Universidade Federal de Itajubá
Programa de Pós-graduação em Engenharia Elétrica

Luiz Eduardo da Silva

Sistema Híbrido Metaheurístico baseado em Colônia de Formigas Paraconsistentes aplicado a Problemas de Otimização em Redes Inteligentes

Tese submetida ao Programa de Pós-graduação em Engenharia Elétrica como parte dos requisitos para a obtenção do título de Doutor em Ciências em Engenharia Elétrica.

Área de Concentração:
Sistemas Elétricos de Potência

Orientador: Prof. Dr. Germano Lambert Torres
Orientador: Prof. Dr. Luiz Eduardo Borges da Silva

Setembro de 2012
Itajubá

Universidade Federal de Itajubá
Programa de Pós-graduação em Engenharia Elétrica

Luiz Eduardo da Silva

Sistema Híbrido Metaheurístico baseado em Colônia de Formigas Paraconsistentes aplicado a Problemas de Otimização em Redes Inteligentes

Tese aprovada por banca examinadora em 14 de Setembro de 2012, conferindo ao autor o título de **Doutor em Ciências em Engenharia Elétrica**

Banca Examinadora:
Prof. Dr. Germano Lambert Torres (orientador) - UNIFEI
Prof. Dr. Luiz Eduardo Borges da Silva (orientador) - UNIFEI
Prof. Dr. Alexandre Rasi Aoki - UFPR
Prof. Dr. Erik Leandro Bonaldi - PS Soluções
Prof. Dr. Giscard Francimeire Cintra Veloso - UNIFEI
Prof. Dr. Maurílio Pereira Coutinho - UNIFEI

Itajubá
2012

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Cristiane N. C. Carpinteiro- CRB_6/1702

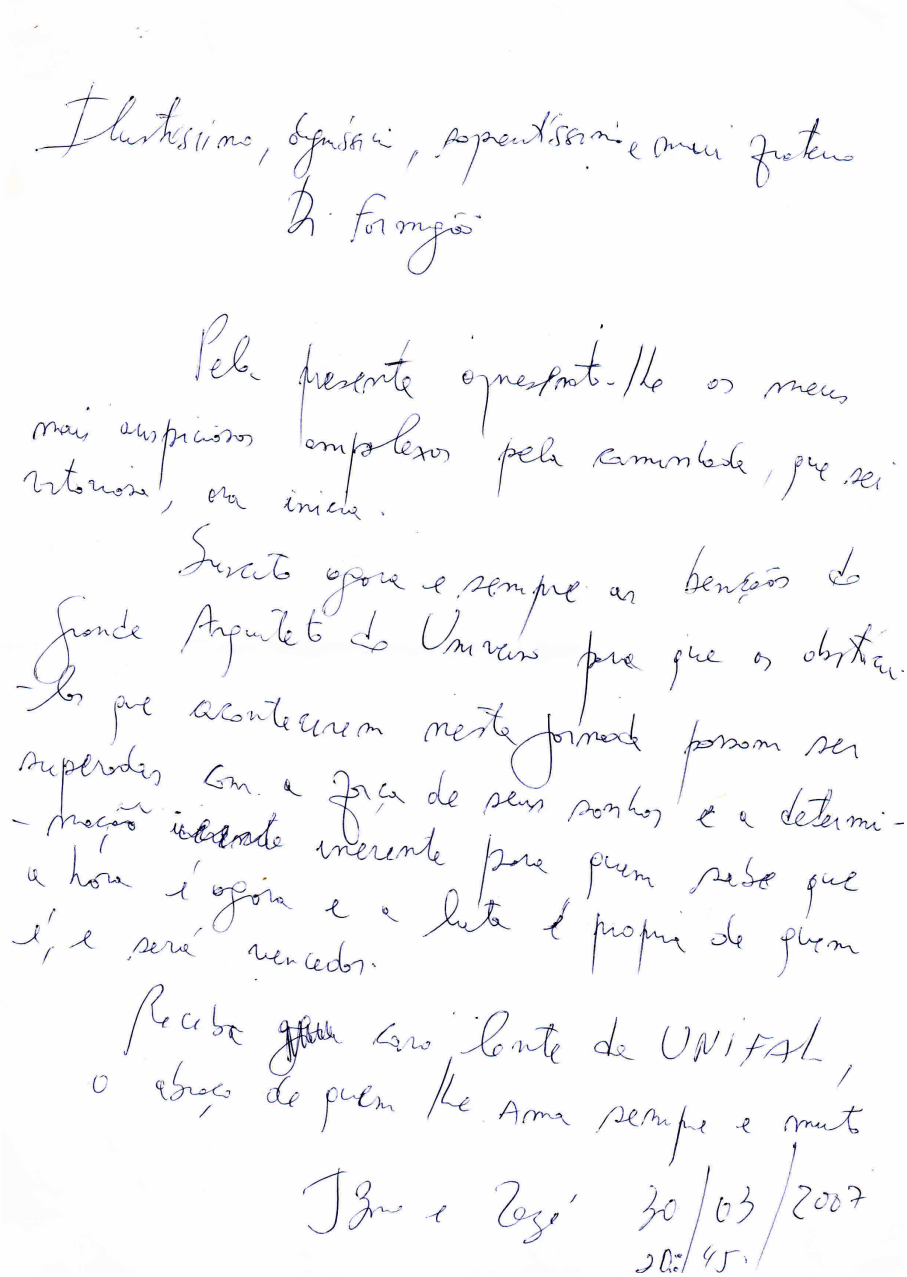
S586s

Silva, Luiz Eduardo da
Sistema híbrido metaheurístico baseado em colônia de formigas para-consistentes aplicado a problemas de otimização em redes inteligentes. / por Luiz Eduardo da Silva. -- Itajubá (MG): [s.n.], 2012.
104 p.: il.

Orientadores: Prof. Dr. Germano Lambert Torres.
Prof. Dr. Luiz Eduardo Borges da Silva.
Tese (Doutorado) – Universidade Federal de Itajubá.

1. Inteligência artificial. 2. Colônia de formigas. 3. Lógica paraconsistente. 4. Redes inteligentes. I. Torres, Germano Lambert, orient. II. Silva, Luiz Eduardo Borges da, orient. III. Universidade Federal de Itajubá. IV. Título.

Dedicatória



Ilustríssimo, dignísimo, aprendizíssimo e amigo fraterno
Dr. Formigão

Pela presente apresento-lhe os meus mais auspiciosos empolgos pela caminhada, que será vitoriosa, em início.

Sinto agora e sempre as bênçãos do Grande Arquiteto do Universo para que os obstáculos que aconteçam neste jornada possam ser superados com a força de seus sonhos e a determinação inerente para quem sabe que a hora é agora e a luta é própria de quem é, e será vencedor.

Receba, caro ilustre de UNIFAL, o abraço de quem lhe ama sempre e muito

J. Bosco e Zezé 30/03/2007
20:45.

Ao “fraterno” irmão J. Bosco

Agradecimentos

Quero agradecer à Deus pela dádiva da vida, esse bem tão precioso mas também tão negligenciado nesse mundo hodierno.

Aos Orientadores Germano Lambert Torres e Luiz Eduardo Borges da Silva pela confiança e colaboração, sem os quais esse trabalho não teria sido realizado.

Agradeço também a Universidade Federal de Alfenas e a Universidade Federal de Itajubá. A primeira, através dos seus gestores, por permitir iniciar esse trabalho. E a segunda por me receber e através dos seus professores e funcionários, me permitir a realização dessa pesquisa.

À minha esposa Regiane, pelo amor, dedicação e paciência. As dificuldades são muitas, principalmente para o orientado que é pai de família e não pode simplesmente parar e largar tudo para fazer o doutorado.

Às minhas filhas Bibiana e Catarina, pelo carinho e pelo amor. Nos momentos mais difíceis, em que a vontade era de desistir e largar tudo, me inspirei em vocês.

À minha mãe Ana Rosa (*in memorian*) e ao meu pai Noel (*in memorian*). A minha mãe, infelizmente não pode curtir comigo a término do meu mestrado. Fiz para ela uma dedicatória especial naquela oportunidade. E agora o meu pai, infelizmente, não está mais entre nós para curtir comigo o término dessa etapa da minha vida acadêmica.

Ao meu irmão João Bosco (*in memorian*), que me incentivou sempre a correr atrás dos meus sonhos. Estava comigo no dia em que iniciei o doutorado e infelizmente não está entre nós para festejar comigo esse momento.

À minha família.

Aos meus colegas de trabalho.

Aos meus alunos.

Resumo

SILVA, L. E. **Sistema Híbrido Metaheurístico baseado em Colônia de Formigas Paraconsistentes aplicado a Problemas de Otimização em Redes Inteligentes**. 2012. Tese (Doutorado) - Instituto de Sistemas Elétricos e Energia, Universidade Federal de Itajubá, Minas Gerais, 2012.

Este trabalho apresenta uma proposta de um algoritmo híbrido de metaheurística denominada colônia de formigas e da lógica não-clássica denominada Lógica Paraconsistente para resolução de problemas de otimização na área de Sistemas Elétricos de Potência. A estratégia de inteligência de bando nomeada Colônia de Formigas tem demonstrado ser uma estratégia interessante para resolução de problemas de otimização combinatória e também no domínico contínuo. Nesta estratégia, o aprendizado das formigas é construído através de uma trilha de feromônios deixada por cada formiga da colônia no espaço do problema. As formigas usam interativamente o nível de feromônio de cada caminho para decidir que caminho seguir. O problema é que esta decisão é muitas vezes incerta ou inconsistente. A lógica clássica não é capaz de tratar este tipo de problema de decisão. Desta forma, propõe-se o uso da Lógica Paraconsistente para aumentar o poder de decisão das formigas da colônia. Nesse trabalho, o algoritmo proposto é aplicado em duas classes de problemas em Sistemas Elétricos de Potência: o restabelecimento de sistemas e o despacho econômico. Com os resultados apresentados pode-se dizer que o sistema híbrido da Lógica Paraconsistente com a Colônia de Formigas demonstra ser uma interessante alternativa para resolução de problemas de otimização.

Palavras-chave: Inteligência Artificial, Colônia de Formigas, Lógica Paraconsistente, Redes Inteligentes.

Abstract

SILVA, L. E. **Hybrid Metaheuristic System based on Paraconsistent Ant Colony applied to Optimization Problems in Smart Grids**. 2012. Tese (Doutorado) - Instituto de Sistemas Elétricos e Energia, Universidade Federal de Itajubá, Minas Gerais, 2012.

This work introduces a proposal for a hybrid metaheuristic algorithm called ant colony and non-classical logic called Paraconsistent logic for solving optimization problems in the area of Electric Power Systems. The strategy of the swarm intelligence named Ant Colony has proved to be an interesting way to solve difficult combinatorial problems. In this strategy, the learning of ants is built through the trail of pheromones left by the each ant in the colony on the problem space. The ants iteratively use the pheromone level of the each path of the solution problem to decide which way to follow. The problem is that this decision is often uncertain or inconsistent. Classical logic can not handle this kind of decision problem. In this sense we use the Paraconsistent Logic for increasing the power of decision to the ants colony. In this work, the proposed algorithm is applied in two classes of problems in Electric Power Systems: the restoration systems and the economic dispatch. With these results, we can say that the hybrid system of Paraconsistent Logic with Ant Colony proves to be an interesting alternative for solving of optimization problems.

Keywords: Artificial Intelligence, Ant Colony, Paraconsistent Logic, Smart Grids.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACO	Otimização por Colônia de Formigas (*Ant Colony Optimization*)
SED	Sistema Elétrico de Distribuição
DE	Despacho Econômico
$\mathcal{MM}$AS	Variação do ACO denominado $\mathcal{MAX} - \mathcal{MIN}$
QUPC	Quadrado Unitário do Plano Cartesiano
LPC	Linha Perfeitamente Consistente
LPI	Linha Perfeitamente Inconsistente
GC	Grau de Certeza
GI	Grau de Incerteza
LPA2v	Lógica Paraconsistente Anotada de duas variáveis
LPA3v	Lógica Paraconsistente Anotada de três variáveis
$ACO_{\mathbb{R}}$	ACO para contínuo (*ACO extended to continuous*)

Lista de Símbolos

τ_{ij}	Feromônio associado ao componente i e j do problema
η_{ij}	Heurística associada ao componente i e j do problema
N^k	Vizinhança da formiga k
α	Relevância do feromônio
β	Relevância da heurística
ρ	Taxa de evaporação de feromônio
$\neg$	Negação
$\wedge$	Conjunção
$\vee$	Disjunção
$\rightarrow$	Implicação
$\top$	Inconsistente
$\bot$	Paracompleto
μ	Grau de crença na lógica paraconsistente
λ	Grau de descrença
ε	Grau de especialidade
δ	Variação da especialidade
η_{NF}	Heurística das chaves Normalmente Fechadas (NF)
τ_{NF}	Feromônio das chaves Normalmente Fechadas
τ_{NA}	Feromônio das chaves Normalmente Abertas (NA)
k	Tamanho do Arquivo de Soluções no algoritmo $\text{ACO}_{\mathbb{R}}$
m	Número de formigas
ξ	Velocidade da convergência no $\text{ACO}_{\mathbb{R}}$
q	Vizinhança em torno da melhor solução no Arquivo de Soluções

Sumário

Capítulo 1

Introdução

O computador é usado, atualmente, com sistemas inteligentes para auxiliar em problemas de tomadas de decisão. Entretanto, entre estes problemas existem aqueles que não podem ser resolvidos nos computadores em tempo viável. O número de testes que devem ser realizados é tão grande que mesmo o melhor dos computadores conhecidos levaria séculos a fim de produzir uma solução exata para o problema.

Na impossibilidade de produzir algoritmos exatos para estes problemas, tem sido cada vez mais utilizados algoritmos probabilísticos para produzir, senão uma solução ótima, ao menos uma boa solução para os problemas. Dentre as técnicas utilizadas na inteligência artificial para encontrar soluções em problemas de computação difícil, existem os algoritmos bioinspirados, que simulam o funcionamento de algum mecanismo humano ou da natureza a fim de representar o processo inteligente utilizado por organismos vivos. Entre as técnicas inteligentes empregadas estão as redes neurais artificiais, os algoritmos genéticos, a programação genética, o bando de partículas e as colônias de formigas.

A técnica de Inteligência Artificial denominada Colônias de Formigas é um sistema bioinspirado que simula o processo realizado pelas formigas de uma colônia na busca por alimentos. Esta técnica é especialmente indicada para problemas de otimização combinatória, onde uma sequência de valores discretos devem ser obtidos para solução do problema.

Para tomada de decisão na estratégia de Colônia de Formigas usa-se somente uma função de probabilidade. Os problemas de incerteza e inconsistência naturais no processo de decisão das formigas não são considerados. Desta forma, neste trabalho é proposto um sistema híbrido da estratégia de colônia de formigas com a lógica paraconsistente. A lógica paraconsistente é uma lógica não-clássica indicada para utilização em problemas de indefinição, conhecimento parcial e inconsistências. Utiliza-se a lógica paraconsistente junto com a estratégia de colônia de formigas para procurar maximizar as vantagens e minimizar as desvantagens inerentes de cada estratégia.

O resultado é um sistema híbrido, fortemente acoplado, que apresenta um comportamento melhor ou igual à melhor estratégia de colônia de formigas implementada.

1.1 Contexto

Os sistemas de energia elétrica, conforme conhecemos hoje, já existem a mais de um século seguindo o proposta original de Tesla e Westinghouse (Hughes , 1983). Durante esse século houve alguma evolução graças ao desenvolvimento tecnológico. No entanto, para alguns especialistas, os sistemas elétricos atuais começam a se tornar antiguados, por deixarem de incorporar alguns novos avanços tecnológicos.

