema PSA se aplica às atividades de produção e controle: a liberação de ordens, sequenciamento, despacho e o monitoramento. Esse autor discute que, apesar de relacionado principalmente à ambientes de produção *job shop* por grande parte da literatura, algoritmos sofisticados para sequenciamento aplicam-se também em outros ambientes, caso as formulações sejam NP-Difíceis.

Embora sistemas PSA sejam normalmente associados a ambientes fabris, seu uso é muito diversificado. Kortenkamp (2003) descreve a complexidade envolvida no planejamento e programação das atividades diárias de astronautas em missões espaciais, afirmando que mesmo em missões não tripuladas, planejamento e execução são tarefas de tempo e trabalho intensivos. O autor reflete sobre a necessidade de sistemas PSA mais automatizados e flexíveis que suportem a operação de processos com maior complexidade. Mais adiante, Policella et al. (2014) apresentam um sistema de planejamento e sequenciamento avançado aplicado à programação de operações sobre um satélite que orbita a terra.

Outros autores, como Errington (1997), Chen e Chen (2002), Gruat-La-Forme et al. (2005), David et al. (2006), Voudourist et al. (2006), Dorne et al. (2008), Hadaya e

Pellerin (2008), Liu e Qi (2008), Malik e Qiu (2008), Hvolby e Steger-Jensen (2010) e Policella et al. (2014), também analisam as vantagens e barreiras da aplicação de sistemas PSA enquanto discutem casos industriais específicos, como a manufatura de componentes eletrônicos, indústria automobilística, produção de celulose e papel entre outros processos fabris, mas também organizações prestadoras de serviços baseados principalmente em recursos humanos.

Segundo Kuroda et al. (2002), uma vez que novas ordens estejam disponíveis o sistema será capaz de gerar um plano para sua execução, porém, se ordens forem canceladas antes de sua execução o replanejamento será necessário. Autores como Chua et al. (2006), Chen et al. (2013) e Montesco et al. (2015), desenvolvem algoritmos para sistemas PSA baseados em regras heurísticas para priorização e ordenação de ordens, seleção de máquinas, definição de recursos, entre outros aspectos de decisão envolvidos.

Os trabalhos de Ning et al. (2008), que exploram maior flexibilidade para a construção de restrições em modelos de otimização com um *framework* em XML (*eXtensible Markup Language*), Li e Ma (2010), que pesquisam e constroem modelos para previsão de demanda e Zhong et al. (2013), que desenvolve aplicações em tempo real utilizando tecnologia RFID (*Radio-Frequency IDentification*), não estão propriamente relacionados ao foco central desta pesquisa, mas representam esforços para o aprofundamento do conhecimento em relação à sistemas PSA.

Um modelo MIP é apresentado por Chen e Ji (2007b) para o problema que integra as funções de planejamento e sequenciamento em um sistema PSA sobre um ambiente *job shop*. O problema proposto por esses autores é NP-Difícil, apenas uma instância foi avaliada, contendo 23 operações a serem sequenciadas provenientes de duas ordens com produtos distintos e, apesar do sucesso em obter-se um solução ótima através desse modelo, foram necessárias mais de 11 horas para execução do exemplo numérico apresentado por Chen e Ji (2007b) que gerou 498 restrições e 337 variáveis (sendo 282 inteiras).

Kung e Chern (2009) tratam de modelos para sistemas PSA que envolvem problemas em ambiente *flow shop* e *job shop* apresentando uma metodologia construtiva para a solução dos problemas. O processo desenvolvido pelos autores é comparado com técnicas simples de ordenação como FIFO (*First In First Out*), EDD (*Earliest Due Date*) e LPT (*Longest Processing Time*), mas também com um modelo MIP.

Steger-Jensen et al. (2011) argumentam que em geral, sistemas de planejamento e sequenciamento sofrem da complexidade de ambientes *job shop*, classe de problemas que é preponderantemente NP-Difícil, mostrando que para um problema com 5 máquinas e 100 tarefas (quantidade típica em uma situação prática) existirão 50000 restrições e 25250 variáveis, tornando pouco atrativa a aplicação de técnicas para otimização tradicionais. Esses autores aplicam G-MLLP (*Goal - Mixed Linear Logic Programming*), que representa uma estratégia de decomposição com uso dos métodos MILP e CP (*Constraint Programming*) em conjunto com a técnica chamada *Goal Programming* (GP), como ferramenta de otimização para um sistema APS.

G-MLLP permite apoiar-se nas propriedades das formulações MILP e CP, observando que restrições fortes em MILP podem ser modeladas como fracas em CP, e vice-versa. Dessa maneira, o modelo apresentado visa minimizar a violação das metas determinadas, ou restrição objetivo (*goal constrain*). Nesse modelo, o usuário especifica uma meta para os custos, ou capacidade disponível para sequenciar a produção. Steger-Jensen et al. (2011) descrevem a decomposição do modelo entre as etapas de planejamento (problema mestre) e sequenciamento (problema escravo). A figura 2.2 foi utilizada pelos autores para descrever a estratégia de decomposição aplicada.

Ornek et al. (2010) e Ozturk e Ornek (2014), também descrevem formulações matemáticas que consideram restrições de capacidade, prazos de entrega, múltiplas sequências de fluxo, máquinas e itens. Ambos os autores propõem a modelagem MIP para a solução dos problemas tratados, que envolvem sistemas PSA.

No restante deste capítulo, serão analisados trabalhos que tratam AEs para a solução

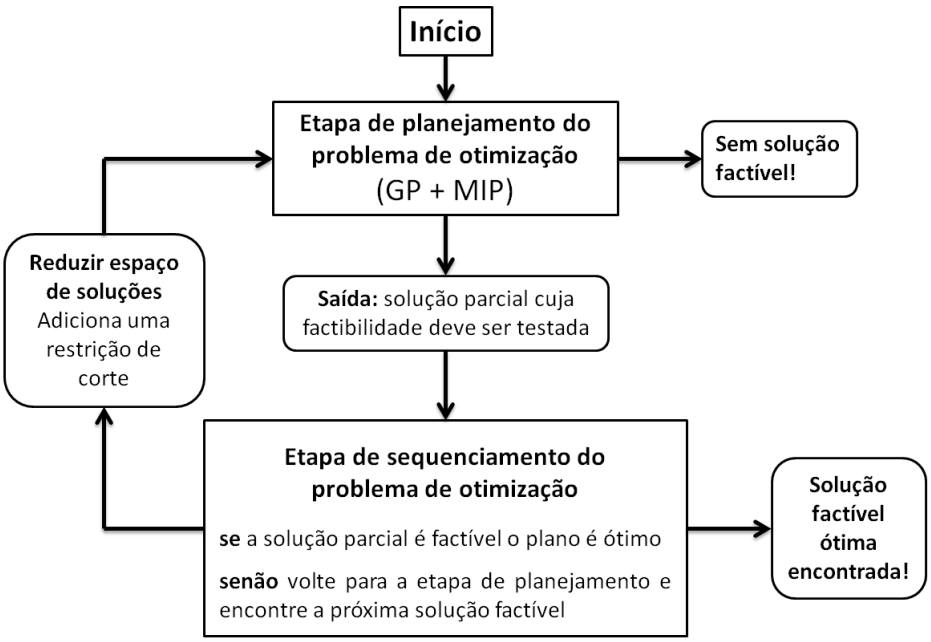


Figura 2.2: Exemplo para Estratégia de Decomposição. Fonte: adaptado de Steger-Jensen et al. (2011)

de problemas combinatórios complexos de programação da produção, mas também planejamento e sequenciamento avançados, com foco nas abordagens propostas para a parametrização e avaliação dos algoritmos implementados. Observa-se que dentre as pesquisas encontradas, nenhum dos autores aplicou estratégias automatizadas para parametrização e muitos avaliam esses algoritmos estocásticos em conjuntos irrisórios de instâncias.

2.1 AEs para Sequenciamento em Máquina Única

Nesta seção são apresentados trabalhos que tratam problemas de máquina única. Foram pesquisados artigos através do portal de busca *www.engineeringvillage.com*, com os termos *single machine* e *scheduling*, restringindo a busca aos anos recentes de 2016 e 2015. Os trabalhos resultantes foram filtrados encontrando-se nove artigos com implementações de AEs.

Conforme descrito anteriormente, o desempenho de AEs depende sensivelmente dos parâmetros que definem sua operação, mas também de características das instâncias submetidas ao algoritmo (Eiben e Smit, 2011a). Tradicionalmente, pesquisadores realizam o processo para configurar seus AEs manualmente (Stützle et al., 2010). No entanto, a parametrização desse tipo de algoritmo representa um problema complexo e susceptível a erro (Hutter et al., 2009).

Tratando um problema de sequenciamento onde as tarefas chegam em momentos distintos para produção, mas são entregues em lotes, Ahmadizar e Farhadi (2015) utilizam regras de dominância para guiar a convergência das respostas do AE desenvolvido, um ICA (*Imperialist Competitive Algorithm*). Um total de 100 instâncias geradas aleatoriamente foi utilizado para avaliação do ICA. Ahmadizar e Farhadi (2015) conduzem experimentos manuais para determinar a melhor configuração de parâmetros para operação do ICA, onde cada possível conjunto de parâmetros foi testado em 10 instâncias selecionadas ao acaso.

Kim e Choi (2015) tratam o problema de sequenciamento que envolve duas máquinas paralelas que foram modeladas com diferentes restrições, onde o objetivo é minimizar o atraso total ponderado na segunda máquina, uma vez que para a primeira não pode entregar tarefas em atraso. Esses autores desenvolvem um algoritmo B&B para obter respostas ótimas e um AG buscando velocidade de solução, uma vez que a estratégia exata consome muito tempo para a obtenção das respostas ótimas para esse problema que é NP-Difícil. Kim e Choi (2015) determinam experimentalmente valores para os parâmetros que definem a operação do AG, porém, não descrevem tais experimentos. Os algoritmos foram avaliados sobre 720 instâncias com tamanho variando entre 8, 10 e 12 tarefas, devido a limitação do algoritmo B&B para obtenção de respostas ótimas.

Moghaddam et al. (2015) definem um problema de sequenciamento com a possibilidade de rejeição das tarefas. Esses autores tratam um modelo em ambiente de máquina única com múltiplos objetivos que busca a minimização do tempo de fluxo

das tarefas aceitadas e da penalidade total para as rejeitadas. Moghaddam et al. (2015) aplicam oito meta-heurísticas diferentes para o problema multi objetivo formulado, todas derivadas dos algoritmos MOSA (*Multi-Objective Simulated Annealing*) e NSGA-II (*Non-dominated Sorting Genetic Algorithm*), que foram parametrizadas manualmente pelos autores, no entanto, não são descritos os experimentos de configuração. Para avaliação dos algoritmos os autores geraram aleatoriamente 66 instâncias com tamanhos variando entre cinco e 200 tarefas.

Para a minimização do atraso total, Wu et al. (2015) tratam um problema que considera uma curva de aprendizado (*learning effect*) determinando que os tempos de processamento das tarefas diminuem com o tempo, significando que os recursos melhoram seu desempenho de acordo com uma curva de aprendizado. Para solucionar o problema Wu et al. (2015) desenvolvem um algoritmo B&B e comparam sua solução com três variações de AGs com parâmetros definidos manualmente através de experimentos preliminares. Esses autores avaliam 1800 instâncias com 15 e 20 tarefas, além de 1800 entre 50 e 100 tarefas, mostrando que a solução baseada em B&B apresenta resultados satisfatórios em relação ao tempo computacional apenas para o primeiro conjunto de instâncias enquanto as soluções baseadas em AGs são eficientes em obter respostas sub ótimas e representam estratégias velozes que podem ser aplicadas em instâncias de maior porte.

Gonçalves et al. (2016) implementam três meta-heurísticas (ILS – *Iterated Local Search*, VG – *Variable Greedy* e SSGA – *Steady-State Genetic Algorithm*) para o problema de sequenciamento com custo de atraso quadrático ponderado e discutem que algoritmos baseados em B&B são capazes de obter respostas ótimas apenas para problemas de pequeno porte. Esses autores também implementam procedimentos de busca local que incorporam características específicas do problema junto às meta-heurísticas. Gonçalves et al. (2016) realizam experimentos preliminares manualmente para determinar o ajuste dos parâmetros das meta-heurísticas com instâncias que variam entre 10, 15, 20, 25,

30, 40, 50, 75, 100, 250 e 500 tarefas, existindo 1250 instâncias geradas aleatoriamente para cada tamanho.

Kır e Yazgan (2016) estudam um problema de sequenciamento com tempos de preparação dependentes da sequencia e penalidades para adiantamento e atraso de tarefas que tem objetivo de minimizar o custo de penalidade total. Esses autores desenvolvem uma Busca Tabu (BT) para construção de soluções iniciais que são então melhoradas a partir de um algoritmo genético.

Para analisar a influência dos parâmetros da busca tabu e do algoritmo genético, Kır e Yazgan (2016) propõem um experimento com 40 instâncias de 20 tarefas. Enquanto fixam os valores dos parâmetros do GA, o máximo de iterações (10, 20 ou 50) e o tamanho da lista (7 ou 10) do algoritmo de busca tabu são variados em um experimento fatorial completo. Da mesma forma, fixos os parâmetros da busca tabu os valores para total de iterações (100 ou 300), probabilidade de cruzamento (0,89 ou 0,99) e probabilidade de mutação (0,01 ou 0,05) que influenciam o desempenho do algoritmo genético são analisados a partir de uma analise da variância (ANOVA – *Analysis Of Variance*). Os melhores parâmetros encontrados seguindo as análises de Kır e Yazgan (2016) são definidos para operar do algoritmo enquanto ele resolve uma instância real de uma indústria alimentícia na Turquia.

Li et al. (2016) desenvolvem um CACO (*Combinational Ant Colony Optimization*) para minimizar o *makespan* em um ambiente de manufatura celular que envolve múltiplas células de máquina única. Para testar o desempenho do algoritmo Li et al. (2016) geram 12 instâncias aleatoriamente. De acordo com o projeto desses autores, o CACO foi testado para as 144 combinações projetadas para os parâmetros e a variância dos resultados foi analisada com o método ANOVA.

M'Hallah e Alhajraf (2016) também aplicam uma estratégia baseada em ACO, utilizando VNS para o melhoramento de respostas. Esses autores geram instâncias aleatoriamente para seus experimentos, contendo 25 níveis diferentes para a geração

de dados, foram criadas 5 instâncias para cada nível com tamanhos variando entre 40, 50, 100, 150, 200, 250 e 300 tarefas. M'Hallah e Alhajraf (2016) descrevem um experimento manual para avaliação dos parâmetros do algoritmo evolutivo implementado, mas apresentam dados desse experimento apenas para duas instâncias, uma com 40 e outra com 50 tarefas.

Ramacher e Mönch (2016) implementam um algoritmo com múltiplos critérios de otimização, NSGA-II, que é combinado com estratégias para exploração de vizinhança. Utilizando 640 instâncias para acessar o desempenho do algoritmo implementado, Ramacher e Mönch (2016) afirmam ter conduzido experimentos preliminares que levaram aos valores de parâmetros apresentados, mas não descrevem tais experimentos.

2.2 AEs Desenvolvidos para PSA

Esta seção aborda os trabalhos que tratam problemas para PSA. Foram pesquisados artigos através do portal de busca *www.engineeringvillage.com*, com o termo *advanced planning and scheduling*. Os trabalhos resultantes foram filtrados encontrando-se 19 artigos com implementações de AEs.

Tratando um modelo que permite a terceirização parcial de tarefas, Lee et al. (2002) apresentam um modelo integrado para um sistema PSA responsável pelas decisões de seleção do equipamento para cada operação, sequenciamento das operações para cada ordem e terceirização. A figura 2.3 mostra como os autores integram as funções presentes em seu sistema.

Em seu modelo, Lee et al. (2002) utilizam a terceirização como forma de garantir os prazos de entrega e tem como objetivo minimizar o tempo de término de cada uma das ordens de produção. Esses autores desenvolvem um AG para a solução do problema e apresentam uma representação dos dados da resposta considerando a seleção de equipamentos e também a terceirização. Os experimentos realizados nesse trabalho

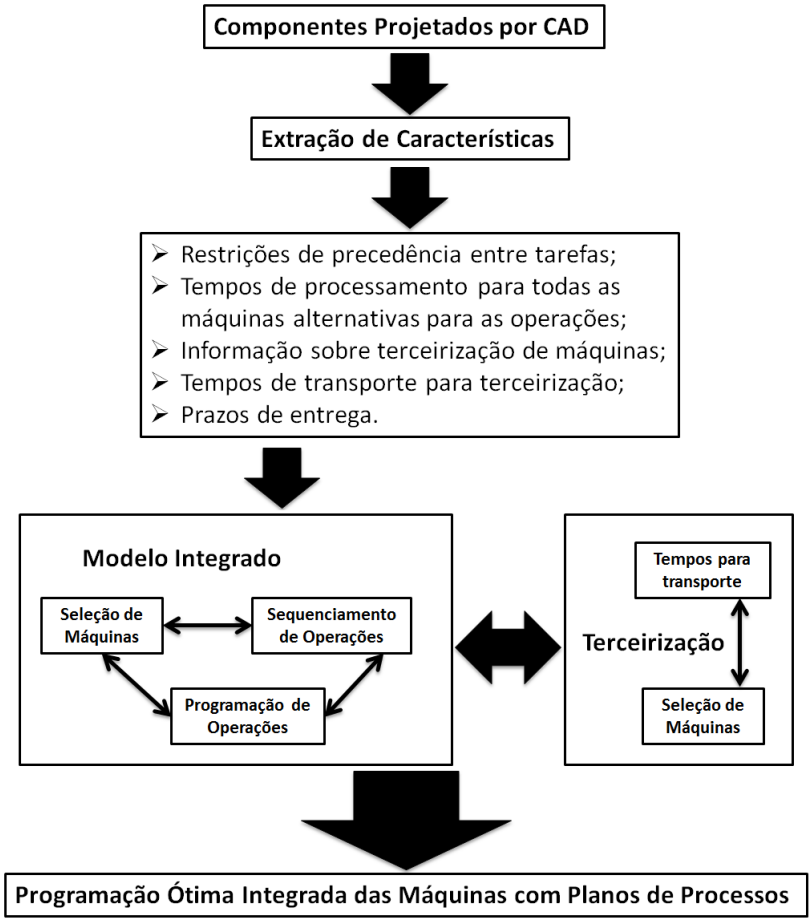


Figura 2.3: Exemplo de Arquitetura para Sistema PSA. Fonte: adaptado de Lee et al. (2002)

consideram um intervalo de valores possíveis para cada um dos parâmetros do algoritmo genético e resultados com diferentes configurações são descritos com a indicação dos melhores valores para os parâmetros. No entanto, os experimentos realizados utilizam apenas cinco instâncias.

Moon et al. (2004) apresentam um modelo para a minimização do tempo total de transição para as ordens de produção sobre um sistema produtivo que opera com diversas fábricas contendo cada uma um conjunto específico de recursos e processos. O modelo PSA desenvolvido pelos autores destina-se ao problema de programação flexível das operações, envolvendo a seleção de recursos e a programação da produção. Nesse ambiente de manufatura, os recursos possuem capacidades finitas e diferentes habilidades, requerendo ainda tempos de processamentos distintos para uma operação específica. Cada ordem inclui uma sequencia de operações com restrições de precedência e a maioria difere entre si quanto a essa sequência, uma ondem pode inclusive apresentar múltiplos caminhos possíveis para sua execução.

Segundo Moon et al. (2004), como o problema sendo tratado é NP-Difícil, não é esperado que um método convencional solucione esse problema de maneira eficiente, assim os autores optam por implementar um AG. A implementação do algoritmo conta com apenas quatro parâmetros configuráveis: tamanho da população, número de gerações, probabilidade de cruzamento e mutação. O processo de configuração desses parâmetros apresentado no trabalho é puramente manual e consistiu do teste de todas as configurações possíveis para os parâmetros sobre um conjunto com problemas de teste. No entanto, os experimentos são conduzidos com apenas quatro instâncias e o melhor resultado é apresentado impossibilitando uma análise mais detalhada dos processo de configuração.

Para Moon e Seo (2005), existe o interesse das empresas em otimizar integralmente seus sistemas surgindo assim o conceito da cadeia de suprimentos. Dessa forma, o processo de planejamento antes fragmentado tende agora a ser tradado de forma integrada

considerando a cadeia de suprimentos em sua totalidade. Esses autores desenvolvem um sistema integral capaz de solucionar as etapas de planejamento e sequenciamento, considerando a seleção de equipamentos para o processamento das operações e o sequenciamento observando que há relações de precedência entre elas, mas organizando operações em paralelo se possível.

Moon e Seo (2005) apresentam um modelo matemático para o problema proposto e afirmam que aumentando o número de ordens e operações existentes, o modelo analítico se torna intratável, sendo necessária uma solução capaz de fornecer respostas rápidas para problemas de tamanho real. Esses autores implementam um AG para resolver o problema apresentado, definindo uma representação de cromossomo para a resposta do problema determinando o método para a criação da solução factível inicial e os operadores aplicados. Moon e Seo (2005) avaliam a influência de diferentes valores para os parâmetros do algoritmo desenvolvido, no entanto, isso é feito testando apenas cinco instâncias distintas.

Altiparmak et al. (2006) desenvolvem um algoritmo genético para solucionar um modelo matemático que trata o processo da tomada de decisão, avaliando múltiplos objetivos, em uma indústria de produtos plásticos na Turquia. O modelo busca minimizar o custo total, maximizar o nível de serviço ao cliente (em termos de entrega em tempo) e ainda maximizar a utilização dos recursos, onde a função objetivo do AG resulta da ponderações dos objetivos singulares e conflitantes.

No trabalho de Altiparmak et al. (2006), os autores testam o algoritmo implementado com dados reais da indústria analisada, avaliando em um primeiro momento diferentes estratégias para a ponderação das funções objetivo e em um segundo estágio de testes comparando as soluções do AG implementado com um algoritmo baseado em SA (*Simulated Annealing*). Os autores apresentam valores para os parâmetros utilizados para o AG, mas não detalham como foram obtidos, além disso, os testes são conduzidos contam com apenas sete diferentes instâncias.

Moon et al. (2006) apresentam um modelo para a minimização do tempo total de processamento que considera a seleção de equipamentos, ou seja, prevê a existência máquinas alternativas para a realização das operações. O ambiente de manufatura tratado é composto de múltiplas fábricas e múltiplos fluxos para os produtos dentro do sistema (*job shop*). Esses autores desenvolvem um algoritmo genético adaptativo (AGA – *Adaptative Genetic Algorithm*) que tem a capacidade de regular automaticamente os operadores de cruzamento e mutação do algoritmo, podendo também aplicar busca local durante o processo de otimização.

Preocupados com o balanço da exploração do espaço de busca pelo seu algoritmo, Moon et al. (2006) criam um mecanismo de decisão para seu AGA que tem a função de avaliar o comportamento de convergência do algoritmo e então atuar sobre a população de cromossomos armazenada. Moon et al. (2006) utilizam os mesmos quatro problemas de teste apresentados em Moon et al. (2004), com exceção dos parâmetros adaptativos, melhores valores para os demais são apresentados, mas não foram descritos os experimentos para sua obtenção.

Zhang e Gen (2006) também propõem uma variação da implementação tradicional do AG para otimização em um sistema PSA, o moGA (*multistage operation-based Genetic Algorithm*), que considera uma escala codificada de prioridades para o sequenciamento das operações e também a codificação das permutações de equipamentos para a seleção das máquinas para as operações. O problema tratado por Zhang e Gen (2006) é similar ao de Moon et al. (2006), mas nesse caso a sequência das operações para cada um dos possíveis produtos não é fixa, possibilitando maior flexibilidade ao sistema de manufatura.

Os autores de Zhang e Gen (2006) testam seu algoritmo com as mesmas instâncias utilizadas em Moon et al. (2004), mas obtendo soluções melhores que os autores anteriores. Zhang e Gen (2006) definem com experimentos manuais os valores dos parâmetros para a operação de seu algoritmo, analisando o impacto de diferentes valores para o

número máximo de gerações e o tamanho da população do algoritmo, mostrando que valores maiores para esses parâmetros tendem a gerar melhores soluções.

Yan et al. (2007) implementam um algoritmo genético híbrido com procedimentos de busca local para solucionar um problema de sequenciamento das operações de uma ordem, considerando relações de precedência. Os autores detalham a formulação dos cromossomos e operadores genéticos propondo um experimento com quatro instâncias de teste e apenas três diferentes configurações distintas para os parâmetros do AG.

Chen e Ji (2007a) trabalham em um problema que considera um ambiente dinâmico onde novas ordens surgem continuamente e o sistema PSA deve ser capaz de minimizar tempos ociosos e penalidades de atrasos tanto para as ordens originais quanto para novas à cada ponto de reprogramação da produção. Quando considerando casos dinâmicos como esse, criar um novo plano à chegada de cada nova ordem nem sempre é satisfatório. Alterar a programação de ordens já agendadas pode elevar os níveis de estoque em processo e ainda sobrecarregar a função de planejamento.

Para tal, os autores de Chen e Ji (2007a) adotam uma política com janelas de tempo "congeladas" e implementam um AG para solucionar o problema que é NP-Difícil. Chen e Ji (2007a) discutem a representação do cromossomo para a resposta desse problema e também os operadores genéticos implementados, mostrando que parâmetros como o número máximo de gerações, tamanho da população entre outros representam valores de entrada para o funcionamento do algoritmo. No entanto, os autores apenas apresentam valores para os parâmetros, não estabelecendo como esses foram obtidos e mostram o resultado do algoritmo em apenas uma instância, comparando o AG com a solução analítica de Chen e Ji (2007b).

Gao et al. (2008) implementam um AG em conjunto com procedimentos de busca local como estratégia de otimização para um sistema PSA que lida com o ambiente de manufatura *job shop* flexível. Esse autores realizam testes sobre diferentes *benchmarks* totalizando 181 instâncias distintas. Apesar de encontrarem soluções melhores

que as anteriores para 38 instâncias, não é feita uma análise sobre o comportamento dos parâmetros que regulam o funcionamento do algoritmo, Gao et al. (2008) apenas mencionam os valores para esse parâmetros indicando que o tamanho da população, dependendo da complexidade do problema sendo resolvido, foi definido como 300 ou 3000.

Dayou et al. (2009) também desenvolvem um AG híbrido com procedimentos de busca local para um modelo PSA. Esses autores avaliam diferentes combinações de três objetivos distintos, a minimização do *makespan*, dos tempos de transferência entre as diferentes máquinas e o balanceamento da carga de trabalho. Dayou et al. (2009) comparam seu algoritmo genético multi-objetivo (MOGA – *Multi-Objetive Genetic Algorithm*) com outros dois similares já existentes na literatura, testando os algoritmos sobre 10 instâncias geradas aleatoriamente, apenas informando os melhores valores para a operação dos algoritmos evolucionários implementados.

Yang e Tang (2009) se referem simultaneamente aos objetivos da minimização do tempo ocioso nos equipamentos e de penalidades para atrasos e adiantamentos. Esses autores apresentam um modelo MIP para o problema proposto e desenvolvem um AG que minimiza uma ponderação desses objetivos. Yang e Tang (2009) apresentam um exemplo ilustrativo para o funcionamento do algoritmo e testa sua eficiência utilizando apenas três instâncias de diferentes tamanhos, variando os parâmetros para o número máximo de gerações e o tamanho da população do AG.

Chen et al. (2010) desenvolvem um algoritmo genético para otimização em um sistema PSA para uma indústria de produtos para iluminação como, aquecedores infra-vermelho, lâmpadas ultravioleta e lasers, que conta com mais de 3000 produtos finais e opera em um sistema MTO. Os autores se preocupam em detalhar o método construtivo usado pelo algoritmo para gerar novas soluções e apresentam exemplos para o sequenciamento de diferentes itens.

O ambiente produtivo tratado por Chen et al. (2010) consiste de um *job shop* onde

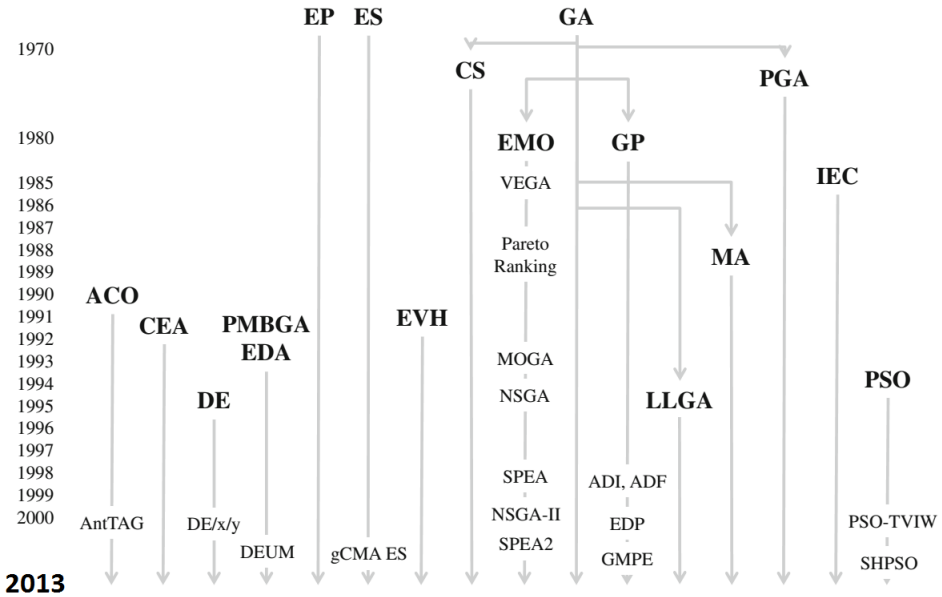
cada produto possui uma estrutura diferente e um nível de prioridade. Nesse trabalho, os autores definem como objetivos da empresa o sequenciamento da produção no chão de fábrica maximizando a utilização com penalidades para entregas adiantadas ou atrasadas (*earliness* e *tardiness*).

Para os testes do algoritmo implementado por Chen et al. (2010), os autores utilizam dados históricos para comparar os resultados que poderiam ser atingidos com a utilização do AG implementado e também os métodos SPT (*Shortest Processing Time*) e EDD. Chen et al. (2010) evidenciam que os resultados obtidos com o algoritmo genético são muito superiores ao sequenciamento realizado pela empresa. Para a configuração do algoritmo utilizado, os valores dos parâmetros são apresentados, mas nenhum processo de configuração é descrito. O conjunto de testes é pequeno, os autores descrevem 35 ordens, sugerindo que apenas uma instância com 35 tarefas foi avaliada.

O trabalho de Gen e Lin (2013) apresenta um *survey* que trata da aplicação de AEs multi objetivo para problemas de sequenciamento em sistemas de manufatura. Esses autores fazem uma classificação hierárquica e temporal dos algoritmos aplicados, a figura 2.4 mostra essa classificação.

Gen e Lin (2013) descrevem as representações das soluções de diversos problemas e algoritmos genéticos, realizando experimentos sobre instâncias de tamanhos variados. No entanto, apesar dos autores deixarem claro que a parametrização desse tipo de algoritmo é essencial ao seu funcionamento, não existem detalhes sobre o processo de configuração utilizado e os autores apenas informam os valores utilizados para os testes. Foram tratadas no trabalho de Gen e Lin (2013) instâncias provenientes de diferentes problemas e utilizando algoritmos variados, mas para um problema específico o número máximo de instâncias testadas foi 20.

Os autores em Bettwy et al. (2014) aplicam um AG para minimização do *makespan* em um sistema PSA, contudo, não acessam o desempenho do algoritmo. As características de fluxo do sistema abordado por Bettwy et al. (2014) são bem definidas, assim



ACO	Ant Colony Optimization	**GA**	Genetic Algorithm
CEA	Coevolutionary Algorithm	**GP**	Genetic Programming
CS	Classifier System	**IEC**	Interactive Evolutionary Computation
DE	Differential Evolution	**LLGA**	Linkage Learning GA
EDA	Estimation of Distribution Algorithms	**MA**	Memetic Algorithm
EMO	Evolutionary Multiobjective Optimization	**PGA**	Parallel GA
EP	Evolutionary Programming	**PMBGA**	Probabilisti Model Building GA
ES	Evolution Strategies	**PSO**	Perticle Swarm Optimization
EVH	Evolvable Hardware		

Figura 2.4: Algoritmos evolucionários aplicados em sistemas PSA. São diversas as aplicações de algoritmos bio-inspirados para otimização em Engenharia de Produção. Fonte: adaptado de Gen e Lin (2013)

como as sequências de montagem e processo de planejamento e sequenciamento são explorados. No entanto, os parâmetros configuráveis do AG implementado não foram descritos pelos autores e não existe análise da experimentação em instâncias.

Jin et al. (2015) implementam um algoritmo chamado *Honey Bee Mating Optimization* (HBMO) em conjunto com métodos VNS para planejamento e sequenciamento. Para avaliar seu algoritmo, Jin et al. (2015) realizam experimentos com um total de 32 instâncias, comparam resultados com o otimizador CPLEX. No entanto, esses autores não descrevem e avaliam os parâmetros configuráveis do algoritmo apresentado.

Para um sistema PSA, Yu et al. (2015) implementam um algoritmo com características híbridas entre AG e PSO, demonstrando seus resultados em apenas uma instância, mas realizando a analise de um cenário reativo a mudança da falha de uma das máquinas.

Chen et al. (2016) comparam os algoritmos SA e busca tabu para um problema de planejamento e sequenciamento. Os autores apontam um experimento manualmente projetado para obtenção dos melhores valores para os parâmetros do algoritmo estocástico (SA), mas não descrevem esse experimento em seu trabalho. Para avaliação dos algoritmos, Chen et al. (2016) utilizaram apenas duas instâncias considerando alta e baixa demanda.

Jin et al. (2016) tratam planejamento e sequenciamento avançados a partir de um AEMO (Algoritmo Evolutivo Multi Objetivo), mencionando os valores aplicados aos parâmetros do algoritmo, contudo, sem detalhar como foram obtidos. Jin et al. (2016) utilizam apenas 24 instâncias para avaliar o desempenho do NSGA-II implementado. Liu et al. (2016) desenvolveram um ACO comparando seus resultados com heurísticas construtivas para 10 instâncias de teste. Esses autores não mencionam os parâmetros ajustáveis do ACO implementado.

2.3 Análise da Revisão de Literatura

A revisão de literatura sobre sequenciamento em ambiente de máquina única considerou trabalhos recentes, referentes aos anos de 2015 e 2016, relacionados a otimização de problemas em geral NP-Difíceis, mas também modelos de otimização para planejamento e sequenciamento avançados sem restrição para a data de publicação na pesquisa sobre PSA. Na literatura apresentada fica evidente a tendência em resolver problemas combinatórios complexos obtendo soluções precisas mas também a necessidade de executar tal tarefa com alta velocidade.

Dos artigos sobre sequenciamento, 28 trabalhos apresentaram algoritmos exatos (p.ex. Davari et al. (2015), Wang et al. (2015a), Ji e Li (2015), Yuan et al. (2015), Zhao e Tang (2015), Han et al. (2015), Keshavarz et al. (2015), Pereira (2016), Ji et al. (2015), Fang e Lu (2016), Boland et al. (2016), Herr e Goel (2016), Yin et al. (2016a), Zhao (2016), Laalaoui e M'Hallah (2016) Gu et al. (2016), Lazarev et al. (2016), Li e Lu (2016), Herr e Goel (2016), Lu e Zhang (2016), Ou et al. (2016), Shioura et al. (2016), Wu e Cheng (2016), Wu et al. (2016), Xingong e Yong (2016), Yin et al. (2016b), Zhao (2016) e Zhao et al. (2016))

Apenas nove artigos encontrados trataram o desenvolvimento de algoritmos evolutivos. A Tabela 2.1, a seguir, resume os trabalhos encontrados com aplicações de AEs para solucionar problemas de sequenciamento em ambiente de máquina única.

