FERNANDA GONÇALVES SOUZA

Método meta-heurístico de colônia de formigas e sua aplicação na alocação de petróleo

São Paulo

2019

FERNANDA GONÇALVES SOUZA

VERSÃO ORIGINAL

Método meta-heurístico de colônia de formigas e sua aplicação na alocação de petróleo

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Mestre em Ciências.

Área de Concentração:
Engenharia Química

Orientador: Prof. Dr. Song Won Park

São Paulo

2019

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Catalogação-na-publicação

SOUZA, FERNANDA GONÇALVES

Método meta-heurístico de colônia de formigas e sua aplicação na alocação de petróleo / F. G. SOUZA -- São Paulo, 2019.
165 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Química.

1.algoritmo de colônia de formiga 2.scheduling 3.meta-heurística 4.petróleo

I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Química II.t.

SOUZA, Fernanda Gonçalves. **Método meta-heurístico de colônia de formigas e sua aplicação na alocação de petróleo**. 2019. 165p. Dissertação (Mestrado em Ciências e Engenharia Química) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Química. 2019

Aprovado em:

Banca Examinadora

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento _____

RESUMO

A meta-heurística de colônia de formigas vem sendo aplicada desde de 1991 para resolver diversos tipos de problemas matemáticos sendo uma parte destes problemas os de otimização de *scheduling*. A alocação de petróleo de curto prazo, que envolve a escolha do petróleo mais adequado para ser processado nas refinarias ao longo do tempo, pode ser modelado como um problema de *scheduling*. Neste trabalho, foi desenvolvido um modelo, composto por porto, terminal, oleoduto e refinaria, capaz de representar o processo de alocação de petróleo a partir da sua chegada em um navio até a refinaria em um horizonte de tempo de poucos dias. Um algoritmo de colônia de formigas foi implementado e parametrizado para essa aplicação e obteve resultado satisfatório, porém demonstrou-se moroso a medida que a dimensão do problema é aumentada.

Palavras-chave: algoritmo de colônia de formiga, *scheduling*, meta-heurística, petróleo

ABSTRACT

The ant colony metaheuristic has been applied since 1991 to solve several kinds of mathematical problems and part of these problems is the scheduling optimization. Short-term oil allocation, which involves choosing the most suitable oil to be processed at refineries over time, can be modeled as a scheduling problem. In this paper, a model was developed, constituted of port, terminal, oil pipeline and refinery, capable of representing the oil allocation process starting from its arrival on a ship and finishing at the refinery during a time horizon of a few days. An ant colony algorithm was implemented and parameterized for this application and a satisfactory result was obtained, but it gets slower as the dimension of the problem increases.

Keywords: ant colony algorithm, scheduling, metaheuristic, oil

LISTA DE FIGURAS

Figura 1 - Hierarquia do planejamento de abastecimento da Petrobras 13
Figura 2 - Tipo de hidrocarbonetos presentes no petróleo 18
Figura 3 - Histórico da produção nacional de petróleo 20
Figura 4 - Dependência externa de petróleo 21
Figura 5 - Cadeia de suprimento petrolífera 23
Figura 6 - Capacidade nominal de armazenamento de petróleo nos terminais brasileiros 24
Figura 7 - Capacidade de armazenamento de petróleo em refinarias no brasil 26
Figura 8 - Tipos de problemas de decisão 27
Figura 9 - Tipos de funcionalidade de uma ferramenta de scheduling industrial 33
Figura 10 - Funcionalidades de resolução do problema em ferramentas de scheduling industrial 35
Figura 11 - Experimento da ponte dupla 51
Figura 12 - Fluxograma simplificado do ACO 56
Figura 13 - Experimento da ponte dupla 68
Figura 14 - Programa principal da experiência da ponte dupla 69
Figura 15 – Programa da função que calcula as probabilidades e as variáveis do problema 70
Figura 16 - Programa da função que calcula o valor da variável 70
Figura 17 - Resultado do experimento da ponte dupla para distâncias iguais 71
Figura 18 - Evolução da probabilidade no exemplo da ponte dupla com distâncias iguais 72
Figura 19 - Resultado do experimento da ponte dupla para distâncias diferentes 73
Figura 20 - Evolução da probabilidade no exemplo da ponte dupla com distâncias diferentes 73
Figura 21 - Sensibilidade do parâmetro de atualização de feromônio no exemplo da ponte dupla 74
Figura 22 - Sensibilidade do parâmetro alfa no exemplo da ponte dupla 75
Figura 23 - Sensibilidade do parâmetro beta no exemplo da ponte dupla 76
Figura 24 - Escolha de cada formiga na resolução do TSP utilizando ACO 77
Figura 25 - Resultado do TSP utilizando ACO 78

Figura 26 - Resultado do TSP para beta =1 80
Figura 27 - Resultado do TSP para alfa = 0,1 81
Figura 28 - Esquema do scheduling envolvendo múltiplos trabalhos em uma máquina 83
Figura 29 - Código Matlab® do objeto *job* 84
Figura 30 - Resultado da função objetivo e quantidade de aberturas do freezer para cada iteração e o *scheduling* final 85
Figura 31 - Esquema da cadeia de suprimento de petróleo de São Paulo 87
Figura 32 - Exemplo de resultado para a variável "np" 90
Figura 33 - Diagrama de Gantt esquemático 93
Figura 34 - Volumes possíveis no tanque 95
Figura 35 - Fluxograma do algoritmo de *scheduling* 97
Figura 36 - Diagrama esquemático da estrutura orientada a objeto 100
Figura 37 - Código Matlab® do Objeto Oleoduto 102
Figura 38 - Estrutura do código Matlab® para o estudo de caso do modelo do porto 105
Figura 39 – Esquema do Estudo de Caso 1 111
Figura 40 - Esquema do estudo de caso 2 113
Figura 41 - Sensibilidade do parâmetro alfa 116
Figura 42 - Sensibilidade do parâmetro beta 117
Figura 43 - Estrutura do código Matlab® para a parametrização do estudo de caso do modelo do porto 119
Figura 44 - Resultado do Descarregamento dos Navios 123
Figura 45 – Nivel dos Tanques em funcao do tempo 124
Figura 46 - Resultado do ACO para o estudo de caso 1 125
Figura 47 - Resultado do descarregamento dos navios 126
Figura 48 - Nível dos tanques em função do tempo 127
Figura 49 - Envio dos tanques para o oleoduto 128
Figura 50 - Volume de petróleo estocado na refinaria em função do tempo 129
Figura 51 - Resultado da função objetivo do estudo de caso 2 130
Figura 52 - Resultado do ACO para o estudo de caso 2 131

LISTA DE TABELAS

Tabela 1 - Quantidade e extensão de dutos em operação, por função, segundo produtos movimentados 25
Tabela 2 - Sensibilidade do parâmetro beta no TSP 79
Tabela 3 - Sensibilidade do parâmetro alfa no TSP 80
Tabela 4 - Sensibilidade do parâmetro de evaporação dos feromônios 81
Tabela 5 - Dados de hora de chegada e tempo de processamento de cada produto 83
Tabela 6 - Variáveis decisórias 90
Tabela 7 - Notações do modelo 91
Tabela 8 - Propriedades de cada tipo de objeto 100
Tabela 9 - Status possíveis para o navio 106
Tabela 10 – Status possíveis para o píer 106
Tabela 11 - Status possíveis para o tanque 107
Tabela 12 - Classificação dos petróleos em grupamentos 111
Tabela 13 - Sensibilidade do parâmetro alfa 116
Tabela 14 - Sensibilidade do parâmetro beta 117
Tabela 15 - Sensibilidade dos parâmetros de feromônio 118
Tabela 16 - Parametrização do ACO utilizando GA 120
Tabela 17 - Resultados das parcelas da função objetivo 122

SUMÁRIO

1. INTRODUÇÃO 12
2. OBJETIVOS 17
2.1 Objetivo Geral do Estudo 17
2.2 Objetivos Específicos do Estudo 17
2.3 Estruturação da Monografia 17
3. O PETRÓLEO E SUA ALOCACAO NAS REFINARIAS 18
3.1 O Petróleo 18
3.2 Cadeia de suprimento de petróleo 21
3.2.1 Planejamento e Tomada de Decisão 26
3.2.2 Alocação de Petróleo 28
4. REVISÃO BIBLIOGRÁFICA 31
4.1 Ferramenta de Scheduling 32
4.2 Problema Combinatorial 36
4.3 Modelo de Otimização 37
4.3.1 Problema de Satisfação de Restrições 40
4.3.2 Programação Linear 41
4.3.3 Ferramentas Matemáticas 41
4.3.4 Modelo de *Scheduling* 42
4.3.5 Teoria das Filas 44
4.4 Meta-heurística 46
4.5 Método meta-heurístico de colônia de formigas 48
4.5.1 Experimento da Ponte Dupla 50
4.5.2 Variantes do ACO 52
5. Algoritmo Colônia de Formigas 56
5.1 Algoritmo Geral do ACO 56

5.2 Cálculo de Probabilidades do ACO 57
5.3 Atualização e Evaporação dos Feromônios 59
5.4 Informação Heurística 62
5.5 Critério de parada 63
5.6 Vantagens e Desvantagens do Método 64
5.7 ACO e Problemas de *Scheduling* 65
6. Avaliação Preliminar do Algoritmo de Colônia de Formigas 67
6.1 Experimento da Ponte Dupla 67
6.1.1 Caminhos de Mesmo Comprimento 70
6.1.2 Caminhos de Comprimentos Distintos 72
6.1.3 Sensibilidade dos Parâmetros 73
6.2 Problema do Caixeiro Viajante 76
6.2.1 Validação dos Resultados 77
6.2.2 Sensibilidade dos Parâmetros 78
6.3 Problema de *scheduling* envolvendo múltiplos trabalhos em uma máquina 81
7. METODOLOGIA 86
7.1 Modelo Geral 88
7.1.1 Definição das Variáveis do Modelo 89
7.1.2 Representação do Domínio do Tempo 91
7.1.3 Caracterização do Petróleo 93
7.1.4 Navios 93
7.1.5 Escolha do Píer para Atracação 94
7.1.6 Escolha do Tanque 94
7.1.7 Passagem para o Oleoduto e Refinaria Destino 95
7.1.8 Estrutura do Modelo de Otimização 96
7.1.9 Função Objetivo 98

7.2 Principais Aspectos da Implementação no Matlab® 98
7.3 Restrições do Modelo 105
7.4 Algoritmo de Colônia de Formigas 109
7.5 Estudo de Caso 1 111
7.6 Estudo de Caso 2 112
8. RESULTADOS E DISCUSSÃO 114
8.1 Parametrização 114
8.2 Estudo de Caso 1 120
8.2.1 Desempenho do ACO no Estudo de Caso 1 124
8.3 Estudo de Caso 2 125
8.3.1 Desempenho do ACO no Estudo de Caso 2 129
9. CONCLUSÃO 132
10. RECOMENDAÇÕES DE PESQUISA FUTURA 134
REFERÊNCIAS 135
APÊNDICE 139
Apêndice A - Código Matlab® para A Ponte Dupla 139
Apêndice B - Código Matlab® para TSP 141
Apêndice C - Dados de Entrada dos Estudos de Caso 143
Apêndice D - Código Matlab® para os Estudos de Caso 145
Apêndice D.1 – Código dos Objetos do Estudo de Caso 145
Apêndice D.1.1 – Código do Objeto: pet 145
Apêndice D.1.2 – Código do Objeto: classe 146
Apêndice D.1.3 – Código do Objeto: navio 146
Apêndice D.1.4 – Código do Objeto: pier 148
Apêndice D.1.5 – Código do Objeto: tq 148
Apêndice D.1.6 – Código do Objeto: oleod 149

Apêndice D.1.7 – Código do Objeto: ref 151
Apêndice D.2 – Código do *Scheduling* do Estudo de Caso.....151
Apêndice D.2.1 – Código do *Scheduling*: Arquivo Porto_Discreto.m 151
Apêndice D.2.2 – Código do *Scheduling*: Arquivo Inicializacao_Objetos.m155
Apêndice D.2.3 – Código do *Scheduling*: Arquivo CalculoProbabilidades.m157
Apêndice D.2.4 – Código do *Scheduling*: Arquivo prob_gen.m.....159
Apêndice D.2.5 – Código do *Scheduling*: Arquivo prob_ans.m.....160
Apêndice D.2.6 – Código do *Scheduling*: Arquivo roleta.m.....161
Apêndice D.2.7 – Código do *Scheduling*: Arquivo oleod_ref.m.....161
Apêndice D.2.8 – Código do *Scheduling*: Arquivo tq_para_oleod.m 161
Apêndice D.2.9 – Código do *Scheduling*: Arquivo navio_para_tq.m 162
Apêndice D.2.10 – Código do *Scheduling*: Arquivo go_amax.m 163
Apêndice D.2.11 – Código do *Scheduling*: Arquivo BalancoMaterial.m ...164
Apêndice D.3 – Código do ACO do Estudo de Caso 164

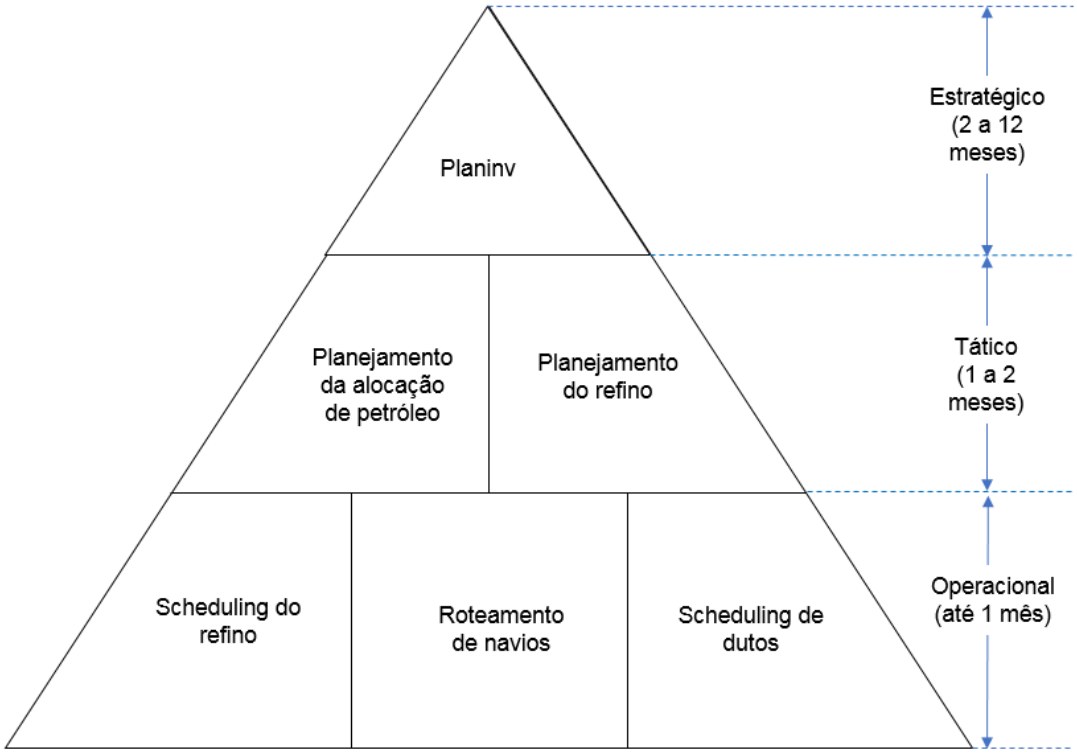
1. INTRODUÇÃO

A partir da década de 60, modelos e sistemas para a otimização e planejamento das atividades de refino e da logística de petróleo e derivados começaram a ser desenvolvidos. Tais sistemas são complexos devido à grande quantidade de variáveis, seu alto grau de relacionamento e à não linearidade dos processos nas refinarias (Oliveira e Hamacher, 2007). Continuamente, as grandes empresas desse setor buscam desenvolver a sua área de planejamento por ser um ponto chave no sucesso de suas atividades, tanto pelo ponto de vista financeiro quanto pelo da segurança organizacional.

Para assegurar um planejamento adequado de atividades no meio corporativo, surgiu a área da pesquisa operacional, a qual realiza a quantificação de fatores para viabilizar o processo de tomada de decisões. Ela utiliza modelagens e simulações de sistemas reais para melhor alocar os recursos disponíveis. (Marins, 2011). A pesquisa operacional (também conhecida como investigação operacional) é utilizada em diversas áreas, tais como: logística, telecomunicações, produção e finanças. Esse ramo da matemática aplicada se fundamenta basicamente nas técnicas de otimização, sua execução pode acarretar diversas dificuldades na etapa de modelagem e no gerenciamento dos sistemas computacionais. Para atender a estas demandas, existem diversos sistemas de modelagem que atuam como ferramentas de apoio à decisão, os quais facilitam o processo de implementação e solução dos modelos utilizando linguagem algébrica (Oliveira e Hamacher, 2007).

A hierarquia de decisões de uma indústria química ou petroquímica segue a seguinte ordem: planejamento estratégico, planejamento tático, planejamento de operações do sistema, planejamento de operação da planta, programação de produção e controle e otimização de processos em tempo real (Moro, 2000). Assim, para a modelagem de um sistema de programação de produção, é necessário serem conhecidas e entendidas todas as etapas hierárquicas anteriores a ela; na Figura 1 (Nunes *et al.*, 2010), pode ser visualizada essa divisão de hierarquia. No nível estratégico, que envolve o período de vários meses a anos, a Petrobras utiliza o Planinv, que consiste em um otimizador linear que possui uma visão geral da empresa, utilizado como base para decisões sobre investimentos.

Figura 1 - Hierarquia do planejamento de abastecimento da Petrobras



Outros autores também trabalham com divisões hierárquicas similares ao apresentado na Figura 1. Neiro (Neiro, 2003) retrata um modelo linear genérico de planejamento de produção, envolvendo três níveis de decisão envolvendo um grande número de variáveis discretas, formando assim um número finito de configurações possíveis. Os 3 níveis são: projeto, expansão de capacidade e operação. Além dos 3 níveis hierárquicos de decisão, ilustrados na Figura 1, alguns autores ainda acrescentam um nível ainda mais na base do triângulo, que é o nível de despacho ou liberação. Esse nível é responsável por decisões em algumas horas ou poucos dias (Shimizu, 2010).

A alocação de petróleo é a primeira etapa do plano tático, e é integralmente dependente da produção de petróleo pelas plataformas e campos terrestres, bem como da infraestrutura logística responsável pelas movimentações da matéria-prima. A adequada alocação irá subsidiar a produção de derivados na qualidade e quantidade desejada de modo a atender o mercado nacional e internacional, e sua falha pode acarretar tanto em diminuição de carga de refinarias até sua parada.

Uma programação adequada de produção proporciona uma associação compatível entre as resoluções de curto, médio e longo prazo e o correto desmembramento das definições do planejamento nas unidades de refino e operações logísticas. A programação de produção se tornou o elo fraco entre duas etapas, mas computacionalmente bem desenvolvidas, que é planejamento de operação de curto prazo nas unidades de processamento e a otimização de processos.(Moro, 2000). Para o êxito de uma aplicação de programação matemática, é fundamental a escolha do sistema de modelagem apropriado; ele deve ser abrangente, com boa interface homem-máquina e simples manutenção.(Oliveira e Hamacher, 2007).

Na Petrobras, o principal sistema utilizado é o PLANAB. O PLANAB (Planejamento do Abastecimento) é uma ferramenta de otimização linear utilizada para fornecer uma estratégia adequada às refinarias, fornecendo indicações da alocação de petróleo, da carga, além da produção de cada uma das principais unidades que compõem as refinarias e o ganho com vendas de petróleo e derivados. Para tal, são necessários diversos dados de entrada, como a previsão de petróleos disponíveis, dos preços de derivados, do mercado, da disponibilidade das unidades de processo, da qualidade dos petróleos, entre diversos outros. Essa ferramenta é utilizada para gerar resultados de curto ou longo prazo, além de também servir para realização de estudos de caso. Cada refinaria tem uma ferramenta própria de otimização, sendo que a maioria delas utiliza o PIMS (*Process Industry Modeling System*) da AspenTech que, assim como o PLANAB, também é uma ferramenta de programação linear e é baseado em rendimentos constantes de cada unidade de processo obtidos por simulação. Mensalmente, todas as refinarias executam este programa de acordo com as indicações do PLANAB, de modo a gerar a programação de cada unidade. Uma outra ferramenta de planejamento de produção é o RPMS (*Refinery and Petrochemical Modeling System*) da Honeywell. Ferramentas computadorizadas de programação de produção (CAS, do inglês *Computing Aided Scheduling*), podem diminuir o tempo necessário para a elaboração de um plano de produção em cerca de 50%. São poucas as ferramentas comerciais disponíveis atualmente, e a carência dessas ferramentas computacionais é a principal causa da falta de harmonia entre os objetivos do planejamento de produção com a operação de processos.(Moro, 2000)

Incertezas como mau tempo e atraso de navios, são algumas das motivações para o desenvolvimento de ferramentas rápidas de resolução de problemas, que possam auxiliar ou suportar decisões tomadas emergencialmente relacionadas com a alocação de petróleo. Segundo Moro (Moro, 2000), na indústria de petróleo são utilizadas ferramentas lineares, pois as incertezas de dados como demanda, preços, qualidade, rendimentos e disponibilidades, são tão expressivas que desestimulam o desenvolvimento de tecnologias mais robustas. Diante disso, os programas atuais focam em achar soluções viáveis.

O problema da alocação de petróleo pode ser representado por um conjunto de cenários possíveis, em função, principalmente, dos petróleos disponíveis e na hora da disposição dos mesmos. Os diferentes tipos de petróleo podem ser disponibilizados tanto através de produção própria, como de produção de terceiros ou importações. A estratégia de alocação desses petróleos depende, além de outros fatores, da produção de petróleo, da qualidade do mesmo, do valor agregado a ele (referente à competição do seu valor de exportação em relação a agregação de valor ao refiná-lo no Brasil) e aos rendimentos dessa matéria-prima em cada refinaria do sistema.

De acordo com Moro (Moro, 2000), programação de produção pode ser elaborada manualmente ou de modo semiautomático, sendo o último através de princípios heurísticos. Silva, Sousa e Runkler (Silva, Sousa e Runkler, 2008) acrescentam que as meta-heurísticas são consideradas fortes ferramentas de *scheduling* e com aptidão para aplicação em sistemas logísticos e, em comparação a outras metodologias meta-heurísticas, o ACO já se mostrou mais eficiente para sistemas dinâmicos do que algoritmos genéticos, por exemplo.

Algoritmos meta-heurísticos inspirados em comportamentos da natureza, como o algoritmo de colônia de formigas (ACO), são cada vez mais empregados para solucionar problemas de otimização já que tem a vantagem de, assim como outros algoritmos meta-heurísticos, possuir um tempo de resolução geralmente aceitável (Tavares Neto e Godinho Filho, 2013). Os métodos ACO são úteis para resolver problemas de otimização discretos que são solucionados através da escolha do melhor caminho. O algoritmo já foi aplicado com sucesso para resolver problemas

clássicos de otimização, como o problema do caixeiro viajante, o problema da mochila, o *job shop scheduling*, o problema quadrático de alocação, o *scheduling* de tarefas em rede e na nuvem, entre outros (Kalra e Singh, 2015).

Esse presente estudo trata-se do desenvolvimento de uma ferramenta de otimização de curto prazo para auxiliar ou fundamentar tomadas de decisão no setor de alocação de petróleo do sistema petrolífero do estado de São Paulo. A meta-heurística de colônia de formigas foi escolhida para este estudo por suas características de trabalhar com sistemas em forma de grafo, além de sua eficiência na resolução de determinados problemas de *scheduling*.

2. OBJETIVOS

2.1 OBJETIVO GERAL DO ESTUDO

Otimização, pelo algoritmo de colônia de formigas, da alocação de petróleo.

2.2 OBJETIVOS ESPECÍFICOS DO ESTUDO

Entender, no contexto da cadeia de suprimento de petróleo, a utilização de ferramentas de tomadas de decisão de curto prazo aplicadas à alocação de petróleo. Para isso, parte-se do empenho em adequar um sistema real a um modelo simplificado que possa ser resolvido em tempo computacional hábil ao escolher um método de resolução coerente.

Compreender a complexidade do processo de tomada de decisão em uma grande empresa e como ferramentas apropriadas são capazes de fomentar a gestão e operação de parte da cadeia logística.

Elaborar um modelo utilizando o algoritmo de colônia de formigas capaz de auxiliar o processo de alocação de petróleo ou servir de base para elaboração de modelos mais complexos.

Interpretar os resultados do modelo implementado e da parametrização necessária para viabilizar o uso da ferramenta e aprimorá-la.

2.3 ESTRUTURAÇÃO DA MONOGRAFIA

Este trabalho foi estruturado da seguinte forma: o capítulo 3 numa apresentação dos conceitos que envolvem o problema da alocação de petróleo; o capítulo 4 consiste de uma revisão bibliográfica dos principais tópicos abordados nesse trabalho; o capítulo 5 discute o algoritmo de colônia de formigas; o capítulo 6 presenta algumas aplicações do algoritmo de colônia de formigas; o capítulo 7 apresenta a metodologia utilizada e a motivação para a escolha; o capítulo 8 discute os resultados do trabalho; o capítulo 9 apresenta as conclusões; o capítulo 10 as recomendações para futuros trabalhos.

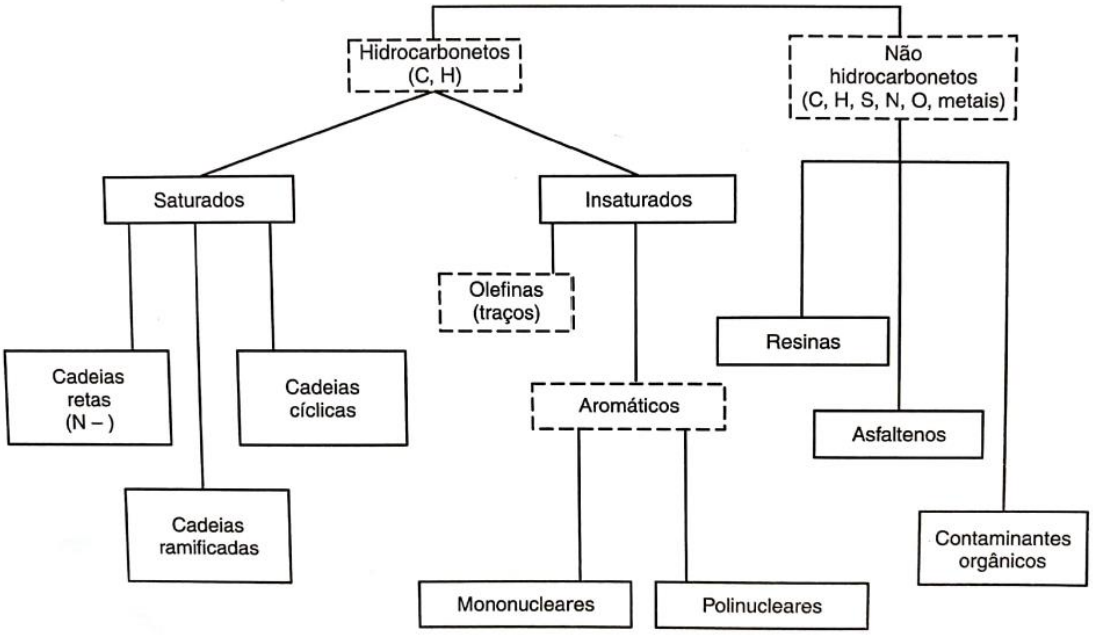
3. O PETRÓLEO E SUA ALOCACAO NAS REFINARIAS

3.1 O PETRÓLEO

O petróleo é uma mistura de líquidos, gases e sólidos em forma de dispersão. A principal classe de substâncias no petróleo são os hidrocarbonetos com 1 a 60 átomos de carbono, os quais correspondem a mais de 90% das substâncias presentes no óleo. Apesar de todos os petróleos possuírem praticamente a mesma composição elementar (com uma proporção similar de átomos de carbono, hidrogênio e oxigênio) suas propriedades se diferem muito, principalmente em relação a densidade e viscosidade (Farah, 2012). A classificação dos tipos de hidrocarbonetos presentes no petróleo é mostrada na

Figura 2.

Figura 2 - Tipo de hidrocarbonetos presentes no petróleo



Fonte: Reproduzido de Farah (2012)

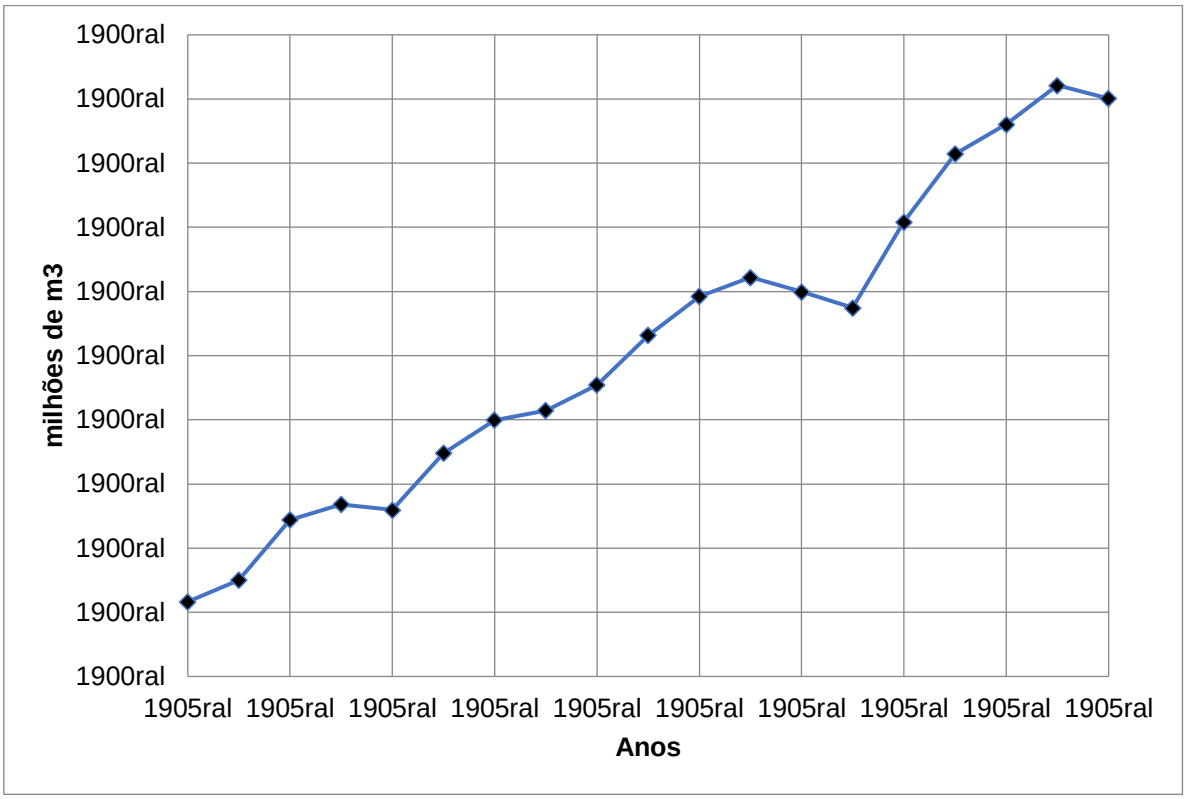
As características químicas e físicas do petróleo (qualidade intrínseca) em conjunto com o processamento em refinaria definem a qualidade de seus derivados. Já a estabilidade do petróleo e de suas misturas depende do equilíbrio entre seus componentes, principalmente entre as resinas e os asfaltenos. As resinas mantêm

os asfaltenos dispersos, impedindo que eles se aglomerem e precipitem (Farah, 2012).

De acordo com Farah (2012), o petróleo é utilizado pela sociedade há cerca de 6 mil anos para impermeabilização, preparo de alimentos, iluminação e aquecimento. No século XIX, grande parte da iluminação pública era realizada utilizando óleo de baleia, até que, em meados do mesmo século, no Canadá, foi desenvolvido uma metodologia de iluminação utilizando querosene, dando início a indústria de petróleo. Mais tarde, em 1887, foi inventado o motor à explosão, o qual consumia outras frações do petróleo como diesel e gasolina, agregando muito mais valor ao “óleo negro” e aumentando sua produção. A partir do século XIX, com o surgimento das primeiras refinarias, cada vez mais aplicações foram identificadas para as diversas frações de petróleo. Entre as aplicações se destacam: o uso de derivados de combustíveis, como os automotivos (gasolina e diesel), de aviação (gasolina e querosene de aviação); a aplicação como outras fontes de energia como o GLP (gás liquefeito de petróleo) e o GN (gás natural); para uso doméstico (gás de cozinha); combustível marítimo (óleo diesel e óleo combustível); combustível industrial (gás, óleo combustível e coque); uso de lubrificantes, parafina, solventes, asfaltos, matérias-primas para outras indústrias, entre outros.

Segundo dados da ANP, a produção de petróleo nacional tem crescido ao longo dos últimos anos. A produção total de petróleo no Brasil, englobando tanto a produção em terra quanto em mar, pode ser visualizada na Figura 3, sendo que cerca de 89% do petróleo produzido nesse período (entre 2000 e 2016) é oriundo da região Sudeste do país.

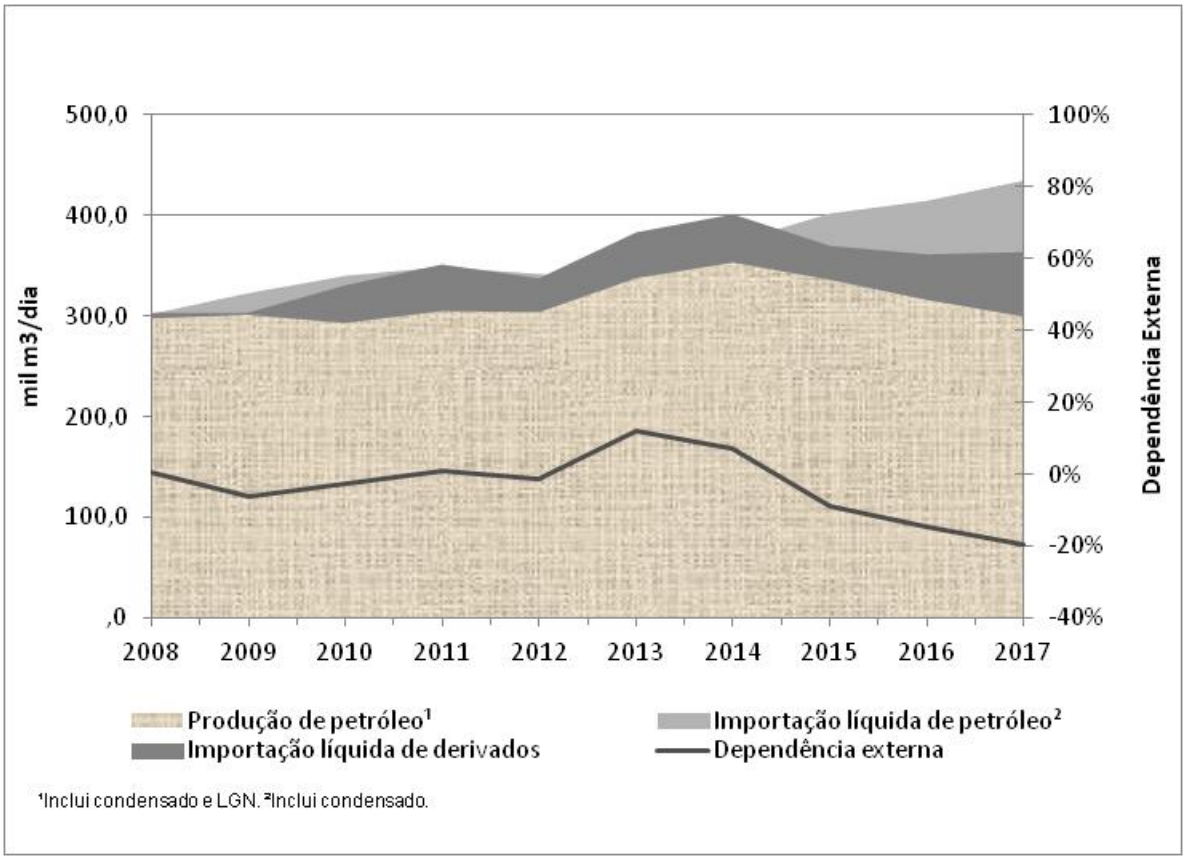
Figura 3 - Histórico da produção nacional de petróleo



Fonte: ANP

Com o aumento da produção de petróleo, nossa dependência com o petróleo importado diminuiu, como pode ser visto na Figura 4, em tonelada equivalente de petróleo (toe), que mostra os dados da EPE (Empresa de Pesquisa Energética) de 2016. Todavia, ainda é necessário a importação de alguns tipos de petróleo, os quais possuem características intrínsecas específicas desejadas para a produção de certos derivados, como os lubrificantes na REDUC.

Figura 4 - Dependência externa de petróleo



Fonte: ANP

3.2 CADEIA DE SUPRIMENTO DE PETRÓLEO

Uma cadeia de suprimentos liga uma empresa desde a fonte inicial de matéria-prima até o consumo do produto acabado (Pires, 2009). No caso da Petrobras, portanto, tal processo envolve desde a extração do petróleo e gás nos campos marítimos e terrestre, ou a sua importação, até o abastecimento de veículos e casas pelos combustíveis gerados, considerando apenas as principais áreas de atuação da empresa.

Segundo Marcellino (2013), o projeto, a análise e o desempenho de cadeias de suprimentos estão em foco em muitas empresas, e essa cadeia é constituída de dois processos principais:

- Planejamento de produção e controle de estoque: a parte do planejamento engloba todo gerenciamento do processo de produção envolvendo a matéria-

prima e seu processamento, ou seja, a aquisição e alocação de matéria-prima, escalonamento do processo de produção. Já a parte de controle de estoque, incorpora a administração do inventário através de políticas de estoque tanto de matérias-primas quanto de produtos intermediários e de produtos finais. Assim, o planejamento de produção e controle de estoque compreendem os subprocessos de produção e armazenagem, e suas interfaces.

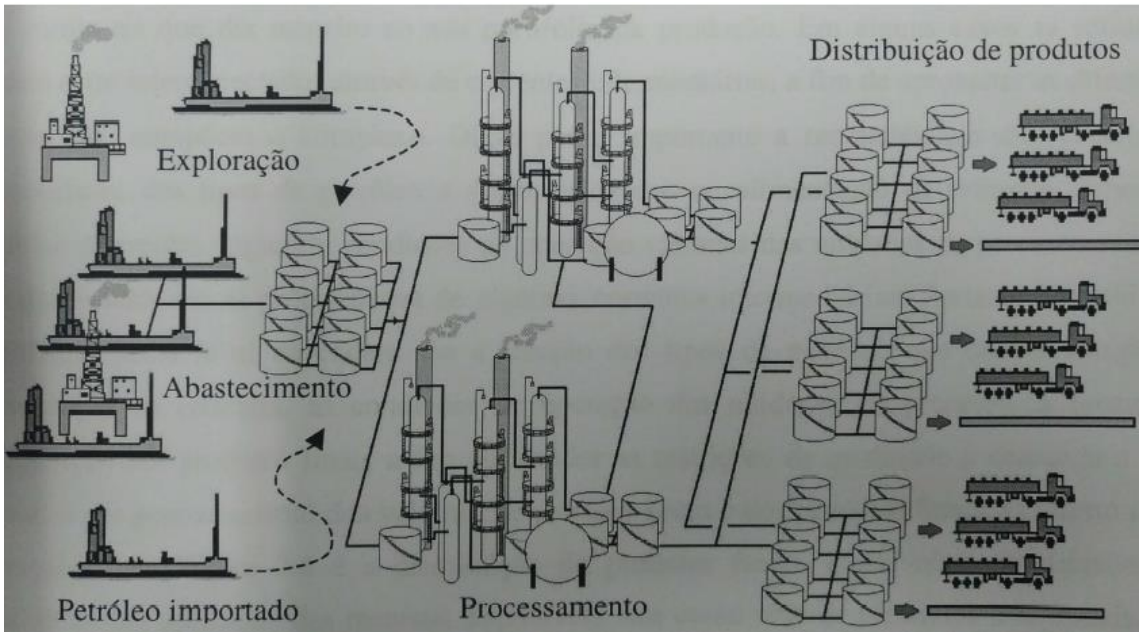
- Processo de distribuição e logística: envolve uma fase mais na frente da cadeia, que é o modo pelo qual os produtos gerados irão se dirigir aos revendedores. Essa movimentação pode ser realizada de modo direto ou através de centros de distribuição. O processo de distribuição e logística engloba, portanto, o gerenciamento da remoção, transporte e entrega do produto final.

Para Pires (2009), os estudos e aplicações na área de logística (que é um subconjunto da gestão da cadeia de suprimentos) existem há muito tempo (desde a Antiguidade) em nossa sociedade, tendo sido responsável por importantes vitórias e derrotas de civilizações inteiras. A gestão da produção vem ganhando mais espaço dentro das empresas industriais e alguns paradigmas foram quebrados, com destaque dos dois a seguir:

- Agregação de valor: antes era considerado que o valor agregado a um determinado produto era determinado apenas pelo processamento por qual ele passou dentro da cadeia de suprimento. Atualmente há uma análise mais criteriosa da cadeia de valor, agregando também os valores logísticos, que não mais são vistos como puros processos de desperdícios que devem ser minimizados.
- Meios e fins: os processos de uma cadeia de suprimentos eram simploriamente classificados como sendo um processo meio ou um processo fim, ou seja, os meios são os que suportam e viabilizam os chamados fins como vender, produzir e entregar. Assim, os processos (meios) eram desvalorizados em relação aos *fins*, o que hoje sabe-se não ser verdade, a exemplo dos casos de sucesso baseados em desenvolvimentos logísticos, da Dell e Wal-Mart.

O conceito de gestão da cadeia de suprimentos surgiu, segundo alguns autores, no começo dos anos 80 a partir de consultores coorporativos ou, ainda, na década de 70, para a maior integração entre a gestão de estoques e a do setor de transportes. Contudo o interesse nessa temática só começou a aumentar na década de 90 (Pires, 2009). As operações de transporte, na maioria das empresas, representam o maior gargalo na cadeia produtiva já que deve apresentar máxima harmonia com as outras etapas de processamento. Um esquema da cadeia de suprimento petrolífera é apresentado na Figura 5 (Pinto, 2000). A exploração do petróleo pode ser em campos marítimos (*offshore*), ou em campos terrestres (*onshore*). Alternativamente, há também a opção de importar petróleo. O petróleo produzido ou importado é então levado para as refinarias por meio de navios, dutos, caminhões, ou uma combinação desses meios de transporte. Este percurso pode também envolver terminais marítimos ou terrestres e subestações de bombeamento.

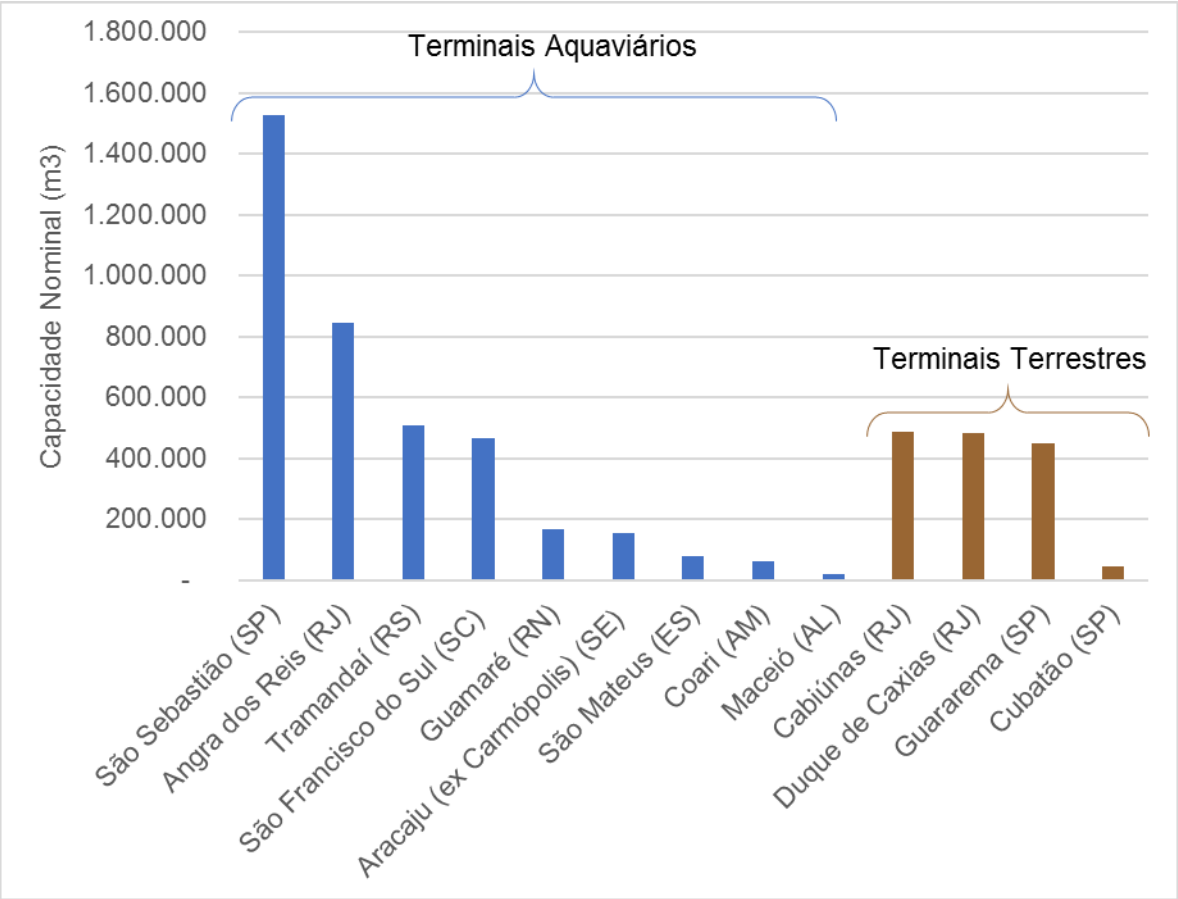
Figura 5 - Cadeia de suprimento petrolífera



Fonte: Pinto (2000)

O Brasil possui terminais terrestres e aquaviários que atendem a necessidade de diversos setores da indústria e comércio. Segundo dados da ANP, a capacidade nominal de armazenamento de petróleo em terminais brasileiros é de 5.298.090 m^3, sendo destes, 3.832.996 m^3 em terminais aquaviário e 1.465.093 m^3 em terminais terrestres, distribuídos conforme a Figura 6.