Com as tecnologias atuais, é possível monitorar uma cirurgia a distância, porém na ocorrência de uma falha no sistema elétrico ainda é necessário que um consumidor faça uma ligação telefônica para informar o problema e equipes de manutenção devam ser deslocadas para restaurar o fornecimento de energia nessa região afetada.

Outros problemas relacionados ao uso de novas tecnologias para a geração e distribuição de energia podem ser citados. A construção de novas centrais de geração de energia, sejam elas hidrelétricas ou térmicas, devem considerar seus impactos sócio-ambientais, dada essa nova preocupação da sociedade moderna. Os sistemas de transmissão também devem levar em consideração os impactos que podem gerar no meio ambiente. Novas fontes não convencionais de geração como painéis fotovoltaicos e usinas eólicas tendem a permitir que consumidores de energia elétrica possam atuar como produtores no novo modelo de geração e distribuição de energia.

Novas tecnologias como o carro elétrico híbrido (Werbos , 2011) apresentam-se como uma alternativa para o armazenamento de energia elétrica produzida por fontes renováveis não despacháveis.

Finalmente, a utilização de novos equipamentos de medição e controle de consumo/produção de energia (*smart meters*) torna possível a construção de um novo sistema de energia elétrica, no qual a produção é controlada de forma mais realista, considerando o consumo online de seus consumidores, ao mesmo tempo em que o consumidor passa a ser também um produtor de energia, alimentando o sistema com o excedente de sua produção local.

Toda essa discussão apresenta um novo conceito denominado *Smart Grid* [1], que consiste em utilizar de forma intensiva a tecnologia da informação e comunicação para possibilitar a implantação de estratégias de otimização e controle das redes elétricas de uma forma muito mais eficiente, confiável e integrada.

Nesse contexto, novos algoritmos devem ser implementados a fim de resolver problemas de otimização e de tomada de decisão nos mais diversos campos da engenharia elétrica e, assim, possibilitar o implantação do conceito de Redes Inteligentes, mais adequados ao mundo digital em que vivemos.

[1] A tradução consagrada para esse termo em português é algo como Rede Inteligente ou Rede Elétrica Inteligente. O mesmo termo é usado em outro contexto em Computação.

1.2 Objetivos

Esta pesquisa visa desenvolver e validar o algoritmo de Formiga Paraconsistente, que é um sistema híbrido da Lógica Paraconsistente com a metaheurística de Colônia de Formigas. Esse algoritmo pode atuar em dois tipos de espaço de busca: o espaço de busca discreto e o espaço de busca contínuo. O algoritmo de Formiga Paraconsistente será aplicado em problemas de Otimização em Sistemas Elétricos de Potência, a fim de exemplificar a sua utilização nessas classes de problemas. Para o domínio discreto será utilizado o problema de Restabelecimento de Sistemas de Distribuição e para o domínio contínuo será utilizado o problema do Despacho Econômico de Carga.

1.2.1 Objetivos Específicos

- Implementar algumas variantes da metaheurística de Colônia de Formigas;
- Implementar uma interpretação da lógica paraconsistente para utilizar com a metaheurística de colônia de formigas;
- Desenvolver um sistema híbrido da lógica paraconsistente e da colônia de formigas para implementação da Formiga Paraconsistente;
- Aplicar a Formiga Paraconsistente ao *benchmark* do Caixeiro Viajante;
- Aplicar a Formiga Paraconsistente ao problema de Restabelecimento de Sistemas Distribuição, direcionado para aplicação em Redes Elétricas Inteligentes;
- Aplicar a Formiga Paraconsistente ao problema de Despacho Econômico de Carga;
- Analisar e publicar os resultados.

1.3 Organização da Pesquisa

Este trabalho está organizado da seguinte forma: no Capítulo 2 são discutidos os problemas de computação difícil e as estratégias utilizadas, dentre elas a metaheurística de Colônia de Formigas. No Capítulo 3 são introduzidos, em linhas gerais, os conceitos e fundamentos da lógica não-clássica denominada Lógica Paraconsistente. Discute-se ainda, neste capítulo, alternativas para interpretação e implementação da lógica paraconsistente. A proposição do algoritmo da Formiga Paraconsistente, que é o híbrido da Lógica Paraconsistente com a metaheurística de Colônia de Formigas é discutido no Capítulo 4, onde são apresentados os primeiros resultados experimentais. A aplicação da Formiga Paraconsistente para problemas de otimização em espaço de busca discreto é apresentado no Capítulo 5. Nesse capítulo esta implementada uma

aplicação da Formiga Paraconsistente para o problema de Restabelecimento de Sistemas de Distribuição. No capítulo 6 é implementada e discutida uma aplicação da Formiga Paraconsistente para problemas de otimização em espaço de busca contínuo. Nesse capítulo esta implementada uma aplicação da Formiga Paraconsistente para o problema de Despacho Econômico de Carga. No capítulo 7 são apresentadas e discutidas as conclusões, contribuições e sugestões de trabalhos futuros com os resultados dessa tese. A lista de publicações do autor, direta ou indiretamente relacionados ao tema desse trabalho estão relacionadas no Apêndice A.

Capítulo 2

Colônia de Formigas

A metaheurística de Otimização por Colônia de Formigas (*Ant Colony Optimization, ACO*) é uma estratégia de inteligência de bandos (*swarm intelligence*) que busca imitar o comportamento inteligente de uma colônia de formigas com o intuito de utilizá-lo na resolução de problemas de otimização combinatória.

Cada formiga individualmente, seguindo regras simples, são capazes de produzir uma solução viável para problemas complexos. A colônia como um todo, através da interação indireta de cada agente independente, é capaz de construir padrões complexos como solução de problemas. Todo este processo é realizado sem um controle central, através de um sistema de computação distribuído realizado por cada agente individual, mediado pela comunicação indireta através da modificação do ambiente.

As formigas seguem as seguintes regras de comunicação (Grassé , 1959):

- As formigas depositam um elemento químico, denominado feromônio, que pode ser percebido pelas outras formigas da colônia. As formigas usam a quantidade de feromônio para decidir que caminho seguir.

- As formigas só percebem o feromônio disponível localmente, ou seja, a formiga tem que alcançar o local onde está depositada a informação deixada pelas outras formigas da colônia.

Na metaheurística ACO, as formigas constroem soluções aleatórias, baseadas nas trilhas de feromônios e também nas informações heurísticas específicas dos problemas. O uso da trilha de feromônios para registrar a experiência acumulada pelas formigas durante a resolução de um problema torna a metaheurística ACO diferente das heurísticas tradicionais.

Este capítulo apresenta a classe de problemas denominados de Otimização Combinatória que é especialmente indicado para a aplicação da metaheurística de Colônia de Formigas. O trabalho que inspirou o desenvolvimento desta estratégia e um pseudo-código da implementação são apresentados e discutidos. Algumas variações da metaheurística ACO são apresentadas, entre elas a estratégia mais bem sucedida denominada $\mathcal{MAX} - \mathcal{MIN}$ Ant System (Stützle e Hoos , 2000). Esta última é utilizada com o híbrido da Lógica Paraconsistente que é proposto neste trabalho.

2.1 Problemas de Otimização Combinatória

Os problemas de otimização combinatória, classificados como problemas NP-Completos de acordo com a teoria de complexidade de algoritmos (Cormem *et al.* , 2002), têm como objetivo maximizar ou minimizar uma função definida sobre certo domínio finito. De uma forma geral, é fácil determinar uma possível solução do problema de otimização combinatória. Mesmo assim, é computacionalmente inviável testar todas as soluções possíveis para um determinado problema.

Um exemplo clássico de problema de otimização combinatória é o problema do caixeiro viajante (Jünger *et al.* , 1997), ilustrado na Figura 2.1. Este problema consiste em determinar um percurso de comprimento mínimo que passa por um conjunto de cidades (representadas pelo símbolo 'x' na Figura 2.1) sem repetir ou deixar de visitar qualquer cidade. O problema pode ser representado por um grafo $G = (V,A)$, no qual os vértices V representam as cidades e as arestas A representam o conjunto de estradas ligando as cidades. Se pensarmos num grafo completamente conectado, todas as cidades estarão diretamente ligadas por uma estrada. O problema do Caixeiro Viajante corresponde a encontrar um caminho Hamiltoniano no grafo G de comprimento mínimo. O caminho Hamiltoniano é o percurso fechado que visita exatamente uma vez cada um dos $v = |V|$ vértices de G. A distância d_{ij} entre as cidades i e j é o valor associado a cada aresta de G.

Ainda que seja fácil encontrar uma solução viável para o problema do caixeiro viajante, é computacionalmente inviável testar todos os roteiros possíveis para encontrar a solução ótima para o problema. O número de roteiros a ser testado fica em torno de $n!$, onde n representa o número de cidades no problema.

Este problema acontece em situações reais, como, por exemplo, o processo de soldagem de circuitos impressos, no qual deve-se determinar o menor o trajeto para que a máquina de solda conecte todos os circuitos da placa. Quanto menor for o trajeto da máquina de solda, mais rápida será a soldagem da placa e, por conseguinte mais placas poderão ser soldadas num intervalo de tempo.

2.1.1 Algoritmos de Aproximação

Na impossibilidade de implementar um algoritmo que encontre soluções exatas em tempo computacionalmente viável para problemas de otimização NP-Completos, são exploradas algumas estratégias que envolvem métodos de programação inteira, métodos probabilísticos e algoritmos de aproximação.

Entre os algoritmos de aproximação pode-se relacionar os métodos construtivos, os algoritmos de busca local e as metaheurísticas. Os métodos construtivos trabalham com soluções parciais, tentando extendê-las a fim de obter soluções completas. Os métodos de busca local promovem modificações em torno de soluções completas do problema com objetivo de melhorá-las. As metaheurísticas podem combinar aspectos dos métodos construtivos e de busca

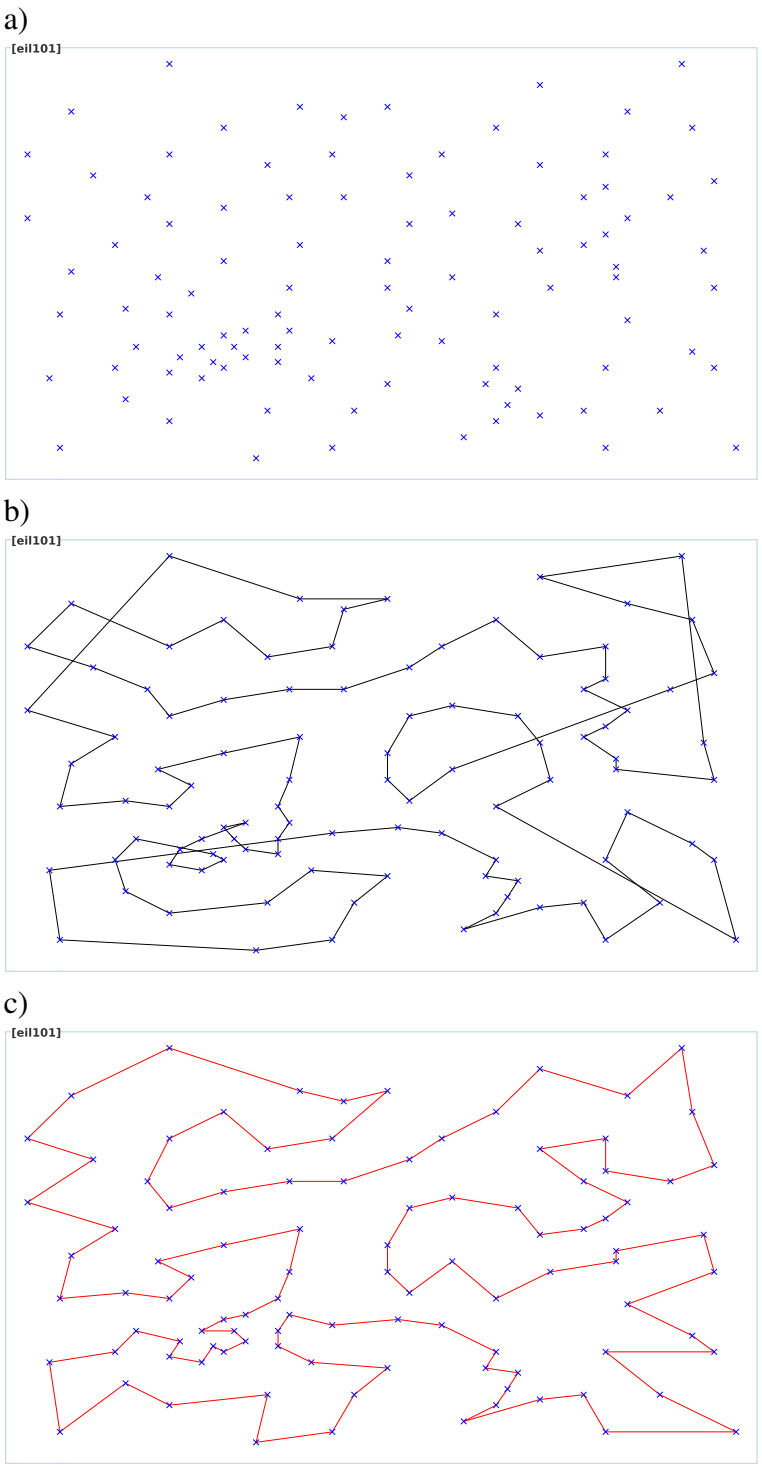


Figura 2.1: *Aplicação do algoritmo do melhor vizinho na instância eil101 da TSPLIB. a) Instância eil101 b) Solução encontrada usando o algoritmo do melhor vizinho e c) Melhor solução conhecida para instância eil101*

local a fim de produzir soluções dos problemas, com complexidade reduzida em relação aos algoritmos exatos e fornecendo, em geral, soluções viáveis de boa qualidade, mas sem garantia de qualidade.

Um algoritmo construtivo simples para resolver o problema do caixeiro viajante, por exemplo, é aquele que começando de uma cidade qualquer escolhe sempre a cidade mais próxima ainda não visitada até que a cidade original é alcançada. Este algoritmo é conhecido como heurística de construção de roteiro de vizinho mais próximo. Um exemplo de solução para o problema denominado eil101, obtido da TSPLIB[1], é apresentado nas Figuras 2.1a, 2.1b e 2.1c.

Na Figura 2.1a está apresentada a instância eil101. Pode-se observar na Figura 2.1b, que apresenta a solução usando o algoritmo de aproximação pelo vizinho mais proximo, algumas ligações distantes entre cidades não vizinhas, o que caracteriza soluções ruins para este problema. De um modo geral, esta estratégia produz soluções muito rápidas, mas as soluções não são de boa qualidade. Na Figura 2.1c está apresentada a melhor solução conhecida para a instância eil101.

2.1.2 Metaheurísticas

As metaheurísticas compõem aspectos dos métodos construtivos e dos algoritmos de busca local. Os métodos construtivos são utilizados nas metaheurísticas a fim de produzir soluções completas para os problemas. Os algoritmos de busca local são empregados nas metaheurísticas para melhorar as soluções encontradas. Os mecanismos de construção de soluções e de melhoramento das soluções encontradas são guiados por um conhecimento heurístico específico de cada problema.

A metaheurística de Otimização por Colônia de Formigas (ACO)(Dorigo e Stützle , 2002; Dorigo *et al.* , 2006a), por exemplo, é baseada em formigas artificiais que são procedimentos utilizados para construir soluções de um problema de otimização de forma probabilística. Através deste procedimento, as formigas adicionam interativamente componentes para a solução levando em conta a experiência acumulada pela colônia, representada pelas trilhas de feromônio e as informações heurísticas, específicas do problema modelado.

Este procedimento permite que as formigas artificiais construam uma grande variedade de soluções possíveis para o problema. Além disso, a experiência acumulada pela colônia associada à heurística do problema pode guiar o algoritmo a obter boas soluções. Desta forma, a metaheurística ACO é uma extensão do algoritmo de construção, apresentado anteriormente. Assim, a metaheurística ACO pode ser aplicada a qualquer problema de otimização para o qual é possível definir algum mecanismo de construção de soluções.