Tabela 2.1: Trabalhos de Sequenciamento que Implementaram AEs

Trabalho	**Algoritmo**	**Instâncias**	**Configuração do Alg.**
Ahmadizar e Farhadi (2015)	ICA	100	Descrita
Kim e Choi (2015)	AG	720	Não Descrita
Moghaddam et al. (2015)	8 Meta-Heurísticas	66	Não Descrita
Wu et al. (2015)	3 AGs	3600	Não Descrita
Gonçalves et al. (2016)	3 Meta-Heurísticas	13750	Não Descrita
Kır e Yazgan (2016)	AG	40	Descrita
Li et al. (2016)	CACO	12	Descrita
M'Hallah e Alhajraf (2016)	ACO	525	Descrita
Ramacher e Mönch (2016)	NSGA-II	640	Não Descrita

A presente revisão de literatura observou cinco pontos principais:

1. Complexidade dos problemas otimizados;
2. Estratégias de otimização aplicadas;
3. Consideração de problemas heterogêneos;
4. Avaliação em sequências de instâncias;
5. Geração de múltiplos cenários.

A grande maioria dos autores de programação da produção evidencia características NP-Difíceis para os problemas tratados em seus trabalhos (item 1). A aplicação de métodos exatos é mais comum aos autores de *scheduling*. No entanto, fica claro que tais algoritmos apresentam dificuldades em obter bons resultados para instâncias de médio e grande porte (mais de 50 tarefas), enquanto uma boa parcela dos autores vale-se de estratégias evolutivas baseadas em algoritmos genéticos e diferentes meta-heurísticas (item 2).

Apesar de alguns trabalhos terem detalhado os experimentos de configuração dos AEs implementados e ser possível observar a existência de instâncias heterogêneas, ou seja, blocos de problemas resolvidos com parâmetros distintos, nenhum dos autores trata claramente a existência de problemas heterogêneos (item 3). Poucos autores exploraram grandes conjuntos de instâncias, apenas Wu et al. (2015) e Gonçalves et al. (2016), sugerindo assim, a avaliação de seus algoritmos por um horizonte de tempo significativo (item 4). Além disso, dentre os autores que desenvolvem AEs, nenhum deles explora sua estrutura para geração de múltiplos cenários (item 5).

Compondo os trabalhos sobre PSA que introduzem o capítulo, 13 deles foram classificados como análises sobre sistemas PSA – estudos de caso e *surveys* em maioria, como os trabalhos de Errington (1997), Shobrys e White (2000), van Eck (2003), Fleischmann et al. (2004), Gruat-La-Forme et al. (2005), David et al. (2006), Liu e Qi (2008), Malik e

Qiu (2008), Hadaya e Pellerin (2008), Giacon e Mesquita (2011), Nanvala (2011), Ivert (2012) e Policella et al. (2014).

Oito trabalhos foram classificados como aqueles que apresentam ferramentas PSA (p.ex. Chen e Chen (2002), Musselman et al. (2002), Voudourist et al. (2006), Lin et al. (2007), Dorne et al. (2008), Rudberg e Thulin (2009), Hvolby e Steger-Jensen (2010) e Santa-Eulalia et al. (2011)). Esses autores tratam sistemas customizados de sua própria autoria, mas também *suite applications* como Oracle e SAP, em geral descrevendo componentes desses sistemas, características de implementação e também organizacionais relativas a sua implantação e utilização.

Da literatura pesquisada sobre PSA, 27 trabalhos desenvolveram algoritmos para otimização. Quatro deles apresentaram modelos MIP (e.g. Chen e Ji (2007b), Ornek et al. (2010), Steger-Jensen et al. (2011) e Ozturk e Ornek (2014)). Todos estes autores, definiram essa estratégia como muito custosa e intratável para problemas de larga escala. No entanto, a aplicaram por dois principais motivos: validação de formulações matemáticas e comparação com outras estratégias não exatas. Quatro trabalhos apresentaram soluções construtivas baseadas em regras de prioridades para ordens de produção e ordenação como SPT e EDD (e.g. Chua et al. (2006), Kung e Chern (2009), Chen et al. (2013)) e Montesco et al. (2015).

Os demais 19 trabalhos desenvolveram AEs diversos para otimização em sistemas PSA (p.ex. Lee et al. (2002), Moon et al. (2004), Moon e Seo (2005), Moon et al. (2006), Zhang e Gen (2006), Altiparmak et al. (2006), Yan et al. (2007), Chen e Ji (2007a), Gao et al. (2008), Yang e Tang (2009), Dayou et al. (2009), Chen et al. (2010), Gen e Lin (2013), Bettwy et al. (2014), Jin et al. (2015), Yu et al. (2015), Chen et al. (2016), Jin et al. (2016) e Liu et al. (2016)).

Outros autores, como Shen (2002), Kuroda et al. (2002), Kortenkamp (2003), Ning et al. (2008) e Li e Ma (2010) e Zhong et al. (2013), também abordaram sistemas PSA, no entanto, focando em outros aspectos que não a otimização de problemas combinatórios

complexos.

A grande maioria dos trabalhos que desenvolvem algoritmos para otimização em sistemas PSA realiza experimentos computacionais em um número muito pequeno de instâncias, relacionados na tabela 2.2.

Tabela 2.2: Trabalhos de PSA que Implementaram AEs

Trabalho	**Algoritmo**	**Instâncias**	**Configuração do Alg.**
Lee et al. (2002)	AG	5	Descrita
Moon et al. (2004)	AG	4	Descrita
Moon e Seo (2005)	AG	5	Descrita
Altiparmak et al. (2006)	AG	7	Não Descrita
Moon et al. (2006)	AGA	4	Não Descrita
Zhang e Gen (2006)	moGA	4	Descrita
Yan et al. (2007)	AG	4	Descrita
Chen e Ji (2007a)	AG	1	Não Descrita
Gao et al. (2008)	AG	181	Não Descrita
Dayou et al. (2009)	MOGA	10	Não Descrita
Yang e Tang (2009)	AG	3	Descrita
Chen et al. (2010)	AG	1	Não Descrita
Gen e Lin (2013)	AG	20	Não Descrita
Bettwy et al. (2014)	AG	0	Não Descrita
Jin et al. (2015)	HBMO	32	Não Descrita
Yu et al. (2015)	AG/PSO	1	Não Descrita
Chen et al. (2016)	SA	2	Não Descrita
Jin et al. (2016)	NSGA-II	24	Não Descrita
Liu et al. (2016)	ACO	10	Não Descrita

Porém, como será visto no próximo capítulo, Wolpert e Macready (1997) apresentam provas matemáticas do risco existente ao testar apenas um pequeno conjunto de instâncias ao comparar o desempenho de algoritmos evolutivos. Mesmo no trabalho de Gao et al. (2008), que avaliam 181 instâncias, não existe uma descrição das variáveis aleatórias que originam tais instâncias que foram compostas a partir da junção de *benchmarks* distintos. Além disso, no trabalho de Bettwy et al. (2014) não existem resultados para a execução do algoritmo proposto.

Com exceção de Kuroda et al. (2002), que menciona em seu texto que sistemas de

otimização quando aplicados para a solução de problemas em ambientes produtivos visam solucionar uma sequência temporal de instâncias relativas a um período de funcionamento da produção, nenhum dos demais autores da literatura sobre PSA menciona tal conceito (item 4).

Dentre os autores que desenvolveram AEs para PSA, em alguns trabalhos é possível observar que diferentes parâmetros são fornecidos aos algoritmos para resolverem conjuntos de teste distintos, ou seja, parametrizações diferentes são aplicadas para as instâncias tratadas nesses trabalhos. No entanto, nenhum desses autores trata explicitamente a existência de instâncias heterogêneas (item 3).

Sobre a complexidade dos ambientes produtivos tratados (item 1), a grande maioria dos trabalhos exploram sistemas definidos como *job shop* e NP-Difíceis. Ivert (2012) observa que, apesar dos esforços no desenvolvimento de sistemas APS estarem principalmente voltados para problemas caracterizados pelo ambiente de manufatura *job shop*, ambientes *flow shop* também apresentam problemas de otimização complexos no cotidiano de suas operações.

Generalizando essa observação de Ivert (2012), outros ambientes, como máquinas paralelas ou mesmo máquina única, também apresentam problemas combinatórios NP-Difíceis, como visto neste mesmo capítulo. Não apenas para ambientes *flow shop* ou *job shop* existem problemas de decisão cujas formulações são NP-Difíceis, mas também muitos problemas em ambiente de máquina única e com máquinas paralelas, sobretudo os problemas que tratam relações de precedência (*setup*) entre as tarefas a serem programadas.

Analisando as estratégias de otimização aplicadas (item 2), constata-se que a maioria dos autores desenvolvem algoritmos genéticos buscando soluções rápidas para problemas complexos. Apesar da literatura que analisa casos da aplicação de sistemas PSA e também de alguns dos fornecedores mais tradicionais (Oracle e SAP), apontarem que uma das vantagens de sistemas PSA é a possibilidade de avaliar múltiplos cenários

(item 5), nenhum dos autores que desenvolveram AEs exploraram seus métodos para demonstrar tal vantagem.

De acordo com a revisão construída, encontramos evidências na literatura comprovando que grande parte das implementações que buscam velocidade para a otimização de formulações para programação da produção aplicam algoritmos evolutivos. Porém, questões da configuração desses algoritmos são tratadas de maneira superficial e a grande maioria dos autores realiza testes extremamente simplificados sobre o desempenho de tais algoritmos com natureza estocástica.

Foram observadas aplicações de AGs, e suas variações, mas também diversas metaheurísticas, dentre elas o ACO fez-se presente. Como observado por Birattari (2009), Stützle et al. (2010) e Eiben e Smit (2011a), a parametrização é vital para o desempenho de todos esses algoritmos configuráveis.

Percebe-se que o foco principal da linha de pesquisa que envolve o *scheduling* em ambiente de máquina única é o desenvolvimento de algoritmos eficientes para problemas específicos. Enquanto para a literatura referente aos sistemas PSA, os trabalhos se dividem entre o desenvolvimento de algoritmos e a discussão das melhores práticas e barreiras para a implantação e aplicação desses sistemas.

Destacam-se os apontamentos dos autores em relação à degeneração dos dados. As soluções otimizadas tornam-se rapidamente obsoletas uma vez que as características do chão de fábrica se alteram com o tempo. Portanto, replanejamento é constantemente necessário. Assim, uma solução de otimização deve ser suficientemente veloz para evitar a degeneração dos dados antes da conclusão do processo de otimização.

Apesar de parte dos trabalhos descrever experimentos para configuração manual dos parâmetros, algumas vezes aplicando análises estatísticas sofisticadas, como ANOVA, para determinar a influência da variação dos parâmetros em relação ao desempenho do algoritmo configurável, muitos autores realizaram experimentos sobre conjuntos pequenos de instâncias, muitas vezes com domínios para os parâmetros ajustáveis

absurdamente limitados, dificultando o acesso aos efeitos desses parâmetros para o desempenho dos algoritmos estocásticos.

É importante observar que a utilização de testes estatísticos paramétricos como ANOVA, pressupõe a normalidade dos dados. No entanto, os autores que aplicam este teste não apresentaram comprovação de que os dados utilizados obedecem uma distribuição específica.

Não obstante, muitos autores apenas mencionam os parâmetros aplicados em seus algoritmos sem detalhar os experimentos conduzidos para tal. A falta de preocupação quanto a configuração minuciosa dos algoritmos evolutivos desenvolvidos foi observada em muitos trabalhos e nenhum dos autores considerou uma estratégia para a configuração automática de AEs. O próximo capítulo apresenta uma revisão de literatura sobre a parametrização de algoritmos evolutivos, buscando esclarecer quais os fatores que determinam o alto desempenho a um AE e como configurar AEs automaticamente.

Capítulo 3

Configuração de Algoritmos Evolutivos

O capítulo anterior apresentou diversos exemplos de problemas combinatórios envolvendo funções objetivo com único alvo para otimização ou mesmo múltiplos critérios de desempenho que devem ser minimizados ou maximizados de acordo com as características de cada problema proposto.

Muitos dos trabalhos analisados se devotam a pesquisa de soluções baseadas em algoritmos evolutivos para obter respostas rápidas em problemas combinatórios complexos. Porém, a aplicação desses algoritmos acarreta a necessidade de solucionar um novo problema combinatório que representa a obtenção dos valores para os parâmetros que proporcionarão o melhor resultado de operação para esse algoritmo. No entanto, a configuração desses algoritmos é pouco explorada na literatura relativa a aplicação de AEs para as áreas pesquisadas na Engenharia de Produção.

Ainda que a literatura esteja repleta de exemplos com algoritmos para parametrização de sistemas nas mais variadas áreas, foi visto que trabalhos aplicando AEs para otimização em sistemas PSA, ou problemas de *scheduling*, muitas vezes suprimem a etapa de projeto para seus algoritmos ou valem-se de soluções manuais para

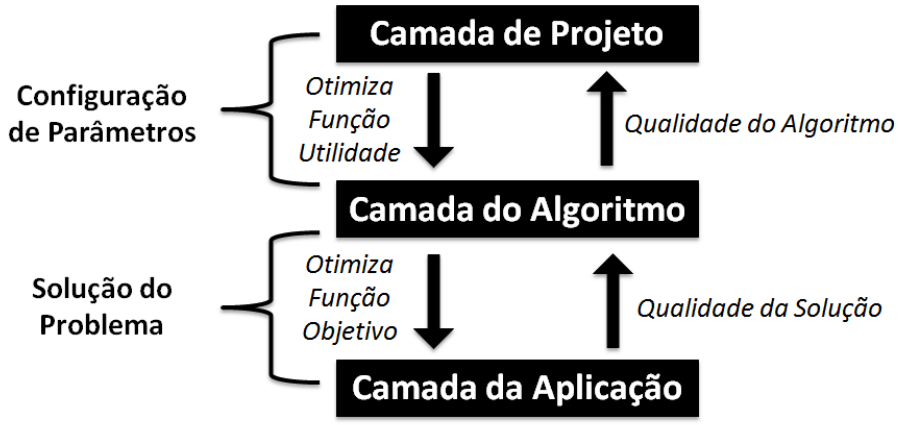


Figura 3.1: Camadas em aplicações de AEs para solução de problemas. Fonte: adaptado de Smit e Eiben (2009)

parametrização.

Já autores da área de computação evolutiva destacam a importância do processo parametrização para a construção de bons algoritmos evolutivos (Eiben et al., 1999; Birattari et al., 2002; Balaprakash et al., 2007; Yuan e Gallagher, 2007; Eiben et al., 2007; Birattari, 2009; Eiben e Smit, 2011a).

Focando no processo para parametrização de algoritmos evolutivos, este capítulo apresenta o conteúdo relativo as práticas mais encontradas na literatura para configuração automática de algoritmos e também uma investigação sobre as necessidades para o desenvolvimento de AEs com alto desempenho.

Smit e Eiben (2009) distinguem três camadas (*layers*) para descrever aplicações de AEs, incluindo dois problemas de otimização. Na figura 3.1 observa-se o problema a ser solucionado pelo AE, definido por uma função objetivo (*fitness function*) e o problema da configuração desse algoritmo, definido como sua função de utilidade (*utility funcion*).

Dessa maneira, solucionar o problema original contido na função objetivo depende da solução, na camada anterior, da configuração do algoritmo. Eiben e Smit (2011a) argumentam que um esquema genérico para a obtenção de bons valores para os parâ-

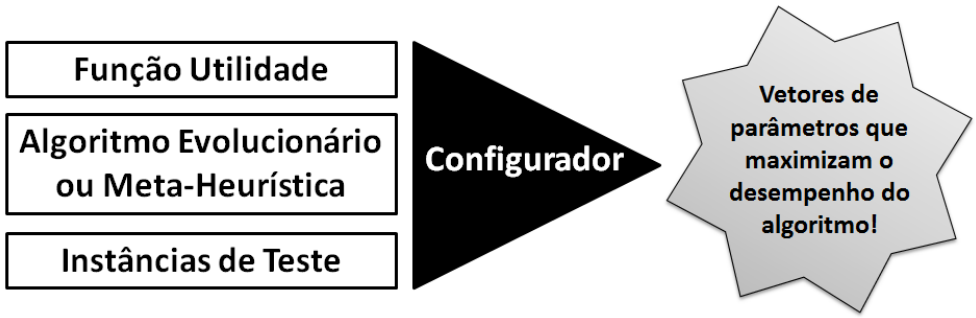


Figura 3.2: Esquema genérico para a configuração de parâmetros. Fonte: adaptado de Eiben e Smit (2011a)

metros de um AE pode ser ilustrado pela figura 3.2, onde o resultado dos valores para a operação de um algoritmo dependem do próprio AE, mas também da função utilidade e das instâncias a serem solucionadas.

Na pesquisa por soluções pelos termos *automatic tuning* e *evolutionary algorithms*, foram encontrados trabalhos contendo apenas o desenvolvimento de AEs, citando a necessidade de sua parametrização, mas não focando nesse aspecto. Por exemplo, Tan et al. (2001), Tan et al. (2003) e Kaji et al. (2008) que desenvolvem algoritmos para otimização com múltiplos objetivos baseados em AEs.

Outros autores como Xu e gen He (2005), analisam a convergência para um MEA (*Memetic Evolutionary Algorithm*). Grosan et al. (2005) desenvolvem um algoritmo baseado em PSO e o comparam com outros algoritmos evolutivos. Chen et al. (2012) avaliam as peculiaridades do tamanho da população para AEs, enquanto outros trabalhos como Joan-Arinyo et al. (2011), Bosman (2012) e Ma et al. (2014) apresentam aplicações de AEs para problemas combinatórios diversos.

Gupta e Kumar (2014) demonstram uma modificação do algoritmo QIEA (*Quantum-Inspired Evolutionary Algorithm*) e mencionam a importância de sua parametrização, mas não desenvolvem um mecanismo para tal problema, apenas relatam os melhores valores dos parâmetros para os diferentes conjuntos de instâncias testados.

Algoritmos evolutivos são bastante utilizados para a configuração de parâmetros em outros algoritmos. Muitos trabalhos descrevem a aplicação de AEs para a configuração de parâmetros ajustáveis em aplicações distintas à estudada neste projeto. Existem trabalhos que empregam algoritmos evolucionários para a configuração de parâmetros em componentes eletrônicos, controladores PID (*Proportional Integral Derivative*) em geral (Herrero et al., 2008; Hu et al., 2010; Chaparro e Sosa, 2011; Junli et al., 2011; Pan et al., 2011; Khoshrooz et al., 2012; Reynoso-Meza et al., 2012, 2013; Fadil e Darus, 2013; Saad et al., 2015; Su e Li, 2015; Moness e Moustafa, 2015; Gutiérrez-Urquídez et al., 2015), mas também em sistemas para aplicações de captura visual (Muñoz-Salinas et al., 2008; Gacto et al., 2009; Ethridge et al., 2010; Lutton et al., 2014). Enquanto o trabalho de Wang et al. (2012) apresenta uma proposta para parametrização de um algoritmo para clusterização e Onieva et al. (2015) aplicam AEs para a configuração de um sistema para veículos autônomos.

Muñoz-Salinas et al. (2008) definem um algoritmo para a configuração de um sistema para detecção visual *fuzzy*. Baseando-se em uma combinação de classificadores que permitem à um robô detectar a presença de portas nas imagens capturadas pela câmera para a solução do problema da detecção de padrões. Esses autores argumentam que o processo de configuração do algoritmo é tedioso e depende do ambiente no qual o robô é utilizado, pois a mudança das condições do ambiente leva a repetição da parametrização do sistema. Muñoz-Salinas et al. (2008) desenvolvem três AEMOs, SPEA (*Strength Pareto Evolutionary Algorithm*), SPEA2, e NSGA-II para a configuração do sistema.

Ethridge et al. (2010) aplicam um AEMO para parametrização de um ν-SVM (*Support Vector Machine*), um algoritmo baseado em aprendizado de máquina, que nesse trabalho é aplicado ao reconhecimento de imagens para o qual os autores realizam testes com bancos de dados sobre o reconhecimento e diagnóstico de doenças, como o câncer de mama. Para Ethridge et al. (2010) a parametrização do ν-SVM representa um problema com objetivos conflitantes e o AEMO é usado para balancear a sensitividade

e especificidade para cada conjunto de instâncias testado.

Jovanovic et al. (2014) apresentam um modelo para design automático de componentes e parâmetros de algoritmos configuráveis para obter árvores de decisão, como uma estrutura de meta-aprendizado. A proposta desses autores emprega uma meta-heurística para realizar a busca por uma configuração aproximadamente ótima para o algoritmo configurável dada uma instância específica do problema. Jovanovic et al. (2014) comparam seus resultados com uma alternativa que emprega força bruta e avalia todas as 960 configurações disponíveis do algoritmo avaliado.

O trabalho de Zaefferer et al. (2014) aplica o software SPO (*Sequential Parameter Optimization*) de Bartz-Beielstein et al. (2005), para configurar algoritmos que buscam otimizar a geometria de separadores ciclônicos industriais.

Os autores em Bezerra et al. (2015), definem um modelo para apoiar a o desenvolvimento de algoritmos para otimização com múltiplos critérios. Bezerra et al. (2015) discutem que AEMOs podem ser formados a partir da extensão de AEs para um objetivo único de otimização. A vantagem apontada para esse tipo de estratégia é a possibilidade de originar e avaliar novas combinações de algoritmos antes impensadas.

García-Sánchez et al. (2016) tratam o efeito do tamanho da população criada por AEs enquanto operando em recursos computacionais paralelizados porém heterogêneos. O conceito de heterogeneidade é explorado nesse trabalho de uma maneira distinta do observado aqui. García-Sánchez et al. (2016) estão focados no problema da alocação de recursos em máquinas paralelas com poder de processamento distinto para execução de AEs, avaliando que, alocar o mesmo volume computacional em recursos heterogêneos é inadequado quando deseja-se minimizar o tempo para solução de um problema com uma estratégia paralelizada.

Andersson et al. (2016) apresentam um modelo onde o algoritmo configurável pode operar com conjuntos de parâmetros distintos, porém não adaptativos, em diferentes estágios de sua execução. Jiang e Yang (2016) tratam problemas dinâmicos para otimi-

zação multi critério, discutindo o impacto de diferentes funções objetivo nos resultados dos gráficos de Pareto.

Bartz-Beielstein et al. (2014) apontam os softwares SPO, REVAC (*Relevance Estimation and Value Calibration of Evolutionary Algorithm*) e IRACE (*Iterated Race for Automatic Algorithm Configuration*) como estado da arte para os sistemas construídos especificamente para configuração automática de algoritmos. Neste projeto o algoritmo desenvolvido será comparado com os resultados obtidos através do IRACE.

A busca por estratégias automáticas para configuração de algoritmos retornou trabalhos que lidam com problemas diversos em Engenharia de Produção. Por exemplo, Munoz-Carpintero e Sáez (2015) discute a parametrização de algoritmos evolutivos para problemas de carregamento e entrega, porém afirmam que a experimentação realizada para analise dos parâmetros não foi conclusiva. Assim, o autor utilizou valores de parâmetros apresentados anteriormente na literatura para configurar seus algoritmos.

Sadeghi e Niaki (2015) abordam um problema de otimização multi critério para gerenciamento de estoques, aplicando a metodologia de Taguchi para construir o experimento que analisa os efeitos dos parâmetros presentes nos algoritmos aplicados. Aqui o método de Taguchi é utilizado para definir quais os parâmetros que mais influenciam os algoritmos, realizando um número menor de experimentos quando comparado com a quantidade necessária para um design fatorial completo aplicado à configuração dos algoritmos.

Switalski e Seredynski (2015) tratam um problema de sequenciamento em ambiente *job shop* e aplicam algoritmos evolutivos. Esses autores discutem diferentes implementações de AEs comentando a complexidade do processo de configuração para os algoritmos implementados. A meta-heurística desenvolvida por Switalski e Seredynski (2015) possui apenas um parâmetro configurável, cuja analise é detalhada pelos autores que evidenciam a simplicidade da configuração do algoritmo desenvolvido.

Wood (2016) discute a combinação de meta-heurísticas e heurísticas para a formação

de algoritmos evolutivos mais robustos para solução de problemas que envolvem otimização discreta para roteamento de recursos. Enquanto, Mishra e Rao (2016) abordam um problema de sequenciamento em ambiente *job shop* e comparam o desempenho de diferentes algoritmos evolutivos sobre 250 instâncias de teste.

Neste ponto torna-se muito importante o trabalho de Rabanal et al. (2015). Esses autores argumentam contra a avaliação de algoritmos estocásticos em conjuntos de instâncias limitados. Segundo esses autores, para *benchmarks* finitos já conhecidos é possível encontrar facilmente configurações para o algoritmo que permitem alta eficiência sobre as instâncias avaliadas., não sendo demonstrada garantia de que para casos desconhecidos tal algoritmo operará satisfatoriamente.

A próxima seção descreve o processo para parametrização de algoritmos configuráveis e define as características desejadas para o desempenho desses algoritmos otimizadores na camada de aplicação (ver figura 3.1). Serão tratadas também questões sobre a eficiência dos métodos de parametrização aplicados na camada de projeto. Em seguida, será apresentada a definição formal do problema da parametrização de algoritmos.

3.1 Projeto de AEs com Alto Desempenho

Encontrar parâmetros apropriados para operação de algoritmos evolucionários é um dos grandes desafios persistentes na área de computação evolutiva. Praticantes e pesquisadores possuem conhecimento de que os valores desses parâmetros influenciam diretamente o desempenho dos algoritmos. No entanto, pouco esforço é destinado ao estudo dos efeitos da parametrização no desempenho de AEs e, na prática, valores são definidos baseando-se em convenções e comparações experimentais limitadas (Eiben e Smit, 2011b).

O potencial máximo de um algoritmo configurável não pode ser atingido a menos que seus parâmetros sejam minuciosamente afinados. Enquanto o processo de encontrar

bons valores para os parâmetros ajustáveis de algoritmos otimizadores é desafiador, importante e representa uma tarefa que consome muito tempo dentro do processo global para o desenvolvimento de algoritmos evolutivos (Balaprakash et al., 2007).

Nannen et al. (2008) afirma que o processo da parametrização de AEs é difícil pois normalmente existe um grande espaço de possibilidades para os parâmetros e pouco conhecimento sobre o efeitos da variação do desempenho para o AE em relação aos valores desses parâmetros.

Segundo Yuan e Gallagher (2007), um dos maiores desafios para o desenvolvimento de AEs está na etapa de projeto, ou design, onde devem ser determinados valores apropriados para os parâmetros proporcionando bom funcionamento para o algoritmo. Esses autores citam três importantes aspectos da configuração de AEs: i) esses algoritmos não são totalmente robustos à seus parâmetros e configurações inapropriadas podem impedir seu funcionamento adequado; ii) comparações de AEs com configurações arbitrárias podem gerar conclusões incorretas; iii) a otimização do problema de parametrização para AEs pode ampliar a base de conhecimento sobre o comportamento desses algoritmos.

Yuan e Gallagher (2007) discutem que, de fato, uma configuração ótima para um AE em geral não existe e depende do conjunto de instâncias tratadas. Para cada instância de um problema, a superfície que define o desempenho do algoritmo em relação aos seus parâmetros pode apresentar características muito distintas. Esses autores afirmam que para se analisar o desempenho de AEs, sistematicamente variando seus parâmetros com uma técnica inteligente de amostragem, técnicas como ANOVA podem ser aplicadas. No entanto, tais técnicas demandam um grande número de experimentos sendo ineficientes para o propósito de parametrizar um algoritmo e métodos de otimização global como os próprios AEs podem ser utilizados. Um AE aplicado na camada de projeto para outro AE é definido como "meta-AE".

De acordo com Eiben e Smit (2011a), a função objetivo é definida como aquela que

deve ser otimizada por um AE, ou qualquer outro algoritmo estocástico configurável, e a função utilidade é aquela que representa a qualidade para um "vetor de parâmetros" (vetor de dados com os valores para cada um dos parâmetro de entrada para o AE) que define a configuração do algoritmo aplicado à uma instância específica do problema. Observando que o primeiro problema, a solução da função objetivo, depende da resposta do problema de parametrização do algoritmo, ou seja, obter vetores de parâmetros que maximizem a função utilidade para as características da instância do problema sendo resolvida.

Indo além, é importante diferenciar os conceitos de algoritmo evolutivo e instância do AE, sendo o primeiro um algoritmo parcialmente especificado que segue características de implementação evolutivas, possuindo um vetor com parâmetros qualitativos que define seus aspectos principais. Adicionando a informação dos parâmetros quantitativos e obtendo a definição completa do algoritmo, define-se uma instância para tal AE.

Parâmetros qualitativos determinam por exemplo o tipo de função para cruzamento aplicada em um AG, enquanto os parâmetros quantitativos são representados em escalas numéricas ordenadas. Um AE possui seus parâmetros qualitativos definidos, uma vez que diferentes composições dos procedimentos evolucionários constroem AEs distintos. Exemplificando, dois AGs que apliquem técnicas de cruzamento distintas (*one-point-crossover* ou *two-point-crossover*), apenas uma diferença em um parâmetro categórico (tipo de cruzamento) distingue dois AEs diferentes. Já uma instância específica de um AE deve conter a especificação completa de seus parâmetros quantitativos além dos qualitativos (Eiben e Smit, 2011a).

Eiben e Smit (2011a) também distinguem o significado de problema e instância do problema: onde o primeiro refere-se a formulação de um modelo representado por variáveis, restrições e uma função objetivo; enquanto uma instância do problema especifica em detalhes os dados de um determinado cenário para essa formulação.

Segundo Eiben et al. (1999), Stützle et al. (2010) e Eiben e Smit (2011a), a maioria

dos pesquisadores definem manualmente os parâmetros para seus algoritmos genéticos. Eiben et al. (1999) citam as seguintes desvantagens da aplicação de um processo manual para o design de algoritmos configuráveis:

- Os parâmetros não são independentes e tentar avaliar todas as combinações existentes é praticamente impossível;
- O processo consome muito tempo, mesmo ignorando as interações entre os parâmetros;
- Vetores com valores de parâmetros selecionados para uma determinada instância não são necessariamente ótimos, mesmo se considerável esforço for aplicado no processo.

Birattari et al. (2002) discutem as desvantagens de métodos enumerativos, aqueles que aplicam força bruta para a solução do problema de afinação. Primeiramente esses métodos não abordam como determinar o número de instâncias a serem testadas. Avaliar poucos problemas de teste impede a construção de estimativas confiáveis, enquanto testar muitas instâncias pode ser computacionalmente proibitivo.

Para Eiben e Smit (2011a), o desempenho de um AE pode ser avaliado de diferentes maneiras:

- Dado um tempo máximo de execução (custo computacional), o desempenho do algoritmo é definido pela *melhor resposta encontrada para função objetivo ao término da execução*;
- Definindo um nível mínimo para a função objetivo, o desempenho do algoritmo é avaliado como *o tempo de execução (custo computacional) necessário para alcançar esse nível*;

- Estabelecendo um tempo máximo de execução e um nível mínimo para a função objetivo, o desempenho é representado pela *resposta binária de sucesso ou falha do algoritmo.*

Eiben e Smit (2011a) também definem o conceito de robustez para AEs, generalizando suas observações para outros algoritmos estocásticos configuráveis. Segundo esses autores, um AE pode ser considerado robusto em relação a três diferentes fatores: 1) as instâncias do problema sendo resolvidas; 2) os parâmetros utilizados para configurar o algoritmo e; 3) a semente aleatória usada para o processo estocástico.

Explicaremos primeiramente o terceiro fator, que muitas vezes é relacionado como único aspecto da robustez de um AE. Esse fator está relacionado à natureza estocástica do algoritmo e da semente aleatória utilizada para executar inúmeros passos dentro dele. Um AE pode apresentar um funcionamento instável, isto é, quando execuções sucessivas para uma mesma instância do problema e um mesmo conjunto de parâmetros produzem resultados muito diferentes. Porém, apenas essa variância observada em execuções sobre uma instância específica não é suficiente para tratar a robustez de um AE.

O primeiro fator, argumentam Eiben e Smit (2011a), está relacionado às características dos dados da instância sendo resolvida pelo algoritmo. Smit e Eiben (2010b) discutem que determinar a configuração ótima para o funcionamento de um algoritmo estocástico para uma determinada instância do problema define um "especialista" para tal instância. Ao contrário, um algoritmo configurado para obter um resultado médio satisfatório sobre um determinado grupo de instâncias é considerado um "generalista".

Wolpert e Macready (1997) realizam comprovações matemáticas baseadas no *No Free Lunch Theorem*, demonstrando que para qualquer algoritmo de otimização estocástico, uma configuração que permite alto desempenho sobre uma classe de instâncias é compensado pelo detrimento do desempenho de outra classe. Estes teoremas resultam da interpretação geométrica do que significa para um algoritmo estar bem adequado para um caso específico de otimização dadas as características geométricas do seu es-

paço de busca. Os teoremas estabelecidos por Wolpert e Macready (1997) também apontam o alto risco existente em comparar o desempenho de algoritmos a partir de um conjunto pequeno de instâncias.

Segundo Smit e Eiben (2010b), caso as instâncias submetidas ao algoritmo se assemelhem aquela utilizada para a definição do "especialista", esse deve obter soluções de alta qualidade para os novos casos. No entanto, caso as instâncias sejam heterogêneas, o "especialista" pode oferecer resultados inferiores. Enquanto, um algoritmo que é configurado para obter um desempenho médio sobre um conjunto de instâncias, o "generalista", apesar de apresentar resultados inferiores para algumas instâncias fornece uma resposta média maior sobre o conjunto todo de instâncias avaliadas.

Eiben e Smit (2011a) utilizam a figura 3.3 para ilustrar os aspectos da robustez de um AE em relação aos dois primeiros fatores observados, as instâncias do problema sendo resolvida e os valores para um parâmetro ajustável do algoritmo.

Observando um limitante hipotético para determinar a qualidade das soluções oferecidas pelo AE para a função objetivo. Ao avaliar as características de um algoritmo em relação à diferentes instâncias de um problema, o AE será altamente aplicável se uma determinada configuração de seus parâmetros proporcionar alta qualidade de soluções em uma grande porção de instâncias, na figura 3.3, este fato está relacionado a variável W. No entanto, se altura X for razoavelmente grande, tem-se que a amplitude da qualidade de soluções é alta e o algoritmo tem maiores chances de falhar para determinadas instâncias.