Figura 6 - Capacidade nominal de armazenamento de petróleo nos terminais brasileiros



Fonte: ANP

Navios petroleiros são responsáveis pelo transporte de petróleo bruto, que receberam um pré-tratamento na plataforma, até os terminais marítimos de armazenagem e uma complexa rede de oleodutos transporta cada petróleo para a refinaria correspondente. Tipos de petróleo diferentes requerem condições particulares de operação das plantas das refinarias e geram produtos com características diferentes (Neiro, 2003). Uma característica interessante na rede de transporte de petróleo é que ela é, geralmente, exclusiva, ou seja, nenhum outro produto é transportado pelos mesmos dutos ou armazenados nos mesmos tanques que o petróleo para evitar risco de contaminação ou combinações indesejadas de propriedades. Os produtos finais podem ser distribuídos por dutos ou veículos de transporte rodoviário, quando o volume for pequeno (Neiro, 2003). Os dutos permitem o transporte de grandes quantidades de produtos e matérias-primas consumindo poucos recursos, sendo estes apenas energia para o bombeamento e

gastos com manutenção, e em curtos intervalos de tempo (Pinto, 2000). Um oleoduto é um tipo de duto, definido pela ANP como:

> Dutos Terrestres de Transporte ou Transferência que movimentam: a) Petróleo, líquidos de gás natural, condensado, derivados líquidos de petróleo e gás liquefeito de petróleo; b) Todos os produtos líquidos cujas operações de movimentação sejam reguladas pela ANP, exceto gases liquefeitos por baixa temperatura. Os Dutos que movimentam hidrocarbonetos líquidos e outros combustíveis tais como, biodiesel, mistura óleo diesel /biodiesel ou etanol, de forma sequencial (polidutos), são considerados Oleodutos. - Regulamento técnico de dutos terrestres para movimentação de petróleo, derivados e gás natural (RTDT 2010).

Segundo dados da EPE, os dutos de derivados de petróleo e de etanol somaram 6.036 km ao final de 2016 (apenas 8 km superior ao montante 2015), sendo 4.834 km de transporte (80%), e 1.202 km de transferência. A infraestrutura nacional dutoviária para movimentação de petróleo e derivados é composta de 454 dutos. A malha de dutoviária brasileira (incluindo oleodutos) é mostrada em números na Tabela 1, na qual outros representa dutos para movimentação de etanol anidro, etanol hidratado, aguarrás e metanol, etano e propano de insumo para petroquímica, gasolina de pirólise e propileno de insumo para indústria petroquímica.

Tabela 1 - Quantidade e extensão de dutos em operação, por função, segundo produtos movimentados

Produtos movimentados	Dutos em operação		
	Função	Quantidade	Extensão (km)
Derivados	Transferência	324	1.157
	Transporte	98	4.794
Gás natural	Transferência	63	2.274
	Transporte	47	9.422
Petróleo	Transferência	32	1.985
Outros	Transferência	32	37
	Transporte	5	40
Total		601	19.709

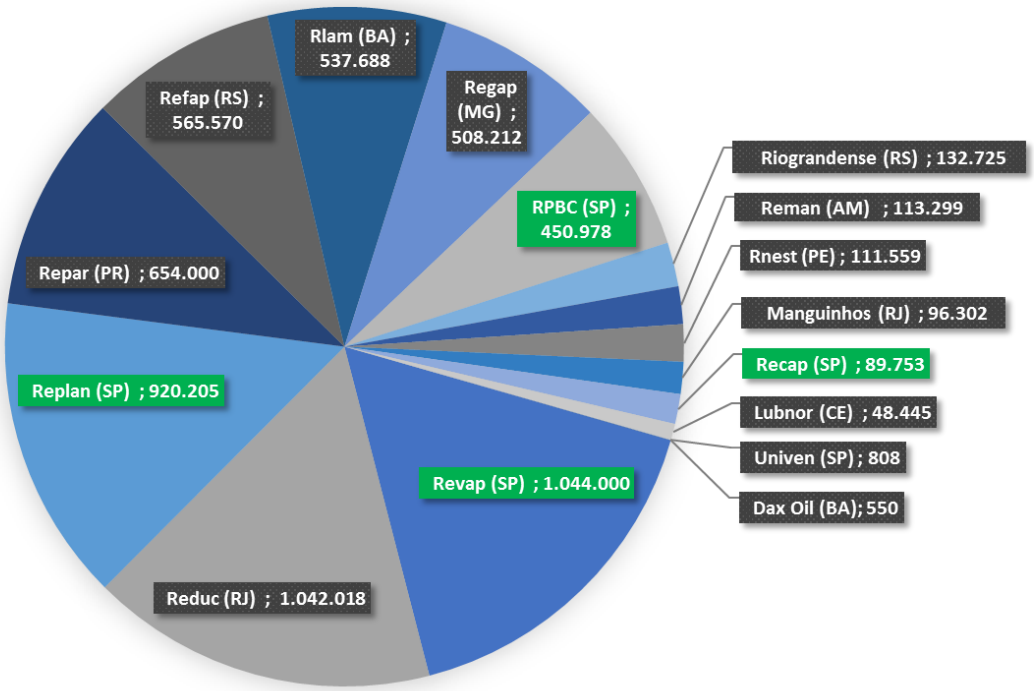
Fonte: ANP

Na refinaria, o óleo cru é, geralmente, armazenado em tanques cilíndricos de teto flutuante, já que eles apresentam menores perdas de componentes voláteis do conteúdo do que tanques de teto fixo. Contudo, eles necessitam de um lastro de cerca de 20% de sua capacidade. Após o tanque receber o petróleo, é necessário

deixá-lo sedimentar por um certo período de tempo (comumente 24h) para separar a emulsão formada da salmoura no óleo (Moro, 2000).

O parque de refino brasileiro é composto por 17 refinarias presentes em quase todas as regiões do país, mas concentradas na região sudeste, principalmente em São Paulo. O armazenamento de petróleo nas refinarias é de 6.316.112 m^3, segundo dados da ANP; deste total, 40% do armazenamento acontece no estado de São Paulo, distribuído de acordo com a Figura 7.

Figura 7 - Capacidade de armazenamento de petróleo em refinarias no brasil



Fonte: ANP

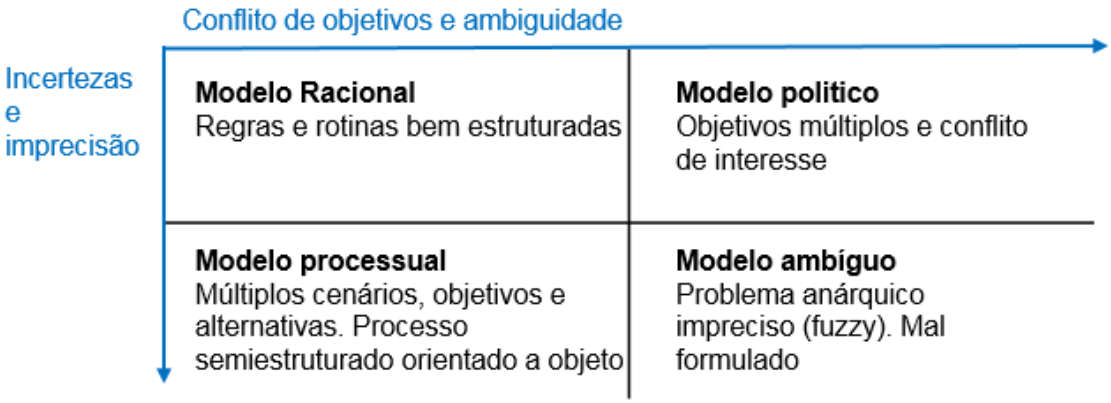
3.2.1 PLANEJAMENTO E TOMADA DE DECISÃO

Para Shimizu (2010), indivíduos possuem liberdade na tomada de decisões e podem se valer de diversas premissas particulares sem critérios bem definidos, enquanto as decisões empresariais envolvem uma maior complexidade de fatores, além de muitos riscos e incertezas. Assim, decisões no nível empresarial devem ser formais, detalhadas, consistentes e transparentes.

Todas as unidades organizacionais estão submetidas a incertezas e, para diminuir o efeito negativo dessas incertezas, é necessário dar ênfase à rapidez das ações e investir em retroalimentações a curto prazo unida com aprendizagem e adaptação constante, distanciando-se, o máximo possível, dos riscos associados às estratégias e decisões de longo prazo. Ademais, quanto mais se adotar decisões no dia a dia, menos a empresa ficará suscetível aos riscos associados a previsões, como, por exemplo, as de mercado, produção, câmbio ou até mesmo clima. Contudo, é importante salientar que a minimização dos riscos nem sempre é traduzida em uma maximização dos lucros (Shimizu, 2010).

Os problemas de tomada de decisão podem ser classificados, de acordo com o nível de incertezas e nitidez dos objetivos, em quatro modelos, como mostrado na Figura 8. Tais modelos não possuem fronteiras bem delimitadas e um mesmo problema pode ser classificado como sendo de mais um tipo.

Figura 8 - Tipos de problemas de decisão



O desenvolvimento de modelos de planejamento e produção do sistema Petrobras, assim, como em qualquer outro empreendimento, é fundamental para o melhor aproveitamento de recursos da empresa. O controle e gestão de estoque é fundamental, pois é necessário encontrar uma faixa aceitável da quantidade e local de inventário para atender as necessidades financeiras e logísticas da operação. Inventário escasso remete a falta de produto (ou matéria-prima), o que pode causar prejuízos imensos, já que o não atendimento de um cliente gera prejuízos financeiros diretos e pode também abalar a imagem e relacionamentos da companhia. Grandes inventários são custosos tanto do ponto de vista físico quanto

do operacional, portanto também deve ser evitado, restando uma faixa estreita de operação que é decidida pela política de estoque.

Para realizar o planejamento da produção de uma indústria química ou petroquímica é necessário o conhecimento em cinco setores: topologia do processo, capacidade da planta, restrições de operação, demanda de mercado e custos envolvidos (Neiro, 2003). A modelagem de uma cadeia de suprimento de petróleo, ou parte dela, requer a familiaridade com cada um desses cinco setores e o encadeamento entre elas.

3.2.2 ALOCAÇÃO DE PETRÓLEO

A vazão de carga das refinarias é definida no nível corporativo como consequência da otimização do lucro da empresa como um todo considerando as diversas restrições do sistema e o atendimento obrigatório de alguns mercados. O processo de alocação de petróleo em cada refinaria é intimamente ligado com a qualidade de acordo com o trecho a seguir:

> As refinarias têm por objetivo produzir derivados para atender o mercado, com máxima rentabilidade, de forma segura e responsável ambientalmente. Para que esse objetivo seja alcançado, o tipo de petróleo a ser processado necessita ser definido em função desses objetivos e dos processos de refino existentes na refinaria. Assim, surge a necessidade de qualificar o petróleo para sua alocação nas refinarias (Farah, 2015).

O planejamento estratégico e tático de uma empresa, além de lidar com um horizonte de tempo diferente do *scheduling*, muitas vezes persegue objetivos diferentes. Enquanto o planejamento foca nos aspectos tecnológicos do processo, o *scheduling* enfatiza a variável tempo, mas deve haver uma integração entre os dois. Assim, o planejamento do processo se torna uma restrição inevitável para o *scheduling*.(Srinivas, Raju e Rao, 2012) Como o planejamento costuma envolver uma área mais ampla de atuação, envolvendo diversos aspectos do negócio, o *scheduling* torna-se uma ferramenta mais específica e precisa, porém sem a visão global do sistema estudado no planejamento. A alocação de petróleo é vista sobre esses dois aspectos:

- Planejamento: define quais petróleos são mais adequados para serem utilizados como carga de cada destilação das diversas refinarias do sistema.

Os principais aspectos considerados nessa fase são as propriedades típicas de cada tipo de petróleo, o mercado de derivados nacional e internacional que subsidia a escolha de quais produtos serão gerados e onde, as capacidades técnicas de cada unidade de processamento e o sistema logístico disponível para a realização das operações de transporte.

- *Scheduling*: deve considerar a qualidade de cada lote específico de petróleo e quais tratamentos ou mistura ele irá requerer nos terminais para posterior envio às refinarias, disponibilidade atual de tanques, bombas e dutos, hora de chegada e atracação de navios, tempo de bombeamento, a campanha executada na unidade em cada período e a velocidade de consumo de óleo pelas refinarias.

A Petrobras é uma empresa de grande porte do setor de petróleo e energia e atua nas áreas de: exploração e produção de petróleo e gás, refino, oferta de gás natural, distribuição, petroquímica e fertilizantes, geração de energia elétrica, produção de biocombustíveis e transporte e comercialização; e opera em bacias de petróleo, refinarias, terminais e dutos, termelétricas, gasodutos e fábricas de fertilizantes. Sua cadeia de suprimento é complexa e envolve, entre outros setores, o setor de petróleo, produtos químicos e derivados diversos. Na Petrobras, o plano do abastecimento é gerado por uma ferramenta de programação linear denominada PLANAB. A partir dela é gerado mensalmente o planejamento de refino, transporte e comercialização (de sigla RTC) com o horizonte de um ano. Essa ferramenta é também utilizada para a realização de cenários de sensibilidade para subsidiar tomadas de decisão e realizar planejamentos condicionados a incertezas inerentes do processo. O PLANAB é provido de diversos dados de entrada relacionados com a disponibilidade de matéria-prima e ativos e a demanda e mercado nacional e internacional. Munido desse plano mensal, cada refinaria do sistema opera seu otimizador local o PIMS (*Process Industry Modeling System*), que fornece dados mais detalhados da operação de cada unidade de refino alinhado com o PLANAB. Segundo Wu, Zhou e Chu (2005), os resultados encontrados pelos otimizadores de cada unidade de produção, comumente, são resultados subótimos ou inviáveis quando unidos para formar o resultado total.

O PLANAB fornece como resultado da otimização do sistema qual petróleo deve ir para cada uma das 13 refinarias em cada 1 dos 12 meses de horizonte do plano, além de indicar qual petróleo deve ser importado e exportado para suprir as necessidades do sistema. Essa ferramenta produz como resultado volumes de petróleo que variam livremente entre o mínimo e máximo permitidos ao modelo, podendo gerar valores impraticáveis, como, por exemplo, mandar 1 mil m^3 de um determinado petróleo para uma refinaria e 99 mil m^3 restante desse mesmo petróleo enviar para outra. O modelo trabalha com intervalos de tempo mensais, não sendo adequado para *scheduling*. Por essas e outras limitações, é preciso utilizar ferramentas complementares para a alocação de petróleo.

É necessário transformar essa indicação em um resultado mais prático condizente com a programação do dia-a-dia dos navios petroleiros, dos terminais e das refinarias. Todo o petróleo indicado para um certo mês deve chegar à refinaria em lotes de maneira adequada a produção e variação de estoque da mesma, respeitando as diversas restrições como ocupação de píeres, capacidade de tanques e vazão de bombas. O papel de uma ferramenta de *scheduling*, nesse contexto, é conciliar todas essas demandas e programar uma sequência de atividades adequada, levando o petróleo apropriado para cada refinaria no momento certo em função da disponibilidade de cada tipo de petróleo a atracar nos terminais. Essa ferramenta de scheduling precisa lidar com intervalos de tempo e horizontes muito menores e enxergar mudanças nos níveis dos tanques e lidar com novas restrições a todo momento, como atraso de navios, queda no consumo de petróleo pela refinaria, indisponibilidade de bombas, entre outras.

4. REVISÃO BIBLIOGRÁFICA

Este trabalho irá abordar aspectos matemáticos da tomada de decisões suportado pelo estado da arte do gerenciamento de uma cadeia de suprimentos de petróleo com foco na alocação de matéria-prima em refinarias. A ascensão dos métodos meta-heurísticos e foco em algoritmos sociais é explicado pela rapidez em que se deseja obter os resultados no universo da programação de fluxo logístico e na eficiência dos mesmos. Além disso, os métodos meta-heurísticos são adequados tanto para problemas NP-difícil (não-polinomiais em tempo) e problemas não-lineares.

Wu, Zhou e Chu (2005) afirmam ser um grande desafio criar uma ferramenta eficiente de *scheduling* de processos que envolvem refinarias de petróleo. Isso faz com que grande parte das decisões do *scheduling* de curto prazo ainda sejam feitas manualmente, mesmo com todo o progresso científico existente na área de planejamento. O motivo dessa dificuldade seria a grande lacuna existente entre as ferramentas apresentadas em diversas pesquisas e sua real aplicabilidade no dia-a-dia. Wolpert e Macready (1997) compara a performance de algoritmos de otimização para uma classe de problemas e conclui que aqueles que são altamente efetivos para alguns problemas não são para outros, assim ele conceitua o teorema do ”não existe almoço grátis” (do inglês*, no free lunch theorem*).

Diferentes tipos de petróleo nunca devem ser misturados, pois apresentam propriedades físicas e químicas muito distintas e que requerem estrutura e condições de processamento também distintas. Assim, em cada local de acúmulo de inventário, deve haver o armazenamento segregado dos principais grupamentos de petróleo de acordo com as suas principais propriedades (Neiro, 2003). Mesmo que dois petróleos sejam estáveis, sua mistura pode ser instável por causa do possível desequilíbrio entre os tipos de hidrocarbonetos de cada um deles com destaque às resinas e asfaltenos.

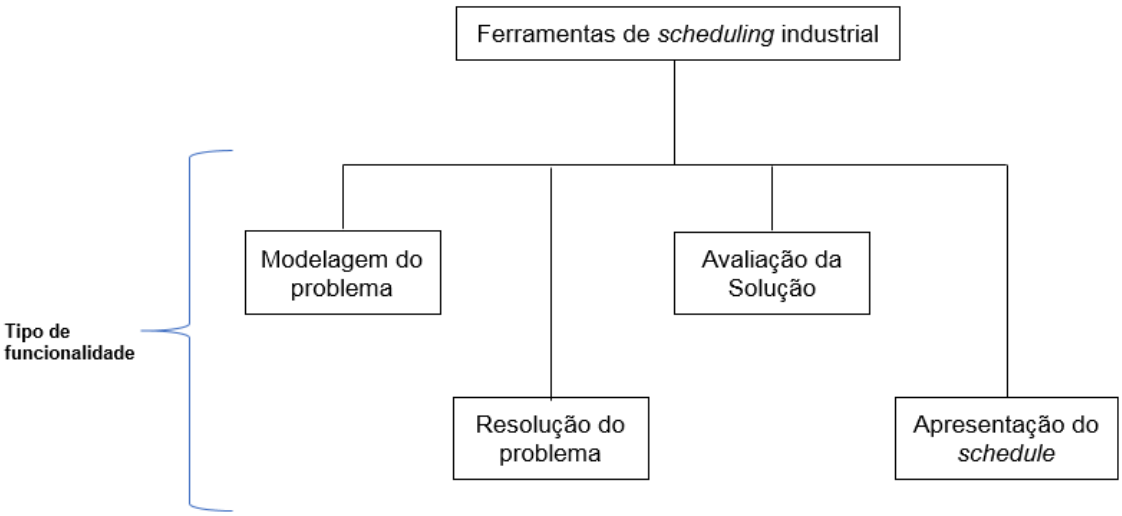
O estudo de sistemas portuários tem sido alvo da pesquisa operacional que explora a grande variabilidade no intervalo de chegada de navios nos portos e os tempos de serviço desses navios quando atracados (Martins, Lüders e Delgado, 2009).

Mas e Pinto (2003) descrevem uma modelagem de um sistema de alocação de petróleo em conjunto com parte de sua cadeia de suprimento. Seu sistema engloba o complexo petrolífero composto de portos, terminais, subestações e refinarias localizados em São Paulo. Uma série de trabalhos, entre artigos, dissertações e teses, utilizaram como base o problema apresentado por Más (2001). Martins, Lüders e Delgado (2009) apresentaram um estudo de simulação usando o *OptQuest* do Arena da programação de atividades de distribuição de petróleo, no terminal portuário GEBAST, contendo portos, refinarias e uma infraestrutura de oleodutos. Nesse trabalho foi simulado o Modelo do Porto. O algoritmo de otimização desenvolvido utiliza uma combinação de metodologias resolutivas, são elas: programação linear, *Scatter Seach*, Busca Tabu e Redes Neurais. Assim, a ideia é reunir a modelagem por programação linear com métodos meta-heurísticos, a fim de evitar as soluções candidatas inadequadas.

4.1 FERRAMENTA DE SCHEDULING

Dada a complexidade das decisões de *scheduling*, as empresas costumam a utilizar *softwares* para fornecer algum tipo de suporte ao usuário humano, já que os softwares possuem melhor memória e capacidade de lidar com as inúmeras restrições que envolvem esse tipo de processo. Essas ferramentas vão desde sistemas interativos, os quais permitem uma verificação automática da viabilidade dos processos de *scheduling*, a sistemas sofisticados, os quais são capazes de otimizar os processos dinamicamente. Cada uma dessas ferramentas possui diferentes funcionalidades, as quais nem sempre são simples de identificar, pois é comum cada fabricante de softwares nomear funcionalidades iguais com nomes diferentes, próprios de cada empresa ou desenvolvedor. De maneira geral, os tipos de funcionalidades podem ser classificados de acordo com a Figura 9 (Dios e Framinan, 2016).

Figura 9 - Tipos de funcionalidade de uma ferramenta de scheduling industrial



Fonte: Dios e Framinan (2016)

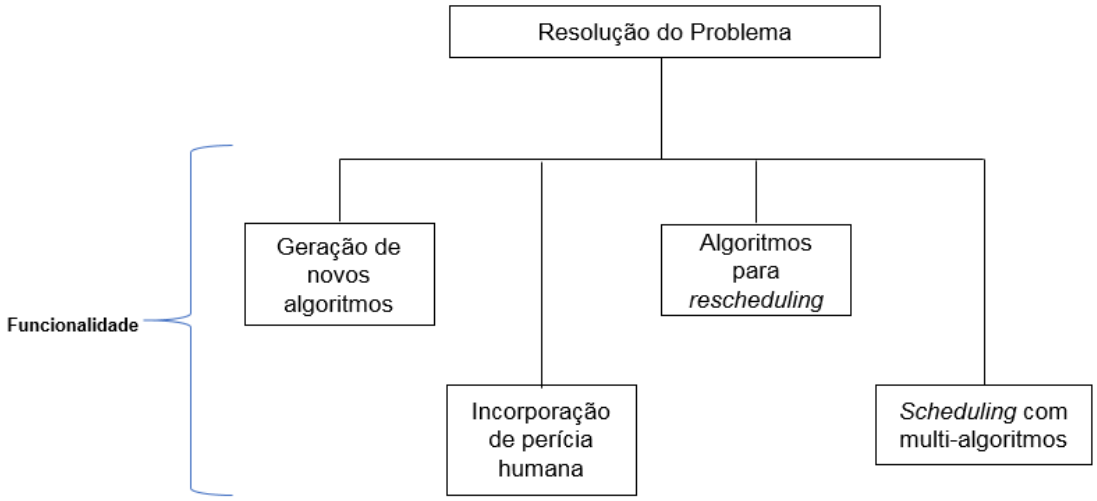
A modelagem do problema é o tipo de funcionalidade que envolve duas funcionalidades principais: a detecção do modelo e a avaliação das restrições. A detecção do modelo consiste na capacidade que o simulador possui de, a partir dos dados de entrada, identificar e adaptar o modelo mais adequado para determinado problema. Existem diferentes abordagens para a detecção de um modelo como a árvore de redução, a árvore de decisão e a detecção de dados brutos. A árvore de redução consiste em um procedimento que identifica pontos comuns do problema em questão com problemas de scheduling já bem estabelecidos e realizar uma modelagem análoga. Não havendo um modelo que se encaixe perfeitamente, o software indica o modelo mais próximo e permite a realização de adaptações para adequação do modelo e associação com um algoritmo. A árvore de decisão consiste na realização de uma sequência de perguntas ao usuário sobre o(s) objetivo(s) do problema e suas restrições e, após a inserção desses dados, fornece o modelo que melhor se adapta entre os modelos pré-cadastrados no *software*. Já a detecção de dados brutos é uma identificação direta do modelo a partir dos dados de entrada (Dios e Framinan, 2016).

A funcionalidade de avaliação das restrições tem a missão de simplificar as restrições inseridas pelo usuário de modo a não desperdiçar esforço computacional com restrições que possuem pouquíssimo ou nenhum impacto na resolução do

problema. Isso pode ser feito incluindo a restrição na própria função objetivo na forma de penalidades. Também pode existir uma classificação das restrições (feita pelo próprio usuário) em *hard* ou *soft*, ou seja, em restrições essenciais e restrições que não são imprescindíveis; assim, é atribuído um peso a cada tipo de restrição estabelecendo uma relação hierárquica entre elas. Uma outra funcionalidade possível é atender apenas às restrições essenciais e as não essências aparecerem como um alerta (ou, do inglês, *warning*) ao usuário no relatório final do programa. Adicionalmente, outra maneira seria o relaxamento das restrições aumentando o espaço de busca (Dios e Framinan, 2016).

O segundo tipo de funcionalidade é a resolução do problema que se divide em 4 diferentes funcionalidades, como mostrado Figura 10. A geração de novos algoritmos pode ser a nível de design ou de usuário, sendo que o primeiro deve ser feito de modo que não altere a estrutura do programa. O usuário pode gerar um algoritmo externamente e importar para o software ou escrever utilizando uma linguagem computacional. Duas outras maneiras seria utilizando a combinação de algoritmos previamente existentes ou inserir uma sequência de regras condicionais que geram o resultado do problema (Dios e Framinan, 2016).

Figura 10 - Funcionalidades de resolução do problema em ferramentas de scheduling industrial



Fonte: Reproduzido de Dios e Framinan (2016).

A incorporação de perícia humana na resolução de problema é realizada adicionando propriedades baseadas na experiência e conhecimento do especialista. Essa incorporação pode ser específica para cada problema ou para um conjunto de problemas. Já os algoritmos para *rescheduling*, são os que lidam com as mudanças nos dados de entrada ou restrições do problema que interfiram no resultado do *schedule* proposto. Já o *scheduling* com multi-algoritmos segrega o modelo completo em submodelos resolvendo cada parte com um algoritmo próprio (Dios e Framinan, 2016).

A avaliação da solução pode ser realizada através das considerações dos objetivos ou da análise de cenários. No primeiro caso, na existência de vários objetivos no problema, o software vai desenvolver algum critério para escolher o melhor resultado, como, por exemplo, atribuindo pesos a cada objetivo ou transformando alguns objetivos em restrições. Já no segundo caso, o sistema fornece todos os cenários possíveis que atendem as especificações dos dados de entradas, e o usuário é capaz de escolher o scheduling que mais conveniente ao seu processo ou sistema. O último tipo de funcionalidade é a apresentação do *scheduling,* o qual define a maneira na qual o software expõe o resultado ao usuário. O resultado pode

ser apresentado em forma de texto ou de diversas formas gráficas adequadas ao sistema em questão (Dios e Framinan, 2016).

Uma ferramenta de scheduling pode ser uma ferramenta simples ou robusta, mas o que define se ela é adequada é a sua capacidade de tratar os dados de entradas, resolver o problema e apresentar a solução de modo a subsidiar as tomadas de decisão do agente daquele sistema ou processo.

4.2 PROBLEMA COMBINATORIAL

Os problemas combinatoriais são geralmente simples de modelar, porém difíceis de resolver (Dorigo e Stützle, 2004). Um problema combinatorial é aquele que pode ser resolvido como uma verificação de um conjunto finito de combinações, mesmo que a quantidade de soluções possíveis seja tão grande que não haja tempo computacional hábil para encontrar todas. Se a quantidade de soluções for suficientemente pequena, o problema pode ser resolvido por enumeração, caso contrário, é necessário utilizar métodos mais apropriados. Os problemas combinatoriais são classificados em função da dificuldade em resolvê-los pela teoria da complexidade computacional (Solnon, 2010).

Para analisar a quantidade de recursos necessários para a resolução de um algoritmo, pode-se utilizar o conceito de complexidade do tempo, o qual dá uma ordem de magnitude da quantidade de comandos elementares que devem ser executadas pelo algoritmo independentemente da configuração do computador ou da linguagem de programação. A complexidade do tempo dá uma ordem de magnitude em vez do número exato de instruções e, geralmente, é função do tamanho dos dados de entrada do problema. Assim, ele indica a escalabilidade do algoritmo, que é definida como a progressão do aumento do tempo de execução à medida que a quantidade de dados de entrada aumenta (Solnon, 2010).

Uma abordagem direta para a solução de problemas de otimização combinatória seria a busca exaustiva através da enumeração de todas as soluções possíveis e a escolha do melhor. Contudo uma abordagem tão simplista é inviável tratando de problemas de médio e longo porte (Dorigo e Stützle, 2004). A complexidade de um problema é avaliada em função da complexidade dos algoritmos necessários para

resolvê-lo. Existem, basicamente, duas classes de complexidade para problemas de decisão, ou seja, problemas que basicamente fazem uma pergunta e a solução de quais (os dados de saída) são sim ou não, são elas as classes P e NP. Sendo P os problemas que podem ser resolvidos em tempo polinomial por uma máquina de Turing determinística e NP a classe que é possível de ser resolvida em tempo polinomial por uma máquina de Turing não determinística (Solnon, 2010)(Dorigo e Stützle, 2004).

Quando um problema “A” pode ser resolvido no tempo polinomial e um problema “B” pode ser transformado em “A” por uma redução a tempo polinomial (procedimento que transforma um problema em outro por um algoritmo de tempo polinomial) significa que problema “B” pode ser resolvido em tempo polinomial. Um problema é NP-difícil, se todos os outros problemas NP podem ser transformados nele por uma redução de tempo polinomial. Um problema NP-difícil é pelo menos tão difícil como qualquer outro problema no NP, mas não necessariamente são NP. Quando problema é NP-difícil é também NP, ele é denominado NP-completo, sendo, portanto, os problemas mais difíceis em NP; um exemplo de problema NP-completo é o Problema do Caixeiro Viajante (ou, do inglês, *Travel Salesman Problem* - TSP) (Dorigo e Stützle, 2004).

O problema de otimização combinatória são um subconjunto dos problemas de otimização no qual um valor de custo positivo é atribuído a cada objeto no espaço de busca por uma função objetivo, que deve ser minimizada. Existem algoritmos determinísticos que garantem a solução ideal para essa classe de problemas, mas geralmente com um alto custo computacional, abrindo espaço para a utilização de algoritmos meta-heurísticos, a fim de calcular soluções precisas em muito menos tempo (Mullen *et al.*, 2009).

4.3 MODELO DE OTIMIZAÇÃO

A cadeia de suprimentos de uma empresa não pode ser tratada como relações individuais de cada etapa do processo e sim como uma cadeia (como o próprio nome já diz) de relações e encadeamento de processos.(Pires, 2009) Portanto, o modelo de otimização proposto não sugere que a otimização de cada etapa do

processo acarreta na otimização do todo, e sim enfatiza a necessidade de uma melhoria integral do processo para alcançar o melhor resultado.

Segundo Weise (2009), um modelo é uma aproximação de um sistema que nos permite raciocinar e deduzir propriedades do mesmo; são, frequentemente, simplificações ou idealização de problemas do mundo real. Eles são definidos deixando de lado os fatos que provavelmente têm um impacto muito pequeno nas conclusões extraídas deles, focando nos fatores mais relevantes. Elaborar modelos perfeitos, com todas as informações reais e precisas é algo muito difícil de ser realizado e também complicado em ser manipulado após completo (Shimizu, 2010). O processo de tomada de decisão no planejamento da produção pode ser tão complexo quanto se queira detalhar as variáveis e fenômenos no modelo utilizado e quanto maior o horizonte a ser trabalhado (Neiro, 2003).

A aplicação de métodos científicos, como a construção de um modelo, para o aperfeiçoamento das operações, decisões e gestão é uma das principais características da pesquisa operacional, que envolve, por exemplo, otimização, simulação, teoria de grafos (Liza, 2005). O objetivo da otimização é encontrar os melhores elementos possíveis de um conjunto de dados de entrada de acordo com um grupo de critérios. Tais critérios são expressos como funções matemáticas, as chamadas funções objetivos, ou também podem ser algoritmos complexos que, por exemplo, envolvem múltiplas simulações (Weise, 2009). A otimização pode ser para minimização ou maximização de uma função objetivo em função dos valores associados às variáveis de decisão, as quais são sujeitas a restrições (ou limites).

Os métodos para resolver problemas de otimização podem ser divididos em dois grupos principais: técnicas determinísticas e técnicas estocásticas. As técnicas de otimização global estocásticas (ou probabilísticas) envolvem elementos probabilísticos e empregando números aleatórios em sua estrutura resolutiva. Algumas vantagens dos métodos estocásticos inclui a simplicidade de entendimento e programação, não exigindo todos os pressupostos do problema de otimização, a grande variedade de problemas que podem resolver, sua capacidade de fornecer resultados robustos para problemas altamente não-lineares mesmo com muitas

variáveis de decisão e convergência mais rápida para uma solução viável (Rangaiah, 2010).

Em teoria, as técnicas estocásticas precisam de iterações infinitas para garantir a convergência para o ótimo global. Contudo, na prática, elas geralmente convergem rapidamente para uma solução aceitável. As metodologias estocásticas podem ser divididas em quatro grupos: (1) técnicas de pesquisa aleatória, (2) métodos evolutivos, (3) métodos de algoritmos sociais e (4) outros métodos. Os métodos de pesquisa aleatórios incluem pesquisa aleatória pura, pesquisa aleatória adaptativa, métodos de duas fases, recozimento simulado e pesquisa de tabu. Os algoritmos evolutivos são métodos de pesquisa baseados em fenômenos populacionais inspirados em características e processos de evolução biológica, como, por exemplo, os algoritmos genéticos, estratégia de evolução, programação genética, programação evolutiva e evolução diferencial. A otimização por inteligência de enxame e a por colônias de formigas (ACO) são exemplos de algoritmos sociais (Rangaiah, 2010).

Um modelo matemático é uma ferramenta de apoio às tomadas de decisão que deve fornecer resultados claros e úteis. A elaboração do modelo envolve a identificação e entendimento da dependência entre variáveis, estabelecimento de hipóteses de simplificações razoáveis da realidade e a formulação do método de resolução apropriado (Liza, 2005). A escolha, ou elaboração, de um modelo depende, além da complexidade do processo em questão, da sua finalidade e do tempo e recursos disponíveis. A complexidade está diretamente associada ao número de variáveis, quantidade de riscos e incertezas, e precisão nos valores das variáveis na elaboração da função objetivo do problema (Shimizu, 2010).

Um modelo de otimização é basicamente composto por uma função objetivo e restrições. Uma restrição é uma condição que deve ser satisfeita, sendo esta determinada por uma relação lógica entre um conjunto de incógnitas, referidas como variáveis. Assim, uma restrição limita os valores que podem ser atribuídos simultaneamente a essas variáveis (Solnon, 2010). Um problema de otimização combinatorial é formalmente descrito através de uma tripla (S, f, Ω), na qual (Dorigo e Stützle, 2004):

$$P \stackrel{\text{def}}{=} (S, f, \Omega)$$

- S: é o conjunto de candidatos solução do problema, sendo cada candidato s $\in$ S;
- f: é a função objetivo a ser maximizada ou minimizada, sendo f uma função de s;
- Ω: são as restrições do problema.

Para resolver P , deve-se encontrar uma solução ótima s^* com um mínimo (*ou máximo*) valor da função objetiva $f(s^*) \leq f(s)$ $(ou\ f(s^*) \geq f(s))$, $\forall s \in S$. Otimização em geral varia por estrutura como mono ou multiobjectivo, com ou sem restrições, ou problemas com ou sem otimização combinatorial.

Desde a década de 50, são realizadas atividades de planejamento e programação de produção nas refinarias de petróleo através da programação linear, porém, apenas a partir da década de 90 que os recursos computacionais permitiram uma utilização mais ampla desses algoritmos de otimização (Liza, 2005). Dependendo do perfil estratégico da companhia, o objetivo do problema de otimização pode se maximizar os lucros ou minimizar os custos (Neiro, 2003).

4.3.1 PROBLEMA DE SATISFAÇÃO DE RESTRIÇÕES

Segundo Solnon (2010), um problema de satisfação de restrições ou, do inglês, *Constraint Satisfaction Problem* (CSP) é definido por um triplo (X, D, C) tal que:

- X é um conjunto finito de variáveis que são às incógnitas do problema;
- D é o conjunto de valores que podem ser atribuídos a xi, ou seja, é uma função que associa um domínio D (xi) a cada variável xi $\in$ X;
- C é um conjunto finito de restrições e cada restrição cj $\in$ C é uma relação entre algumas variáveis de X; denotamos esse conjunto de variáveis por var (cj).

Resolver um CSP (X, D, C) envolve a atribuição de valores às variáveis de forma que todas as restrições sejam satisfeitas e sua complexidade depende dos domínios (D) das variáveis e do tipo de restrições utilizadas. O problema é denominado polinomial quando todas as restrições são equações ou inequações lineares e todos

os domínios são intervalos numéricos contínuos. Adicionalmente, existem casos nos quais as restrições podem ser uma fórmula matemática arbitrária e os domínios são intervalos numéricos contínuos, aumentando a complexidade do problema (Solnon, 2010).

Nem todos os CSP são problemas de otimização, mas neste trabalho o foco será nessa classe. Quando as restrições de um CSP permitem a existência de muitas soluções diferentes que as satisfazem, o CSP é dito ser sub-restrito. Neste caso, algumas soluções podem ser preferidas diante de outras e essas podem ser expressas na forma de uma função objetivo tornando o CSP um problema de otimização com restrições (Solnon, 2010).

4.3.2 PROGRAMAÇÃO LINEAR

Uma diversa gama de problemas de *scheduling* pode ser descrita como MILP, já que envolvem variáveis contínuas e discretas. O método normalmente empregado para sua resolução é o processo enumerativo "*branch and bound*", o qual envolve a relaxação da integralidade do modelo, ou seja, transforma as variáveis inteiras em reais e resolve o sistema como uma otimização linear. No caso de as variáveis da solução ótima serem fracionárias (e não inteiras como desejado) é necessário desenvolver uma árvore de busca e resolver um problema LP para cada nó, encontrando resultados inteiros. Contudo, nem todos os nós são analisados, já que ele elimina soluções não prováveis. O valor da diferença entre a função objetivo do caso relaxado para o inteiro é chamado de *relaxation gap* ou diferença de integralidade ou relaxação e, segundo alguns autores, o tempo de resolução do algoritmo costuma ser menor quanto menor for essa diferença (Moro, 2000).

A programação linear inteira mista possui a desvantagem de não ser possível estabelecer condições de otimalidade devido a sua natureza combinatorial. Se fossem contabilizados todos os possíveis resultados haveria 2^n possibilidades, sendo n o número de variáveis decisórias (Pinto, 2000).

4.3.3 FERRAMENTAS MATEMÁTICAS

Um elemento matemático bastante utilizado em problemas de otimização, especialmente em problemas de sequenciamento de tarefas são os grafos. Segundo Marins (2011), um grafo é uma estrutura que corresponde a um par de conjuntos G = (N, E), sendo:

- “N” um conjunto de entidades. Podendo representar, por exemplo, pontos, locais, pessoas, áreas geográficas ou equipamentos;
- “E” um conjunto de inter-relações entre os elementos de N. Por exemplo: estradas, parentescos e fronteiras entre áreas geográficas.

Um grafo pode ser representado na forma matricial, sendo então composto por uma matriz de adjacência ou uma matriz de incidência. A matriz de adjacência contém os dados de quais elementos estão relacionados entre si, sendo o número “1” equivalente a sim e o “0” a não. Já a matriz de incidência é usada para grafos orientados, sendo o número “1” equivalente a direção positiva, o “0” a não relacionados diretamente, e o número “-1” equivalente a direção negativa (sentido contrário). O grau de um grafo é o número de arestas que cortam cada nó (Marins, 2011). Uma maneira de impedir que certos arcos do grafo sejam inviáveis na solução final é associar pesos aos arcos indesejáveis, com magnitude suficientemente grande de modo que ele nunca esteja presente na solução ótima final (Dorigo e Stützle, 2004).

O Matlab® é uma linguagem para cálculo numérico extensivo e pode ser utilizado para a resolução de problemas de otimização, porém, é difícil de utilizar quando o problema é de médio ou grande porte. Linguagens algébricas de modelagem de alto nível, como o GAMS e o AIMMS, foram projetadas especialmente para a resolução de modelos de otimização, sendo, portanto, mais diretos e práticos nessa linha de problemas (Liza, 2005). Esse ponto de vista supõe um tratamento de otimização por programação não-meta-heurística. Entretanto, é certo que uma computação massiva exige também técnicas de computação paralela e em nuvem, cada vez mais disponível atualmente.

4.3.4 MODELO DE *SCHEDULING*

O propósito da resolução de um problema de *scheduling* é mapear atividades para adequar serviços otimizando um ou mais objetivos (Kalra e Singh, 2015). Os três componentes que estão sempre presentes em problemas de *scheduling* são: a alocação de tarefas, a sequência das atividades e o tempo de utilização dos recursos durante cada tarefa (Liza, 2005). Para solucionar um problema de programação de produção, ele deve ser descrito a partir de quatro atividades básicas: selecionar a quantidade de atividades que serão executadas dentro de um horizonte de tempo; alocar recursos para executar essas atividades considerando suas disponibilidades; definir a sequência cronológica das atividades; determinar a hora e os intervalos de tempo da utilização de recursos e instalações necessários para cada atividade (Moro, 2000).

Rabadi (2016) classifica como *Spatial Scheduling* uma classe de problemas de *scheduling* mais complexa que as tradicionais, já que, além de incluir as variáveis de tempo e prazo de processamentos, inclui limitações bidimensionais de espaço (ex.: construção naval, montagem de aeronaves e indústrias pesadas). Assim, esse conceito não é aplicado aos problemas que envolvem restrições volumétricas e sim restrições de espaço plano.

Oliveira, Hamacher e Almeida (2009) defendem que os problemas reais de *scheduling* em refinarias de petróleo são altamente não lineares, contêm grande quantidade de variáveis e são majoritariamente modelados com programação linear inteira, dificultando sua resolução com programação linear. O uso de heurísticas como o método evolucionário dos algoritmos genéticos é uma alternativa para encontrar soluções viáveis para esse tipo de problema em tempo hábil. Oliveira descreve a modelagem, utilizando algoritmos genéticos, do *scheduling* da produção de óleo combustível e asfalto em refinarias, encontrando soluções satisfatórias. Já comparando o desempenho do algoritmo genético com a programação linear inteira mista, foi observado que, apesar do método heurístico apresentar melhor desempenho, ele é altamente dependente de uma boa parametrização, a qual deve baseada em dados experimentais.

Bazgosha, Ranjbar e Jamili (2017) apresenta a resolução de um problema similar ao de programação paralela de máquinas com minimização do tempo de

sequenciamento (ou, do inglês, *makespan*), o que é um problema NP-difícil (ou, do inglês, *NP-hard*). Este problema de *scheduling* de operações portuárias foi representado como um modelo de programação linear inteira e é composto de múltiplas estações de carregamento/descarga idênticas com restrições de horários de desatracamentos e de inventário. Ele aborda diferentes soluções meta-heurísticas e enfatizou o desempenho do algoritmo de cuco em detrimento do algoritmo genético e de enxame de abelhas na resolução de problemas de *scheduling*.

O número de intervalos de tempo utilizado deve representar bem o sistema, pois, caso contrário, pode não ser possível alcançar o resultado otimizado almejado. Contudo, um número muito alto pode inviabilizar a solução (Liza, 2005).

Segundo Xu *et al.* (2017), a maioria das refinarias compra petróleo bruto, o que deve ser otimamente integrado com seu processamento, para que a potencial rentabilidade de uma refinaria possa ser atingida. Esta é a razão da importância do *front end* (processos entre o navio petroleiro e a refinaria) no gerenciamento da cadeia de suprimentos das refinarias, especialmente sob as condições atuais das competições globais e rígidas regulamentações ambientais. Xu *et al.* (2017) criou um modelo de *scheduling* em grande escala para descarregamento, transferência e processamento de petróleo bruto, usando tempo contínuo, com o objetivo de simular e otimizar (maximizando o lucro) as operações de petróleo bruto *front-end* e refinaria simultaneamente.

4.3.5 TEORIA DAS FILAS

A demanda por qualquer tipo de serviço é uma variável fundamental na modelagem de um processo logístico e o manuseio errado dessa variável (tanto num modelo teórico quanto no processo real) pode acarretar em atrasos na execução de serviços e/ou em filas, ambos custosos. Uma modelagem satisfatória deve ser capaz de absorver as flutuações de demanda usando o critério de otimização escolhido. Marins (2011) afirma que a teoria das filas pode ser aplicada, em diversos projetos, como, por exemplo, no dimensionamento de postos de pedágio, número de berços

de atracação em portos, número de guichês num banco, do número de linhas de uma Central Telefônica, e de equipes de manutenção.

Um sistema de filas é composto por quatro elementos básicos: o processo de chegada, processo de atendimento (ou organização da fila), disciplina de atendimento e capacidade do sistema (Marins, 2011).

Para o processo de chegada, é necessário saber para um determinado instante (t) qual o número de solicitações do serviço desde o tempo inicial (t0 = 0) ao instante t. Esse dado pode ser representado como uma distribuição dos intervalos de tempo entre chegadas sucessivas ou como uma distribuição do número de chegadas em intervalos disjuntos. Essa distribuição pode ser determinística, como no caso de sistemas automatizados de uma linha de montagem, ou probabilísticas, como a chegada de clientes em um estabelecimento. Também é possível que o processo de chegada aconteça em grupos como caravanas chegando em shows, sendo assim, o tamanho de cada grupo uma variável aleatória (Marins, 2011).

O processo de atendimento, que pode ser individual ou em grupo, possui uma característica essencial: a duração dos atendimentos ou o número de atendimentos. A duração dos atendimentos pode ser constante, pode ser função de outras variáveis determinísticas ou pode ser aleatória (Marins, 2011).

A disciplina de atendimento se refere ao modo pelo qual o atendimento é feito em função de sua requisição, o que é uma característica de cada sistema. O modo mais usual é o FIFO (do inglês, *First in First Out*) ou, em português, PEPS (Primeiro que Entra Primeiro que Sai), o qual dá a prioridade do atendimento ao elemento que chegou primeiro. Outro tipo de disciplina de atendimento comum é o LIFO (do inglês, *Last in First Out*) ou, em português UEPS (Último que Entra Primeiro que Sai), o qual pode ser usado em gerenciamentos de estoques, por exemplo. Também pode existir algum tipo de sistema de prioridades, permitindo que alguns elementos passem na frente de outros na fila, como o caso de emergências em hospitais (Marins, 2011).