[1] Biblioteca de instâncias do problema do caixeiro viajante (Reinelt , 1991). Disponível no endereço eletrônico: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

2.2 Inspiração Biológica

A metaheurística ACO foi inspirada nas experiências realizadas em (Deneubourg *et al.* , 1990) e (Goss *et al.* , 1989), usando uma colônia de formigas. Para realizar este experimento, os pesquisadores construíram um ambiente com somente dois caminhos possíveis ligando o ninho à fonte de alimentação da colônia, conforme Figura 2.2.

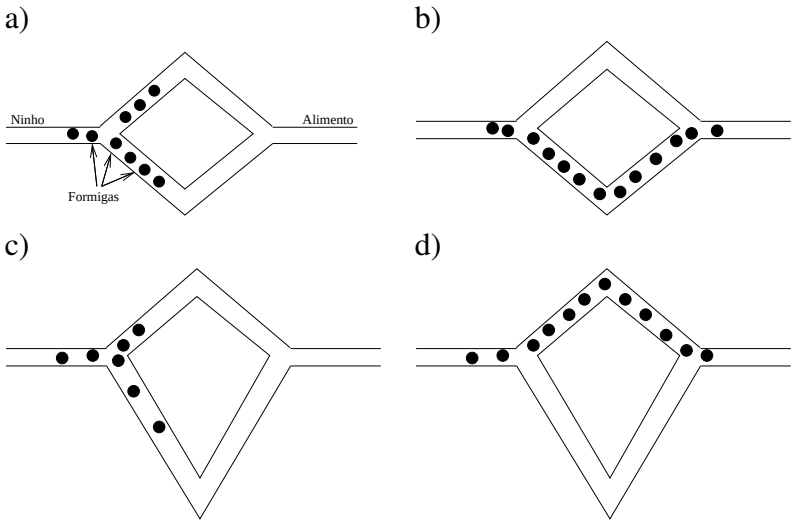


Figura 2.2: *Experimento real com formigas. a) situação inicial com as pontes do mesmo comprimento b) convergência das formigas para um dos caminhos c) mesmo experimento com pontes de comprimento diferente d) escolha do melhor caminho pelas formigas da colônia*

No trabalho de Deneubourg et al. (Deneubourg *et al.* , 1990), os dois caminhos que ligam o ninho a fonte de alimento tem o mesmo comprimento, conforme ilustrado nas Figuras 2.2a e 2.2b. Após um período inicial em que as formigas experimentam aleatoriamente os dois caminhos, uma das duas pontes apresenta uma maior concentração de feromônio e passa a ser escolhida pela maioria das formigas da colônia. A probabilidade que as formigas escolhecem a ponte superior ou a ponte inferior é 50% nos experimentos realizados. Goss et al. (Goss *et al.* , 1989) considerou uma variação do experimento original em que cada caminho tem um comprimento diferente e a formiga para chegar ao alimento e depois voltar ao ninho tem que escolher entre o maior ou o menor caminho, conforme ilustrado nas Figuras 2.2c e 2.2d. A observação experimental foi que, depois de uma fase transitória, a maioria das formigas passou a usar o caminho mais curto. Pode-se observar, também, que a probabilidade da colônia selecionar o caminho mais curto aumenta quando a diferença do comprimento dos caminhos é maior.

A seleção do caminho mais curto pode ser explicada em termos de realimentação positiva e do comprimento diferente dos caminhos, e é realizada através de uma forma indireta de comunicação mediada pela modificação local do ambiente através do depósito de feromônio nos caminhos. As formigas, quando vão do ninho para o alimento e vice-versa, depositam feromônio. Quando elas chegam num ponto de decisão, como a interseção da ponte esquerda ou direita,

```
1    metaHeuristicaACO() {
2        inicializacoes();
3        while !condicaoDeTermino() {
4            constroiSolucoes();
5            atualizaFeromonio();
6            [executaBuscaLocal();]
7        }
8    }
```

Figura 2.3: *Pseudo-código do algoritmo para metaheurística ACO*

elas tomam uma decisão probabilística, baseada na quantidade de feromônio que elas percebem nos dois ramos. Este comportamento tem um efeito autocatalítico porque o fato de escolher um caminho aumenta a probabilidade de que este caminho seja escolhido pela formiga e por outras formigas novamente no futuro. No início do experimento não há nenhum feromônio nos caminhos e, portanto as formigas vão do ninho para a comida escolhendo qualquer um dos caminhos com a mesma probabilidade.

As formigas que escolhem o caminho mais curto chegam primeiro ao alimento, devido ao comprimento diferente dos caminhos. Quando elas fazem o caminho de volta, tomam novamente uma decisão, se a trilha que elas passaram tiver uma quantidade maior de feromônio, esta terá uma probabilidade maior de ser escolhida. Durante este processo iterativo, o feromônio é depositado no caminho mais curto com uma taxa maior do que no caminho mais longo, tornando o caminho mais curto mais e mais atrativo até que todas as formigas passem a usá-lo.

2.3 Pseudo-código da metaheurística ACO

A partir do experimento descrito na seção 2.2, foi sugerido por Dorigo (Dorigo , 1992), em sua tese de doutorado, uma implementação de um algoritmo que simula o processo de busca por alimentos por uma colônia de formigas. O algoritmo originalmente proposto foi utilizado para resolução do problema de otimização combinatória denominado Problema do Caixeiro Viajante. Este trabalho original suscitou o desenvolvimento de diversas variações para a metaheurística de Colônia de Formigas. Os diversos trabalhos desenvolvidos utilizam instâncias do problema do caixeiro viajante como *benchmark*, para comparação dos resultados obtidos pelos algoritmos propostos.

A metaheurística ACO pode ser resumida da seguinte forma: as formigas da colônia, de forma assíncrona e concorrente, constroem soluções para o problema modelado através da definição de caminhos no grafo G que representa o problema. A escolha de cada formiga é feita através de uma decisão probabilística, levando-se em consideração a trilha de feromônios e uma informação heurística. Durante o processo de construção ou depois que a formiga finalizou um percurso no grafo G, as formigas da colônia podem avaliar a solução que foi construída e assim, depositar feromônio no caminho a fim de privilegiar a melhor solução encontrada pela colônia.

```
1     constroiSolucoes() {
2         inicializaFormigas();
3         while !obtiveramSolucoesCompletas() {
4             escolhaProximo();
5         }
6         calculaValorSolucao();
7     }
```

Figura 2.4: *Pseudo-código da rotina de construção de soluções das formigas da colônia*

De uma forma bem simplificada a metaheurística ACO pode ser apresentada através do pseudo-código da Figura 2.3. A rotina denominada *inicializações()*, na linha 2, compõe as operações que devem ser realizadas no início do processo de construção. Nesta rotina estão implementadas operações como criação das estruturas para representar as instâncias do problema e a colônia de formigas. A repetição (linhas 3 até 7), delimita as três operações fundamentais realizadas neste algoritmo até que a rotina *condiçãoDeTermino()* obtenha sucesso. A condição de término pode ser o tempo máximo determinado para busca, um número máximo de iterações pré-estabelecido ou então até que um parâmetro de qualidade da solução tenha sido alcançado. As operações fundamentais da metaheurística ACO são: a) *constroiSoluções()*, na linha 4, que constroi uma solução viável para cada formiga da colônia; b) *atualizaFeromônio()*, na linha 5, que realiza a atualização de estatísticas e estruturas utilizadas na implementação da metaheurística, entre elas a estrutura que representa a taxa de feromônio no caminho de soluções e c) *executaBuscaLocal()*, na linha 6, que executa alguma operação adicional como, por exemplo, algum algoritmo de Busca Local em torno de cada solução encontrada a fim de melhorá-la. Esta operação é opcional, mas pode determinar a qualidade e a velocidade de convergência da metaheurística para uma solução.

2.3.1 A construção de soluções pelas formigas

O algoritmo original para a metaheurística ACO foi denominado AS (*Ant System*) (Dorigo , 1992; Dorigo e Socha , 2006; Dorigo *et al.* , 1996). Para utilizar o AS, o problema a ser resolvido, deve ser modelado através de um grafo, conforme já foi discutido, onde os vértices representam componentes do problema e as arestas determinam o custo da transição entre os componentes do problema. Cada formiga k, ao se mover no grafo, espalha feromônio no seu caminho. A formiga possui a memória do caminho já percorrido e dos vértices já visitados, com o intuito de evitar passar pelo mesmo vértice mais de uma vez. Quando a formiga está no vértice i, ela escolhe o próximo vértice j usando a função de probabilidade, conforme Equação 2.1.

A rotina de construção de solução (*constroiSoluções()*) da metaheurística ACO está representado no pseudo-código da Figura 2.4. Esta rotina é executada de forma assíncrona e concorrente pelas formigas da colônia. As formigas são inicializadas (linha 2 da Figura 2.4) e posicionadas num componente inicial aleatório do problema. No caso do problema do caixeiro

viajante, cada formiga tem o seu roteiro iniciado com a informação de nenhuma cidade visitada e cada formiga é posicionada numa cidade inicial aleatória.

A repetição das linhas 3 até 5 da Figura 2.4 é executada até que todas as formigas da colônia obtenham uma solução completa para o problema. No caso do problema do caixeiro viajante, por exemplo, esta repetição termina depois que todas as formigas produzem um roteiro passando uma única vez por cada cidade.

A escolha do próximo componente de solução do problema é realizado na rotina *escolha-Proximo()* (linha 4 da Figura 2.4). Esta escolha é feita a partir da função de probabilidade definida na Equação 2.1.

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \tag{2.1}$$

Onde:

- k representa cada formiga da colônia.
- $\tau_{ij}(t)$ determina a quantidade de feromônio entre os vértices i e j na iteração t.
- η_{ij} representa a função heurística (específica para cada problema). Para o problema do caixeiro viajante, por exemplo, uma boa função heurística pode ser dada pelo inverso da distância entre as cidades, conforme Equação 2.2.

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{2.2}$$

- α representa a relevância da trilha de feromônio.
- β representa a relevância da informação heurística.
- N_i^k representa a vizinhança ainda não visitada a partir do vértice i.

Depois que as formigas da colônia constroem soluções completas é atribuída uma nota para cada solução encontrada (linha 6 da Figura 2.4).

2.3.2 A atualização do Feromônio

Para representar de forma mais aproximada o processo de comunicação utilizado pela colônia de formigas, na implementação da metaheurística ACO está previsto um mecanismo de evaporação do feromônio no espaço de solução. A Equação 2.3 mostra como é o processo de evaporação no algoritmo original de ACO.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{2.3}$$

Onde:

- $\tau_{ij}(t)$ determina a quantidade de feromônio na iteração t.
- ρ representa a taxa de evaporação de feromônio a cada iteração.
- m é a quantidade de formigas.

A taxa de evaporação ρ é determinada experimentalmente. O valor não pode ser muito alto para possibilitar a convergência das soluções das formigas, nem muito baixo para evitar a convergência precipitada de soluções sub-ótimas pelas formigas da colônia.

2.3.3 Variações da metaheurística ACO

Diversas variações deste algoritmo original foram desenvolvidas com o intuito de melhorar o desempenho e obter melhores resultados. Entre as variações temos os algoritmos como *Ant System Elitist* baseado numa estratégia elitista (Dorigo , 1992), baseado em *rank* (AS_{rank}) (Bullnheimer *et al.* , 1999), *Ant Colony System* (*ACS*) (Dorigo e Gambardella , 1997) e $\mathcal{MM}$AS, ou $\mathcal{MAX}-\mathcal{MIN}$ *Ant System* (Stützle e Hoos , 2000), (Dorigo *et al.* , 2006b).

A variação baseada em estratégia elitista consiste em premiar a melhor solução construída com uma quantidade adicional de feromônio. O AS_{rank} é, de certa forma, uma variação do algoritmo elitista, onde somente um rank das melhores soluções é permitido atualizar as trilhas de feromônio, com um taxa proporcional a qualidade da solução. E o *Ant Colony System*, que introduziu a possibilidade de atualização de feromônio local, além da atualização realizada ao final do processo de construção.

Dentre as variações propostas, uma das mais bem sucedidas implementações da metaheurística ACO é o algoritmo denominado $\mathcal{MAX}-\mathcal{MIN}$ *Ant System*. As principais inovações nesta implementação são que somente a melhor formiga atualiza a trilha de feromônios e que o feromônio está limitado pelas variáveis τ_{max} e τ_{min}, isto é, o feromônio não pode ultrapassar o limite superior a τ_{max}, nem o limite inferior a τ_{min}. A atualização do feromônio nesta implementação é dada pela Equação 2.4.

$$\tau_{ij}(t+1) = [(1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}^{best}]_{\tau_{min}}^{\tau_{max}} \tag{2.4}$$

Onde:

- $\Delta\tau_{ij}^{best}$ representa as arestas do caminho construído pela melhor formiga da colônia.
- $[x]_{\tau_{min}}^{\tau_{max}}$ representa que o valor x está restrito ao intervalo $[\tau_{max}, \tau_{min}]$.

Os valores de τ_{max} e τ_{min} podem ser definidos empiricamente ou através de valores específicos do problema. A melhor formiga pode ser a melhor formiga da iteração ou a melhor formiga geral.

2.4 Considerações Finais

Neste capítulo apresenta-se o histórico de desenvolvimento e a implementação da metaheurística para problemas de otimização combinatória denominada Colônia de Formigas.

As formigas da colônia deixam um componente químico denominado feromônio que é percebido pelas outras formigas. Esse feromônio define o modo de comunicação indireta das formigas quando estas percorrem um território em busca de alimento. De maneira similar, o algoritmo que imita a colônia realiza a atualização desses depósitos de feromônio, no espaço do problema que está sendo pesquisado pelas formigas. O resultado é que essa estratégia, relativamente simples tem apresentado bons resultados com problemas de computação difícil.

Diversas variantes do algoritmo original têm sido desenvolvidas no intuito de aprimorar alguns aspectos e tornar a metaheurística de Colônia de Formigas mais robusta e eficiente.

No próximo capítulo são apresentados os fundamentos teóricos e práticos da Lógica Paraconsistente, a fim de apresentar os conceitos necessários para implementação do sistema híbrido capaz de compor uma inteligência de bando através da metaheurística ACO associada à capacidade de raciocinar com situações de inconsistência e conhecimento incerto, a partir da Lógica Paraconsistente.

Capítulo 3

Lógica Paraconsistente

A lógica paraconsistente é uma lógica não-clássica que permite tratar situações de inconsistência, indefinições e conhecimentos parciais em processo de decisão. Através da lógica paraconsistente é possível raciocinar de forma adequada, tratando com as inconsistências inerentes das situações do mundo real.

Este capítulo apresenta um histórico das lógicas não-clássicas e dentre elas da lógica paraconsistente e métodos de interpretação da Lógica Paraconsistente considerando a sua estrutura teórica apresentada em trabalhos relevantes de pesquisas anteriores, como em (Abe , 1992; Da Costa *et al.* , 1990; IEE , 1987; Kleene , 1952; Pearl , 1993; SBC , 1996). A partir destas interpretações são desenvolvidos modos de aplicações que farão tratamento de conhecimento incerto, traduzindo estes conceitos teóricos em práticos.

3.1 Lógica Clássica

A lógica clássica é a base e o fundamento das ciências modernas. Foi através do trabalho de Aristóteles (384 a 322 a.C.) na Macedônia que teve o início da lógica clássica aristotélica. Aristóteles, nos seu trabalho, estabeleceu um conjunto de regras para as conclusões que se pode extrair de proposições sobre um conhecimento.

Para estabelecer relações entre as proposições, foi criada uma linguagem na qual estas proposições só podem ser qualificadas como verdadeiras ou falsas.