Constatando que a real representação da função utilidade para a parametrização deve considerar uma dimensão diferente para cada parâmetro ajustável, o segundo gráfico da figura 3.3 apenas ilustra o desempenho para a variação dos valores de um único parâmetro. Para o segundo fator apontado por Eiben e Smit (2011a), a robustez do AE em relação aos próprios parâmetros que determinam seu funcionamento está relacionada a variável Y. Se a largura de Y for grande, o algoritmo é bastante tolerante

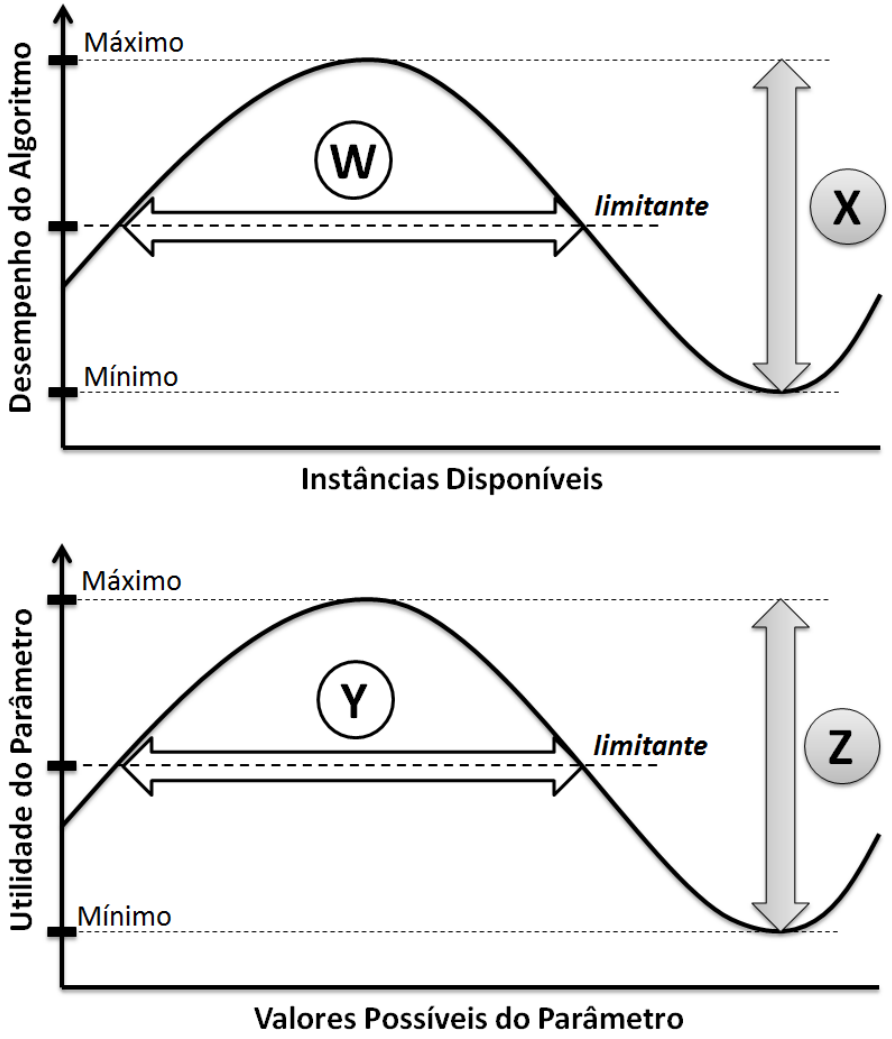


Figura 3.3: Propriedades para robustez de um AE: W – Aplicabilidade, X – Falibilidade, Y – Tolerância e Z – Configurabilidade. Fonte: adaptado de Eiben e Smit (2011a)

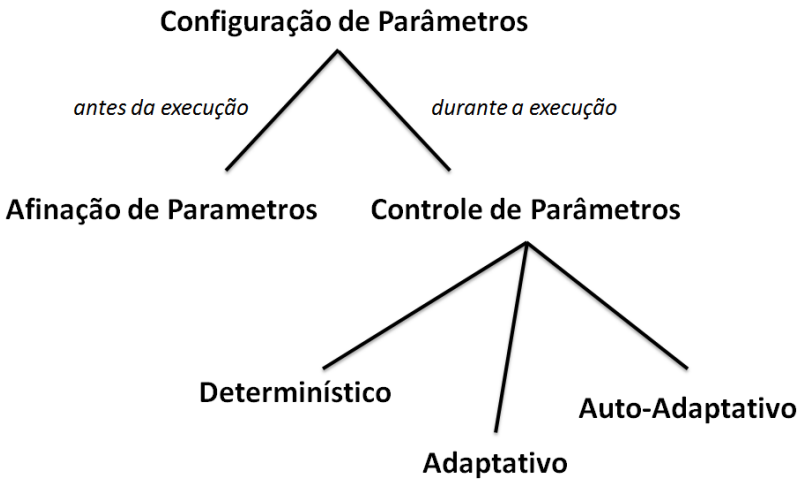


Figura 3.4: Taxonomia global para parametrização em AEs. Fonte: adaptado de Eiben et al. (1999)

em relação à variações em seus parâmetros. No entanto, se essa variação ocasionar alta amplitude na qualidade das soluções, e Z for grande, o algoritmo é altamente flexível e variações em seus parâmetros levarão a mudanças extremas em seu funcionamento, isto é, determinadas configurações oferecerão bons resultados enquanto outras obterão respostas muito ruins para a função objetivo sendo otimizada.

Eiben et al. (1999) utilizam a figura 3.4 para definir a taxonomia global para algoritmos de parametrização. Para os autores, na área da computação evolutiva existem duas metodologias distintas para a determinação de parâmetros para AEs: a afinação de parâmetros (*parameter tuning* ou simplesmente parametrização); e o controle de parâmetros (*parameter control*). Estratégias para o controle de parâmetros visam ajustar seus valores durante a execução do algoritmo, enquanto a afinação de parâmetros se remete ao processo que visa obter os melhores valores de parâmetros antes da operação do algoritmo, que serão fixados durante sua inicialização e imutáveis na sua execução.

Eiben et al. (1999) argumentam que definir o AE com um conjunto estático de parâmetros é inapropriado, dadas as características do problema de parametrização, que depende das instâncias fornecidas. Apesar de descrever com muitos detalhes as

possibilidades de parametrização para diversos operadores de algoritmos genéticos, esses autores não apresentam estratégias de solução para a afinação de parâmetros focando-se na classificação das estratégias para parametrização *on-line* e nos meios que podem ser aplicados para variar e controlar parâmetros de um algoritmo genético em tempo de execução.

Eiben et al. (2007) revisam e ampliam o trabalho feito por Eiben et al. (1999), discutindo as características de algoritmos genéticos e seus parâmetros com casos de estudos envolvendo AEs, especificamente abordando a evolução histórica do algoritmo ES (*Evolution Strategies*). Esses autores descrevem os modelos e parâmetros para esses algoritmos e em seguida voltam-se para estratégias adaptativas buscando variar e controlar tais parâmetros.

Stützle et al. (2010) abordam o tema da configuração de meta-heurísticas e a categorizam em procedimentos *off-line* e *on-line*, similarmente à Eiben et al. (1999). Estratégias *off-line* devem ser executadas previamente à utilização do algoritmo, tendo como objetivo obter a melhor configuração possível executando um experimento estocástico sobre o domínio dos parâmetros configuráveis e as instâncias conhecidas para o treinamento do AE. Os valores obtidos devem então fazer parte do algoritmo otimizador, que irá operar em um segundo momento com instâncias inéditas do problema.

Já os procedimentos *on-line* representam algoritmos adaptativos que devem ser configurados em tempo de execução sobre a prerrogativa de encontrar a melhor parametrização para cada novo problema combinatório oferecido ao algoritmo otimizador. A literatura mostra que as experiências com a afinação *off-line* tem se mostrado superiores às com a configuração *on-line*, principalmente pelo fato do procedimento de afinação *on-line* não dispensar a necessidade da *off-line*, uma vez que os parâmetros devem possuir valores iniciais para operação do algoritmo, podendo também existir parâmetros fixos mesmo em uma estratégia adaptativa (Stützle et al., 2010).

Na revisão da literatura que trata da adaptação em tempo de execução dos parâ-

metros para algoritmos de otimização baseados em colônias de formigas, Stützle et al. (2010) encontram 22 trabalhos. Esses autores concluem que apesar de existirem esforços significativos com relação ao desenvolvimento de soluções baseadas na parametrização em tempo de execução, poucos mostraram vantagens computacionais claras em seus métodos.

Os trabalhos de Lin e Gen (2009), Osorio et al. (2011), Guo et al. (2012) e Wang et al. (2015b) desenvolvem estratégias adaptativas visando balancear a exploração do espaço de busca para os algoritmos. Segundo esses autores, tentando evitar que o AE convirja para mínimos locais em estágios iniciais da busca é importante aumentar a diversidade da população priorizando abranger melhor todo o espaço de busca, ou seja, aumentar a exploração definida como *exploration*. No entanto, para potencializar resultados de alta qualidade é necessário aumentar a exploração de soluções vizinhas à outras boas respostas encontradas, priorizando assim a exploração definida como *exploitation*. Segundo esses autores, estratégias que modificam as características de operação do algoritmo em tempo de execução permitem balancear melhor os esforços do algoritmo, entre *exploration* e *exploitation*, potencializando as chances de obter melhores resultados do processo de busca.

Zeng et al. (2010), Martínez et al. (2011), Xu et al. (2016) e Lin et al. (2016) também desenvolvem estratégias adaptativas para AEs, mas focados em problemas multi objetivo. Pellegrini et al. (2012) revisam os métodos adaptativos propostos para o ACO, argumentando que as estratégias apresentados esse algoritmo até o momento dessa pesquisa não demonstram melhorias para seu desempenho, e em diversos casos atingem resultados piores que as versões não adaptativas.

Os autores de Aleti e Moser (2016) apresentam uma revisão extensiva dos esforços referentes a implementações de algoritmos adaptativos. Segundo Aleti e Moser (2016), a principal vantagem para o emprego de estratégias adaptativas encontra-se na eliminação da necessidade do processo de configuração, permitindo o uso por praticantes que não

possuem conhecimento profundo sobre o algoritmo otimizador aplicado. No entanto, Stützle e Hoos (2000) afirmam que estratégias adaptativas não dispensam a necessidade de parametrização *off-line*, pois dificilmente a estratégia adaptativa engloba todos os parâmetros configuráveis do algoritmo e ainda valores iniciais devem ser apresentados.

O presente projeto busca a mesma vantagem apontada por Aleti e Moser (2016), no entanto, sem aplicar estratégias adaptativas, mas utilizando apenas a parametrização *off-line* apoiada por um sistema para o controle do meta-aprendizado para a configuração de AEs. Possibilitando assim, que o praticante não possua conhecimentos técnicos avançados em relação ao algoritmo configurável, mas conseguindo obter parâmetros ideais para a operação do algoritmo sobre instâncias inéditas.

Ainda que técnicas para parametrização *on-line* ofereçam a oportunidade de alcançar maior precisão e flexibilidade para a parametrização de algoritmos. Não representa um objetivo deste trabalho aplicar estratégias adaptativas. Porém, não é descartada essa possibilidade para pesquisas futuras.

Focando-se em estratégias de configuração *off-line*, doravante denominada simplesmente como "parametrização", e definindo um "vetor de parâmetros" como a composição dos valores que determinam uma instância específica para um algoritmo configurável. Eiben e Smit (2011a) argumentam que, algoritmos para parametrização funcionam essencialmente sob o paradigma de gerar e testar, ou seja, escolher um vetor de parâmetros para a função utilidade e avaliar o resultado obtido na função objetivo. Dessa maneira, os algoritmos para parametrização são primeiramente classificados como *não iterativos*, aqueles que executam o processo para gerar vetores de parâmetros uma única vez, durante sua inicialização, e *iterativos* que podem gerar novos vetores durante o processo de execução.

De acordo com Eiben e Smit (2011b), um bom algoritmo para parametrização é aquele que procura maximizar a função utilidade, mas otimizar outros três aspectos - $A \cdot B \cdot C$ - que definem o esforço computacional para sua execução. A variável A

refere-se ao número de vetores testados pelo algoritmo. B é o número de instâncias avaliadas para cada vetor do AE e C é o número de iterações executadas internamento pelo próprio AE em cada execução.

Métodos que tentam minimizar A desenvolvem estratégias inteligentes – normalmente iterativas – para a geração dos novos vetores de parâmetros. Procurar por um bom vetor de parâmetros para a operação do AE é essencialmente um problema de busca para os quais AEs representam uma boa estratégia de solução. O algoritmo no camada de projeto (ver figura 3.1) é chamado de meta-AE e qualquer estratégia evolutiva pode ser aplicada para sua solução.

Métodos para otimizar o valor de B tentam reduzir o número de execuções do algoritmo para um dado vetor de parâmetros, ou seja, avaliar um vetor sobre o menor número de instâncias possíveis. Estratégias aplicadas para esse fim são: triagem estatística, ranqueamento e seleção. Exemplos desses métodos são Algoritmos de Corrida (AC), ou *Racing Algorithms*, como o F-Race de Birattari et al. (2002). Tais algoritmos realizam procedimentos estatísticos para determinar a utilidade dos vetores de parâmetros, excluindo aqueles com baixo potencial e ampliando o volume de testes para os vetores que apresentam maior potencial de obter alta utilidade para as instâncias testadas. Segundo Eiben e Smit (2011b), a principal diferença entre métodos de corrida distintos são os testes estatísticos aplicados.

Algoritmos que buscam minimizar $A \cdot B$ são raros na literatura. Como exemplos podemos citar o trabalho de Yuan e Gallagher (2007), que une as técnicas dos ACs e o meta-AE, o IRACE de Balaprakash et al. (2007), o REVAC++ de Smit e Eiben (2009) e o FocusedILS de Hutter et al. (2009). Mecanismos para reduzir C (o número de iterações por execução do AE) visam terminar o processo do algoritmo antes de seu número máximo de avaliações. Apenas um trabalho foi encontrado com aplicação de uma estratégia para minimizar o valor de C, o trabalho de Ugolotti e Cagnoni (2014) que será analisado adiante.

Eiben e Smit (2011a) descrevem as técnicas aplicadas para o processo da parametrização de algoritmos, mas observam que nenhuma delas efetivamente minimiza o resultado de $A \cdot B \cdot C$, tais técnicas apenas se esforçam na "tentativa" de minimizar o custo computacional para o processo de parametrização, enquanto o objetivo primordial do problema é maximizar a função utilidade.

A seção seguinte define o problema da parametrização de algoritmos e mais adiante são tratadas as estratégias encontradas na literatura que apresentaram algoritmos para parametrização automática de AEs.

3.2 Definição do Problema de Parametrização

Nannen e Eiben (2006) descrevem o problema da parametrização de um algoritmo configurável definindo $M = \{\theta_1, ..., \theta_k\}$ como a lista dos k parâmetros com um domínio finito para cada um deles. χ_M representa o produto cartesiano dos domínios para os parâmetros, ou seja, o tamanho do espaço de busca para o problema de afinação. Assim, uma parametrização C é uma distribuição $C(x)$ sobre os possíveis valores de $x \in \chi_M$.

O objetivo final da parametrização de uma meta-heurística, respeitando o conceito de uma sequência de instâncias, é selecionar uma devida configuração que maximize o desempenho da meta-heurística para os problemas a que ela será submetida no futuro (Birattari, 2009).

Birattari et al. (2002) e Birattari (2009) definem o problema de parametrização de uma meta-heurística a partir de oito componentes $\{\Theta, I, P_I, t, c, P_C, C, T\}$, onde:

1. Θ: representam os possíveis candidatos para configuração do algoritmo;

2. I: tipicamente o conjunto infinito de instâncias que podem compor uma sequência;

3. P_I: é a probabilidade da seleção de uma determinada instância sobre o conjunto I;

4. t: é o tempo computacional associado para o processamento de cada instância;

5. c: representa uma variável aleatória para o custo da solução encontrada utilizando uma configuração θ em uma instância i por $t(i)$ segundos;

6. P_C: é a probabilidade associada com um custo c encontrado com a execução da instância i por $t(i)$ segundos usando a configuração θ, ou $P_C(c|\theta,i)$;

7. C: é o critério sendo otimizado em relação a θ, ou $C(\theta|\Theta, I, P_I, P_C, t)$;

8. T: é o tempo total disponível para a experimentação das configurações candidatas.

Sendo o objetivo do problema de parametrização obter $\bar{\theta}$ de maneira que $\bar{\theta} = argmin_\theta C(\theta)$.

Hutter et al. (2009) também apresenta uma definição formal para o problema da configuração de algoritmos, mas a definição de Birattari (2009) fornece com mais detalhes os componentes que envolvem o problema de parametrização.

3.3 Soluções Automáticas para Parametrização de Algoritmos

Conforme descrito anteriormente, um bom algoritmo para a parametrização visa maximizar a função utilidade enquanto minimiza o custo computacional para sua execução definido pelas variáveis $A \cdot B \cdot C$. Esta seção analisa os trabalhos encontrados com algoritmos para parametrização automática de AEs em relação à esses aspectos.

Birattari et al. (2002) criticam a alocação da mesma quantidade de recursos computacionais para cada configuração, ou seja, quando todos os vetores de parâmetros são avaliados um mesmo número de vezes. Esses autores argumentam que métodos anteriores, como o B-Race que enumera e avalia todas as configurações de um SVM,

não são eficazes para a configuração de uma meta-heurística devido a complexidade desse problema.

Com o propósito de definir um procedimento automático para encontrar bons parâmetros para a operação de meta-heurísticas, Birattari et al. (2002) definem formalmente o problema para a parametrização de uma meta-heurística. Esses autores apresentam o F-Race, um algoritmo de corrida que utiliza o teste estatístico de Friedman. Esses autores aplicam o F-Race para configurar um ACO desenvolvido para o problema do caixeiro viajante (TSP – *Traveling Salesman Problem*).

O F-Race cria uma lista de candidatos com vetores de parâmetros para o algoritmo configurável e os utiliza para solucionar um conjunto de instâncias do problema tratado em um procedimento iterativo. O processo descarta os vetores assim esses demonstrem resultados ruins para um grupo de instâncias, quando comparados com os demais a partir do teste de Friedman. Segundo Balaprakash et al. (2007), o F-Race utiliza um design fatorial completo para criar a lista de candidatos a partir do espaço amostral dos parâmetros, no entanto, nenhum dos autores descrevem tal procedimento, mas afirmam que ele reduz significativamente o espaço dos candidatos aos vetores de parâmetros. O trabalho de Birattari et al. (2002) apresenta uma estratégia para minimizar o valor de B, enquanto A é definido pelo design fatorial completo. Ainda segundo Balaprakash et al. (2007), para configurar algoritmos com muitos parâmetros e domínios extensos a estratégia de design fatorial completo é impraticável.

Nannen e Eiben (2006) apresentam o método CRE (*Calibration and Relevance Estimation*) com os objetivos de estimar quanto um determinado AE é sensitivo aos valores de cada um dos seus parâmetros e também estabelecer bons valores para esses parâmetros. Os autores ainda destacam que bons AEs também devem possuir baixa complexidade para sua configuração.

Tratando um modelo econômico chamado *Simulated Economic Environment* (SEE), Nannen e Eiben (2006) desenvolvem um AE customizado que inclui parâmetros rela-

cionados aos dados de entrada das instâncias do problema. Avaliando as distribuições marginais que relacionam cada parâmetro com o desempenho do AE, o método CRE é capaz de avaliar a influência de cada parâmetro para a eficiência do algoritmo e definir as distribuições para os parâmetros que originam bons AEs. O CRE é um método que permite comparar diferentes vetores de parâmetros e a partir de um processo evolutivo determinar os melhores valores para os parâmetros do algoritmo configurável.

Estendendo o trabalho de Birattari et al. (2002), Balaprakash et al. (2007) testam duas modificações para o sistema F-Race. A primeira proposta é gerar um grande número de vetores candidatos aleatoriamente, a segunda é baseada em um modelo que gera pequeno conjuntos de vetores com parâmetros a cada iteração selecionando as configurações elite e direcionando a geração de novos em torno de soluções com alta qualidade.

O processo aleatório definido por Balaprakash et al. (2007) utiliza distribuições uniformes e é chamado RSD/F-Race (*Random Sampling Design F-Race*), enquanto a versão iterativa I/F-Race (*Iterated F-Race*) utiliza o *feedback* através do processo iterativo para guiar a probabilidade de escolha para criar novos vetores de parâmetros. Esses autores aplicam seu método de parametrização em três algoritmos configuráveis e problemas distintos: i) um ACO para o TSP; ii) um método de busca chamado *Estimation-based Local Search* (ELS) para problemas combinatórios de otimização estocástica (*Stochastic Combinatorial Optimization Problems* – SCOP); iii) um algoritmo SA para o VRP-SD (*Vehicle Routing Problem with Stochastic Demands*). O RSD/F-RACE minimiza apenas A, enquanto o I/F-RACE, doravante denominado apenas como IRACE, representa uma técnica iterativa combinada com o algoritmo de corrida e tenta minimizar as variáveis $A \cdot B$.

De acordo com Yuan e Gallagher (2007), a aplicação de ACs para parametrização de AEs difere da seu uso original para o problema da seleção de modelos (MSP – *Model Selection Problems*) no campo de estudos que trata o aprendizado de máquinas. No

MSP, o número para pontos de teste é limitado, enquanto, para a parametrização de um AE, os pares envolvendo configurações e instâncias possíveis é ilimitado.

No MSP, tipicamente um grupo de M candidatos para modelos de aprendizagem supervisionada é avaliado em um conjunto de dados contendo N instâncias de teste. O princípio geral desses algoritmos é o mecanismo *Leave-One-Out Cross Validation* (LOOCV) que calcula o erro de um modelo em relação aos $M - 1$ candidatos, aquele com menor erro após N avaliações é considerado o melhor modelo. Exemplificando, o algoritmo 1 descreve o processo de um AC tradicional (Yuan e Gallagher, 2007).

Algoritmo 1: Pseudo-Código do Algoritmo de Corrida. Fonte: adaptado de Yuan e Gallagher (2007)

algoritmo Algoritmo de Corrida (AC)
enquanto *existirem instâncias para avaliar* ***OU*** *existe mais de um modelo para testar* **faça**
1. Selecionar aleatoriamente uma nova instância
2. Calcular o erro LOOCV para cada modelo ainda existente
3. Aplicar um teste estatístico com comparação pareada para comparar os modelos
4. Descartar modelos significativamente piores que outros

fim
retorno Retornar o modelo restante com o menor erro
fim Algoritmo de Corrida

Para Yuan e Gallagher (2007) a vantagem do método de corrida é não testar todos os modelos (ou aqui vetores de parâmetros) em todas as instâncias disponíveis, e utilizar uma metodologia estatística para descartar o mais cedo possível aqueles que apresentam pior desempenho.

Yuan e Gallagher (2007) apresentam duas estratégias distintas de meta-AEs combinados com um algoritmo de corrida para a parametrização de um AG com o vetor de parâmetros descrito por $\{P, P_c, P_m, S, C, E\}$ onde P é o tamanho da população, P_c e

P_m são as probabilidades de cruzamento e mutação, S, C e E são parâmetros categóricos que definem a estratégia de seleção, tipo de cruzamento e aplicação de elitismo respectivamente. Esses autores utilizam seu AG para solucionar o instâncias de um *benchmark* do problema *100-bit One-Max*.

O primeiro meta-AE de Yuan e Gallagher (2007) é aplicado apenas para a parametrização de $\{P, P_c, P_m\}$, com domínios contínuos. Enquanto o segundo meta-AE apresenta uma formulação mais complexa e resolve o problema da parametrização AG considerando todos os parâmetros, mas enumera os categóricos (11 estratégias de seleção, 3 opções de cruzamento e 2 para o elitismo) somando um total de 66 instâncias do algoritmo para as quais $\{P, P_c, P_m\}$ são parametrizados. As estratégias apresentadas por Yuan e Gallagher (2007) minimizam as variáveis $A \cdot B$.

Nannen et al. (2008) complementam o esforço anterior de Nannen e Eiben (2006), e implementam o REVAC (*Relevance Estimation and Value Calibration*), um algoritmo para estimar distribuições (EDA – *Estimation of Distribution Algorithm*). Começando com 100 vetores de parâmetros aleatórios, o REVAC iterativamente gera novas distribuições marginais com crescente valor esperado para o desempenho do AE em um processo que cria novos vetores a partir de uma distribuição que é avaliada e atualizada a cada iteração.

Nannen et al. (2008) utilizam uma biblioteca do Java para criar AEs o *Evolutionary Computation toolkit* – (ECJ). Desenvolvendo 120 diferentes instâncias de AEs, variando entre três e seis parâmetros livres cada uma, gerando problemas baseados em quatro tipos de superfícies para otimização (*fitness landscapes*) com 10 dimensões para testar tais algoritmos. Cada AE foi submetido ao processo de configuração cinco vezes, sendo que em cada execução o REVAC gera até 1000 vetores diferentes com valores de parâmetros.

Smit e Eiben (2009) combinam técnicas de refinamento e algoritmos de corrida com o REVAC, propondo o REVAC++. Os autores comparam as variações da aplicação

do REVAC combinado apenas com refinamento ou corrida e também ambos. Para comparar os resultados do REVAC, Smit e Eiben (2009) utilizam pacotes disponíveis para configuração automática de outros autores, como o SPO apresentado em Bartz-Beielstein et al. (2004) e Bartz-Beielstein et al. (2005).

Tratando instâncias do problema de otimização definido por funções *Rastrigin* com 20 dimensões, Smit e Eiben (2009) aplicam o ECJ para criar algoritmos configuráveis e testar sua parametrização. Os métodos avaliados por esses autores, que combinam técnicas de corrida com meta-AEs, oferecem estratégias de solução que minimizam as variáveis $A \cdot B$.

Segundo Hutter et al. (2009), o ParamILS, que aplica o algoritmo FocusedILS, é um método de busca iterativo sobre espaços de parametrização para algoritmos configuráveis (ILS – *Iterated Local Search*). Esses autores descrevem métodos para otimizar os parâmetros do algoritmo relacionando um determinado conjunto de instâncias do problema para treinamento.

Primeiramente, Hutter et al. (2009) descrevem o sistema BasicILS que define o processo básico da busca iterativa. Já o sistema FocusedILS, derivado do BasicILS, utiliza uma método para comparar duas parametrizações de acordo com o conceito de dominância, determinando que um vetor de parâmetros é superior a outro se a avaliação de pelo menos o mesmo número de testes foi realizado com os dois algoritmos e o primeiro fornece resultados sempre iguais ou melhores que o segundo. De acordo com Hutter et al. (2009), o método FocusedILS permite que um vetor de parâmetros seja descartado assim que avaliado sobre um número mínimo de instâncias diferentes e comprovado que outro vetor ofereceu melhores resultados para esse bloco de instâncias. A estratégia iterativa do BasicILS tenta minimizar A, enquanto a estratégia apresentada pelo FocusedILS se esforça em minimizar $A \cdot B$.

Hutter et al. (2009) aplicam seu sistema para a configuração da versão 10.0 do algoritmo CPLEX, que conta com 80 parâmetros que influenciam o desempenho do

algoritmo. Utilizando o *benchmark* denominado *Regions100*, com 2000 instâncias do problema CATS *Combinatorial Auctions*, mas também para os algoritmos SAPS (*Scaling And Probabilistic Smoothing*) e SPEAR (*Fast Bit-vector Arithmetic Theorem Prover*) ambos para amostrando 2000 instâncias de dois *benchmarks*, um com 20000 exemplos de teste para o problema SW-GCP (*Graph Colouring Problem (GCP) based on the Small World (SW)*) e 23000 instâncias do problema *Quasi-group Completion Problem* (QCP).

Birattari (2009) segue a perspectiva do aprendizado de máquina e descreve o problema de configuração para uma meta-heurística e as características do método F-Race, RSD/F-Race e I/F-Race para solucionar esse problema. Configurando um algoritmo ILS para o problema *Quadratic Assignment Problem* (QAP) e implementações de ACOs para o TSP, mas também o problema *Timetabling Problem* (TT).

Smit e Eiben (2010a) testam seu sistema para melhorar a parametrização do algoritmo campeão na disputa CEC-2005 sobre o *benchmark CEC-2005 Test-suite*. Esses autores demonstram que o desempenho do algoritmo otimização vencedor foi ainda melhor após sua parametrização através do REVAC.

Smit e Eiben (2010b) utilizam um AG com operações básicas para testar suas hipóteses utilizando o REVAC. As instâncias de teste escolhidas são determinadas por diferentes funções dimensionais (*Ackley, Griewank, Sphere, Rastrigin,* e *Rosenbrock*). Os experimentos realizados por esses autores determinam que a função *Rosenbrock* apresenta maior complexidade para a configuração dos melhores valores para a operação do algoritmo, de forma que, o "generalista" encontrado foi aquele com bom desempenho sobre essa função e razoável sobre as demais.

A análise de Smit e Eiben (2010b) sobre as configurações "especialistas" para seu algoritmo mostra que, para a função esférica (que é a mais simples entre as funções) seu "especialista" apresentou baixo desempenho sobre as demais funções. O "especialista" encontrado para a função *Ackley* supera o da função *Rastrigin* para todas as funções e

os "especialistas" para *Rosenbrock* e *Griewank* apresentaram resultados similares.

Segundo Li et al. (2011), sequências UIO (*Unique Input Output*) são usadas em FSM (*Finite State Machines*) e a literatura demonstra aplicações de AEs para a otimização de UIOs. Esse autores desenvolvem uma estratégia chamada *Fitness-Probability Clouds* (FPC), que determina quatro diferentes classificadores para problemas UIO, para caracterizar as possíveis instâncias.

Dessa forma, Li et al. (2011) desenvolvem uma estratégia em duas etapas, uma de configuração utilizando um algoritmo SVM e outra de predição da melhor configuração para seu AE em cada nova instância do problema. Na primeira fase um AE é treinado de acordo com as instâncias de teste para o problema UIO, avaliando o desempenho denotado por E_{ij} que relaciona um conjunto de parâmetros j a uma instância do problema i. Em uma segunda fase, novas instâncias são apresentadas ao algoritmo que deve determinar quais conjuntos de parâmetros são mais adequados a essa nova instância, ou seja, predizer boas parametrizações para a solução do problema inédito.

O AE implementado por Li et al. (2011) possui apenas 72 configurações distintas e todas foram avaliadas, portanto essa solução não visa minimizar o número de vetores testados (A). Os valores de B e C também são constantes no processo apresentado por Li et al. (2011), que é focado na configuração minuciosa de um algoritmo otimizado, porém não tem por objetivo minimizar o custo computacional para tal processo.

Kattan e Arif (2012) aplicam um PSO baseado em SM (*Surrogate Modelling*) para a parametrização de um AG, utilizando o método RBFN (*Radial Basis Function Networks*) para, a partir dos pontos avaliados dentro dos domínios de cada parâmetro contínuo, determinar aproximações das funções que definem o comportamento desses parâmetros.

Para testar seu algoritmo, Kattan e Arif (2012) definem a camada da aplicação com os problemas: *NK-Landscape* unimodal e multimodal, além de outros três problemas de otimização contínua. Kattan e Arif (2012) aplica um meta-AE, mas o número de testes por vetor (B) é constante, todas as configurações são avaliadas para um mesmo grupo

de instâncias.

Tanabe e Fukunaga (2013) aplicam um modelo que permite utilização de diferentes vetores de parâmetros em um AE para a solução do TSP e do QAP, chamado RHIM (*Random Heterogeneous Island Model*). Nesse modelo, as ilhas representam configurações distintas para o funcionamento do AE. Tais ilhas são denominadas aleatórias pois os vetores de parâmetros que as definem são uniformemente selecionados a partir do domínio determinado para cada um dos parâmetros do AE. Tal estratégia baseada em ilhas não procura a minimização de A ou mesmo B, no entanto, seus autores relatam que a possibilidade de configurações heterogêneas possibilitou a obtenção de 100% de respostas ótimas em diversos *benchmarks*.

Implementando um meta-AE, Riff e Montero (2013) descrevem a aplicação de um AG para o processo de parametrização e comparam seus resultados com outros softwares: ParamILS e REVAC. Na camada da aplicação, Riff e Montero (2013) utilizam o problema NK-Landscape e no *layer* do algoritmo outro AG. Nesse trabalho o número de testes para cada vetor (B) é constante, mas A é minimizado com o meta-AE.

Yeguas et al. (2014) desenvolvem um sistema baseado em BN (*Bayesian Networks*) e na metodologia CBR (*Case-Based Reasoning*) para estimar a melhor configuração e maximizar o desempenho de AEs, que são aplicados a solução do RIP (*Root Identification Problem*). Como os dois AEs testados por esses autores, possuem respectivamente 875 e 735 modos de configuração distintos, foram avaliados todos os vetores de parâmetros possíveis com 50 execuções cada. Portanto, no trabalho de Yeguas et al. (2014), B é constante e igual a 50 multiplicado pelo total de instâncias testadas e A é máximo, ou seja, todos os vetores de parâmetros são testados.

Ugolotti e Cagnoni (2014) implementam um NSGA-II, como meta-AE, para a parametrização de algoritmos visando balancear a velocidade de execução e a qualidade para um DE (*Differential Evolution*) e um PSO na camada do algoritmo. Funções *Rastrigin* e *Griewank* com 10 e 30 dimensões foram utilizadas na camada da aplicação.

Ugolotti e Cagnoni (2014) discutem que a aplicação de um AEMO permite avaliar diferentes critérios para o desempenho dos AEs sendo parametrizados como robustez e velocidade.

Tratando-se de um meta-AE, o algoritmo de Ugolotti e Cagnoni (2014) minimiza A enquanto cria vetores de parâmetros usando uma estratégia inteligente e foi o único trabalho encontrado que avalia a variável C, tratando o problema de parametrização com múltiplos objetivos, maximizando a função utilidade mas minimizando o número de gerações executadas. No entanto, nessa proposta o valor de B permanece constante.

Para Sinha et al. (2014), a aplicação de um algoritmo configurável a um problema de otimização é representado por dois subproblemas. Esses autores abordam esse problema em um modelo multi nível, onde a configuração do algoritmo define o nível inferior do problema e a aplicação do algoritmo define o segundo nível do problema.

Sinha et al. (2014) desenvolvem um algoritmo chamado *Bilevel Automated Parameter Tuning* (BAPT), uma abordagem baseada em um meta-AE, avaliando seus resultados para parametrização de um DE que busca solucionar problemas envolvendo funções de superfícies para otimização. Esses autores fornecem uma formulação matemática para o problema, considerando que a parametrização deve ser solucionada para uma instância específica. O trabalho de Sinha et al. (2014) oferece a minimização da variável A, no entanto, B é desconsiderado uma vez que os problemas de parametrização e otimização são solucionados de maneira integrada para cada instância do problema da camada de aplicação.

Por sua vez, Vecek et al. (2016) argumentam que meta-heurísticas devem ser comparadas utilizando os melhores valores de parâmetros para todos os algoritmos envolvidos. Esse autores desenvolvem um meta-AE, denominado *CRS-Tuning* que incorpora um sistema de ranqueamento chamado *Chess Rating System for Evolutionary Algorithms* (CRS4EAs).

Para avaliar o algoritmo de configurador implementado, Vecek et al. (2016) apre-

sentam a parametrização de dois algoritmos configuráveis, *Artificial Bee Colony* (ABC) e DE, comparando seus resultados com outros softwares já existentes para o mesmo propósito, IRACE (de Balaprakash et al. (2007)) e REVAC (de Nannen et al. (2008)). O algoritmo apresentado por Vecek et al. (2016) se aplica a minimização de A, mas executa cada vetor de parâmetros sempre para um mesmo número de instâncias.

A próxima seção resume e analisa mais detalhes dos resultados desta revisão de literatura.

3.4 Análise da Revisão de Literatura

Dentre os trabalhos referenciados, 10 apresentaram apenas implementações de AEs, mencionando a importância da sua parametrização, mas sem apresentar estratégias robustas para a solução desse problema (Tan et al., 2001, 2003; Xu e gen He, 2005; Grosan et al., 2005; Kaji et al., 2008; Chen et al., 2012; Joan-Arinyo et al., 2011; Bosman, 2012; Ma et al., 2014; Gupta e Kumar, 2014). Outros 21 foram classificados fora do escopo desta pesquisa por tratarem da parametrização em outros tipos de sistemas, como a configuração de controladores eletrônicos, sistemas de captura visual, clusterização, veículos autônomos, entre outros (Muñoz-Salinas et al., 2008; Herrero et al., 2008; Gacto et al., 2009; Ethridge et al., 2010; Hu et al., 2010; Chaparro e Sosa, 2011; Junli et al., 2011; Pan et al., 2011; Khoshrooz et al., 2012; Reynoso-Meza et al., 2012; Wang et al., 2012; Reynoso-Meza et al., 2013; Fadil e Darus, 2013; Lutton et al., 2014; Zaefferer et al., 2014; Jovanovic et al., 2014; Saad et al., 2015; Su e Li, 2015; Moness e Moustafa, 2015; Gutiérrez-Urquídez et al., 2015; Onieva et al., 2015).

Wolpert e Macready (1997) apresentam comprovações matemáticas baseadas no *No Free Lunch Theorem* de que testar poucas instâncias representa um risco ao se avaliar o desempenho de um algoritmo evolucionário. Enquanto o livro de Bartz-Beielstein et al. (2014) relaciona as melhores práticas existentes para a configuração de algoritmos.

Por sua vez, Rabanal et al. (2015) afirma que testes sobre algoritmos estocásticos devem considerar um conjunto não limitado de instâncias, uma vez que é simples obter bons parâmetros para operação de AEs em *benchmarks* finitos.

Bezerra et al. (2015) se dedica a configuração de AEMOs, tema que pode ser abordado em pesquisas futuras. García-Sánchez et al. (2016) trata a paralelização de AEs em máquinas heterogêneas. Andersson et al. (2016) Jiang e Yang (2016) consideram a parametrização de algoritmos evolutivos avaliando peculiaridades distintas ao foco deste projeto.