Já a capacidade do sistema é um limite máximo de números de elementos que poderão utilizar aquele serviço, o que pode existir ou não, por limitações físicas.

Após o tingimento dessa capacidade, alguns elementos não são aceitos para entrar na fila (Marins, 2011).

4.4 META-HEURÍSTICA

De acordo com Simon (2013):

> Heurística vem da palavra grega $\eta\upsilon\rho\iota\sigma\kappa\omega$, que é traduzida como *eurisko* em inglês. A palavra significa encontrar ou descobrir. É também a fonte da exclamação inglesa *eureka*, que usamos para expressar triunfo quando descobrimos algo ou resolvemos um problema. Os algoritmos heurísticos são métodos que usam regras gerais ou abordagens de senso comum para resolver um problema. Geralmente, não se espera que os algoritmos heurísticos encontrem a melhor resposta para um problema, mas espera-se que eles encontrem soluções "próximas o suficiente" para o melhor. O termo metaheurística é usado para descrever uma família de algoritmos heurísticos. A maioria, se não todos, os algoritmos evolucionários que discutimos neste livro podem ser implementados de muitas maneiras diferentes e com muitas opções e parâmetros diferentes. Portanto, todos eles podem ser chamados de metaheurísticas. Por exemplo, a família de todos os algoritmos de otimização de colônia de formigas pode ser chamada de metaheurística de colônia de formigas.

Assim, os métodos meta-heurísticos podem ser considerados métodos heurísticos escritos de forma genérica afim de abranger uma diversidade maior de problemas de otimização. Segundo Aloise *et al.* (2002), as meta-heurísticas são métodos flexíveis e por isso propicia uma adaptação mais simples a problemas reais utilizando modelos dinâmicos. Além disso elas são conhecidas por unir ferramentas que realizam buscas robustas no espaço solução com as de escape de ótimos locais.

As meta-heurísticas são muitas vezes utilizadas para encontrar boas soluções, as quais podem ser ótimas ou não, e sua aplicação deve ser modelada caso a caso. As principais características que diferenciam os algoritmos meta-heurísticos são: critério de escolha de uma solução inicial; definição da vizinhança N(s) de uma solução s; critério de seleção de uma solução vizinha dentro de N(s); critério de parada (Martins, Lüders e Delgado, 2009). Técnicas baseadas em meta-heurísticas atingem soluções ótimas em um tempo aceitável. Para problemas que não necessariamente precisam de um ótimo global, mas de um bom resultado (ótimo local) em pouco

tempo, como otimização em nuvem, as metodologias meta-heurísticas são de grande aplicabilidade (Kalra e Singh, 2015).

A resolução de um problema utilizando um modelo determinístico torna-se mais complexa, ou mesmo inviável, quando a relação entre um conjunto candidato a solução e a função objetivo a ser perseguida não é matematicamente direta ou a dimensionalidade do espaço de busca é muito alta. A resolução determinística envolveria uma enumeração exaustiva dentro do espaço de busca, o que não é viável mesmo para problemas relativamente pequenos em virtude do alto esforço computacional e tempo empregado. Esse tipo de limitação dos algoritmos determinísticos abriu espaço para o crescimento dos algoritmos probabilísticos que que vêm sendo desenvolvidos e estudados há cerca de 55 anos. Muitos deles são baseadas na técnica de Monte Carlo e trocam a garantia de atingimento do ótimo global por um tempo de execução mais curto (Weise, 2009).

No campo da computação meta-heurística, os processos bio-inspirados vêm ganhando destaque por seus algoritmos serem inteligentes e adaptativos tal qual os organismos biológicos. A inteligência coletiva (*swarm intelligence*) tem se mostrado como técnicas interessantes entre as heurísticas, elas partem do conceito de que a inteligência coletiva de um grupo de indivíduos é mais do que a soma da inteligência individual. Com base nesse princípio, foram identificados uma grande variedade de algoritmos como: otimização de colônias de formigas, bando de aves, bactérias e peixes; muitos desses algoritmos são também eficientes na resolução de otimização multiobjetivo (Kar, 2016).

Na última década, cada vez mais *softwares* comerciais têm incorporado heurísticas em seus algoritmos de resolução, com o objetivo de identificar melhores arranjos para os parâmetros escolhidos pelos usuários. Alguns exemplos já aplicados em softwares comerciais são: algoritmos genéticos, algoritmos miméticos, algoritmos culturais e colônia de formigas (Martins, Lüders e Delgado, 2009).

Uma heurística basicamente é um método de solução de um problema por tentativa e erro. Metaheurístico é um nível genérico ou de heurística de nível superior que é mais geral do que heurística simples, e a computação metaheurística é um cálculo adaptativo que aplica regras heurísticas gerais na solução de uma categoria de

problemas computacionais. Portanto metaheurística pode ser descrito, ainda valendo a definição de “otimização”, como

$$MH \stackrel{\text{def}}{=} \left(O, A, R^c, R^i, R^o\right)$$

- O : é o conjunto de métodos metaheuristicos escolhidos (metaheurístico, adaptativo, automotivo, tentativa e erro, cognitivo, etc);
- A : é o conjunto de algoritmos genéricos (algoritmo genético, enxame de partículas, algoritmo evolucionário, colônia de formigas, etc);
- $R^c = O \times A$: é o conjunto de relações internas;
- $R^i \subseteq A' \times A$, com $A' \wedge A \sqsubseteq C'$, é o conjunto de relações de entrada; onde C' é o conjunto de conceitos externos e c é um ambiente de conceito. Por exemplo, por simplicidade $R^i = C' \times c$;
- $R^o = c \times C'$: é o conjunto de relações de saída;

Essa definição é muito formal, apesar dela surgir várias questões que abrangem os fatores chaves de uma computação evolucionária, e pode ser melhor explorada com (Hussain *et al., 2018*). Diferentes descrições de métodos metaheurísticos podem ser vistos em (Gogna; Tayal, 2013). Aqui o interesse principal está diretamente em aplicação do método de otimização por colônia de formigas.

4.5 MÉTODO META-HEURÍSTICO DE COLÔNIA DE FORMIGAS

O algoritmo colônia de formigas, ou do inglês ACO (*Ant Colony Optimization*) é um método de otimização meta-heurístico populacional proposto em 1991 por Marco Dorigo e colegas e aprimorado por outros autores (Dorigo e Stützle, 2004). Ele se baseia no comportamento real de formigas, as quais se movimentam de modo organizado em busca de seu objetivo, seja ele encontrar comida ou o caminho de volta ao formigueiro. Este algoritmo foi primeiramente utilizado para resolver um problema de caixeiro viajante (PCV), o qual busca encontrar o percurso otimizado para que o caixeiro visite todas as cidades que precisa.(Tavares Neto e Godinho Filho, 2013) O algoritmo já foi aprovado para outros problemas *benchmarket* como o problema da mochila e também tem diversas aplicações na área de telecomunicação (Bedi e Singh, 2013).

Ele é utilizado para solucionar problemas de otimização contínuos ou discretos e já foi validado para diversas situações, tal qual para o problema de *scheduling* (planejamento de tarefas), o qual, comumente, é classificado como NP-difícil, ou seja, são algoritmos não-determinísticos polinomiais. Um problema de *scheduling* consiste em alocar uma série de tarefas ao longo do tempo, que devem ser executadas utilizando os recursos disponíveis, respeitando as restrições e maximizando ou minimizando uma dada função objetivo (Tavares Neto e Godinho Filho, 2013).

Muitos já notaram formigas andando uma atrás da outra como se seguissem uma trilha, e provavelmente pararam para pensar como aquela movimentação funcionava ou mesmo realizou experimentos de bloquear sua rota para observar as ações das formigas. Biólogos já explicitaram a capacidade de comunicação de alguns animais de mesma espécie através de feromônios (ou feromonas), que são substâncias secretadas por insetos e mamíferos para efeito de reconhecimento e comunicação através do odor exalado; essas substâncias químicas foram descobertas na década de 1960 (Monmarché, Guinand e Siarry, 2010). Desde a década de 90, esse fenômeno inspirou profissionais da área computacional na resolução de problemas de otimização e a metodologia de colônia de formigas é a de mais sucesso e a de maior reconhecimento entre as técnicas que se baseiam no comportamento real de formigas, já que pode ser utilizado, com alta performance, em vários tipos de problemas (Dorigo e Stützle, 2004). Muitas espécies de formigas têm o aparelho ocular pouco desenvolvido ou são até cegas, fazendo com que a comunicação química entre elas seja ainda mais aplicada.

Insetos que vivem em sociedade são um conjunto de indivíduos que compõem uma estrutura organizacional, a qual proporciona a cada um deles a capacidade de realizar tarefas complexas que, muitas vezes, eles não conseguiriam realizar sozinhos. Esse fenômeno foi uma inspiração para o ACO, que utiliza essa auto-organização para resolver problemas computacionais (Dorigo e Stützle, 2004). Apesar de ser impossível prever o que cada indivíduo fará a cada momento, essa aparente bagunça se transforma em um sistema eficiente, pois mesmo que alguns indivíduos falhe a missão é realizada. Adicionalmente, cada ente do grupo segue se adaptando às necessidades da colônia. O conceito de auto-organização é capaz de

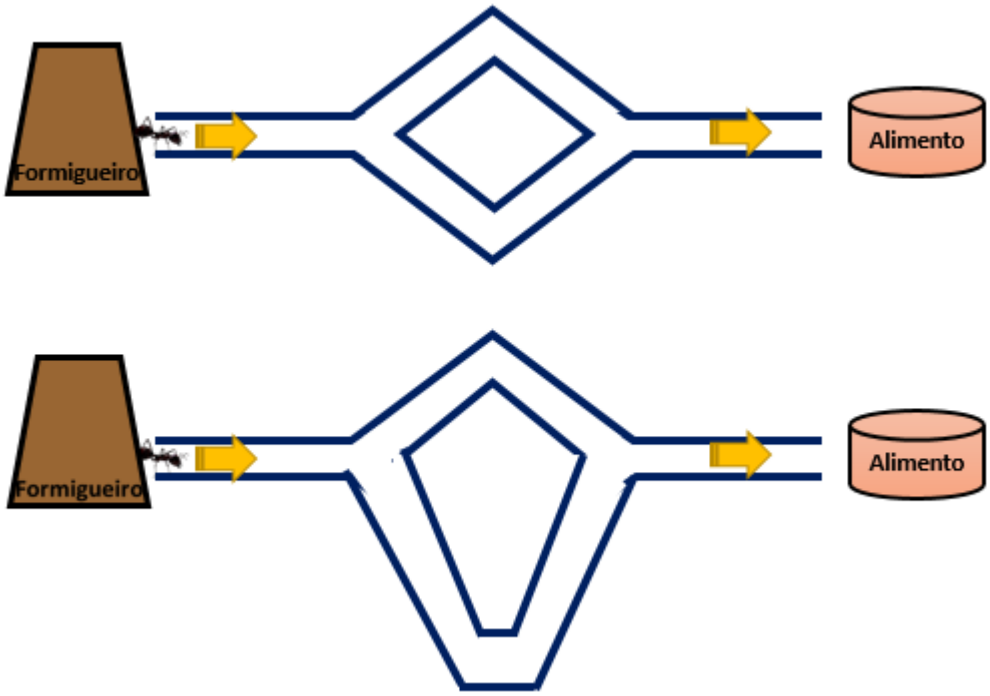
predizer a maioria dos fenômenos macroscópicos através da definição das regras no nível microscópico (Monmarché, Guinand e Siarry, 2010).

Os movimentos coordenados das formigas foram inspiração para diversas metodologias, que se baseiam, por exemplo, no conceito de castas de formigas, na qual cada uma é responsável por uma categoria de trabalho, no transporte cooperativo, escolha da linhagem, busca por alimentos e achar o caminho para o ninho. Formigas de forrageamento deixa um rastro de feromônios que aumenta a probabilidade de as próximas formigas seguirem esse mesmo caminho, indicando a outras formigas forrageadoras o caminho até a fonte de alimento. Quando encontra o alimento, ela faz o caminho de volta para o ninho reforçando trilha de feromônios, garantindo que outras formigas também sigam a trilha. Ao acabar o alimento a trilha deixa de ser reforçada até que ela não seja mais perceptível (Dorigo e Stützle, 2004). Essa comunicação indireta entre as formigas (e também em outros animais) é conhecida como estigmergia e refere-se a um sistema emergente auto organizado a partir da modificação do ambiente local pelos indivíduos (Mullen *et al.*, 2009).

4.5.1 EXPERIMENTO DA PONTE DUPLA

Do inglês "*Double Bridge Experiment*", foi um experimento desenvolvido e executado por Jean-Louis Deneubourg em 1989 junto com sua equipe. Eles construíram uma ponte dupla, que servia de passagem para formigas (de uma espécie Argentina), ligando o formigueiro a uma fonte de alimento. O experimento foi realizado, em um ambiente controlado, construindo dois tipos de ponte: um com os dois caminhos do mesmo tamanho e outro com os caminhos de tamanhos distintos, como ilustrado na Figura 11, e a porcentagem de formigas que seguia cada trilha era registrada ao longo do tempo (Dorigo e Stützle, 2004).

Figura 11 - Experimento da ponte dupla



Fonte: Dorigo e Stützle, 2004

Na primeira parte do experimento, utilizando os caminhos de mesma distância, as formigas foram deixadas livres para escolher a maneira de chegar à comida; no começo as escolhas eram aleatórias, mas, com o passar do tempo, um dos caminhos era preferenciado em detrimento do outro. Isso acontece pois, aleatoriamente, umas das trilhas foi ficando com uma quantidade maior de feromônios depositados atraindo mais formigas por esse caminho e, consequentemente, aumentando a concentração de feromônios até se tornar um caminho preferencial. Adicionalmente, como os feromônios vão evaporando com o tempo, o caminho “errado” vai sendo esquecido e cada vez menos utilizado (Dorigo e Stützle, 2004).

Na segunda parte do experimento, um dos caminhos é mais curto que outro (metade do tamanho) e, com o tempo, foi escolhido como caminho prioritário. Assim como na primeira parte do experimento, no início, as escolhas são aleatórias; depois, como o caminho menor demora metade do tempo para ser percorrido, a concentração de feromônios cresce mais rapidamente, atraindo cada vez mais formigas em um processo autocatalítico (ou feedback positivo). Contudo, ainda há formigas que

continuam percorrendo o caminho mais longo, o que pode ser interpretado como uma escolha exploratória de alguns indivíduos na busca de novas fontes de alimento (Dorigo e Stützle, 2004).

Em uma terceira parte do experimento, apenas o caminho mais longo é disponibilizado, fazendo com que as formigas o escolham. Após meia hora, o caminho mais curto é adicionado às opções (voltando ao *layout* do segundo experimento), porém as formigas continuaram usando o caminho mais longo, por estar rica em feromônios, enquanto caminho mais curto era utilizado apenas pelos indivíduos que exploravam novos caminhos, não sendo suficiente para formar uma trilha de feromônios forte o bastante para influenciar muitos outros, mesmo a evaporação favorecendo esse fenômeno (Dorigo e Stützle, 2004).

4.5.2 VARIANTES DO ACO

Colorni, Dorigo e Maniezzo (1992) desenvolveu um algoritmo baseado em colônias de formigas, no qual é proposta uma nova abordagem para a solução e otimização distribuída de problemas com base no resultado munido de baixo nível de interações entre uma gama de agentes simples cooperantes que não estão cientes de seu comportamento cooperativo. A metodologia foi desenvolvida e aplicada com sucesso para a resolução do problema do caixeiro viajante. A princípio, foram propostas três diferentes versões do AS (*Ant System*): *ant-density* (Colorni, Dorigo e Maniezzo, 1992)**;** *ant-quantity* (Colorni, Dorigo e Maniezzo, 1992)**;** e *ant-cycle* (Dorigo, 1992). As primeiras duas versões atualizavam o feromônio logo após cada passo dentro da iteração, enquanto, na última versão (*ant-cycle*), a quantidade de feromônio só era atualizada depois que todas as formigas construíam a resposta, ou seja, ao final de cada iteração. Atualmente, quando se refere ao AS na refere-se a versão *ant-cycle* que se demonstrou mais vantajosa que as outras duas (Dorigo e Stützle, 2004).

A primeira variante do AS original foi o EAS (*Elitist Ant System*), o qual valoriza ainda mais os melhores resultados através de um incremento extra na atualização dos feromônios no percurso responsável pelo melhor resultado (*best-so-far*), como se fosse uma formiga extra no algoritmo. Essa formiga adiciona uma quantidade de

feromônio proporcional ao adicionado pelas outras por um fator “e” que define o peso desse incremento (Dorigo e Stützle, 2004).

Na variante AS *rank,* cada formiga deposita uma quantidade de feromônio que é inversamente proporcional a sua classificação no ranking de melhores resultados de função objetivo; empates no *ranking* podem ser resolvidos aleatoriamente. Em cada iteração, somente as melhores formigas do *ranking* e a formiga que produziu a melhor FO até agora dentre todas as iterações, depositam feromônio; essa última sempre contribui com a maior quantidade. Na solução do problema do caixeiro viajante, entre outros, um número adequado de formigas que depositam feromônio em cada iteração é de 6 (Dorigo e Stützle, 2004).

Uma variante do AS, muito estudada e adaptada, é o MMAS (*Max-Min Ant System*), na qual apenas a melhor formiga (*best-so-far*) ou a melhor da iteração deposita o feromônio, aumentando o risco de estagnação em mínimos locais. Esse efeito é contornado limitando o valor máximo de feromônios depositado em cada trecho. O valor inicial dos feromônios é o limite superior permitido e, se o algoritmo alcança estagnação (nenhuma resposta melhor é encontrada em um certo número de iterações) ele é reinicializado (Dorigo e Stützle, 2004).

Uma diferença crucial entre as variantes corresponde ao modo de construção de soluções, que pode ser em paralela ou sequencial. Na implementação paralela, a cada passo da construção da solução todas as formigas realizam sua decisão (escolhem para qual nodo seguir), já na implementação sequencial, apenas após uma formiga encontrar uma solução completa a próxima começa a construir sua solução. Se tratando de AS, os dois modos de construção da solução acarretam em resultados bastante similares, enquanto que, em outros algoritmos ACO, como o ACS, isso não ocorre (Dorigo e Stützle, 2004).

A variante BWAS (*Best-Worst Ant System*) utilize uma regra agressiva de atualização dos feromônios (similar ao MMAS e o ACS), na qual, além de apenas a melhor formiga depositar o feromônio, a pior formiga tem feromônios removidos dos trechos que não são coincidentes com os da melhor formiga. O BWAS utiliza frequentes reinicializações dos feromônios, aumentando a capacidade exploratória do algoritmo. Adicionalmente, ele emprega a mutação de feromônios, conceito oriundo de algoritmos evolucionários (Dorigo e Stützle, 2004).

O ACS (*Ant Colony System*) é um tipo de ACO que possui três principais diferenças em relação ao AS (Dorigo e Stützle, 2004):

- O ACS valoriza mais a informação de experiências acumuladas pelas formigas do que a AS, através do uso de uma regra de escolha mais agressiva.
- A evaporação e o depósito de feromônios ocorrem apenas nos arcos pertencentes à melhor rota até o momento.
- Quando uma formiga utiliza um arco, ela remove parte do feromônio do arco para aumentar a exploração de caminhos alternativos para as outras formigas desta iteração. Consequentemente, o algoritmo nunca converge para apenas um resultado.

A cada dia vem surgindo novas modalidades de algoritmos de colônia de formigas, algumas delas apresentam melhorias, baseadas em outros métodos ou não, enquanto outras apresentam combinações com outros algoritmos. O SIACO (do inglês, *Social Interaction Ant Colony Optimization*) é um exemplo de adaptação recente do ACO apresentada por Alves *et al.* (2018); essa variante do algoritmo acrescenta uma interação social baseada na teoria do jogo entre duas formigas da colônia para decidir a quantidade de feromônio depositado. Delgoshaei *et al.* (2019) utiliza uma combinação entre o ACO e o SA (do inglês, *Simulated Annealing*) para resolver um problema de *scheduling* de uma da cadeia de suprimento. Para isso ele utiliza a melhor capacidade do AS de escapar de ótimos locais para melhorar o ACO e consegue um resultado satisfatório para a maioria dos dados testados.

Cecilia *et al.* (2018) aplica um método que substitui o tradicional método da roleta para a escolha do próximo passo a partir das probabilidades; o método chamado SS-Roulette (do inglês, Scan and Stencil Roulette) consiste em primeiro armazenar a matriz completa de probabilidades e aplicar o método da roleta tradicional em paralelo para todos os cálculos e, adicionalmente, uma checagem no padrão de escolho, dando mais robustez ao processo. Adicionalmente, o autor utiliza CUDA, do inglês*, Compute Unified Device Architecture*, que um conjunto de ferramentas desenvolvido pela NVIDIA, como um mecanismo auxiliar a

atualização dos feromônios depositados pelas formigas, as quais realizam o trajeto paralelamente.

5. ALGORITMO COLÔNIA DE FORMIGAS

5.1 ALGORITMO GERAL DO ACO

O termo Algoritmo de Colônia de formigas (ou ACO) é um termo genérico para os algoritmos baseados no comportamento de formigas de reais utilizado para resolver problemas de otimização discreta de natureza estática ou dinâmica.(Dorigo e Stützle, 2004) De forma simplificada, o algoritmo ACO pode ser descrito em forma de fluxograma, como mostrado na Figura 12.(Bedi e Singh, 2013)

Figura 12 - Fluxograma simplificado do ACO

Fonte: Bedi e Singh (2013)

O ACO é um algoritmo estocástico de busca que possui dois tipos possíveis de convergência: a convergência em valor e a convergência em solução. Quando se trata de problemas que possuem mais de uma solução ótima, o objetivo do algoritmo será convergir para qualquer um deles. Portanto, a convergência em valor avalia a

probabilidade de que o algoritmo gere uma solução ótima pelo menos uma vez. Todavia, a convergência em solução analisa a probabilidade de que o algoritmo atinja um estado que continua gerando a mesma solução ideal. Embora a convergência em solução aparente ser mais desejada que a convergência de valor, na otimização só é necessário encontrar a solução ideal uma vez, já que após encontrada o problema é resolvido e o algoritmo pode ser interrompido; isso faz com que a convergência de valor seja suficiente para o sucesso da otimização (Dorigo e Stützle, 2004).

5.2 CÁLCULO DE PROBABILIDADES DO ACO

A deposição do feromônio é um meio indireto de comunicação entre as formigas que serve como reforço positivo para favorecer o melhor caminho (ou opção) a ser escolhido (Neto e Filho, 2013). Essa deposição possui um valor inicial, que é um dos parâmetros do algoritmo e que é constante durante a primeira iteração, independentemente do número de formigas aplicado. Após a primeira iteração, quando todas as formigas já percorreram o caminho (ou, do inglês, *path*) a quantidade de feromônio depositado será proporcional ao valor da função objetivo, sendo essa conta podendo ser realizada de diferentes maneiras. Todavia, a taxa de deposição pode ser constante, ou seja, a cada iteração será depositada a mesma quantidade de feromônio no caminho utilizado para chegar no melhor valor da função objetivo dentre as formigas da iteração.

Segundo Aloise *et al.* (2002), o mecanismo evaporativo dos feromônios ajuda a evitar seu acúmulo ilimitado, o que levaria a uma estagnação do processo de busca dentro do espaço de soluções. Esse decaimento, usualmente linear, é uma estratégia de reforço negativo, contribuindo para a escolha entre as opções de cada formiga, nas próximas iterações, para as mais adequadas (Neto e Filho, 2013).

O cálculo da probabilidade de cada formiga depende de dois principais fatores: o fator heurístico e fator associado ao feromônio. O fator heurístico é calculado antes mesmo da primeira iteração e permanece inalterado no andamento do algoritmo, enquanto o fator relacionado ao feromônio é atualizado a cada iteração (Mullen *et*

al., 2009) O algoritmo de colônia de formigas pode ser descrito pela Equação 1 (Dorigo e Blum, 2005):

$$p_m(c_k^i) = \frac{\left[\tau_k^i\right]^\alpha . \left[\eta\left(c_k^i\right)\right]^\beta}{\sum_j \left[\tau_j^i\right]^\alpha . \left[\eta\left(c_j^i\right)\right]^\beta} \quad (1)$$

sendo c_k^i um componente da solução, τ uma função do conjunto solução que associa a cada componente à função objetivo, η uma função que associa a cada componente da solução um valor heurístico, α um valor positivo associado à quantidade de feromônio, β um valor positivo associado à informação heurística. Os índices representam: m, a formiga; i, a iteração; k, um possível elemento do conjunto solução; j todos os elementos do conjunto solução. Munido dos valores da matriz de probabilidades, é aplicada *random proportional rule,* ou seja, uma metodologia que aleatoriamente escolhe o conjunto solução ponderado pelo valor das probabilidades calculadas. A probabilidade $p_m(c_k^i)$ assume o valor 0 se essa transição para o nodo ij infringir alguma restrição. Para resolver o problema do caixeiro viajante, entre outros, diferentes variantes do ACO se adaptam aos valores de parâmetro: α = 1; 2 < β < 5.

Uma forma adaptada do algoritmo utilizado nesse trabalho é apresentada na Equação 2 a seguir:

$$p_m(c_k^i) = \frac{\left[Q.\tau_k^i\right]^\alpha . \left[\eta\left(c_k^i\right)\right]^\beta}{\sum_j \left[\tau_j^i\right]^\alpha . \left[\eta\left(c_j^i\right)\right]^\beta} \quad (2)$$

onde Q, ou constante de atualização do feromônio que adequa a grandeza da variável do conjunto solução com a do feromônio. A constante de atualização tem o papel de adequar a grandeza das variáveis envolvidas nas equações, por isso ela é utilizada em dois momentos: na atualização do feromônio, que conta com este fator para adequar a grandeza da função objetivo com a da quantidade de feromônio de cada iteração e no cálculo das probabilidades para adequar a ordem de grandeza do fator determinístico com o heurístico. Contudo, Colorni, Dorigo e Maniezzo (1992) afirmam que este parâmetro extra não aumenta a performance do algoritmo.

O fator heurístico não é único para cada problema e sua escolha é um dos pontos chaves para o sucesso da implementação do ACO. Considerando a experiência da ponte dupla, explicitada anteriormente, a escolha do fator heurístico é simples, pois o resultado depende apenas do tamanho de cada trecho, ou seja, a função objetivo deste problema seria uma função direta da distância entre o formigueiro e a fonte de comida. Para grande parte dos problemas em que as meta-heurísticas são uma abordagem conveniente, não há uma relação direta entre a função objetivo e todas as variáveis que são dados de entrada do mesmo, pois, se houvesse, era de se esperar que uma abordagem determinística fosse mais apropriada. Assim, a escolha do fator heurístico não é direta, e pode ser relacionado com apenas uma variável ou com várias, de acordo com a necessidade de cada problema.

5.3 ATUALIZAÇÃO E EVAPORAÇÃO DOS FEROMÔNIOS

Existem várias versões do algoritmo de colônia de formigas, assim como de cada etapa, como a de atualização (*feedback* positivo) e da evaporação dos feromônios (*feedback* negativo). Segundo Raghavendra e Prasanna (2011), as versões modificadas do algoritmo original de Dorigo e Stützle (2004) tentam balancear os fatores de intensidade e diversidade na atualização e evaporação dos feromônios. Esse equilíbrio é essencial para o desempenho satisfatório do algoritmo, pois a formiga intensifica a busca na região em que ocorreu os melhores resultados até certo ponto do andamento do algoritmo, tendo grandes chances de ficar presa a um ótimo local, enquanto aquela que tem uma forte diversificação, mas sem intensificação, pode demorar muito para encontrar o ótimo global. Assim, com o balanceamento ideal entre a intensificação e a diversificação tem-se a melhor performance do algoritmo tanto em qualidade do resultado quanto em tempo de convergência. Uma técnica que pode ser empregada é o mecanismo de aglomeração que atualiza a quantidade de feromônio depositando uma mesma quantidade de feromônio na região do entorno do melhor resultado para cada iteração; essa metodologia adiciona mais diversidade no processo de intensificação, pois não reforça apenas o melhor resultado como no algoritmo original.

As formigas podem depositar feromônios associados a estados ou, com transições de estado. E essa deposição pode ocorrer a cada estado durante a construção da

solução, o que é conhecido, do inglês, como *online step-by-step pheromone trial update*, alternativamente à atualização tardia de feromônio, que é o caso em que o feromônio só é atualizado após a função objetivo é encontrada. Um aprimoramento possível ao reforço positivo do depósito de feromônio pode ser adicionar uma quantidade extra de feromônio a melhor solução encontrada até o momento (Mullen *et al.*, 2009).

Independentemente do método de atualização, é formada uma matriz de trilhas de feromônios que guarda as informações vinculadas aos conjuntos solução que resultaram no sucesso (melhor função objetivo) do problema.(Colorni *et al.*, 1994) A atualização é realizada primeiramente diminuindo a quantidade total de feromônios em cada trecho por um fator constante e, em seguida, aumentando a quantidade de feromônio de acordo com a regra escolhida (Dorigo e Stützle, 2004).

A atualização do feromônio é realizada segundo a Equação 3.(Dorigo e Blum, 2005):

$$\tau_j^i = \tau_j^i + \Delta\tau^k \tag{3}$$

Alternativamente ao algoritmo clássico (ACO), que utiliza o cálculo de probabilidade demonstrado acima, existem muitas variações, cada qual com suas particularidades. Uma delas, o ACS, utiliza a regra proporcional pseudoaleatória a qual, invés de utilizar a probabilidade $p_m(c_k^i)$ em todos os casos, a utiliza com probabilidade $(1-q_0)$, sendo q_0 um valor de 0 a 1; com probabilidade q_0 é utilizada a solução que maximiza o termo $\left[\tau_k^i\right]^\alpha.\left[\eta\left(c_k^i\right)\right]^\beta$ (Dorigo e Blum, 2005).

A evaporação dos feromônio é feita através da Equação 4(Dorigo e Stützle, 2004):

$$\tau_j^i \leftarrow (1-\rho)\tau_j^i \tag{4}$$

sendo 0 < ρ ≤ 1 a constante de taxa de evaporação. Um arco que não é percorrido pelas formigas tem sua quantidade de feromônio reduzida exponencialmente com o número de iterações. Para resolver o problema do caixeiro viajante, entre outros, diferentes variantes do ACO se adaptam aos valores de parâmetro: α = 1; 2 < β < 5; 0,02 < ρ < 0,5.d

Algumas variantes do ACO como o AS (*Ant System*) elitista, o AS baseado em rankings e o Max-Min AS são diferenciadas pela metodologia de atualização dos feromônios.(Dorigo e Stützle, 2004) Em uma variante do ACO, a quantidade de feromônios é atualizada de acordo com uma estratégia elitista (como em algoritmos evolucionários), na qual a cada iteração, apenas os caminhos que levaram ao melhor resultado até agora têm os feromônios depositados. Esse mecanismo aumenta a intensificação do algoritmo, e evita a estagnação precoce do processo de busca (Dorigo e Blum, 2005).

Foi observado, em algumas espécies de formiga, que quando a densidade de feromônio se torna muito alta em um dos trechos, os indivíduos passam a escolher o percurso menos congestionado. Assim, essas formigas formam uma nova ramificação, garantindo uma eficiência maior, já que evita o tráfego intenso na trilha congestionada e forma um novo caminho até a comida.

O ACS (Dorigo e Stützle, 2004) utiliza a regra proporcional pseudoaleatória para cálculo das probabilidades e a atualização da trilha de feromônios (evaporação e depósito), aplica-se apenas aos arcos de da melhor formiga até o momento (*best-so-far ant*), ver Equação 6, enquanto no AS, a atualização acontece em todos os arcos. Assim, a complexidade computacional da atualização do feromônio em cada iteração no ACS é reduzida.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall(i,j) \in T^{bs} \tag{5}$$

Além da regra geral de atualização dos feromônios, o ACS também aplica uma regra imediatamente após cada passo da formiga, apresentada na Equação 7 a seguir.

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0 \tag{6}$$

onde $0 < \xi < 1$ e τ_0 é o mesmo valor de feromônio inicial utilizado na primeira iteração. Considerando as Equações 5 e 6, a quantidade de feromônio em cada arco nunca será menor que τ_0.

Segundo Colorni *et al.* (1994), cada formiga, se isolada (isto é, se $\alpha = 0$), se moveria de acordo com uma heurística local e gananciosa originando movimentos que levam a ótimos locais. Essa prática não origina bons resultados porque essas melhorias locais gananciosas costumam a levar a resultados finais muito ruins, por isso o que ocorre é que cada formiga contribui para uma parte da distribuição da trilha,

caracterizando o *feedback* positivo (processo autocatalítico), o qual converge em boas soluções rapidamente, sem ficar preso em ótimos locais.

Para escolher o valor da quantidade inicial de feromônio, uma boa prática segundo Dorigo e Stützle (2004), é utilizar um valor ligeiramente superior à quantidade esperada de feromônio a ser depositada pelas formigas em uma iteração. Uma sugestão de cálculo de valor inicial é dada pela Equação 7 a seguir:

$$\tau_{ij} = \tau_0 = \frac{m}{C^{nn}}, \quad \forall(i, j) \tag{7}$$

na qual m é o número de formigas e C^{nn} é uma estimativa de resultado da função objetivo para a primeira iteração. Desse modo, após a primeira iteração o algoritmo já começa a se desenvolver livre da influência da estimativa inicial; se ela fosse um valor muito alto, as primeiras iterações iriam ser desperdiçadas.

5.4 INFORMAÇÃO HEURÍSTICA

Heurísticas são diretrizes utilizadas para orientar a pesquisa ao atribuir valores às variáveis, porém elas não são 100% confiáveis; uma heurística que é frequentemente usada consiste na escolha dos valores mais promissores (Solnon, 2010). A informação puramente heurística do ACO, também conhecida como atratividade, é representada pela matriz de visibilidade e corresponde aos valores da variável η_{ij} já explicitada no capítulo 5.2. Segundo Socha e Dorigo (2008), essa variável é a estimativa do desejo das formigas de realizarem a transição do nodo i ao j.

De acordo com Dorigo e Stützle (2004), a heurística deve permitir a geração de um grande número de soluções de partida diferentes, não enviesando excessivamente as soluções iniciais do problema e permitindo maior exploração dos possíveis resultados. Uma maneira de encontrar as heurísticas mais apropriadas para seu problema pode ser se fazendo os seguintes questionamentos:

- Quais são os componentes de solução apropriados?
- Como é medida a contribuição da variável na função objetivo?
- É sempre possível construir soluções candidatas viáveis?

- Quantas soluções diferentes podem ser geradas com essa heurística?

5.5 CRITÉRIO DE PARADA

Não existe um critério de parada universal para as diversas meta-heurísticas (Dorigo e Stützle, 2004). O critério de parada, ou seja, o parâmetro que irá decidir que o algoritmo de otimização foi terminado, pode ser o atingimento do número máximo de iterações ou do tempo computacional determinado (Neto e Filho, 2013). O critério de parada também pode envolver mais de um parâmetro para que aumente a viabilidade do cenário ótimo ser atingido. Assim, um parâmetro que indique a convergência a um resultado, como a frequência de aparecimento de uma solução considerada boa em detrimento de outras, unido com um parâmetro de número máximo de iterações ou tempo computacional, seria indicado.

Uma função de critério de parada em um algoritmo de otimização tem como dados de entrada algumas variáveis do problema, como o número de iterações até aquele momento, os melhores valores de função objetivo, o tempo decorrido desde o início do processo. Essa função pode ser composta de uma combinação de tipos critérios de paradas complementares ou apenas de um deles. Como dados de saída essa função fornece uma resposta binária de acordo com os critérios estabelecidos; assim, se o resultado for 0 (falso) o algoritmo passa para aproxima iteração e se o resultado for 1 (verdadeiro) o algoritmo é finalizado naquela iteração e retorna o(s) resultado(s) obtido(s). Alguns tipos de critério de paradas são citados a seguir (Weise, 2009):

- <u>O tempo computacional máximo especificado ser atingido</u>: para definir o critério por tempo computacional é necessário levar em consideração o tempo necessário para cada iteração.
- <u>O número máximo especificado de iterações ser atingido</u>: é um critério particularmente interessante para pesquisadores, mas nem sempre adequado para aplicações comerciais.
- <u>Falta de melhoria no resultado é observada</u>: se a função objetivo deixa de progredir (aumentar o valor se o objetivo for maximizar ou diminuí-lo se o objetivo for minimizar) é um forte indício que a otimização foi atingida o

algoritmo ficou “preso” em um ótimo local. Assim, se a qualidade do resultado para de demonstrar avanço durante um determinado número de iterações, o algoritmo, provavelmente não se desenvolverá mais.

- Solução suficientemente boa é atingida: o critério de parada é o próprio cumprimento do objetivo da otimização se esse for possível de ser diretamente identificado.

Segundo Dorigo e Stützle (2004), não tem outra maneira de medir o desempenho ACO além de executar testes experimentais extensivos.

5.6 VANTAGENS E DESVANTAGENS DO MÉTODO

Além do ACO, existem muitos outros métodos meta-heurísticos, como *simulated annealing* (SA) conhecido como têmpera simulada, recozimento simulado ou arrefecimento simulado; *tabu search* (TS) ou busca tabu; *guided local search* (GLS) ou busca local guiada; greedy randomized adaptive search procedures (GRASP); *iterated local search* (ILS) ou busca local iterativa; *evolutionary computation* (EC) ou computação evolucionária; e *scatter search* ou busca dispersa. Todas essas meta-heurísticas podem ser agrupadas em construtivas ou baseadas em busca local, sendo o ACO e o GRASP métodos construtivos e as outras meta-heurísticas citadas baseadas em busca local. Além dessa classificação, é possível dividir as meta-heurísticas em: nas que uma população de soluções (como ACO e EC) e nas que manipulam uma única solução (as outras citadas). A utilização do recurso da memória também é um fator que diferencia as metodologias, sendo que o ACO, TS e GLS o utilizam e as outras não; no ACO são os feromônios os responsáveis pelo mecanismo de memória. Logo, o ACO une várias características que o distingue das outras meta-heurísticas, já que é o único método construtivo que utiliza uma população (colônia de formigas) que desenvolvem soluções a partir de uma forma de memória indireta (feromônios artificiais) (Dorigo e Stützle, 2004).

O ACO tem sua aplicação reconhecida para algumas classes de problemas, como para problemas NP-difícil, problemas dinâmicos que envolvem a identificação do caminho mais curto como a variante dinâmica do problema do caixeiro viajante e problemas em que a arquitetura computacional é espacialmente distribuída (Dorigo e Stützle, 2004).

Uma das vantagens do método (assim como outros algoritmos meta-heurísticos) é o tempo de resolução geralmente aceitável, apesar de não, necessariamente, encontrar o ótimo global (Tavares Neto e Godinho Filho, 2013). Adicionalmente, o ACO é considerado um método robusto que possui um forte caráter adaptativo, respondendo bem a mudanças, passível de ser utilizado em aplicações dinâmicas (Bedi e Singh, 2013).

Apesar do algoritmo possuir convergência garantida, o tempo de atingimento varia, além de ser propenso a permanecer confinado em ótimos locais. O código de programação não é muito direto e intuitivo (Bedi e Singh, 2013). De acordo com Luan *et al.* (2019), a eficiência do algoritmo ACO é influenciada pelo número total de formigas empregado, sendo que, quanto maior o número de formigas melhor o resultado até chegar a um limite (ponto crítico) no qual essa melhora é estagnada em torno de um determinado valor.

5.7 ACO E PROBLEMAS DE *SCHEDULING*

A otimização por colônia de formigas já foi aplicada com sucesso para resolver problemas de *scheduling*, a exemplo daqueles envolvendo uma única máquina com opção de terceirização (Neto e Filho, 2013) e utilizando máquinas paralelas não relacionadas (Kumar *et al.*, 2011).

O ACO se mostrou mais eficiente do que um conjunto de algoritmos baseados em técnicas de programação dinâmica para resolver problemas de otimização de *scheduling* multicritério envolvendo apenas uma etapa de processamento (ou máquina única) com opção limitada de terceirização (Neto e Filho, 2013).

Elaborar uma representação em formas de grafo para o problema de *scheduling* é um dos primeiros passos para a resolução por ACO (Neto e Filho, 2013). Assim, é possível relacionar de forma simplificada relações entre os elementos do problema e trabalhar na resolução em forma matricial.

O algoritmo de otimização de problemas de *scheduling* pode ser descrito na forma de pseudocódigo, como mostrado a seguir – adaptado (Dorigo e Stützle, 2004):

Procedimento ACO

    Inicialização dos valores de feromônio

Conjunto solução = conjunto vazio
Enquanto o número de iterações máximo não for atingido faça
Atividades de *schedule*
Construção de soluções das formigas
Atualização dos feromônios
Fim das atividades de *schedule*
Fim do Enquanto
Fim do procedimento ACO

De acordo com Colorni et al. (1994), um problema de *scheduling* do tipo *job shop*, o AS (*Ant System*) pode ser aplicado utilizando restrições em cada transição entre nodos para identificar a possibilidade de realização de cada permutação. Isso pode ser feito através de uma lista tabu de sequências na trajetória e de limitações associadas a solução do problema. Para operacionalizar essas transições, pode-se afirmar que todas as formigas partem do ponto u_0, sendo que se houverem vários pontos de partidas no problema consideraremos esse ponto precedente às origens do *scheduling*.

Para a solução do problema de *scheduling* em Colorni et al. (1994), o número de formigas foi definido para ser igual ao número n de tarefas e o número de iterações utilizadas foram 3000 e foi considerado o melhor resultado dentre 5 corridas. Foi identificado que a magnitude da sensibilidade dos parâmetros com a dimensão do problema é muito baixa. Quando ocorre a estagnação do resultado do algoritmo, situação em que todas as formigas fazem o mesmo caminho, o AS não explora novas soluções e, portanto, o melhor caminho alcançado antes da estagnação não será mais melhorado. Experimentalmente foi identificado que apenas configurações de parâmetros muito particulares fizeram esse sistema entrar no comportamento de estagnação: utilizando altos valores do parâmetro α. O melhor valor ρ foi 0,7 em virtude da necessidade algoritmo “esquecer” parte da experiência adquirida no passado para ser capaz de explorar melhor as novas informações globais recebidas.

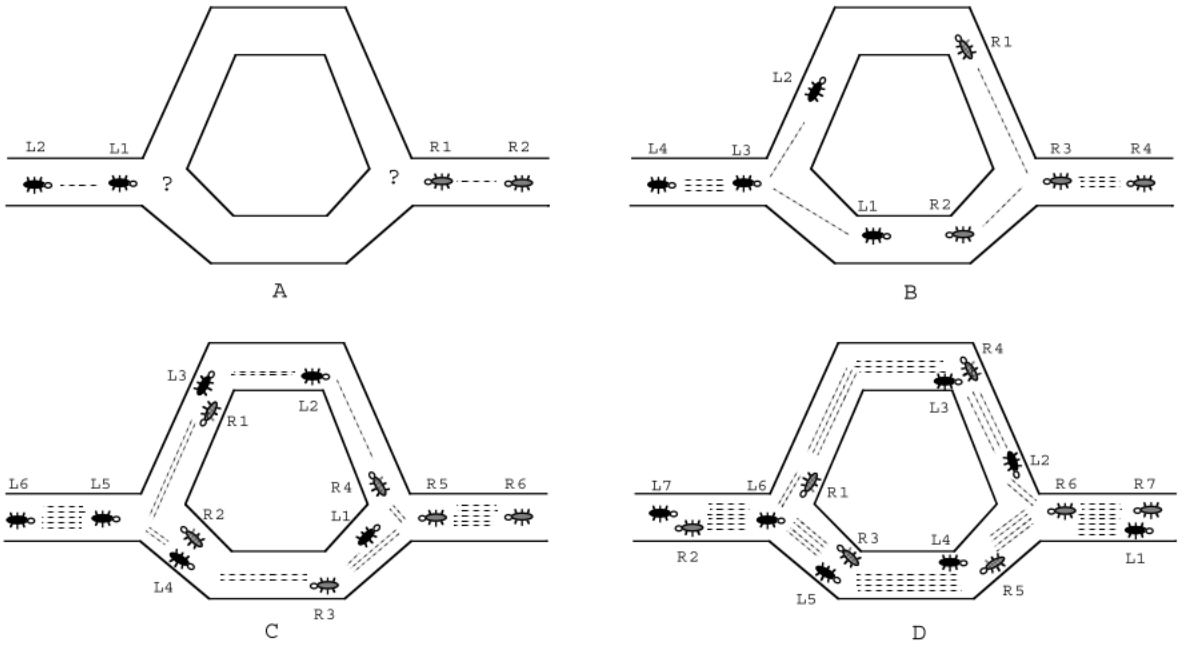
6. AVALIAÇÃO PRELIMINAR DO ALGORITMO DE COLÔNIA DE FORMIGAS

Neste capítulo serão apresentadas três aplicações do algoritmo de colônia de formigas elaboradas em Matlab®: o experimento da ponte dupla, o problema do caixeiro viajante e um problema de scheduling em uma máquina. Os dois primeiros com ênfase na influência dos parâmetros em cada um deles e os resultados obtidos e o ultimo com uma análise aplicada a um problema simples de *scheduling*.

6.1 EXPERIMENTO DA PONTE DUPLA

O experimento da ponte dupla consiste, basicamente das 4 etapas mostradas na Figura 13, na qual as linhas tracejadas representam a quantidade de feromônio depositado: A) As formigas precisam escolher entre dois caminhos. B) As formigas escolhem aleatoriamente entre os dois caminhos disponíveis. C) Considerando que as formigas se movem na mesma velocidade e sendo esta constante, as formigas que escolhem o caminho mais curto chegam ao destino mais rápido do que as que escolheram o caminho mais longo. D) O caminho mais curto acumula feromônios em uma taxa maior, tornando esse caminho mais propício a ser seguido pelas próximas formigas.

Figura 13 - Experimento da ponte dupla



Fonte: Dorigo e Gambardella (1997)

A aplicação do ACO ao experimento da ponte dupla, explicado com mais detalhes no tópico 4.5.1 e representado na Figura 13, foi implementado no Matlab®. A informação heurística utilizada é o inverso da distância entre o formigueiro e a fonte de alimento e a função objetivo é a minimização da distância percorrida.