De maneira formal, pode-se descrever os princípios fundamentais da lógica clássica da seguinte forma: Seja p uma proposição qualquer e sejam os símbolos $=, \rightarrow, \neg, \wedge$ e $\vee$, representando respectivamente as operações de igualdade, implicação, negação, conjunção e disjunção. Então tem-se:

1. $p = p$ (princípio da igualdade), toda proposição é idêntica a si mesma.
2. $p \rightarrow p$ (princípio da identidade proposicional), toda proposição implica ela mesma.
3. $p \vee \neg p$ (princípio do terceiro excluído): de duas proposições contraditórias, uma delas é verdadeira.
4. $\neg(p \vee \neg p)$ (princípio da não-contradição): de duas proposições contraditórias, uma delas é falsa.

A linguagem formal da lógica clássica, sustentadas por estes princípios simples deu suporte para o desenvolvimento do pensamento lógico da humanidade.

3.2 Lógica Não-Clássica

Nem todas as situações do mundo real podem ser classificadas como verdadeiras ou falsas. Em certas situações é difícil ou até mesmo impossível estabelecer os limites entre o falso e o verdadeiro. Estes limites são muitas vezes indefinidos, incertos, ambíguos ou contraditórios.

Pesquisas em Inteligência Artificial (IA) (Russell e Norvig , 2004) visam incorporar e simular inteligência humana ou bioinspiradas em processos de tomada de decisão. Para tomar decisão, nem sempre é possível afirmar de forma categórica se um proposição é verdadeira ou falsa, como exige a lógica clássica.

Alguns novos estudos em IA propõem outros sistemas de tomada de decisão diferentes da lógica clássica, que tenham fundamentos não tão rígidos como da lógica clássica. Para responder de forma satisfatória as situações em que a lógica clássica não é apropriada, foram propostas as lógicas não-clássicas. As lógicas não-clássicas são aquelas que derrogam os princípios do terceiro excluído e da não-contradição para permitir indefinições, ambiguidades e contradições nos seus fundamentos.

Entre as várias lógicas denominadas lógicas não-clássicas tem-se a lógica paraconsistente, que derroga o princípio da não-contradição da lógica clássica e admite o tratamento de contradição nas suas formulações.

3.3 História da Lógica Paraconsistente

As Lógicas Paraconsistentes tiveram dois precursores, A. N. Vasil'év e J. Lukasiewicz, em 1910, em trabalhos não relacionados (da Silva Filho , 1999) e (da Silva Filho *et al.* , 2008). Os dois pesquisadores chamaram a atenção para o fato que alguns princípios da lógica aristotélica poderiam ser revisados. Todavia seus trabalhos são muito limitados, pois se referem apenas a análise crítica do princípio da contradição.

De fato, o primeiro lógico a formular um Cálculo Proposicional Paraconsistente dentro das normas atuais de rigor, foi o lógico polonês S. Jáskowski em 1948. Jáskowski foi levado a trabalhar em sua lógica de natureza paraconsistente chamada Lógica Discursiva, por influência de Lukasiewicz. Porém ele se limitou unicamente à elaboração de um Cálculo Proposicional Discursivo, sem estendê-lo a uma lógica de ordem superior, que sem muito rigor, é aquela que possui vários quantificadores de forma a poder manipular a igualdade.

O primeiro lógico a construir Cálculos Proposicionais, Cálculos Quantificacionais com ou sem Igualdade, Teorias de Descrições e Teorias de Conjuntos Paraconsistentes, foi Newton C. A. da Costa, no Brasil, na década de 50, (Da Costa , 1990, 1974; Da Costa e Abe , 1999;

Da Costa *et al.* , 1990, 1991).

A própria lógica de Jáskowski só foi desenvolvida a partir da década de 60, quando Da Costa e os discípulos poloneses de Jáskowski, L. Dubikajtis e J. Kotas trataram de ampliar o Cálculo Proposicional Discursivo, chegando a edificar Cálculos Quantificacionais Discursivos de Ordem Superior e Teorias de Conjuntos Discursivos; mais ainda, Da Costa e Kotas estenderam a idéia básica de Jáskowski, que definia a Lógica Discursiva a qualquer operador unário, em qualquer sistema, Modal ou não.

Após esses trabalhos iniciais, as Lógicas Paraconsistentes evoluíram tanto, que hoje já se tornaria difícil seguir a literatura a elas relacionada, em toda a sua extensão. Pelos mais variados motivos, as Lógicas Paraconsistentes se converteram em um grande campo de atividade de pesquisa, nos mais importantes centros do mundo.

3.4 Fundamentos da Lógica Paraconsistente

De uma forma geral, pode-se definir a lógica paraconsistente como aquela que é capaz de fundamentar sistemas dedutivos inconsistentes, ou seja, que admitam contradição entre os seus teoremas, mas que não sejam triviais. Evidentemente, se uma contradição pode ser aceita sob o prisma dessas lógicas, então nelas não pode ser válida a regra clássica segundo a qual, de uma contradição se pode deduzir qualquer proposição formulável.

Os enunciados demonstrados como verdadeiros em um teoria são denominados teoremas. Uma teoria é denominada trivial se todos as sentenças formuladas nesta teoria forem teoremas. Uma teoria é denominada consistente se entres os seus teoremas não existir um teorema que seja a negação de outro teorema. Caso exista contradição numa teoria, esta teoria é denominada inconsistente.

A lógica paraconsistente (paralelo, ao lado da consistência) é aquela que trabalha com os fundamentos de teorias inconsistentes e não triviais, ou seja, uma lógica paraconsistente é capaz de manipular sistemas inconsistentes de informações sem o risco de trivialização.

O papel da Lógica Paraconsistente, é ser um suporte formal para a existência de teorias que digam respeito a contradições e inconsistências do mundo real. A esse respeito, é oportuno citar Da Costa quando diz que:

> "...chamamos contradição ou inconsistência a qualquer par de proposições, uma das quais é a negação da outra. A Lógica Paraconsistente não exclui a possibilidade de que ambas as proposições de uma contradição seja verdadeira. Ela não exclui em particular, a existência de contradições verdadeiras reais, isto é, de contradições cujos componentes se refiram ao mundo real. Pensamos que a priori, não podemos eliminar essa possibilidade. Saber se há no mundo real contradições verdadeiras, constitui um assunto empírico somente decidível por meio das Ciências Empíricas" (Da Costa *et al.* , 1990).

3.4.1 A Lógica Paraconsistente Modelando Conhecimento Humano

A descrição de algumas situações do nosso cotidiano podem ser inconsistentes e indefinidas. Para simplificar o entendimento da proposta e o significado da Lógica Paraconsistente, realçando a importância da sua aplicação em situações onde é difícil aplicar a Lógica Clássica, apresenta-se um exemplo.

EXEMPLO: Considere uma reunião de departamento para decidir pela compra de um novo equipamento. Se a opinião de todos os membros do departamento é unânime, a decisão pode ser tomada de forma assertiva, como na lógica clássica. Na maioria das vezes as opiniões e interesses são contraditórios. Alguns membros do departamento podem até mesmo não ter opinião formada, gerando indefinições. A análise de todas as opiniões, sejam elas contraditórias, indefinidas, contra e a favor a compra do novo equipamento para o departamento pode originar a busca de novas informações que possibilitem a tomada de decisão. Observa-se neste exemplo que a decisão tomada não segue a lógica clássica, pois é baseada nas evidências trazidas pelas diferentes opiniões.

O comportamento humano apresentado neste exemplo pode ser modelado usando a lógica paraconsistente. A lógica Paraconsistene é mais completa e adequada para tratar situações reais em que ocorrem inconsistências e indefinições. E por isso pode ser aplicada em sistemas de controle e em processos de tomada de decisão.

Considerando que as situações de inconsistências são frequentes, as Lógicas Paraconsistente estão sendo aplicadas nos desenvolvimentos de projetos para aplicações em diversas áreas de conhecimento. Os resultados obtidos nos estudos da Lógica Paraconsistente Anotada (AAA , 1987; Abe , 1992; Da Costa *et al.* , 1990; IEE , 1987) demonstram que é plenamente viável a aplicação da Lógica Paraconsistente Anotada em situações de inconsistências e que os resultados podem ser interagidos com a Lógica Convencional ou Binária ou ainda com outros tipos de Lógicas Não-Clássicas.

3.4.2 Lógica Proposicional Paraconsistente Anotada

Para ser possível a elaboração de um algoritmo ou a implementação de circuitos que funcionem com a Lógica Paraconsistente é apresentado um resumo da linguagem formal que compõe a Lógica Proposicional Paraconsistente Anotada P_τ (da Silva Filho , 1999; da Silva Filho *et al.* , 2008). A teoria aqui apresentada, de forma sucinta contém as principais definições, e é suficiente para que se possa aplicar os principais conceitos da Lógica Paraconsistente e traduzi-los através de um algoritmo prático e em sinais elétricos através dos circuitos eletrônicos. Em (Abe , 1992) é feito um amplo estudo destas lógicas onde o autor demonstra teoremas de correção e completeza para os cálculos Q_τ (Lógica de Primeira Ordem).

A Lógica Paraconsistente recebe a seguinte definição: Seja T uma teoria fundada sobre uma lógica L, e suponha-se que a linguagem de L e T contenha um símbolo para a negação. A teoria T diz-se inconsistente se ela possuir teoremas contraditórios, isto é, tais que uma é a negação

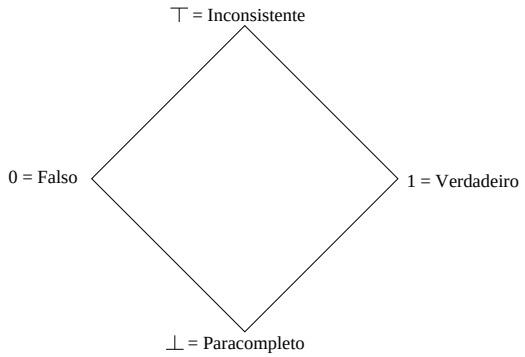


Figura 3.1: *Reticulado finito através do diagrama de hasse*

da outra; caso contrário, T diz-se consistente. A teoria T diz-se trivial se todas as fórmulas de L forem teoremas de T, caso contrário, T chama-se não trivial. No mundo real, as inconsistências são importantes e não podem ser desprezadas porque são as informações que trazem fatos relevantes modificando, por vezes, completamente o resultado da análise. A existência da inconsistência é que induz ao sistema promover buscas procurando novas e esclarecedoras informações com consultas a outros informantes, para se obter uma conclusão mais real e confiável. As principais definições da Lógica Proposicional Paraconsistente Anotada P_τ são apresentadas a seguir de forma sucinta.

Fixando um reticulado finito denominado de Reticulado de Valores Verdades, $\tau =< |\tau|, \leq >$, τ é reticulado, se:

1. $\forall x, x \leq x$ (reflexividade);
2. Se $x \leq y$ e $y \leq x \Rightarrow x = y$ (anti-simétrica);
3. Se $x \leq y$ e $y \leq z \Rightarrow x \leq z$ (transitividade);
4. $\forall x, y \in |\tau|$, existe um supremo de x e y que é denotada por $x^{\vee}y$;
5. $\forall x, y \in |\tau|$, existe um ínfimo de x e y que é denotada por $x^{\wedge}y$;

Os seguintes símbolos são associados a este reticulado:

- $\bot \Rightarrow$ indica o mínimo de τ;
- $\top \Rightarrow$ indica o máximo de τ;

A representação de um reticulado finito se faz usualmente através do diagrama de Hasse, de acordo com a Figura 3.1.

Fixando um operador: $\neg|\tau| \Rightarrow |\tau|$ que terá intuitivamente o significado da negação da Lógica P_τ. Logo:

$\neg(1) = 0; \neg(0) = 1; \neg(\bot) = \top; \neg(\top) = \bot$

A linguagem de P_τ é composta do seguinte vocabulário:

1. Variáveis proposicionais: $p_1, p_2, p_3, \ldots, p_n, \ldots$

2. Conectivos: $\neg$ (negação), $\wedge$ (conjunção ou AND), $\vee$ (disjunção ou OU) e $\rightarrow$ (implicação);

3. Constantes anotacionais: $\mu, \lambda, \theta, \ldots$;

4. Símbolos auxiliares: $(,), \ldots$.

As fórmulas de P_τ são definidas pela seguinte definição indutiva generalizada:

1. Se p é uma variável proposicional e λ é uma constante anotacional, então p_λ é uma fórmula atômica;

2. Se A é uma fórmula, então $\neg A$ é uma fórmula;

3. Se A, B são fórmulas, então $A \wedge B, A \vee B$, e $A \rightarrow B$ são fórmulas;

4. Uma expressão é uma fórmula se e somente se for obtida pela aplicação de umas das regras anteriores.

A fórmula $\neg A$ é lida como "negação de A".
A fórmula $A \wedge B$ é lida como "conjunção de A e B".
A fórmula $A \vee B$ é lida como "disjunção de A e B".
A fórmula $A \rightarrow B$ é lida como "implicação de B por A".

Intuitivamente uma fórmula atômica p_μ é lida como: "Creio na proposição p com grau de crença de no máximo μ, ou até $\mu(\leq \mu)$".

Se p é uma letra proposicional e $\mu \in |\tau|$, então uma fórmula atômica do tipo $\neg^k p_\mu$ onde $k > 0$, denomina-se hiper-literal ou simplesmente literal. As demais fórmulas denominam-se fórmulas complexas. O estudo da semântica das lógicas P_τ é apresentado de modo resumido da seguinte forma. Uma interpretação relativa às Lógicas Anotadas P_τ é uma função $I : P \rightarrow |\tau|$, onde o P é o conjunto das variáveis proposicionais. A cada interpretação I, associa-se uma valoração: $V_I : F \rightarrow 0, 1$, onde F é o conjunto de todas as fórmulas. A valoração, V_I, é definida indutivamente por:

1. Se p é uma letra proposicional, então:

$$V_I(p_\mu) = 1 \Longleftrightarrow I(p) \geq \mu$$
$$V_I(p_\mu) = 0 \Longleftrightarrow I(p) \leq \mu$$
$$V_I(\neg^k p_\mu) = V_I(\neg^{k-1} p_{\sim\mu}) \text{ onde, } k \geq 1$$

2. Se A e B são fórmulas quaisquer, então:

$$V_I(A \rightarrow B) = 1 \Longleftrightarrow V_I(A) = 0 \text{ ou } V_I(B) = 1$$
$$V_I(A \wedge B) = 1 \Longleftrightarrow V_I(A) = 1 \text{ e } V_I(B) = 1$$
$$V_I(A \vee B) = 1 \Longleftrightarrow V_I(A) = 1 \text{ ou } V_I(B) = 1$$

Definições:

Uma interpretação relativa à $P_\tau, I : P \rightarrow |\tau|$, se diz inconsistente se existir $p \in P$ e $\mu \in |\tau|$ tal que: $V_I(p_\mu) = 1 = V_I(\neg p_\mu)$.

Uma interpretação relativa à $P_\tau, I : P \rightarrow |\tau|$, se diz trivial se existir $p \in P$ e $\mu \in |\tau|$ tal que: $V_I(p_\mu) = 0$.

Uma interpretação relativa à $P_\tau, I : P \rightarrow |\tau|$, se diz paraconsistente se for inconsistente e não trivial.

A Lógica P_τ se diz Paraconsistente se ela admitir uma interpretação paraconsistente.

3.4.3 Representação dos Reticulados da Lógica Paraconsistente Anotada

Através de uma análise intuitiva da Lógica Paraconsistente Anotada, a fórmula atômica p_μ que é lida como: Creio na proposição p com grau de crença de no máximo μ, ou até $\mu(\leq \mu)$ considera o grau de crença como sendo uma constante anotacional do reticulado. Isto nos induz a afirmar que cada grau de crença atribuído à proposição é um valor que está contido no conjunto de valores composto pelas constantes anotacionais do reticulado $\{\top, V, F, \bot\}$.