Munoz-Carpintero e Sáez (2015), Sadeghi e Niaki (2015), Switalski e Seredynski (2015), Wood (2016) e Mishra e Rao (2016) tratam problemas ligados à Engenharia de Produção. Esses autores consideram a parametrização do algoritmos configuráveis empregados como um importante passo para obtenção de boas soluções, mas não oferecem uma estratégia para resolver tal problema.

Outros 34 trabalhos foram agrupados como contendo estratégias para a parametrização de AEs, tanto *on-line* como *off-line*. Destacam-se nove trabalhos do grupo de pesquisa da *University of Amsterdam* (e.g. Eiben et al. (1999), Nannen e Eiben (2006), Eiben et al. (2007), Nannen et al. (2008), Smit e Eiben (2009), Smit e Eiben (2010b), Smit e Eiben (2010a), Eiben e Smit (2011a) e Eiben e Smit (2011b)) e sete da *Université Libré de Bruxelles* (e.g. Birattari et al. (2002), Balaprakash et al. (2007), Birattari (2009), Birattari et al. (2009), Hutter et al. (2009), Stützle e Hoos (2000) e Pellegrini et al. (2012)), além de 18 compostos por outros autores (e.g. Yuan e Gallagher (2007), Lin e Gen (2009), Zeng et al. (2010), Osorio et al. (2011), Li et al. (2011), Martínez et al. (2011), Guo et al. (2012), Kattan e Arif (2012), Tanabe e Fukunaga (2013), Riff e Montero (2013), Yeguas et al. (2014), Ugolotti e Cagnoni (2014), Sinha et al. (2014), Wang et al. (2015b), Xu et al. (2016), Lin et al. (2016), Aleti e Moser (2016) e Vecek et al. (2016)).

A revisão construída permitiu compreender primeiramente como tratar o desempe-

nho de algoritmos configuráveis. As características estocásticas ocasionam variância no resultado de amostras no mesmo experimento, bons algoritmos e sementes aleatórias são capazes de minimizar esse ruído. No entanto, outros dois fatores também são importantes: as características de desempenho relacionadas à diferentes instâncias de problemas aplicadas ao algoritmo e a possibilidade da existência de classes heterogêneas dentre essas instâncias; e a influência dos parâmetros que definem a operação do algoritmo em relação ao seu desempenho.

Outro ponto importante relevado pela revisão apresentada é a necessidade de minimizar o custo computacional do processo da parametrização de algoritmos configuráveis. No entanto, a busca por um vetor de parâmetros que maximize a função utilidade depende das instâncias disponíveis para o problema e envolve a amostragem e avalização de inúmeros vetores para que comprovar qual é o vetor de parâmetros que maximiza o desempenho do algoritmo sobre as instâncias fornecidas. Deseja-se sempre testar o menor número de configurações (A), descartando-as o mais cedo possível uma vez que se encontre evidências estatísticas de que uma é inferior às demais (B), mas também minimizar o número de iterações necessárias para a execução do algoritmo configurável (C). A tabela 3.1 resume os trabalhos que trataram a parametrização *off-line*, foco deste projeto.

Tabela 3.1: Problemas e Estratégias Aplicadas em Algoritmos de Parametrização Automáticos

Trabalho	Alg. Configurador	Alg. Otimizador	Problema	*A*	*B*	*C*
Birattari et al. (2002)	F-Race	ACO	TSP	Design Fatorial Completo	Corrida	cte.
Nannen e Eiben (2006)	CRE	AE Customizado	SEE	Meta-AE	cte.	cte.
Balaprakash et al. (2007)	RSD/F-race, IRACE	ACO, ELS, SA	TSP, SCOP, VRP-SD	Aleatório, Iterativo	Corrida	cte.
Yuan e Gallagher (2007)	$(1+\lambda)ES$, $(\mu+\lambda)ES$	AG	100-bit One-Max	Meta-AE	Corrida	cte.
Nannen et al. (2008)	REVAC	ECJ	Superfícies	Meta-AE	cte.	cte.
Smit e Eiben (2009)	CMA-ES (Racing), REVAC (Racing), SPO	ECJ	Superfície: Rastrigin	Meta-AE	Corrida	cte.
Hutter et al. (2009)	BasicILS, FocusedILS, ParamILS	CPLEX, SAPS, SPEAR	QCP, SW-GCP, CATS	Iterativos	cte., Corrida	cte.
Birattari (2009)	F-Race, RSD/F-Race, IRACE	ILS, ACO	TSP, QAP, TT	Iterativo (I/F-race)	Corrida	cte.
Birattari et al. (2009)	F-Race, RSD/F-Race, IRACE	ACO	TSP	Iterativo (I/F-race)	Corrida	cte.
Smit e Eiben (2010b)	REVAC	AG	Superfícies	Meta-AE	Corrida	cte.
Smit e Eiben (2010a)	REVAC	G-CMA-ES	CEC-2005 Test-suite	Meta-AE	Corrida	cte.
Li et al. (2011)	SVM	AE	UIO	cte.	cte.	cte.
Kattan e Arif (2012)	PSO+SM+RBFN	AG	NK-Landscape	Meta-AE com Preditor	cte.	cte.
Tanabe e Fukunaga (2013)	RHIM	DE	TSP, QAP	cte.	cte.	cte.
Riff e Montero (2013)	AG	AG	Superfícies	Meta-AE	cte.	cte.
Yeguas et al. (2014)	BN + CBR	CHC, PBIL	RIP	cte.	cte.	cte.
Ugolotti e Cagnoni (2014)	NSGA-II (Qualidade, Velocidade)	DE, PSO	Superfícies	Meta-AE (Qualidade)	cte.	Restrição (Velocidade)
Sinha et al. (2014)	BAPT	DE	Superfícies	Meta-AE	cte.	cte.
Vecek et al. (2016)	CRS-Tuning	ABC, DE	Superfícies	Meta-AE	cte.	cte.

Dentre os 34 trabalhos que tratam da parametrização de algoritmos, 15 apresentaram definições e/ou aplicações de estratégias para o controle de parâmetros (Eiben et al., 1999, 2007; Lin e Gen, 2009; Stützle et al., 2010; Zeng et al., 2010; Osorio et al., 2011; Martínez et al., 2011; Eiben e Smit, 2011a,b; Guo et al., 2012; Pellegrini et al., 2012; Wang et al., 2015b; Xu et al., 2016; Lin et al., 2016; Aleti e Moser, 2016). Portanto, não foram relacionados na tabela 3.1.

Dos trabalhos descritos na tabela 3.1, alguns não procuram minimizar os esforços do processo de parametrização. Muitos dos pesquisadores que mostram meta-AEs não os classificam como tal. O centro de pesquisa Holandês se dedica principalmente à parametrização de algoritmos genéticos aplicando processos *off-line*, mas também *on-line*, enquanto os pesquisadores situados na *Université Libré de Bruxelles* focam-se na parametrização de ACOs e não desenvolvem técnicas adaptativas para controle de parâmetros.

Estratégias adaptativas são encaradas como formas de minimizar os esforços necessários para a parametrização *off-line* de algoritmos evolucionários. Como tais estratégias não suprimem totalmente a necessidade de parametrização, esta pesquisa foca-se no desenvolvimento de um mecanismo para parametrização *off-line*, mas acrecido de uma estratégia para o controle de resposta, buscando a automatização do processo de configuração para AEs. Não obstante, técnicas adaptativas representam uma alternativa para a melhoria dos processos evolutivos em AEs e devem ser exploradas em pesquisas futuras.

Foram encontrados trabalhos que tratam classes heterogêneas de instâncias, estratégias para minimizar o custo computacional do processo de configuração e também meios para se avaliar o desempenho de algoritmos configuráveis como AEs, meta-heurísticas e outros algoritmos de busca estocásticos.

O conteúdo apresentado na seção 3.1 descreveu como analisar o desempenho de algoritmos configuráveis, além de quais são os objetivos quando parametrizando tais

algoritmos e também técnicas para automatizar o processo de configuração. Até o momento, apenas um trabalho demonstrou uma aplicação de *Swarm Intellingence* para a parametrização de algoritmos evolutivos, ver Kattan e Arif (2012). Nenhum trabalho aplicando ACO como meta-AE foi encontrado.

Observamos estratégias distintas para a geração dos vetores de parâmetros (A) e também para definir a quantidade de testes em cada vetor (B), mas apenas Ugolotti e Cagnoni (2014) apresentaram uma solução para minimizar C. Destacamos duas estratégias principais para a parametrização de algoritmos, a aplicação de meta-AEs e ACs. Essas estratégias não são excludentes e são aplicadas para fins distintos. Meta-AEs são utilizados para minimizar os esforços do processo de configuração em relação ao número de vetores criados, enquanto os algoritmos de corrida realizam testes para determinar vetores de utilidade inferior e descartá-los em futuras iterações. Tais estratégias descrevem processos eficazes para a parametrização de algoritmos.

Observando os resultados quanto a configuração de parâmetros em tempo de execução, além da indispensabilidade em determinar-se inicialmente os valores para dos parâmetros. O presente projeto concentra-se no desenvolvimento de um meta-ACO para a parametrização *off-line*, mas de maneira inovadora, ativada automaticamente através de um processo para controle da qualidade de resposta e gerenciamento de parametrizações para um algoritmo configurável que visa otimizar uma função objetivo.

Como apontado por Rabanal et al. (2015), este projeto conta com a geração aleatória de instâncias e com a representação formal do domínio das variáveis aleatórias que compõem o modelo estudado.

Os resultados desta pesquisa consoam com afirmações encontradas na literatura, como os trabalho de Wood (2016) que combina meta-heurísticas e heurísticas para solução de problemas complexos em Engenharia de Produção. Assim como apresentado por Li et al. (2011), a solução proposta neste trabalho foi implementada com múltiplas camadas, também contendo uma fase de predição. Aqui o problema consiste em classi-

ficar uma nova instância, se necessário parametrizar e atualizar a base de informação sobre a operação do otimizador, e por fim solucionar o novo problema.

Destaca-se a diferença deste trabalho em relação ao de Sinha et al. (2014) que executa a parametrização para cada nova instância, aqui as particularidades de cada caso também são observadas, mas o processo de parametrização é invocado apenas quando necessário utilizando uma camada de implementação inteligente.

Da mesma forma que no trabalho de Tanabe e Fukunaga (2013), a camada OAA permite aplicar múltiplos vetores de parâmetros para o AE, classificando-os em classes com vetores especializados para determinados grupos de instâncias. Todavia, diferentemente da proposta de Tanabe e Fukunaga (2013), esta pesquisa ofereceu uma solução que foca na minimização das variáveis $A \cdot B \cdot C$, que determinam o número de iterações necessárias para a parametrização do AE, ao invés de aplicar força bruta para o tratamento de instância heterogêneas.

Avaliando as características de desempenho apontadas por Eiben e Smit (2011a), de acordo com as propriedades para configuração de um AE robusto: W – Aplicabilidade, X – Falibilidade, Y – Tolerância e Z – Configurabilidade (ver figura 3.3), altas amplitudes para as variáveis X (falibilidade) e Z (configurabilidade) são interpretadas como um risco para o bom desempenho de um AE. Enquanto, alta aplicabilidade significa que o AE atende as características de uma grande variação de instâncias e boa tolerância está relacionada a baixa oscilação no desempenho do AE variando seus parâmetros.

A dimensão de X determina a variação de qualidade para as respostas de um AE em diferentes instâncias, como a camada OAA trata casos heterogêneos usando múltiplos grupos com vetores de parâmetros distintos, esta proposta visa evitar a falha de um configuração fixa para uma meta-heurística reduzindo a amplitude de X. Será visto que mesmo com a aplicação da camada OAA falhas ainda são possíveis, devendo ser detectadas utilizando uma estratégia de controle.

A amplitude da variável Z determina o quanto oscila a qualidade das respostas de

um AE, enquanto são alterados os valores de seus parâmetros ajustáveis. Determinados pontos nos domínios de alguns parâmetros podem causar o mal funcionamento desse tipo de algoritmo. Também é possível que alguns parâmetros causem influência sobre outros o que dificulta ainda mais o processo de parametrização para um AE. Dessa forma, é necessário conhecimento específico e experiência sobre algoritmos evolutivos para determinar tais domínios.

Esta pesquisa destinou-se apenas a redução dos riscos causados pela alta amplitude de X. No entanto, outras estratégias podem ser aplicadas em implementações futuras visando analisar a influência dos parâmetros sobre o desempenho do algoritmo e também dependências entre eles. Por exemplo, é possível empregar uma estratégia similar ao CRE de Nannen e Eiben (2006) que analisa a variância dos resultados da função utilidade enquanto os parâmetros do algoritmo são alterados durante o processo de parametrização.

Analisando os principais componentes que causam a variância no desempenho do AE (PCA – *Principal Component Analysis*), parâmetros que pouco influenciam os resultados finais do AE podem ser fixados, ou ter seu domínio simplificado, minimizando a complexidade do AE e da sua parametrização. Aplicações estatísticas mais avançadas podem visar a análise da correlação entre os valores dos parâmetros para o resultado final do AE, com objetivo de estudar a influência dos parâmetros entre si.

Motivam também o presente trabalho, outros esforços que no passado desenvolveram meta-heurísticas com objetivo de solucionar problemas de otimização na Engenharia de Produção (e.g. Tavares Neto (2010), Tavares Neto e Godinho Filho (2011b), Tavares Neto e Godinho Filho (2011a), Tavares Neto (2012), Tavares Neto e Godinho Filho (2012), Tavares Neto e Godinho Filho (2013), Frascati (2014), Frascati et al. (2014) e Tavares Neto et al. (2015)).

Os autores de Frascati (2014) e Frascati et al. (2014), além dos tradicionais parâmetros presentes em um ACO, utilizam uma estratégia para a função de visibilidade

do ACO que pondera todos os dados de entrada do problema combinatório a partir de parâmetros exponenciais. Ainda que o algoritmo tenha demonstrado altíssimo potencial para aplicações práticas, notou-se grande complexidade para sua parametrização, mesmo com a utilização do software IRACE para o processo.

Representa um dos objetivos deste trabalho, fornecer uma estrutura computacional capaz de parametrizar um algoritmo evolutivo de maneira autônoma sempre que necessário, oferecendo assim um meio para que praticantes na área da Engenharia de Produção, não especializados em algoritmos evolutivos, possam aplicá-los, neste caso para a otimização de um problema de programação da produção, sem se preocuparem com a parametrização dos mesmos, mas com garantias da qualidade de resposta oferecida por tais algoritmos.

Capítulo 4

Otimização com Aprendizado Autônomo

Além dos trabalhos revisados sobre estratégias para configuração automática de algoritmos, vistos no capítulo anterior, outros trabalhos que abordam tópicos como configuração autônoma (termo *autonomous tuning*) foram pesquisados. Khan e Mitschele-Thiel (2008) estudaram a configuração automática de sistemas para segurança em redes de computadores. Esse trabalho aborda como um algoritmo é capaz de reconfigurar aspectos da segurança em máquinas ao longo de uma rede. Outros autores, como Kamath et al. (2015), dedicam-se à configuração automática de parâmetros em sistemas de um veículo para exploração ambientes não estruturados, como o terreno de outros planetas.

Para Soto et al. (2016), busca autônoma está relacionada com o controle de parâmetros dos algoritmos otimizadores. Esses autores aplicam o conceito de adaptação supervisionada em um algoritmo baseado em programação por restrições (CP – *Constraint Programming*). De acordo com Soto et al. (2016), encontrar os melhores parâmetros para algoritmos baseados em CP é um tarefa complexa uma vez que os valores para esses parâmetros dependem das instâncias sendo solucionadas. Para tratar esse problema, os autores utilizam uma estrutura equivalente a um meta-AE, com um algoritmo baseado

em ABC (*Artificial Bee Colony*). O meta-AE opera em uma fase inicial onde a nova instância é avaliada para as diferentes possibilidades de enumeração para o algoritmo CP por um tempo de processamento pré estabelecido. Após essa fase de treinamento, os resultados são analisados e o meta-otimizador decide quais são os melhores parâmetros para operar o algoritmo exato baseado em CP, dadas as características da nova instância.

No trabalho de Soto et al. (2016), busca autônoma está relacionada à execução de um processo para parametrização do algoritmo otimizador para cada nova instância resolvida. Tendo revisado a literatura sobre configuração automática de algoritmos e evidenciando que o termo (*autonomous tuning*) não retornou pesquisas relevantes, além de não haver descrição na literatura de um meta-AE baseado em ACO, nenhum trabalho foi encontrado aplicando o conceito de algoritmo autônomo como apresentado nesta tese.

Neste trabalho, Otimização com Aprendizado Autônomo é caracterizada por uma camada de implementação que contém um algoritmo para aprendizado de máquina que gerencia o processo da parametrização, realizado automaticamente por um algoritmo evolutivo, que é responsável por obter melhores valores para os vetores de parâmetros para outro algoritmo evolutivo aplicado à otimização de um problema NP-Difícil.

Este capítulo descreve um estudo de caso para a implementação da camada OAA para solucionar um problema de programação da produção. A próxima seção apresenta o caso estudado, o ACO implementado e seus parâmetros ajustáveis. A seção 4.2 define os mecanismos utilizados para a validação dos resultados deste projeto. Um modelo matemático, solucionado por um pacote comercial (CPLEX), foi aplicado para a obtenção de soluções ótimas para cenários de pequeno porte e uma heurística foi implementada para comparação entre os casos de grande porte. Um software para configuração automática de algoritmos (IRACE) foi aplicado para obtenção de candidatos elite para os valores dos parâmetros ajustáveis da estratégia evolutiva (ACO).

Por fim, a seção 4.3 detalha o desenvolvimento da camada de Otimização com Aprendizado Autônomo responsável por eliminar a necessidade de configuração do algoritmo ao longo de sua operação. A camada OAA contou com a implementação de um meta-AE para parametrização automática baseado em ACO, componentes para aprendizado de máquina e um mecanismo para controle da qualidade de resposta da meta-heurística (ACO) de otimização durante os processos de parametrização e aprendizado.

4.1 Caso de Estudo

Para a aplicação da estratégia proposta, tomou-se como exemplo o problema NP-Difícil que visa a minimização do custo total de terceirização das tarefas. O modelo tem como restrição garantir que nenhuma das tarefas seja entregue com atraso. Os atrasos, representados pela variável T_j, devem ser sempre menores ou iguais a zero, adiantamentos são tolerados. Considera-se sempre possível terceirizar uma tarefa e garantir que ela seja entregue dentro do prazo sob a pena de um custo para a terceirização, definido por o_j para cada tarefa. Dessa maneira quando os prazos de entrega (d_j) não puderem ser cumpridos, o modelo é forçado a recorrer à terceirização para eliminação dos atrasos.

Formalmente, a tabela 4.1 define as variáveis utilizadas para uma formulação MIP que é resultado do aprimoramento de um modelo previamente implementado em Frascati et al. (2014). Foi verificado que as equações de *Miller–Tucker–Zemlin* (MTZ) não eram necessárias, bastando restringir os valores para a diagonal principal da variável binária de decisão x_{ij}, como é visto nas equações adiante.

Tabela 4.1: Variáveis do Modelo

Variável	Descrição
n	Número de tarefas para sequenciamento
T_j	Atraso, para $j = 1...n$
C_j	Tempo de término, para $j = 1...n$
d_j	Data de entrega, para $j = 1...n$
p_j	Tempo de processamento, para $j = 1...n$
o_j	Custo de terceirização, para $j = 1...n$
s_{ij}	Tempo de *setup* para execução da tarefa j imediatamente após a tarefa i, para $\{i,j\} = 1...n$
x_{ij}	Variáveis de decisão binárias para determinar que uma tarefa j é sequenciada imediatamente após o trabalho i, para $\{i,j\} = 1...n$
y_j	Variáveis binárias para determinar se uma tarefa é terceirizada ($y_j = 1$) ou não ($y_j = 0$), para $j = 1...n$
G	Variável maior que qualquer possível C_j

A função objetivo $f(j)$, na equação 4.1, define o modelo matemático como a multiplicação do custo de terceirização pela variável de decisão (y_j) que determina se as tarefas são terceirizadas.

$$minf(j) = \sum y_j \cdot o_j \qquad \forall j \in (0 < j \leq n) \tag{4.1}$$

Os atrasos para todas as tarefas não devem ser maiores que zero.

$$T_j \geq C_j - d_j \qquad \forall j \in (0 < j \leq n) \tag{4.2}$$

$$T_j \leq 0 \qquad \forall j \in (0 < j \leq n) \tag{4.3}$$

Os tempos de processamento e *setup* são somados de acordo com as equações 4.4, para obter os tempos de término das tarefas sequenciadas em máquina única. G é uma variável que deve ser maior que qualquer possível C_j.

$$C_j \geq C_i + p_j + s_{ij} - G \cdot (1 - x_{ij}) \qquad \forall \begin{cases} \{i,j\} \in (0 \leq i \leq n, 0 < j \leq n) \\ i \neq j \end{cases} \tag{4.4}$$

$$G = \sum\nolimits_j^n (p_j + \sum\nolimits_i^n s_{ij}) \tag{4.5}$$

$$C_0 = 0 \tag{4.6}$$

Todas as tarefas devem ser sequenciadas ou terceirizadas. Uma matriz binária x_{ij} é definida para armazenar a sequência das tarefas. Quando $x_{ij} = 1$, a tarefa i é processada imediatamente antes da tarefa j. A variável y_j indica se uma tarefa j é terceirizada ($y_j = 1$) ou não ($y_j = 0$). Para garantir que uma tarefa é sequenciada ou terceirizada e prevenir que a mesma tarefa seja escolhida mais de uma vez as equações 4.7 e 4.8 são necessárias.

$$\sum_i (x_{ij} + y_j) = 1 \qquad \forall j \in (0 \leq j \leq n) \tag{4.7}$$

$$\sum_j x_{ij} \leq 1 \qquad \forall i \in (0 \leq i \leq n) \tag{4.8}$$

As equações 4.9 garante o balanço da sequência.

$$\sum_i x_{ki} = \sum_j x_{jk} \qquad \forall k \in (0 \leq k \leq n) \tag{4.9}$$

Por fim, a diagonal principal de x_{ij} deve ser igual a zero. x e y são variáveis binárias e as restantes são inteiras e maiores que zero.

$$x_{ii} = 0 \qquad \forall i \in (0 \leq i \leq n) \tag{4.10}$$

$$x_{ij}, y_i \in \{0,1\} \qquad \forall i,j \in (0 \leq i \leq n, 0 \leq j \leq n) \tag{4.11}$$

$$p_j, s_{ij}, d_j, o_j \geq 0 \qquad \forall i,j \in (0 \leq i \leq n, 0 \leq j \leq n) \tag{4.12}$$

Para solucionar o problema apresentado de maneira veloz foi desenvolvido um algoritmo baseado em colônias de formigas que é resultado do aprimoramento de trabalhos anteriores. Na implementação apresentada por Frascati et al. (2014), os métodos de busca local são aplicados apenas ao melhor agente computacional encontrado. Enquanto, neste trabalho o refinamento é realizado para todos os agentes antes da sua comparação. O código implementado em C/C++ para o ACO também sofreu melhorias em suas instruções garantindo maior velocidade de execução. Além disso, esta implementação conta com um método de busca local que não foi apresentado em Frascati et al. (2014), a busca iterativa gananciosa. Além disso, as possibilidades de refinamento foram inseridas como um parâmetro categórico para a operação do algoritmo.

Simplificadamente, um ACO é composto de três principais funções como mostrado no algoritmo 2 de Dorigo e Stützle (2002).

Algoritmo 2: Pseudocódigo ACO. Fonte: adaptado de Dorigo e Stützle (2002)

```
1 Meta-heurística ACO
2 para Número de Iteração da Meta-heurística faça
3 |   ConstruçãoDeRespostas()
4 |   AtualizaçãoDeFeromônios()
5 |   RefinamentoDeRespostas() [Opcional]
6 fim
7 fim Meta-heurística ACO
```

Descrevendo brevemente o ACO aplicado para otimização do problema de *scheduling*

NP-Difícil, a regra de transição para a construção de respostas é definida avaliando a matriz de feromônios artificiais (τ) e a função de visibilidade (η). Para o ACO implementado estas funções estão definidas nas equações 4.13, 4.14, 4.15, 4.16 e 4.17.

τ_{ij} representa a matriz de feromônios artificiais, η_{ij} é a função de visibilidade definida a partir dos dados de entrada com a ponderação dos parâmetros α_1, α_2, α_3, α_4, μ_1, μ_2, μ_3 e μ_4 para os casos de sequenciamento interno ou terceirização. P_{ij}^1 determina a probabilidade para o sequenciamento interno de uma tarefa j após a ordem i e P_{ij}^2 determina essa probabilidade para a terceirização das tarefas.

$$\eta_{ij} = \begin{cases} \eta_{ij}^1, & \textit{para sequenciar uma tarefa} \\ \eta_{ij}^2, & \textit{para terceirizar uma tarefa} \\ 0, & \textit{para tarefas já visitadas} \end{cases} \qquad \forall\{i,j\} \in (0 \leq i \leq n, 0 < j \leq n) \tag{4.13}$$

$$\eta_{ij}^1 = \frac{1}{{p_j}^{\alpha 1}} \cdot \frac{1}{{d_j}^{\alpha 2}} \cdot \frac{1}{{o_j}^{\alpha 3}} \cdot \frac{1}{{s_{ij}}^{\alpha 4}} \qquad \forall\{i,j\} \in (0 \leq i \leq n, 0 < j \leq n) \tag{4.14}$$

$$\eta_{ij}^2 = \frac{1}{{p_j}^{\mu 1}} \cdot \frac{1}{{d_j}^{\mu 2}} \cdot \frac{1}{{o_j}^{\mu 3}} \cdot \frac{1}{{s_{ij}}^{\mu 4}} \qquad \forall\{i,j\} \in (0 \leq i \leq n, 0 < j \leq n) \tag{4.15}$$

$$P_{ij}^1 = \frac{\tau_{ij} \cdot (\eta_{ij}^1)^{\beta 1}}{\sum_j (\tau_{ij} \cdot (\eta_{ij}^1)^{\beta 1}) + \sum_j (\tau_{i0} \cdot (\eta_{ij}^2)^{\beta 2})} \qquad \forall\{i,j\} \in (0 \leq i \leq n, 0 < j \leq n) \tag{4.16}$$

$$P_{ij}^2 = \frac{\tau_{i0} \cdot (\eta_{ij}^2)^{\beta 2}}{\sum_j (\tau_{ij} \cdot (\eta_{ij}^1)^{\beta 1}) + \sum_j (\tau_{i0} \cdot (\eta_{ij}^2)^{\beta 2})} \qquad \forall\{i,j\} \in (0 \leq i \leq n, 0 < j \leq n) \tag{4.17}$$

O algoritmo 3 define o processo estocástico para a regra de transição do ACO.

Algoritmo 3: Pseudocódigo para a regra de transição do ACO.

```
1 função Regra de Transição
2 para 1 até o total de tarefas existentes faça
3 |   escolha um número aleatório entre 0 e 1
4 |   selecione a próxima tarefa que será sequenciada ou terceirizada de acordo
  |     com as equações 4.16 e 4.17
5 fim
6 executar Opção de Busca Local
7 retornar O_π (lista encadeada com a sequência das tarefas) e
8           S_π (vetor com as tarefas terceirizadas)
9 fim Regra de Transição
```

A função de bonificação da matriz de feromônios artificiais é executada de maneira global e elitista, apenas a melhor solução obtida dentre os agentes computacionais tem seus valores amplificados, de acordo com um parâmetro para a taxa de bonificação $(\tau_{ij} + \rho_d)$. A evaporação é realizada sobre toda a matriz à cada iteração utilizando um parâmetro que define a taxa de evaporação $(\tau_{ij} \cdot \rho_e)$. Um valor máximo (ω_u) e mínimo (ω_l) controlam a amplitude dos dados da matriz de feromônios artificiais.

Adicionalmente, foram implementados métodos de busca local que tem por objetivo refinar a qualidade das respostas construídas pelos agentes. Os métodos de busca local aplicados são baseados em movimentos de remoção, inserção e troca entre os componentes da resposta, tarefas terceirizadas e sequenciadas internamente.

Os algoritmos 4, 5 e 6 definem os métodos para exploração de vizinhança.

Algoritmo 4: Pseudocódigo para o método de inserção.

```
1 função Inserir
2 para cada tarefa terceirizada (se (existirem) faça
3 |   tente inserir a tarefa terceirizada entre as sequenciadas onde, os custos de
  |     produção podem ser tratados sem gerar atrasos
4 fim
5 fim Inserir
```

Algoritmo 5: Pseudocódigo para o método de troca entre posições.

```
1 função Trocar
2 para cada tarefa terceirizada (se (existirem) faça
3 |   tente trocar a tarefa por cada uma das tarefas sequenciadas onde, os custos
  |     de produção e setup da tarefa previamente terceirizada podem ser tradados
  |     com a capacidade interna sem gerar atrasos e o custo de terceirização da
  |     tarefa previamente sequenciada é menor que aquela atualmente terceirizada
4 fim
5 fim Trocar
```

Algoritmo 6: Pseudocódigo da função de busca local gulosa.

```
1 função Iterativa Gananciosa (porcentagem)
2 enquanto porcentagem não é atingida faça
3 |   terceirizar uma tarefa sequenciada
4 fim
5 executar Inserir() (algoritmo 4)
6 fim Iterativa Gananciosa
```

Com estas três funções de busca local foram construídas nove opções para o refinamento das respostas do ACO, descritos na tabela 4.2.

Tabela 4.2: Opções de Busca Local

Valor do Parâmetro	Descrição
0	nenhuma busca local é aplicada
1	Inserir
2	Trocar
3	Iterativa
4	Inserir + Trocar
5	Inserir + Iterativa
6	Trocar + Iterativa
7	Inserir + Trocar + Iterativa
8	Iterativa + Trocar
9	Trocar + Inserir

Por fim, o ACO resultante possui 19 parâmetros, dos quais 16 são configuráveis. Dez deles estão relacionados com a função de transição (definidos por α, β e μ), outros quatro regem as funções da matriz de feromônios (definidos por ω e ρ), além de dois

parâmetros para os procedimentos de busca local (L e I).

Os três parâmetros: número de agentes computacionais, a quantidade de gerações e o critério de parada, não são configuráveis. Estes valores estão ligados às estruturas de repetição do algoritmo e definem o esforço computacional empregado para sua execução.

A escolha elitista para a atualização dos feromônios artificiais permite que um número pequeno de agentes computacionais seja aplicado. Portanto, seguindo uma escolha de projeto com objetivo de minimizar o tempo de execução apenas **cinco** agentes são criados a cada geração. Experimentos preliminares revelaram que versões bem configuradas do ACO convergiam em torno de 250 iterações, ou menos. Portanto, o número máximo para gerações de agentes foi definido como **500** e o critério de parada estabelece que o algoritmo termina caso não encontre melhorias após **50%** do total de iterações.

Os oito parâmetros das funções de visibilidade (α_1, α_2, α_3, α_4, μ_1, μ_2, μ_3 e μ_4) foram definidos em uma escala real entre 0,01 e 4, considerando duas casas decimais. O parâmetro ρ_e – taxa de evaporação – tem domínio entre 0,8 e 0,95 e os parâmetros β_1 e β_2 – que influenciam a regra de transição – são definidos entre 0,9 e 1, todos estes reais considerando duas casas decimais.

Os demais parâmetros são: ω_l – nível inferior da matriz de feromônios artificiais; ω_u – nível superior da matriz; ρ_d – taxa de bonificação. A porcentagem para o método de busca iterativa é definido por I e as opções de busca local são acessadas através de L.

Para o IRACE todos os parâmetros reais são tratados com o mesmo número de casas decimais. Portanto, alguns parâmetros numérico ordinais foram considerados como categóricos com escalas discretas definidas na tabela 4.3.

Para a configuração do domínio das variáveis ajustáveis através do sistema *Darwin Toolkit* (DT) é possível definir parâmetros reais a partir do seu valor máximo e mínimo, mas também especificando o intervalo para a escala real. O sistema *Darwin Toolkit* foi implementado neste projeto e contem uma interface que viabiliza a configuração e

Tabela 4.3: Domínios dos Parâmetros Ajustáveis Categóricos

Parâmetro	Domínio
ω_l	(5, 10, 15)
ω_u	(20, 30, 40)
ρ_d	(1,0; 1,1; 1,2; 1,3; 1,4; 1,5; 1,6; 1,7; 1,8; 1,9; 2,0; 2,1; 2,2; 2,3; 2,4; 2,5; 2,6; 2,7; 2,8; 2,9; 3,0; 3,1; 3,2; 3,3; 3,4; 3,5; 3,6; 3,7; 3,8; 3,9; 4,0)
I	(0,10; 0,15; 0,20; 0,25; 0,30; 0,35; 0,40; 0,45; 0,50)
L	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

aplicação da camada OAA e também a geração e execução de instâncias em bateladas.

4.2 Mecanismos de Validação

O objetivo para a etapa de validação é definir se a estratégia proposta (a camada OAA) atinge resultados semelhantes a um configurador automático de algoritmos considerado na literatura como o estado-da-arte para esse tipo de aplicação, o IRACE.

Os autores em Nannen et al. (2008) mencionam que, para problemas de *scheduling*, é bastante complexo encontrar um conjunto de parâmetros que proporcione resultados satisfatórios para todas as características das instâncias disponíveis. Observando as afirmações de Rabanal et al. (2015) e os teoremas apresentados por Wolpert e Macready (1997), buscou-se realizar uma ampla representação das possibilidades de valores para as variáveis de entrada, construindo um montante estatisticamente significante de exemplares para as instâncias de teste.

Consequentemente, o processo para geração de instâncias visa avaliar o impacto de distintas configurações dos dados de entrada para o desempenho dos algoritmos implementados. Para este propósito, foram projetados 208 diferentes níveis para os casos de teste, usando combinações com uma variedade de amplitudes para distribuições uniformes. Descrevendo os 208 níveis a seguir:

- Ambos os custos de produção e *setup* foram gerados usando distribuições unifor-

mes variando entre [1;10], [1;25], [1;50] e [1;100] (exceto nos casos onde os tempos de *setup* fossem muito maiores que os de processamento). A tabela 4.4 lista as 13 possibilidades para combinações das distribuições de processamento e *setup*.

- Os custos de terceirização consistem em quatro diferentes níveis com distribuições uniformes entre [5;95], [20;70], [35;65] e [50;50]. Observando que quando os custos de terceirização são constantes, o problema é reduzido à minimização do número de tarefas terceirizadas.

- Os prazos de entrega foram amostrados com distribuições uniformes definidas pelo multiplicador $P = \sum_j p_j + s_{ij}$ que simplesmente calcula o tempo para processamento de todas as tarefas mais uma sequência genérica para os tempos de *setup*, considerando FIFO (*First In-First Out*). As distribuições para os quatro níveis das datas de entrega são: [0,2·P;0,5·P], [0,2·P;0,7·P], [0,2·P;0,9·P] e [0,2·P;1,1·P]. Estas distribuições consideram instâncias com a possibilidade de existir capacidade para demanda requisitada (que resultarão em nenhum custo de terceirização), mas também casos onde será necessária a subcontratação para garantir as restrições impostas.

Para casos de pequeno porte foram avaliadas 18720 instâncias e 4160 para casos de maior porte, divididos em 10 conjuntos, sempre contendo múltiplos exemplos dos 208 níveis de problemas descritos. A distinção feita entre pequeno e grande porte é definida pelo tamanho das instâncias em que o algoritmo CPLEX foi capaz de comprovar respostas ótimas em um tempo significativo, apenas até 12 tarefas. A tabela 4.5 sumariza os conjuntos de instâncias criados.