O estudo do experimento da ponte dupla foi dividido três partes:

1. Utilizando duas opções de caminhos de mesmo comprimento entre o formigueiro e a fonte de comida. O resultado esperado é de que, como as primeiras formigas escolham o caminho percorrido aleatoriamente, e que os feromônios sejam depositados, majoritariamente, em um desses dois caminhos. Isso ocorre, pois, o caminho eventualmente escolhido primeiro irá concentrar mais feromônios com o tempo. Assim, em cada experimento, as formigas irão convergir pra um dos caminhos sem preferência de um deles.
2. Utilizando duas opções de caminhos, sendo que a segunda rota tem o dobro do comprimento da primeira. O resultado esperado é de que, as formigas sempre convirjam para a rota de menor distância, já que a concentração de feromônios nesse percurso é maior.

3. Um estudo de sensibilidade de parâmetros para a parte dois, evidenciando como a parametrização influencia fortemente no resultado da otimização mesmo que para um exemplo simples em que a função objetiva é obtida diretamente pelos fatores heurísticos do problema.

A implementação do algoritmo é mostrada na Figura 13 que representa o programa principal, onde *Heu* é a variável que representa a heurística do problema, α um valor positivo associado à quantidade de feromônio, β um valor positivo associado à informação heurística, fer0 é o valor inicial do feromônio em cada um dos trechos (é sempre igual para os dois trechos), ii é a iteração, m a formiga e FO a função objetivo.

Figura 14 - Programa principal da experiência da ponte dupla

```
%Distâncias
d(1) = 1;
d(2) = 1;

%Heurística
Heu = [1/d(1) 1/d(2)]; %Caminhos iguais
imax = 1;
jmax = 2;
n = 50; %iterações
ants = 100; %quantidade de formigas por iteração
fer_atu = 0.3;
alfa = 1;
beta = 1;
fer0 = 0.1;
fer = [fer0, fer0];

for  ii=1:n
    fer_save(ii,:) = fer;
    for m=1:ants
        [x(m), prob_x(m,:) ] = prob_gen( imax, jmax, fer, alfa, beta, Heu);
        FO(m) = 1/d(x(m));
        [~,bestsoFar] = find(FO == max(FO));
        FO_ant(ii,m) = FO(m);
        x_ant(ii,m) = x(m);
    end
    prob(ii,:) = prob_x(bestsoFar(1),:);

    %Atualização
    fer(x(bestsoFar(1))) = fer(x(bestsoFar(1))) + fer_atu;
    FO_best(ii) = FO(bestsoFar(1));
end
```

Nas Figura 15 e Figura 16 são as funções chamadas pelo programa principal. A função da Figura 15 calcula a probabilidade que cada formiga tem de seguir por um dos dois caminhos, de acordo com os parâmetros do problema e da quantidade de

feromônios depositados. A função da Figura 16 é uma roleta de um ponteiro que, de posse dos valores das probabilidades escolhe o resultado da variável.

Figura 15 – Programa da função que calcula as probabilidades e as variáveis do problema

```
function [ var, prob ] = prob_gen( imax, jmax, fer, alfa, beta, Heu)
%Calcula as probablidades e o resultado da variável decisória
      soma = zeros(imax,1);
      for i=1:imax
            for j=1:jmax
            soma(i) = soma(i) + fer(i,j)^alfa*Heu(i,j)^beta;
            end
      end

      prob = zeros(imax,jmax);%inicializando variável

      for i=1:imax
          for j=1:jmax
              prob(i,j) = (fer(i,j)^alfa*Heu(i,j)^beta)/soma(i);
          end
      end

      var = zeros(1,imax);
      for i=1:imax
          var(i) = roleta(prob(i,:));
      end
end
```

Figura 16 - Programa da função que calcula o valor da variável

```
function f = roleta(prob)
    r = rand;
    C = cumsum(prob);
    f = find(r <= C, 1, 'first');
end
```

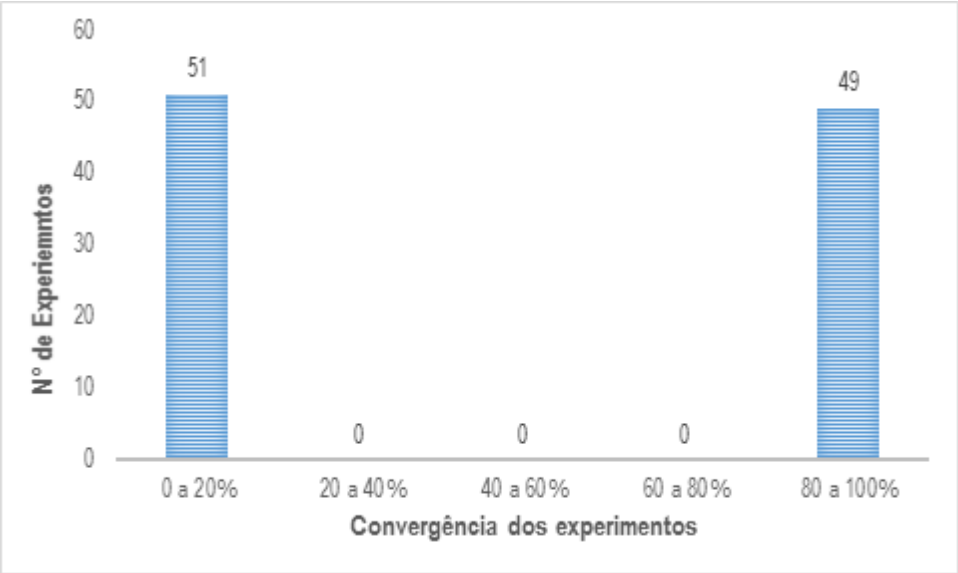
Os principais parâmetros do algoritmo, os quais são sujeitos do estudo sensibilidade são: a atualização do feromônio, alfa e beta.

6.1.1 CAMINHOS DE MESMO COMPRIMENTO

Considerando as duas distâncias “d” iguais a 1 unidade de distância, rodando o algoritmo 100 vezes, o resultado obtido para as rodadas é mostrado no gráfico da Figura 17. Foram utilizadas 100 formigas em 100 iterações e esse experimento foi repetido 100 vezes. Das 100 vezes que o algoritmo foi rodado (dos 100 experimentos), 51 delas de 0 a 20% das formigas convergiram para o caminho 1 e em 49 deles de 80 a 100% delas convergiram para o caminho 1. Esse resultado

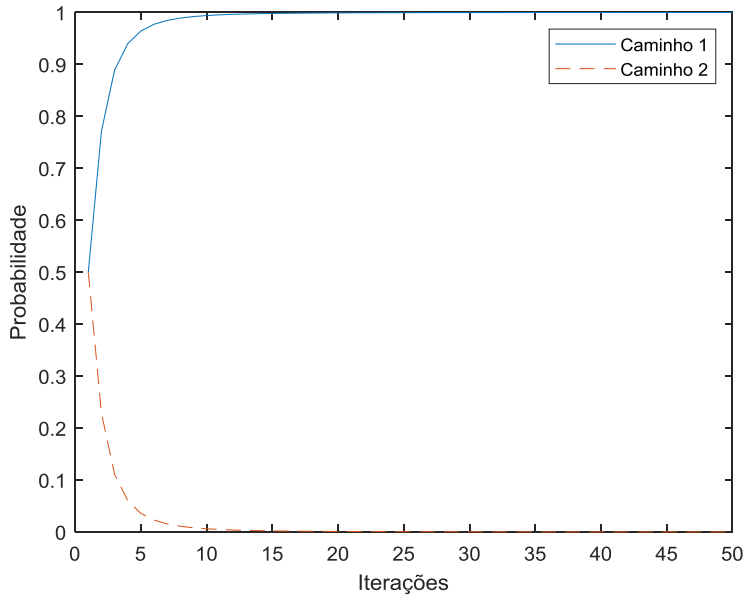
comprova a hipótese inicial na qual as formigas na primeira iteração iriam aleatoriamente para um dos caminhos fazendo com que a quantidade de feromônio nesse caminho escolhido primeiro aumente até que praticamente todas as formigas convirjam para um destes. O algoritmo completo se encontra no Apêndice A desse trabalho.

Figura 17 - Resultado do experimento da ponte dupla para distâncias iguais



Um exemplo da convergência em um dos experimentos, representado pela evolução da variável probabilidade de cada formiga escolher um dos dois caminhos a cada iteração é mostrado na Figura 18; nota-se que na primeira iteração as probabilidades de cada caminho são idênticas, já que a informação heurística tem igual peso para as duas escolhas. Tanto a convergência quanto a resposta final são muito influenciados pelos parâmetros, o que será mostrado no tópico 4.1.3.

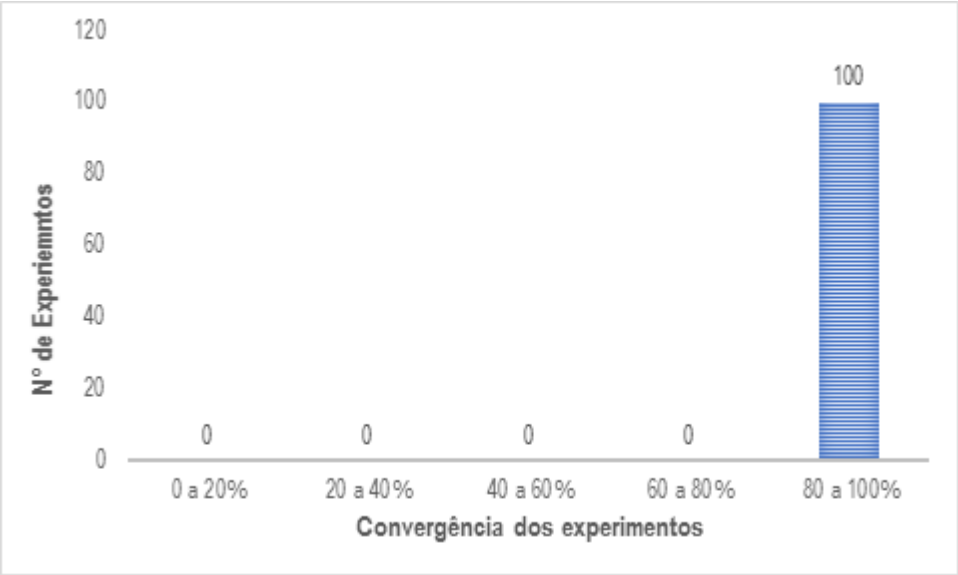
Figura 18 - Evolução da probabilidade no exemplo da ponte dupla com distâncias iguais



6.1.2 CAMINHOS DE COMPRIMENTOS DISTINTOS

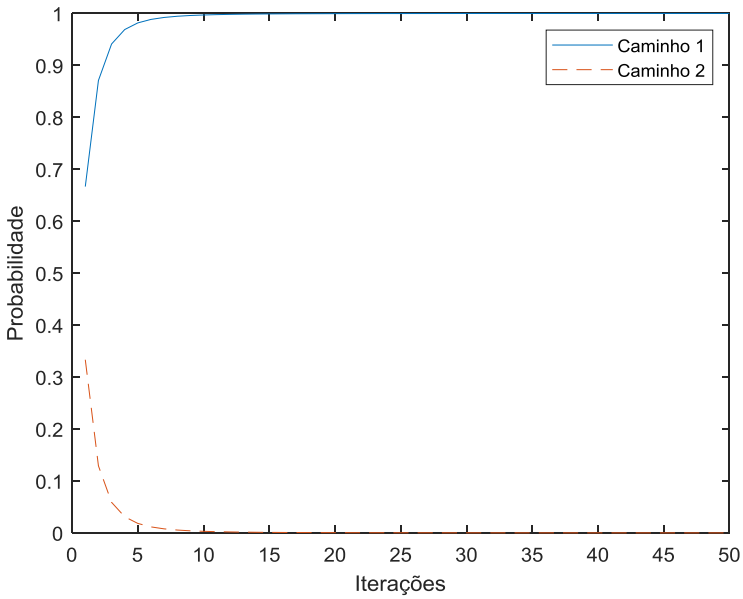
Assim como no caso de distâncias iguais (Tópico 4.1.1), foram utilizadas 100 formigas em 100 iterações e esse experimento foi repetido 100 vezes. Considerando o caminho 2 com o dobro da distância do caminho 1, rodando o algoritmo 100 vezes, o resultado obtido para as rodadas é mostrado no gráfico da Figura 19. Das 100 vezes que o algoritmo foi rodado (dos 100 experimentos), em todos os eles de 80 a 100% das formigas na última iteração convergiram para o caminho 1.

Figura 19 - Resultado do experimento da ponte dupla para distâncias diferentes



A evolução da probabilidade em função das iterações é mostrada na Figura 20. Graças a informação heurística em função do comprimento de cada trecho, nota-se que já na primeira iteração, a probabilidade de escolha do caminho 1 é maior do no caminho 2. O problema converge rapidamente antes das 10 primeiras iterações.

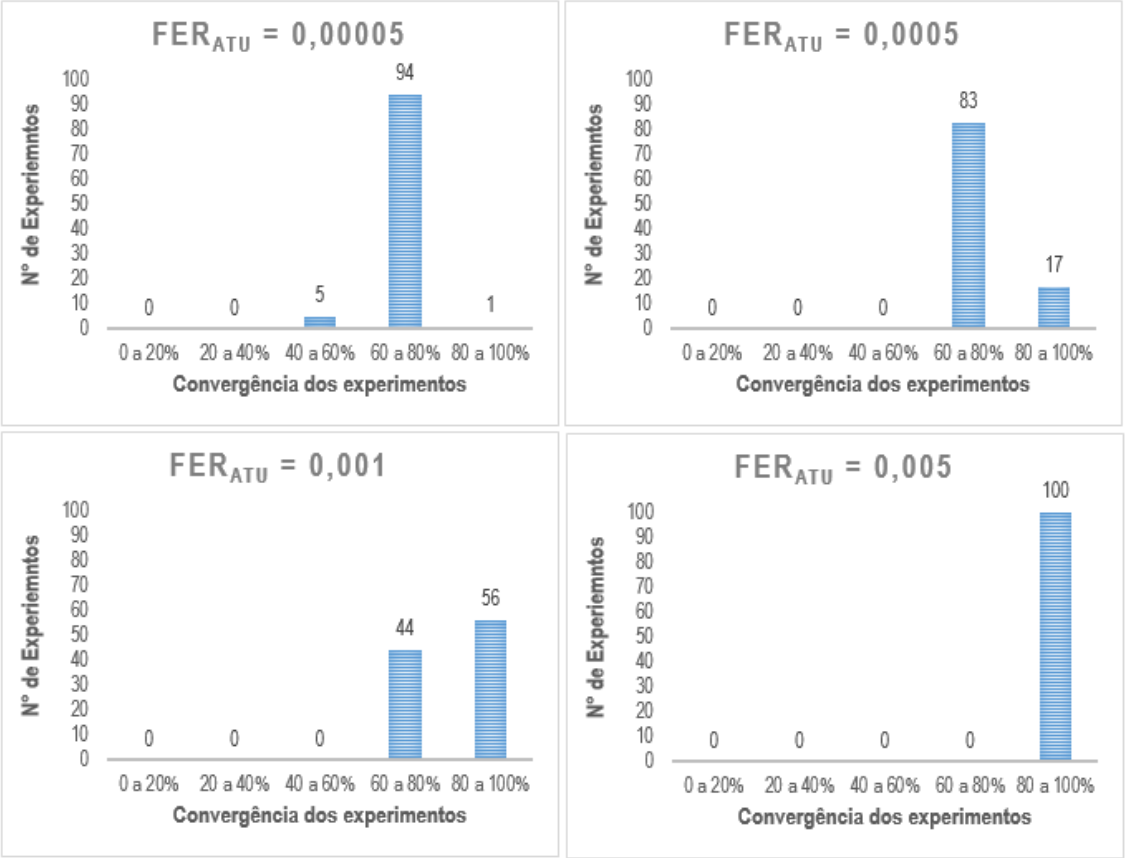
Figura 20 - Evolução da probabilidade no exemplo da ponte dupla com distâncias diferentes



6.1.3 SENSIBILIDADE DOS PARÂMETROS

O resultado da sensibilidade de três parâmetros é apresentado nesse tópico. Os parâmetros mantidos fixos durante toda a sensibilidade foram a quantidade de iterações (n = 100), de formigas (m =100), de experimentos (exp = 100), de feromônio inicial em cada trecho (fer0 = 0,1) e o parâmetro heurístico que é o inverso da distância. O primeiro parâmetro é ser estudado é o de atualização de feromônios (fer_{atu}). Mantendo alfa e beta iguais a 1, a quantidade de feromônio acrescentado em cada iteração pela formiga de melhor resultado foi variada conforme a Figura 21.

Figura 21 - Sensibilidade do parâmetro de atualização de feromônio no exemplo da ponte dupla

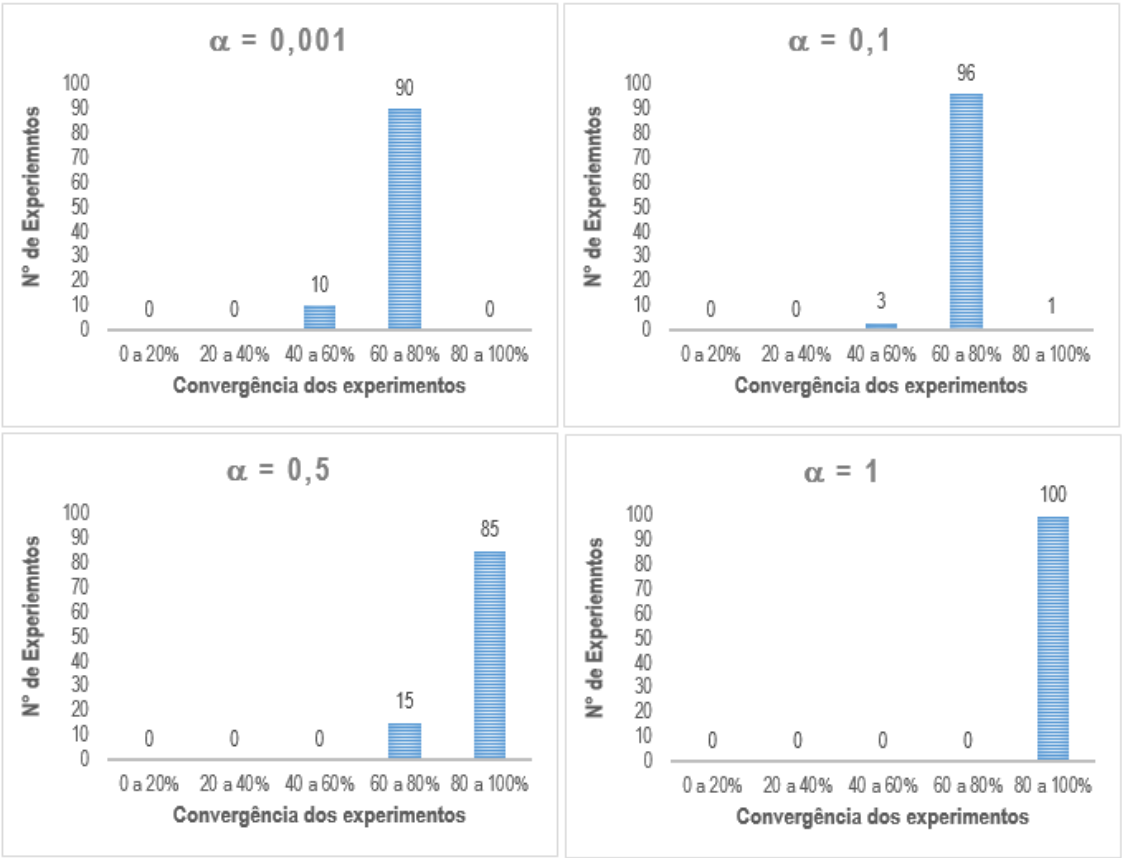


A partir do valor final da sensibilidade (fer_{atu} = 0,005) o resultado passa a se repetir, ou seja, todas as iterações convergem para o resultado desejado (menor caminho). Quando a função objetivo tem uma relação direta com as variáveis, como no exemplo da ponte dupla, já era de se esperar que quanto maior a atualização dos feromônios melhor a convergência do algoritmo, já que ela representa a influência

do “acerto” das formigas nas iterações anteriores na escolha das formigas nas próximas iterações.

O segundo parâmetro é ser estudado é o de influência de feromônios (α). Mantendo fer_{atu} e beta iguais a, respectivamente, 0,005 e 1, o valor de α em cada iteração foi variado conforme a Figura 22. Utilizando o valor de atualização de feromônios que apresentou o melhor resultado em sua sensibilidade, o parâmetro alfa ainda se mostra decisivo na convergência do algoritmo. Quanto maior o valor de alfa maior a convergência.

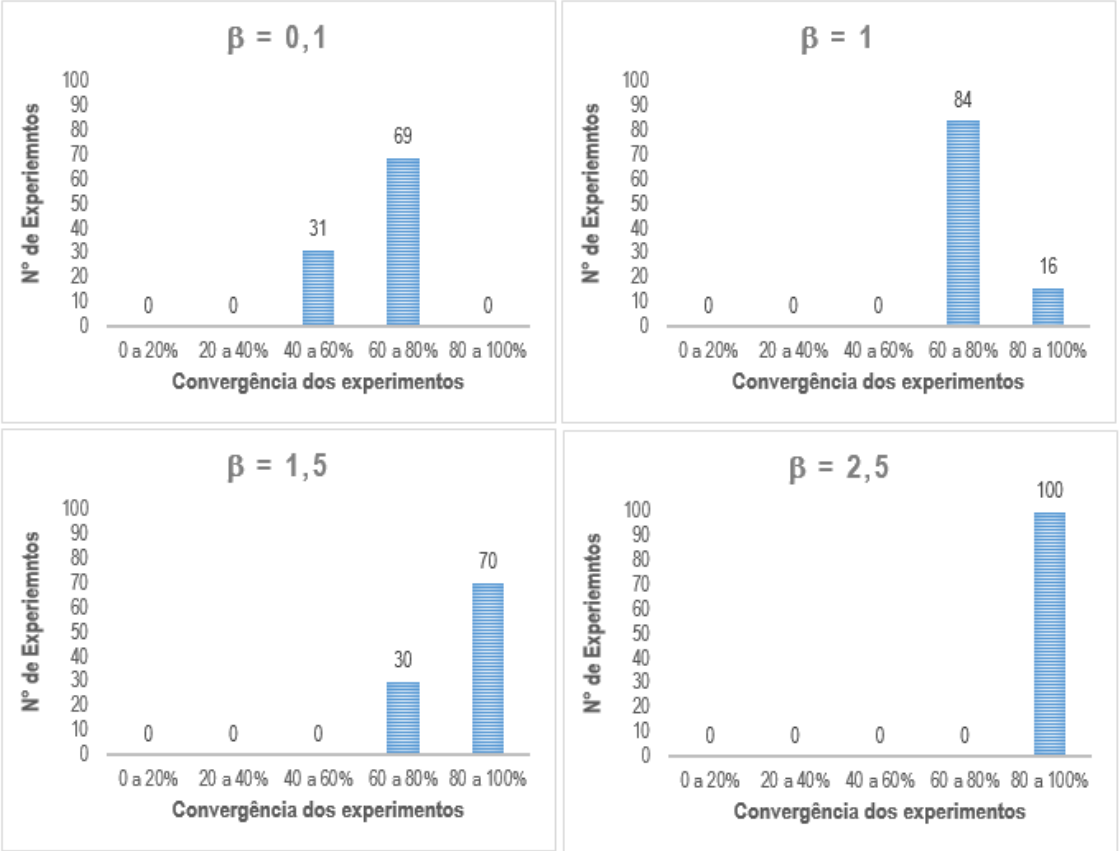
Figura 22 - Sensibilidade do parâmetro alfa no exemplo da ponte dupla



O terceiro parâmetro é ser estudado é o de influência do fator heurístico (β). Mantendo fer_{atu} e alfa iguais a, respectivamente, 0,0005 e 1, o valor de β em cada iteração foi variado conforme a Figura 23. Utilizando o valor de atualização de feromônios que apresentou o melhor resultado em sua sensibilidade, o parâmetro alfa ainda se mostra decisivo na convergência do algoritmo. Quanto maior o valor de

beta maior a convergência. Foi necessário utilizar um valor menor da atualização de feromônio afim de mostrar melhor a influência do beta na convergência.

Figura 23 - Sensibilidade do parâmetro beta no exemplo da ponte dupla



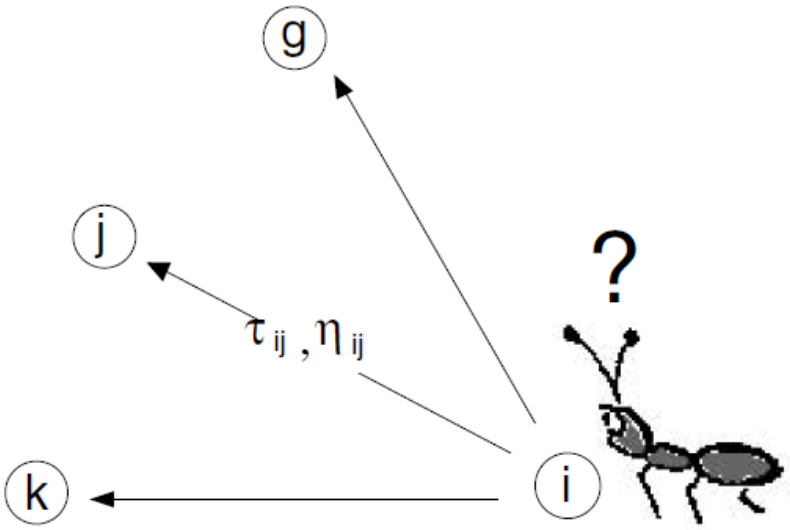
Os três parâmetros exercem grande influência no resultado, mesmo se tratando de um exemplo tão simples com esse da Ponte Dupla. A relação entre eles e a convergência é direta por se tratar de um problema linear e com a função objetiva definida justamente em função dos dados de entrada e da heurística estar perfeitamente relacionada com a função objetivo. Assim, quanto maiores os valores de alfa, beta e atualização do feromônios, melhor o resultado.

6.2 PROBLEMA DO CAIXEIRO VIAJANTE

O Problema do Caixeiro Viajante (ou, do inglês, *Travelling Salesman Problem* - TSP) é a otimização do trajeto de um vendedor que, a partir de uma cidade (ou ponto) inicial, precisa minimizar a distância percorrida passando por determinado conjunto de cidades-clientes, de modo que cada cidade é visitada apenas uma vez e o ponto

final é a mesma cidade em que se iniciou o trajeto. Com a finalidade de resolver o TSP usando ACO, é necessário calcular a probabilidade de cada formiga se mover da cidade para a cidade j conforme ilustrado na Figura 24. (Dorigo e Stützle, 2004)

Figura 24 - Escolha de cada formiga na resolução do TSP utilizando ACO



Fonte: Dorigo e Stützle (2004)

Para entender um pouco melhor a complexidade do TSP Simon (2013) afirma

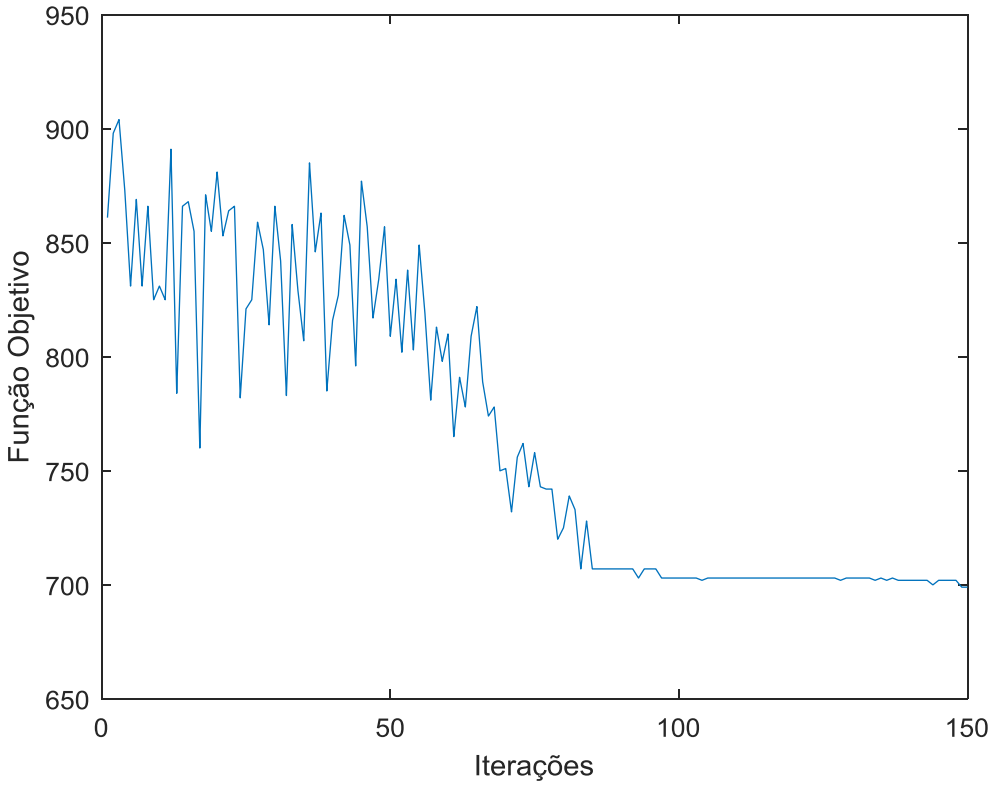
> Há algo entre 10^{20} e 10^{24} grãos de areia na Terra. Se cada grão de areia na Terra fosse um planeta parecido com a Terra com a mesma quantidade de areia que a Terra, então o número de possíveis rotas TSP de 50 cidades ainda seria muito maior do que o número total de mini grãos de areia. Obviamente, é impossível fazer uma pesquisa de força bruta para a solução de um problema tão grande.

Para resolver um problema de TSP de 42 cidades utilizando o algoritmo de colônia de formigas, as cidades e suas distâncias são modeladas em forma de grafos e a heurística utilizada é análoga à empregada no problema da ponte dupla, ou seja, é o inverso das distâncias. Assim, nas primeiras iterações, a probabilidade da próxima cidade a ser visitada ser uma mais próxima é maior do que as mais distantes.

6.2.1 VALIDAÇÃO DOS RESULTADOS

Foi utilizado um exemplo com 42 cidades para ilustrar a implementação e avaliação do desempenho do ACO com o problema do TSP. Os dados, assim como o código em Matlab®, podem ser encontrados no Apêndice B. O resultado ótimo global do problema é a menor distância possível de ser percorrida passando por todas as cidades, que é de 699 unidades de distância.(Reinelt, 1991) Na Figura 25 , pode ser observado que o resultado foi alcançado; foram necessárias 150 iterações com 100 formigas cada, os parâmetros utilizado foram α = 1, β = 4, evaporação do feromônio = 0,06, atualização do feromônio = 1/FO e feromônio inicial = 0.1111 (razão entre o número de formigas e a estimativa de função objetivo inicial que era de 900). Os parâmetros foram obtidos por tentativa e erro.

Figura 25 - Resultado do TSP utilizando ACO



6.2.2 SENSIBILIDADE DOS PARÂMETROS

Como esse exemplo é mais complexo que o da ponte dupla, para melhor estudo da parametrização, foi necessário realizar diversas rodadas do programa e identificar, de modo estatístico, as melhores soluções. Assim, cada coluna das tabelas de sensibilidade, mostradas a seguir, mostram o resultado de 20 rodadas; em cada

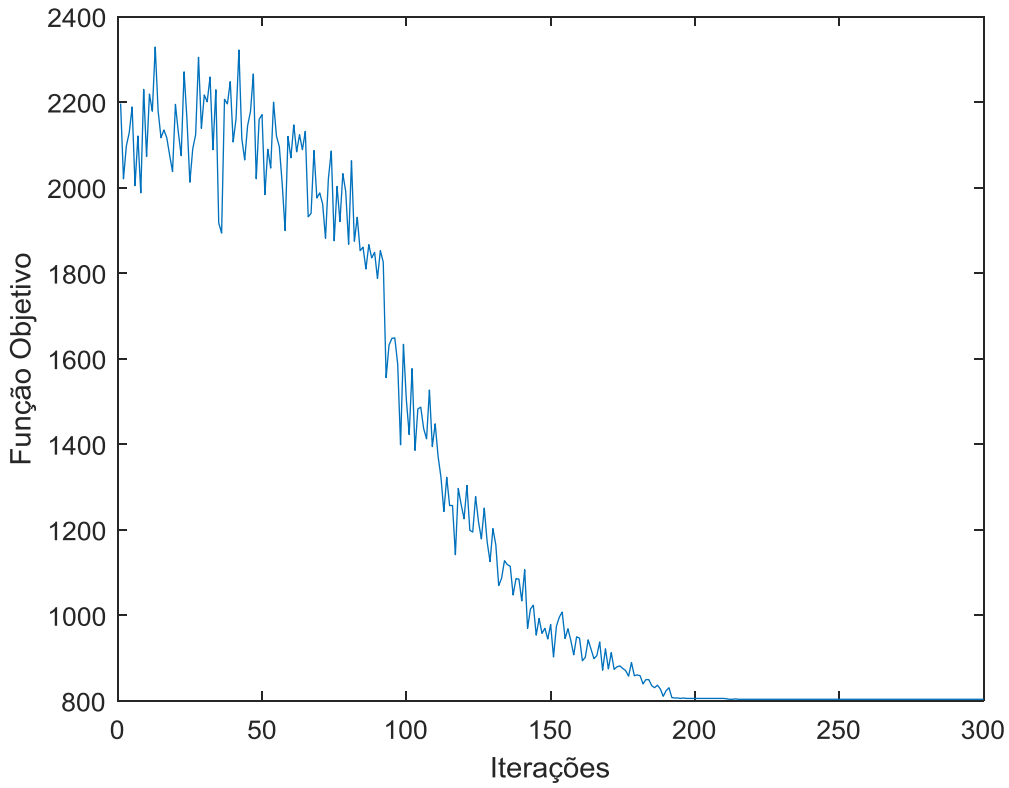
uma dessas rodadas é identificado o FO da última iteração (que pode ter convergido para um valor ou não) e é anotado nas tabelas, para cada uma das 20 rodadas, qual o FO de mínimo encontrado entre elas e qual a média. Primeiramente, foi realizada a sensibilidade do parâmetro beta, já que este problema se demonstrou fortemente dependente deste parâmetro. O resultado pode ser visualizado na Tabela 2, na qual as duas primeiras linhas são os valores para os parâmetros que permanecerão constantes ao longo do estudo de sensibilidade e na terceira a quinta linha estão os três parâmetros a serem estudados. Percebe-se que o melhor valor de beta é 4, tanto considerando o valor mínimo de FO, quanto a média e que, para valores muito baixos de beta, o resultado piora muito.

Tabela 2 - Sensibilidade do parâmetro beta no TSP

Parâmetro	**Sens.1**	**Sens.2**	**Sens.3**	**Sens.4**	**Sens.5**	**Sens.6**
fer_0	0,1111	0,1111	0,1111	0,1111	0,1111	0,1111
fer_{atu}	1	1	1	1	1	1
fer_{evap}	0,06	0,06	0,06	0,06	0,06	0,06
alfa	1	1	1	1	1	1
beta	0,1	0,5	1	2	4	10
FO mínimo	1284,0	854,0	705,0	705,0	699,0	707,0
FO médio	1374,9	927,6	747,4	723,3	705,9	708,1

Interessante salientar que para valores ruins de beta, como na sensibilidade 3 em que beta = 1, o algoritmo necessita de mais iterações para convergir (não apenas as 150 utilizadas na sensibilidade) e, apesar da convergência ocorrer, é encontrado um ótimo local, como pode ser visto na Figura 26.

Figura 26 - Resultado do TSP para beta =1



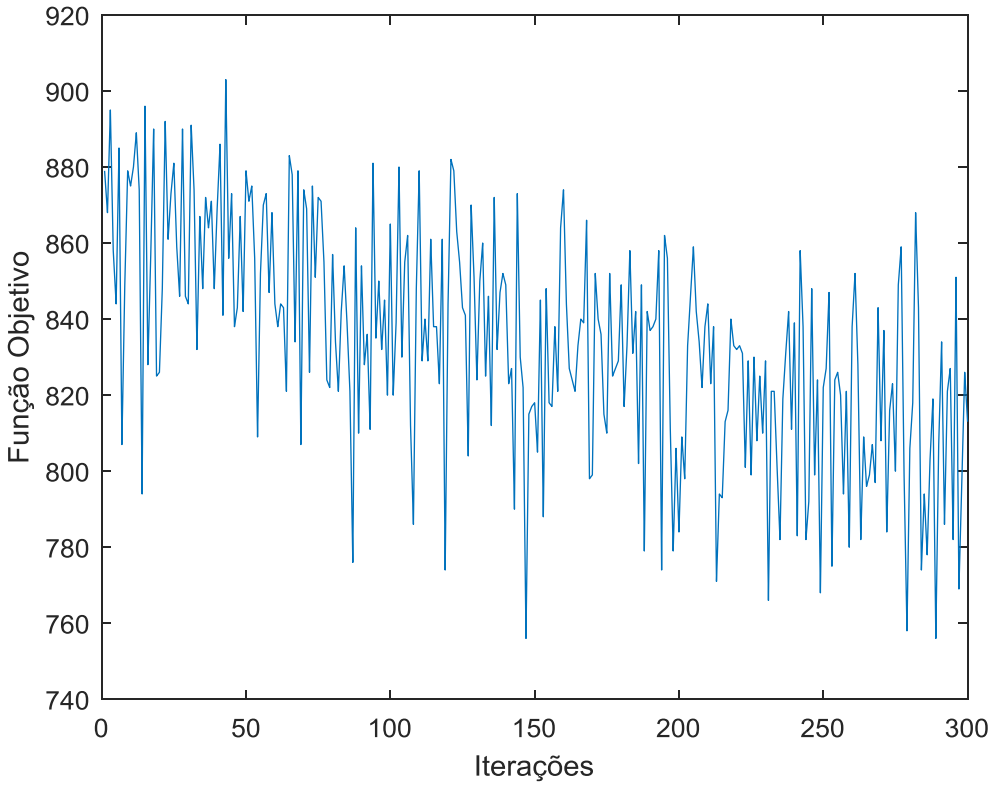
Na Tabela 3 abaixo, é mostrado resultado da sensibilidade do parâmetro alfa. O melhor resultado encontrado para alfa foi 1, o qual encontrou o ótimo global do problema.

Tabela 3 - Sensibilidade do parâmetro alfa no TSP

Parâmetro	**Sens.1**	**Sens.2**	**Sens.3**	**Sens.4**	**Sens.5**	**Sens.6**
fer0	0,1111	0,1111	0,1111	0,1111	0,1111	0,1111
fer_{atu}	1	1	1	1	1	1
fer_{evap}	0,06	0,06	0,06	0,06	0,06	0,06
alfa	0,1	0,5	1	2	5	10
beta	4	4	4	4	4	4
FO mínimo	765,0	707,0	699,0	703,0	703,0	703,0
FO médio	833,7	708,5	705,9	710,8	720,6	720,8

Para um valor inadequado de alfa, como na sensibilidade 1 da Tabela 3, nota-se, na Figura 27, que o sistema fica muito mais nervoso e não consegue convergir, mesmo com um aumento no número de iterações.

Figura 27 - Resultado do TSP para alfa = 0,1



O último parâmetro analisado é o de evaporação do feromônio e o resultado de sua sensibilidade é apresentado na Tabela 4. Os melhores resultados encontrados foram para os valores de 0,06 e de 0,07.

Tabela 4 - Sensibilidade do parâmetro de evaporação dos feromônios

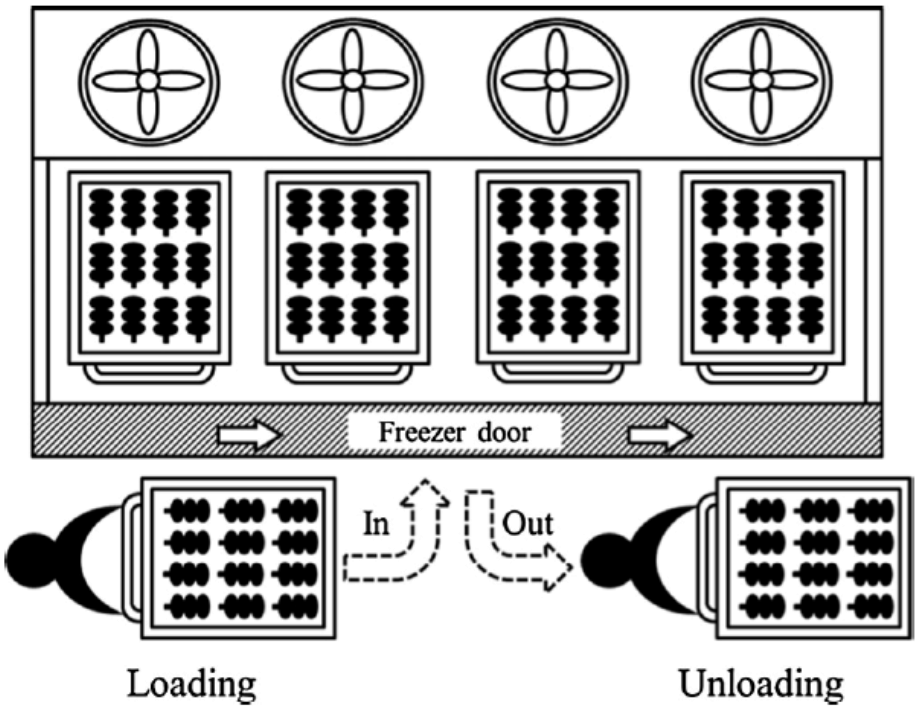
Parâmetro	Sens.1	Sens.2	Sens.3	Sens.4	Sens.5	Sens.6
fer_0	0,1111	0,1111	0,1111	0,1111	0,1111	0,1111
fer_{atu}	1	1	1	1	1	1
fer_{evap}	0,001	0,01	0,05	0,06	0,07	0,1
alfa	1	1	1	1	1	1
beta	4	4	4	4	4	4
FO mínimo	762,0	769,0	703,0	699,0	699,0	699,0
FO médio	838,9	816,4	705,8	705,4	705,4	707

6.3 PROBLEMA DE *SCHEDULING* ENVOLVENDO MÚLTIPLOS TRABALHOS EM UMA MÁQUINA

Problemas de *scheduling* são, comumente, classificados como NP-difícil, sendo, portanto, muito complicado e às vezes impossível, de serem resolvidos por métodos determinísticos. As meta-heurísticas são consideradas fortes ferramentas de *scheduling*, e, entre alas se destaca algoritmo de colônia de formigas. Nesse tópico será apresentada a solução de um problema de *scheduling* da indústria alimentícia e os resultados comparados com uma metodologia heurística. O algoritmo de colônia de formiga mostrou-se satisfatório em melhorar a resposta do problema, porém não foi capaz de atingir a solução ótima.

Chatavithee, Piewthongngam e Pathumnakul (2015), apresentam um estudo sobre o processo de congelamento por jateamento de ar da indústria de alimentos congelados, o qual processa vários produtos com taxas de processamento variáveis numa única máquina, como pode ser visto na Figura 28. O problema é resolvido utilizando um modelo de programação linear inteira mista e um algoritmo heurístico. O objetivo é minimizar os custos associados às atividades de *scheduling* na máquina, considerando: o horário de chegada e o tempo de processamento dos trabalhos não são uniformes e assumidos como determinísticos; os trabalhos podem ser carregados na máquina depois que eles chegam, se a capacidade da máquina estiver disponível, e eles podem ser descarregados da máquina após o processamento ser concluído, permanecer na máquina por mais tempo do que o tempo de processamento requerido é permitido. Existem três custos envolvidos nas decisões de *scheduling*: o custo da interrupção da máquina ao carregar um trabalho enquanto a máquina está operando em outros trabalhos, o custo de interrupção da máquina; o custo de um trabalho esperando para ser carregado na máquina após a chegada, a espera para carregar o custo; o custo do trabalho permanecendo na máquina por mais tempo do que o tempo de processamento necessário, chamado de espera para descarregar o custo.

Figura 28 - Esquema do scheduling envolvendo múltiplos trabalhos em uma máquina



Fonte: Chatavithee, Piewthongngam e Pathumnakul (2015)

O modelo deve processar 20 produtos na máquina de jateamento de ar, a qual pode processar 4 produtos de cada vez em um horizonte de 480 unidades de tempo. A hora de chegada de cada produto e os tempos necessários para o processamento de cada um deles são mostrados na Tabela 5. O custo de espera para carregar, o de espera para descarregar e o de interrupção da máquina são, respectivamente, 42 custos unitários/tempo unitário, 14 custos unitários/tempo unitário, 356 custos unitários/interrupção. Após um milhão de iterações, o estudo de Chatavithee, Piewthongngam e Pathumnakul (2015) alcançou o valor da função objetivo de 38500 e, após a aplicação do seu modelo heurístico, atingiu 35346.

Tabela 5 - Dados de hora de chegada e tempo de processamento de cada produto

Produtos:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Hora de chegada	0	4	13	37	38	41	51	69	82	94	99	107	110	120	124	135	143	155	170	188
Tempo de processamento	32	78	48	86	76	34	64	87	38	53	85	46	47	46	42	64	86	46	54	49

Nesse trabalho, o modelo foi implementado utilizando programação orientada a objeto em Matlab®, no qual cada elemento a ser processado no freezer corresponde a um objeto com propriedades de tempo de chegada, tempo de entrada, tempo a partir do qual pode deixar o freezer, tempo de saída do freezer e status, sendo o status modificado ao longo do *scheduling* através de um método do objeto. O objeto foi denominado *job* e suas propriedades e métodos podem ser visualizados na Figura 29.