As anotações neste reticulado são consideradas multivaloradas e seguem as regras determinadas pelo diagrama de Hasse, da Figura 3.1. As proposições são acompanhadas de anotações que por sua vez atribuem o grau de crença correspondente a cada variável proposicional. Cada sentença proposicional, que por abuso de linguagem denomina-se de proposição, virá acompanhada de um grau de crença que determinará a conotação de Verdade ou de Falsidade da fórmula.

- $p_\top$ = A anotação ou grau de crença atribui uma conotação de inconsistência à proposição p.
- p_1 = A anotação ou grau de crença atribui uma conotação de verdade à proposição p.
- p_0 = A anotação ou grau de crença atribui uma conotação de falsidade à proposição p.
- $p_\bot$ = A anotação ou grau de crença atribui uma conotação de indefinição à proposição p.

Por exemplo, um sistema de controle de um robô que esteja funcionando com sinais provenientes de dois sensores, para cumprir uma tarefa de colocar uma peça sobre uma mesa, deve ter a certeza nas suas informações de que a mesa se encontra no lugar correto. Com as informações e utilizando as notações da Lógica Paraconsistente Anotada, tem-se que:

p = A mesa está no lugar correto e μ = Grau de Crença;

As situações no mundo real possíveis de acontecer seriam:

- $p_\top$ = A mesa está e não está no lugar correto (um sensor detecta e o outro não).
- p_1 = A mesa está no lugar correto (os dois sensores detectam).

- p_0 = A mesa não está no lugar correto (nenhum dos sensores detecta).
- $p_\perp$ = Nada se pode afirmar, i.e., indefinição (as amplitudes dos sinais vindos dos dois sensores são insuficientes para detecção).

Na situação de inconsistência, o robô vai procurar a informação de outro sensor que pode estar melhor localizado, se movimentar lateralmente ou mesmo lançar mão dos dados armazenados na sua base de conhecimento que atuaria como sinal decisório nas contradições. Nas situações de verdade e falsidade a ação a ser tomada não oferece nenhuma discussão. Quando a situação é de indefinição a ação deve ser no sentido de aumentar a sensibilidade dos sensores como: limpeza das lentes, focalização, ajuste da potência da alimentação elétrica, etc.

Neste exemplo de aplicação da Lógica Paraconsistente Anotada, que faz uso do reticulado, foram consideradas quatro situações, das quais duas delas, a inconsistência e a indefinição, são estranhas à Lógica Clássica no contexto que foram empregadas. Os Graus de Crença, correspondentes as constantes anotacionais do reticulado, são quatro e como visto, os seus valores foram relacionados com as quatro situações expostas.

3.4.4 Lógica Paraconsistente Anotada de Dois Valores - LPA2v

A Lógica Paraconsistente Anotada apresentada até agora traz anotação com uma única componente, onde para cada proposição, associa-se uma única constante anotacional ou grau de crença do reticulado. Em (Da Costa *et al.* , 1990) é apresentada uma nova proposta de reticulado que traz dois valores na anotação. A aplicação da Lógica Paraconsistente Anotada com este reticulado é uma proposta muito interessante do ponto de vista prático, porque permite maior controlabilidade, mais possibilidades computacionais, acarretando com isso, uma melhora significativa no desempenho dos programas computacionais e circuitos de controle que vão ser aplicados nos sistemas. As anotações podem ser interpretadas como evidências e, portanto, quando o circuito recebe informações contraditórias estas evidências possuem um papel importante na tomada de decisão.

Aplicando-se o raciocínio evidencial da Lógica Paraconsistente Anotada proposta, dois valores são agora associados a uma anotação do reticulado, denominando esta lógica de Lógica Paraconsistente Anotada de Dois Valores, LPA2v. O primeiro valor da anotação representa a evidência favorável à proposição p, e o segundo valor representa a evidência contrária à proposição p. Denomina-se de crença a evidência favorável e de descrença a evidência contrária. O grau de crença é denotado pela letra μ_1 e o grau de descrença é denotado pela letra μ_2. Com estas considerações, cada constante anotacional do reticulado é agora representada pelo par (μ_1, μ_2). Um reticulado de Hasse de anotação com dois valores é apresentado conforme figura 3.2.

As definições para as anotações e para o operador $\neg$ vem a seguir. Se p é uma fórmula básica e operador $\neg : |\tau| \rightarrow |\tau|$ é definido como: $\neg[(\mu_1, \mu_2)] = (\mu_2, \mu_1)$ onde $(\mu_1, \mu_2) \in \{x \in \Re | 0 \leq x \leq 1\}$, considera-se (μ_1, μ_2) como anotação de p. As coordenadas μ_1 e μ_2 podem ser lidas como o

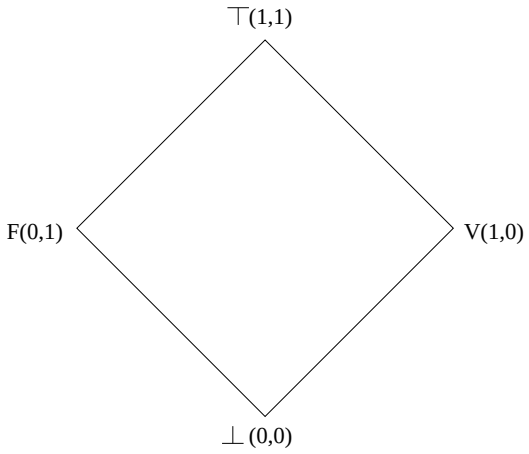


Figura 3.2: *Diagrama de Hasse da LPA2v*

grau de crença atribuído a p e o grau de descrença atribuído a p, respectivamente. Os valores dos graus de crença e descrença são completamente independentes. Portanto, como um valor não depende absolutamente do outro é possível qualquer variação dos valores no intervalo real fechado entre 0 e 1.

3.4.5 Outra interpretação da LAP2v

Em (Martins , 2003; Martins *et al.* , 2007) utiliza-se o conceito de Quadrado Unitário do Plano Cartesiano (QUPC) para formular uma nova proposta de interpretação para a lógica paraconsistente de dois valores (LPA2v).

Nesta proposta para a lógica paraconsistente de dois valores, uma interpretação é uma função $I : P \rightarrow |\tau|$ e que para cada interpretação I deve-se associar uma valoração $V_I : F \rightarrow [0,1] \times [0,1]$, sendo que F é o conjunto de todas as fórmulas que podem ser visualizadas no QUPC.

No QUPC, a abcissa, com valores válidos no intervalo real fechado [0,1] representa o grau de crença μ de uma proposição e o eixo das ordenadas, com valores no mesmo intervalo [0,1] representa o grau de descrença λ da proposição. A Figura 3.3a apresenta os pontos notáveis do QUPC.

O segmento $\overline{FV}$ do QUPC é denominado de *linha perfeitamente consistente* (LPC) e o segmento $\overline{\top\bot}$ denomina-se *linha perfeitamente inconsistente* (LPI), conforme ilustrado na Figura 3.3a. Na LPC, dado o valor de crença, o valor de descrença é o seu complemento com relação à unidade,

$$\mu + \lambda - 1 = 0$$

Da mesma forma, na LPI, para cada valor de crença corresponde um valor de descrença com a mesma intensidade,

$$\mu - \lambda = 0$$

Dado um par (μ, λ) associado a uma proposição, define-se o *grau de incerteza* (GI) pela

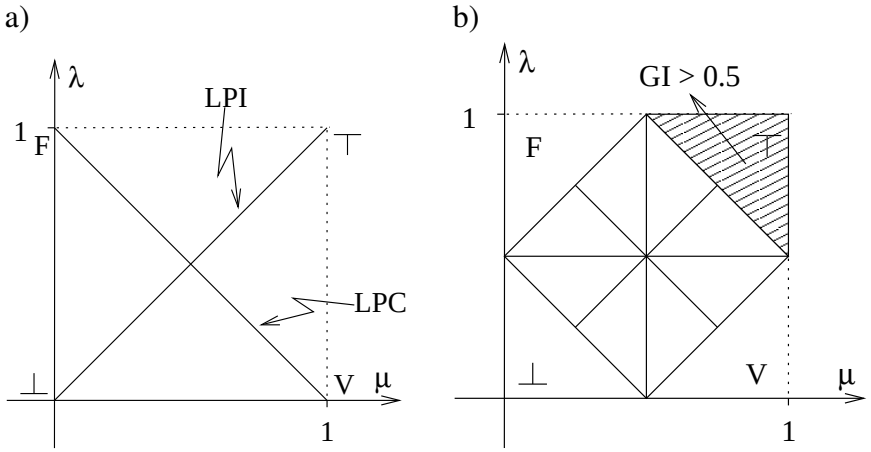


Figura 3.3: *Quadrado Unitário do Plano Cartesiano. a) Pontos notáveis e a linha perfeitamente inconsistente, LPI e a linha perfeitamente consistente, LPC do QUPC. b) Delimitações de regiões no QUPC com a inclusão de novos pontos notáveis e a identificação da região totalmente inconsistente*

Equação 3.1 e o *grau de certeza* (GC) pela Equação 3.2.

$$GI = \mu + \lambda - 1 \tag{3.1}$$

$$GC = \mu - \lambda \tag{3.2}$$

Através das equações do GC e GI pode-se verificar que, à medida que um par ordenado (μ, λ) do QUPC se distancia da LPI, em direção ao par $(1,0)$ ocorre um aumento do grau de certeza da proposição associada, até chegar ao seu valor máximo que é 1, situado no ponto V. De forma análoga, conforme um par ordenado (μ, λ) da QUPC se distancia da LPI, em direção ao par $(0,1)$, ocorre um aumento do grau de incerteza da proposição associada até chegar ao seu valor máximo que é 1, situado no ponto F.

Em (Martins , 2003) sugere-se ainda a inclusão de mais quatro pontos notáveis no QUPC. O ponto de coordenada (1,0.5), denominado *quase-verdade*, qV, o ponto de coordenada (0.5,1), denominado *quase-falso*, qF, o ponto de coordenada (0,0.5) denominado *quase-não-verdade*, $q\sim V$ e o ponto de coordenada (0.5,0), denominado *quase-não-falso*, $q\sim F$, conforme ilustrado na Figura 3.4. Com o acréscimo deste pontos, o QUPC fica dividido em novas regiões que recebem denominações de acordo com a proximidade aos pontos extremos do reticulado. As regiões do QUPC são assim nomeadas:

- $\top$: inconsistente
- $\bot$: paracompleto
- F: falso
- V: verdade

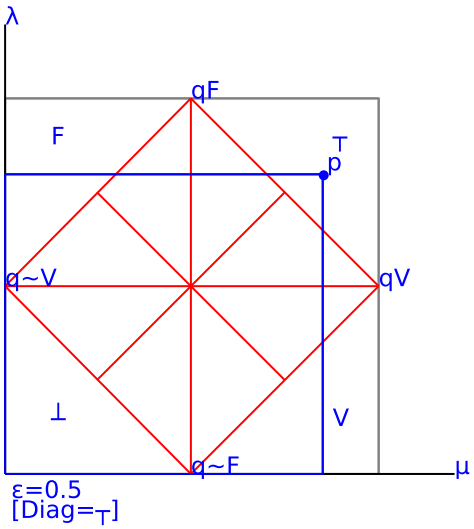


Figura 3.4: *Exemplo de identificação de um ponto na região* $\top$ *= inconsistente no QUPC*

- $V \rightarrow qV$: verdade tendendo a quase-verdade
- $\top \rightarrow qV$: inconsistente tendendo a quase-verdade
- $F \rightarrow qF$: falso tendendo a quase-falso
- $\top \rightarrow qF$: inconsistente tendendo a quase-falso
- $F \rightarrow q{\sim}V$: falso tendendo a quase-não-verdade
- $\bot \rightarrow q{\sim}V$ paracompleto tendendo a quase-não-verdade
- $V \rightarrow q{\sim}F$: verdade tendendo a quase-não-falso
- $\bot \rightarrow q{\sim}F$: paracompleto tendendo a quase-não-falso

Intuitivamente, as regiões no QUPC representam o valor associado a uma proposição $p_{\mu\lambda}$ dados os valores do grau de crença e do grau de descrença da proposição. A região $\top$, por exemplo é a região no QUPC onde $GI > 0.5$, conforme ilustrado na Figura 3.3b.

Dado uma proposição $p_{\mu\lambda}$, o valor da lógica paraconsistente associado a p é determinado pela região do QUPC no qual o ponto, cujas coordenadas são $p = (\mu, \lambda)$ está incluído. Na Figura 3.4 por exemplo, os valores de μ e λ são 0.85 e 0.80, repectivamente e o valor lógico de p é inconsistente ($\top$). O algoritmo da Figura 3.5 apresenta como obter um diagnóstico a partir dos valores de μ e λ de uma proposição p qualquer. Dado o valor do grau de crença μ e o valor do grau de descrença λ, a rotina retorna o estado lógico paraconsistente, correspondente

A partir desta interpretação da lógica paraconsistente de duas variáveis, define-se as operações lógicas de negação, conjunção e disjunção. A operação de negação, por exemplo, é obtida através da seguinte fórmula. Dada uma proposição $P(\mu, \lambda)$, a negação nesta interpretação da lógica paraconsistente é conforme a Equação 3.3.

$$\neg P(\mu, \lambda) = (1-\mu, 1-\lambda) \tag{3.3}$$

Sejam $P(\mu_P, \lambda_P)$ e $Q(\mu_Q, \lambda_Q)$, duas proposições quaisquer com seus graus de crença e descrença. A aplicação do conectivo lógico $\vee$(ou) através do QUPC e do grau de certeza e do grau de incerteza das proposições P e Q é dado pela Equação 3.4.

$$P \vee Q = (\max(\mu_P, \mu_Q), \min(\lambda_P, \lambda_Q)) \tag{3.4}$$

De forma similar a aplicação do conectivo lógico $\wedge$(e) através do QUPC e do grau de certeza e do grau de incerteza das proposições P e Q é dado pela Equação 3.5.

$$P \wedge Q = (\min(\mu_P, \mu_Q), \max(\lambda_P, \lambda_Q)) \tag{3.5}$$

Aplica-se o algoritmo da Figura 3.5 para determinar o estado lógico paraconsistente resultante da aplicação de cada operador lógico. Os parâmetros da função *diagnóstico* do algoritmo são o grau de crença μ e o grau de descrença λ. No algoritmo, o *grau de certeza*, GC e o *grau de incerteza*, GI são calculados conforme as equações Equação 3.2 e Equação 3.1. Os valores C_1, C_2, C_3 e C_4 representam respectivamente o valor superior de controle de certeza, valor inferior de controle de certeza, valor superior de controle de incerteza e valor inferior de controle de incerteza. Estes valores são utilizados nas condições do algoritmo para identificar as regiões relacionadas a cada estado lógico.

O sistema híbrido apresentado neste capítulo não utiliza das operações lógicas. Estas questões são relevantes para demonstrar que a lógica paraconsistente pode extender a lógica clássica, isto é, toda fórmula válida na lógica clássica pode ser extendida e interpretada usando a lógica paraconsistente.

3.4.6 A Extensão da Lógica Paraconsistente Anotada de Três Variáveis

No trabalho de Martins (Martins , 2003), apresenta-se uma extensão para a interpretação da lógica paraconsistente anotada de duas variáveis, LPA2v, acrescentando-se mais uma dimensão ao reticulado, denominado *grau de especialidade*. A ideia é considerar a opinião de um especialista na determinação do valor lógico associado as proposições.

Desta forma é acrescido mais um eixo ao plano cartesiano da LPA2v, para obtermos um cubo analisador, conforme ilustrado nas Figuras 3.7, 3.8 e 3.9.