Distinguindo os experimentos computacionais conduzidos pelo número de tarefas e pelas relações entre distribuições para as variáveis das instâncias. Algumas observações são importantes. Primeiro, os casos de teste exploram uma ampla variedade de níveis e tamanhos para as instâncias, o que pode complicar o processo de parametrização do

Tabela 4.4: Distribuições para Tempos de Processamento e Preparação

Tempos de Processamento	***Tempos de Setup***
$[1; 100]$	$[1; 10]$
$[1; 100]$	$[1; 25]$
$[1; 100]$	$[1; 50]$
$[1; 100]$	$[1; 100]$
$[1; 50]$	$[1; 10]$
$[1; 50]$	$[1; 25]$
$[1; 50]$	$[1; 50]$
$[1; 50]$	$[1; 100]$
$[1; 25]$	$[1; 10]$
$[1; 25]$	$[1; 25]$
$[1; 25]$	$[1; 50]$
$[1; 10]$	$[1; 10]$
$[1; 10]$	$[1; 25]$

algoritmo. Outro ponto interessante, é complexidade apresentada pelos oito parâmetros definidos em escalas reais, contando 400^8 possibilidades de configuração, implicando que um experimento manualmente projetado é virtualmente impossível.

O IRACE necessita de três entradas: (i) uma implementação de meta-heurística, (ii) um conjunto de instâncias e (iii) a definição do domínio dos parâmetros configuráveis. O programa executa um único experimento com cada candidato para o vetor de parâmetros. Aplicando testes estatísticos, o IRACE avalia os melhores candidatos para solucionar as instâncias disponíveis. Assim, quando existem múltiplos casos de treinamento o vetor de parâmetros resultante será aquele com melhores resultados para todo o grupo de instâncias (López-Ibáñez et al., 2011).

Experimentos realizados anteriormente, em Frascati (2014), revelaram que escolher aleatoriamente uma única instância, ou apenas algumas, para o processo de afinação do IRACE, origina vetores com parâmetros mal configurados para os demais casos. Portanto, todos os experimentos de parametrização utilizaram as primeiras 208 instâncias de cada conjunto, definido na tabela 4.5, contendo um exemplar para cada nível de problema para os tamanhos gerados.

Tabela 4.5: Geração de Instâncias

Número de Tarefas	**Total de Casos**
10	6240
11	6240
12	3120
10, 11 ou 12	3120
20	1040
40	1040
60	624
80	416
100	416
20, 40, 60, 80 ou 100	624

Para a configuração do IRACE, executar um pequeno número de iterações pode terminar o processo de afinação prematuramente, sem encontrar os candidatos elite existentes. Ao contrário, executar demasiadas iterações pode tornar os candidatos restantes muito especializados em características de instâncias específicas (López-Ibáñez et al., 2016).

Foram executados três experimentos de afinação para cada conjunto de instâncias, com diferente número de iterações para o IRACE: 1000, 5000 e 10000. Para cada processo de afinação o melhor candidato obtido foi selecionado, resultando em 30 melhores vetores de parâmetros para a operação do ACO. Assim, a avaliação conduzida aqui é feita apenas sobre as configurações elite apresentadas pelo IRACE e não a partir de todo o espaço amostral de possibilidades para o algoritmo configurável.

Para validação do ACO em instâncias de pequeno porte, foi implementado um modelo MIP em Python 3.4 utilizando a biblioteca CPLEX. Além disso, foi implementada uma heurística construtiva para solucionar casos de grande porte baseada em outros problemas de *scheduling* observando a literatura para casos similares. Neste caso, o trabalho de Nawaz et al. (1983) oferece uma heurística chamada NEH (algoritmo 7), de acordo com as iniciais de seus autores, que se assemelha à formulação matemática aqui apresentada.

Algoritmo 7: Pseudocódigo da Heurística NEH. Adaptado de Nawaz et al. (1983)

Algoritmo NEH
para cada *ordem* j **faça**
  $S_j = \sum_{i=1}^{m} p_{ij}, \bigvee j \in J$
fim
Ordene as tarefas em ordem decrescente para S_j formando a sequência $j_1, j_2, ..., j_n$
Selecione as duas primeiras tarefas (j_1 e j_2) e ordene-as de forma a obter o menor *makespan*
para cada *tarefa restante* $(j_3, ..., j_n)$ **faça**
  Fixe a ordem das tarefas previamente sequenciadas e insira a tarefa na posição da sequência que resulta no menor *makespan* parcial possível entre todas as possibilidades de posições para inserção
fim
fim Algoritmo NEH

Esta heurística foi adaptada para o caso tradado, como mostrado no algoritmo 8. Sabendo que métodos de busca local são capazes de oferecer melhorias a qualidade de resposta para outros algoritmos de busca. Os mesmo métodos utilizados em conjunto com o ACO, foram aplicados para a definição de uma rotina capaz de potencializar os resultados do algoritmo 8. O algoritmo 9 detalha um procedimento que executa a heurística NEH-T e finitas vezes os métodos para exploração de vizinhança.

Algoritmo 8: Pseudocódigo para heurística NEH-T.

```
1  Algoritmo NEH-T
2  Ordene as tarefas na ordem crescente em relação à d_j formando a sequência
     j_1, j_2, ..., j_n
3  Selecione as duas primeiras tarefas (j_1 e j_2) e ordene-as de forma a obter o
     menor makespan, mas garantindo que os prazos de entrega sejam atendidos
4  se os prazos para entrega de j_1 e/ou j_2 não foram atendidos então
5  |  terceirize j_1 e/ou j_2 buscando o menor custo total de terceirização possível
   |    para que os prazos de entrega sejam cumpridos
6  fim
7  para as tarefas restantes (j_3, ..., j_n) faça
8  |  Fixe a ordem das tarefas previamente sequenciadas
9  |  se houver posições na sequência para inserir a tarefa que não gerem atrasos
   |    então
10 |  |  Insira a tarefa na posição que resulta no menor makespan parcial possível
11 |  fim
12 |  senão
13 |  |  terceirize a tarefa
14 |  fim
15 fim
16 fim Algoritmo NEH-T
```

Algoritmo 9: Rotina combinando heurística e buscas locais

```
1  Rotina NEH-T+SLS
2  Resolver problema com o Algoritmo NEH-T
3  iterações = 5; incremento = 5; incrementoporcentagem = 5%;
4  máximaporcentagem = 50%; mínimaporcentagem = 10%;
5  executar Inserir()
6  executar Trocar()
7  porcentagematual = máximaporcentagem
8  enquanto porcentagematual >= mínimaporcentagem faça
9  |  para iterações faça
10 |  |  executar Iterar(porcentagematual)
11 |  |  executar Trocar()
12 |  fim
13 |  iterações = iterações + incremento
14 |  porcentagematual = porcentagematual - incrementoporcentagem
15 fim
16 fim NEH-T+SLS
```

Neste ponto é importante destacar que os componentes descritos nesta seção e na anterior não são considerados como contribuições inéditas desta pesquisa, apenas fazem parte do ferramental necessário para o escopo do projeto.

Para a comparação entre os algoritmos para as instâncias de pequeno porte, as respostas ótimas foram obtidas através de um modelo matemático. No entanto, para os demais casos as respostas ótimas são desconhecidas. Assim, a comparação entre os algoritmos, para as instâncias de maior porte, pode resultar em vitórias e derrotas quando diferentes configurações do ACO são comparadas entre si, ou com o algoritmo NEH-T+SLS.

Portanto, para as instâncias de pequeno porte o interesse está na capacidade dos algoritmos não exatos em obter respostas ótimas de maneira veloz e no *gap* observado para as respostas não ótimas. Enquanto para os casos de médio e grande porte serão avaliados as distâncias entre vitórias e derrotas dos algoritmos observados.

Os resultados obtidos mostram que diferentes vetores de parâmetros apresentam comportamentos distintos para as instâncias testadas. Mesmo com o emprego de softwares para configuração automática de algoritmos, existe a necessidade do projeto de experimentos para avaliar os resultados da parametrização automática e deve ser tomada de decisão sobre quais parâmetros aplicar para instâncias específicas. Casos heterogêneos impõem múltiplas configurações para solucionar diferentes instâncias. Surge a necessidade de decidir qual configuração é mais adequada a uma nova instância do problema de otimização.

Estas questões, o projeto do experimento para avaliação das parametrizações e a tomada de decisão sobre a melhor configuração, são solucionadas pela da camada OAA de maneira inteligente.

4.3 Desenvolvimento da Camada OAA

Este projeto oferece a proposta de um sistema capaz de buscar configurações especialistas para instâncias específicas classificando-as em grupos. Esta metodologia segue conceitos apresentados por Wolpert e Macready (1997), que comprovam matematicamente a existência de instâncias heterogêneas para aplicações com algoritmos de busca estocásticos, e por Smit e Eiben (2009), que apresentam os conceitos de configurações "generalistas" e "especialistas".

Surgindo uma nova instância, as configurações previamente armazenadas são avaliadas e um grau de similaridade com a nova instância é calculado. Comprovada semelhança entre execuções do passado, não existe a necessidade de reconfiguração do algoritmo otimizador.

A figura 4.1 ilustra os componentes que compõem o problema de otimização autônoma aqui estudado. Notam-se as três camadas apresentadas por Smit e Eiben (2009): aplicação, algoritmo e projeto (ver figura 3.1). Além delas, a camada OAA inclui um algoritmo de controle um outro para aprendizado.

Três blocos dependem de características específicas dos modelos sendo otimizados: (i) o algoritmo de controle; (ii) o próprio modelo de otimização e; (iii) o algoritmo evolutivo otimizador. O mecanismo de controle deve ser desenvolvido de acordo com o problema tratado e o algoritmo de otimização pode conter customizações específicas ao modelo do problema sendo otimizado, a formulação contendo as variáveis e restrições necessárias é definida na camada da aplicação. Estes blocos podem ser substituídos para trabalhar com outro problema de otimização. Os outros componentes da camada OAA não possuem relação com as especificidades do modelo matemático sendo otimizado e podem ser aplicados genericamente.

A interpretação deste cenário mostra que o subproblema da otimização para um modelo matemático qualquer (neste caso um problema de *scheduling*) depende da para-

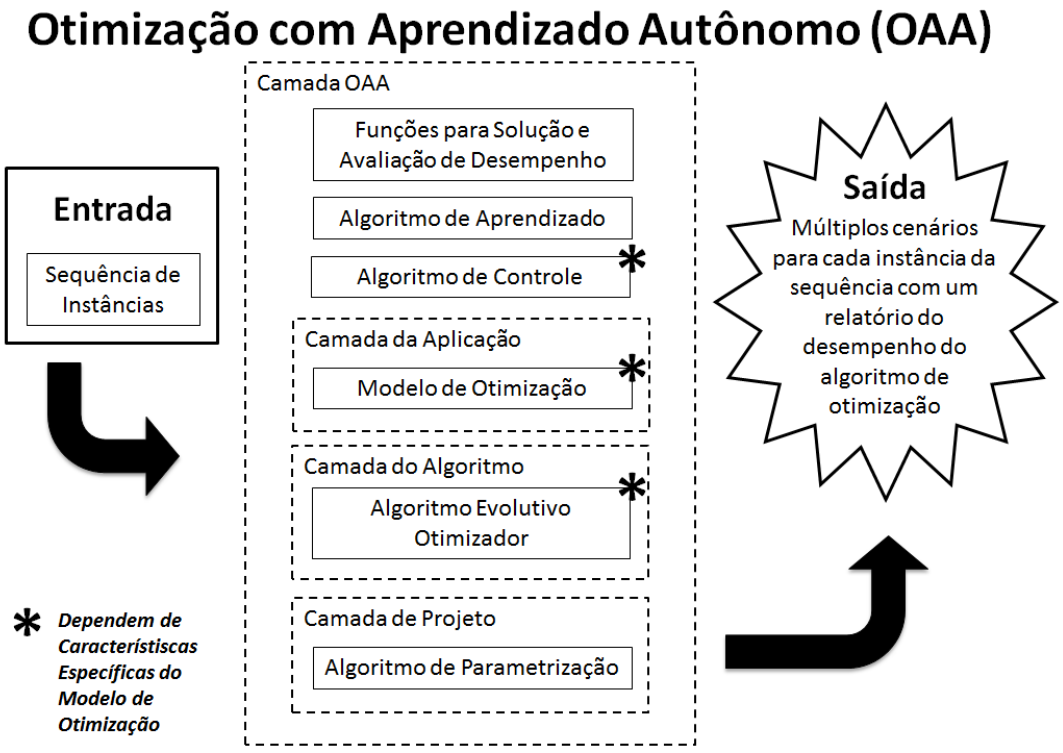


Figura 4.1: Arquitetura do sistema em camadas.

metrização do AE aplicado e ainda da decisão sobre a qualidade esperada para a solução fornecida pelo algoritmo, avaliada aqui com uma resposta de controle. Solucionando estes componentes de forma integrada obtem-se uma ferramenta para otimização capaz de lidar com instâncias heterogêneas ao longo da vida útil do otimizador, o que elimina a necessidade de configuração prévia ou mesmo para manutenção já que agrega essa funcionalidade de maneira inteligente.

A camada de aplicação e a camada do algoritmo foram descritas anteriormente e são definidas respectivamente pelo modelo de programação da produção e pela meta-heurística configurável (ACO). O bloco de controle será definido pelo algoritmo NEH-T+SLS (ver algoritmo 9).

Em seguida serão abordados os aspectos relativos à Razão de Desempenho (RD), definida para as funções de controle do sistema. Também serão descritos os algoritmos de aprendizado e parametrização, além das funções de solução e avaliação de desempenho das respostas finais.

4.3.1 Razão de Desempenho (RD)

Para avaliar os resultados do algoritmo configurável em relação a uma medida comparativa de controle será utilizada uma Razão de Desempenho, definida de acordo com a equação 4.18 para problemas de minimização com um único objetivo. A medida RD é baseada na resposta de controle obtida através outro método veloz, como o algoritmo NEH-T+SLS. Assim, RD é uma razão que compara a resposta de controle com a do otimizador.

$$RD = arredondar\left(\frac{R.Controle - R.Otimizador}{R.Controle + \epsilon}\right) \tag{4.18}$$

Em alguns pontos do sistema serão aceitas apenas soluções do otimizador superiores a resposta de controle utilizada para comparação, portanto, RD será positivo. No

entanto, RD negativo também é possível quando a resposta de controle for melhor que a do otimizador. O resultado negativo para a razão de desempenho traduz o mal funcionamento para o algoritmo otimizador quando aplicado um determinado par contendo (vetor de parâmetros, instância do problema).

Obtendo valor de RD próximo a zero (0) significa que o algoritmo é pouco eficiente em superar a resposta de controle, já um resultado próximo a um (1) significa que o algoritmo de controle aplicado é pouco eficiente para a instância tratada. Casos particulares devem ser tratados, pois é possível que para um problema de minimização a regra construtiva forneça resposta igual a zero. Enquanto, para o problema de maximização a resposta do otimizador eventualmente pode ser igual a zero, neste caso, a minimização do problema oferecer resultado igual a zero se traduz no total atendimento da demanda sem necessidade de terceirização. Dessa forma, uma função de arredondamento e uma constante (ϵ) igual a um valor que tende à mas é diferente de zero, evita a divisão nula e mantém o resultado significativo para RD.

O valor de RD para uma configuração específica do algoritmo mede a relação de desempenho (função utilidade) para um vetor de parâmetros (θ) comparado com a resposta obtida por outro algoritmo para controle de qualidade. Avaliando uma família de vetores candidatos (Θ), com θ vetores de parâmetros (nomeados θ_1, θ_2, ..., θ_n) para um algoritmo configurável definimos $E(RD)$ como a média dos valores de RD para as n configurações de uma família e $\sigma(RD)$ o desvio padrão calculado para esta amostra.

Para testar também a semente aleatória que determina os processos estocásticos do algoritmo otimizador, repetições sobre um mesmo par (vetor de parâmetros, instância do problema) oferecem a oportunidade de avaliar a estabilidade do algoritmo. Assim, $\sigma(RD)$ pode representar o desvio padrão calculado a partir dos resultados de RD em repetições da execução do otimizador com o mesmo par (vetor de parâmetros, instância do problema).

$E(RD)$ representa o valor esperado para a razão de desempenho dada uma de-

terminada família de vetores e $\sigma(RD)$ é o desvio padrão dessa variável aleatória. O domínio de RD é definido no intervalo que vai de menos infinito, quando a resposta do otimizador é infinitamente pior que a da regra de controle, até máximo valor um (1), quando o contrário acontece.

Quanto ao resultado da estabilidade de cada família, avaliado a partir de $\sigma(RD)$, espera-se que esse valor seja pequeno demostrando que um mesmo vetor quando testado por vezes sucessivas em uma mesma instância apresenta comportamento com baixa variância e, portanto, o algoritmo é estável para esse par (vetor de parâmetros, instância do problema).

A falha da família em resolver a nova instância pode ser interpretada com a avaliação de $RD < 0$, ou seja, famílias de vetores que apresentarem resultados negativos em RD para uma nova instância serão menos indicadas à sua solução.

As variáveis $E(RD)$ e $\sigma(RD)$, bem como a avaliação de valores negativos para RD em cada par (vetor de parâmetros, instância do problema) para uma determinada família de vetores, fornecerão dados para a interpretação das medidas de desempenho para o algoritmo configurável enquanto esse resolve uma instância inédita.

Deseja-se que os membros em uma família de vetores possuam altos resultados para RD e também com um baixo desvio entre eles, resultando em alto $E(RD)$ e baixo $\sigma(RD)$, mas ainda com baixos resultados para $\sigma(RD)$ enquanto avaliando a estabilidade de um único vetor de parâmetros.

4.3.2 Algoritmo de Parametrização

Para se obter um bom algoritmo de parametrização, é necessário encontrar vetores de parâmetros com alto desempenho para a função utilidade, ou seja, indivíduos elite para configurar o algoritmo otimizador, mas também minimizar o custo computacional do processo definindo um algoritmo que vise diminuir os esforços descritos pelas variáveis $A \cdot B \cdot C$ que correspondem as iterações necessárias para parametrizar um algoritmo

configurável.

Foi implementado um meta-AE com objetivo definir uma família de vetores com valores para os parâmetros do algoritmo otimizador que ofereçam alto desempenho para uma determinada instância. Foi implementado um meta-ACO que cria agentes computacionais para explorar o espaço de busca armazenando suas informações em uma matriz de feromônios. Um processo iterativo guia os agentes convergindo para soluções com alto desempenho. O algoritmo 10 define a solução implementada.

Foi definida uma matriz de feromônios artificias para o meta-ACO como Ω_{ij}, que representa cada parâmetro configurável do algoritmo otimizador (i) e os possíveis valores associados em seu domínio dispostos em j. Essa matriz deverá ser inicializada com um valor máximo para todos os pontos e limitada também por um valor mínimo.

A seleção de um novo conjunto de parâmetros corresponde a regra de transição do meta-ACO. Para cada parâmetro i, uma função densidade de probabilidade é gerada a partir dos valores contidos na matriz de feromônios e a cada iteração o agente computacional deve escolher aleatoriamente os valores para cada parâmetro.

A função de visibilidade foi considerada constante, observando que para a presente implementação as oportunidades de configuração do algoritmo parametrizador devem ser mínimas senão nulas, pois é importante que o sistema de parametrização seja robusto e opere de maneira autônoma. No entanto, é possível que existam formas de obter melhores pesos para as escolhas de valores dos parâmetros aumentando a velocidade de convergência desse algoritmo, mas isso não será analisado neste momento.

Algoritmo 10: Pseudocódigo do meta-AE (ACO) para parametrização.

```
1  algoritmo Parametrizar (algoritmo, instância) – meta-ACO
2  Obter o limitante de controle para a nova instância
3  Criar uma matriz de feromônios artificiais, com taxas de evaporação global,
     evaporação local, bonificação, nível mínimo e nível máximo para o problema de
     parametrização do algoritmo otimizador
4  repita
5  |  Criar um agente computacional sorteando valores para os parâmetros
   |    baseando-se na matriz de feromônios
6  |  se a resposta for igual ou superior a da regra de controle então
7  |  |  Armazenar o vetor, executar a evaporação global e bonificar as posições
   |  |    do vetor de parâmetros sorteado
8  |  fim
9  |  senão
10 |  |  Evaporar apenas as posições do vetor sorteado
11 |  fim
12 |  se o mínimo de vetores foi armazenado MAS a restrição de desvio não foi
   |    atendida então
13 |  |  Eliminar o pior vetor armazenado
14 |  fim
15 até o máximo de iterações OU ter armazenado o mínimo de vetores;
16 se o máximo de iterações foi atingido e o mínimo de vetores não foi armazenado
     então
17 |  se nenhum vetor foi armazenado então
18 |  |  Criar família de vetores padrão
19 |  fim
20 |  senão
21 |  |  Copiar os vetores existentes até o mínimo necessário
22 |  fim
23 |  Apresentar aviso de erro no processo de parametrização
24 fim
25 Retornar as melhores respostas obtidas
```

Definindo o número de parâmetros ajustáveis para um algoritmo como n, e sendo m_i o número de posições existentes para os possíveis valores do domínio de um determinado parâmetro. A equação 4.19 determina a função de probabilidade para a escolha do valor para cada um dos parâmetros ajustáveis de um algoritmo.

$$P_{ij} = \frac{\Omega_{ij}}{\sum_j \Omega_{ij}} \qquad \forall\{i,j\} \in (0 < i \leq n, 0 < j \leq m_i) \tag{4.19}$$

Após a inicialização da matriz de feromônios, começa o processo da geração de agentes computacionais para avaliar o espaço de busca que define a função utilidade. Em cada iteração um agente será criado e avaliado em relação a resposta de controle.

O processo de evaporação e deposição foi adaptado para reagir aos resultados da variável RD, que é a medida de avaliação do algoritmo otimizador quando comparado com a opção de controle. Dessa forma, agentes que retornarem resultado com $RD > 0$ terão as posições do vetor θ sorteado na iteração bonificados na matriz de feromônios. A evaporação local será determinada pelos agentes que retornarem valores negativos para a razão de desempenho ($RD < 0$) e a evaporação global ocorre quando um vetor de parâmetros é armazenado ($RD > 0$). Dessa forma a cada iteração os resultados ruins para a utilidade são desencorajados diminuindo os valores das probabilidades, enquanto resultados bons são reforçados.

Para parâmetros contínuos, tanto evaporação como deposição de feromônios serão realizadas a partir de distribuições normais centradas nos valores sorteados e com desvio padrão proporcional a precisão do respectivo domínio. Enquanto que para os parâmetros categóricos essas funções afetarão apenas o ponto único que os define na matriz de feromônios.

Segundo a equação 4.19, cada iteração do algoritmo pode ser interpretada como a seleção de um valor (j) para cada um dos parâmetros (i), seguido da execução do

algoritmo otimizador com tal conjunto de valores para avaliação da função objetivo do problema de otimização.

Durante o processo de parametrização, agentes com resultados positivos serão inseridos em uma família de vetores (Θ), com θ_n vetores, onde n representa o índice do vetor e deve conter ao menos o mínimo de vetores definidos para a família.

Os vetores restantes devem ser "especialistas" para a instância avaliada. Essa especialização de uma família será determinada por um processo iterativo que remove o vetor com pior resultado e ativa o meta-ACO para criar um novo vetor que deverá possuir desempenho superior ao vetor removido, proporcionando a convergência do processo.

Este processo iterativo pode ser controlado estabelecendo-se um número máximo de iterações, mas também com uma restrição para o desvio entre a utilidade dos vetores de uma família ($\sigma(RD)$). O usuário definirá o valor de $\sigma(RD)$ para controlar o desempenho do processo de parametrização. Estabelecendo valores pequenos para $\sigma(RD)$, o processo iterativo que remove o vetor com pior desempenho forçará que os novos vetores concentrem-se em torno da média, que aumenta a cada iteração com a remoção do pior indivíduo. Enquanto valores maiores para $\sigma(RD)$ devem permitir que o processo de parametrização termine em um número menor de iterações, mas diminuindo a precisão do processo, significando que a família será menos precisa quanto à sua especialização para uma determinada instância.

O número máximo de iterações para o meta-ACO deve evitar que o processo não termine. No entanto, a finalização do prematura da configuração dos parâmetros ajustáveis pode acarretar alguns problemas. O algoritmo de parametrização descreve duas possibilidades de falha para o processo: i) o número de iterações foi executado mas não foi possível armazenar o mínimo de vetores desejados e ii) nenhum vetor foi armazenado em todas as iterações disponíveis.

Para contornar a primeira falha, mais simples, os vetores armazenados foram copiados para preencher o mínimo definido. A segunda falha é mais complexa e representa

que um indício de que o algoritmo otimizador aplicado não pode ser configurado para as características da instância tratada. De maneira simplificada, este ponto foi contornado criando uma família de vetores com dados padrões, que não necessariamente se adequam a nova instância. No próximo capítulo serão tratados com mais detalhes as ocorrências e peculiaridades destas possíveis falhas no processo de parametrização.

Durante a criação de agentes computacionais podem futuramente ser adicionados procedimentos de busca local com objetivo de explorar as vizinhanças das soluções criadas em busca de vetores que ofereçam melhor utilidade para a instância avaliada, potencializando a velocidade de convergência desse processo.

O meta-ACO aqui proposto cria vetores inteligentemente a partir de um processo baseado na meta-heurística ACO, restrição principal observada em relação a variável A (que determina o número de vetores avaliados para a parametrização em $A \cdot B \cdot C$). Este meta-ACO tem o número de iterações minimizado quando o processo for terminado porque a família contem o o mínimo de vetores e atende a restrição imposta para $\sigma(RD)$, ambos especificados pelo usuário.

Outra distinção deste meta-ACO é a aplicação de uma função que aceita ou rejeita novos vetores. O algoritmo 10 mostra a comparação dos resultados da utilidade dos vetores com um limitante. Destaca-se que a heurística aplicada para controle influenciará o desempenho do sistema. Caso o método empregado ofereça resultados ruins o algoritmo otimizador pode ser mal parametrizado.

Como o algoritmo de parametrização aqui proposto destina-se apenas a configuração de vetores "especialistas" e trata sempre uma única instância dentro de uma sequência, não existem múltiplas instâncias para avaliar os vetores de parâmetros e descartá-los com o mínimo de avaliações possível. ACs destinam-se a analisar o desempenho dos vetores sobre um conjunto de instâncias, descartando conjuntos de parâmetros significativamente inferiores com o menor número de avaliações possível, a partir de um teste estatístico para comparação não pareada.

Entende-se que métodos de corrida visam minimizar o esforço da parametrização para a obtenção de configurações "generalistas". O custo computacional do processo é minimizado uma vez que configurações incapazes de oferecer bom desempenho em algumas das instâncias do conjunto são descartadas, não necessitando serem avaliados em todas as instâncias disponíveis.

Todavia, a criação de configurações "especialistas" aliada a classificação de novas instâncias às famílias de vetores previamente construídas confere características "generalistas" à família na medida em que casos inéditos são solucionados por seus vetores.

Nesta implementação, B (que determina o número de instâncias avaliadas para cada vetor de parâmetros em $A \cdot B \cdot C$) não é considerado, uma vez que o processo de parametrização não é executado para múltiplas instâncias. No entanto, uma vez que o algoritmo visa minimizar a necessidade de reconfiguração, executando a parametrização para o mínimo de instâncias possíveis, a variável B também é tratada, mas de maneira distinta as pesquisas anteriores.

Quanto a variável C, o ACO da camada de aplicação contém uma restrição que termina sua execução antes de seu número máximo de iterações, mas a velocidade do algoritmo não foi tratada como um variável sendo otimizada da mesma forma que em Ugolotti e Cagnoni (2014).

De acordo com o número máximo de iterações definido para o ACO na camada do algoritmo, neste caso 500, uma porcentagem do máximo de iterações, por exemplo 50%, deve definir o término das avaliações caso em 250 iterações consecutivas não houver melhoria na resposta obtida pela meta-heurística. Experimentos preliminares determinaram valores razoáveis para esta restrição de forma que o custo computacional seja minimizado sem perdas na precisão deste ACO na camada do algoritmo.

Implementações futuras podem ampliar o escopo deste algoritmo de parametrização, para que ele possa tratar parâmetros que definem a velocidade do método na camada do algoritmo, com uma estrutura multi objetivo visando maximizar a utilidade e minimizar

o número de iterações.

O resultado oferecido por este algoritmo de parametrização é uma família com vetores de parâmetros para o algoritmo configurável que lhe confere alto desempenho sobre uma instância específica. O próximo tópico define as características do algoritmo de aprendizado que contêm componentes para classificação de famílias e ranqueamento de vetores. Tais processos servem para detecção de instâncias heterogêneas e para automatizar a decisão sobre quais os melhores vetores de parâmetros para solucionar uma nova instância, podendo também determinar a execução da parametrização do algoritmo configurável.

4.3.3 Algoritmos para Aprendizado Autônomo

O algoritmo de aprendizado proposto neste trabalho é composto por dois processos principais: (i) um processo responsável por classificar novas instâncias como heterogêneas ou homogêneas à alguma família de vetores existente, determinando se há necessidade da parametrização do algoritmo otimizador para a nova instância e; (ii) um processo definido para eliminar vetores excedentes quando novos são incluídos em famílias já existentes. Os processos para classificação e ranqueamento são baseados em conceitos básicos sobre aprendizado de máquina, eles são similares à algoritmos que operam com a lógica de reforço iterativo, como por exemplo, o ACO.

O algoritmo 11 descreve a estrutura implementada para o aprendizado autônomo de configurações para o ACO aplicado aqui para a otimização de uma função objetivo de *scheduling*.

Algoritmo 11: Pseudocódigo para a camada OAA

```
1  algoritmo Otimização com Aprendizado Autônomo (OAA)
2  para cada nova instância do problema faça
3  |  Classificar (famílias de vetores)
4  |  se nenhuma família foi aprovada então
5  |  |  Parametrizar(algoritmo, nova instância)
6  |  |  se o limite de famílias foi atingido então
7  |  |  |  adicionar os novos vetores ao número de famílias mais próximas
8  |  |  fim
9  |  fim
10 |  Ranquear (vetores da família melhor classificada)
11 |  Resolver (nova instância)
12 fim
```

O processo de parametrização é invocado apenas quando não existem vetores de parâmetros conhecidos que são capazes de fornecer respostas com alta qualidade para uma nova instância do problema.

Apresentamos três maneiras para avaliar uma família de vetores: $E(RD)$, $\sigma(RD)$ e a análise da existência de vetores com RD negativo. Durante o processo de parametrização forçamos valores positivos para RD em relação a instância aplicada. Portanto, espera-se que uma família de vetores apresente resultados positivos em RD para uma nova instância, se essa for homogênea àquela utilizada para a criação da família.

Dessa forma, a classificação consiste em aceitar ou rejeitar famílias para resolver as novas instâncias. Para a classificação da nova instância, uma avaliação positiva para $E(RD)$ caracteriza que a família conhecida para a operação do algoritmo otimizador é adequada para a solução da nova instância, ou em outras palavras, que a informação armazenada pelos θ_n vetores é válida para as características do novo problema combinatório, sendo esse homogêneo ao avaliado durante a etapa de parametrização, interpretando que os indivíduos de Θ carregam a informação necessária para solucionar a nova instância.

Já uma avaliação negativa para $E(RD)$ determina que a população conhecida

não é capaz de resolver a nova instância, sendo essa heterogênea, e uma nova etapa de parametrização deve ser determinada para a nova instância. Classificar o novo caso como heterogêneo significa que uma nova população de indivíduos elite deve ser criada para as novas características da instância do problema combinatório. Mantendo-se a população previamente conhecida (Θ_1) relacionada a uma classe de instâncias anteriormente avaliadas, cria-se uma nova população Θ_2 para uma nova classe de instâncias e repete-se o processo de parametrização.

A partir do primeiro problema heterogêneo o sistema de parametrização passará a operar com mais de uma população (Θ_m) para a execução do algoritmo otimizador. Nesse momento, havendo diferentes grupos de problemas heterogêneos, uma nova instância deverá ser classificada como pertencente a uma determinada classe (m). O procedimento é repetido para cada uma das famílias existentes, sendo determinada aquela com maior valor de $E(RD)$ como a mais adequada para solucionar a nova instância.

Uma opção binária pode oferece ao usuário aceitar famílias com uma determinada porcentagem de resultados negativos para RD. O algoritmo 12 descreve o processo de classificação para famílias de vetores.

Algoritmo 12: Pseudocódigo para o processo de classificação.

```
1 algoritmo Classificar (famílias de vetores)
2 Obter o limitante de controle para a nova instância
3 para cada família de vetores faça
4 |   Executar o AE com cada vetor de parâmetros da família
5 |   Calcular E(RD) e σ(RD)
6 |   Aprovar famílias que apresentarem porcentagem de aprovação (%) vetores
  |     com resultado igual ou melhor que o valor de controle disponível
7 fim
8 Ranquear famílias de acordo com E(RD), σ(RD) e a porcentagem de vetores
    aprovados
```

O algoritmo de parametrização retorna um número mínimo de vetores com valores para os parâmetros ajustáveis. No entanto, se o limite de famílias for atingido, estes vetores devem ser inseridos em outras famílias já existentes. Aumentando a quantidade

de vetores em uma família, o processo de ranqueamento é responsável por eliminar o excesso de vetores. O algoritmo 13 demonstra a implementação do método para avaliação da estabilidade dos vetores e responsável pelo descarte de informações.

Algoritmo 13: Pseudocódigo do método para ranqueamento de vetores.

```
1 algoritmo Ranquear (vetores de parâmetros)
2 para cada vetor de parâmetros faça
3 |   repita
4 |   |   Executar o AE com este vetor
5 |   até o número de avaliações de estabilidade;
6 fim
7 Ranquear vetores de acordo com E(RD) e σ(RD), eliminando os piores
    resultados que excedem o máximo de vetores por família
```

Uma rotina que realiza sucessivas amostras com os melhores vetores ranqueados de uma família foi desenvolvida para coletar as melhores respostas distintas obtidas, oferecendo múltiplos cenários para a programação das tarefas de uma nova instância do problema. O algoritmo 14 define a solução de um novo caso, uma vez que são conhecidos vetores que fornecerão respostas de alta qualidade para esta instância.

Algoritmo 14: Pseudocódigo da rotina de solução que armazena múltiplos cenários.

```
1 algoritmo Resolver (nova instância)
2 para cada um dos melhores vetores de parâmetros começando pelo menos
    ranqueado faça
3 |   repita
4 |   |   Executar o AE com este vetor
5 |   |   Adicionar o incremento ao número de avaliações
6 |   até o número de avaliações para o vetor;
7 fim
8 Retornar as melhores respostas obtidas
```

A velocidade dos processos definidos pelos algoritmos que compõem a camada OAA pode ser controlada pelas variáveis que aparecem sublinhadas nos algoritmos descritos. A seguir são detalhados os modos de operação para a camada OAA, contendo valores para as variáveis que definem sua velocidade.

4.3.4 Modos de Operação para Camada OAA

Para a primeira avaliação desta implementação, dois modos de operação foram aplicados para a camada OAA. A tabela 4.6 descreve os valores para as variáveis sublinhadas nos algoritmos 10 a 14.

Tabela 4.6: Modos de Operação para Camada OAA

Variáveis	**Rápido**	**Padrão**
evaporação global	0,975	0,975
evaporação local	2,5	2,5
bonificação	5	5
nível mínimo	10	10
nível máximo	100	100
máximo de iterações	1000	1000
mínimo de vetores	5	10
restrição de desvio	0,10	0,15
número de famílias	5	10
porcentagem de aprovação	0,8	0,9
avaliações de estabilidade	2	3
máximo de vetores	15	30
melhores vetores	3	4
avaliações	3	3
incremento	3	3
melhores respostas	1	1

A distinção dos modos de operação se dá principalmente pelo tamanho das famílias de vetores e sua quantidade, mas também pela precisão dos processos de parametrização e classificação.

Além disso, para oferecer uma interface de configuração para a camada OAA, mas também a execução de instâncias em bateladas a partir dos algoritmos implementados neste projeto para avaliação, foi implementado o software chamado *Darwin Toolkit* (DT), ilustrado pela figura 4.2.