Figura 29 - Código Matlab® do objeto *job*

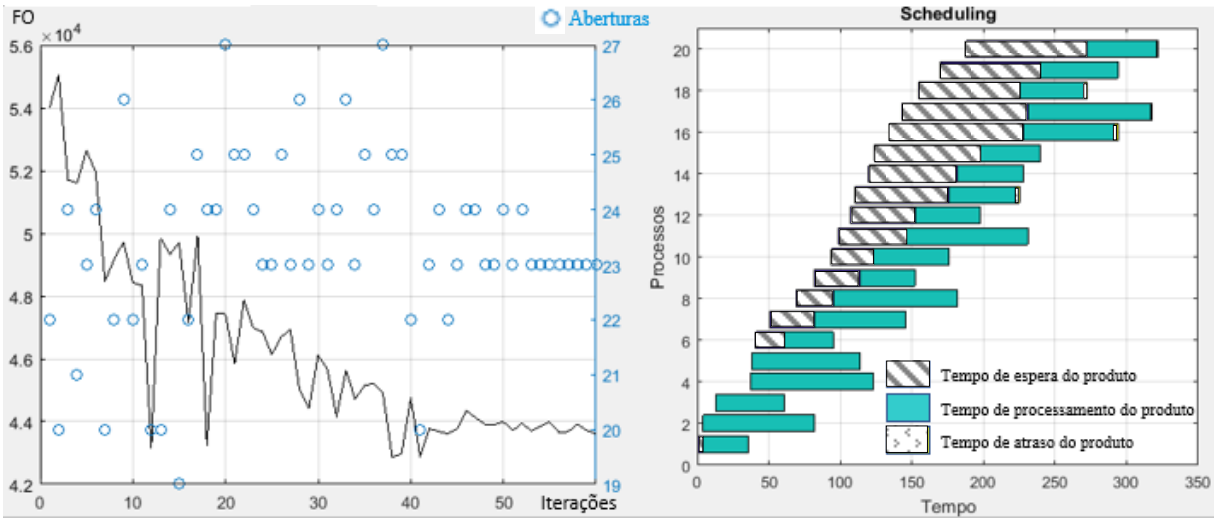
```
1   classdef job < handle
2       properties
3           aj  %Aj - dado de entrada
4           pj  %Pj - dado de entrada
5           in  %dado de saída
6           due %dado de saída
7           out %dado de saída
8           st = 0; %status
9               %st=0 => Indisponível
10              %st=1 => disponível p/ entrar no freezer
11              %st=2 => no freezer
12              %st=3 => disponível p/ sair no freezer
13              %st=4 => saiu do freezer
14      end
15      methods
16          function j = job(a, b, c, d, e, f)%constructor
17              if nargin == 6
18                  j.aj = a;
19                  j.pj = b;
20                  j.in = c;
21                  j.due = d;
22                  j.out = e;
23                  j.st = f;
24              end
25          end
26          function addst(j, i, newst) %Adiciona um status
27              g = j.st(1:i-1);
28              h(1,1:400) = newst; %Repete o novo st para os próximos tempos
29              j.st = [g h];
30          end
31      end
32  end
```

Foram escolhidas duas variáveis binárias para definir a solução do problema. A primeira variável corresponde a quais horários o freezer deverá abrir e a segunda a quais objetos devem entrar ou sair do freezer a cada abertura.

A escolha do fator heurístico é um dos pontos chave para o sucesso da implementação do ACO. Neste problema foi utilizado como parâmetro heurístico que o freezer pode abrir 20 vezes. Os valores dos parâmetros do ACO utilizados (obtidos por tentativa e erro) foram: quantidade inicial de feromônio 0,1; alfa = 1, beta = 0,5, evaporação do feromônio 0,01, 50 formigas e 80 iterações. A atualização dos feromônios é feita apenas para os melhores resultados (menores FO).

A solução do problema encontrada pela metodologia proposta é mostrada na Figura 30. A solução do algoritmo converge para um ótimo local do problema, pois este mostrou-se altamente dependente de uma boa parametrização. Ao trabalhar na calibração dos parâmetros, ficou claro que a taxa de evaporação baixa em relação à taxa de atualização de feromônio leva a estagnação em ótimos locais enquanto seu aumento acarreta a não convergência do modelo.

Figura 30 - Resultado da função objetivo e quantidade de aberturas do freezer para cada iteração e o *scheduling* final



7. METODOLOGIA

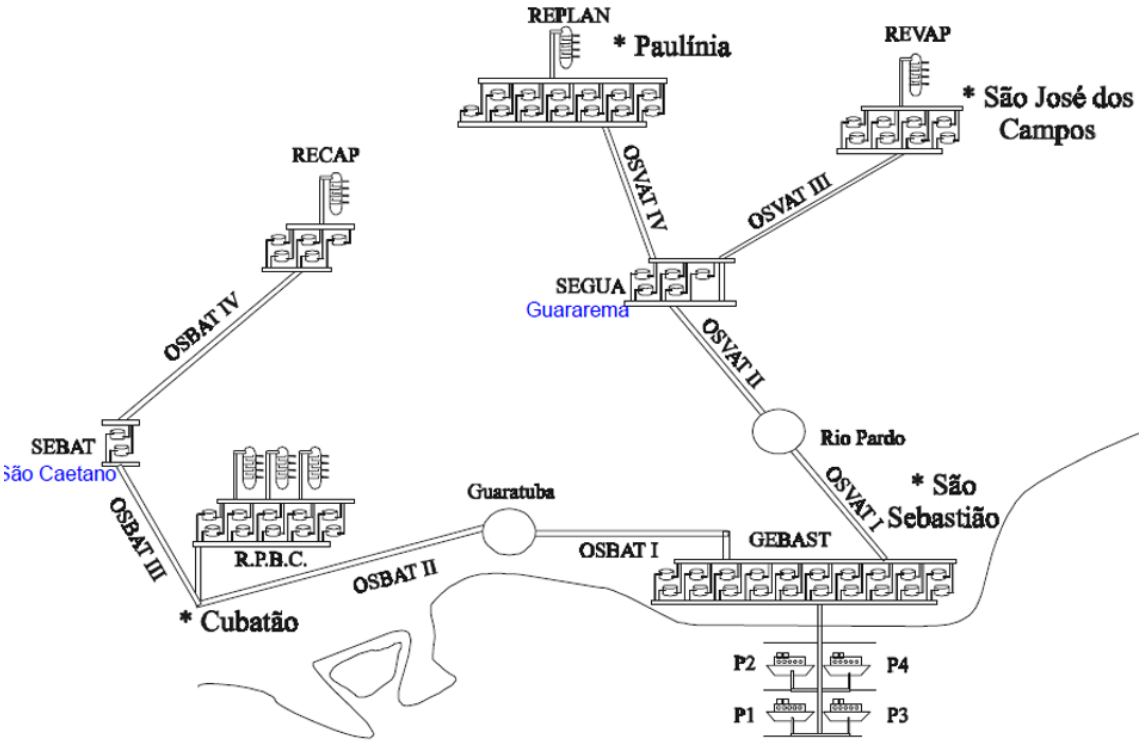
Já é reconhecido que a otimização tradicional determinística não é adequada para representar fielmente o comportamento das principais cadeias de suprimento reais, já que elas envolvem variáveis com um valor de incerteza expressivo e variação de propriedades com o tempo. Um modelo que envolve incertezas pode ser empregado como uma ferramenta auxiliar na avaliação de performance de uma unidade de refino frente a diversas mudanças circunstanciais (como mudança de mercado). Grande parte do gerenciamento da cadeia de suprimento de petróleo é, até então, baseada em modelos lineares simples e decisões heurísticas. A sua modelagem comumente consiste de subsistemas da cadeia de suprimentos em virtude da grande complexidade em trabalhar com o sistema completo (Neiro, 2003).

O método não determinístico escolhido para estudar sua aplicação no sistema de alocação de petróleo foi o método meta-heurístico de colônia de formigas. Para tal, é necessário modelar o processo de alocação de curto prazo de modo que esteja adequado à aplicação do ACO e que seu resultado possa ser avaliado diante de resultados determinísticos.

Em Neiro (2003), foi descrita uma cadeia de suprimento real da Petrobras, por ser a única empresa brasileira que atua em todos os segmentos da cadeia, desde a exploração do petróleo bruto, o refino e a distribuição de petróleo e derivados. O conjunto de refinarias do estado de São Paulo é responsável por cerca de 50% do volume total de matéria-prima processada no Brasil. Más (2001) descreveu um modelo do complexo da cadeia de suprimento de petróleo no qual será baseado esse estudo.

O modelo consiste de um sistema constituído de portos, dutos, terminais, subestações e refinarias, todos estes localizados em São Paulo. As refinarias envolvidas são a RECAP, REPLAN, REVAP e RPBC; o terminal é o de São Sebastião; as subestações são as de Cubatão (SEBAT) e de Guararema (SEGUA) em conjunto com estações de bombeamento do Rio Pardo e de Guaratuba; e os principais dutos são o OSVAT e o OSBAT. O sistema simplificado pode ser visualizado na Figura 31 adaptada de Más e Pinto (2003).

Figura 31 - Esquema da cadeia de suprimento de petróleo de São Paulo



Este esquema pode ser dividido em 4 partes: atracação do navio no porto, transferência do petróleo para os tanques dos terminais, transferência do petróleo dos tanques para as subestações e/ou estações de bombeamento de apoio e envio do petróleo às refinarias. Nesse trabalho serão apresentados 2 estudos de caso que representam, de modo simplificado, o terminal de São Sebastião e as refinarias REPLAN e REVAP, porém sem a modelagem da subestação entre elas.

Os petróleos são classificados em grupamentos de acordo com suas propriedades de misturas que são função de caraterísticas intrínsecas de cada óleo. O mérito de propriedades de misturas não será tratado em detalhes neste trabalho, mas suas consequências são levadas em conta em forma de premissas e fatores de decisões do modelo. Cada petróleo pode pertencer a mais de um grupamento, isso se deve ao fato de um mesmo petróleo possuir diversas características que torna sua mistura favorável com uns e desfavorável com outros. Por exemplo, alguns petróleos podem formar depósitos quando misturados com um petróleo não compatível, enquanto alguns grupamentos têm maior proximidade de propriedades primordiais para o refino, como grau API e teor de enxofre. Assim, os tanques do terminal só podem

receber os óleos pertencentes a algumas classes, sendo que um mesmo petróleo pode pertencer a várias classes. Cada navio petroleiro possui uma capacidade de transporte de matéria-prima e é considerado que essa capacidade é sempre completamente utilizada e o bombeamento do produto para os tanques dos portos é integral.

Um terminal é composto por um conjunto de píeres; o terminal de São Sebastião modelado é composto de quatro píeres que comportam certas classes de navio baseado na capacidade no navio, o que está diretamente ligado ao seu tamanho. Os píeres são de uso exclusivo de recebimento de navios petroleiros, ou seja, a movimentação de outros produtos ou derivados não são contemplados no modelo.

Todos os tanques, seja ele encontrado no terminal, em subestações ou em refinarias, são tanques de teto flutuante que necessitam de um volume mínimo (lastro) e possuem um limite máximo de capacidade. Os dutos possuem diâmetros pré-determinados, vazão máxima definida na fase de projeto, e podem receber qualquer tipo de petróleo, dessa maneira, não há restrições físicas para o bombeamento de qualquer produto. O mesmo oleoduto pode transportar simultaneamente diferentes tipos de petróleo ou derivados, sendo este tipo de transporte realizado em lotes e cada um dos lotes consistem de um petróleo ou derivado diferente.

O padrão de demanda da refinaria é considerado fixo, ou seja, para o horizonte de tempo modelado, a refinaria consome uma carga de petróleo fresco fixa, o que é perfeitamente aceitável se tratando de uma programação de curto prazo. A alimentação da unidade de destilação é realizada de maneira contínua, contudo os tanques de armazenamento de petróleo bruto nas refinarias têm capacidade para nutri-la por alguns dias. Assim não é necessário ser bombeado continuamente óleo para a refinaria, mesmo que não haja interrupção de sua operação.

7.1 MODELO GERAL

Para o estudo do desempenho do método de otimização por colônia de formigas no *scheduling* de alocação de petróleo, foi desenvolvido um modelo utilizando os dados de entrada e as considerações do problema modelado no trabalho de Más (2001)

envolvendo o modelo do porto. No terminal SEBAT (São Sebastião), é considerada a existência de 20 tipos de petróleo a ser recebido, então este terminal possui 20 tanques de armazenamento. Já as refinarias podem receber cerca de 10 tipos diferentes de petróleo. Analogamente ao petróleo, os derivados também são armazenados individualmente de acordo com suas especificações. No trabalho do Más (2001), o problema foi resolvido utilizando MILP e separando a modelagem em partes menores e resolvendo-as separadamente. O método da decomposição resolve problemas MILP dividindo-os em subproblemas e resolvendo cada um por vez (Pinto, 2000). Assim, a cadeia de suprimento de petróleo pode ser repartida para facilitar a resolução, podendo separar um modelo para o porto, outro para as subestações e outro para as refinarias, por exemplo.

Um navio petroleiro possui mais de um tanque em seu interior e possui a capacidade de armazenar mais de um tipo de petróleo, segregado ou não. Essa característica é adicionada ao modelo, de forma a gerar mais restrições quanto ao grupamento que pode ir para cada tanque do terminal e vazão total empregada.

O píer é modelado com uma restrição de capacidade, assim alguns deles podem receber qualquer tipo de navios, enquanto outros têm um limite máximo de capacidade do navio a atracar. Esse tipo de restrição é bem comum nos sistemas de terminais reais, que possuem, inclusive, restrições diferentes para atracação e desatracação em virtude da capacidade de bombeamento dos terminais.

Segundo Dorigo e Stützle (2004), tentativa e erro guiou todos os primeiros pesquisadores e ainda orienta a maioria das pesquisas em andamento no estudo da aplicabilidade do ACO aos diversos problemas de otimização.

7.1.1 DEFINIÇÃO DAS VARIÁVEIS DO MODELO

A complexidade do assunto da alocação de petróleo não é justificada apenas pela grande quantidade de variáveis envolvidas, mas sim pelo alto grau de incerteza embutida em muitas delas. Assim, duas medidas devem ser tomadas para a elaboração de uma boa metodologia de solução para este tipo de problema: diminuir a quantidade de variáveis àquelas estritamente necessárias; modelar as principais incertezas associadas.

Neste estudo, existem 5 variáveis decisórias com as dimensões apresentadas na Tabela 6. Como o horizonte de tempo do estudo de caso 1 é de 96h e o intervalo de tempo de um décimo de hora, o número de intervalos de tempo é de 960.

Tabela 6 - Variáveis decisórias

Variável	Descrição	Dimensão
np	Associa cada navio a um píer	navio (4)
ct	Associa cada petróleo a um tanque no tempo	petróleo x tempo (4x960)
Ect	Quantidade de eventos de transferência do navio para o tanque	píer (2)
Ect_point	Hora em que se inicia cada evento Ect	píer x Ect
htq	Hora em que o tanque começa o *settling*	tempo / intervalo de tempo (96/0,1)

A variável “np” é uma variável vetorial de dimensão “n” navios por 1, na qual o número indica que aquele navio irá atracar naquele píer. No exemplo da Figura 32 para 3 navios e 2 píeres, indica que o navio 1 irá atracar no píer 1 e o navio 2 e 3 irão atracar no píer 2.

Figura 32 - Exemplo de resultado para a variável "np"

$$\begin{array}{cc} & \text{Píeres} \\ \text{Navios} & \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix} \end{array}$$

A variável “ct” associa cada petróleo “c” a um tanque “t”, respeitando sempre a restrição de que um petróleo só pode ir para o tanque esse ele pertencer a mesma classe que o tanque pertence. Essa variável é uma matriz em que as linhas são os petróleos e o conteúdo (o valor da célula) é número do tanque. Após a definição das variáveis de evento (“Ect” e “Ect_point”) a matriz passa a ter a dimensão petróleo por tempo, em que cada coluna é um tempo e os valores da matriz são números inteiros que correspondem ao número do tanque.

A variável “Ect” surge como uma ferramenta de aproximação do modelo da realidade, fazendo com que não seja possível a mudança de alinhamento entre navios e tanques em cada intervalo discreto de tempo. Já a transferência entre tanque e oleoduto é feita através de lotes, que tem o tamanho definido pelo volume disponível no tanque. Esse volume nem sempre é o máximo possível (que seria a diferença entre o volume máximo e volume mínimo do tanque) graças à variável htq, que decide os momentos em que o tanque vai começar a sedimentar, independentemente do volume de petróleo contido nele. Após ser definida a quantidade de eventos (variável “Ect”), a variável Ect_point é calculada, decidindo em quais tempos os eventos irão ocorrer em cada píer; essa variável independe de “np”.

Os principais dados de entrada (constantes) são: grupamentos dos petróleos, o estado inicial (como volume inicial nos tanques, navios, oleoduto e refinaria), a hora de chegada dos navios, os custos do petróleo, o valor do petróleo no terminal e na refinaria, a velocidade máxima de cada petróleo no oleoduto e o horizonte de tempo.

Dados de entrada flexíveis: intervalos de tempo dentro do horizonte; quantidades mínimas e máxima de eventos e os parâmetros do ACO.

As notações do modelo são apresentadas na Tabela 7. Foi utilizada a variável "c" para representar o petróleo cru (do inglês, *crude*) para não confundir com a variável "p" associada a probabilidade.

Tabela 7 - Notações do modelo

Notação	Descrição
c	petróleo
cl	classe de petróleo
n	navio
p	píer
tq	tanque
o	oleoduto
r	refinaria
t	intervalo de tempo

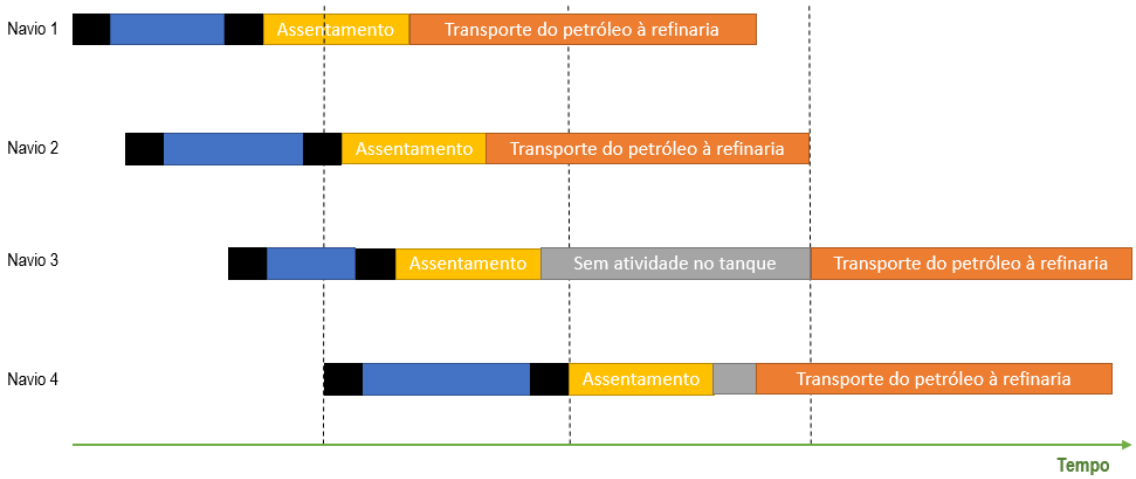
7.1.2 REPRESENTAÇÃO DO DOMÍNIO DO TEMPO

Uma representação discreta do domínio do tempo não é adequada quando os intervalos de tempos das atividades se diferem muito, pois a unidade de tempo necessária seria muito pequena, podendo inviabilizar a resolução do problema ou torná-la demasiadamente longa (Liza, 2005). Uma abordagem comum para a representação do domínio no tempo é a definição de intervalos de tempo para cada processo e quando eles são definidos globalmente, são chamadas de eventos (Moro, 2000). Cada evento pode possuir um intervalo fixo de tempo ou ser delimitado pelo término de uma tarefa específica.

A escolha do tempo discreto diminui o grau de liberdade do problema. Apesar da abordagem de tempo discreta ser utilizada, para impedir que a cada pequeno intervalo de tempo, novos alinhamentos sejam feitos, o que tornaria o modelo mais afastado da realidade, foram criados eventos englobando vários intervalos discretos de tempo. Os intervalos de tempo, ou do inglês *time slots*, são definidos localmente, ou seja, cada evento possui seus respectivos intervalos, os quais não possuem valores pré-definidos e sim tem sua dimensão como uma das variáveis do problema a ser otimizado.

Na Figura 33, em preto, são representados os intervalos de tempo de atracamento e desatracamento dos navios, em azul, o tempo em que o navio descarrega e simultaneamente preenche o respectivo tanque do terminal, em amarelo o tempo de assentamento do petróleo no tanque, em laranja, o intervalo de tempo do transporte da matéria-prima à refinaria através do oleoduto e, em cinza, o tempo sem atividade, ou seja, sem nenhuma tarefa sendo realizada.

Figura 33 - Diagrama de Gantt esquemático



Não são considerados tempos de transição entre uma atividade e outra, ou seja, uma atividade é imediatamente seguida da próxima. Apenas alguns eventos de transição conhecidamente mais demorados, como atracação e desatracação de navios são considerados.

7.1.3 CARACTERIZAÇÃO DO PETRÓLEO

O tipo de petróleo é uma variável discreta e, portanto, pode ser trabalhado numa análise combinatorial, não serão aqui consideradas as propriedades de mistura. Assim cada petróleo é associado ao grupamento baseado em suas propriedades, sendo que cada petróleo pode fazer parta de vários grupamentos diferentes, no estudo de caso, mas só são permitidos petróleos de um mesmo grupamento se misturarem.

7.1.4 NAVIOS

Os navios petroleiros considerados nesse problema, possuem um só petróleo armazenado em seu interior. De modo a compatibilizar os resultados dessa modelagem com o apresentado por Más (2001), o navio que possuiria dois tipos de petróleo foi repartido em dois navios, cada qual com um dos petróleos, e para que ele se comportasse como se fosse apenas um navio, foi necessário zerar o tempo de desatracação do primeiro navio e de atracação do segundo, além de adicionar uma restrição ao problema de que esses dois navios devem sempre utilizar o

mesmo píer. A única desvantagem desta abordagem é a que não haverá possibilidade de bombeamento desses petróleos para os tanques do terminal de forma intercalada, diminuindo levemente o grau de liberdade do problema.

7.1.5 ESCOLHA DO PÍER PARA ATRACAÇÃO

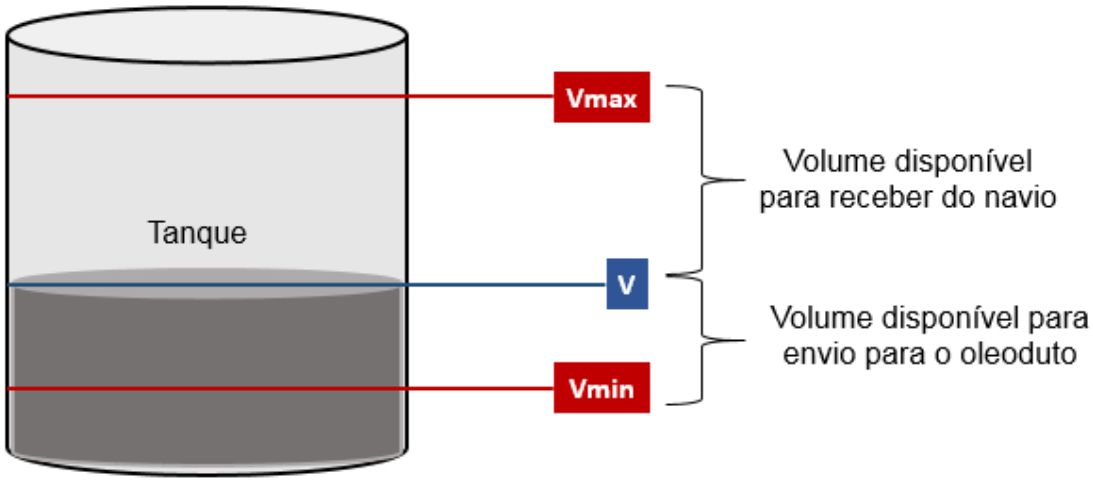
A disciplina de atendimento escolhida para a escolha do píer para atracação foi a FIFO (*First in First Out*), ou seja, o primeiro navio a chegar será o primeiro a atracar, independentemente do seu conteúdo ou capacidade. Assim, o navio que chega primeiro escolhe um dos píeres, e os próximos navios atracam no que estiver disponível.

Os píeres podem possuir custo de utilização diferentes, o qual aumenta com a capacidade máxima do navio que pode atracar nele, ou seja, os píeres capazes de atracar navios maiores (com maior capacidade de armazenamento de produto) são mais caros do que os que só são capazes de atracar navios de pequeno de porte. Pode existir uma restrição na qual o tamanho do navio implica em qual píer está apto a o receber.

7.1.6 ESCOLHA DO TANQUE

Cada um dos tanques pode receber alguns tipos de petróleo baseado no grupamento a que ele pertence, ou seja, cada tanque é associado a apenas um grupamento (ou classe) de petróleo e todos os petróleos que pertencem a esse predeterminado grupamento são passíveis de irem para esse tanque. Após a atracação no navio, o petróleo é enviado para um ou mais tanques do terminal respeitando sempre se o petróleo do navio é da classe correspondente do tanque. Cada tanque possui um volume mínimo e volume máximo possível de ocupação, sendo o volume ao longo do tempo sempre entre esses dois extremos, conforme a Figura 34.

Figura 34 - Volumes possíveis no tanque



A mudança de status do tanque para "sedimentando" ocorre sempre que ele atingir sua capacidade máxima ou em outras situações explicitadas a seguir. O tempo de sedimentação de 24h pode ser maior se não houver oleoduto disponível para mandar a refinaria. O tanque para de receber petróleo do navio em três situações. A primeira é se o navio esvazia, ou seja, o petróleo contido nele é todo transferido para o(s) tanque(s) correspondente(s). A segunda possibilidade é de quando o tanque atinge sua capacidade máxima. A última situação possível é a de quando a variável ”htq”, que define os tempos no qual o tanque irá iniciar a sedimentação está ativa no momento; quando isto ocorre, mesmo que o tanque não esteja cheio, ele para de receber o petróleo e começa seu período de sedimentação.

O tanque para de enviar para o oleoduto em três situações. A primeira delas, e a mais intuitiva, é quando não houver mais produto disponível a ser enviado, ou seja, no momento em que o tanque atinge seu volume mínimo, o que é um dado de entrada do problema. A segunda possibilidade é de a refinaria não poder mais receber o petróleo, o que acontece quando ela atinge sua capacidade máxima. A terceira situação possível é a variável “ct” indicar mudança de envio do petróleo para outro taque, o que ocorre nas horas “Ect_point” determinadas.

7.1.7 PASSAGEM PARA O OLEODUTO E REFINARIA DESTINO

Cada tanque, após encher até um determinado volume (que não, necessariamente, é sua capacidade máxima, necessita aguardar um período de repouso de no mínimo

24h para a separação de sedimentos. Passado este período, o tanque está pronto para descarregar no oleoduto para encaminhar o produto à refinaria.

O oleoduto é diretamente interligado a refinaria, sem possibilidade de outros alinhamentos. A vazão no duto é definida pela classe de petróleo que se encontra em seu interior e que limita a vazão máxima do oleoduto. As vazões permitidas no problema são ou vazão máxima ou vazão zero (duto parado). O duto é considerado sempre completamente cheio e só recebe produto concomitantemente com a ação de enviar produto à refinaria.

Cada refinaria agrega um valor diferente para cada tipo de petróleo baseado no rendimento e valor comercial dos derivados produzidos pela mesma. Como o estudo da operação das refinarias não é foco de estudo deste trabalho, o perfil de produção é aproximado como função exclusiva do tipo de petróleo empregado como matéria-prima, e o valor dos derivados produzido como também invariáveis. Assim, é gerado um fator multiplicativo ao valor do petróleo que representa o valor agregado pela refinaria a ele. Este número será utilizado como um “fator de atratividade” da refinaria e é parte da receita da função objetivo. É considerado que a refinaria possui consumo constante de petróleo ao longo do horizonte.

7.1.8 ESTRUTURA DO MODELO DE OTIMIZAÇÃO

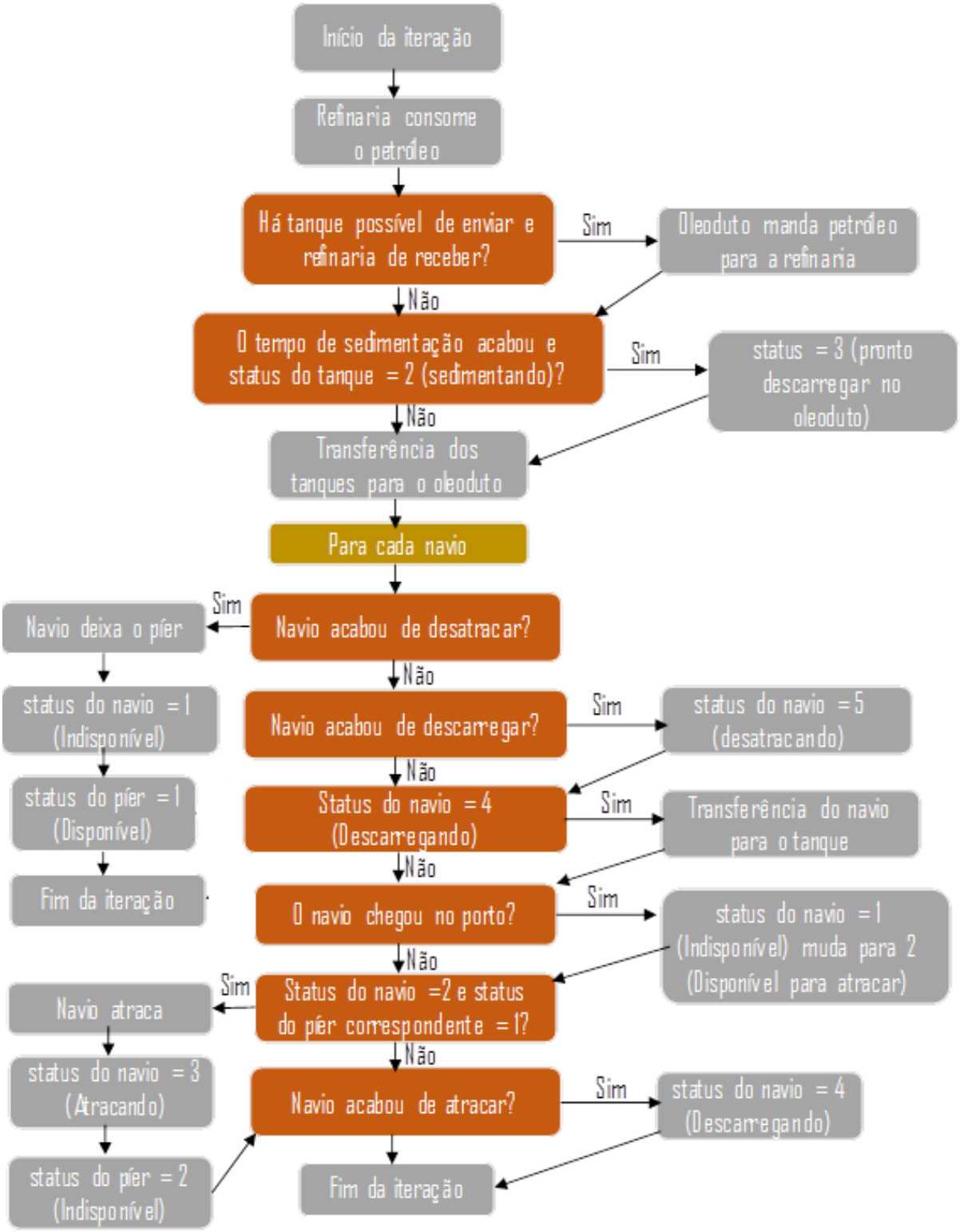
O modelo de otimização é composto de um programa principal, que chama o programa de *scheduling* na quantidade equivalente ao produto entre o número total de iterações e da quantidade de formigas utilizadas na otimização. Além desses dois programas fundamentais, existem também uma gama de programas auxiliares, como o que carrega os dados do problema e a roleta, que calcula os resultados das variáveis de acordo com as probabilidades encontradas.

Como a modelagem no Matlab® é sequencial, o corpo da modelagem dispõe de estruturas condicionais e loops para auxiliar a elaboração do algoritmo. Para visualizar o código completo, vide o “Apêndice D” deste trabalho.

A sequência de tarefas é fixa, caracterizando um *flowshop*, ou seja, não é possível voltar a tarefas anteriores pois existe apenas um sentido possível: navio >> píer >> tanque do terminal >> oleoduto >> refinaria. O algoritmo de *scheduling,* elaborado

nessa dissertação, é esquematizado na Figura 35, sendo a iteração repetida a cada intervalo de tempo discreto. Esse algoritmo foi concebido visando a implementação da otimização no Matlab® utilizando programação orientada a objeto.

Figura 35 - Fluxograma do algoritmo de *scheduling*



7.1.9 FUNÇÃO OBJETIVO

A função objetivo (FO) é a maximização do lucro da operação, a qual possui a parcela positiva do valor do petróleo na refinaria, do valor do petróleo no terminal e diversas parcelas negativas relacionadas ao custo do óleo, à utilização dos píeres e tanques, entre outros custos. Ela pode ser sintetizada na Equação 8 a seguir:

$$FO = \sum \mathrm{Re}\,ceitas - \sum Custos \tag{8}$$

A função objetivo só é calculada para os casos viáveis, ou seja, que atende todas as restrições do problema, diminuindo o esforço computacional.

As receitas são o valor do petróleo na refinaria e seu valor no terminal. Já os custos do modelo são segregados em quatro partes:

- Custo do petróleo que está no navio no início do horizonte de tempo
- Custo de sobrestadia dos navios, quando houver. Esse custo é devido quando certo navio ultrapassa o tempo máximo que ele pode estar no porto sem gerar um custo de sobrestadia, o que, neste problema, é o tempo de dois dias. Atentar para que esse tempo se refere ao navio no porto e não no píer, ou seja, começa a contar desde que o navio se encontra disponível para atracar independente se a atracação já ocorreu.
- Custo de utilização do píer calculado em relação ao tempo de permanência do navio ocupando o píer, ou seja, desde o início da sua atracação até o fim da desatracação. Esse custo varia para cada um dos píeres, sendo que aquele com maio flexibilidade de porte de navio a ser atracado tende a ser mais oneroso.
- Custo da interface dos petróleos no duto é calculado sempre que dois petróleos de classes diferentes são enviados ao duto, entrando em contato e formando um volume de interface entre eles.

7.2 PRINCIPAIS ASPECTOS DA IMPLEMENTAÇÃO NO MATLAB®

A implementação do programa de otimização de *scheduling* no Matlab® foi feita utilizando linguagem orientada a objeto. Segundo The MathWorks (2001), o termo programação orientada a objetos descreve uma abordagem para escrever

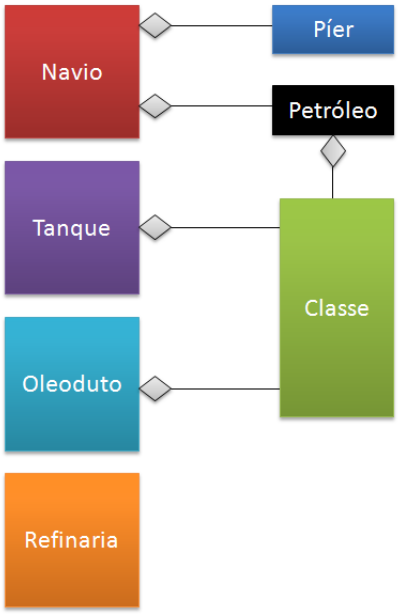
programas que enfatizam o uso de classes e objetos, as quais permitem adicionar novos tipos de dados e operações ao Matlab®. A classe de uma variável descreve a estrutura da variável e indica os tipos de operações e funções que podem ser aplicadas a ela, já um objeto é uma instância de uma classe específica. Assim, as classes podem ser consideradas como novos tipos de dados com comportamentos específicos definidos para a classe ou como novos itens utilizados como entidades únicas. As classes podem ser adicionadas ao seu ambiente Matlab® especificando uma estrutura Matlab® que fornece armazenamento de dados para o objeto e criando um diretório de classe contendo arquivos M (m.file, extensão do Matlab®) que operam no objeto. Esses arquivos M contêm os métodos (ou funções) para a classe.

A principal vantagem da programação orientada a objeto é aumentar significativamente a reutilização de código e facilitar a manutenção e ampliação de seus programas se forem utilizadas classes bem projetadas (The MathWorks, 2001).

O recurso interativo de classes mais utilizado neste trabalho foi o de agregação, no qual se pode criar classes de modo que um objeto contém outros objetos. Isso é apropriado quando um tipo de objeto faz parte de outro tipo de objeto. Uma classe criada pelo usuário herda da classe de estrutura (*structure*) do Matlab® (The MathWorks, 2001). Assim, trabalhar com objetos diminui significativamente a complexidade do algoritmo quando este envolve um número muito grande de variáveis que estão correlacionadas. Assim, utilizando a programação orientada a objetos diminui-se a utilizações de variáveis em demasia e a utilização de matrizes multidimensionais, que são, em geral, difíceis de acompanhar ao longo do programa. Então, apesar de existirem muitas variáveis, as operações que precisaram ser feitas com elas são comuns, podendo ser agrupadas e simplificadas para a celeridade do programa.

Para o estudo de caso de alocação de petróleo, foi implementada, no Matlab®, uma modelagem de *scheduling* utilizando orientação objeto de modo a assemelhar a estrutura escolhida à realidade física do problema, facilitando a compreensão do código elaborado. Foram utilizados, no total, sete classes (tipos diferentes de objetos), são eles: petróleo, classe (ou grupamento) de petróleo, navio, píer, tanque, oleoduto e refinaria. Eles são relacionados conforme a Figura 36 abaixo:

Figura 36 - Diagrama esquemático da estrutura orientada a objeto



As setas indicam o sentido da agregação entre os objetos, ou seja, qual objeto contém o outro, como, por exemplo o objeto navio contém os objetos píer e petróleo, sendo que o objeto petróleo contém o objeto classe, que por sua vez é também contido pelos objetos tanque e oleoduto. O objeto refinaria é um consumidor de petróleo ligado exclusivamente ao oleoduto no programa, mas seus objetos não são relacionados.

As propriedades de cada objeto são apresentadas na Tabela 8 abaixo:

Tabela 8 - Propriedades de cada tipo de objeto

Objeto	Propriedade	Definição
pet	CL	Grupamento do petróleo
	Custo	Custo do petróleo
	classe	Grupamento do petróleo
classe	Valor_tq	Valor da classe no tanque
	Valor_ref	Valor da classe na refinaria
	Interf	Valor da interface com outra classe
	Qmax	Máxima vazão do grupamento cl no duto
pier	Custo	Custo de utilização do píer por tempo
	st	Status
navio	H0	Hora de chegada do navio em horas
	Hsob	Hora máxima p/ não caracterizar sobrestadia
	Hent	Hora do início da atracação

	Hftq	Hora do fim do bombeamento para o tanque
	Hf	Hora da saída do navio do píer
	Csob	Custo de sobrestadia por tempo
	Qmin	Vazão mínima de descarregamento do navio n
	Qmax	Vazão máxima de descarregamento do navio n
	pet	Petróleo contido no navio
	V	Volume de petróleo no navio em mil m^3
	V0	Volume inicial de petróleo no navio em mil m^3
	saveV	Guarda os valores de V para controle e elaboração de gráficos
	pier	Píer associado ao navio
	Tent	Intervalo de tempo necessário para a atracação no navio no píer
	Tsai	Intervalo de tempo necessário para a desatracação no navio no píer
	st	Status
tq	Vmin	Capacidade mínima de armazenamento do tanque
	Vmax	Capacidade máxima de armazenamento do tanque
	V	Volume de petróleo no tanque
	V0	Volume inicial de petróleo no tanque
	saveV	Guarda os valores de V para controle e elaboração de gráficos
	Cl	Grupamento do petróleo
	Qmax	Vazão máxima
	H_switch	Impede que o tanque receba do navio ao mesmo tempo que mande para o oleoduto
	R	Preço por unidade de volume do grupamento de petróleo no tanque do terminal
	st	Status
	pet	Petróleo contido no tanque
	Hset	Hora em que começou o *settling*
oleod	V	Vetor de volumes na sequencia que entra no oleoduto
	Cl	Vetor de classes correspondentes aos volumes na sequencia que entra no oleoduto
	st	Status
	Qmax	Máxima vazão no oleoduto
	Cinterf	Custo da interface entre as classes
ref	Vmin	Capacidade mínima de armazenamento na refinaria
	Vmax	Capacidade máxima de armazenamento na refinaria
	V	Volume de petróleo na refinaria
	saveV	Guarda os valores de V para controle e elaboração de gráficos
	Cons	Consume de petróleo na refinaria em mil m3/h
	Rpet	receita com petróleo recebido

Nos próximos tópicos serão descritas cada uma dessas variáveis e a metodologia de formulação aplicada. Para exemplificar a modelagem de um objeto, a Figura 37, apresenta o objeto oleoduto com suas propriedades e métodos, para visualizar o código de outros 5 objetos, vide Apêndice D.

FIGURA 37 - CÓDIGO MATLAB® DO OBJETO OLEODUTO

```
classdef oleod < handle
    %OLEODUTO
    %   Detailed explanation goes here

    properties
        V %vetor de volumes na sequencia que está no oleoduto (não é a
                        %sequencia em que entra no oleoduto, é o oposto)
        Cl %vetor de classes correspondentes aos volumes na sequencia que
                        %entra no oleoduto
        Qmax
        Cinterf = 0; %custo da interface entre as classes
        st = 1 %status do oleoduto:
            %1-recebendo/enviando
            %2-parado
    end

    methods
        function o = oleod(a, b, c)
            if nargin == 3
                o.V = a;
                o.Cl = b;
                o.st = c;
            end
        end
```

```
        function addV(o, newV, newClasse)
            %Carregando o Oleoduto
            if newClasse == o.Cl(1) %se a classe a ser recebida é a mesma
                                    %que já se está recebendo
                o.V(1) = o.V(1) + newV;
            else
                for i=1:length([newClasse.Interf]) %Custo de interface
                    for j=1:length([newClasse.Interf])
                        if newClasse.Interf(i) == o.Cl(1).Interf(j)
                            Cost_interf = newClasse.Interf(i);
                        end
                    end
                end
                o.Cinterf = o.Cinterf + Cost_interf;
                o.Cl = [newClasse o.Cl];
                o.V = [newV o.V ];
            end

            %Descarregando o Oleoduto
            if o.V(end) > newV
                o.V(end) = o.V(end) - newV;
            else
                Vres = o.V(end);%Volume residual da última classe no
                                %oleoduto
                o.V = [o.V(1:(length([o.V])-1))]; %Retira o último elementc
                                                  %do vetor de volumes
                o.Cl = [o.Cl(1:(length([o.Cl])-1))]; %Retira o último
                                                 %elemento do vetor de classes
                o.V(end) = o.V(end) - (newV - Vres);
            end
        end
    end
end
```

A estrutura do programa no Matlab® pode ser visualizada na Figura 38, a qual facilita o entendimento do código do programa completo mostrado no Apêndice D. O código é composto de um programa principal responsável pelo carregamento dos dados do problema, que se encontram em documentos de texto (extensão .txt). Depois do carregamento dos dados, são inicializados os parâmetros do problema de *scheduling* e do ACO. Os parâmetros de *scheduling* são: o horizonte de tempo, o passo (mínimos intervalos de tempo discretos), as quantidades mínimas e máximas de eventos possíveis para cada navio e a tolerância do balanço material, a qual garante que o processo de *scheduling* foi feito respeitando o balanço volumétrico do problema. Os parâmetros do ACO são: quantidade de iterações, quantidade de formigas por iteração, quantidade de feromônio inicial, quantidade de feromônio evaporado por iteração, constante de atualização do feromônio (que é multiplicada por um valor de função objetivo) e o valor das heurísticas de cada variável. Após a

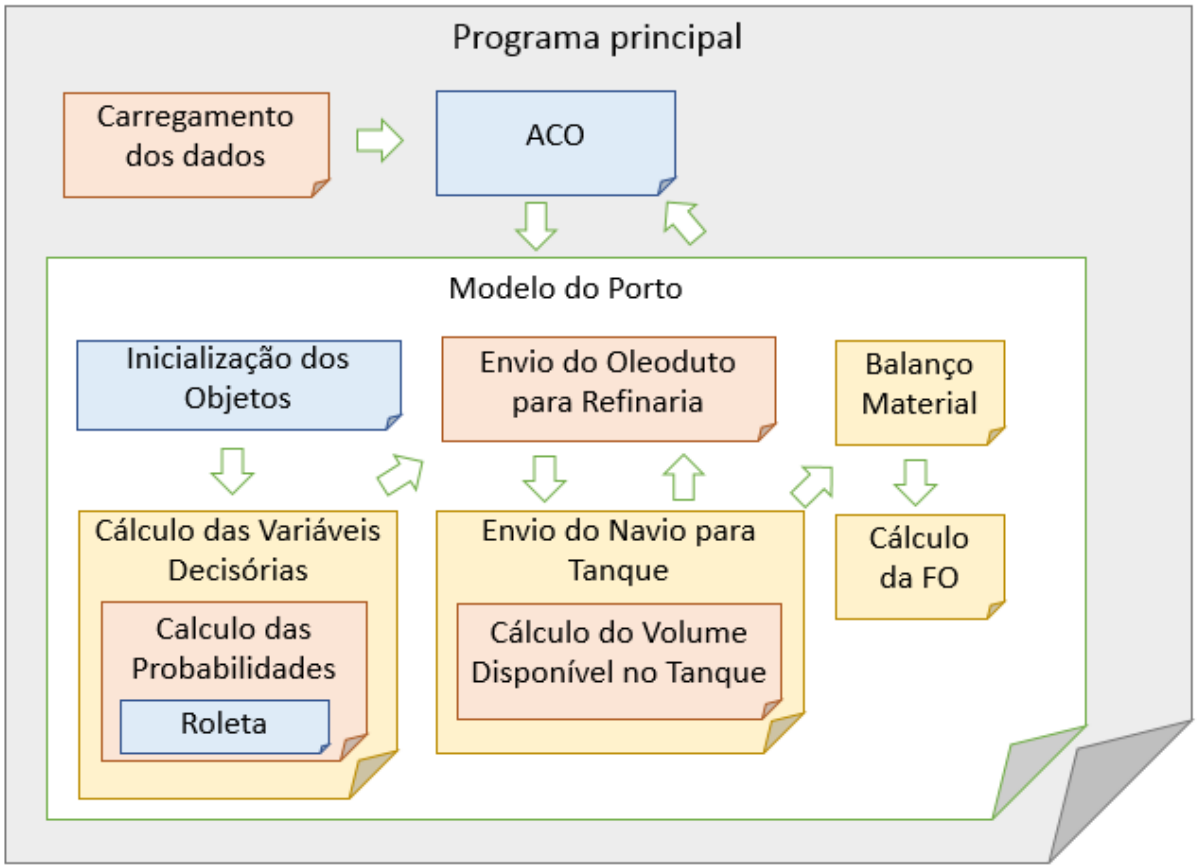
declaração de todos esses parâmetros, é possível iniciar o modelo de *scheduling* para cada formiga da primeira iteração as quais executam o programa do modelo do porto de modo individual e consecutivo.