Dos *especialistas* espera-se o mínimo de indecisões, inconsistência e desconhecimento. Dado uma proposição ao especialista espera-se obter uma decisão coerente e determinada, conforme a lógica clássica. Dos *neófitos* admite-se qualquer decisão, dado sua inexperiência. No trabalho original de Martins (Martins , 2003), considera-se especialista quando $\varepsilon = 1$ e quando $\varepsilon = 0$ temos o neófito.

Com a variação do grau de especialidade, temos uma deformação no plano que identifica as

```
1  diagnostico (μ, λ){
2    dc ← μ − λ
3    du ← (μ + λ) − 1
4    c1 ← 0.5
5    c2 ← −0.5
6    c3 ← 0.5
7    c4 ← −0.5
8    if dc ≥ c1 return V
9    if |dc| ≥ |c2| return F
10   if du ≥ c3 return ⊤
11   if |du| ≥ |c4| return ⊥
12   if dc = 0 and du = 0 return ?
13   if dc ≥ 0 and dc < c1 and du ≥ 0 and du < c3 {
14     if du < c3/c1 ∗ dc
15      return V → qV
16     return ⊤ → qV
17   }
18   if dc > c2 and dc ≤ 0 and du ≥ 0 and du < c3 {
19     if |du| < |c3/c2 ∗ dc|
20      return F → qF
21     return ⊤ → qF
22   }
23   if dc > c2 and dc < 0 and du > c4 and du ≤ 0 {
24     if |du| < |c4/c2 ∗ dc|
25      return F → q∼V
26     return ⊥ → q∼V
27   }
28   if dc ≥ 0 and dc < c1 and du > c4 and du ≤ 0 {
29     if du ≥ c4/c1 ∗ dc
30      return V → q∼F
31     return ⊥ → q∼F
32   }
33 }
```

Figura 3.5: *Pseudo-código da rotina que faz o diagnóstico usando o QUPC.*

```
1  | diagnostico (μ, λ, ε){
2  |   dc ← μ − λ
3  |   du ← (μ + λ) − 1
4  |   c1 ← ε
5  |   c2 ← −ε
6  |   c3 ← 1 − ε
7  |   c4 ← ε − 1
.  |   (...)
33 | }
```

Figura 3.6: *Pseudo-código do diagnóstico usando o QUPC para LPA3v, considerando o parâmetro adicional de especialidade* ε

diversas regiões de diagnóstico (valores lógicos) da lógica paraconsistente. Por exemplo, para $\varepsilon = 0.25$ temos o cubo e as regiões da lógica paraconsistente, conforme ilustrado na Figura 3.7. A região *verdade* e a região *falso* são maiores que as regiões *inconsistente* e *paracompleto*. Para $\varepsilon = 0.5$, temos as regiões dos estados da lógica paraconsistente exatamente igual aquelas da LPA2v. Para $\varepsilon = 0.75$, as regiões *inconsistente* e *paracompleta* são maiores que *falso* e *verdade*.

O mesmo valor do grau de crença e do grau de descrença de um proposição, produz diferentes estados lógicos resultantes, em função do grau de especialidade. Assim, para $\varepsilon = 0.25$, $\mu = 0.9$ e $\lambda = 0.4$, o estado paraconsistente resultante é V, conforme ilustrado na Figura 3.7 [1]. Para $\varepsilon = 0.5$, $\mu = 0.9$ e $\lambda = 0.4$, o estado paraconsistente resultante é ainda $V \rightarrow qV$, conforme ilustrado na Figura 3.8. Para $\varepsilon = 0.75$, $\mu = 0.9$ e $\lambda = 0.4$, o estado paraconsistente resultante é $\top$, Figura 3.9.

O algoritmo da Figura 3.5 é facilmente adaptado para considerar um parâmetro adicional do grau de especialidade. As mudanças no algoritmo estão representadas na Figura 3.6. Basicamente a função de diagnóstico para LPA3v inclui um novo parâmetro de especialidade ε. Este valor é utilizado para determinar os valores de controle superior e valores de controle inferior usados para delimitar as regiões dos estados lógicos. Os novos valores são $C_1 \leftarrow \varepsilon$, $C_2 \leftarrow -\varepsilon$, $C_3 \leftarrow 1 - \varepsilon$ e $C_4 \leftarrow \varepsilon - 1$. As outras linhas do algoritmo de diagnóstico permanecem as mesmas.

A partir da fixação de um valor de especialidade ε, podemos usar o mesmo raciocínio usado para a aplicação dos operadores lógicos de negação $\neg$, de disjunção $\vee$ e conjunção $\wedge$, usados para LPA2v, conforme as Equação 3.3, Equação 3.4 e Equação 3.5.

Adota-se, nesse trabalho, uma interpretação simétrica ao trabalho original, com o intuito de capturar o aprendizado das formigas da colônia durante o processo de construção de soluções, e dessa forma, controlar o processo de convergência da colônia para a melhor solução no final de uma série de tentativas, conforme será detalhado no Capítulo 4.

[1] Um simulador tridimensional do cubo analisador da lógica paraconsistente está disponível em www.bcc.unifal-mg.edu.br/~luizedu/lpa3v.jnlp para melhor visualização.

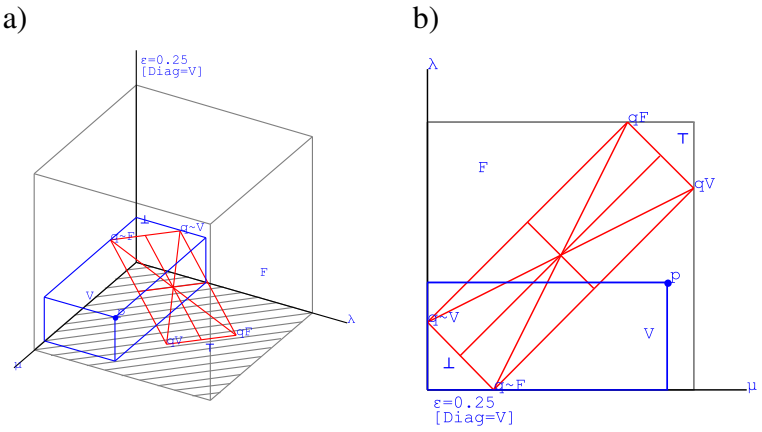


Figura 3.7: *Cubo analisador da lógica paraconsistente anotada de três variáveis. Determinação do valor lógico para* $\mu = 0.9$ *e* $\lambda = 0.4$ *e* $\varepsilon = 0.25$*. a) representação no cubo analisador b) projeção no plano* $\mu \times \lambda$

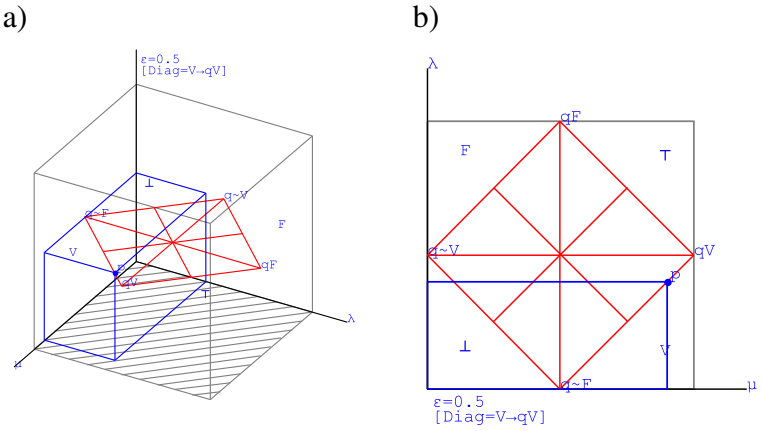


Figura 3.8: *Cubo analisador da lógica paraconsistente anotada de três variáveis. Determinação do valor lógico para* $\mu = 0.9$ *e* $\lambda = 0.4$ *e* $\varepsilon = 0.50$*. a) representação no cubo analisador b) projeção no plano* $\mu \times \lambda$

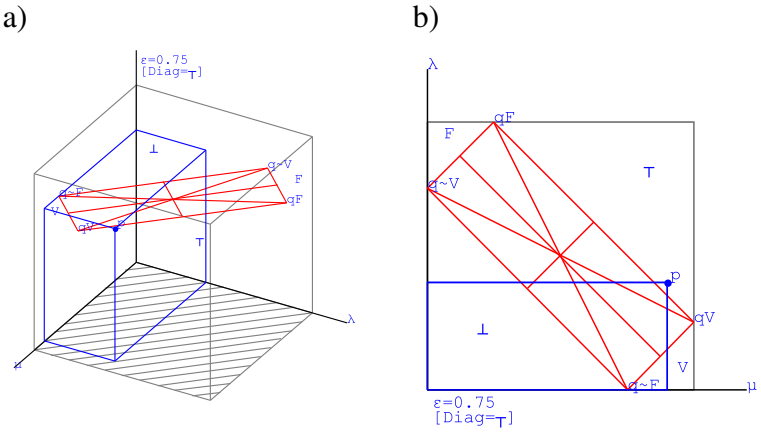


Figura 3.9: *Cubo analisador da lógica paraconsistente anotada de três variáveis. Determinação do valor lógico para* $\mu = 0.9$ *e* $\lambda = 0.4$ *e* $\varepsilon = 0.75$. *a) representação no cubo analisador b) projeção no plano* $\mu \times \lambda$

3.5 Considerações Finais

Este capítulo apresenta a lógica não-clássica que é capaz de tratar com situação de incerteza e inconsistência que ocorrem comumente em problemas de tomada de decisão, denominada Lógica Paraconsistente. Alternativas para implementação desta teoria são apresentadas a fim de possibilitar a utilização da Lógica Paraconsistente num híbrido com a metaheurística de Colônia de Formigas.

A proposição da Formiga Paraconsistente, que é um algoritmo híbrido da metaheurística de Colônia de Formigas com a lógica paraconsistente é apresentada no próximo capítulo.

Capítulo 4

Proposição da Formiga Paraconsistente

Neste capítulo é apresentada a proposição do algoritmo da Formiga Paraconsistente, que é o sistema híbrido que compõe a metaheurística de Colônia de Formigas com a Lógica Paraconsistente. O principal objetivo do uso da Lógica Paraconsistente é capturar o aprendizado das formigas durante o processo de construção de soluções visando melhorar o processo de tomada de decisão das formigas.

O Sistema proposto é aplicado a uma instância de *benchmark* de otimização, denominado Problema do Caixeiro Viajante, a fim de verificar a possibilidade e eficiência do híbrido proposto. Os resultados obtidos são apresentados e discutidos neste capítulo.

4.1 A Proposta da Formiga Paraconsistente

Para implementação do algoritmo da Formiga Paraconsistente, o algoritmo original de construção de soluções da metaheurísitca ACO, da Figura 2.4 é modificado na chamada da rotina que escolhe o próximo vizinho (linha 4 do algoritmo). Ao invés da rotina *escolhaProximo()* do algoritmo original é utilizado a rotina *escolhaProximoParaconsistente(*ε*)* apresentado no pseudo-código da Figura 4.3. A ideia é verificar, durante o processo de construção de soluções das formigas da colônia, se a trilha de feromônios deixada no espaço do problema é suficiente para tomada de decisão usando a lógica paraconsistente.

O processo de tomada de decisão das formigas da colônia, usando a lógica paraconsistente é determinado pelas Equações 4.1, 4.2 e 4.3 e está representado na Figura 4.1.

$$x = \text{argmax}_{l \in N_i^k} [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta} \tag{4.1}$$

Onde:

- *argmax* retorna o vizinho cujo $[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}$ é máximo
- N_i^k é a vizinhança ainda não visitada a partir do vértice i

$$y = \text{argmin}_{l \in N_i^k} [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta} \tag{4.2}$$

Onde:

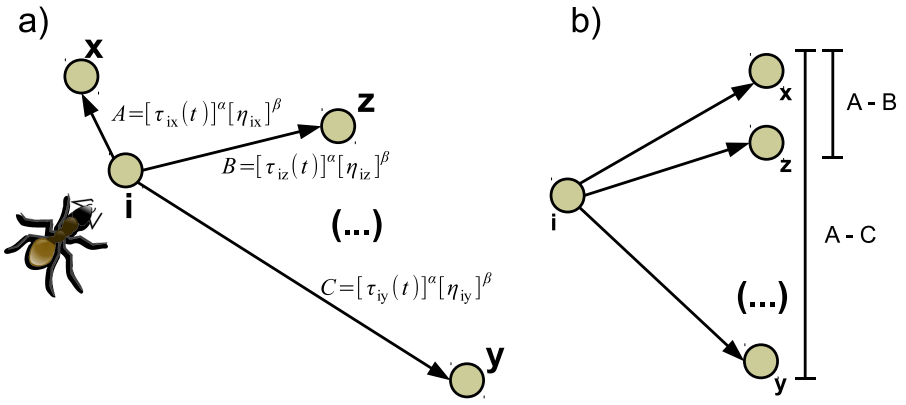


Figura 4.1: *Representação gráfica da tomada de decisão usando a lógica paraconsistente. a) o melhor, o segundo melhor e o pior caminho. b) x = melhor vizinho de i, z = segundo melhor vizinho de i e y = pior vizinho de i.*

- *argmin* retorna o vizinho cujo $[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}$ é mínimo

$$z = \text{argmax}_{l \in N_i^k - \{x\}} [\tau_{il}]^{\alpha}[\eta_{il}]^{\beta} \tag{4.3}$$

Para a tomada de decisão usando o algoritmo LPA3v, cada formiga k usa o conhecimento que está sendo construído pela colônia, representado pelo produto entre feromônio, τ e informação heurística, η no espaço do problema. O grau de evidência favorável μ para o caminho com maior produto $\tau \times \eta$ é adotado como 100%, $\mu = 1$. O grau de evidência desfavorável deste melhor vizinho é calculado como,

$$\lambda = \frac{[\tau_{ix}]^{\alpha}[\eta_{ix}]^{\beta} - [\tau_{iz}]^{\alpha}[\eta_{iz}]^{\beta}}{[\tau_{ix}]^{\alpha}[\eta_{ix}]^{\beta} - [\tau_{iy}]^{\alpha}[\eta_{iy}]^{\beta}} \tag{4.4}$$

Onde x, y e z são dados pelas Equações 4.1, 4.2 e 4.3. Conforme ilustrado na Figura 4.1, a formiga compara o "conhecimento" construído pela colônia para o seu melhor vizinho (x), o segundo melhor vizinho (z) e o pior vizinho (y). As diferenças nos teores de feromônio (τ) vezes heurística (η), determinam o grau de descrença para o melhor vizinho, considerando a lógica paraconsistente. Diferenças muito grandes tendem a produzir resultados inconsistentes. Dada uma especialidade, calcula-se o estado paraconsistente correspondente. Se o estado resultante é *totalmente verdade*, o melhor vizinho é utilizado de forma determinística na solução que está sendo construída. Caso contrário, o algoritmo de construção de soluções utiliza a rotina probabilística *escolhaPróximo()*, conforme o algoritmo da metaheurística ACO original.

Para considerar o aprendizado que está sendo realizado pela colônia de formigas, o algoritmo da Figura 4.3 tem um parâmetro que determina a variação do grau de especialidade. A especialidade ε é função das iterações e do parâmetro δ, conforme definido na Equação 4.5. Quanto menor a iteração mais inexperiente é a formiga para tomar decisões e quanto maior a iteração, mais especialista é a formiga.