A figura 4.3 mostra as possibilidades de configuração para o sistema *Darwin Tookit*. Nele é possível definir os algoritmos que operam nas camadas internas à OAA, assim

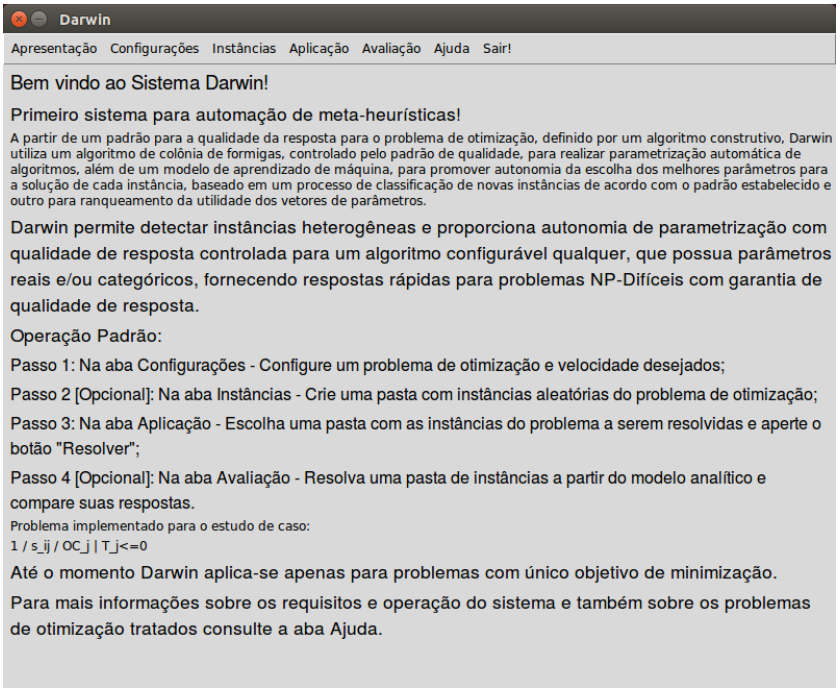


Figura 4.2: Aba de apresentação do sistema *Darwin Toolkit*.

como as variáveis que definem a velocidade da camada, e ainda configurar opções para o algoritmo otimizador configurável, por exemplo, definir o domínio das variáveis ajustáveis.

Adicionalmente, foi implementado um módulo para a geração de instâncias aleatórias, capaz de gerar blocos com casos de teste com diferentes opções. A figura 4.4 detalha o módulo que cria instâncias utilizando um gerador criado para os 208 níveis de problemas descritos anteriormente.

Na figura 4.5 encontra-se a aba que possibilita executar uma sequência de instâncias utilizando a camada OAA. Enquanto, a 4.6 ilustra o módulo capaz de executar sequências com casos de teste a partir dos algoritmos singulares implementados neste projeto.

O próximo capítulo detalha os resultados obtidos para comparação da camada OAA com os demais algoritmos implementados para sua avaliação.

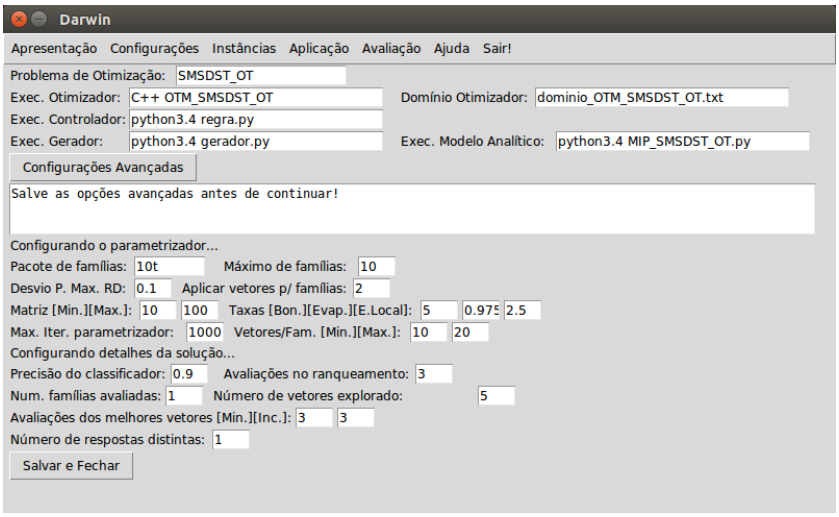


Figura 4.3: Aba de configuração para o software *Darwin Toolkit*.

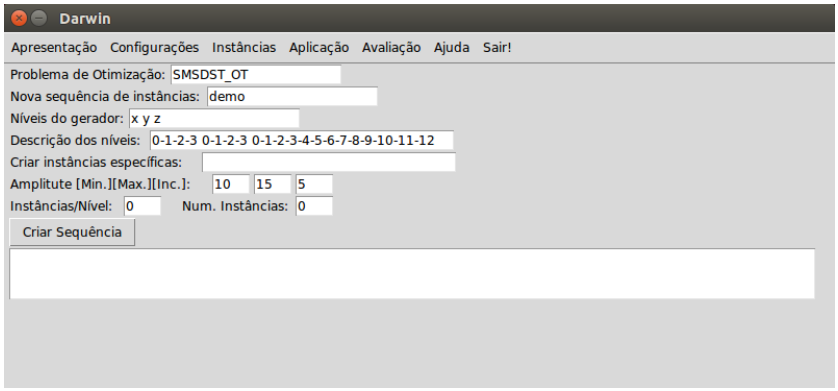


Figura 4.4: Módulo para geração de instâncias aleatórias.

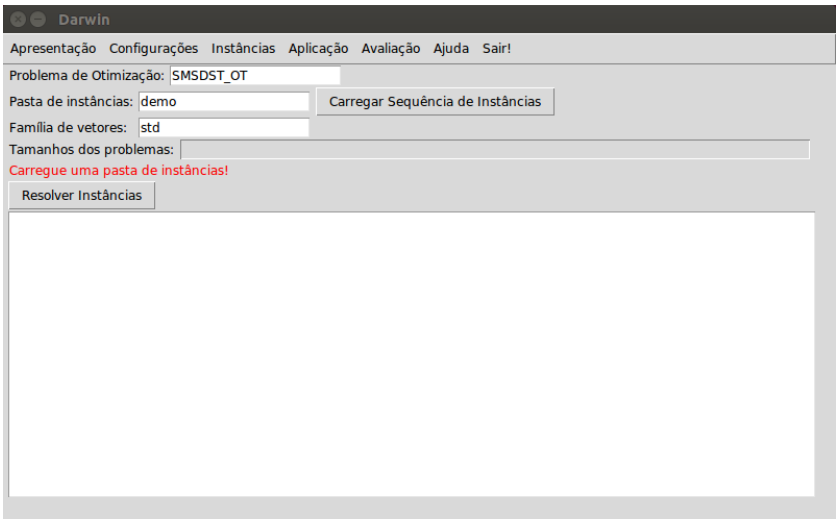


Figura 4.5: Opções para execução da camada OAA através do DT.

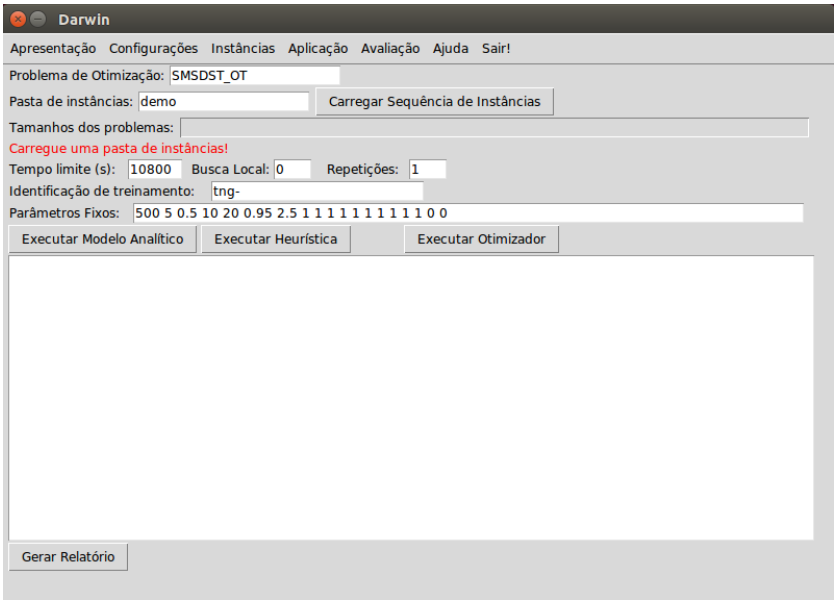


Figura 4.6: Execução de instâncias em bateladas com diferentes algoritmos no sistema DT.

Capítulo 5

Experimentos Computacionais e Análise dos Resultados

A principal vantagem do uso de um software para afinação automática é avaliar apenas candidatos aos parâmetros que definem as melhores possibilidades para operação do algoritmo configurável. Para tal, foi construído um roteiro experimental que executou três parametrizações utilizando o IRACE para cada um dos 10 conjuntos de instâncias disponíveis. Uma máquina Intel(R) Core(TM) i7-3770 CPU @ 3.50GHz com 12Gb de memória RAM foi utilizada em todos os experimentos.

Apenas as pequenas instâncias foram solucionadas pelo CPLEX, executando até a obtenção da solução ótima para todos os casos. Para os algoritmos estocásticos, ACO e NEH-T+SLS, 20 execuções foram amostradas para as instâncias pequenas e 10 para os casos de maior porte, avaliando a melhor resposta das amostras.

A tabela 5.1 detalha os 30 melhores vetores de parâmetros obtidos com a afinação do IRACE, a segunda coluna refere-se ao número de iterações executadas no experimento de afinação e a terceira contêm resultados para os 16 parâmetros ajustáveis na seguinte ordem: [ω_l, ω_u, ρ_d, ρ_e, α_1, α_2, α_3, α_4, μ_1, μ_2, μ_3, μ_4, β_1, β_2, I, L].

O método exato foi executado uma única vez para cada problema de teste, compro-

Tabela 5.1: Algoritmos ACO Obtidos com Afinação Automática do IRACE

Conjunto	**Iter.**	**Vetor de Parâmetros**
10 tarefas	1k	5 20 0.81 2.5 1.54 2.48 2.26 0.64 1.46 3.26 2.18 0.17 0.94 0.92 0.20 7
10 tarefas	5k	15 20 0.92 3.9 0.04 0.1 0.62 1.07 0.37 1.27 0.27 3.59 0.92 0.91 0.40 4
10 tarefas	10k	5 20 0.82 2.8 0.98 2.02 3.24 1.42 2.59 3.79 3.38 0.71 0.97 0.90 0.25 6
11 tarefas	1k	15 40 0.8 1.4 0.76 1.49 1.68 0.27 3.27 1.32 2.8 0.72 0.90 1.0 0.20 7
11 tarefas	5k	10 40 0.81 2.2 0.51 2.56 1.02 2.42 2.41 3.17 0.82 2.45 0.90 0.96 0.15 7
11 tarefas	10k	10 30 0.93 3.0 0.87 0.56 1.5 1.04 2.85 1.73 3.09 1.79 0.97 0.90 0.10 4
12 tarefas	1k	5 40 0.82 2.7 1.6 0.49 3.6 1.43 0.96 2.2 3.82 2.69 0.94 0.94 0.25 4
12 tarefas	5k	10 40 0.89 3.5 1.33 0.72 2.42 1.52 3.02 0.06 2.2 0.95 0.91 1.0 0.25 4
12 tarefas	10k	5 20 0.89 4.0 1.06 0.67 3.39 1.55 2.09 1.62 2.66 0.81 0.94 0.98 0.25 1
10-12 tarefas	1k	5 40 0.85 1.5 1.07 0.01 2.93 0.15 0.02 3.01 3.92 2.95 0.91 0.95 0.20 8
10-12 tarefas	5k	5 40 0.91 1.7 2.44 0.69 1.4 1.34 2.44 2.77 0.26 1.33 0.98 0.95 0.20 1
10-12 tarefas	10k	5 40 0.88 3.1 0.65 0.52 1.28 0.57 3.11 2.62 2.18 2.63 0.97 0.99 0.35 8
20 tarefas	1k	5 30 0.95 3.4 0.16 3.2 2.54 0.56 0.16 2.88 3.84 1.66 0.95 1.0 0.10 7
20 tarefas	5k	10 30 0.8 1.6 1.23 0.01 3.59 3.48 0.34 3.63 3.47 1.96 0.94 0.91 0.45 4
20 tarefas	10k	5 30 0.84 1.6 0.04 0.79 2.81 2.53 1.56 3.77 0.68 1.65 0.93 0.92 0.15 7
40 tarefas	1k	5 40 0.92 3.7 3.54 0.99 3.29 2.93 2.43 1.67 3.83 0.18 0.92 0.99 0.10 4
40 tarefas	5k	10 30 0.91 3.0 1.61 0.7 3.89 3.44 1.13 2.11 3.33 1.61 0.93 0.96 0.10 7
40 tarefas	10k	5 40 0.87 3.6 0.62 0.32 3.55 1.83 3.19 2.3 2.85 2.54 0.93 0.92 0.15 9
60 tarefas	1k	15 40 0.81 2.4 3.11 2.09 3.73 0.91 3.65 0.02 2.55 0.66 0.95 0.99 0.20 7
60 tarefas	5k	15 30 0.87 1.2 2.22 1.17 3.47 3.66 0.31 2.47 3.11 0.98 0.92 1.0 0.10 7
60 tarefas	10k	5 30 0.92 3.1 0.62 0.53 3.94 3.27 1.11 3.11 3.87 3.68 0.95 0.99 0.25 9
80 tarefas	1k	10 20 0.81 1.8 1.16 0.94 3.23 2.42 2.44 3.36 3.78 3.11 0.99 0.99 0.35 9
80 tarefas	5k	5 30 0.94 2.6 1.57 0.51 3.6 3.7 3.14 3.29 2.23 1.82 0.91 0.91 0.10 7
80 tarefas	10k	10 40 0.87 1.9 1.4 0.71 3.78 3.13 3.98 3.16 3.58 2.11 1.0 0.99 0.50 9
100 tarefas	1k	5 30 0.83 3.3 2.64 0.87 3.87 2.99 1.59 3.71 3.36 0.46 0.96 0.98 0.15 8
100 tarefas	5k	10 30 0.92 3.6 1.83 0.61 2.86 3.97 0.17 3.83 2.68 1.36 0.95 0.98 0.10 7
100 tarefas	10k	10 30 0.94 2.4 0.87 0.07 3.82 3.32 3.61 3.39 3.24 3.85 0.97 0.92 0.45 9
20-100 tarefas	1k	10 30 0.85 3.8 0.81 0.1 0.98 3.59 1.37 2.38 0.78 0.64 0.93 0.99 0.10 7
20-100 tarefas	5k	5 20 0.83 1.7 1.18 0.09 3.61 3.8 0.76 3.02 2.71 3.85 0.96 0.96 0.20 4
20-100 tarefas	10k	15 40 0.85 2.6 1.27 0.1 3.45 3.3 3.27 1.17 3.49 1.69 0.93 0.98 0.15 7

vando as respostas ótimas para as instâncias avaliadas. No entanto, o CPLEX requer muitos recursos computacionais e para problemas com 12 tarefas, alguns casos levaram mais de uma hora para completar sua solução. O uso do modelo matemático teve como objetivo validar os algoritmos estocásticos com experimentos em instâncias de pequeno porte. A heurística construtiva também foi implementada para validação da meta-heurística ACO. A figura 5.1 contem os tempos de execução para o modelo resolvido pela biblioteca CPLEX.

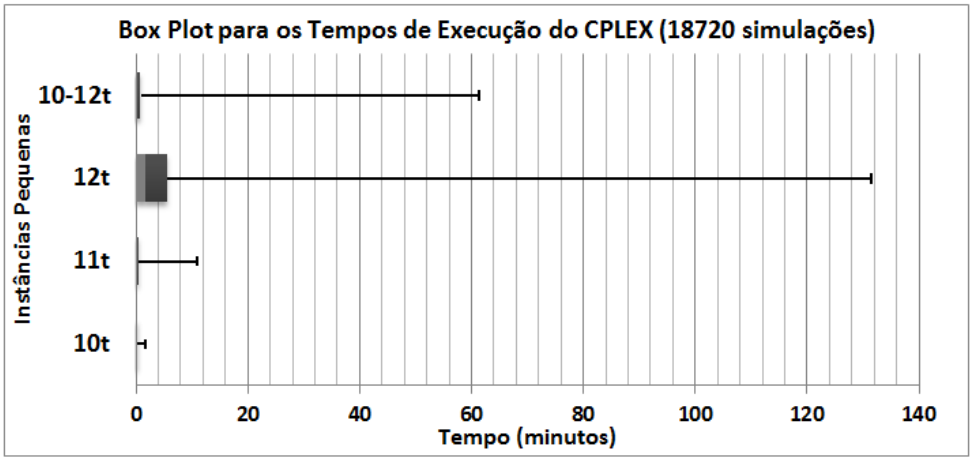


Figura 5.1: Tempos de execução para o CPLEX resolvendo as pequenas instâncias.

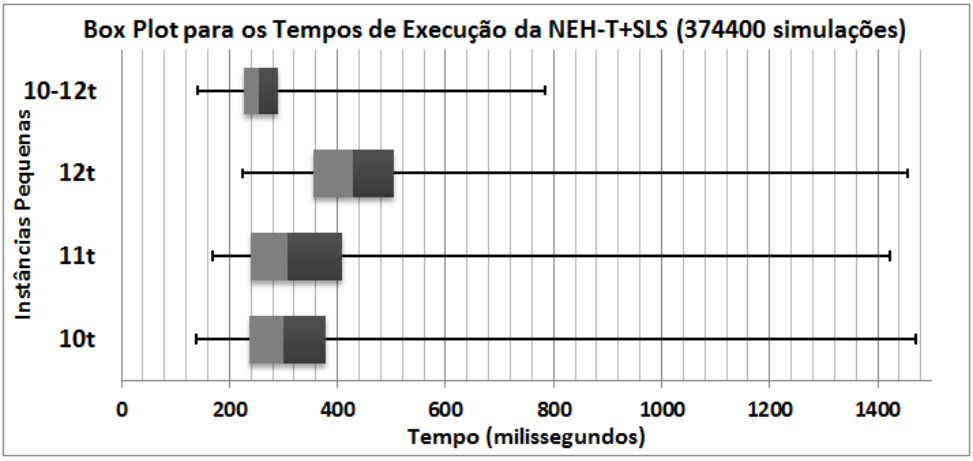


Figura 5.2: Tempos de execução para a heurística NEH-T+SLS resolvendo as pequenas instâncias.

Cada instância de pequeno porte foi solucionada 20 vezes com a heurística construtiva. A figura 5.2 descreve os tempos de execução para o algoritmo NEH-T+SLS.

Os 10 conjuntos de instâncias foram divididos em dois grupos, um com pequenas (quatro conjuntos testados) e outro com grandes instâncias (seis sequências de casos). Cada um dos 10 conjuntos foi solucionado pelas três configurações extraídas do IRACE e o melhor candidato foi determinado. Em seguida, cada conjunto foi solucionado pelos melhores candidatos dos demais conjuntos em um mesmo grupo de instâncias (pequenas ou grandes). Portanto, cada sequência com casos de pequeno porte foi solucionado por seis variações do ACO e as maiores instâncias por oito diferentes configurações da meta-heurística.

A figura 5.3 descreve os tempos de execução para o ACO solucionando as pequenas instâncias. Enquanto, as figuras 5.4 e 5.5 contêm os tempos de execução para os casos de maior porte.

De acordo com os resultados obtidos, pode se dizer que ambos os métodos estocásticos são velozes. Para pequenas instâncias, a magnitude dos tempos de execução está na casa dos milissegundos. Enquanto, para os casos de grande porte os tempos podem ser medidos em segundos, nunca ultrapassando meio minuto para qualquer uma das simulações executadas.

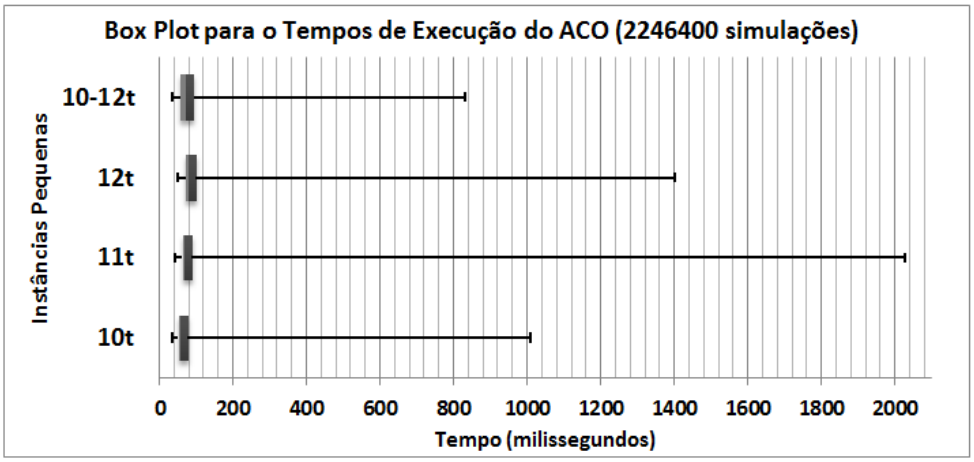


Figura 5.3: Tempos de execução para a meta-heurística ACO resolvendo as pequenas instâncias.

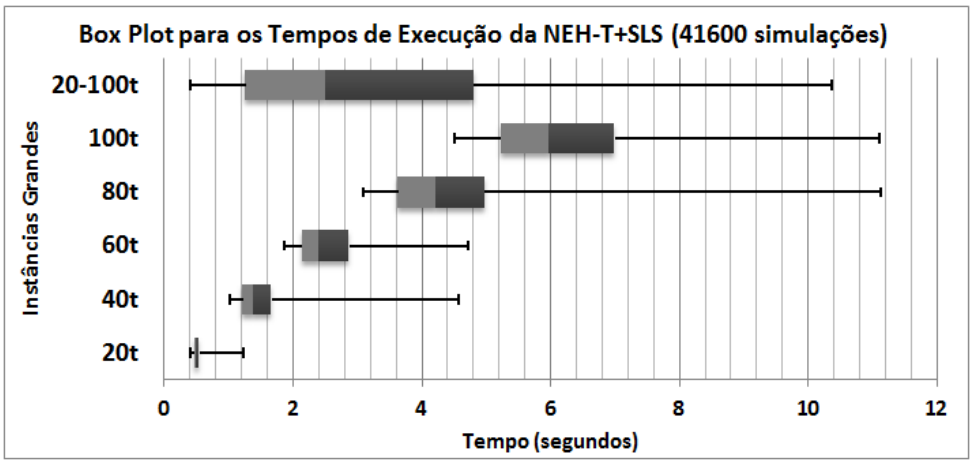


Figura 5.4: Tempos de execução para a heurística NEH-T+SLS resolvendo as grandes instâncias.

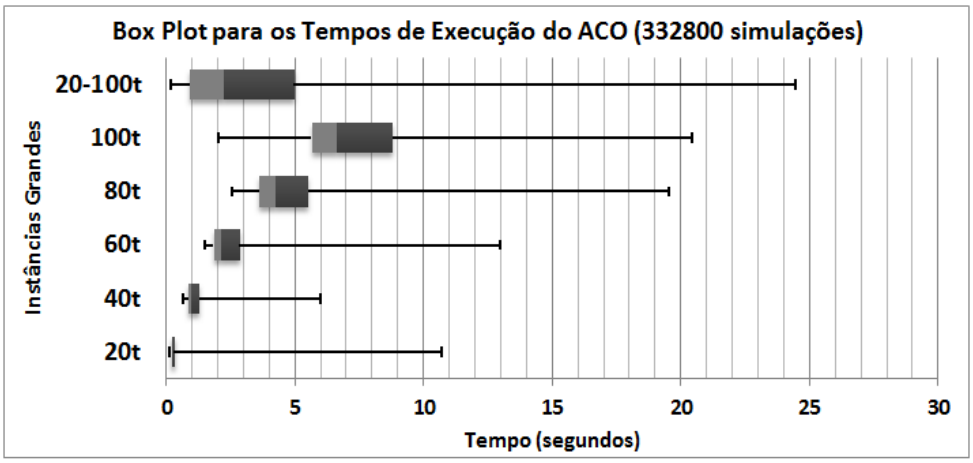


Figura 5.5: Tempos de execução para a meta-heurística ACO resolvendo as grandes instâncias.

As figuras 5.6, 5.7, 5.8 e 5.9 apresentam os tempos de execução para a camada OAA operando com diferentes velocidades para os conjuntos com grandes e pequenas instâncias. Como camada OAA executa múltiplas vezes o algoritmo ACO com diferentes vetores de parâmetros, o processo toma mais tempo que a execução singular da meta-heurística, mas ainda assim, os tempos para as pequenas instâncias são medidos em segundos para ambas as velocidades da camada OAA e todas as pequenas instâncias foram solucionadas em até 174 segundos.

Para as instâncias de grande porte sendo resolvidas pela camada OAA com a configuração rápida, a maioria dos tempos de execução da camada OAA são menores que uma hora. Apenas 10,3%, 7% e 3,8% das instâncias respectivamente para os conjuntos com 100, 80 e entre 20 e 100 tarefas, excederam uma hora de processamento. Já para a operação da camada OAA com velocidade padrão 6% dos problemas de 60 tarefas levaram mais de uma hora para completar sua solução e 69,2%, 29,6% e 11,5% respectivamente para os conjuntos com 100, 80 e entre 20 e 100 tarefas.

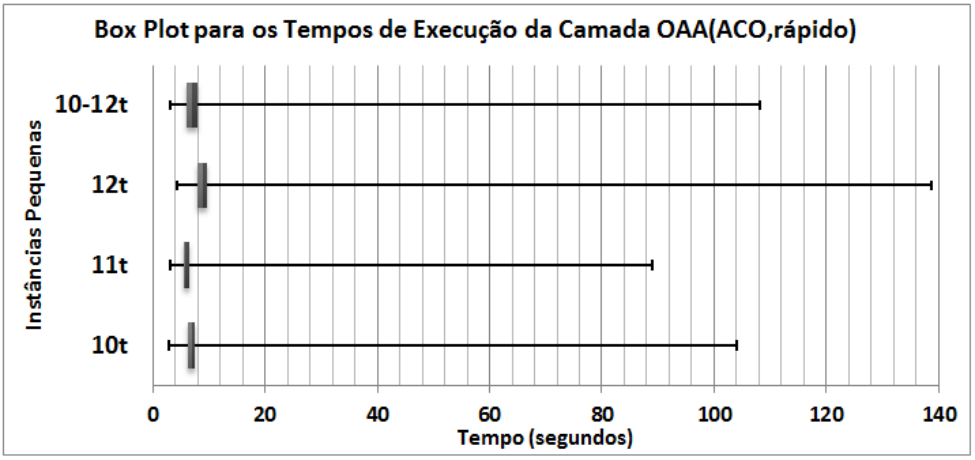


Figura 5.6: Tempos de execução para camada OAA resolvendo as pequenas instâncias no modo rápido.

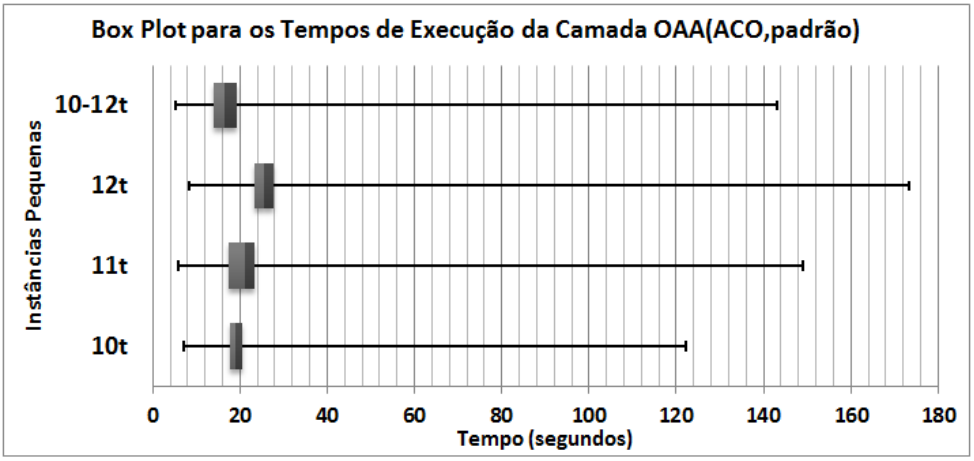


Figura 5.7: Tempos de execução para camada OAA resolvendo as pequenas instâncias no modo padrão.

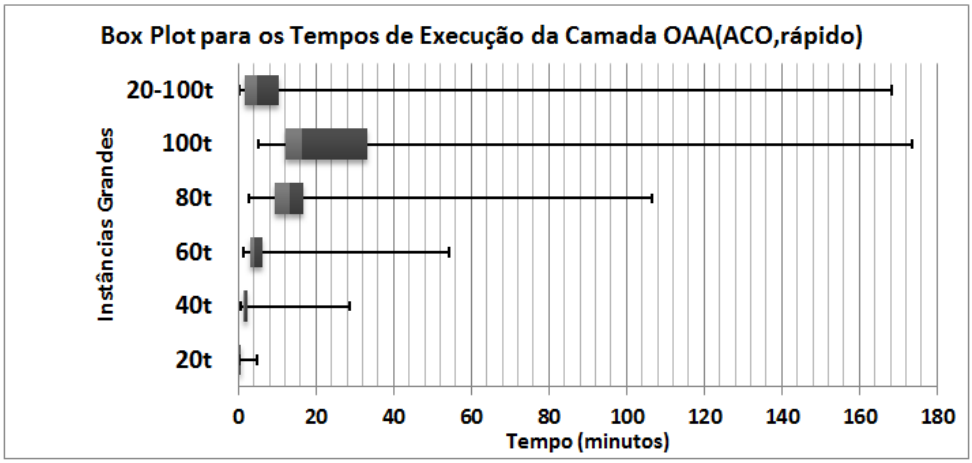


Figura 5.8: Tempos de execução para camada OAA resolvendo as grandes instâncias no modo rápido.

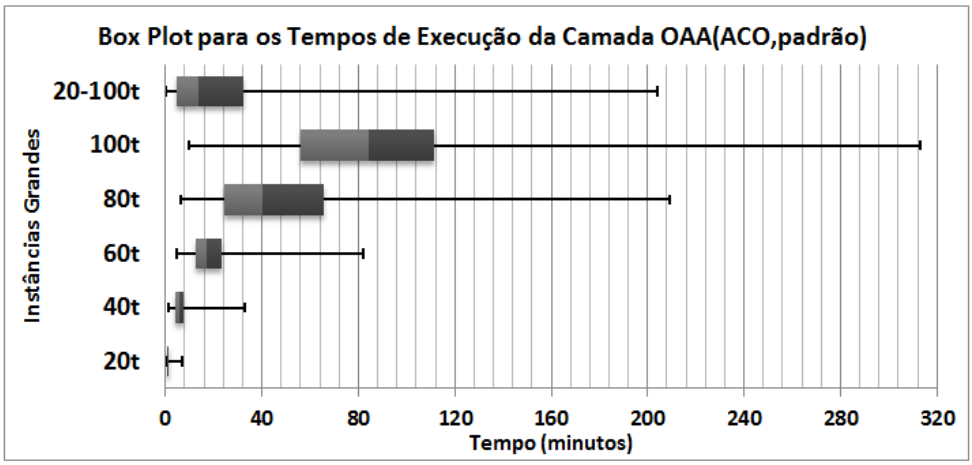


Figura 5.9: Tempos de execução para camada OAA resolvendo as grandes instâncias no modo padrão.

Nas tabelas seguintes (entre 5.2 e 5.11), as versões dos algoritmos ACO são identificadas pelo conjunto de instâncias utilizando para a fase de treinamento com IRACE e o número de iterações aplicado. Assim, *ACO10t1k* apresenta o melhor conjunto de parâmetros obtido pela afinação do ACO com as primeiras 208 instâncias para o conjunto de 10 tarefas, executando o IRACE por 1000 iterações.

As tabelas 5.2, 5.3, 5.4 e 5.5 descrevem os resultados para as pequenas instâncias. A primeira coluna refere-se ao algoritmo aplicado, a segunda contem as respostas médias entre as instâncias solucionadas, na terceira coluna é apresentada a porcentagem de respostas não ótimas e a última mostra o *gap* médio obtido, que representa a diferença entre a resposta ótima do CPLEX e a do algoritmo avaliado. A célula destacada em cada coluna representa o melhor resultado obtido para o critério avaliado. A linha com o título "Melhores ACOs" avalia a combinação entre os resultados dos candidatos elite simulados para o respectivo conjunto de instâncias e as duas últimas linhas das tabelas contêm os dados para a aplicação da camada OAA.

Tabela 5.2: Análise da Qualidade de Resposta para o Conjunto com 10 Tarefas (6240 instâncias)

Algoritmo	**Resposta Média**	**Não Ótimo**	***Gap* Médio**
CPLEX	60,705	—	—
NEH-T + SLS	63,464	10,03%	43,33%
ACO10t1k	60,794	0,87%	[gray].75 16,78%
ACO10t5k	[gray].75 60,778	[gray].75 0,50%	24.15%
ACO10t10k	61,015	2,61%	19,46%
ACO11t1k	60,838	1,17%	18,64%
ACO12t5k	61,085	2,07%	30,10%
ACO10-12t10k	60,876	1,30%	21,58%
Melhores ACOs	60,708	0,02%	26,36%
OAA(ACO,rápido)	60,973	1,91%	22,99%
OAA(ACO,padrão)	60,728	0,22%	16,94%

Para as instâncias com 10 tarefas, o algoritmo com melhor desempenho foi *ACO10t5k*, mas *ACO10t1k* retornou menor *gap* médio. Ainda que o segundo algoritmo tenha

apresentado mais respostas não ótimas, os resultados se mantiveram mais próximos ao ótimo para o *ACO10t1k*.

Tabela 5.3: Análise da Qualidade de Resposta para o Conjunto com 11 Tarefas (6240 instâncias)

Algoritmo	**Resposta Média**	**Não Ótimo**	***Gap* Médio**
CPLEX	59,797	—	—
NEH-T + SLS	63,025	12,63%	40,56%
ACO11t1k	[gray].75 60,112	2,34%	[gray].75 22,42%
ACO11t5k	60,317	3,48%	24,82%
ACO11t10k	60,281	2,98%	26,96%
ACO10t5k	60,135	[gray].75 1,84%	30,55%
ACO12t5k	60,452	3,51%	30,90%
ACO10-12t10k	60,181	2,63%	24,31%
Melhores ACOs	59,803	0,05%	20,62%
OAA(ACO,rápido)	59,904	0,80%	22,30%
OAA(ACO,padrão)	59,880	0,77%	18,13%

Tabela 5.4: Análise da Qualidade de Resposta para o Conjunto com 12 Tarefas (3120 instâncias)

Algoritmo	**Resposta Média**	**Não Ótimo**	***Gap* Médio**
CPLEX	59,066	—	—
NEH-T + SLS	62,771	15,42%	38,29%
ACO12t1k	60,578	9,20%	27,14%
ACO12t5k	59,884	5,61%	24,37%
ACO12t10k	60,013	7,18%	21,98%
ACO10t5k	59,771	[gray].75 4,26%	27,70%
ACO11t1k	[gray].75 59,623	4,62%	[gray].75 20,25%
ACO10-12t10k	59,825	5,35%	23,72%
Melhores ACOs	59,088	0,26%	14,39%
OAA(ACO,rápido)	59,406	2,40%	23,86%
OAA(ACO,padrão)	59,183	1,35%	14,68%

Nas instâncias com 11 tarefas, o melhor resultado foi apresentado pelo *ACO11t1k*, porém o *ACO10t5k* alcançou mais respostas ótimas. *ACO11t1k* também obteve melhor desempenho que as outras configurações adquiridas com a afinação específica para os

conjuntos contendo 12 tarefas e com tamanhos variados entre 10 e 12 ordens. As tabelas 5.4 e 5.5 mostram que, apesar da afinação conduzida para estes conjuntos ter produzido bons candidatos para os parâmetros do algoritmo, as versões elite para as pequenas instâncias foram *ACO10t5k* e *ACO11t1k*.