Entrando no modelo do porto, a fase inicial é a inicialização dos objetos. A inicialização dos objetos consiste na associação dos dados carregados anteriormente a suas correspondentes instâncias dos objetos. Com todos os objetos inicializados, o programa que calcula as variáveis decisórias é executado. Nesse programa, calcula-se, para cada variável, qual a probabilidade de resultado baseado nos parâmetros do ACO, utilizando a Equação 1 (explicada no tópico 5.2) e, a partir dessa matriz de probabilidades, executa-se o programa roleta, o qual escolhe o valor que de fato essa variável irá assumir baseado nas probabilidades recém determinadas. Com o valor das variáveis decisórias em mãos, é possível iniciar o processo de *scheduling*.

O estado inicial do problema engloba o píer vazio com os navios a chegar, os tanques com volumes diversos (entre os limites mínimos e máximos), o oleoduto cheio de petróleo fluindo para a refinaria que possui volume de estoque inicial superior ao mínimo. Desse estado inicial, para dar encaminhamento ao *scheduling*, inicia-se da “ponta” do problema, ou seja, primeiro o programa consome o petróleo na refinaria, o oleoduto envia mais petróleo e depois são modeladas as ações dos tanques e navios, sempre conferindo a sinergia entre esses processos. Após o fim do horizonte de *scheduling*, é feito o balanço material, e se averiguado que ocorreu tudo justo, a função objetivo é calculada e os resultados são enviados para o programa inicial, salvos e se dá início ao modelo do porto para a próxima formiga.

Assim que todas as formigas terminarem a execução do modelo do porto, ocorre a evaporação dos feromônios seguida da atualização dos mesmos baseado nos melhores resultados obtidos.

Figura 38 - Estrutura do código Matlab® para o estudo de caso do modelo do porto



7.3 RESTRIÇÕES DO MODELO

Parte das restrições foram aplicadas em forma de matrizes binárias, as quais limitam as combinações possíveis entre as variáveis de modo que apenas as soluções viáveis são calculadas. Assim, muitas restrições do problema foram modeladas como impossibilidades, ou seja, não há como ser uma das soluções do problema; um exemplo disso é a restrição de quais tipos de petróleo podem ir para cada tanque do terminal, de acordo com a classe a que pertencem.

Uma restrição comum neste tipo de problema é a restrição de capacidade. Restrição de capacidade é aquela relativa aos limites mínimos e máximos de uma dada variável, o que, neste caso será invariável com o tempo. Este tipo de restrição é essencial para a modelagem adequada de navios, tanques, oleodutos e refinarias. Não há limites de recursos, como a energia elétrica necessária para o bombeamento ou menção a quaisquer custos variáveis do processo/sistema.

As equações serão descritas usando uma das notações de programação orientada a objeto do Matlab®, na qual um ponto representa a associação de um objeto e uma propriedade deste. Para exemplificar, um objeto da classe “oleod” possui cinco propriedades, entre elas as propriedades “V” (vetor de volumes na sequencia que entra no oleoduto) e “st” (status). Ao criar um conjunto de objetos da classe “oleod” denominado “oAll”, para ter acesso ao primeiro oleoduto desse conjunto, usa-se a notação “oAll(1)”. Para as propriedades de volume e de status desse oleoduto, utiliza-se, respectivamente, as notações “oAll(1).V” e “oAll(1).st”. Para acessar apenas o segundo volume do primeiro oleoduto, utiliza-se a notação “oAll(1).V(2)”

A representação utilizada para as notações (ou índices) e para as variáveis decisórias foram apresentadas no Tópico 7.1.1.

As restrições do problema de otimização são apresentadas a seguir.

Restrição 1: cada navio só pode ter um status sendo cada um deles apresentados na Tabela 9.

Tabela 9 - Status possíveis para o navio

Status	Descrição
1	Indisponível
2	Disponível para atracar
3	Atracando
4	Descarregando
5	Desatracando

Restrição 2: cada píer só pode ter um status sendo cada um deles apresentados na Tabela 10.

Tabela 10 – Status possíveis para o píer

Status	Descrição
1	Disponível
2	Indisponível

Restrição 3: cada navio atraca em, no máximo, um píer conforme a Equação 9.

$$\sum_{p} np(n,p) \leq 1 \tag{9}$$

Restrição 4: o navio fica disponível para atracar se o status anterior dele for “Indisponível” e se a hora de chegada dele já estiver passado e se ele possui volume de petróleo em seu interior, conforme a Equação 10.

$$navioAll(n).st \leftarrow 2 \Leftrightarrow navioAll(n).st = 1 \ \& \ navioAll(n).H0 \geq H(t) \ \& \ navioAll(n).V > 0 \quad (10)$$

Restrição 5: cada navio só pode atracar se o píer estiver disponível conforme a Equação 11.

$$navioAll(n).st \leftarrow 3 \Leftrightarrow navioAll(n).pier.st = 1 \quad (11)$$

Restrição 6: cada navio só pode atracar se o status anterior dele for “Disponível para atracar” conforme a Equação 12.

$$navioAll(n).st \leftarrow 3 \Leftrightarrow navioAll(n).st = 2 \quad (12)$$

Restrição 7: cada navio só pode começar a mandar o petróleo para o tanque se o status anterior dele for “atracando” conforme a Equação 13.

$$navioAll(n).st \leftarrow 4 \Leftrightarrow navioAll(n).st = 3 \quad (13)$$

Restrição 8: cada navio só pode desatracar se o status anterior dele for “Desatracando” conforme a Equação 14.

$$navioAll(n).st \leftarrow 5 \Leftrightarrow navioAll(n).st = 4 \quad (14)$$

Restrição 9: cada píer só fica “Indisponível” quando um navio atraca nele, conforme Equação 15.

$$navio(n).st = 3 \Rightarrow navio(n).pier.st \leftarrow 2 \quad (15)$$

Restrição 10: cada tanque só pode ter um status sendo cada um deles apresentados na Tabela 11

Tabela 11 - Status possíveis para o tanque

Status	Descrição
1	Carregando
2	Sedimentando (min.24h p/ começar a descarregar)
3	Pronto para descarregar

4	Descarregando

Restrição 11: cada tanque só pode começar a sedimentação se ele atingir o volume máximo, ou se a variável *htq* estiver ativa, conforme Equação 16.

$$tqAll(tq).st \leftarrow 2 \quad \Leftrightarrow \quad tqAll(tq).st = 1 \quad \& \, (htq(H(t), tq) = 1 \; ou \; tqAll(tq).V = tqAll(tq).Vmax \tag{16}$$

Restrição 12: cada tanque só pode ficar disponível para descarregar petróleo no oleoduto se o status anterior dele for "sedimentando" e se já tiverem passado 24h da hora em que começou o estado de sedimentação conforme a Equação 17.

$$tqAll(tq).st \leftarrow 3 \quad \Leftrightarrow \quad tqAll(tq).st = 2 \quad \& \, H(t) \geq tqAll(tq).Hset \; + 24 \tag{17}$$

Restrição 13: só pode haver apenas um tanque conectado a cada oleoduto conforme a Equação 18

$$tqAll(tq).st \leftarrow 4 \; \Leftrightarrow tqAll.st \; \neq 4 \tag{18}$$

Restrição 14: o volume de cada tanque só pode variar entre a capacidade mínima e a capacidade máxima conforme Equação 19.

$$tqAll(tq).Vmin \leq tqAll(tq).V \leq tqAll(tq).Vmax \tag{19}$$

Restrição 15: se o volume disponível no tanque não for o suficiente para ser enviado na vazão máxima no intervalo de tempo mínimo da discretização (int) o volume não é enviado conforme Equação 20.

$$tqAll(tq).Q \neq 0 \Leftrightarrow \frac{tqAll(tq).Vmax - tqAll(tq).Vmin}{int} \geq tqAll(tq).Qmax \tag{20}$$

Restrição 16: se o tanque está descarregando e sua vazão passa a ser nula (não é mais possível descarregar), seu status muda para carregando conforme Equação 21.

$$tqAll(tq).st = 4 \; \& \; tqAll(tq).Q = 0 \Rightarrow tqAll(tq).st = 1 \tag{21}$$

Restrição 17: o oleoduto só pode transferir volume para a refinaria na vazão máxima permitida conforme Equação 22. A vazão é definida pela classe de petróleo

existente no oleoduto naquele momento, sendo a vazão máxima a mínima entre as vazões máximas das classes contidas no oleoduto.

$$oAll(o).Q = \min(oAll(o).Cl.Qmax) \tag{22}$$

Restrição 18: o volume de petróleo nas refinarias só pode variar entre a capacidade mínima e a capacidade máxima da mesma conforme Equação 23.

$$ref(r).Vmin \leq ref(r).V \leq ref(r).Vmax \tag{23}$$

7.4 ALGORITMO DE COLÔNIA DE FORMIGAS

Algoritmos de otimização, muitas vezes, avaliam iterativamente os candidatos da solução para melhorar a ótima. Por isso é importante enfatizar a diferença entre avaliar a função objetivo e realizar uma iteração no modelo. A avaliação da função objetivo é realizada cada vez que se encontra um conjunto candidato a solução, o que no caso do algoritmo de colônia de formiga, representa a ação de uma única formiga. Uma iteração refere-se a uma rodada em um ciclo de um algoritmo e é uma repetição de uma sequência específica de instrução dentro do programa (Weise, 2009).

Cada formiga escolhe o valor da próxima variável baseado numa decisão probabilística guiada pela melhor resolução do passo em questão. O melhor passo não necessariamente define o melhor caminho global, por isso a decisão probabilística ponderada pela solução ótima local de cada passo, representa um avanço na resolução do problema, mas ao mesmo tempo pode, em alguns casos, levar a um atraso na resolução. Para gerar o número aleatório é utilizado o método de Monte Carlo baseado em uma distribuição uniforme de números aleatórios. Para a escolha do resultado, tendo em mãos as probabilidades, foi implementado o método da roleta com apenas um ponteiro.

O algoritmo implementado só permite a escolha de cada formiga pelos caminhos viáveis; sendo os caminhos inviáveis impedidos por condicionais no código, evitando uso de tempo computacional com formigas que levam a soluções impossíveis. O critério de parada estabelecido foi o do atingimento do número máximo de iterações.

Existem pelo menos dois tipos possíveis de convergência em um algoritmo de otimização estocástica, são eles a convergência em valor e convergência em solução. Ao estudar a convergência de valor, estamos interessados em avaliar a probabilidade de que o algoritmo gere uma solução ótima pelo menos uma vez, enquanto a convergência em solução avalia a probabilidade de que o algoritmo convirja a um estado que continue resultando na mesma solução ideal (Dorigo e Blum, 2005).

Foi utilizada a atualização tardia de feromônio, que é o caso em que o feromônio só é atualizado após a função objetivo é encontrada. A atualização é feita proporcionalmente ao valor encontrado da função objetivo para a melhor formiga (maior FO) de cada iteração.

Definição da quantidade inicial de feromônio (fer_0) pode ser feita utilizando a razão entre o número total de formigas e o valor estimado para a função objetivo na primeira iteração.

$$fer_0 = 40/4000 = 0{,}01$$

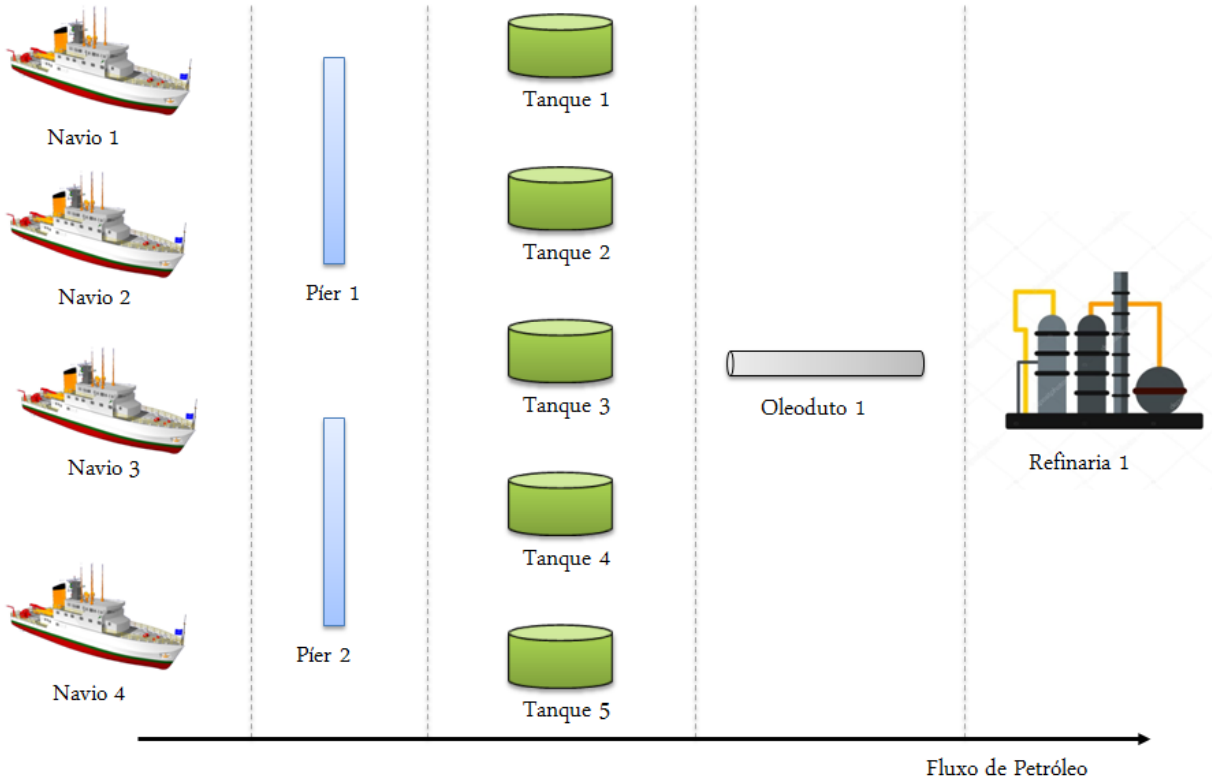
Os parâmetros heurísticos foram desenvolvidos individualmente para cada uma das variáveis, de forma a enviesar a resolução do problema nas primeiras iterações conforme mostrado a seguir:

- np: o píer com menor custo de utilização é favorecido, sendo o fator heurístico o inverso deste custo.
- Ect: a quantidade de eventos no envio do petróleo do navio para o tanque é favorecida quanto maior a quantidade de eventos.
- Ect_point: a decisão dos tempos de início de cada evento não possui heurística, lembrando que a utilização de heurística é opcional.
- ct: os petróleos vão preferencialmente para os tanques que contém o maior valor de petróleo na refinaria, sendo o fator heurístico esse valor normalizado pelo maior valor.
- htq: a probabilidade de nenhum tanque começar a sedimentação antes de encher completamente é 20 vezes maior do que um dos tanques começar. O fator heurístico para todos os tanques é igual.

7.5 ESTUDO DE CASO 1

Este modelo é composto por quatro navios, um terminal contendo dois píeres e cinco tanques, um oleoduto e uma refinarias hipotética; não foram modeladas as subestações nem os sistemas auxiliares de bombeio. O modelo do estudo de caso alocação de petróleo é ilustrado na Figura 39, na qual o sentido do fluxo de petróleo é indicado pela seta na parte inferior.

Figura 39 – Esquema do Estudo de Caso 1



Os quatro petróleos selecionados foram distribuídos em quatro grupamentos, como mostrado na Tabela 12; essa categorização pode ser representada na forma de uma matriz de variáveis binárias, na qual o número “1” indica que o petróleo pertence ao grupo e o “0” que não pertence. Na simulação, essa informação aparece como uma relação de agregação entre o objeto classe (grupamento) e o objeto petróleo.

Tabela 12 - Classificação dos petróleos em grupamentos

	Grupamento 1	Grupamento 2	Grupamento 3	Grupamento 4
Petróleo 1	x	-	-	x

Petróleo 2	x	x	-	-
Petróleo 3	-	-	x	x
Petróleo 4	x	x	x	x

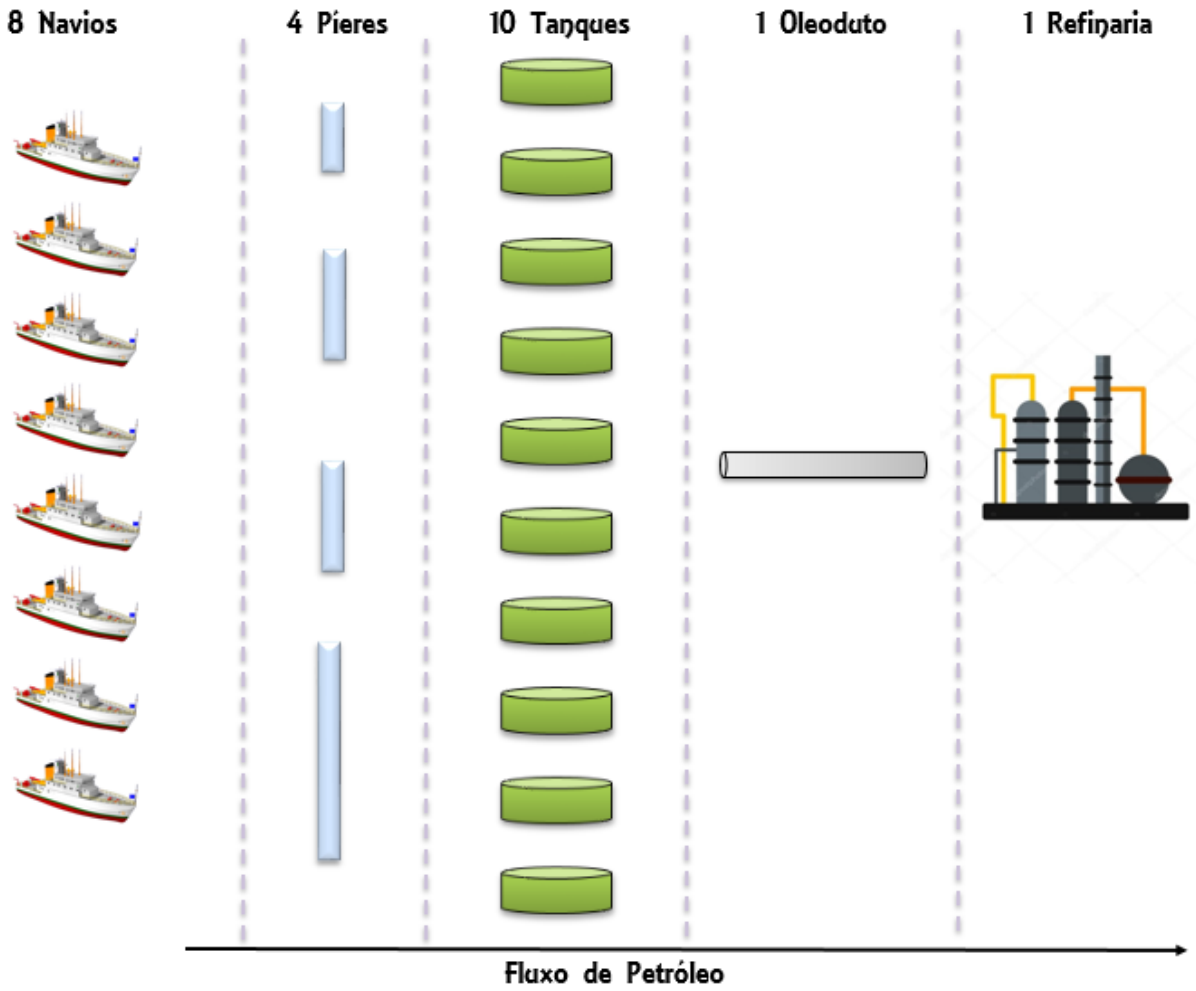
Os dados completos do estudo de caso estão apresentados no apêndice C.

O modelo pode escolher entre 5 e 20 eventos em cada píer ao longo do horizonte de 96h dividido em intervalos de 0,1h. Na escolha do píer, existem 6 possibilidades de atracação, já que são 2 píeres e os navios 2 e 3 só podem escolher o mesmo píer (pois simulam 1 navio com 2 petróleos diferentes). O algoritmo pode escolher entre 5 e 20 eventos, que podem iniciar em qualquer tempo, desde que consecutivos. As 96h do horizonte de tempo são divididas em intervalos de tempo de 0,1h totalizando 960 segmentos de tempo.

7.6 ESTUDO DE CASO 2

O estudo de caso 2 é uma extensão do estudo de caso 1, possuindo mais elementos (navios, píeres e tanques), horizonte de tempo maior e uma restrição a mais. Esse estudo é composto por oito navios, um terminal contendo quatro píeres e dez tanques, um oleoduto e uma refinarias hipotética como pode ser visualizado na Figura 40. Não foram modeladas as subestações nem os sistemas auxiliares de bombeio. A restrição a mais que esse problema possui é de que nem todos os navios podem atracar em todos os píeres por causa de um limite de capacidade dos píeres. São sete petróleos classificados nos em cinco grupamentos.

Figura 40 - Esquema do estudo de caso 2



8. RESULTADOS E DISCUSSÃO

A formulação e desenvolvimento do modelo de *scheduling* empregado representa o produto mais laborioso desse trabalho e a modelagem completa pode ser encontrada em detalhes no Apêndice D. A modelagem passou por 4 etapas principais:

1. Definir quais seriam as variáveis do modelo de modo a permitirem todas as liberdades possíveis de resposta e, ao mesmo tempo, não serem redundantes para que não haja dispêndio desnecessário de processamento computacional.
2. Definição das heurísticas que seriam utilizadas e seus respectivos valores, de forma que elas ajudem a encontrar o resultado, mas sem restringir o algoritmo.
3. Escolher o tipo de arquitetura seria utilizada na programação entre a imperativa (código direto), a funcional (utilizando funções) e a orientada a objeto.
4. Programar o sequenciamento de tarefas de modo que elas não se sobreponham e que o balanço de massa e demais restrições sejam respeitados.
5. Definir o valor das constantes do problema que interferem diretamente na qualidade da resposta e no tempo de processamento computacional empregado; são elas: o intervalo de tempo em que o horizonte será dividido, o número de formigas e o número de iterações.

8.1 PARAMETRIZAÇÃO

Os principais parâmetros de otimização do ACO são a quantidade inicial de feromônio (fer_0), a constante de deposição do feromônio (fer_{atu}), a constante de evaporação do feromônio (fer_{evap}), alfa e beta. Os três primeiros parâmetros são utilizados no início e no fim de cada iteração, exercendo a função de inicializar a quantidade de feromônio e realizar a deposição e evaporação do feromônio. Os dois últimos parâmetros (alfa e beta) são utilizados no meio da iteração para o cálculo da probabilidade de escolha de cada valor das variáveis do modelo.

Como a função objetivo não possui uma relação direta com as variáveis e as informações heurísticas utilizadas não são, necessariamente, diretamente relacionadas com a função objetivo, os parâmetros não possuem uma calibração direta como no experimento da ponte dupla (ver Tópico 6.1.3). Por conseguinte, o processo de parametrização foi dividido em três fases:

- Levantar quais dos cinco parâmetros influenciam mais o resultado e de qual modo. Assim, pode-se identificar aqueles que possuem uma relação mais direta com o resultado e os que são mais difíceis de serem parametrizados.
- Realizar a sensibilidade dos parâmetros mais simples utilizando alguns valores-chave para eles e mantendo os outros constantes.
- Com os valores encontrados para os parâmetros mais simples, realizar a parametrização dos parâmetros mais complexos utilizando Algoritmo Genético.

Na primeira fase, foi identificado que a quantidade de feromônio inicial é redundante com os valores de atualização e evaporação, por isso ele não precisa ser parametrizado. Foi feito um estudo de sensibilidade para identificar, entre os quatro parâmetros restantes, quais deles eram o que tinham maior influência na performance do programa deixando a quantidade de feromônio inicial sempre fixa e uniformemente distribuída. Para cada um deles o algoritmo foi rodado 50 vezes e desses 50 valores de função objetivo encontrados foi calculado seu média simples e identificado o valor máximo. Para cada uma das 18 sensibilidades foram utilizadas 50 iterações, 30 formigas, passo de 30 min, as quais exigiram aproximadamente 15 min de tempo de processamento computacional cada.

A sensibilidade do parâmetro alfa foi realizada utilizando os valores de 0, 0,5, 1, 2, 2,5, 5 e 10, de acordo com a Figura 41 com detalhes na Tabela 13. Constata-se que, para pequenos valores de alfa a função objetivo atinge valores semelhantes, porém a partir de algum valor entre 3 e 5 o FO médio passa a cair bastante de valor. A partir de alfa = 5, começa a ocorrer uma grande oscilação no seu resultado de cada iteração, pois altos valores desse parâmetro aumenta a probabilidade de escolha da variável baseada apenas nos feromônios depositados, levando à ocorrência de estagnação em mínimos locais e, consequentemente, diminuindo o FO médio

dessas iterações. Os melhores valores de alfa foram os da sensibilidade 3 e 5 por apresentarem o maior FO máximo e maior FO médio sem um desvio padrão muito alto em relação ao outros.

Figura 41 - Sensibilidade do parâmetro alfa

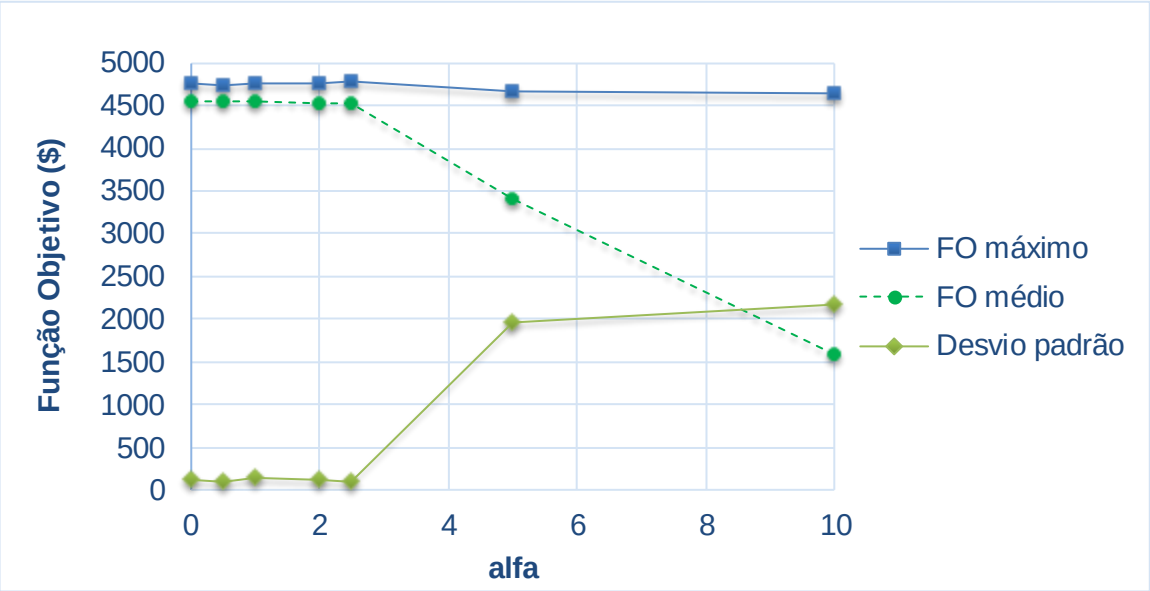


Tabela 13 - Sensibilidade do parâmetro alfa

Parâmetro	Sens.1	Sens.2	Sens.3	Sens.4	Sens.5	Sens.6	Sens.7
fer_0	0,1	0,1	0,1	0,1	0,1	0,1	0,1
fer_{atu}	0,01	0,01	0,01	0,01	0,01	0,01	0,01
fer_{evap}	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001
alfa	0	0,5	1	2	2,5	5	10
beta	1	1	1	1	1	1	1
FO máximo	4765	4741	4765	4752	4792	4661	4648
FO médio	4544	4543	4544	4518	4531	3413	1594
Desvio padrão	111	94	131	111	106	1950	2167

Para valores altos de alfa começa a haver maior dispersão entre os valores de FO encontrados, como pode ser observado na Figura 41, a qual mostra o grande aumento do desvio padrão com o aumento do valor de alfa. Esse resultado unido com o fato do FO máximo também ter diminuído com o aumento do valor de alfa demonstra que os valores menores são mais adequados a essa modelagem.

A sensibilidade do parâmetro beta foi realizada utilizando os valores de 0, 0,5, 2,5, 3, 5, 10 e 100, de acordo com a Figura 42 com detalhes na Tabela 14. Os valores

foram escolhidos de modo a tentar identificar alguma tendência, que não podia ser percebida até o valor 10, mas adicionando-se o valor de 100 percebe-se uma pequena piora no resultado. Nota-se que o parâmetro beta exerce pouca influência no resultado, principalmente quando comparado com o parâmetro alfa. Tanto os valores máximo, médio e o desvio padrão da função objetivo para todos os valores de beta foram similares. Os valores de beta iguais a 3 e 5 apresentaram os melhores resultados, apesar da pouca diferença quando comparado com os outros valores.

Figura 42 - Sensibilidade do parâmetro beta

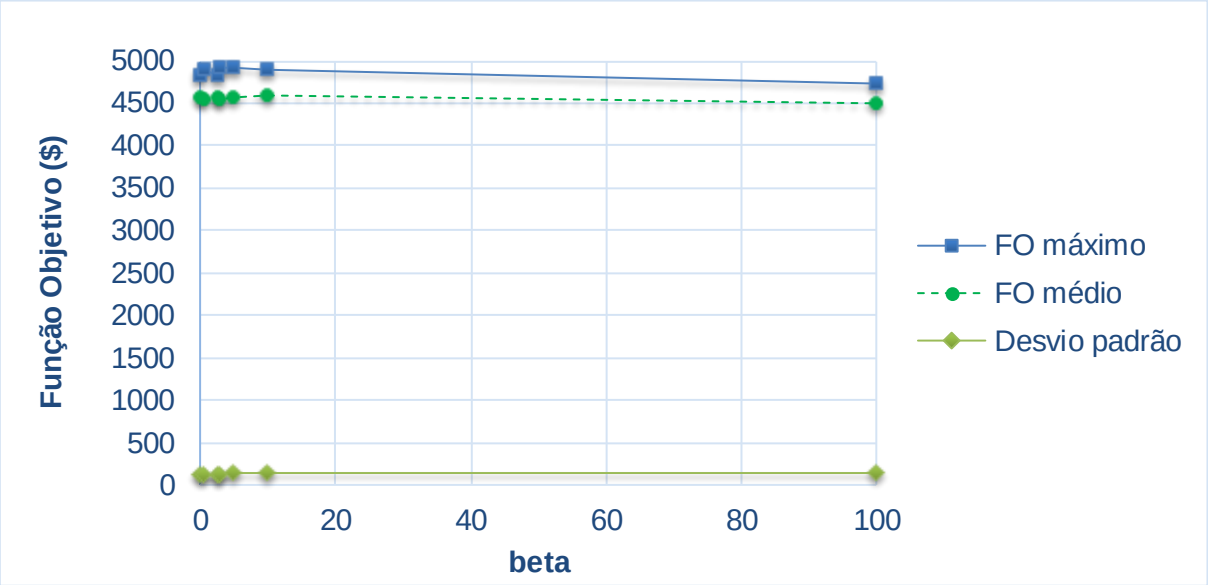


Tabela 14 - Sensibilidade do parâmetro beta

Parâmetro	Sens.8	Sens.9	Sens.10	Sens.11	Sens.12	Sens.13	Sens.14
fer_0	0,1	0,1	0,1	0,1	0,1	0,1	0,1
fer_{atu}	0,01	0,01	0,01	0,01	0,01	0,01	0,01
fer_{evap}	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001
alfa	1	1	1	1	1	1	1
beta	0	0,5	2,5	3	5	10	100
FO máximo	4816	4885	4832	4916	4914	4893	4736
FO médio	4567	4547	4562	4555	4559	4597	4501
Desvio padrão	120	120	127	121	142	142	146

A sensibilidade dos parâmetros de feromônio foi realizada de acordo com a Tabela 15, na qual, pode-se observar, comparando a sensibilidade 15 com a 18, que a evaporação dos feromônios (fer_{evap}) apresenta baixa influência no resultado. Já

comparando a sensibilidade 16 com a 17, percebe-se a alta dependência do resultado com a constante de atualização de feromônios (fer_{atu}).

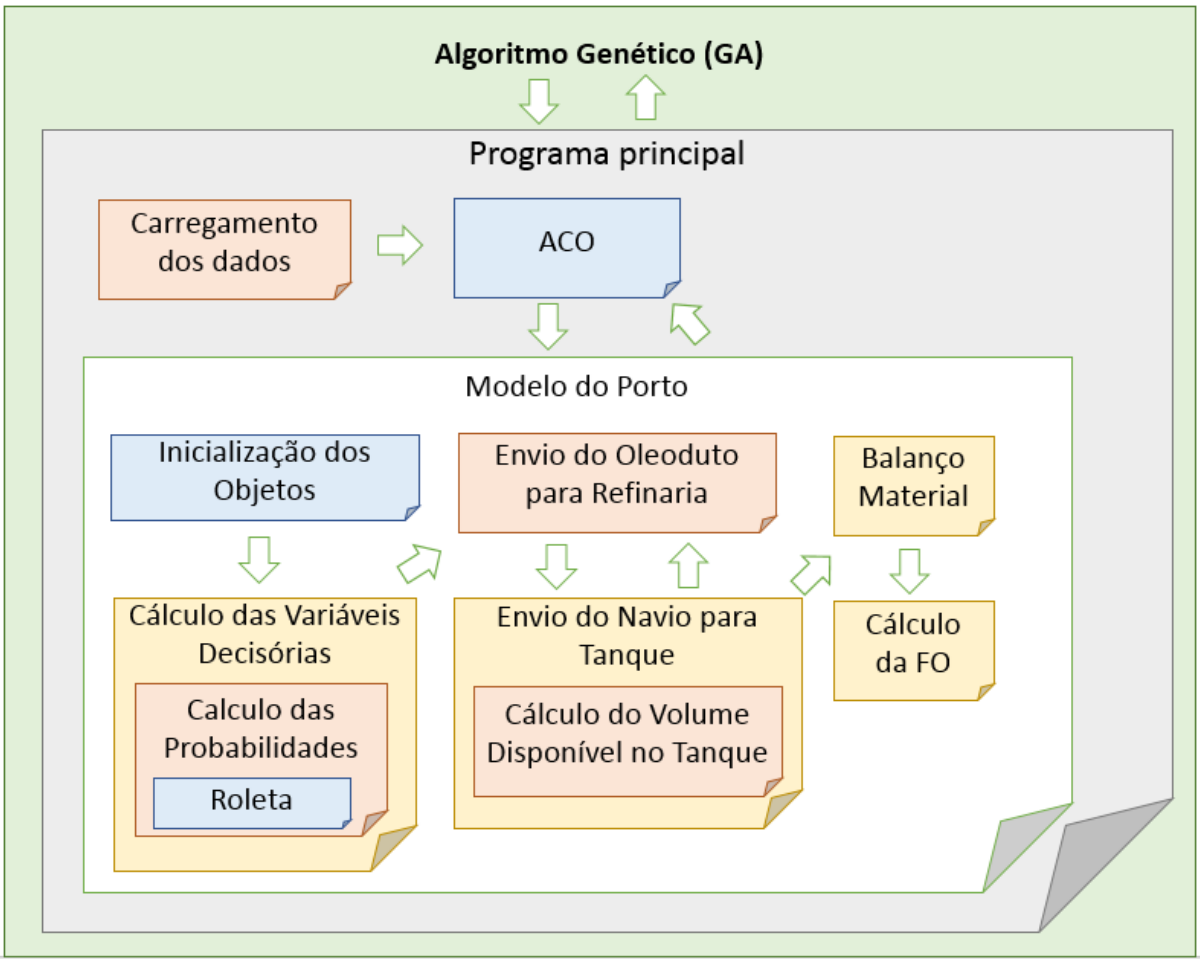
Tabela 15 - Sensibilidade dos parâmetros de feromônio

Parâmetro	Sens.15	Sens.16	Sens.17	Sens.18
fer_0	0,1	0,1	0,1	0,1
fer_{atu}	0,001	0,01	0,005	0,001
fer_{evap}	0,0001	0,001	0,001	0,001
alfa	2,5	2,5	2,5	2,5
beta	1	1	1	1
FO máximo	4821	4553	4807	4862
FO médio	4584	4228	3689	4546
Desvio padrão	92	610	540	115

Por conseguinte, a segunda fase da sensibilidade mostrou que, entre os quatro parâmetros analisados, alfa e beta apresentaram comportamentos mais padronizados, podendo ser observado uma tendência no caso do alfa e uma dependência pequena no caso do beta, enquanto os parâmetros de atualização e evaporação dos feromônios demonstraram maior intervenção no resultado. Diante disso, foi escolhido realizar a terceira fase da sensibilidade em cima dos últimos dois parâmetros, fixando os valores de alfa e beta naqueles que suscitaram os melhores resultados na segunda fase.

Depois de realizada a sensibilidade inicial dos parâmetros, identificando quais os que mais influenciam na dinâmica da resposta, ainda é necessário realizar uma parametrização mais rigorosa. Utilizando a otimização com algoritmos genéticos (do inglês, *Genetic Algorithm*), foi possível fazer a parametrização do algoritmo do estudo de caso deste trabalho. Para tal, foi necessário transformar o programa principal em uma *function,* a qual é chamada diversas vezes pelo novo programa principal, que contém a otimização por GA, conforme o esquema da Figura 43.

Figura 43 - Estrutura do código Matlab® para a parametrização do estudo de caso do modelo do porto



A função de otimização com GA faz parte da *toolbox* do Matlab® e foram utilizadas as configurações padrão do mesmo. Os resultados foram encontrados utilizando intervalos de tempo fixo de 25 min, aplicado ao estudo de caso 1 apresentado no tópico 7.5. Cada uma das sensibilidades utilizando o método da Figura 43 (sensibilidades 19 a 23) precisou de mais de 8h para convergir para um resultado. Às variáveis alfa e beta foi permitido uma variação entre 10^{-6} e 10, enquanto para as variáveis de depósito e evaporação de feromônios foi permitido a variação entre 10^{-8} e 1. Foram utilizadas nas sensibilidades de 19 a 22 apenas 20 iterações e 5 formigas, devido ao alto tempo computacional exigido em realizar o ACO na quantidade de iterações necessárias para o GA.

Como os melhores da segunda etapa da parametrização foram os valores para alfa de 2,5 e o de 1, enquanto o de beta foi 3, esses valores foram utilizados para as

sensibilidades 19 e 20, encontrando os valores de FO máximo da Tabela 16. Em seguida foram feitas as sensibilidades 21 e 22, agora com os quatro parâmetros com incógnita, sendo que o fundo cinza nos valores da tabela representa aquelas variáveis que ficaram fixas em cada sensibilidade. Dado a aleatoriedade inerente da otimização, o resultado da parametrização oscila um pouco, como mostrado na diferença de resultados entre as sensibilidades 21 e 22.

Tabela 16 - Parametrização do ACO utilizando GA

Variáveis	Sens.19	Sens.20	Sens.21	Sens.22
fer_0	0,1	0,1	0,1	0,1
alfa	2,5	1	0,0428	0,0533
beta	3	3	0,0492	0,0644
fer_{atu}	0,0322	0,0291	0,5040	0,8154
fer_{evap}	0,0041	0,0117	44,03	37,17
FO máximo	4894	4893	4934	4925

Em virtude da não linearidade do modelo e do caráter aleatório dos algoritmos (GA e ACO) vários ótimos diferentes podem ser encontrados na parametrização, ou seja, mesmos parâmetros podem encontrar resultados diferentes e o mesmo resultado pode ser encontrado por um conjunto de parâmetros diferente. Os dois casos em que foram encontrados os maiores valores de função objetivos são mostrados na Tabela 16. Entre as parametrizações 21 e 22 a função objetivo variou 0,2%, enquanto os parâmetros do problema variaram entre 16% (constante de evaporação do feromônio) e 62% (constante de atualização do feromônio).

8.2 ESTUDO DE CASO 1

O primeiro estudo de caso é composto de quatro navios, dois píeres, cinco tanques, um oleoduto e uma refinaria. Os resultados serão comparados com o encontrado pelo Más (2001); forma utilizados os mesmos dados de entrada, sendo a única premissa diferente a presença de um navio a mais, pois a modelagem desse trabalho só suporta a presença de um petróleo por navio. A refinaria modelada é uma refinaria hipotética que representa a REPLAN e a REVAP modeladas como uma unidade. Afim de tornar os dois modelos comparáveis, adicionou-se uma restrição para que esses dois navios atracassem sempre num mesmo píer, sendo

que o primeiro não possui tempo de desatracação e o segundo não possui tempo de atracação. Essa premissa diferenciada influencia apenas no fato de que, nesse modelo, não é possível o bombeamento do segundo petróleo sem ter terminado o primeiro. Assim, enquanto o estudo de caso de Más (2001) possui 3 navios, sendo que um deles contém 2 petróleos diferentes, o deste trabalho possui quatro navios, cada um com um petróleo diferente. Outro aspecto tratado diferentemente entre os dois modelos é o tempo. Enquanto o modelo do Más (2001) utiliza tempo continuo dotado eventos, nessa dissertação foi utilizado tempo discreto e uma variável auxiliar que simula os eventos afim de aproximar o modelo da realidade e tornar os resultado comparáveis.

O modelo é livre para escolher entre 5 e 20 eventos por píer, ou seja, em cada píer a quantidade possível de troca de envio de um navio para um tanque é de no máximo 20 vezes. Foi necessário impor um limite máximo, pois o tempo na modelagem é discreto, e se deixasse livre escolha, poderia acontecer de a cada poucos minutos ocorrerem novos alinhamentos, o que afastaria o modelo da realidade.

Como já explicitado anteriormente, a função objetivo é composta das seguintes parcelas: receita de óleo armazenado nas refinarias, receita de óleo armazenado no porto, custo de crus dos navios, custo de operação de píeres, custo de sobrestadia e custo de interface no oleoduto. O acompanhamento dessas parcelas da função objetivo teve papel fundamental na criação das variáveis do problema, na escolha da heurística e na parametrização do problema.

Como o píer 2 possui menor custo de utilização que o píer 1 e não há nenhuma restrição de utilização deles, o resultado ótimo tende a ser colocar todos os navios no píer 1. Contudo, alocar todos os navios no mesmo píer torna o processo de encher os tanques do terminal mais demorado e o valor do petróleo a refinaria é maior, além de existir a restrição de estoque mínimo de petróleo da refinaria.

Utilizando poucas formigas (de 20 a 30) o algoritmo atingia, no máximo, a FO em torno de $4920, então, para resolver o problema foram utilizadas 500 formigas e 150 iterações e o tempo da rodada passou de poucos minutos a algumas horas. O melhor resultado encontrado foi de $5008, enquanto o modelo linear inteiro misto do Rodrigo Más (2001) encontrou a função objetivo de $5.281, como pode ser mostrado

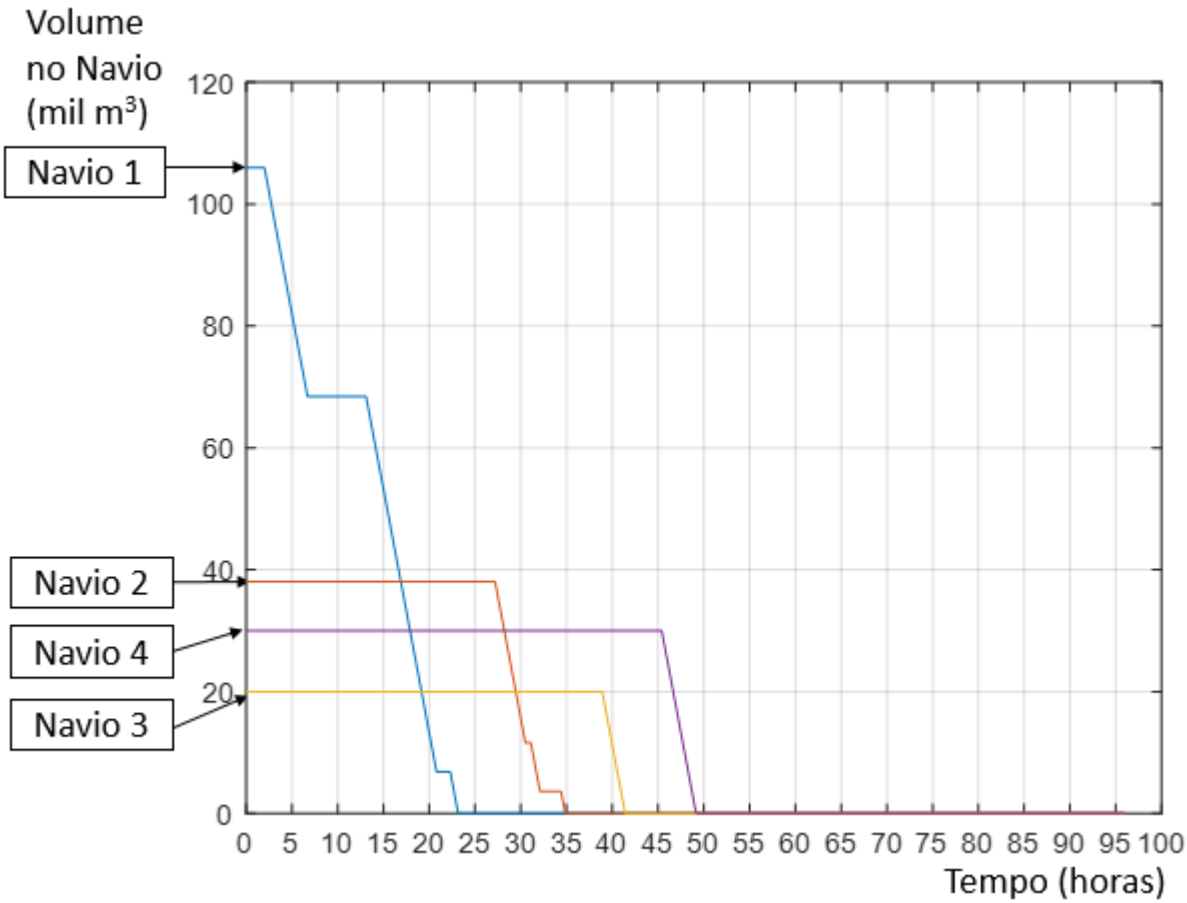
na Tabela 17. O resultado do ACO associou todos os navios ao píer 2, deixando o píer 1, que é mais caro, vago durante todo o horizonte do problema. Não houve custo de sobrestadia, demonstrando que os navios não ficaram tempo maior que o permitido atracados no píer, porém o custo de operação dos píeres não foi otimizado porque os navios passaram parte do tempo que estavam atracados sem bombear petróleo. Essa demora no bombeamento foi em parte responsável pela chegada de menos petróleo na refinaria e maior manutenção do mesmo nos terminais, explicitado pela primeira e segunda linha da Tabela 17.