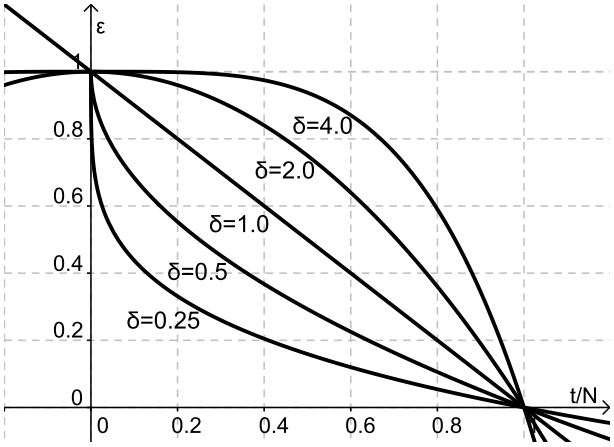


Figura 4.2: *Gráfico da variação da especialidade em função do parâmetro* δ

$$\varepsilon = 1 - \left(\frac{t}{N}\right)^{\delta} \tag{4.5}$$

Onde:

- t, representa a iteração corrente.
- N, representa o número máximo de iterações, que determina a *condiçãoDeTermino()* do algoritmo ACO.
- δ, determina como é a variação de especialidade durante as iterações.

Valores diferentes de δ produzem variações de especialidade conforme ilustrado na Figura 4.2. Para $\delta = 1$, a variação da especialidade ε é linear em relação as iterações, $t = 0$ temos $\varepsilon = 1$. Para $t = N$ temos $\varepsilon = 0$. Ou seja, no final das iterações as formigas tomam as decisões de forma determinística, em função somente das trilhas de feromônio criadas pela colônia de formigas. Para $\delta > 1$, a variação de especialidade é maior no final das iterações. Para $0 < \delta < 1$ a variação de especialidade é menor no final das iterações.

As formigas iniciam o processo de construção de soluções sem conhecimento. Em função das iterações, as formigas vão se tornando cada vez mais especialistas, podendo decidir de forma determinística qual caminho seguir, considerando somente o teor de feromônio nas trilhas. Conforme discutido em (Silva *et al.* , 2009), esta proposta serve para regular a convergência das formigas da colônia.

A forma como está definida a integração da metaheurística ACO com a Lógica Paraconsistente Anotada de Três Variáveis, permite a criação de sistemas híbridos com qualquer variação da metaheurísitca. Na próxima seção apresenta-se a aplicação do algoritmo proposto para a variação da metaheurística denominada $\mathcal{MAX} - \mathcal{MIN}$ *Ant System* ($\mathcal{MM}$AS).

```
1  escolhaProximoParaconsistente (ε) {
2    x ← argmax_{l∈N_i^k} [τ_il]^α [η_il]^β
3    y ← argmin_{l∈N_i^k} [τ_il]^α [η_il]^β
4    z ← argmax_{l∈N_i^k−{x}} [τ_il]^α [η_il]^β
5    μ ← 1.0
6    λ ← ([τ_ix]^α [η_ix]^β − [τ_iz]^α [η_iz]^β) / ([τ_ix]^α [η_ix]^β − [τ_iy]^α [η_iy]^β)
7    if diagnostico (μ, λ, ε) = V
8      return x
9    else
10     escolhaProximo() // Conforme Equação 2.1
11 }
```

Figura 4.3: *Pseudo-código da rotina de escolha do próximo vizinho usando a lógica paraconsistente para o algoritmo de construção de soluções da ACO*

4.2 Como Funciona a Colônia de Formigas Paraconsistente

A fim de ilustrar o funcionamento da Formiga Paraconsistente, apresenta-se nesta seção um acompanhamento da execução do algoritmo para uma instância pequena do problema do caixeiro viajante. Os processos de tomada de decisão e os resultados obtidos pelas formigas da colônia são apresentados e discutidos.

A instância de problema considerada neste exemplo ilustrativo é composta de seis cidades c_1, c_2, c_3, c_4, c_5 e c_6 cujas coordenadas são: $c_1 = (2,2)$, $c_2 = (0,5)$, $c_3 = (4,3)$, $c_4 = (5,0)$, $c_5 = (5,7)$, $c_6 = (7,3)$. O problema está ilustrado na Figura 4.4.

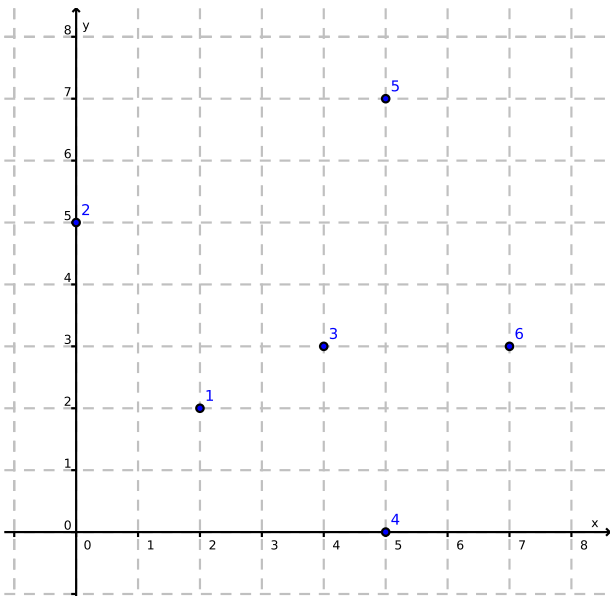


Figura 4.4: *Exemplo ilustrativo para 6 cidades*

No início do algoritmo é construída uma estrutura para armazenar o produto $\tau \times \eta$ entre as

cidades. O valor inicial é determinado através de uma parâmetro do algoritmo. Neste exemplo, consideramos que o valor inicial é 0.5, conforme ilustrado na Tabela 4.1.

Tabela 4.1: *Tabela com os valores de* $\tau \times \eta$ *entre todas as cidades no passo inicial do algoritmo*

$\tau \times \eta$	c_1	c_2	c_3	c_4	c_5	c_6
c_1	0	0.5	0.5	0.5	0.5	0.5
c_2	0.5	0	0.5	0.5	0.5	0.5
c_3	0.5	0.5	0	0.5	0.5	0.5
c_4	0.5	0.5	0.5	0	0.5	0.5
c_5	0.5	0.5	0.5	0.5	0	0.5
c_6	0.5	0.5	0.5	0.5	0.5	0

Cada formiga é posicionada aleatoriamente numa cidade inicial, para este exemplo considera-se 6 formigas que foram posicionadas inicialmente nas seguintes cidades:

- Formiga 1 é posicionada na cidade c_5;
- Formiga 2 é posicionada na cidade c_1;
- Formiga 3 é posicionada na cidade c_3;
- Formiga 4 é posicionada na cidade c_2;
- Formiga 5 é posicionada na cidade c_2;
- Formiga 6 é posicionada na cidade c_1.

A partir da sua cidade inicial, cada formiga, em paralelo, constrói uma solução completa para o problema, ou seja, um roteiro partindo desta cidade inicial e percorrendo todas as cidades sem passar duas vezes e sem evitar qualquer cidade. Como nenhum conhecimento foi construído ainda e o fator $\tau \times \eta$ é o mesmo entre quaisquer duas cidades, nesta iteração inicial as formigas constroem soluções completamente aleatórias. Os roteiros construídos pelas formigas nesta primeira iteração são:

- Formiga 1 = [5, 6, 4, 1, 2, 3], cujo comprimento do percurso é 23.884;
- Formiga 2 = [1, 6, 3, 4, 5, 2], cujo comprimento do percurso é 27.252;
- Formiga 3 = [3, 5, 2, 1, 6, 4], cujo comprimento do percurso é 24.981;
- Formiga 4 = [2, 4, 5, 6, 1, 3], cujo comprimento do percurso é 30.350;
- Formiga 5 = [2, 3, 1, 4, 5, 6], cujo comprimento do percurso é 29.066;
- Formiga 6 = [1, 6, 5, 3, 4, 2], cujo comprimento do percurso é 27.533.

Estas soluções estão representadas na Figura 4.5. Na Figura, não está representada a ligação que existe entre a última cidade e a cidade inicial (identificada com a marca ⊠) em cada roteiro, com o intuito de simplificar a identificação da sequência de cidades que compõem o roteiro. A formiga que encontrou a melhor solução na primeira iteração é a formiga 1, cujo comprimento do percurso é 23,884. Considera-se neste exemplo a distância euclidiana para medir o comprimento dos percursos.

Na próxima iteração o processo de escolha das formigas é repetido. Antes porém é realizada a atualização das trilhas de feromônio, conforme indicado no algoritmo da metaheurísitica de Colônia de Formigas apresentada na Figura 2.3 (linha 5).

Inicialmente todo o feromônio é evaporado considerando uma taxa de evaporação, que para este exemplo é igual a $\rho = 0.02$. A matriz de feromônios fica então, conforme representado na Tabela 4.2.

Tabela 4.2: *Matriz de feromônios após a evaporação por* $\rho = 0.02$*, ao final da primeira iteração*

$\tau \times \eta$	c_1	c_2	c_3	c_4	c_5	c_6
c_1	0	0.49	0.49	0.49	0.49	0.49
c_2	0.49	0	0.49	0.49	0.49	0.49
c_3	0.49	0.49	0	0.49	0.49	0.49
c_4	0.49	0.49	0.49	0	0.49	0.49
c_5	0.49	0.49	0.49	0.49	0	0.49
c_6	0.49	0.49	0.49	0.49	0.49	0

A Matriz $\tau \times \eta$ é recalculada, conforme a Equação 4.6. Neste exemplo usa-se $\alpha = 1$ e $\beta = 2$. A heurística η é calculada com o inverso da distância entre as cidades, conforme Equação 2.2.

$$[\tau_{ij}]^{\alpha} \times [\eta_{ij}]^{\beta} \quad \forall i, j \in \{1,2,3,4,5,6\} \tag{4.6}$$

Por fim, o caminho escolhido pela melhor formiga é premiado com uma taxa extra de feromônio igual ao inverso do comprimento do roteiro. O resultado está representado na Tabela 4.3.

Tabela 4.3: *Tabela com os valores de* $\tau \times \eta$ *entre todas as cidades no final da primeira iteração*

$\tau \times \eta$	c_1	c_2	c_3	c_4	c_5	c_6
c_1	0	0.038	0.098	0.038	0.014	0.019
c_2	0.038	0	0.024	0.01	0.017	0.009
c_3	0.098	0.024	0	0.049	0.029	0.054
c_4	0.038	0.01	0.049	0	0.01	0.038
c_5	0.014	0.017	0.029	0.01	0	0.024
c_6	0.019	0.009	0.054	0.038	0.024	0

As formigas são novamente posicionadas em cidades iniciais aleatórias para construir novas soluções. A cidade inicial de cada formiga na segunda iteração é:

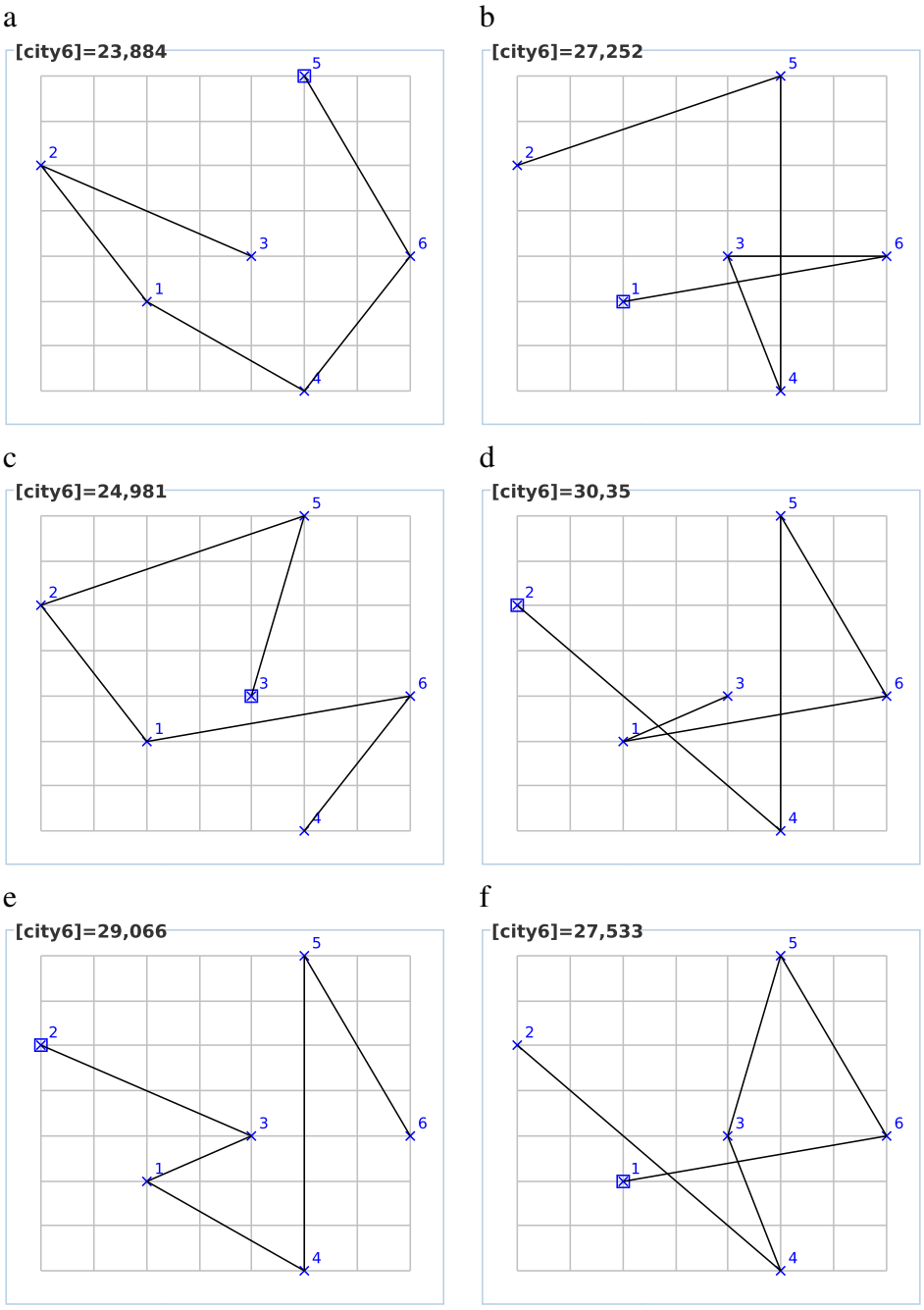


Figura 4.5: *Roteiros encontrados pelas formigas na primeira iteração do algoritmo de Formiga Paraconsistente. a) Formiga 1, b) Formiga 2, c) Formiga 3, d) Formiga 4, e) Formiga 5 e f) Formiga 6*

- Formiga 1 é posicionada na cidade c_4;
- Formiga 2 é posicionada na cidade c_3;
- Formiga 3 é posicionada na cidade c_4;
- Formiga 4 é posicionada na cidade c_5;
- Formiga 5 é posicionada na cidade c_4;
- Formiga 6 é posicionada na cidade c_3.

Em paralelo, cada formiga constrói uma solução completa para o problema. A formiga 1 tenta decidir usando lógica paraconsistente a próxima cidade a usar na solução. A partir da sua cidade inicial c_4 a formiga 1 encontra a seguinte situação:

- O melhor vizinho de c_4 é o vizinho c_3, com $\tau \times \eta = 0,049$
- O segundo melhor vizinho de c_4 é o vizinho c_1 (ou c_6), com $\tau \times \eta = 0,038$
- O pior vizinho de c_4 é o vizinho c_2 (ou c_5), com $\tau \times \eta = 0,010$

Calculados os valores dos graus de crença e descrença obtém-se os seguintes resultados:

- O grau de crença do melhor vizinho é 1.0;
- O grau de descrença do melhor vizinho é 0.71;
- A especialidade nesta iteração é 0.6.

A decisão da formiga 1, considerando a lógica paraconsistente não é assertiva ($\neq V$), conforme ilustrado no cubo LPA3v da Figura 4.6. Desta forma a formiga 1 usa o processo de decisão probabilística original da metaheuristica de Colônia de Formigas. A próxima cidade escolhida pela formiga 1 é a cidade c_1.