Apenas para o conjunto com 10 tarefas, a camada OAA com a velocidade rápida obteve resultados inferiores as versões singulares do ACO, para os conjuntos de 11, 12 e 10 a 12 tarefas as respostas médias da camada OAA superam todos os candidatos elite obtidos com o IRACE.

Para os blocos com casos de 10, 11, 12 e 10 a 12 tarefas, a camada OAA com a velocidade padrão apresentou resultados significativamente melhores que as configurações singulares do ACO, mostrando que esta estratégia é eficaz em classificar conjuntos de parâmetros obtidos para instâncias no passado, mas também obter novas configurações para a meta-heurística se necessário. Nestas instâncias, as soluções apresentadas pela camada OAA se mantiveram muito próximas a combinação de seis candidatos elite fornecidos pelo IRACE.

Tabela 5.5: Análise da Qualidade de Resposta para o Conjunto com 10 a 12 Tarefas (3120 instâncias)

Algoritmo	**Resposta Média**	**Não Ótimo**	***Gap* Médio**
CPLEX	61,028	—	—
NEH-T + SLS	64,380	12,53%	41,55%
ACO10-12t1k	61,717	3,88%	28,78%
ACO10-12t5k	62,264	7,92%	25,08%
ACO10-12t10k	61,500	3,17%	24,18%
ACO10t5k	61,428	2,24%	29,00%
ACO11t1k	[gray].75 61,349	[gray].75 2,18%	[gray].75 23,99%
ACO12t5k	61,692	3,69%	29,21%
Melhores ACOs	61,051	0,16%	23,26%
OAA(ACO,rápido)	61,261	1,41%	26,97%
OAA(ACO,padrão)	61,092	0,45%	23,50%

A combinação dos esforços de diferentes configurações para o ACO apresentou

melhoria significativa, visualizada pela junção dos resultados obtidos pelos candidatos elite fornecidos pelo IRACE, mas também pelos resultados da camada OAA que realiza operações com o ACO utilizando múltiplos vetores de parâmetros. Vale observar que para os casos com 10 tarefas, os melhores resultados dos seis ACOs em conjunto ofereceram apenas uma resposta não ótima dentre 6240 avaliadas e a camada OAA mostrou-se eficaz em seu propósito, destacando que melhorias em seus processos podem ainda aumentar sua eficiência.

Avaliando as instâncias de grande porte, o método exato se torna impraticável devido ao custo computacional para sua execução. Nestas instâncias a heurística construtiva serviu como padrão de comparação. No entanto, como os resultados do algoritmo NEH-T+SLS não são comprovadamente ótimos, vitórias e derrotas podem ser distinguidos nas tabelas 5.6, 5.7, 5.8, 5.9, 5.10 e 5.11. Empates entre os algoritmos são o complemento da soma das colunas três e quatro, não possuindo *gap*. A quinta coluna nestas tabelas apresenta quanto melhor foi o desempenho do ACO para as instâncias onde ele obteve melhor resultado que a heurística, na sexta coluna é mostrado o contrário. Menores *gaps* para as derrotas demonstram melhor desempenho para o ACO, enquanto maior diferença para as vitórias tem o mesmo significado.

Tabela 5.6: Análise da Qualidade de Resposta para o Conjunto com 20 Tarefas (1040 instâncias)

Algoritmo	**Resp. Média**	**Vitórias**	**Derrotas**	***GapV* Médio**	***GapD* Médio**
NEH-T + SLS	75,526	—	—	—	—
ACO20t1k	67,356	30,87%	6,15%	42,27%	[gray].75 14,89%
ACO20t5k	68,364	30,87%	8,46%	41,95%	29,24%
ACO20t10k	[gray].75 65,423	[gray].75 36,63%	[gray].75 3,65%	44,00%	18,54%
ACO40t5k	67,230	32,02%	6,06%	42,91%	23,09%
ACO60t10k	67,456	32,60%	6,63%	42,29%	27,46%
ACO80t5k	66,020	34,42%	4,42%	44,93%	24,14%
ACO100t10k	67,117	33,17%	6,25%	43,18%	28,72%
ACO20-100t10k	65,803	35,10%	4,23%	[gray].75 44,67%	21,34%
Melhores ACOs	62,867	41,25%	0,48%	48,92%	9,54%
OAA(ACO,rápido)	65,052	36,35%	3,94%	46,41%	19,46%
OAA(ACO,padrão)	64,816	37,40%	2,50%	44,98%	11,99%

Para as instâncias com 20 tarefas, *ACO20t10k* obteve melhor desempenho. Ambas

as velocidades para a camada OAA retornaram respostas significativamente melhores que as configurações singulares do ACO para todos os critérios avaliados. No entanto, a melhoria proporcionada pela combinação de oito candidatos elite obtidos com o IRACE ainda supera os resultados da camada OAA.

Nos casos com 40 tarefas, apesar do *ACO40t5k* ter apresentado melhor resultado em um dos critérios, *ACO80t5k* retornou melhores respostas para a maioria dos aspectos avaliados. Para os casos com 40 tarefas ou mais, a camada OAA apresentou resultados inferiores quando comparado a algumas versões do ACO. Possibilidades para esta queda de desempenho são: a heurística de controle (NEH-T+SLS) também apresenta queda de qualidade para instâncias de maior porte e, além disso, os processos que compõem a camada OAA são demasiadamente simplificados. Investimentos nesta proposta de implementação podem contornar barreiras observadas nesta pesquisa e são sugeridos no próximo capítulo.

Tabela 5.7: Análise da Qualidade de Resposta para o Conjunto com 40 Tarefas (1040 instâncias)

Algoritmo	**Resp. Média**	**Vitórias**	**Derrotas**	***GapV* Médio**	***GapD* Médio**
NEH-T + SLS	121,936	—	—	—	—
ACO40t1k	112,392	28,75%	15,19%	42,99%	25,46%
ACO40t5k	108,441	32,98%	6,73%	41,04%	[gray].75 16,23%
ACO40t10k	109,530	32,40%	11,54%	43,26%	23,33%
ACO20t10k	110,561	31,63%	8,46%	37,27%	17,74%
ACO60t10k	104,488	35,19%	7,60%	52,62%	23,99%
ACO80t5k	[gray].75 103,817	[gray].75 36,06%	[gray].75 6,63%	52,72%	23,47%
ACO100t10k	105,201	34,33%	9,52%	[gray].75 53,20%	24,72%
ACO20-100t10k	107,826	32,50%	9,42%	46,85%	22,73%
Melhores ACOs	98,314	39,71%	1,63%	61,04%	12,98%
OAA(ACO,rápido)	107,811	33,94%	8,08%	44,60%	25,20%
OAA(ACO,padrão)	107,235	34,52%	6,54%	44,67%	26,14%

Entre os casos com 60 tarefas, *ACO60t10k* alcançou a melhor resposta média para as meta-heurísticas testadas, mas outras configurações também mostraram resultados significativos. *ACO80t5k* apresentou o melhor resposta média e número de vitórias para as instâncias de 40 tarefas, mas nos casos com 80 ordens outra configuração apresentou melhores resultados.

Tabela 5.8: Análise da Qualidade de Resposta para o Conjunto com 60 Tarefas (624 instâncias)

Algoritmo	**Resp. Média**	**Vitórias**	**Derrotas**	***GapV* Médio**	***GapD* Médio**
NEH-T + SLS	168,960	—	—	—	—
ACO60t1k	179,135	12,66%	28,21%	23,81%	30,83%
ACO60t5k	166,490	20,35%	20,35%	28,82%	21,53%
ACO60t10k	[gray].75 149,170	32,05%	12,82%	52,90%	28,78%
ACO20t10k	162,654	23,72%	17,63%	28,54%	[gray].75 16,41%
ACO40t5k	155,127	29,81%	[gray].75 11,22%	36,78%	18,22%
ACO80t5k	150,564	31,09%	13,46%	51,47%	28,11%
ACO100t10k	149,519	[gray].75 32,69%	14,58%	[gray].75 53,05%	29,76%
ACO20-100t10k	155,490	29,97%	15,71%	43,24%	27,35%
Melhores ACOs	139,367	36,86%	4,97%	59,61%	14,84%
OAA(ACO,rápido)	166,511	26,28%	16,83%	31,27%	40,09%
OAA(ACO,padrão)	161,994	31,09%	16,35%	38,21%	46,36%

O desempenho da camada OAA para as grandes instâncias é razoavelmente superior a heurística NHE-T+SLS, utilizada como algoritmo de controle. Quando comparando a camada OAA com a velocidade padrão e as versões singulares do ACO para as instâncias de grande porte, pode-se observar que a porcentagem de derrotas resultantes da camada OAA foi menor que a melhor configuração do ACO para todos os conjuntos, exceto para aquele com 60 tarefas.

A camada OAA apresentou respostas similares as configurações singulares do ACO para as grandes instâncias. No entanto, os *gaps* observados são maiores para as derrotas e menores em relação as vitórias, conforme aumenta o tamanho das instâncias tratadas.

Tabela 5.9: Análise da Qualidade de Resposta para o Conjunto com 80 Tarefas (416 instâncias)

Algoritmo	**Resp. Média**	**Vitórias**	**Derrotas**	***GapV* Médio**	***GapD* Médio**
NEH-T + SLS	206,409	—	—	—	—
ACO80t1k	199,440	23,56%	24,28%	46,06%	30,30%
ACO80t5k	190,017	26,68%	17,79%	50,02%	26,54%
ACO80t10k	190,550	27,64%	21,15%	51,49%	27,95%
ACO20t10k	205,889	17,79%	20,91%	23,35%	18,66%
ACO40t5k	194,036	24,52%	[gray].75 13,22%	35,75%	[gray].75 18,08%
ACO60t10k	188,236	28,37%	17,31%	50,57%	27,09%
ACO100t10k	[gray].75 187,353	[gray].75 28,85%	18,75%	[gray].75 53,09%	27,44%
ACO20-100t10k	199,829	22,84%	21,39%	41,44%	28,84%
Melhores ACOs	175,385	31,25%	6,01%	59,76%	16,42%
OAA(ACO,rápido)	201,260	25,96%	12,50%	34,97%	52,17%
OAA(ACO,padrão)	202,399	27,40%	11,06%	33,61%	65,39%

Tabela 5.10: Análise da Qualidade de Resposta para o Conjunto com 100 Tarefas (416 instâncias)

Algoritmo	**Resp. Média**	**Vitórias**	**Derrotas**	***GapV* Médio**	***GapD* Médio**
NEH-T + SLS	264,209	—	—	—	—
ACO100t1k	254,873	22,36%	23,08%	42,23%	25,03%
ACO100t5k	250,038	26,68%	20,43%	48,81%	36,01%
ACO100t10k	[gray].75 241,017	[gray].75 28,85%	19,71%	[gray].75 53,81%	29,92%
ACO20t10k	266,322	16,11%	22,84%	22,71%	19,49%
ACO40t5k	248,632	24,76%	[gray].75 13,94%	34,60%	[gray].75 16,52%
ACO60t10k	241,740	28,85%	18,99%	52,18%	30,31%
ACO80t5k	246,300	27,40%	18,75%	47,36%	30,45%
ACO20-100t10k	258,608	22,12%	23,08%	40,39%	29,32%
Melhores ACOs	223,764	30,77%	7,69%	62,76%	16,06%
OAA(ACO,rápido)	264,988	25,00%	12,98%	33,36%	66,52%
OAA(ACO,padrão)	255,678	29,33%	7,93%	36,94%	94,52%

Contando as células destacadas nas tabelas para as instâncias de grande porte, *ACO100t10k* é o melhor candidato para os parâmetros do ACO, seguido pelas configurações *ACO40jobs5k* e *ACO80jobs5k* foram os candidatos elite para os casos de grande porte.

Tabela 5.11: Análise da Qualidade de Resposta para o Conjunto com 20 a 100 Tarefas (624 instâncias)

Algoritmo	**Resp. Média**	**Vitórias**	**Derrotas**	***GapV* Médio**	***GapD* Médio**
NEH-T + SLS	163,421	—	—	—	—
ACO20-100t1k	165,804	26,76%	25,48%	34,39%	41,76%
ACO20-100t5k	147,737	32,05%	15,87%	50,51%	35,13%
ACO20-100t10k	150,793	30,93%	14,42%	42,38%	32,83%
ACO20t10k	157,087	26,60%	14,26%	25,66%	19,58%
ACO40t5k	148,941	30,93%	[gray].75 8,49%	36,14%	[gray].75 17,14%
ACO60t10k	145,381	33,81%	13,94%	50,31%	33,02%
ACO80t5k	146,542	33,49%	11,70%	46,21%	33,83%
ACO100t10k	[gray].75 145,272	[gray].75 34,13%	14,26%	[gray].75 51,48%	35,61%
Melhores ACOs	132,702	38,30%	3,37%	61,85%	16,08%
OAA(ACO,rápido)	150,745	33,81%	8,97%	39,77%	56,16%
OAA(ACO,padrão)	151,995	31,89%	7,85%	37,97%	58,47%

Estes experimentos mostraram que, apesar das versões do ACO obtidas pelo treinamento com instâncias de tamanhos variados (os conjuntos com 10 a 12 e 20 a 100 tarefas) terem apresentado desempenho satisfatório, estes não foram os candidatos elite avaliados. Mostrando que simplesmente realizar o processo de afinação com uma grande variedade de tamanhos para as instâncias não garantiu o maior desempenho sobre todos

os conjuntos de teste.

Ainda que a maior parte das configurações do ACO tenha apresentado melhor desempenho que a heurística, até os melhores candidatos do ACO falharam em retornar respostas superiores ou similares ao algoritmo NEH-T+SLS para alguns casos. No entanto, quando combinando os esforços dos ACOs, o número de vitórias aumenta e o de derrotas cai significativamente. Um fato importante extraído destes experimentos é a melhoria apresentada pela combinação de candidatos elite para a meta-heurística, observando as vantagens do processo automático para afinação de algoritmos.

A camada OAA também conta com um algoritmo para parametrização automática, o meta-ACO que é operado de maneira autônoma por outro processo inteligente. Este algoritmo possibilitou obter respostas significativamente melhores que as versões obtidas pelo IRACE para ACO nas pequenas instâncias. Enquanto, para os casos de maior porte, a camada OAA retornou respostas similares aos candidatos elite extraídos pelo IRACE, mas é observada uma queda para a qualidade das respostas na medida que a escala dos problemas aumenta.

Outro ponto interessante pode ser observado nos *gaps* onde a heurística retornou melhores respostas. Olhando nas tabelas 5.7 a 5.11, *ACO20t10k* e *ACO40t5k* superam os demais candidatos para os *gaps* das derrotas. Observa-se que, um candidato elite configurado para as grandes instâncias (60, 80 e 100) é complementar a um candidato elite encontrada para as menores instâncias (20 e 40), analisando que os candidatos configurados para as casos de 60, 80 e 100 tarefas alcançam maior taxa de respostas ótimas nestes conjuntos, mas os candidatos resultantes da configuração para 20 e 40 tarefas são mais eficazes em retornar respostas com menores *gaps* para as derrotas.

Esta observação aponta fortemente para o fato de que não existe apenas um melhor candidato para os parâmetros de operação da meta-heurística e a combinação de candidatos elite representa uma melhoria significativa para o desempenho do ACO. A combinação dos candidatos elite fornecidos pelo IRACE se mostrou superior aos

resultados obtidos com a camada OAA e o meta-ACO desenvolvidos neste projeto. No entanto, muitas oportunidades de melhoria destacadas no próximo capítulo podem aumentar a eficiência desta implementação. A principal vantagem fornecida por esta proposta é eliminar a necessidade de configurar e avaliar um AE antes de colocá-lo em operação, reduzindo fortemente o tempo necessário para o processo de desenvolvimento desse tipo de algoritmo.

A tabela 5.12 avalia separadamente os processos que compõem a camada OAA: Parametrização (P); Classificação (C); Ranqueamento (R) e Solução (S). A primeira coluna da tabela 5.12 contem os conjuntos de instâncias solucionados e a segunda apresenta o modo de operação aplicado na camada OAA. A terceira coluna apresenta quantas vezes o processo de parametrização foi invocado e a quarta detalha o número de iterações médio para estas chamadas. Da quinta a oitava coluna são apresentadas as médias de iterações do ACO para os quatro processos dentro da camada OAA, observando que a parametrização não é executada para todas as instâncias, mas os demais processos são invocados para todos os novos caso. Por fim, a última coluna desta tabela mostra a quantidade de falhas encontradas pelo algoritmo 10 no processo de parametrização.

Não encontrar nenhum vetor de parâmetros que permita à meta-heurística fornecer resultados equivalentes ao fornecido pela heurística de controle, pode significar que esta meta-heurística não é adequada as características do espaço de busca da instância específica. Como na camada OAA a otimização de um determinado cenário é controlada por uma solução limitante, é possível identificar instâncias com maior complexidade para a meta-heurística empregada. Esta funcionalidade consiste em outra vantagem muito importante dentro desta proposta.

Sem uma metodologia com controle de qualidade não é possível detectar se candidatos elite, obtidos com experimentos em instâncias de treinamento, garantem boas respostas para casos inéditos. Com a opção de controle, instâncias com maior complexi-

Tabela 5.12: Análise dos processos que compõem a camada OAA.

Instâncias	**Operação**	**Chamadas de Parametrização**	**Média de Iter. por Chamada**	**Média de Execuções**				**Falhas**
				P.	C.	R.	S.	
10t	OAA(ACO,rápido)	56	46,68	0,42	37,82	18,77	18	0
	OAA(ACO,padrão)	190	90,56	2,76	143,43	56,45	30	0
11t	OAA(ACO,rápido)	92	60,76	0,90	37,65	19,36	18	0
	OAA(ACO,padrão)	181	88,70	2,57	144,08	55,32	30	0
12t	OAA(ACO,rápido)	120	18,42	0,71	40,13	18,96	18	0
	OAA(ACO,padrão)	238	47,06	3,59	141,89	54,28	30	0
10-12t	OAA(ACO,rápido)	114	24,77	0,91	41,78	19,66	18	0
	OAA(ACO,padrão)	172	35,27	1,94	137,76	52,95	30	0
20t	OAA(ACO,rápido)	43	67,37	2,79	36,58	18,23	18	0
	OAA(ACO,padrão)	97	180,45	16,83	148,88	57,14	30	0
40t	OAA(ACO,rápido)	133	177,11	22,65	64,20	19,65	18	5
	OAA(ACO,padrão)	187	271,24	48,77	225,02	58,38	30	4
60t	OAA(ACO,rápido)	127	363,02	73,88	85,77	19,71	18	14
	OAA(ACO,padrão)	145	440,12	102,27	357,43	49,06	30	19
80t	OAA(ACO,rápido)	110	323,42	85,52	140,00	19,74	18	12
	OAA(ACO,padrão)	131	386,79	121,80	503,32	56,03	30	14
100t	OAA(ACO,rápido)	134	300,62	96,83	124,63	20,70	18	20
	OAA(ACO,padrão)	158	342,42	130,06	658,46	53,94	30	20
20-100t	OAA(ACO,rápido)	135	238,07	51,51	84,03	20,42	18	12
	OAA(ACO,padrão)	184	293,39	86,51	281,20	61,11	30	12

dade podem ser detectadas e mais esforços direcionados à estes casos, seja solucionando um cenário específico de otimização ou em relação ao aprofundamento do estudo das razões de algumas instâncias serem mais complexas para determinadas meta-heurísticas.

A tabela 5.12 mostra que as falhas ocorreram apenas para as instâncias de maior porte, a partir de 40 tarefas. Havendo um total de 132 falhas que correspondem a 4,2% das 3120 instâncias avaliadas nos conjuntos onde foram detectadas falhas na parametrização para alguma instância (0,58% avaliando o total de 22880 casos simulados).

Na tabela 5.12 é possível observar que a parametrização da meta-heurística é mais complexa para os casos de maior porte. Enquanto os processos de classificação e ranqueamento estão sujeitos aos valores das variáveis que definem a velocidade da camada OAA, dependendo principalmente do número de famílias e vetores de parâmetros armazenados. Já o processo de solução depende apenas da quantidade dos melhores vetores explorados pelo processo e o número de avaliações executadas por vetor.

A classificação das famílias de vetores foi o processo mais custoso da camada OAA e deve ser alvo de futuras melhorias para potencializar a eficiência do sistema. De forma

geral, os algoritmos implementados são bastante simplificados e melhorias poder ser facilmente incorporadas pois o sistema possui uma arquitetura definida em camadas.

A camada OAA elimina a necessidade de parametrização do algoritmo, função que foi integrada de maneira inteligente dentro do processo implementado. Além disso, esta proposta permite a detecção de instâncias problemáticas que devem ser tratadas com maior cuidado. Este projeto demonstrou com sucesso a aplicabilidade deste novo paradigma, a Otimização com Aprendizado Autônomo. Pesquisas futuras podem ampliar o escopo das implementações apresentadas, aumentando a eficiência deste processo.

Mais informações podem ser reveladas com futuras análise dos dados experimentais e devem compor novos estudos. A fase de projeto para uma meta-heurística representa um problema drástico e o avanço da ciência nesta direção pode prover melhor compreensão para este processo.

Capítulo 6

Considerações Finais

Este trabalho descreveu os passos para o projeto, com foco na parametrização, de uma meta-heurística utilizando ferramentas para afinação automática. Foi visto que problemas de *scheduling* estão sujeitos à existência de instâncias heterogêneas. Para casos altamente complexos, como a formulação NP-Difícil apresentada neste projeto, necessita-se de uma meta-heurística significativamente complexa, dificultando o processo da obtenção dos valores que melhor configuram tal algoritmo para solucionar uma instância específica do problema.

Neste contexto, foi apresentada uma estratégia que vai além da simples parametrização automática, implementando uma solução capaz de identificar instâncias heterogêneas que emprega uma funcionalidade para controle de qualidade para identificar quando a parametrização é necessária, garantindo o bom funcionamento do algoritmo configurável para casos inéditos.

Interesse neste modelo de otimização para programação da produção está relacionado a formulação fortemente NP-Difícil, que incorpora características de formulações como o problema caixeiro viajante (TSP) e o problema da mochila (Knapsack). Para solucionar este caso, foram melhorados o modelo matemático e o ACO previamente desenvolvidos por Frascati (2014). Ainda que o modelo MIP seja fundamental para

a definição formal do problema e validação para casos de menor porte, os resultados mostraram que o método exato não é eficiente para solucionar casos de larga escala encontrados em cenários industriais reais. Em alguns casos testados com 12 tarefas o CPLEX levou mais de duas horas para obter a solução ótima.

O ACO é uma estratégia flexível e veloz, facilmente adaptada para diferentes casos. As simulações demonstraram que o ACO avaliado é um método eficiente para esta formulação NP-Difícil e instâncias com 100 tarefas foram solucionadas em questão de segundos. No entanto, parametrizar o algoritmo mostrou-se ser um ponto decisivo para a obter bons resultados de otimização com a meta-heurística ACO.

Convencionalmente, após a implementação de um AE deve-se iniciar seu processo de validação, que normalmente, consiste da implementação de uma solução exata e/ou heurística para comparação, geração de instâncias aleatórias seguido da parametrização (automática ou manual) do AE para um conjunto de treinamento, simulações com instâncias distintas as anteriores e análise dos resultados. Estas etapas podem levar dias ou mesmo meses para serem completadas.

Nos experimentos conduzidos em Frascati et al. (2014) a parametrização automática foi conduzida sorteando uma única instância ao acaso dentre os 208 níveis para casos de teste possíveis, mas também pequenos grupos com oito instâncias. Essa estratégia mostrou que os vetores de parâmetros especialistas para instâncias específicas falhavam para casos heterogêneos. Neste trabalho foram tomados sempre um exemplar para cada nível de problema, ou seja, as primeiras 208 instâncias de cada conjunto avaliado. Tal estratégia resultou em maior eficiência do IRACE, pois este é um AC iterativo que busca determinar a melhor configuração generalista.

O pacote IRACE e seu processo automático para parametrização possibilita focar a avaliação apenas entre os candidatos elite para as configurações da meta-heurística, fornecendo valores para os parâmetros que estatisticamente conferiram maior desempenho médio sobre um conjunto com instâncias de treinamento. No entanto, não dispensa

a necessidade de experimentação para avaliação dos valores resultantes para os parâmetros e ainda, a utilização destes valores não garante a qualidade de respostas oferecida para casos inéditos, principalmente se estes forem heterogêneos aos casos utilizados para a parametrização.

A camada OAA proporciona uma vantagem clara, não é necessário decidir pela parametrização do AE, processo que é realizado de maneira inteligente com o uso de técnicas para aprendizado de máquina e um algoritmo para controle da qualidade de respostas. Esta proposta é distinta ao parametrização *on-line*, ou controle de parâmetros, uma vez que estes não são ajustados em tempo de execução, este trabalho consistiu da implementação de um meta-ACO para parametrização *off-line* controlada por um algoritmo de aprendizado e outro para o controle de qualidade.

A partir destas funcionalidades, a camada OAA acelera o próprio processo de desenvolvimento para algoritmos evolutivos. Eliminando as etapas de experimentação, avaliação e decisão sobre as melhores configurações para operação do AE, a camada OAA permite que o desenvolvedor concentre-se na implementação e validação do AE e considerando a possibilidade de instâncias heterogêneas, enquanto a análise do seu desempenho será realizada durante a operação do algoritmo com casos inéditos.

Para operação da camada OAA, observam-se dois requisitos: i) o algoritmo sendo configurado deve ser veloz e de natureza evolutiva ou estocástica e; ii) deve ser possível a implementação de uma estratégia, também veloz, para criação de um limitante aplicado ao controle do sistema.

A análise dos dados mostrou que diferentes valores para os parâmetros e também composições distintas de algoritmos, como a aplicação de busca local, podem alterar significativamente o comportamento da meta-heurística. Portanto, diferentes algoritmos podem obter melhores soluções para casos específicos e a comparação de múltiplas versões do ACO revelou-se uma boa estratégia para o casos com instâncias heterogêneas, principalmente quando há grande variação no tamanho dos problemas tratados.

A camada OAA conta com uma estratégia para parametrização automática, um meta-ACO, aliada a um mecanismo para controle da qualidade das respostas. Além do poder de detectar instâncias heterogêneas em uma sequência, a solução proposta oferece a possibilidade de gerar múltiplos cenários agregando ainda uma resposta alternativa para controle da qualidade das respostas oferecidas pela meta-heurística. Como o objetivo neste projeto consistia em observar a melhor resposta oferecida pela camada OAA, esta opção não foi aplicada nos experimentos executados, mas foi implementada e pode ser controlada por uma opção que permite ao usuário definir o número de melhores respostas encontradas que ele deseja analisar na conclusão do processo de otimização.

A estratégia para o controle da qualidade de resposta permite que casos complexos sejam detectados e estudados com mais detalhes, colaborando com o avanço no conhecimento sobre o problema tratado, mas também com relação aos algoritmos estudados. A porcentagem de falhas encontradas não é irrelevante, sobretudo para instâncias de grande porte que são consideradas como a realidade de casos industriais para problemas de *scheduling*. A utilização de AEs sem uma estratégia de controle para qualidade de resposta representa um risco para a operação desse tipo de algoritmo sem a presenta de um especialista.

Os algoritmos inéditos apresentados neste projeto definem um novo paradigma, a Otimização com Aprendizado Autônomo (algoritmos 10, 11, 12, 13 e 14), e consistem da principal contribuição apresentada.

O meta-ACO (algoritmo 10) representa uma implementação inédita para solucionar o problema da parametrização automática de algoritmos. Este método foi implementado baseando-se em ACO, mas integrando um limitante para controle da qualidade resposta dentro da parametrização. Os algoritmos 11, 12, 13 e 14 representam um procedimento evolutivo baseado em aprendizado de máquina e o algoritmo 11 armazena os dados fornecidos quando o método de parametrização automática é invocado. Enquanto, o algoritmo 12 classifica novos casos e determina em quais conjuntos as novas informações

são armazenadas e o algoritmo 13 é responsável por descartar dados obsoletos. O método definido pelo algoritmo 14 visa coletar múltiplos cenários produtivos aplicando configurações distintas do algoritmo otimizador configurável.

Esta proposta dedicou-se a minimizar potenciais falhas para a operação de um AE otimizando seu funcionamento com um meta-ACO, mas também detectando instâncias heterogêneas com aplicação de múltiplos vetores de parâmetros para configuração de um AE. De acordo com os aspectos observados por Eiben e Smit (2011a), a camada OAA torna o AE mais robusto em relação a amplitude da variável X, minimizando a falibilidade do algoritmo (ver figura 3.3). Futuros investimentos nesta implementação podem adicionar a robustez quanto aos resultados de Z, originando uma camada de algoritmos com capacidades para controlar e garantir qualidade para a operação de um AE, considerando instâncias heterogêneas e evitando o mal funcionamento do AE, enquanto restringe os domínios de seus parâmetros com análises estatísticas. Acima de tudo, a principal vantagem da técnica apresentada é realizar estes processos de maneira autônoma sem a necessidade da supervisão de um especialista.

É importante salientar que esta implementação constituiu a prova do conceito proposto, mostrando que é possível classificar parâmetros para determinadas instâncias do problema, minimizando os esforços com a parametrização, mas principalmente detectando casos de maior complexidade que podem ser alvo de estudos futuros em pesquisas, mas também servindo para alertar praticantes sobre instâncias problemáticas durante a operação da meta-heurística de otimização.

Os resultados apresentados mostram que a camada OAA é eficiente em encontrar respostas significativamente melhores que as configurações singulares do ACO para instâncias com 20 tarefas ou menos. Já para os casos com mais de 40 tarefas a camada OAA apresentou resultados menos satisfatórios. No entanto, os processos implementados para a camada OAA são bastante simplificados e melhorias podem promover o aumento da eficiência do sistema.

Foi visto que a heurística de controle (NEH-T+SLS) também apresenta queda de qualidade para instâncias de maior porte. Tanto a estratégia de controle como os processos da camada OAA podem ser alvos de melhorias no futuro.

Dentre os principais pontos observados neste projeto, as seguintes propostas para melhoria desta implementação e estudos futuros podem ser citadas:

- *Melhorias para a funcionalidade de controle:* pode ser substituída a heurística de controle por uma meta-heurística, considerando um sistema onde múltiplos AEs são parametrizados e controlam sua qualidade entre si, ou um cenário onde existem múltiplas meta-heurísticas e heurísticas competindo para atingir maior qualidade de resposta;

- *Implementar uma função de visibilidade para o meta-ACO:* analisar a variância no desempenho do algoritmo em relação a variação dos valores de seus parâmetros pode permitir minimizar a complexidade do AE e da sua parametrização. Estes resultados podem determinar a simplificação do domínio de configuração do AE alterando-se as probabilidades de escolhas no meta-ACO com uma função de visibilidade. E assim, tratando potenciais falhas causadas pela alta amplitude da variável Z que define a robustez do AE em relação aos seus parâmetros;

- *Melhorias para o processo de classificação:* pode ser implementado um mecanismo que permita não avaliar todas as famílias com vetores de parâmetros, amostrando-as de maneira inteligente antes de determinar qual deve ser aplicada para a solução da nova instâncias, podendo também escolher apenas uma amostra dos vetores contidos na família para este processo;

- *Adição de busca local ao meta-ACO:* durante o processo iterativo do algoritmo de parametrização podem ser empregados métodos para o refinamento de respostas;

- *Parametrização de AEMOs:* como citado em Bezerra et al. (2015), pesquisas

futuras pode devotar-se ao estudo da parametrização de algoritmos evolutivos que otimizam múltiplos critérios de desempenho para um problema de otimização NP-Difícil;

- *Aplicação de meta-AEMOs:* como visto em Ugolotti e Cagnoni (2014), algoritmos de parametrização podem avaliar mais de um critério de desempenho enquanto buscam melhores valores para os parâmetros ajustáveis de um AE;

- *Implementações em paralelo:* podem visar aumentar ainda mais a velocidade, escalonando os processos presentes na camada OAA em paralelo. O paralelismo pode complementar os processos implementados, adicionando uma atividade de parametrização, similar a um AC, que é realizada após determinar-se uma classe de vetores para um grupo de instâncias, realizando testes estatísticos para avaliar e encontrar melhores valores para os parâmetros buscando aumentar a generalização dos vetores da classe para o grupo de casos.

- *Aplicação de estratégias adaptativas:* é possível investir em estratégias com parâmetros adaptativos em conjunto com outros fixos que são configurados utilizando a Otimização com Aprendizado Autônomo;

- *Aplicações da camada OAA para outros problemas de otimização:* novos investimentos podem contemplar o desenvolvimento de meta-heurísticas e heurísticas de controle para aplicação da camada OAA em outros modelos de otimização, potencialmente com parcerias em indústrias, podendo tratar outros ambientes produtivos (máquinas paralelas, *flow shop* ou *job shop*) ou mesmo casos mais complexos com modelos de otimização para planejamento e sequenciamento avançado, ou ainda para problemas NP-Difíceis oriundos de outras áreas da ciência.

A análise da variação de desempenho do algoritmo enquanto são alterados os seus parâmetros ajustáveis pode fornecer maiores detalhes sobre seu comportamento e também

maior conhecimento sobre a complexidade das instâncias sendo tratadas. O processo de parametrização pode ser melhorado adicionando processos similares aos encontrados nos ACs para comparação estatística entre vetores de parâmetros. Os demais processos também estão sujeitos a melhorias de implementação, o que é facilitado uma vez que esta proposta apresenta uma implementação em camadas.

Implementações com valores fixos para os parâmetros ajustáveis não são adequadas para casos com instâncias heterogêneas. Praticantes devem avaliar com cuidado o processo de parametrização proposto para meta-heurísticas que incorporam pacotes de otimização. Um algoritmo com bom desempenho em testes iniciais pode se tornar obsoleto caso sujeito à instâncias com características distintas das usadas na fase de parametrização. Enquanto os fornecedores desses tipos de estratégias para otimização deve adicionar manutenção em seus produtos se a meta-heurística operar com valores fixos para os parâmetros, observando que até algoritmos adaptativos podem possuir parâmetros imutáveis.

Esta pesquisa tem o potencial para publicações inéditas como: a apresentação de uma metodologia para experimentação e validação de AEs utilizando métodos de configuração automática e demonstrando a existência de instâncias heterogêneas para problemas de *scheduling*; a descrição inédita da camada para Otimização com Aprendizado Autônomo e os componentes que definem este novo paradigma e; a possibilidade da geração de múltiplos cenários coletados a partir de execuções com vetores de parâmetros distintos para uma meta-heurística com avaliação da qualidade de resposta.

A camada OAA permite o controle da qualidade de resposta do algoritmo configurável aplicado, possibilitando a detecção de instâncias complexas onde o algoritmo configurável encontrou dificuldades para obtenção de soluções com alta qualidade. Tal funcionalidade apresenta uma vantagem para aqueles que aplicam esse tipo de algoritmo, pois podem identificar pontos críticos em sua operação, mas sobretudo, esta característica permite que instâncias problemáticas sejam detectadas e estudadas po-

tencializando o avanço do conhecimento sobre as características dos algoritmos e casos tratados.

Além de ser eficiente e eficaz para a parametrização e controle da qualidade de resposta para a aplicação de um AE, o mta-ACO, a camada OAA e o sistema *Darwin Toolkit* implementados neste projeto, amparam e facilitam o processo de desenvolvimento para um AE, fornecendo funcionalidades para avaliação de instâncias em bateladas, mas sobretudo, eliminando a necessidade da tomada de decisão sobre os melhores parâmetros do AE para solucionar uma instância específica.

Os resultados apresentados neste trabalho, tanto em relação a velocidade resultante para a camada OAA como quanto a qualidade das respostas oferecidas, encorajam fortemente a perseguição de novos avanços nesta direção. Este projeto proporcionou enriquecimento da ciência no sentido de promover autonomia para a operação de AEs, mostrando que a aplicação de múltiplos sistemas baseados em inteligência artificial operando em conjunto têm o potencial de oferecer soluções extremamente robustas.

Referências Bibliográficas

Ahmadizar, F. e Farhadi, S. (2015). Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers & Operations Research*, 53:194–205.