Tabela 17 - Resultados das parcelas da função objetivo

Valores em $	**ACO**	**MILP (Más, 2001)**
Receita de óleo nas refinarias	43.555	50.288
Receita de óleo no porto	-16.388	-22.888
Custo de crus dos navios	22.026	22.026
Custo de sobrestadia	0	0
Custo de operação de píeres	129	81
Custo de interface	4	12
Função Objetivo (FO)	5.008	5.281

O descarregamento do navio para o melhor resultado encontrado é mostrado na Figura 44, na qual pode ser observado que, em virtude de todos os navios terem sido atracados no mesmo píer, um navio só começa a ser descarregado após o fim da desatracação do navio anterior; o primeiro navio atraca assim que chega no porto, enquanto a atracação dos outros ocorre em sequência, assim que o píer 2 fica disponível.

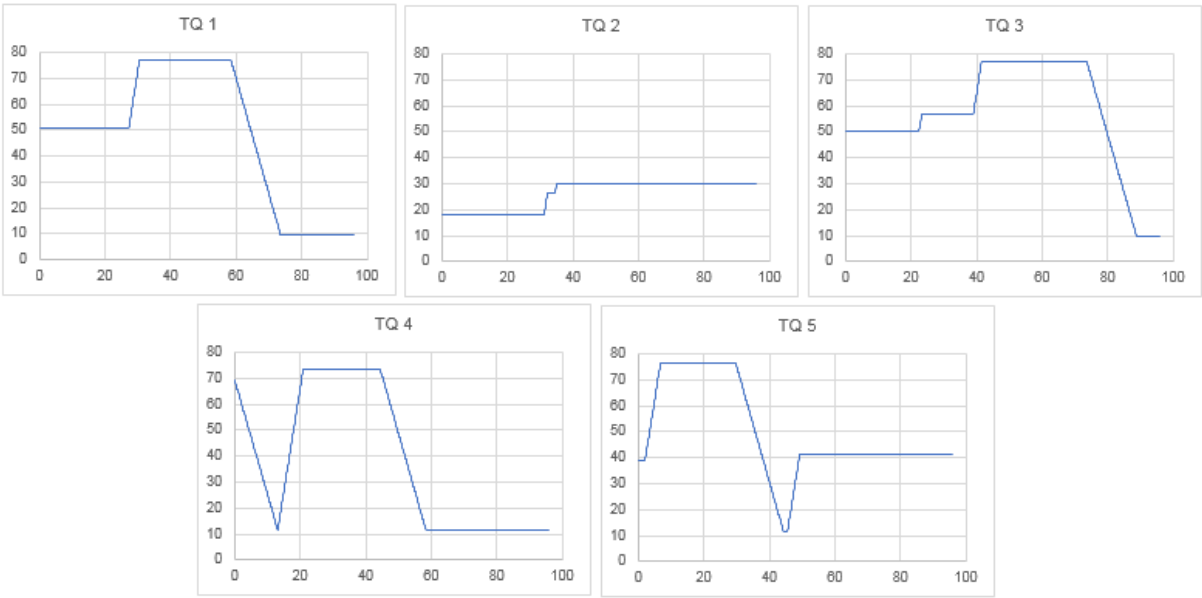
FIGURA 44 - RESULTADO DO DESCARREGAMENTO DOS NAVIOS



O total de eventos (mudança de tanque a ser enviado o petróleo) em cada um os navios foi de 3, 6, 7 e 1. Assim, o último navio (Navio 4) optou por enviar todo seu conteúdo de petróleo para apenas 1 dos tanques.

Na Figura 45, é mostrado o gráfico de nível dos tanques, em mil m^3, do problema, no qual pode-se observar a variação do volume de petróleo em cada um dos tanques em função do tempo. Como cada tanque tem que passar por um período de 24h parado antes de bombear petróleo para o oleoduto, é observado na Figura 38, esses patamares de no mínimo 24h seguidos da diminuição do volume armazenado. Verifica-se que o tanque 5 é aquele que recebe todo volume do navio 4, pela hora em que ele começa a encher.

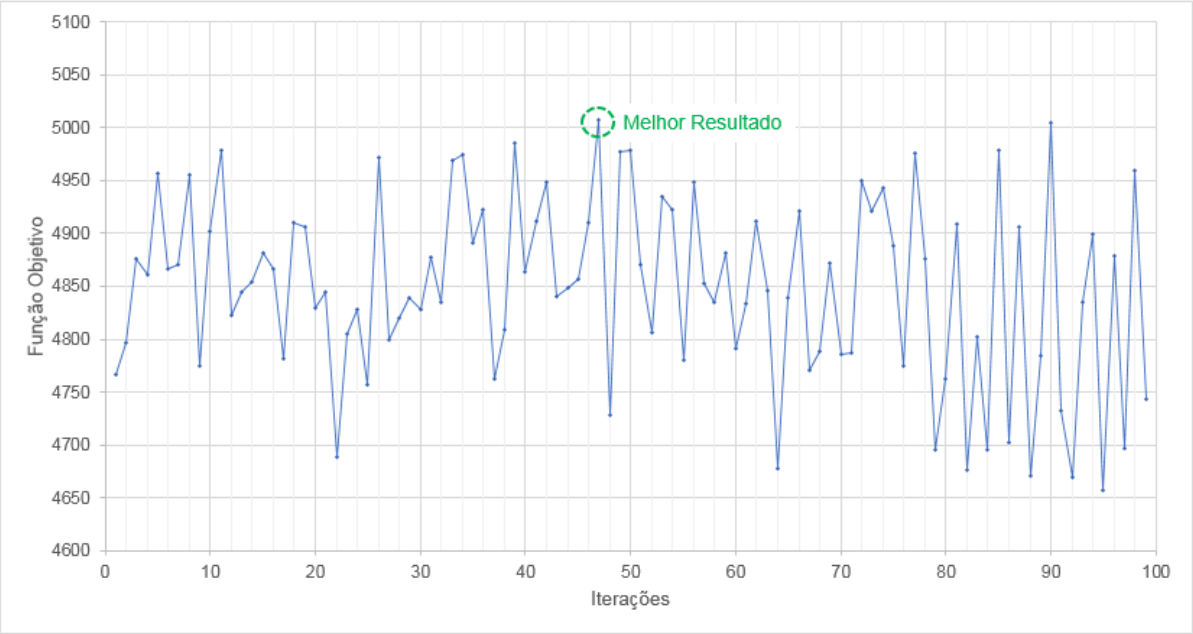
FIGURA 45 – NIVEL DOS TANQUES EM FUNÇÃO DO TEMPO



8.2.1 DESEMPENHO DO ACO NO ESTUDO DE CASO 1

O resultado da otimização em função das iterações utilizando 500 formigas por iteração é mostrado na Figura 46 Mesmo que em algumas rodadas a função objetivo apresente valores baixos, muitos inclusive negativos, esses valores não podem ser visualizados na Figura 46, pois esta apresenta apenas o resultado da formiga que performou melhor em cada iteração.

Figura 46 - Resultado do ACO para o estudo de caso 1



Aumentar o número de formigas aumenta a capacidade exploratória do algoritmo, mas, ao mesmo tempo, deixa as iterações mais lentas.

Mesmo não atingindo o resultado ótimo esperado, seja por limitação da modelagem ou pela natureza do algoritmo, o resultado encontrado foi satisfatório, e demonstra a habilidade do ACO em encontrar respostas viáveis de forma rápida quando munido de uma boa parametrização.

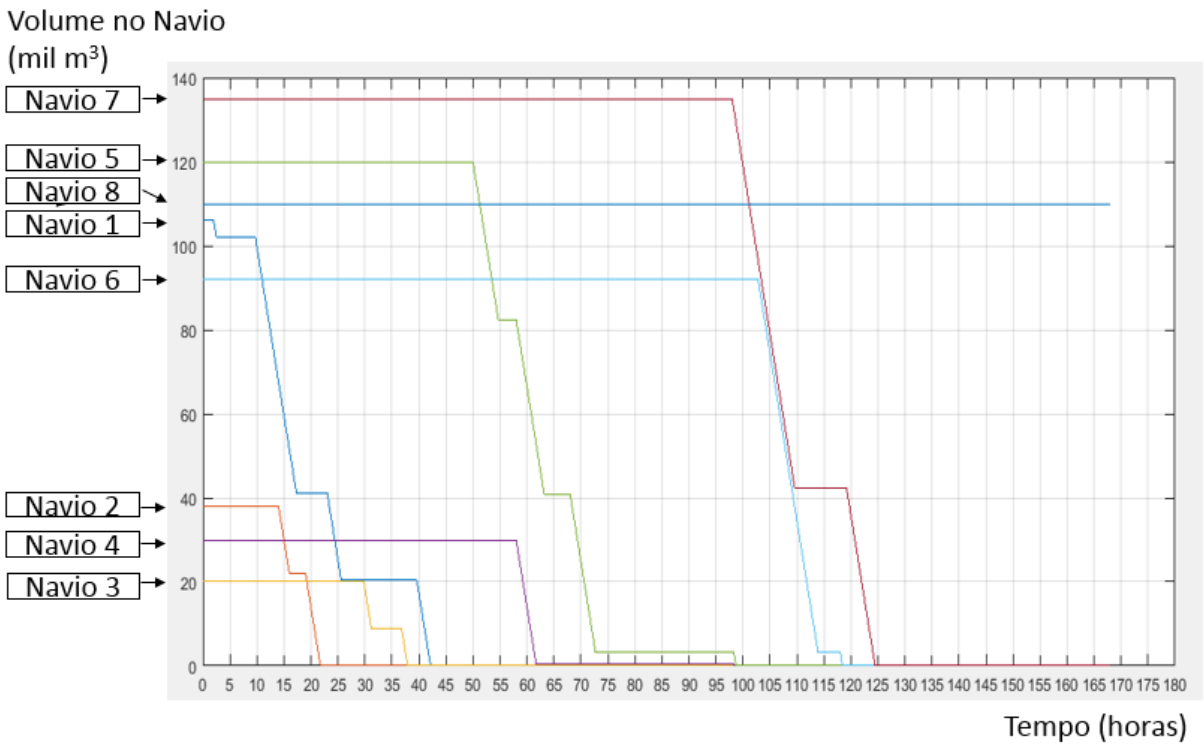
Quando a complexidade do problema aumenta (aumentando a quantidade de navios ou tanques, por exemplo) a parametrização utilizando algoritmo genético torna-se impraticável por causa da alta demanda computacional. Foi utilizado um computador com processador i5 3210-M 2.5 GHz com 6 Gb de memória RAM.

8.3 ESTUDO DE CASO 2

O segundo estudo de caso é composto de sete navios, quatro píeres, dez tanques, um oleoduto e uma refinaria. Além de uma restrição extra: nem todos os navios podem ser associados a todos os píeres disponíveis, pois alguns píeres tem uma limitação de porte do navio que é possível atracar (limitação física).

Foi utilizado passo de 0,1h e a possibilidade de escolha entre 10 e 20 eventos por navio durante o horizonte de tempo. Os parâmetros do ACO foram α=2, β=1, fer0=0,1, constante de atualização dos feromônio de 10^-5, e de evaporação de 0,0178, 1000 formigas e 100 iterações. O melhor resultado alcançado atingiu a FO de $476 e esse terá sua solução analisada. O volume no navio em função do tempo é mostrado na Figura 47. Como o valor do óleo na refinaria é maior do que no terminal e inexistente no navio (o petróleo no navio é considerado apenas um custo) nota-se que o fato do navio 8 não ter descarregado no terminal no horizonte proposto diminui o valor da função objetivo, sendo um dos motivos para o baixo desempenho dessa solução encontrada.

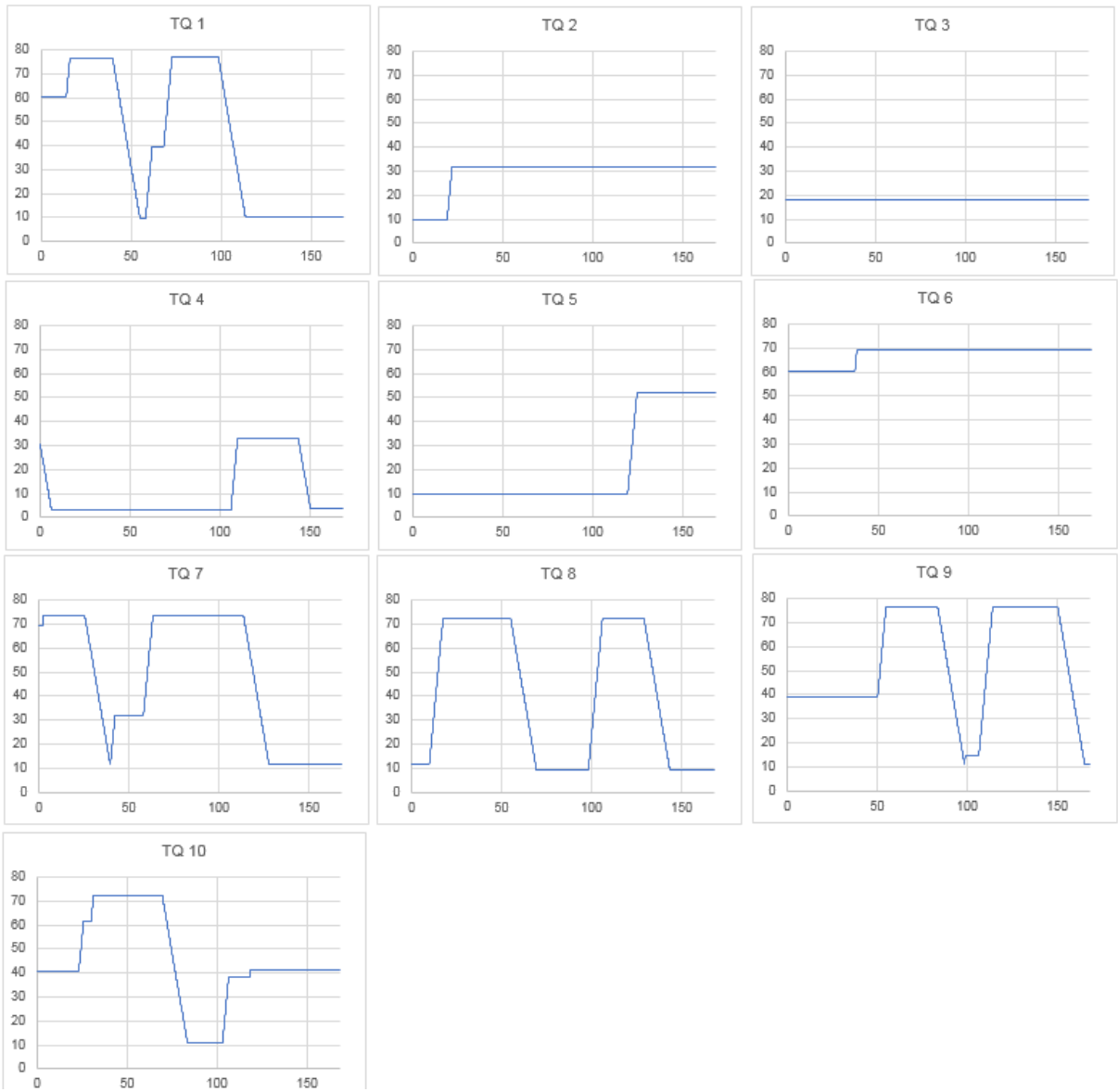
Figura 47 - Resultado do descarregamento dos navios



O volume de cada tanque, em mil m^3, em função do tempo, em horas, é visualizado na Figura 48. Analisando a variação dos níveis do tanque (Figura 48), nota-se algumas possíveis desotimizações: o tanque 3 não foi utilizado em nenhum momento, permanecendo vazio durante todo horizonte e os tanques 2 e 6 encheram até seus limites de capacidades, mas não descarregaram no oleoduto. Para confirmar se esse resultado foi realmente fruto de uma desotimização, é necessário

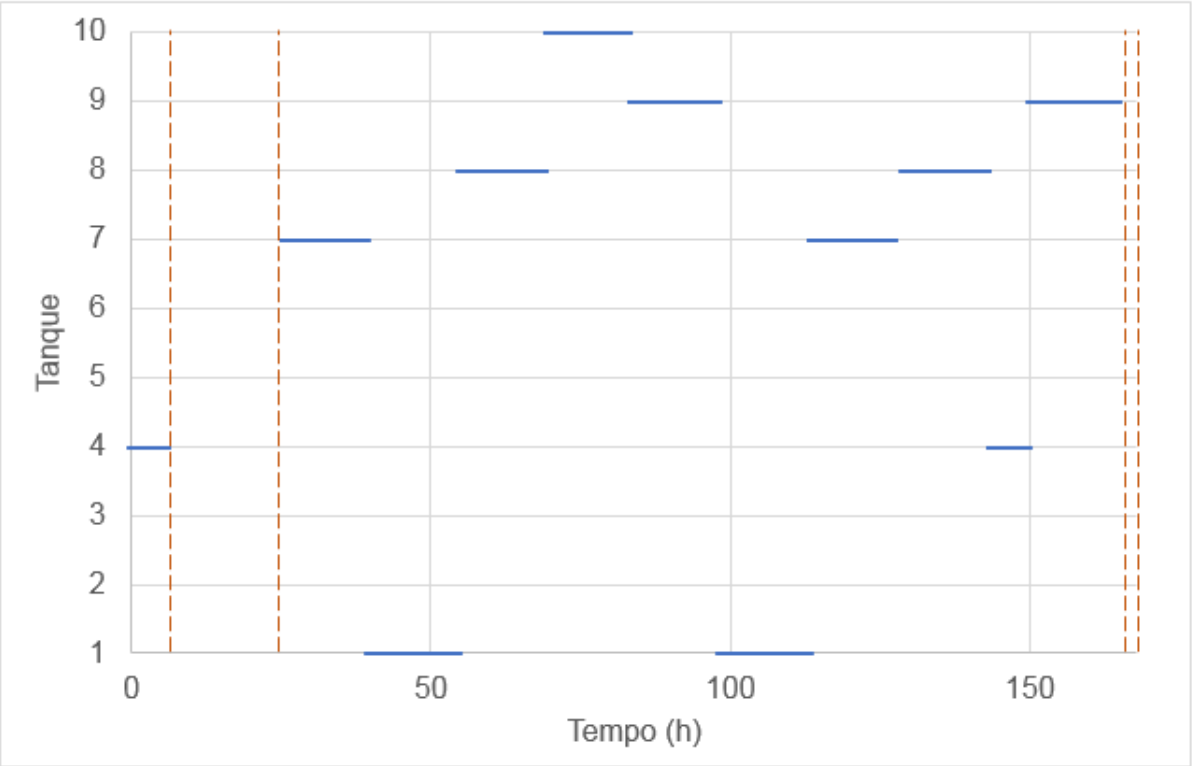
checar a utilização do oleoduto, ou seja, a porcentagem do tempo em que o oleoduto estava operando, mandando o petróleo para a refinaria.

Figura 48 - Nível dos tanques em função do tempo



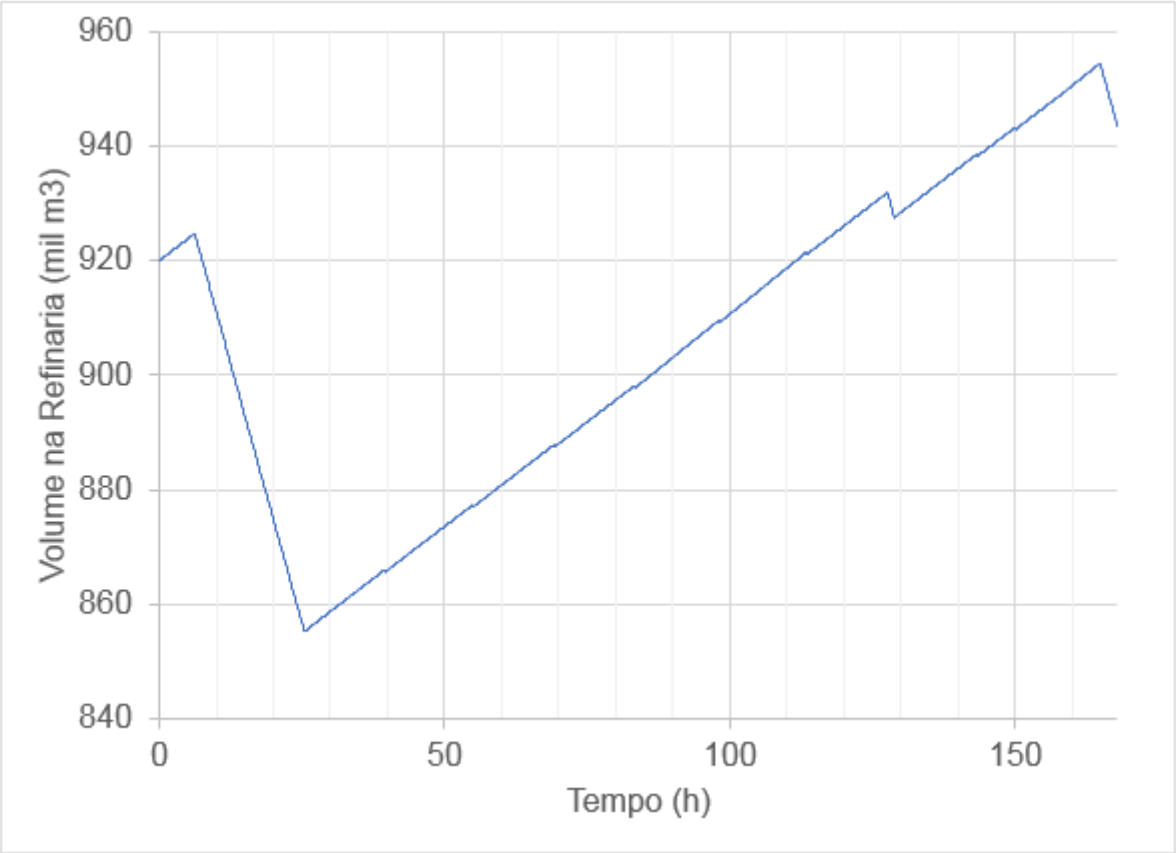
A Figura 49 mostra os intervalos de tempo em que cada tanque (representados pelos números de 1 a 10 no eixo y do gráfico) envia petróleo para o oleoduto. As linhas tracejadas delimitam o intervalo de tempo em que o oleoduto fica ocioso, ou seja, nem recebe nem envia petróleo para a refinaria, pois não há tanque enviando. A utilização do oleoduto é de 86%, isto é, ele fica ocioso 14% do horizonte de tempo.

Figura 49 - Envio dos tanques para o oleoduto



Os níveis mínimos e máximos de petróleo permitidos na refinaria do problema são de 186 e 1516 mil m^3, respectivamente e na Figura 50, observa-se que os limites foram respeitados, mas que, apesar da vazão do oleoduto ser capaz de manter os níveis da refinaria subindo durante o horizonte, houveram momentos de queda. Lembrando que o problema considera consumo constante de petróleo pela refinaria de 3,625 mil m^3 por hora.

Figura 50 - Volume de petróleo estocado na refinaria em função do tempo



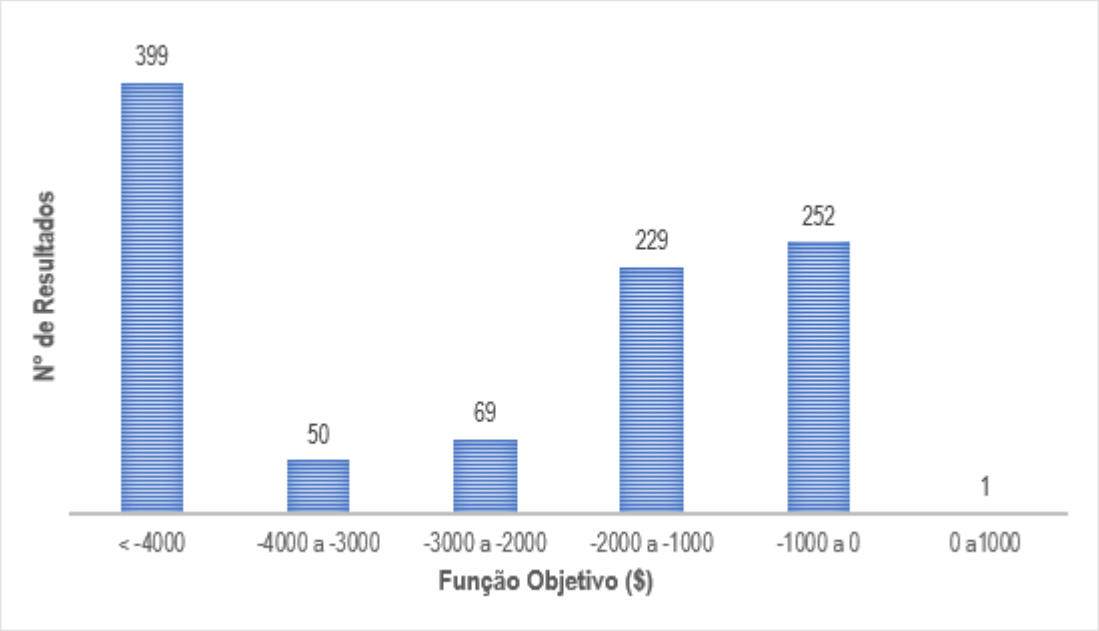
8.3.1 DESEMPENHO DO ACO NO ESTUDO DE CASO 2

A parametrização do estudo de caso 2, utilizando GA, demonstrou-se inviável devido ao grande dispêndio de tempo sem resultado; após o programa ficar 1 semana rodando, a otimização foi interrompida. Assim, esse problema apresenta desvantagem de não estar munido de uma boa parametrização devido a essa limitação.

Más (2001) encontrou o valor de função objetivo, no modelo inteiro misto, em torno de $14000. Para esse mesmo problema, utilizando ACO, utilizando uma pequena quantidade de formigas (de 30 a 80 por iteração), o modelo não estava conseguindo atingir bons resultados, inclusive a grande maioria dos valores de FO encontrados eram negativos. Isso demonstra que o estudo de caso 2 é um problema que tem grandes chances de dar prejuízo, já que a grande maioria dos resultados originam um valor negativo de diferença entre receitas e custo.

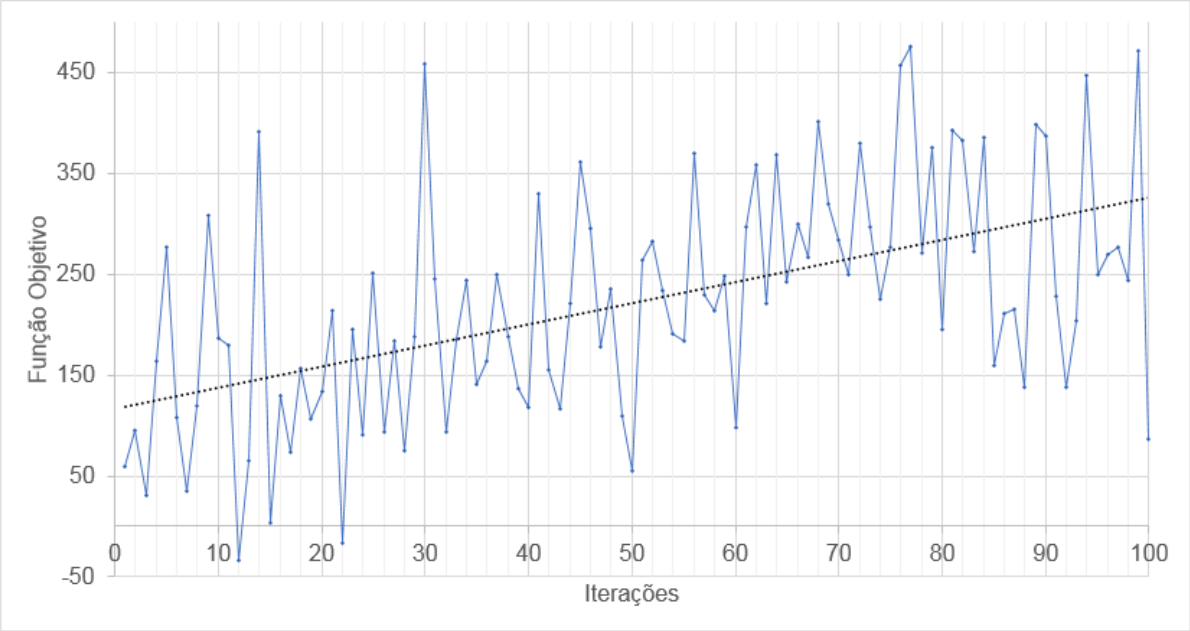
Calculando a função objetivo 1000 vezes, análogo a fazer apenas 1 iteração do algoritmo utilizando 1000 formigas, obtemos o resultado da Figura 51. Pode-se observar que, sem o artifício de atualização dos feromônios do ACO, ou seja, apenas utilizando a aleatoriedade, a probabilidade de obter um valor positivo de FO é muito baixa, nesse caso o resultado foi de 1 em 1000.

Figura 51 - Resultado da função objetivo do estudo de caso 2



Utilizando um número maior de formigas que o estudo de caso 1 (no estudo de caso 2 foram utilizadas 1000 formigas por iteração) e mantendo os outros parâmetros, foi possível perceber uma grande melhora no desempenho do ACO, como pode ser visualizado na Figura 52. Contudo, para 100 iterações com essa grande quantidade formigas, o programa necessita de mais de 2 dias para executar.

Figura 52 - Resultado do ACO para o estudo de caso 2



Apesar do valor da FO do estudo de caso 2 não ter atingido valores muito altos, percebe-se uma melhora no resultado ao longo das iterações e a presença de poucos valores negativos.

9. CONCLUSÃO

O Algoritmo de Colônia de formigas é uma metaheurística que já foi aplicada com sucesso a uma grande variedade de problemas, inclusive problemas de *scheduling*. Os principais pontos fortes desse algoritmo são a facilidade de adaptação para diversos tipos de problema e a velocidade de convergência. O principal ponto fraco é a grande dependência de uma boa parametrização e sua especificidade.

O uso de programação orientada objeto para modelagem de problemas com um grande número de variáveis, mostrou-se vantajosa, principalmente na facilidade de entendimento do código, ao diminuir a necessidade de criação de novas variáveis ou de matrizes com mais de 3 dimensões. Os modelos simulados demonstraram-se extremamente sensíveis aos valores dos parâmetros do algoritmo, principalmente da constante de atualização do feromônio, demonstrando a importância de sua calibração. A limitação de hardware foi o maior impeditivo para implementação de um modelo mais robusto, já que a parametrização, que é essencial para um desempenho satisfatório do ACO, exige uma quantidade exaustiva de rodadas do algoritmo demandando grande tempo computacional.

No capítulo 6, foram mostradas três diferentes aplicações do algoritmo de colônia de formiga utilizando problemas distintos em complexidade e estrutura, que foram o problema de Ponte Dupla, do Caixeiro Viajante e um de *scheduling* em uma máquina. O primeiro é de simples convergência e possui uma heurística bastante intuitiva; sua parametrização revelou que até os mais simples dos problemas, se mal parametrizados, podem não alcançar bons resultados. Já o problema do caixeiro viajante é um benchmark para análise de algoritmos, ele é NP-hard, mas possui uma heurística já bem validada e vários níveis de complexidade dependendo, principalmente, do número de cidades utilizadas. Sua parametrização demonstrou-se sensível e, apesar de apresentar uma convergência na maioria dos casos, muitas vezes convergia a ótimos locais. O problema de *scheduling* estudado, apesar de simples quando comparado aos estudos de caso, não atingiu o ótimo global, pois, além de ter a dificuldade da elaboração de uma heurística adequada, não houve empenho para a parametrização estruturada do ACO e, possivelmente, o número de formigas e iterações limitou a performance do algoritmo.

O problema de alocação de petróleo está presente em diversas etapas do planejamento e tomada de decisões da cadeia de petróleo, sendo por isso alvo de diferentes abordagens gerenciais e matemáticas dependendo do âmbito inserido. Este trabalho teve como foco as decisões de curto prazo, que são caracterizadas como julgamentos rápidos que priorizam agilidade em encontrar soluções viáveis à descoberta de ótimos globais, que é o foco de planejamentos de médio e longo prazo. Assim, foram analisados dois estudos de caso que modelam o *scheduling* de petróleo do navio até as refinarias REPLAN e REVAP através do terminal de São Sebastião. Os estudos de caso se diferenciaram pelo grau de complexidade e o desempenho do ACO foi bastante influenciado nesse sentido. Para os dois estudos, a parametrização foi realizada preliminarmente como uma sensibilidade simples e os parâmetros que se mostraram mais significativos foram submetidos a uma parametrização utilizando Algoritmo Genético. Partindo dos parâmetros encontrados, o primeiro estudo de caso, mais simples, atingiu bons resultados, mas, para tal, foi necessário o uso de um número alto de formigas, resultando em grande dispêndio de processamento computacional. Já o segundo estudo de caso apresentou melhora na função objetivo ao longo das interações, mas a limitada quantidade de formigas e iterações provenientes de falta de capacidade de processamento não permitiu que o resultado chegasse próximo do ótimo.

10. RECOMENDAÇÕES DE PESQUISA FUTURA

Utilizar o modelo de *scheduling* desenvolvido neste trabalho para estudar sistemas mais complexos de alocação ou de outras parcelas da cadeia de suprimento.

Realizar a parametrização de modo mais preciso munido uma capacidade de processamento mais adequada para essa complexidade de problema.

Foi escolhido aqui um caso de otimização onde a programação linear inteira mista (MILP) está bem resolvida, e principalmente, a formulação do problema está muito bem elaborada e bem descrita para MILP, de tal modo que ACO gere soluções subótimas em relação a MILP. No futuro deve-se prospectar casos que sejam de difícil definição do problema ou que sejam *NP-hard* (*nonpolynomial in time*) em otimização combinatorial onde MILP encontra desafios.

Ainda, os algoritmos metaheurísticos e computação evolucionária para casos de mais de 3 dimensões sofre severas restrições computacionais. Com o aumento da facilidade de computação paralela (HPC – *High Performance Computing*) será possível analisar melhor a diferença entre a escolha de parâmetros (chamados "*toy problems*") em termos de sensibilidade, hibridizar métodos, analisar vizinhança da busca, diversificação (*exploration*), intensificação (*explotation*) da busca, pesquisa local e global, métodos gerais de escape de mínimos locais, influência da população inicial e pontos de partida, e ainda mais, explorar regiões inapropriadas, não promissoras ou até mesmo fora da região sob restrições do problema (que gere agentes mais promissores). Por ser algoritmo metaheurístico, também será possível explorar problemas de otimização multiobjetivo diretamente como compromisso de vários objetivos independentes, fugindo de ponderação de vários objetivos com pesos que resultem em uma função objetiva única como em NLIMP. Essas são as poucas possibilidades de uma lista maior que diferencia método heurístico de NLIMP em *scheduling*, onde quase não havia literatura de aplicação de ACO.

Finalmente, por ser um método computacional heurístico, deve haver uma análise do que é uma "convergência estocástica", mas tal análise está além de presente trabalho preliminar de exploração do uso de ACO em *scheduling*.

REFERÊNCIAS

ALOISE, D.; NORONHA, T. F. DE; MAIA, R. DA S.; BITTENCOURT, V. G.; ALOISE, D. J. **Heurísticas de colônia de formigas com path-relinking para o problema de otimização da alocação de sondas de produção terrestre–SPT**. XXXIV SBPO Simpósio Brasileiro de Pesquisa Operacional, 2002.

ALVES, D. R. DE S.; SERRA NETO, M. T. R.; FERREIRA, F. DOS S.; TEIXEIRA, O. N. **SIACO: A Novel Algorithm Based on Ant Colony Optimization and Game Theory for Travelling Salesman Problem** Proceedings of the 2nd International Conference on Machine Learning and Soft Computing. **Anais** ACM. p. 62-66. 2018.

ANP. **Anuário Estatístico 2016**. Disponível em: <http://www.anp.gov.br/wwwanp/publicacoes/anuario-estatistico/2441-anuario-estatistico-2016#Seção 2 ¿ Indústria Nacional do Petróleo e Gás Natural>. Acesso em: 20 nov. 2017.

BAZGOSHA, A.; RANJBAR, M.; JAMILI, N. **Scheduling of loading and unloading operations in a multi stations transshipment terminal with release date and inventory constraints**. Computers and Industrial Engineering, v. 106, p. 20–31, 2017.

BEDI, M. K.; SINGH, S. **Comparative Study of Two Natural Phenomena Based Optimization Techniques**. International Journal of Scientific & Engineering Research v. 4, p. 1–4, 2013.

CECILIA, J. M.; LLANES, A.; ABELLÁN, J. L.; GÓMEZ-LUNA, J.; CHANG, L. W.; HWU, W. M. W. **High-throughput Ant Colony Optimization on graphics processing units**. Journal of Parallel and Distributed Computing, v. 113, p. 261–274, 2018.

CHATAVITHEE, P.; PIEWTHONGNGAM, K.; PATHUMNAKUL, S. **Scheduling a single machine with concurrent jobs for the frozen food industry**. Computers and Industrial Engineering, v. 90, p. 158–166, 2015.

COLORNI, A., DORIGO, M., & MANIEZZO, V. **Distributed optimization by ant colonies.** Proceedings of the First European Conference on Artificial Life, p. 134–142, 1992.

COLORNI, A.; DORIGO, M.; MANIEZZO, V.; TRUBIAN, M. **Ant System for Job-shop Scheduling**. Belgian Journal of Operations Research, Statistics and Computer Science, v. 34, n. 1, p. 39–53, 1994.

DELGOSHAEI, A.; DELGOSHAEI, A.; ARAM, A. K.; ALI, A. **A robust optimization approach for scheduling a supply chain system considering preventive maintenance and emergency services using a hybrid ant colony optimization and simulated annealing algorithm.** Uncertain Supply Chain Management, v. 7, p. 251–274, 2019.

DIOS, M.; FRAMINAN, J. M. **A review and classification of computer-based manufacturing scheduling tools**. Computers and Industrial Engineering, v. 99, p. 229–249, 2016.

DORIGO, M. Optimization, **Learning and Natural Algorithms** [in Italian]. PhD Thesis, Politecnico di Milano, 1992.

DORIGO, M.; BLUM, C. **Ant colony optimization theory: A survey**. Theoretical Computer Science, v. 344, n. 2–3, p. 243–278, 2005.

DORIGO, M.; GAMBARDELLA, L. M. **Ant colony system: a cooperative learning approach to the traveling salesman problem**. IEEE Transactions on Evolutionary Computation, v. 1, n. 1, p. 53–66, 1997.

DORIGO, M.; STÜTZLE, T. **Ant Colony Optimization**. MIT Press 2004.

FARAH, M. A. **Petróleo e Seus Derivados**. RIo de Janeiro: LTC, 2012.

GHAITH RABADI. **Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling**. v. 236. Springer. 2016.

GOGNA, A.; TAYAL, A. **Metaheuristics : review and application.** Journal of Experimental & Theoretical Artificial Intelligence, [s. l.], v. 25, n. 4, p. 503–526, 2013

HUSSAIN, K. et al. **Metaheuristic research : a comprehensive survey**. Artificial Intelligence Review, [s. l.], 2018.

KALRA, M.; SINGH, S. **A review of metaheuristic scheduling techniques in cloud computing**. Egyptian Informatics Journal, v. 16, n. 3, p. 275–295, 2015.

KAR, A. K. Bio inspired computing – **A review of algorithms and scope of applications.** Expert Systems With Applications, v. 59, p. 20–32, 2016.

KUMAR, K. M. S.; SELLADURAI, V.; RAJA, K.; ELAGOVAN, K. **Ant Colony Approach for Makespan Minimization on Unrelated Parallel Machines**. International Journal of Engineering Science and Technology (IJEST) v. 3, n. 4, p. 3113–3120, 2011.

LIZA, J. C. **Programação de producao de plantas multiprodutos atraves de tecnicas de otimizacao mista-inteira.** Tese de Doutorado. Universidade de São Paulo (USP), 2005.

LUAN, J.; YAO, Z.; ZHAO, F.; SONG, X. **A novel method to solve supplier selection problem: Hybrid algorithm of genetic algorithm and ant colony optimization**. Mathematics and Computers in Simulation, v. 156, p. 294–309, 2019.

MARCELLINO, F. J. DE M. **Planejamento Integrado da Cadeia de Suprimentos da Indústria do Petróleo Baseado em Agentes Holônicos.** Tese de Doutorado. Universidade de São Paulo (USP), 2013.

MARINS, F. A. S. **Introdução à pesquisa operacional.** Sáo Paulo: *Cultura Acadêmica: Universidade Estadual Paulista*. 2011.

MARTINS, M. S. R.; LÜDERS, R.; DELGADO, M. R. **ANÁLISE DA OTIMIZAÇÃO COM SIMULAÇÃO DA TRANSFERÊNCIA DE PETRÓLEO EM COMPLEXOS PORTUÁRIOS**. XLI SBPO 2009 - Pesquisa Operacional na Gestão do Conhecimento, p. 1039–1050, 2009.

MÁS, R. **Otimização da Programação de Suprimento de Petróleo**.Tese de Doutorado. Universidade de São Paulo (USP), 2001.

MAS, R.; M PINTO, J. E. **A Mixed-Integer Optimization Strategy for Oil Supply in Distribution Complexes**. Optimization and Engineering, v. 4, p. 23–64, 2003.

MONMARCHÉ, N.; GUINAND, F.; SIARRY, P. **Artificial Ants - From Collective Intelligence o Real-life Optimization and Beyond**. Londres: Wiley-ISTE, 2010.

MORO, L. F. L. **Técnicas de otimização mista-inteira para o planejamento e programação de produção em refinarias de petróleo**. Tese de Doutorado. Universidade de São Paulo (USP), 2000.

MULLEN, R. J.; MONEKOSSO, D.; BARMAN, S.; REMAGNINO, P. **A review of ant algorithms**. Expert Systems With Applications, v. 36, n. 6, p. 9608–9617, 2009.

NEIRO, S. M. DA S. **Planejamento integrado de produção-distribuição de cadeias de suprimentos petrolífera**. Tese de Doutorado. Universidade de São Paulo (USP), 2003.

NETO, R. F. T.; FILHO, M. G. **Otimização por colônia de formigas para o problema de sequenciamento de tarefas em uma única máquina com terceirização permitida.** Gest. Prod., v. 20, n. 2000, p. 76–86, 2013.

NUNES, P. M.; OLIVEIRA, F.; HAMACHER, S.; HAMACHER, P.; TEIXEIRA, W.; MUNCK, F. **ANÁLISE DO PLANEJAMENTO DE ABASTECIMENTO DA CADEIA DE PETRÓLEO NO BRASIL**. Rio Oil & Gas, p. 1–10, 2010.

OLIVEIRA, F.; HAMACHER, S. **DESENVOLVIMENTO DE UMA FERRAMENTA GRÁFICA PARA SISTEMAS DE OTIMIZAÇÂO DO ABASTECIMENTO DE PETRÓLEO E DERIVADOS**. XXXIX SBPQ A Pesquisa Operacional e o Desenvolvimento Sustentável. Anais. 2007.

OLIVEIRA, F.; HAMACHER, S.; ALMEIDA, M. R. **Process industry scheduling optimization using genetic algorithm and mathematical programming**. Journal of Intelligent Manufacturing, v. 22, n. 5, p. 801–813, 2009.

PINTO, J. M. **Planejamento e programação de operações de producao e distribuicao em refinarias de petroleo.** Tese de Livre-Docente. Universidade de São Paulo (USP), 2000.

PIRES, S. R. I. **Gestao da Cadeia de Suprimentos**. Segunda ed. Sao Paulo: GEn-Atlas. 2016.

RAGHAVENDRA, G. S.; PRASANNA, K. N. **Cluster Integrated Updation Strategies for ACO Algorithms**. International Journal of Computer Applications, v. 30, n. 2, p. 18–24, 2011.

RANGAIAH, G. P. **Stochastic Global Optimization - Techniques and Applications in Chemical Engineering**. Singapura: World Scientific Publishing, 2010.

REINELT, G. **TSPLIB - A Traveling Salesman Problem Library**. ORSA Journal on Computing, v. 3, n. 4, p. 376–384, 1991.

SHIMIZU, T. **Decisão nas Organizações**. Terceira ed. São Paulo: Atlas. 2010.

SILVA, C. A.; SOUSA, J. M. C.; RUNKLER, T. A. **Rescheduling and optimization of logistic processes using GA and ACO**. Engineering Applications of Artificial Intelligence, v. 21, n. 3, p. 343–352, 2008.

SIMON, D. **EVOLUTIONARY OPTIMIZATION ALGORITHMS Biologically-Inspired and Population-Based Approaches to Computer Intelligence**. John Wiley & Sons, 2013.

SOCHA, K.; DORIGO, M. **Ant colony optimization for continuous domains**. European Journal of Operational Research, v. 185, n. 3, p. 1155–1173, 2008.

SOLNON, C. **Ant Colony Optimization and Constraint Programming**. Londres: Wiley, 2010.

SRINIVAS, P. S.; RAJU, V. R.; RAO, C. S. P. **Optimization of Process Planning and Scheduling using ACO and PSO Algorithms**, v. 2, n. 10, p. 343–354, 2012.

TAVARES NETO, R. F.; GODINHO FILHO, M. **Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research.** Engineering Applications of Artificial Intelligence, v. 26, n. 1, p. 150–161, 2013.

THE MATHWORKS, I. **MATLAB® The Language of Technical Computing**. 6. ed. Natick: 2000.

WEISE, T. **Global Optimization Algorithms – Theory and Application.** Segunda Edition. University of Kassel.-2008.

WOLPERT, D. H.; MACREADY, W. G. **No Free Lunch Theorems for Optimization**. Ieee Transactions on Evolutionary Computation, v. 1, n. 1, p. 67–82, 1997.

WU, N.; ZHOU, M.; CHU, F. **Short-term Scheduling for Refinery Process : Bridging the Gap between Theory and Applications**. v. 10, n. 2, p. 162–174, 2005.

XU, J.; ZHANG, S.; ZHANG, J.; WANG, S.; XU, Q. **Simultaneous scheduling of front-end crude transfer and refinery processing.** Computers and Chemical Engineering, v. 96, p. 212–236, 2017.

APÊNDICE

APÊNDICE A - CÓDIGO MATLAB® PARA A PONTE DUPLA

O código completo do experimento da ponte dupla é apresentado a seguir. E é divido em três programas: o programa principal e duas funções, a função 1 calcula as probabilidades e a função 2 é a roleta que escolhe o resultado baseado nas probabilidades calculadas na função 1.

Programa principal:

```
%Distâncias
d(1) = 1;
d(2) = 1;

%Heurística
Heu = [1/d(1) 1/d(2)]; %Caminhos iguais
imax = 1;
jmax = 2;
n = 100; %iterações
ants = 100; %quantidade de formigas por iteração
fer_atu = 0.3;
alfa = 3;
beta = 1;
fer0 = 0.1;

exp = 100;
branch1 = zeros(1,5);

for k =1:exp
fer = [fer0 fer0];
clear FO
clear x

for  ii=1:n
    fer_save(ii,:) = fer;
    for m=1:ants
        [x(m), prob_x ] = prob_gen( imax, jmax, fer, alfa, beta, Heu);
        FO(m) = 1/d(x(m));
        [~,bestsoFar] = find(FO == max(FO));

        FO_ant(ii,m) = FO(m);
        x_ant(ii,m) = x(m);
    end
    %Evaporação
    %fer = fer*(1 - fer_evap);

    %Atualização
    fer(x(bestsoFar(1))) = fer(x(bestsoFar(1))) + fer_atu;
    FO_best(ii) = FO(bestsoFar(1));
end
```

```
for i=1:n
    cam(i,1) = length(find(x_ant(i,:)==1));
    cam(i,2) = length(find(x_ant(i,:)==2));
end

trafego1 = cam(n,1)/ants; %Trafego no caminho 1
trafego2 = cam(n,2)/ants; %Trafego no caminho 2
%histogram(trafego1)
%plot(1:n, trafego1)%, 1:n, trafego2)
%plot(1:n, fer_save(:,1), 1:n, fer_save(:,2))

if trafego1(end)< 0.2
    branch1(1,1) = branch1(1,1) + 1;
end
if trafego1(end)>= 0.2 && trafego1(end)< 0.4
    branch1(1,2) = branch1(1,2) + 1;
end
if trafego1(end)>= 0.4 && trafego1(end)< 0.6
    branch1(1,3) = branch1(1,3) + 1;
end
if trafego1(end)>= 0.6 && trafego1(end)< 0.8
    branch1(1,4) = branch1(1,4) + 1;
end
if trafego1(end)>= 0.8
    branch1(1,5) = branch1(1,5) + 1;
end

end
branch1 %Quantidade de formigas que escolheram o caminho 1
```

Função 1:

```
function [ var, prob ] = prob_gen( imax, jmax, fer, alfa, beta, Heu)
%Calcula as probablidades e o resultado da variável decisória

soma = zeros(imax,1);
for i=1:imax
    for j=1:jmax
        soma(i) = soma(i) + fer(i,j)^alfa*Heu(i,j)^beta;
    end
end

prob = zeros(imax,jmax);%inicializando variável

for i=1:imax
    for j=1:jmax
        prob(i,j) = (fer(i,j)^alfa*Heu(i,j)^beta)/soma(i);
    end
end

var = zeros(1,imax);
for i=1:imax
    var(i) = roleta(prob(i,:));
end

end
```

Função 2:

```
function f = roleta(prob)
    r = rand;
    C = cumsum(prob);
    f = find(r <= C, 1, 'first');
end
```

APÊNDICE B - CÓDIGO MATLAB® PARA TSP

Dados de entrada (matriz de coordenadas entre as 42 cidades):

Cidade	x	y
1	170	85
2	166	88
3	133	73
4	140	70
5	142	55
6	126	53
7	125	60
8	119	68
9	117	74
10	99	83
11	73	79
12	72	91
13	37	94
14	6	106
15	3	97
16	21	82
17	33	67
18	4	66
19	3	42
20	27	33
21	52	41
22	57	59
23	58	66
24	88	65
25	99	67
26	95	55
27	89	55
28	83	38
29	85	25
30	104	35
31	112	37

32	112	24
33	113	13
34	125	30
35	135	32
36	147	18
37	147,5	36
38	154,5	45
39	157	54
40	158	61
41	172	82
42	174	87

Código Matlab

```
function Result = TSP_ex
% clc
% clear all

D = importdata('dantzig42_d.txt');
Dsize = length(D);
%C = importdata('coordenadas.txt');
%plot(CC(:,1),CC(:,2),'o')

n = 200; %iterações
ants = 1000; %quantidade de formigas por iteração
fer_evap = 0.06;
alfa = 0.1;
beta = 4;
fer_atu = 1;

fer0 = ants/900;
fer(1:Dsize,1:Dsize) = fer0;

imax = 1;
jmax = Dsize;
z_it = zeros(n,Dsize);
for  ii=1:n
    for m=1:ants
        %Heurística
        Heu = 1./D; %Caminhos iguais
        Heu(1:1+size(Heu,1):end) = 0; %substitui os valores
ifinitos por zero

        z(m,1) = 1;
        Heu(:,1) = 0;
        [z(m,2), prob_x(m,:) ] = prob_gen( imax, jmax,
fer(1,:), alfa, beta, Heu(1,:)); %A primeira cidade é a 1
        Heu(:,z(m,2)) = 0;
```

```
        for k=3:Dsize
            [z(m,k), prob_x(m,:) ] = prob_gen( imax, jmax,
fer(z(m,k-1),:), alfa, beta, Heu(z(m,k-1),:));
            Heu(:,z(m,k)) = 0;
        end

        FO(m) = 0;
        for i=2:Dsize
            FO(m) = FO(m) + D(z(m,i-1),z(m,i));
        end
        FO(m) = FO(m) + D(z(m,Dsize),z(m,1));

        [~,bestsoFar] = find(FO == min(FO));
    end
    z_it(ii,:) = z(bestsoFar(1),:);
    z_bin = zeros(Dsize); %z binário
    for i = 1:Dsize-1
        z_bin(z(bestsoFar(1),i),z(bestsoFar(1),i+1)) = 1;
    end
    z_bin(z(bestsoFar(1),Dsize),1) = 1;

    %Evaporação
    fer = fer*(1 - fer_evap);

    %Atualização
    fer = fer + fer_atu/FO(bestsoFar(1))*z_bin;
    FO_best(ii) = FO(bestsoFar(1));
end

plot(FO_best)
xlabel('Iterações')
ylabel('Função Objetivo')

Result = FO_best(end);

end
```

APÊNDICE C - DADOS DE ENTRADA DOS ESTUDOS DE CASO

Os dados de entrada do estudo de caso alocação de petróleo são mostrados a seguir.

Navio	*n.H0*	*n.Hsob*	*n.Csob*	*n.Qmin*	*n.Qmax*
N1	0	48	0.8333	0	8
N2	12	60	0.4167	0	8

N3	12	60	0.4168	0	8
N4	24	72	0.4167	0	8

Píer	***pier.Custo***
p1	5.0314
p2	2.5157

n.pet	**c5**	**c8**	**c27**	**c38**
N1	106	0	0	0
N2	0	38	0	0
N3	0	0	20	0
N4	0	0	0	30

tq	***tq.V*min**	***tq.V*max**	***tq.V*0**	**Grupamento**	***o.Cl.Qmax***
t1	9.766	77.329	50.63	1	4.47
t2	10.457	73.077	18.232	3	4.39
t3	9.654	77.355	49.997	4	4.429
t4	11.37	73.745	69.524	5	4.447
t5	11.022	77.001	39.015	6	4.511

Refinaria	***r.V0***	***R.Vmin***	***R.Vmax***	***r.consumo***
REVAP	920	186.06	1515.487	3.625

Grupamento	***r.cl.valor***	***tq.cl.valor***
cl1	135.2201	129.8113
cl3	132.0755	126.7925
cl4	133.9622	128.6037
cl5	138.3648	132.8302
cl6	137.1069	131.6227

Petróleo	**pet.custo**
c5	116.9811
c8	109.7484
c27	100.6289
c38	114.7798

Interface	**cl1**	**cl3**	**cl4**	**cl5**	**cl6**
cl1	0	1.57235	0.62895	1.57235	0.9434

cl3	1.57235	0	0.94335	3.14465	2.5157
cl4	0.62895	0.94335	0	2.2013	1.57235
cl5	1.57235	3.14465	2.2013	0	0.62895
cl6	0.9434	2.5157	1.57235	0.62895	0

n.Tent	**p1**	**p2**
N1	2	2
N2	2	2
N3	0	2
N4	2	2

n.Tsai	**p1**	**p2**
N1	2	2
N2	2	0
N3	2	2
N4	2	2

APÊNDICE D - CÓDIGO MATLAB® PARA OS ESTUDOS DE CASO

A modelagem no Matlab® foi dividida em dois programas um programa principal, no qual são definidos os valores dos parâmetros de otimização do algoritmo e uma sub-rotina que aplica o algoritmo de colônia de formigas ao sistema proposto. Os dados (variáveis de entrada) do problema são carregados a partir de arquivos de texto (extensão .txt), que foram apresentados em tabelas no Apêndice C.

APÊNDICE D.1 – CÓDIGO DOS OBJETOS DO ESTUDO DE CASO

APÊNDICE D.1.1 – CÓDIGO DO OBJETO: PET

```
classdef pet < handle
    %Petróleo
    properties
        CL %grupamento do petroleo
        Custo %custo do petróleo
        classe %grupamento do petroleo
    end

    methods
      function petclasse = get.classe(obj)
         petclasse = obj.classe;
```

```
      end

    end
end
```

APÊNDICE D.1.2 – CÓDIGO DO OBJETO: CLASSE

```
classdef classe < handle
    %Classe do petróleo
    properties
        Valor_tq %valor da classe no tanque
        Valor_ref %%valor da classe na refinaria
        Interf %Valor da inerface com outra classe
        Qmax %Máxima vazão do grupamento cl no duto o
    end
end
```

APÊNDICE D.1.3 – CÓDIGO DO OBJETO: NAVIO

```
classdef navio < handle
    %NAVIO
    properties
       H0 %Hora de chegada do navio em horas
       Hsob %Hora máxima p/ não caracterizar sobrestadia
       Hent %Hora do início da atracação
       Hftq %Hora do fim do bombeamento para o tanque
       Hf = 0; %Hora da saída do navio do píer
       Csob %Custo de sobrestadia [$/h]
       Qmin %Vazão mínima de descarregamento do navio n
       Qmax %Vazão máxima de descarregamento do navio n
       pet %Petróleo contido no navio
       V %volume de petróleo em mil m3
       V0 %volume inicial de petróleo em mil m3
       saveV %Guarda os valores de V para controle e
elaboração de gráficos
       pier %pier associado ao navio
       Tent = 2; %Intervalo de tempo necessário para a
atracação no navio no píer
       Tsai = 2; %Intervalo de tempo necessário para a
desatracação no navio no píer
        st = 1 %status do navio:
        %1-Indisponivel
        %2-Disponivel para atracar
        %3-Atracando
```

```
        %4-Descarregando
        %5-Desatracando

    end

    methods
        function n = navio(h0, hsob, c, qmin, qmax, v, savev,
p)
            if nargin == 8
                n.H0 = h0;
                n.Hsob = hsob;
                n.Csob = c;
                n.Qmin = qmin;
                n.Qmax = qmax;
                n.V = v;
                n.saveV = savev;
                n.pier = p;
            end
        end

        function addV(n,  newV, newst)
            n.V = n.V + newV;
            n.saveV(length(n.saveV)+1) = n.V;
            n.st = newst;
        end

        function mude_st(n,  newst)
            n.st = newst;
            [x, ~] = size(n.saveV);
            n.saveV((x+1), :) = n.V;
        end
        function r = csob(n) %Custo de sobrestadia [$]
            if n.Hf <= n.Hsob
                r = 0;
            else
                r = (n.Hf - n.Hsob)*n.Csob;
            end
        end
        %____
        function petnavio = get.pet(obj)
            petnavio = obj.pet;
        end
        function piernavio = get.pier(obj)
            piernavio = obj.pier;
        end
        %____
        function r = H0tq(n)
            r = n.Hent + n.Tent;
        end
        function r = schd(obj)
```

```
            r = [obj.H0 obj.Hent obj.H0tq obj.Hftq obj.Hf];
        end
        function r = tpier(n)%tempo que o navio ocupa o píer
            if n.Hf ~= 0 %se o navio deixou o pier
                r = n.Hf - n.Hent;
            else
                r = 0;
            end
        end
    end
end
```

APÊNDICE D.1.4 – CÓDIGO DO OBJETO: PIER

```
classdef pier < handle
    %PIER
    properties
       Custo %Custo de utilização do píer [$/h]
       st = 1 %status do pier:
        %1-Disponivel
        %2-Indisponivel
    end
end
```

APÊNDICE D.1.5 – CÓDIGO DO OBJETO: TQ

```
classdef tq < handle
    %TQ : Tanque
    properties
       Vmin %Capacidade mínima de armazenamento do tanque t
       Vmax %Capacidade máxima de armazenamento do tanque t
       V %Volume atualmente preenchido do tanque
       V0 %Volume inicial preenchido do tanque
       saveV %Guarda os valores de V para controle e
elaboração de gráficos
       Cl %grupamento do petroleo
       Qmax %Vazão máxima
       H_switch = -1 %Hora em que o tq muda p/ st=1. Evita
navio enviando p/tq junto com tq enviando p/ oleod
       R %Preço por unidade de volume do grupamento de
petróleo no tanque do terminal
       st = 1 %status do tanque:
                %1-carregando
                %2-settling (min.24h p/ começar a descarregar)
                %3-pronto para descarregar
```

```
                %4-descarregando
        pet = 0; %Identifica o petroleo no tanque.
        Hset = 0; %Hora que começou o settling
    end

    methods
        function tanque = tq(vmin, vmax, v, sv, cl, qmax,
s)%constructor
            if nargin == 9
                tanque.Vmin = vmin;
                tanque.Vmax = vmax;
                tanque.V = v;
                tanque.saveV = sv;
                tanque.Cl = cl;
                tanque.Qmax = qmax;
                tanque.st = s;
            end
        end

        function addV(tanque,  newV, petroleo, newst)
            tanque.V = tanque.V + newV;
            tanque.saveV((length(tanque.saveV)+1)) = tanque.V;
            tanque.pet = petroleo;
            tanque.st = newst;
        end

        function mude_st(tanque,  newst)
            tanque.st = newst;
            tanque.saveV((length(tanque.saveV)+1)) = tanque.V;
        end

        function sendV(tanque,  deltaV)
            tanque.V = tanque.V - deltaV;
            tanque.saveV((length(tanque.saveV)+1)) = tanque.V;
        end

        function r = Vo(tanque) %Volume possível de ser
descarregado
            r = tanque.V - tanque.Vmin;
        end
    end

end
```

APÊNDICE D.1.6 – CÓDIGO DO OBJETO: OLEOD

```
classdef oleod < handle
```

```
    %OLEODUTO
    properties
        V %vetor de volumes na sequencia que está no oleoduto
(não é a sequencia em que entra no oleoduto, é o oposto)
        Cl %vetor de classes correspondentes aos volumes na
sequencia que entra no oleoduto
        Qmax
        Cinterf = 0; %custo da interface entre as classes
        st = 1 %status do oleoduto:
            %1-recebendo/enviando
            %2-parado
    end

    methods
        function o = oleod(a, b, c)
            if nargin == 3
                o.V = a;
                o.Cl = b;
                o.st = c;
            end
        end
        function addV(o, newV, newClasse)
            %Carregando o Oleoduto
            if newClasse == o.Cl(1) %se a classe a ser
recebida é a mesma que já se está recebendo
                o.V(1) = o.V(1) + newV;
            else
                for i=1:length([newClasse.Interf]) %Custo de
interface
                    for j=1:length([newClasse.Interf])
                        if newClasse.Interf(i) ==
o.Cl(1).Interf(j)
                            Cost_interf = newClasse.Interf(i);
                        end
                    end
                end
                o.Cinterf = o.Cinterf + Cost_interf;
                o.Cl = [newClasse o.Cl];
                o.V = [newV o.V ];
            end

            %Descarregando o Oleoduto
            if o.V(end) > newV
                o.V(end) = o.V(end) - newV;
            else
                Vres = o.V(end);%Volume residual da última
classe no oleoduto
                o.V = [o.V(1:(length([o.V])-1))]; %Retira o
último elemento do vetor de volumes
```

```
                o.Cl = [o.Cl(1:(length([o.Cl])-1))]; %Retira o
último elemento do vetor de classes
                o.V(end) = o.V(end) - (newV - Vres);
            end
        end

    end
end
```

APÊNDICE D.1.7 – CÓDIGO DO OBJETO: REF

```
classdef ref < handle
    %REFINARIA
    properties
        Vmin %[1000 m3]
        Vmax %[1000 m3]
        V %[1000 m3]
        saveV
        Cons %Consumo de petroleo [1000 m3/h]
        Rpet = 0; %receita com petróleo recebido
    end

    methods
        function r = ref(vmin, vmax, v)%constructor
            if nargin == 3
                r.Vmin = vmin;
                r.Vmax = vmax;
                r.V = v;
            end
        end
        function addV(r,  newV, newClasse)
            r.V = r.V + newV;
            r.saveV((length(r.saveV)+1)) = r.V;
            if newV > 0
                r.Rpet = r.Rpet + newClasse.Valor_ref*newV;
            end
        end
    end
end
```

APÊNDICE D.2 – CÓDIGO DO *SCHEDULING* DO ESTUDO DE CASO

APÊNDICE D.2.1 – CÓDIGO DO *SCHEDULING*: ARQUIVO PORTO_DISCRETO.M

```
function [FO, RC, np, ct, Ect, Ect_point, htq] =
Porto_Discreto(TXT, Hz, H, t, int, passo, alfa, beta, fer_np,
fer_ct, fer_Ect, fer_Ectpoint, fer_htq, Heu_np, Heu_ct,
Heu_Ect, Heu_Ectpoint, Heu_htq, deltaE, tol, Emin, Emax)

Inicializacao_Objetos %m.file

%Rearranja o vetor navioAll por ordem de chegada (H0)
[H0ordem, I] = sort([navioAll.H0]);
temp = navioAll;
for i=1:length(navioAll)
    navioAll(i) = temp(I(i));
end
clear temp;
piervago =[p1 p2];

CalculoProbabilidades %m.file

%----------------------------- Schedule ----------------------
tab_oleod = zeros(Hz,5);
%Associa os píeres definidos em np aos navios
for i=1:length(navioAll)
    navioAll(i).pier = pierAll(np(i));
end

flag = zeros(Hz,length(navioAll));
for t=1:int
    oleod_ref %m.file

    %Tanque começa a descarregar
    for i=1:length(tqAll)
        past24 = tqAll(i).Hset + 24 - 1;
        if (H(t) >= past24) && (tqAll(i).st == 2)
            tqAll(i).st = 3; %Tq está pronto para começar a
descarregar no oleod
            tqAll(i).Hset = 0;
        end
    end

    tq_para_oleod %m.file

    for in=1:length(navioAll)
        flag(t,in) = 0;
        if isempty(navioAll(in).Hftq) == 0 %Navio deixa o pier
            x = navioAll(in).Hftq + navioAll(in).Tsai;
            if H(t) >= x && navioAll(in).st == 5
                flag(t,in) = 1;
                mude_st(navioAll(in), 1); %navio fica
indisponivel
```

```
                navioAll(in).pier.st = 1; %pier fica
disponivel
                navioAll(in).Hf = H(t);
            end
        else
            x = sum([navioAll(in).V]);
            if x == 0 && navioAll(in).st == 4
                flag(t,in) = 2;
                mude_st(navioAll(in), 5); %navio desatracando
                navioAll(in).Hftq = H(t-1);
            else
                if navioAll(in).st == 4
                    flag(t,in) = 3;
                    navio_para_tq %m.file
                else
                    if navioAll(in).st == 1 && navioAll(in).Hf
== 0 %Se o navio estiver disponível e o navio i ainda não
atracou
                        if H(t) >= navioAll(in).H0 %se o navio
está disponível para atracar
                            flag(t,in) = 4;
                            mude_st(navioAll(in), 2);
                        end
                    end

                    if navioAll(in).pier.st == 1 &&
navioAll(in).st == 2
                        flag(t,in) = 5;
                        mude_st(navioAll(in), 3); %navio
atracando
                        navioAll(in).pier.st = 2; %pier fica
indisponivel
                        navioAll(in).Hent = H(t);
                    end

                    if isempty(navioAll(in).Hent) == 0
                        x = navioAll(in).Hent +
navioAll(in).Tent;
                        if H(t) >= x && navioAll(in).st == 3
                            flag(t,in) = 6;
                            mude_st(navioAll(in), 4); %navio
descarregando
                        end
                    end
                end
            end
        end

    end
BalancoMaterial %m.file
```

```
%Função objetivo
    %Maximizar o lucro

%Custos
%Petróleo no navio + Sobrestadia + Píer + Interface
C1 = 0;
for i=1:length(navioAll)
    C1 = C1 + navioAll(i).V0*navioAll(i).pet.Custo;
end

C2 = 0; %inicializando variável
for i=1:length(navioAll)%Cálculo do custo de sobrestadia
    C2 = C2 + navioAll(i).csob;
end

C3 = 0; %inicializando variável
for i=1:length(navioAll)%Cálculo de utilização do píer
    if isempty(navioAll(i).Hent)==0 %se o navio atracou
        utilizacao = navioAll(i).tpier*pierAll(np(i)).Custo;
        if utilizacao ~= 0
            C3 = C3 + utilizacao;
        else %se o navio ainda não saiu
            C3 = C3 + (Hz -
navioAll(i).Hent)*pierAll(np(i)).Custo;
        end
    end
end

C4 = 0;
for i=1:length(oAll)
    C4 = C4 + oAll(i).Cinterf;
end

%Receitas
%Petróleo no terminal + na refinaria
R1 = sum([refAll.Rpet]); %Petróleo na refinaria
R2 = sum([tqAll.V]*[tqAll.R]') - sum([tqAll.V0]*[tqAll.R]');
%Petróleo no terminal

RC = [R1 R2 C1 C2 C3 C4]';

%Função objetivo
FO = R1 + R2 - C1 - C2 - C3 - C4; %em $
end
```

APÊNDICE D.2.2 – CÓDIGO DO *SCHEDULING*: ARQUIVO INICIALIZACAO_OBJETOS.M

```
%Modelo do Porto - Exemplo 1
%3 navios, 2 píeres, 5 tanques

%------------------------ Classes de Petróleo ----------------
-------------
CL1 = classe;
CL3 = classe;
CL4 = classe;
CL5 = classe;
CL6 = classe;

classeAll = [CL1 CL3 CL4 CL5 CL6];
for i=1:length(classeAll)
    classeAll(1,i).Valor_ref = TXT(6).data(i,1);  %valor da
classe na refinaria
    classeAll(1,i).Valor_tq = TXT(6).data(i,2);  %valor da
classe no tanque/terminal
    classeAll(:,i).Interf = TXT(8).data(:,i); %custo da
interface
    classeAll(1,i).Qmax = TXT(4).data(i,5);%Máxima vazão do
grupamento cl no duto o
end

%--------------------------------Petróleos--------------------

%Associa cada petróleo ao objeto pet
pet05 = pet;
pet08 = pet;
pet27 = pet;
pet38 = pet;
%Parâmetros dos petróleos
petAll = [pet05 pet08 pet27 pet38];
petCL = TXT(12).data;
for i=1:length(petAll)
    petAll(1,i).CL = TXT(12).data(i,:);%grupamento do petroleo
    petAll(1,i).Custo = TXT(7).data(i,:);%custo do petróleo
    for j=1:length(classeAll)%associa o petróleo às classes
        if petCL(i,j) == 1
            petAll(1,i).classe =  [petAll(1,i).classe
classeAll(1,j)];
        end
    end
end

%---------------------------------Píeres--------------------
```

```
p1 = pier;
p2 = pier;
pierAll = [p1 p2];
for i=1:length(pierAll)
    pierAll(1,i).Custo = TXT(2).data(i); %Custo de utilização
do píer [$/h]
end

%----------------------------------Navios--------------------
Front_Brea = navio;
Pedreiras_1 = navio; %coloca o petróleo do Pedreiras em 2
navios diferentes
Pedreiras_2 = navio;
Reboucas = navio;
navioAll = [Front_Brea Pedreiras_1 Pedreiras_2 Reboucas];

for i=1:length(navioAll)
    navioAll(1,i).H0 = TXT(1).data(i,1); %Hora de chegada do
navio em horas
    navioAll(1,i).Hsob = TXT(1).data(i,2); %Hora máxima p/ não
caracterizar sobrestadia
    navioAll(1,i).Csob = TXT(1).data(i,3); %Custo de
sobrestadia [$/h]
    navioAll(1,i).Qmin = TXT(1).data(i,4); %[10^3 m3/h]Vazão
mínima de descarregamento do navio n
    navioAll(1,i).Qmax = TXT(1).data(i,5); %[10^3 m3/h]Vazão
máxima de descarregamento do navio n
    temp = TXT(3).data(i,:); %vetor em que as colunas (pet)
sem volume são zero
    navioAll(1,i).V0 = temp(temp ~= 0); %volume de petróleo em
mil m3
    navioAll(1,i).V = navioAll(1,i).V0; %volume de petróleo em
mil m3
    navioAll(1,i).saveV =  navioAll(1,i).V; %Guarda os valores
de V
    np_cap = TXT(9).data; %possibilidade do navio atracar no
píer
end
clear temp;

%Petróleo contido no navio
Front_Brea.pet = pet05;
Pedreiras_1.pet = pet08;
Pedreiras_2.pet = pet27;
Reboucas.pet = pet38;

%----------------------------------Tanques--------------------
TQ3234 = tq;
TQ3237 = tq;
TQ3239 = tq;
```

```
TQ3241 = tq;
TQ3243 = tq;

tqAll = [TQ3234 TQ3237 TQ3239 TQ3241 TQ3243];
tqAll(4).st = 4;%Tanque que já começa descarregando no
oleoduto
for i=1:length(tqAll)
    tqAll(1,i).V0 = TXT(4).data(i,3); %Quantidade inicial de
petróleo no tanque t
    tqAll(1,i).V = tqAll(1,i).V0; %Quantidade inicial de
petróleo no tanque t
    tqAll(1,i).saveV =  tqAll(1,i).V; %Guarda os valores de V
    tqAll(1,i).Vmin = TXT(4).data(i,1); %Capacidade mínima de
armazenamento do tanque t
    tqAll(1,i).Vmax = TXT(4).data(i,2); %Capacidade máxima de
armazenamento do tanque t
    tqAll(1,i).Qmax = TXT(4).data(i,5); %Vazão máxima !deveria
estar associado ao duto
    tqAll(1,i).Cl = classeAll(i);
    tqAll(1,i).R = TXT(6).data(i,2); %Preço por unidade de
volume do grupamento de petróleo no terminal
end

%--------------------------------Oleoduto---------------------
-----------
OSVAT = oleod;
OSVAT.V = [30 35]; %Inventário inicial no oleoduto
OSVAT.Cl = [CL5 CL3];
OSVAT.st = 1; %status do oleoduto: 1-recebendo
oAll = [OSVAT];

%--------------------------------Refinarias-------------------
REVAP = ref;
refAll = [REVAP];
for i=1:length(refAll)
    refAll(1,i).V = TXT(5).data(i,1);
    refAll(1,i).saveV = TXT(5).data(i,1); %guarda volume V
    refAll(1,i).Vmin = TXT(5).data(i,2);
    refAll(1,i).Vmax = TXT(5).data(i,3);
    refAll(1,i).Cons = TXT(5).data(i,4); %consumo de petróleo
na refinaria
end
```

APÊNDICE D.2.3 – CÓDIGO DO *SCHEDULING*: ARQUIVO CALCULOPROBABILIDADES.M

```
%---------------------- Cálculo das probabilidades -----------
```

```
    %---- Escolha do Píer-----

prove = 1;
imax = length(navioAll);
jmax = length(pierAll);
while prove>0
    check_np = zeros(length(navioAll),length(pierAll));
    [ np, prob_np ] = prob_gen( imax, jmax, fer_np, alfa,
beta, Heu_np);
    np(4) = np(3); %ajuste para simular 2 petroleos no mesmo
navio %foi zerada a desatracacao do navio 3 e a atracação do
navio 4
    for i=1:imax
        check_np(i,np(i)) = 1;
    end
    prove = sum(sum(check_np > np_cap)); %checa se algum navio
foi associado a um píer que não tem a capacidade para recebê-
lo
end
    %---Eventos ----
%Quantidade de eventos de transferência de petróleo do navio
para o tanque
for i=1:length(pierAll)
    [Ect_temp, prob_Ect] = prob_gen(1, deltaE, fer_Ect(i,:),
alfa, beta, Heu_Ect);
    Ect(i) = Ect_temp + Emin - 1;
end

%Hora do início dos eventos de transferência de petróleo do
navio para o tanque
Ect_point = zeros(length(pierAll),Emax);
for i=1:length(pierAll)
    s = size(fer_Ectpoint);
    fer_temp(1:s(2),1:s(3)) = fer_Ectpoint(i,:,:);
    [Ect_point_temp, prob_Ectpoint] = prob_gen(Ect(i), int,
fer_temp, alfa, beta, Heu_Ectpoint);
    Ect_point(i,1:length(Ect_point_temp)) =
round(sort(Ect_point_temp)); %o round retira os decimais. Ex.:
125.00 = 125, para poder utilizar a função repelem
end

    %---Escolha do Tanque ----

% Heurística: escolher o tanque cujo petróleo pré-existente
possui o maior
% valor na refinaria
% Método: criar lista com ordem de preferência de tanques para
cada
% petróleo
```

```
imax = length(petAll);
jmax = length(tqAll);

 [ ct_temp, prob_ct ] = prob_ans( imax, jmax, Emax, petAll,
fer_ct, alfa, beta, Heu_ct, tqAll );

ct =  repelem(ct_temp(:,1),1,int); %garante que a variável
estará preenchida independente da qtdade de eventos.
for j=1:length(pierAll)
    for i=2:Ect(np(j))
        delta = Ect_point(np(j),i) - Ect_point(np(j),i-1);
        for k=1:length(petAll)
            if np(k)==j
                ct(k , Ect_point(np(j),i-
1)+1:Ect_point(np(j),i)) = repelem(ct_temp(k,i),1,[delta]);
            end
        end
    end
end
  %---Escolha do volume até o qual o tanque vai encher antes
de começar o assentamento do petróleo ----
    % Heurística: maximizar o volume

imax = int;
jmax = length(tqAll) + 1;%A cada instante um dos tanques ou
nenhum deles irá começar o settling

[ htq_temp, prob_htq ] = prob_gen( imax, jmax, fer_htq, alfa,
beta, Heu_htq);
htq = htq_temp - 1; %onde zero quer dizer nenhum dos tanques
```

APÊNDICE D.2.4 – CÓDIGO DO *SCHEDULING*: ARQUIVO PROB_GEN.M

```
function [ var, prob ] = prob_gen( imax, jmax, fer, alfa,
beta, Heu)
%Calcula as probablidades e o resultado da variável decisória

soma = zeros(imax,1);
    for i=1:imax
        for j=1:jmax
            soma(i) = soma(i) + fer(i,j)^alfa*Heu(i,j)^beta;
        end
    end

prob = zeros(imax,jmax);%inicializando variável
```

```
for i=1:imax
    for j=1:jmax
        prob(i,j) = (fer(i,j)^alfa*Heu(i,j)^beta)/soma(i);
    end
end

var = zeros(1,imax);
for i=1:imax
    var(i) = roleta(prob(i,:));
end
```

APÊNDICE D.2.5 – CÓDIGO DO *SCHEDULING*: ARQUIVO PROB_ANS.M

```
function [ ct, prob_ct ] = prob_ans( imax, jmax, col, petAll,
fer_ct, alfa, beta, Heu, tqAll )
%PROB_ANS Calcula as probablidades e o resultado da variável
decisória
%   Dados de entrada: imax, jmax, petCL, parâmetros ACO, tqAll
%   Dados de Saída: ct, prob_ct

soma = zeros(1,imax);
for i=1:imax
    for j=1:jmax
        if sum(petAll(i).classe == tqAll(j).Cl) == 1 %se a
classe do petróleo é igual a do tanque
            soma(i) = soma(i) +
fer_ct(i,j)^alfa*(Heu(i,j)*tqAll(j).Cl.Valor_tq)^beta;
        end
    end
end

prob_ct = zeros(imax,jmax);%inicializando variável

for i=1:imax
    for j=1:jmax
         if sum(petAll(i).classe == tqAll(j).Cl) == 1 %se a
classe do petróleo é igual a do tanque
            prob_ct(i,j) =
(fer_ct(i,j)^alfa*(Heu(i,j)*tqAll(i).Cl.Valor_tq)^beta)/soma(i
);
         end
    end
end

ct = zeros(imax,col);
for i=1:imax
    for j=1:col%!Qtdade de tanques na lista de preferência
```

```
        ct(i,j) = roleta(prob_ct(i,:));
    end
end
end
```

APÊNDICE D.2.6 – CÓDIGO DO *SCHEDULING*: ARQUIVO ROLETA.M

```
function f = roleta(prob)

    C = cumsum(prob);

    r = rand*C(end); %Esse produto normaliza o r, pois no
prob_ans a soma das probabilidades não é 1 por causa das
condicinais.

    f = find(r <= C, 1, 'first');

end
```

APÊNDICE D.2.7 – CÓDIGO DO *SCHEDULING*: ARQUIVO OLEOD_REF.M

```
for i=1:length(refAll)
    consumo = -refAll(i).Cons*passo;
    addV(refAll(i), consumo);
end

%A vazão máxima no oleoduto é a vazão máxima do petróleo mais
denso

%Cálculo da vazão
    %A vazão máxima é a menor das vazões máximas das classes
no oleoduto
for i=1:length(oAll)
    oAll(i).Qmax = min([oAll(i).Cl.Qmax]);
end
```

APÊNDICE D.2.8 – CÓDIGO DO *SCHEDULING*: ARQUIVO TQ_PARA_OLEOD.M

```
%A vazão possível ou é Qmax ou é zero
    %Volume que a refinaria pode receber
    o_ref = 0;
    for i=1:length(refAll)
        Vref(i) = refAll(i).Vmax - refAll(i).V;
```

```
        Qmax_possivel(i) = Vref(i)*passo;
        if Qmax_possivel(i)>= oAll.Qmax %!válido p/ um
oleoduto e uma refinaria
            o_ref = 1; %o oleoduto pode transferir para a
refinaria se houver tanque p/ enviar
        end
    end

%Descarregamento do Tanque
for i=1:length(tqAll)%impede que mais de um tq tenha status
descarregando
    tqAll(i).H_switch = 0;
    if ismember(4,[tqAll.st]) == 0 && tqAll(i).st == 3 %!A
preferência fica na ordem dos tanques => flexibilizar
        tqAll(i).st = 4;
    end
end

for i=1:length(tqAll)
    if tqAll(i).st == 4 %status descarregando
        if o_ref == 1 %refinaria pode receber
            Vol_des = oAll(1).Qmax*passo; %Volume a ser
descarregado do tanque para o oleoduto !1 oleoduto
            Vol_disp = tqAll(i).V - tqAll(i).Vmin; %Volume
disponivel no tq p/ ser descarregado
            if Vol_disp >= Vol_des
                sendV(tqAll(i), Vol_des); %Envia do tq p/ o
oleod
                addV(oAll(1), Vol_des, tqAll(i).Cl); %Oleoduto
recebe e envia o petróleo !1 oleoduto
                addV(refAll(1), Vol_des, tqAll(i).Cl);
%refinaria recebe o petróleo !1 refinaria
            else
                tqAll(i).st = 1; %tanque passa a poder receber
do navio e para de enviar para oleoduto
                tqAll(i).H_switch = H(t);
                oAll(1).st = 2; %oleoduto nao recebe petroleo
            end
        end
    end
end
```

APÊNDICE D.2.9 – CÓDIGO DO *SCHEDULING*: ARQUIVO NAVIO_PARA_TQ.M

```
% Associa petróleos aos tanques na ordem definida em ct
go_amax %m.file
Vadd = navioAll(in).Qmax*passo;
if ((Vadd <= amax(in)) || (navioAll(in).V < Vadd)) &&(H(t) ~=
tqAll(ct(in,t)).H_switch)
```

```
    if (navioAll(in).V < Vadd) %Se o navio não tiver volume
suficente para esvaziar no Qmax, ele esvazia todo na vazão
possível
        Vadd = navioAll_V(in);
    end
    if tqAll(ct(in,t)).st == 1 %Status carregando
        addV(navioAll(in), -Vadd, navioAll(in).st); %addV(n,
newV, totalpet, qpet, newst)
        st_aux = tqAll(ct(in,t)).st;
        addV(tqAll(ct(in,t)), Vadd, navioAll(in).pet(1),
st_aux(end)); %adiciona o volume mas mantém o status
        Vdis = tqAll(ct(in,t)).Vmax - tqAll(ct(in,t)).V;
%volume disponivel que enche o tq
        if (Vadd > Vdis) || (tqAll(ct(in,t)).V >=
tqAll(ct(in,t)).Vmax) || htq(t) == tqAll(ct(in,t))%se o tq nao
pode mais receber do navio !só funciona pq a vazão de todos os
navios são iguais OU se o volume no tanque já é maior ou igual
ao determinado pelo ACO
            mude_st(tqAll(ct(in,t)), 2); %novo status -
settling. O volume se mantém e o tempo do evento é nulo
            tqAll(ct(in,t)).Hset = H(t);
        end
    end
end
navioAll_V = [navioAll.V];

%Tanque começa a descarregar
past24 = tqAll(ct(in,t)).Hset + 24 - 1;
if (H(t) >= past24) && (tqAll(ct(in,t)).st == 2)
    tqAll(ct(in,t)).st = 3; %Tq está pronto para começar a
descarregar no oleod
    tqAll(ct(in,t)).Hset = 0;
end
```

APÊNDICE D.2.10 – CÓDIGO DO *SCHEDULING*: ARQUIVO GO_AMAX.M

```
navioAll_V = [navioAll.V];
amax = zeros(length(navioAll_V),1); %Volume máximo que pode
entrar no tanque
for j=1:length(petAll) %LOOP amax
    if navioAll_V(j) >= (tqAll(ct(j,t)).Vmax -
tqAll(ct(j,t)).V) %se o volume disponível no navio for maior
do que o disponível no tanque
       amax(j) = (tqAll(ct(j,t)).Vmax - tqAll(ct(j,t)).V);
    end
    if navioAll_V(j) < (tqAll(ct(j,t)).Vmax -
tqAll(ct(j,t)).V)
```

```
        amax(j) = navioAll_V(j);
    end
end
```

APÊNDICE D.2.11 – CÓDIGO DO *SCHEDULING*: ARQUIVO BALANCOMATERIAL.M

```
deltaV_Navio = sum([navioAll.V]) - sum([navioAll.V0]);
deltaV_Terminal = sum([tqAll.V]) - sum([tqAll.V0]);
deltaV_Refinaria = refAll(1).V - refAll(1).saveV(1);
consumo_Refinaria = refAll(1).Cons*Hz;

Balanco = [deltaV_Navio, deltaV_Terminal, deltaV_Refinaria];
Desvio_Balanco = sum(Balanco) + consumo_Refinaria;

if Desvio_Balanco > tol || Desvio_Balanco < - tol
    Desvio_Balanco
    pause
end
```

APÊNDICE D.3 – CÓDIGO DO ACO DO ESTUDO DE CASO

```
clc
clear all
close all
LoadTXT
Inicializacao_Objetos %m.file
%-------------------------Parâmetros da otimização ------------

Hz = 96; %Horizonte de tempo do scheduling [h]

passo = 0.5; %[h]
Emin = 8; Emax = 12; %Eventos mínimos e máximos
n = 300; %iterações
ants = 40; %quantidade de formigas por iteração

fer_evap = 0.0178;
fer_atu = 10^-6; %É multiplicado pela FO
alfa = 0.6680;
beta = 4.1611;
fer0 = 0.1;
xHeuhtq = 20; %Heurística de quantas vezes é mais provável
nenhum tq começar a sedimentação do que algum tanque, em cada
tempo
```

```
tol = 10^-4; %tolerância do desvio de balanço material
[1000m3]

deltaE = Emax - Emin + 1;
int = Hz/passo; %96;%intervalos de tempo %+passo pq começa do
zero
t = 1; %Índice de H(hora) - inicialização
H = [0:passo:Hz]; %Hora

fer_np(1:length(navioAll),1:length(pierAll)) = fer0; %navio
>>> píer
fer_Ect(1:length(pierAll),1:deltaE) = fer0;
fer_Ectpoint(1:length(pierAll),1:Emax,1:int) = fer0;
fer_ct(1:length(petAll),1:length(tqAll)) = fer0; %petróleo >>>
tanque
fer_htq(1:int,1:length(tqAll)+1) = fer0; %Hora em que o tanque
começa o settling

%Heurísticas ------------------------------------------------
    %Píer mais barato
    for i=1:length(navioAll)
        for j=1:length(pierAll)
            Heu_np(i,j) = 1/pierAll(j).Custo; %navio >>> píer
        end
    end

    %Mais eventos
    for i=1:Emax
        Heu_Ect(1,i) = fer0 + i*10^-3;
    end

    %Grupamento com maior valor na refinaria
    for i=1:length(petAll)
        for j=1:length(tqAll)
            Heu_ct(i,j) =
tqAll(j).Cl.Valor_ref/tqAll(1).Cl.Valor_ref; %petróleo >>>
tanque
        end
    end

    Heu_htq(1:int,1) = fer0*xHeuhtq; %maior probabilidade de
nenhum dos tanque settling
    Heu_htq(1:int,2:length(tqAll)+1) = fer0; %Hora em que o
tanque começa o settling

    %Sem Heurística
    Heu_Ectpoint(1:Emax,1:int) = fer0;

    htq = zeros(ants,int);
    Ect_point = zeros(ants,length(pierAll),Emax);
```

```
for ii=1:n
    Ect_point(1:ants,1:length(pierAll),1:Emax) = zeros;
    for m=1:ants
        [FO(m), RC(:,m), np(m,:), ct(m,:,:), Ect(m,:),
Ect_point(m,:,:), htq(m,:)] = Porto_Discreto(TXT, Hz, H, t,
int, passo, alfa, beta, fer_np, fer_ct, fer_Ect, fer_Ectpoint,
fer_htq, Heu_np, Heu_ct, Heu_Ect, Heu_Ectpoint, Heu_htq,
deltaE, tol, Emin, Emax);
    end

    %Evaporação
    fer_np = fer_np*(1 - fer_evap); %navio >>> pier
    fer_ct = fer_ct*(1 - fer_evap); %petróleo >>> tanque
    fer_Ect = fer_Ect*(1 - fer_evap);
    fer_Ectpoint = fer_Ectpoint*(1 - fer_evap);
    fer_htq = fer_htq*(1 - fer_evap); %Hora em que o tanque
começa o settling

    %Atualização
    %FO_best(ii) = FO(bestAnt(1))+25000; %%Estudo de caso 2,
compensa os FO negativos
    [~,bestAntcomp] = find(FO == max(FO));
    bestAnt = bestAntcomp(1);
    FO_best(ii) = FO(bestAnt);
    RC_best(:,ii) = RC(:,bestAnt);

    [a,b] = size(fer_np);
    for ka=1:a
        for kb=1:b
            if np(bestAnt,ka) == kb
                if FO_best(ii)>0
                    fer_np(ka,kb) = fer_np(ka,kb) +
fer_atu*FO_best(ii);
                end
            end
        end
    end
    np_it(ii,:) = np(bestAnt,:);

    [a,b] = size(fer_ct);
    for ka=1:a
        for kb=1:b
            if ct(bestAnt,ka,kb) == kb
                if FO_best(ii)>0
                    fer_ct(ka,kb) = fer_ct(ka,kb) +
fer_atu*FO_best(ii);
                end
            end
        end
    end
```

```
    ct_it(ii,:,:) = ct(bestAnt,:,:);

    [a,b] = size(fer_Ect);
    for ka=1:a
        for kb=1:b
            if Ect(bestAnt, ka) == kb + Emin -1
                if FO_best(ii)>0
                    fer_Ect(ka,kb) = fer_Ect(ka,kb) +
fer_atu*FO_best(ii);
                end
            end
        end
    end
    Ect_it(ii,:) = Ect(bestAnt,:);

    [a,b,c] = size(fer_Ectpoint);
    for kb=1:a
        for kc=1:b
            if Ect_point(bestAnt,kb) == kc
                if FO_best(ii)>0
                    fer_Ectpoint(:,kb,kc) =
fer_Ectpoint(:,kb,kc) + fer_atu*FO_best(ii);
                end
            end
        end
    end
    Ect_point_it(ii,:,:) = Ect_point(bestAnt,:,:);

    [a,b] = size(fer_htq);
    for ka=1:a
        for kb=2:b
            if htq(bestAnt,ka) == kb
                if FO_best(ii)>0
                    fer_htq(ka,kb+1) = fer_htq(ka,kb+1) +
fer_atu*FO_best(ii) ;
                end
            end
        end
        if htq(bestAnt,ka) == 0
            if FO_best(ii)>0
                fer_htq(ka,1) = fer_htq(ka,1) +
fer_atu*(10*FO_best(ii)) ;
            end
        end
    end
    htq_it(ii,:,:) = htq(bestAnt,:,:);

end

plot(FO_best)
```

```
[~, colmax] = max(FO_best);

RCtab = [RC_best(:,end); RC_best(:,colmax)];
param = [passo; Emin; Emax; n; ants; fer_evap; fer_atu; alfa;
beta; fer0; xHeuhtq];

%Excel:
tab0 = table(param);
tab1 = table(FO_best');
tab2 = table(RCtab);
filename = 'EstudodeCaso1_Parametros.xlsx';
writetable(tab0 ,filename,'Sheet',1,'Range','C1');
writetable(tab1 ,filename,'Sheet',1,'Range','D1');
writetable(tab2 ,filename,'Sheet',1,'Range','C14');
```