Aleti, A. e Moser, I. (2016). A systematic literature review of adaptive parameter control methods for evolutionary algorithms. *ACM Comput. Surv*, 49(3):35 pages. DOI: http://dx.doi.org/10.1145/2996355.

Altiparmak, F., Gen, M., Lin, L., e Paksoy, T. (2006). A genetic algorithm approach for multi-objective optimization of supply chain networks. *Computers & Industrial Engineering*, 51:196–215.

Andersson, M., Bandaru, S., e Ng, A. H. (2016). Tuning of multiple parameter sets in evolutionary algorithms. Em *GECCO 16*. DOI: http://dx.doi.org/10.1145/2908812.2908899.

Baker, K. R. (1974). *Introdution to Sequencing and Scheduling*. Wiley & Sons.

Balaprakash, P., Birattari, M., e Stützle, T. (2007). *Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement, chapter Hybrid Metaheuristics in Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Bartz-Beielstein, T., Branke, J., Mehnen, J., e Mersmann, O. (2014). Evolutionary algorithms. *WIREs Data Mining Knowl Discov*, 4:178–195.

Bartz-Beielstein, T., Lasarczyk, C., , e Preub, M. (2005). Sequential parameter optimization. Em *Congress on Evolutionary Computation*, pgs. 773–780. IEEE Press.

Bartz-Beielstein, T., Parsopoulos, K., e Vrahatis, M. (2004). Analysis of particle swarm optimization using computational statistics. Em *ICNAAM-2004 Extended Abstracts*.

Bettwy, M., Deschinkel, K., e Gomes, S. (2014). An optimization tool for process planning and scheduling. Em *IFIP AICT*, volume 438, pgs. 443–450.

Bezerra, L. C. T., nez, M. L.-I., e Stützle, T. (2015). Automatic component-wise design of multi-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*. DOI 10.1109/TEVC.2015.2474158.

Birattari, M. (2009). Tuning metaheuristics: A machine learning perspective. *Studies in Computational Intelligence*, 197.

Birattari, M., Stutzle, T., Paquete, L., e Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. Em *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*.

Birattari, M., Yuan, Z., Balaprakash, P., e Stutzle, T. (2009). F-race and iterated f-race: An overview. Technical report, IRIDIA, Université libre de Bruxelles, Belgium.

Boland, N., Clement, R., e Waterer, H. (2016). A bucket indexed formulation for nonpreemptive single machine scheduling problems. *Journal on Computing*, 28(1):14–30. http://dx.doi.org/10.1287/ijoc.2015.0661.

Bosman, P. A. N. (2012). On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization. *IEEE Transactions on Evolutionary Computation*.

Chaparro, E. R. e Sosa, M. L. (2011). Coordinated tuning of a set of static var compensators using evolutionary algorithms. *IEEE Trondheim PowerTech*.

Chen, J. C., Chen, Y.-Y., Chen, T.-L., e Lin, J. Z. (2016). Comparison of simulated annealing and tabusearch algorithms in advanced planning and scheduling systems for tft-lcd colour filter fabs. *International Journal of Computer Integrated Manufacturing*. http://dx.doi.org/10.1080/0951192X.2016.1145805.

Chen, J. C., Huang, P. B., Wu, J.-G., Lai, K. Y., Chen, C.-C., e Peng, T.-W. (2013). Advanced planning and scheduling for tft-lcd color filter fab with multiple lines. *International Journal of Advanced Manufacturing Technology*.

Chen, J.-K. e Chen, A. (2002). Ts advanced planning & scheduling system. Em *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*.

Chen, K. e Ji, P. (2007a). A genetic algorithm for dynamic advanced planning and scheduling (daps) with a frozen interval. *Expert Systems with Applications*.

Chen, K. e Ji, P. (2007b). A mixed integer programming model for advanced planning and scheduling (aps). *European Journal of Operational Research*, 181:515–522.

Chen, K., Ji, P., e Wang, Q. (2010). A case study for advanced planning and scheduling (aps). *J Syst Sci Syst Eng*.

Chen, T., Tang, K., Chen, G., e Yao, X. (2012). A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*.

Chua, T., Wang, F., Cai, T., e Yin, X. (2006). A heuristics-based advanced planning and scheduling system with bottleneck scheduling algorithm. Em *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on*.

Davari, M., Demeulemeester, E., Leus, R., e Nobibon, F. T. (2015). Exact algorithms for single-machine scheduling with time windows and precedence constraints. *Journal of Scheduling*. DOI 10.1007/s10951-015-0428-y.

David, F., Pierreval, H., e Caux, C. (2006). Advanced planning and scheduling systems in aluminium conversion industry. *International Journal of Computer Integrated Manufacturing*.

Dayou, L., Pu, Y., e Ji, Y. (2009). Development of a multiobjective ga for advanced planning and scheduling problem. *Int J Adv Manuf Technol*.

Dorigo, M. e Stützle, T. (2002). *The ant colony optimization metaheuristic: Algorithms, applications, and advances. Handbook of Metaheuristics, p. 251-285*. Kluwer Academic Publishers.

Dorne, R., Voudouris, C., e Owusu, G. (2008). An advanced planning and scheduling platform for service enterprise. Em *4th International Conference on Innovations in Information Technology*.

Eiben, A. e Smit, S. (2011a). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*.

Eiben, A. E., Hinterding, R., e Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 3(2).

Eiben, A. E., Michalewicz, Z., Schoenauer, M., e Smith, J. E. (2007). Parameter control in evolutionary algorithms. *Parameter Setting in Evolutionary Algorithms Studies in Computational Intelligence*, 54:19–46.

Eiben, A. E. e Smit, S. K. (2011b). *Evolutionary Algorithm Parameters and Methods to Tune Them, chapter 2 in Autonomous Search*. Springer-Verlag Berlin Heidelberg.

Errington, J. (1997). Advanced planning & scheduling (aps): a powerful emerging technology:. *The Institution of Electrical Engineers. Printed and published by the IEE, Savoy Place, London WC2R OBL, UK.*

Ethridge, J., Ditzler, G., e Polikar, R. (2010). Optimal v-svm parameter estimation using multi objective evolutionary algorithms. Em *IEEE Congress on Evolutionary Computation (CEC)*.

Fadil, M. A. e Darus, I. Z. M. (2013). Evolutionary algorithms for self-tuning active vibration control of flexible beam. Em *Australian Control Conference*.

Fang, Y. e Lu, X. (2016). Online parallel-batch scheduling to minimize total weighted completion time on single unbounded machine. *Information Processing Letters*, 116:526–531.

Fleischmann, B., Stadtler, H., Surie, C., Wagner, M., e Meyr, H. (2004). *Supply Chain Management and Advanced Planning Concepts, Models, Software and Case Studies Third Edition*. Springer.

Frascati, G. (2014). Problema de programação da produção com setup dependente da sequência e terceirização permitida: aplicando a meta-heurística de otimização por colônia de formigas. Master's thesis, Universidade Federal de São Carlos.

Frascati, G., Tavares Neto, R. F., e Nagano, M. S. (2014). Ant colony algorithms to solve the outsourcing enabled problem with setup-dependent single-machine scheduling. Em *VIII ALIO/EURO Workshop on Applied Combinatorial Optimization*.

Gacto, M. J., Alcalá, R., e Herrera, F. (2009). *Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems, in Soft Computing.* Springer-Verlag.

Gambardella, L. M. e Dorigo, M. (1996). Solving symmetric and asymmetric tsps by ant colonies. *IEEE Press*, pgs. 622–627.

Gao, J., Sun, L., e Gen, M. (2008). A hybrid genetic and variable neighborhood des-

cent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*.

García-Sánchez, P., Romero, G., González, J., Mora, A. M., Arenas, M. G., Castillo, P. A., Fernandes, C., e Merelo, J. J. (2016). Studying the effect of population size in distributed evolutionary algorithms on heterogeneous clusters. *Applied Soft Computing*, 38:530–547.

Gen, M. e Lin, L. (2013). Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. *J Intell Manuf*.

Giacon, E. e Mesquita, M. A. (2011). Levantamento das práticas de programação detalhada da produção: um survey na indústria paulista. *Gestão de Produção*, 18(3):487–498.

Gonçalves, T. C., Valente, J. M. S., e Schaller, J. E. (2016). Metaheuristics for the single machine weighted quadratic tardiness scheduling problem. *Computers & Operations Research*, 70:115–126.

Grosan, C., Abraham, A., e Nicoara, M. (2005). Performance tuning of evolutionary algorithms using particle sub swarms. Em *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. SYNASC 2005.*

Gruat-La-Forme, F. A., Botta-Genoulaz, V., Campagne, J.-P., e Millet, P.-A. (2005). Advanced planning and scheduling system: An overview of gaps and potential sample solutions. Em *International Conference on Industrial Engineering and Systems Management*, pgs. pp.683–695, Marrakech, Morocco.

Gu, M., Lu, X., Gu, J., e Zhang, Y. (2016). Single-machine scheduling problems with machine aging effect and an optional maintenance activity. *Applied Mathematical Modelling*, 000:1–10. article in press.

Guo, Y., Chen, W.-N., e Zhang, J. (2012). Enhancing the performance of evolutionary algorithms: a novel maturity-based adaptation strategy. Em *IEEE World Congress on Computational Intelligence*.

Gupta, S. e Kumar, N. (2014). Parameter tuning in quantum-inspired evolutionary algorithms for partitioning complex networks. Em *Annual Conference on Genetic and Evolutionary Computation*.

Gutiérrez-Urquídez, R., Valencia-Palomo, G., Rodríguez-Elias, O., e Trujillo, L. (2015). Systematic selection of tuning parameters for efficient predictive controllers using a multiobjective evolutionary algorithm. *Applied Soft Computing*, 31:326–338.

Hadaya, P. e Pellerin, R. (2008). Determinants of advance planning and scheduling systems adoption. Em *The Third International Conference on Software Engineering Advances*.

Han, B., Zhang, W., Lu, X., e Lin, Y. (2015). On-line supply chain scheduling for single-machine and parallel-machine configurations with a single customer: Minimizing the makespan and delivery cost. *European Journal of Operational Research*, 244(704-714).

Herr, O. e Goel, A. (2016). Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints. *European Journal of Operational Research*, 248:123–125.

Herrero, J. M., Blasco, X., Martínez, M., e Sanchis, J. (2008). *Multiobjective Tuning of Robust PID Controllers Using Evolutionary Algorithms, chapter Applications of Evolutionary Computing in Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Hu, H., Xu, L., Wei, R., e Zhu, B. (2010). Multi-objective tuning of nonlinear pid controllers for greenhouse environment using evolutionary algorithms. Em *IEEE Congress on Evolutionary Computation (CEC)*.

Hutter, F., Hoos, H. H., Leyton-Brown, K., e Stutzle, T. (2009). Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*.

Hvolby, H.-H. e Steger-Jensen, K. (2010). Technical and industrial issues of advanced planning and scheduling (aps) systems. *Computers in Industry*, 61:845–851.

Ivert, L. K. (2012). Shop floor characteristics influencing the use of advanced planning and scheduling systems. *Production Planning & Control*.

Ji, M., Yang, Q., Yao, D., e Cheng, T. (2015). Single-machine batch scheduling of linear deteriorating jobs. *Theoretical Computer Science*, 580:36–49.

Ji, P. e Li, L. (2015). Single-machine group scheduling problems with variable job processing times. *Mathematical Problems in Engineering*, 2015(758919):9.

Jiang, S. e Yang, S. (2016). Evolutionary dynamic multiobjective optimization: Benchmarks and algorithm comparisons. *IEEE Transactions on Cybernetics*. Digital Object Identifier 10.1109/TCYB.2015.2510698.

Jin, L., Zhang, C., e Shao, X. (2015). An effective hybrid honey bee mating optimization algorithm for integrated process planning and scheduling problems. *Int J Adv Manuf Technol*, 80:1253–1264.

Jin, L., Zhang, C., Shao, X., Yang, X., e Tian, G. (2016). A multi-objective memetic algorithm for integrated process planning and scheduling. *Int J Adv Manuf Technol*, 85:1513–1528.

Joan-Arinyo, R., Luzón, M., e Yeguas, E. (2011). Parameter tuning of pbil and chc evolutionary algorithms applied to solve the root identification problem. *Applied Soft Computing*.

Jovanovic, M., Delibasic, B., Vukicevic, M., Suknovic, M., e Martic, M. (2014). Evolutionary approach for automated component-based decision tree algorithm design. *Intelligent Data Analysis*, 18:63–77.

Junli, L., Jianlin, M., e Guanghui, Z. (2011). Evolutionary algorithms based parameters tuning of pid controller. Em *Chinese Control and Decision Conference (CCDC)*.

Kaji, H., Ikeda, K., e Kita, H. (2008). Acceleration of parametric multi-objective optimization by an initialization technique for multi-objective evolutionary algorithms. Em *IEEE World Congress on Computational Intelligence*.

Kamath, A. K., Lachat, D., Watt, M., e Menon, P. P. (2015). Automated parameter tuning of a generic rover dynamics model for autonomous planetary space capability development, verification and validation of space applications. Em *American Control Conference*.

Kattan, A. e Arif, M. (2012). Pso based on surrogate modeling as meta-search to optimise evolutionary algorithms parameters. Em *Proceedings of the 14th annual conference on Genetic and evolutionary computation*.

Keshavarz, T., Savelsbergh, M., e Salmasi, N. (2015). A branch-and-bound algorithm for the single machine sequence-dependent group scheduling problem with earliness and tardiness penalties. *Applied Mathematical Modelling*, 39:6410–6424.

Khan, N. Z. e Mitschele-Thiel, A. (2008). A step towards an autonomous tuning engine design for self-protection and self-configuration in t. sobh et al. (eds.), novel algorithms and techniques in telecommunications, automation and industrial electronics. *Springer Science+Business Media B.V.*, pgs. 454–457.

Khoshrooz, R. A., Vahid, M. A. D., Mirshams, M., Homaeinezhad, M., e Ahadi, A. (2012). Novel method on using evolutionary algorithms for pd optimal tuning. *Applied Mechanics and Materials*.

Kim, D. G. e Choi, J. Y. (2015). Sum-of-processing-times-based two-agent single-machine scheduling with aging effects and tardiness. *Mathematical Problems in Engineering*, 2015(768148):12.

Kır, S. e Yazgan, H. R. (2016). A sequence dependent single machine scheduling problem with fuzzy axiomatic design for the penalty costs. *Computers & Industrial Engineering*, 92:95–104.

Kortenkamp, D. (2003). A day in an astronaut's life: Reflections on advanced planning and scheduling technology. *IEEE Intelligent Systems*.

Kung, L.-C. e Chern, C.-C. (2009). Heuristic factory planning algorithm for advanced planning and scheduling. *Computers & Operations Research*, 36:2513–2530.

Kuroda, M., Shin, H., e Zinnohara, A. (2002). Robust scheduling in an advanced planning and scheduling environment. *International Journal of Production Research*.

Laalaoui, Y. e M'Hallah, R. (2016). A binary multiple knapsack model for single machine scheduling with machine unavailability. *Computers & Operations Research*, 72:71–82.

Lazarev, A. A., Arkhipov, D. I., e Werner, F. (2016). Scheduling jobs with equal processing times on a single machine: minimizing maximum lateness and makespan. *Optimization Letters*. DOI 10.1007/s11590-016-1003-y.

Lee, Y. H., Jeong, C. S., e Moon, C. (2002). Advanced planning and scheduling with outsourcing in manufacturing supply chain. *Computers & Industrial Engineering*, 43:351–374.

Li, D., Meng, X., Li, M., e Tian, Y. (2016). An aco-based intercell scheduling approach for job shop cells with multiple single processing machines and one batch processing

machine. *Journal of Intelligent Manufacturing*, 27:283–296. DOI 10.1007/s10845-013-0859-2.

Li, G. e Lu, X. (2016). Approximation algorithms for the single-machine scheduling with a period of maintenance. *Optimization Letters*, 10:543–562. DOI 10.1007/s11590-015-0881-8.

Li, J., Lu, G., , e Yao, X. (2011). *Fitness Landscape-Based Parameter Tuning Method for Evolutionary Algorithms for Computing Unique Input Output Sequences, chapter Neural Information Processing in Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Li, S. e Ma, L. (2010). A forecast netting and consumption model for advanced planning and scheduling. Em *International Conference on Logistics Systems and Intelligent Management*.

Lin, C.-H., Hwang, S.-L., e Wang, E. M.-Y. (2007). A reappraisal on advanced planning and scheduling systems. *Industrial Management & Data Systems*, 107(8):1212 – 1226.

Lin, L. e Gen, M. (2009). *Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation, in Soft Computing*. Springer-Verlag.

Lin, Q., Liu, Z., Yan, Q., Du, Z., Coello, C. A. C., Liang, Z., Wang, W., e Chen, J. (2016). Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm. *Information Sciences*, 339:332–352.

Liu, L. e Qi, E.-S. (2008). Research of production logistics system in car factory based on advanced planning and scheduling. Em *4th International Conference on Wireless Communications, Networking and Mobile Computing*.

Liu, X., Ni, Z., e Qiu, X. (2016). Application of ant colony optimization algorithm in integrated process planning and scheduling. *Int J Adv Manuf Technol*, 84:393–404.

López-Ibáñez, M., Pérez Cáceres, L., Dubois-Lacoste, J., Stützle, T., e Birattari, M. (2016). The irace package: User guide. Technical Report TR/IRIDIA/2016-004, IRIDIA, Université Libre de Bruxelles, Belgium.

Lu, L. e Zhang, L. (2016). Single-machine scheduling with production and rejection costs to minimize the maximum earliness. *Journal of Combinatorial Optimization*. DOI 10.1007/s10878-016-9992-0.

Lutton, E., Gilbert, H., Cancino, W., Bach, B., Pallamidessi, J., Parrend, P., e Collet, P. (2014). Visual and audio monitoring of island based parallel evolutionary algorithms. *Journal of Grid Computing*.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., e Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. tr/iridia/2011-004, iridia. Technical report, IRIDIA, Université libre de Bruxelles, Belgium.

Ma, H., Simon, D., Fei, M., Shu, X., e Chen, Z. (2014). Hybrid biogeography-based evolutionary algorithms. *Engineering Applications of Artificial Intelligence*.

Madureira, A., Falcão, D., e Pereira, I. (2012). Ant colony system based approach to single machine scheduling problems: Weighted tardiness scheduling problem. *Nature and Biologically Inspired Computing*, pgs. 86–91.

Malik, M. M. e Qiu, M. (2008). A review of planning and scheduling in the pulp and paper supply chain. Em *Proceedings of the 2008 IEEE IEEM*.

Martínez, S. Z., Oropeza, E. G. Y., e Coello, C. A. C. (2011). *Self-adaptation Techniques Applied to Multi-Objective Evolutionary Algorithms, chapter Learning and Intelligent Optimization in Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg.

Merkle, D. e Middendorf, M. (2003). Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Appl. Intell*, 1:105–111.

M'Hallah, R. e Alhajraf, A. (2016). Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem. *Journal of Scheduling*, 19:191–205. DOI 10.1007/s10951-015-0429-x.

Mishra, S. K. e Rao, C. S. P. (2016). Performance comparison of some evolutionary algorithms on job shop scheduling problems. *IOP Conf. Series: Materials Science and Engineering*, 49:12–41.

Moghaddam, A., Yalaoui, F., e Amodeo, L. (2015). Efficient meta-heuristics based on various dominance criteria for a single-machine bi-criteria scheduling problem with rejection. *Journal of Manufacturing Systems*, 34:12–22.

Moness, M. e Moustafa, A. M. (2015). Tuning a digital multivariable controller for a lab-scale helicopter system via simulated annealing and evolutionary algorithms. *Transactions of the Institute of Measurement and Control*, 37:1254–1273.

Montesco, R. A. E., Pessoa, M. A. O., e Blos, M. F. (2015). Scheduling heuristic resourced-based on task time windows for aps (advanced planning and scheduling) systems. Em *IFAC-PapersOnLine*, volume 48, pgs. 2273–2280.

Moon, C., Kim, J. S., e Gen, M. (2004). Advanced planning and scheduling based on precedence and resource constraints for e-plant chains. *International Journal of Production Research*.

Moon, C. e Seo, Y. (2005). Evolutionary algorithm for advanced process planning and scheduling in a multi-plant. *Computers & Industrial Engineering*, 48:311–325.

Moon, C., Seo, Y., Yun, Y., e Gen, M. (2006). Adaptive genetic algorithm for advanced planning in manufacturing supply chain. *J Intell Manuf*, 17:509–522.

Mullen, R. J., Monekosso, D., Barman, P., e Remagnino, P. (2009). A review of ant algorithms. *Expert Systems with Applications*, 36:9608–9617.

Munoz-Carpintero, D. e Sáez, D. (2015). A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. *Transportation Science*, 49(2):239–253.

Musselman, K., O'Reilly, J., e Duket, S. (2002). The role of simulation in advanced planning and scheduling. Em *Proceedings of the 2002 Winter Simulation Conference*.

Muñoz-Salinas, R., Aguirre, E., Cordón, O., , e García-Silvente, M. (2008). Automatic tuning of a fuzzy visual system using evolutionary algorithms: Single-objective versus multiobjective approaches. *IEEE Transactions on Fuzzy Systems*.

Nannen, V. e Eiben, A. (2006). A method for parameter calibration and relevance estimation in evolutionary algorithms. Em *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)*.

Nannen, V., Smit, S. K., e Eiben, A. E. (2008). *Costs and Benefits of Tuning Parameters of Evolutionary Algorithms, chapter Parallel Problem Solving from Nature in Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Nanvala, H. (2011). Use of genetic algorithm based approaches in scheduling of fms: A review. *International Journal of Engineering Science and Technology*.

Nawaz, M., Enscore Jr, E. E., e Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. *OMEGA*, 11:91–95.

Ning, W., Zhong, H., Lei, Y., e An, M. (2008). Constraint engine in advanced planning and scheduling. Em *Proceedings of the IEEE International Conference on Automation and Logistics*.

Onieva, E., Hernandez-Jayo, U., Osaba, E., Perallos, A., e Zhang, X. (2015). A multi-objective evolutionary algorithm for the tuning of fuzzy rule bases for uncoordinated intersections in autonomous driving. *Information Sciences*, 321:14–30.

Ornek, A., Ozpeynirci, S., e Ozturk, C. (2010). A note on a mixed integer programming model for advanced planning and scheduling (aps). *European Journal of Operational Research*.

Osorio, K., Luque, G., e Alba, E. (2011). Distributed evolutionary algorithms with adaptive migration period. Em *11th International Conference on Intelligent Systems Design and Applications (ISDA)*.

Ou, J., Zhong, X., e Li, C.-L. (2016). Faster algorithms for single machine scheduling with release dates and rejection. *Information Processing Letters*, 116:503–507.

Ozturk, C. e Ornek, A. M. (2014). Operational extended model formulations for advanced planning and scheduling systems. *Applied Mathematical Modelling*.

Pan, I., Das, S., e Gupta, A. (2011). Handling packet dropouts and random delays for unstable delayed processes in ncs by optimal tuning of pid controllers with evolutionary algorithms. *ISA Transactions*.

Pellegrini, P., Stützle, T., e Birattari, M. (2012). A critical analysis of parameter adaptation in ant colony optimization. *Swarm Intell*, 6:23–48.

Pereira, J. (2016). The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective. *Computers & Operations Research*, 66:141–152.

Policella, N., Oliveira, H., e Benzi, E. (2014). Benefits of using advanced planning and scheduling technology: The alphasat tdp operations case. Em *SpaceOps Conferences*, Pasadena, CA.

Rabanal, P., Rodríguez, I., e Rubio, F. (2015). On the uselessness of finite benchmarks to assess evolutionary and swarm methods. Em *GECCO 15*. DOI: http://dx.doi.org/10.1145/2739482.2764672.

Ramacher, R. e Mönch, L. (2016). An automated negotiation approach to solve single machine scheduling problems with interfering job sets. *Computers & Industrial Engineering*, xxx:xxx–xxx. Article in Press.

Reynoso-Meza, G., Sanchis, J., Blasco, X., e Herrero, J. M. (2012). Multiobjective evolutionary algorithms for multivariable pi controller design. *Expert Systems with Applications*.

Reynoso-Meza, G., Sanchis, J., Blasco, X., e Martínez, M. (2013). Algoritmos evolutivos y su empleo en el ajuste de controladores del tipo pid: Estado actual y perspectivas. *Revista Iberoamericana de Automática e Informática industrial*.

Riff, M.-C. e Montero, E. (2013). A new algorithm for reducing metaheuristic design effort. *Congress on Evolutionary Computation, Cancún, México*.

Rudberg, M. e Thulin, J. (2009). Centralised supply chain master planning employing advanced planning systems. *Production Planning & Control*.

Saad, M. S., Jamaluddin, H., e Darus, I. Z. M. (2015). Active vibration control of a flexible beam using system identification and controller tuning by evolutionary algorithm. *Journal of Vibration and Control*, 21:2027–2042.

Sadeghi, J. e Niaki, S. T. A. (2015). Two parameter tuned multi-objective evolutionary algorithms for a bi-objective vendor managed inventory model with trapezoidal fuzzy demand. *Applied Soft Computing*, 30:567–576.

Santa-Eulalia, L. A., D'Amours, S., Frayret, J.-M., Menegusso, C. C., e Azevedo, R. C. (2011). *Advanced Supply Chain Planning Systems (APS) Today and Tomorrow,*

Supply Chain Management - Pathways for Research and Practice. InTech, Available from: http://www.intechopen.com/books/supply-chain-management-pathways-for-research- and-practice/advanced-supply-chain-planning-systems-aps-today-and-tomorrow.

Shen, W. (2002). Distributed manufacturing scheduling using intelligent agents. *IEEE Intelligent Systems*.

Shioura, A., Shakhlevich, N. V., e Strusevich, V. A. (2016). Application of submodular optimization to single machine scheduling with controllable processing times subject to release dates and deadlines. *Journal on Computing*, 28(1):148–161. http://dx.doi.org/10.1287/ijoc.2015.0660.

Shobrys, D. E. e White, D. C. (2000). Planning, scheduling and control systems: why can they not work together. *Computers and Chemical Engineering*, 24.

Sinha, A., Malo, P., Xu, P., e Deb, K. (2014). A bilevel optimization approach to automated parameter tuning. Em *GECCO 14*. http://dx.doi.org/10.1145/2576768.2598221.

Smit, S. e Eiben, A. (2009). Comparing parameter tuning methods for evolutionary algorithms. Em *Congress on Evolutionary Computation, 2009. CEC '09. IEEE*.

Smit, S. e Eiben, A. (2010a). Beating the 'world champion' evolutionary algorithm via revac tuning. Em *Congress on Evolutionary Computation (CEC), 2010 IEEE*.

Smit, S. e Eiben, A. (2010b). *Parameter Tuning of Evolutionary Algorithms: Generalist vs. Specialist, chapter Applications of Evolutionary Computation in Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg.

Soto, R., Crawford, B., Olivares, R., Galleguillos, C., Castro, C., Johnson, F., Paredes, F., e Norero, E. (2016). Using autonomous search for solving constraint satisfaction

problems via new modern approaches. *Swarm and Evolutionary Computation*, 30:64–77.

Steger-Jensen, K., Hvolby, H.-H., Nielsen, P., e Nielsen, I. (2011). Advanced planning and scheduling technology. *Production Planning & Control*, 22(8):800–808.

Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., de Oca, M. M., Birattari, M., e Dorigo, M. (2010). Parameter adaptation in ant colony optimization. Technical Report 2, IRIDIA, Institut de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle Université Libre de Bruxelles.

Stützle, T. e Hoos, H. M. (2000). Max–min ant system. *Future Generation Computer Systems*, pgs. 889–914.

Su, C. e Li, H. (2015). A novel interactive preferential evolutionary method for controller tuning in chemical processes. *Chinese Journal of Chemical Engineering*, 23:398–411.

Switalski, P. e Seredynski, F. (2015). Scheduling parallel batch jobs in grids with evolutionary metaheuristics. *Journal of Scheduling*. DOI 10.1007/s10951-014-0382-0.

Tan, K. C., Lee, T. H., e Khor, E. F. (2001). *Incrementing Multi-objective Evolutionary Algorithms: Performance Studies and Comparisons, chapter Evolutionary Multi-Criterion Optimization in Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Tan, K. C., Yang, Y. J., Goh, C. K., e Lee, T. H. (2003). Enhanced distribution and exploration for multiobjective evolutionary algorithms. Em *The 2003 Congress on Evolutionary Computation. CEC '03.*

Tanabe, R. e Fukunaga, A. (2013). Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms. Em *IEEE Congress on Evolutionary Computation.*

Tavares Neto, R. (2010). Proposta de solução de problemas de scheduling considerando possibilidade de terceirização usando a técnica de otimização por colônia de formigas. Master's thesis, Universidade Federal de São Carlos.

Tavares Neto, R. (2012). *Moving to Outsourcing in Production Scheduling: Quantitative Approaches. In: Fabiano E. Molinelli, Leonardo S. Paccagnella. (Org.). Economics of Regulation and Outsourcing.* Nova Publishers.

Tavares Neto, R. e Godinho Filho, M. (2011a). An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. *Computers & Operations Research*, 38:1286 –1293.

Tavares Neto, R. e Godinho Filho, M. (2011b). A software model to prototype ant colony optimization algorithms. *Expert Systems with Applications*, 38:249–259.

Tavares Neto, R. e Godinho Filho, M. (2012). Otimização por colônia de formigas para o problema de sequenciamento de tarefas em uma única máquina com terceirização permitida. *Gestão & Produção.*

Tavares Neto, R. e Godinho Filho, M. (2013). Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*, 26:150–161.

Tavares Neto, R., Godinho Filho, M., e Silva, F. (2015). An ant colony optimization approach for the parallel machine scheduling problem with outsourcing allowed. *Journal of Intelligent Manufacturing*, 26:527–538.

Ugolotti, R. e Cagnoni, S. (2014). Analysis of evolutionary algorithms using multi-objective parameter tuning. Em *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation.*

van Eck, M. (2003). Advanced planning and scheduling is logistics everything? a research on the use(fulness) of advanced planning and scheduling systems. Master's thesis, Vrije Universiteit Amsterdam Faculty of Sciences Mathematics and Computer science departments Paper for Business mathematics and Informatics.

Vecek, N., Mernik, M., Filipic, B., e Crepinsek, M. (2016). Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms. *Information Sciences*, 372:446–469.

Voudourist, C., Owusu, G., Dorne, R., e McCormick, A. (2006). Fos: An advanced planning and scheduling suite for service operations. Em *International Conference on Service Systems and Service Management*, volume 2, pgs. 1138–1143.

Wang, D.-J., Kang, C.-C., Shiau, Y.-R., Wu, C.-C., e Hsu, P.-H. (2015a). A two-agent single-machine scheduling problem with late work criteria. *Soft Comput.* DOI 10.1007/s00500-015-1900-5.

Wang, R., Purshouse, R. C., e Fleming, P. J. (2012). Local preference-inspired co-evolutionary algorithms. Em *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation.*

Wang, S. L., Ng, T. F., Jamil, N. A., e Samuri, S. M. (2015b). Self-adapting approach in parameter tuning for differential evolution. Em *The 2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2015).*

Wolpert, D. H. e Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 1.

Wood, D. A. (2016). Evolutionary memetic algorithms supported by metaheuristic profiling effectively applied to the optimization of discrete routing problems. *Journal of Natural Gas Science and Engineering*, 35:997–1014.

Wu, C.-C., Yin, Y., Wu, W.-H., Chen, H.-M., e Cheng, S.-R. (2015). Using a branch-and-bound and a genetic algorithm for a single-machine total late work scheduling problem. *Soft Comput*, 20(DOI 10.1007/s00500-015-1590-z):1329–1339.

Wu, C.-H., Lee, W.-C., Lai, P.-J., e Wang, J.-Y. (2016). Some single-machine scheduling problems with elapsed-time-based and position-based learning and forgetting effects. *Discrete Optimization*, 19:1–11.

Wu, L. e Cheng, C.-D. (2016). On single machine scheduling with resource constraint. *Journal of Combinatorial Optimization*, 31:491–505. DOI 10.1007/s10878-014-9721-5.

Xingong, Z. e Yong, W. (2016). Two-agent scheduling problems on a single-machine to minimize the total weighted late work. *Journal of Combinatorial Optimization*. DOI 10.1007/s10878-016-0017-9.

Xu, W., Chong, A., Karaguzel, O. T., e Lam, K. P. (2016). Improving evolutionary algorithm performance for integer type multi-objective building system design optimization. *Energy and Buildings*, 127:714–729.

Xu, X. e gen He, H. (2005). *A Theoretical Model and Convergence Analysis of Memetic Evolutionary Algorithms, chapter Advances in Natural Computation in Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Yan, P., Liu, D., Yuan, D., e Yu, J. (2007). Genetic algorithm with local search for advanced planning and scheduling. Em *Third International Conference on Natural Computation*.

Yang, J. e Tang, W. (2009). Preference-based adaptive genetic algorithm for multi-objective advanced planning and scheduling problem. Em *Proceedings of the 2009 IEEE IEEM*.

Yeguas, E., Luzón, M., Pavón, R., Laza, R., Arroyo, G., e Díaz, F. (2014). Automatic parameter tuning for evolutionary algorithms using a bayesian case-based reasoning system. *Applied Soft Computing*, 15:185–195.

Yin, Y., Wang, Y., Cheng, T., Wang, D.-J., e Wu, C.-C. (2016a). Two-agent single-machine scheduling to minimize the batch delivery cost. *Computers & Industrial Engineering*, 92:16–30.

Yin, Y., Xu, J., Cheng, T. C. E. E., Wu, C.-C., e Wang, D.-J. (2016b). Approximation schemes for single-machine scheduling with a fixed maintenance activity to minimize the total amount of late work. *Wiley Online Library (wileyonlinelibrary.com)*. DOI 10.1002/nav.21684.

Yu, M., Zhang, Y., Chen, K., e Zhang, D. (2015). Integration of process planning and scheduling using a hybrid ga/pso algorithm. *Int J Adv Manuf Technol*, 78:583–592.

Yuan, B. e Gallagher, M. (2007). *Combining Meta-EAs and Racing for Difficult EA Parameter Tuning Tasks, chapter Parameter Setting in Evolutionary Algorithms in Studies in Computational Intelligence*. Springer Berlin Heidelberg.

Yuan, J. J., Ng, C. T., e Cheng, T. C. E. (2015). Two-agent single-machine scheduling with release dates and preemption to minimize the maximum lateness. *Journal of Scheduling*, 18(DOI 10.1007/s10951-013-0360-y):147–153.

Zaefferer, M., Breiderhoff, B., Naujoks, B., Friese, M., Stork, J., Fischbach, A., Flasch, O., e Bartz-Beielstein, T. (2014). Tuning multi-objective optimization algorithms for cyclone dust separators. Em *GECCO 14*.

Zeng, F., Low, M. Y. H., Decraene, J., Zhou, S., e Cai, W. (2010). Self-adaptive mechanism for multi-objective evolutionary algorithms. Em *Proceedings of the International MultiConference of Engineers and Computer Scientists*.

Zhang, H. e Gen, M. (2006). Effective genetic approach for optimizing advanced planning and scheduling in flexible manufacturing system. Em *GECCO'06*.

Zhao, C. (2016). Common due date assignment and single-machine scheduling with release times to minimize the weighted number of tardy jobs. *Japan Journal of Industrial and Applied Mathematics*, 33:239–249. DOI 10.1007/s13160-015-0205-5.

Zhao, C., Hsu, C.-J., Wu, W.-H., Cheng, S.-R., e Wu, C.-C. (2016). Note on a unified approach to the single-machine scheduling problem with a deterioration effect and convex resource allocation. *Journal of Manufacturing Systems*, 38:134–140.

Zhao, C. e Tang, H. (2015). Due-window assignment for a single machine scheduling with both deterioration and positional effects. *Asia-Pacific Journal of Operational Research*, 32(1550014):11.

Zhong, R. Y., Li, Z., Pang, L., Pan, Y., Qu, T., e Huang, G. Q. (2013). Rfid-enabled real-time advanced planning and scheduling shell for production decision making. *International Journal of Computer Integrated Manufacturing*.