A formiga 2 é a próxima a escolher uma cidade para o seu roteiro. A partir da sua cidade inicial c_3, a formiga 2 encontra a seguinte situação:

- O melhor vizinho de c_3 é o vizinho c_1, com $\tau \times \eta = 0.098$;
- O segundo melhor vizinho de c_3 é o vizinho c_6, com $\tau \times \eta = 0.054$;
- O pior vizinho de c_3 é o c_5, com $\tau \times \eta = 0.024$.

Calculados os valores dos graus de crença e descrença obtém-se os seguintes resultados:

- O grau de crença do melhor vizinho é 1.0;

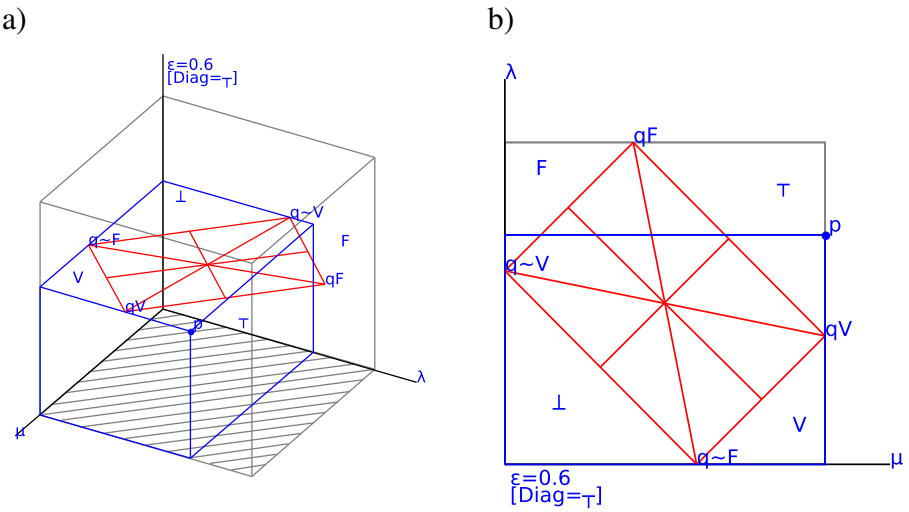


Figura 4.6: *Cubo analisador da lógica paraconsistente anotada de três variáveis. Determinação do valor lógico para* $\mu = 1.0$ *e* $\lambda = 0.71$ *e* $\varepsilon = 0.60$*. a) representação no cubo analisador b) projeção no plano* $\mu \times \lambda$

- O grau de descrença do melhor vizinho é 0.37;
- A especialidade nesta iteração é 0.6.

O diagnóstico da formiga 2, considerando a lógica paraconsistente é igual a V, conforme ilustrado no cubo LPA3v da Figura 4.7. Desta forma a formiga 2 escolhe a cidade c_1, seu melhor vizinho, para compor o seu caminho.

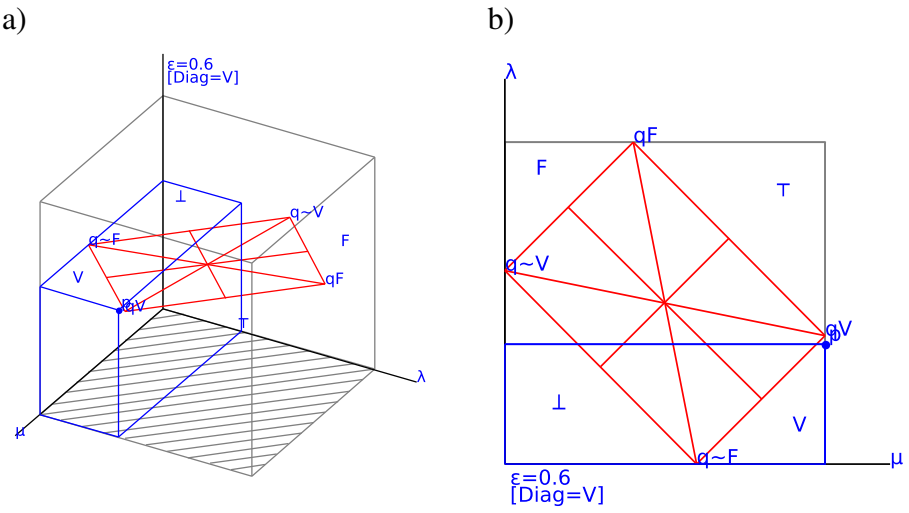


Figura 4.7: *Cubo analisador da lógica paraconsistente anotada de três variáveis. Determinação do valor lógico para* $\mu = 1.0$ *e* $\lambda = 0.37$ *e* $\varepsilon = 0.60$*. a) representação no cubo analisador b) projeção no plano* $\mu \times \lambda$

Usando este algoritmo, as formigas vão construindo as suas soluções e o melhor resultado encontrado por todas as formigas da colônia é armazenado entre as iterações só sendo atualizado

se um resultado melhor for encontrado. Os resultados encontrados pelas formigas 1 e 2 ao final da segunda iteração estão representados na Figura 4.8.

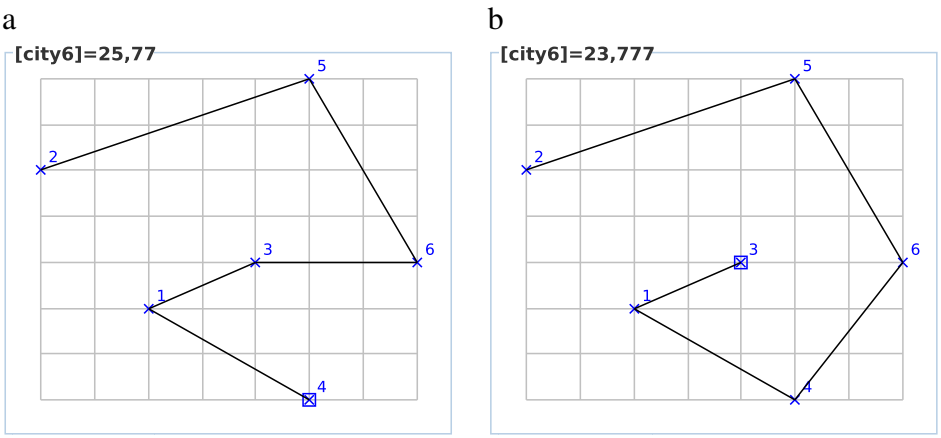


Figura 4.8: *a) Roteiro encontrado pela formiga 1 na segunda iteração, b) Roteiro encontrado pela formiga 2 na segunda iteração*

O melhor roteiro encontrado em todas as iterações está representado na Figura 4.9.

Na próxima seção é apresentada a comparação da Formiga Paraconsistente, deste trabalho com a melhor estratégia da metaheurística de Colônia de Formigas, aplicada para uma instância maior (com 225 cidades) do problema do caixeiro viajante.

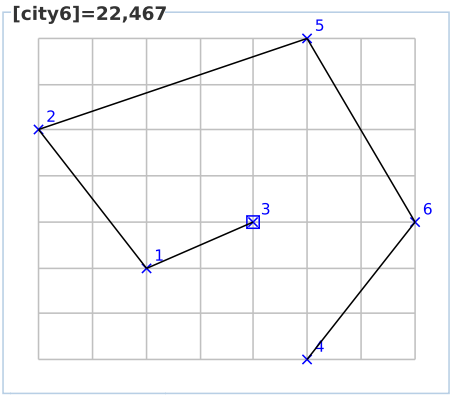


Figura 4.9: *Melhor roteiro encontrado ao final de todas as iterações*

4.3 Resultados Experimentais

Para demonstrar o desempenho da Formiga Paraconsistente, apresenta-se nesta seção uma implementação com a variação de ACO denominada $\mathcal{MAX} - \mathcal{MIN}$ *Ant System*. Para a realização dos testes, a implementação original do algoritmo $\mathcal{MM}$AS e o seu híbrido com a lógica paraconsistente foram aplicados para a solução do problema do caixeiro viajante (TSP). Conforme já foi comentado, este problema é um excelente *benchmark* para avaliação dos algoritmos

da metaheurística ACO. A descrição formal do problema do TSP para o modelo de feromônio da ACO pode ser encontrado em (Dorigo e Gambardella , 1997).

A fim de realizar os testes, a $\mathcal{MM}$AS padrão e a $\mathcal{MM}$AS híbrida com a lógica paraconsistente, foram utilizadas para resolução da mesma instância de TSP, disponível no repositório TSPLIB (Reinelt , 1991), denominada 'tsp225.tsp'. Para cada algoritmo foram realizados 5 experimentos, com 2000 repetições por experimento, sendo estas repetições divididas em 5 tentativas de 400 iterações, conservando, entre as tentativas, a melhor solução encontrada entre todas as formigas. Os demais parâmetros utilizados por ambas as técnicas foram os seguintes: o número de formigas utilizadas igual ao número de cidades da instância do problema, neste caso 225 formigas, $\alpha = 1$, $\beta = 2$, $\rho = 0.02$. Segundo (Stützle e Hoos , 2000) estes parâmetros são boas escolhas para o problema de otimização do caixeiro viajante. Além disso não foi utilizada qualquer operação de busca local para nenhum dos algoritmos.

Como métrica de distância entre as cidades do problema, adotou-se a distância euclidiana com o arredondamento. Isso foi feito porque os valores ótimos dos percursos de tais problemas estão disponíveis (Reinelt , 1991), sendo possível, então, uma interessante análise destes com os resultados encontrados por ambas as estratégias da ACO apresentadas.

Para o híbrido de $\mathcal{MM}$AS com a lógica paraconsistente foi utilizada a variação da convergência definida pelo valor de $\delta = 0.5$. conforme ilustrado na Figura 4.2.

Por fim, para cada um dos modelos apresentados, foram realizados 5 experimentos com as configurações estabelecidas, salvando em cada experimento a melhor solução encontrada pela colônia de formigas.

Os resultados apresentados nos experimentos realizados para as metodologias da estratégia da ACO, a $\mathcal{MM}$AS padrão e do $\mathcal{MM}$AS híbrido com a lógica paraconsistente estão sintetizados na Tabela 4.4.

Tabela 4.4: *Resultados da estratégias $\mathcal{MM}$AS e da Formiga Paraconsistente para a instância da TSPLIB denominada 'TSP225'*

Estratégia	1	2	3	4	5	Média	Desvio
$\mathcal{MM}$AS	4029	3999	**3996**	4094	4010	4025.6	40.36
Formiga Paraconsistente	4008	4070	**3975**	4024	4031	4021.6	34.62

A Tabela 4.4 apresenta o resultado de cada teste realizado, a média e o desvio padrão para as estratégias de $\mathcal{MM}$AS e $\mathcal{MM}$AS paraconsistente. Para a instância considerada o resultado ótimo é 3919. As Figuras 4.10a, 4.10b e 4.10c, apresentam o resultado ótimo, o melhor resultado encontrado pela estratégia $\mathcal{MM}$AS e o melhor resultado encontrado para a estratégia $\mathcal{MM}$AS paraconsistente.

Pela Tabela 4.4, comparando a média e o desvio padrão de ambas as abordagens, pode-se observar que a estratégia $\mathcal{MM}$AS da ACO com a lógica paraconsistente obteve resultados melhores que a estratégia $\mathcal{MM}$AS original para esse instância em questão. Dessa forma, a Formiga Paraconsistente demonstra ser uma variação original e interessante da metaheurística de colônia

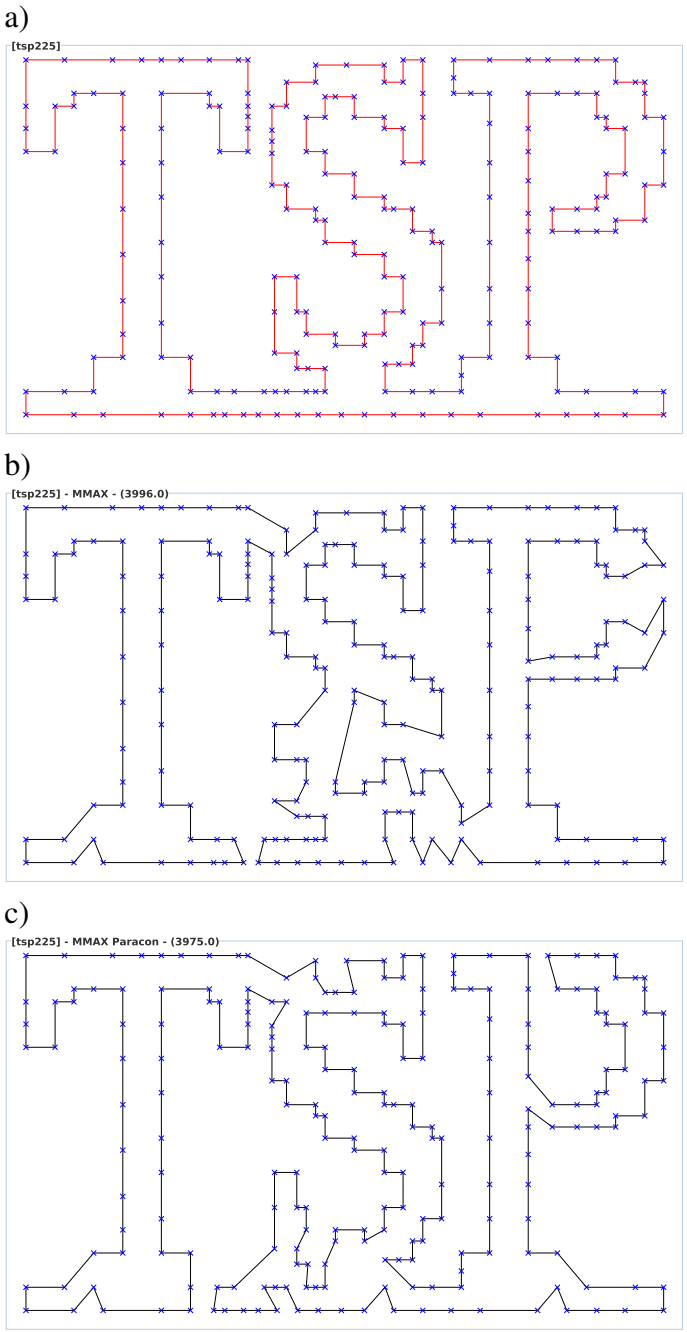


Figura 4.10: *Aplicação das estratégias $\mathcal{MMAS}$ e $\mathcal{MMAS}$ paraconsistente na instância tsp225 da TSPLIB. a) solução ótima b) melhor solução encontrada usando a estratégia $\mathcal{MMAS}$ e c) melhor solução encontrada usando a estratégia $\mathcal{MMAS}$ paraconsistente*

de formigas. Para melhorar esses resultados, existem alguns parâmetros para ajustar como: (a) a forma da variação de especialidade definido pelo parâmetro δ; (b) a forma de determinar o grau de evidência favorável μ e o (c) grau de evidência desfavorável λ utilizado pelas formigas da colônia durante o processo de tomada de decisão. Estes parâmetros adicionais permitem controlar de forma mais precisa o processo de convergência das formigas da colônia, conforme será discutido na Seção 4.3.1.

4.3.1 A Convergência na Formiga Paraconsistente

A fim de demonstrar a convergência crescente da colônia, pode-se medir a distância entre cada solução de cada formiga. Uma boa medida de distância entre dois roteiros s e s' é dado pela Equação 4.7, conforme sugerido em (Stützle e Hoos , 2000), que conta o número de arcos diferentes em cada roteiro.

$$d(s,s') = n - |\{(i,j) : (i,j) \in s \wedge (i,j) \notin s'\}| \tag{4.7}$$

Onde n é o número de locais no roteiro. A distância média das soluções de todas as formigas da colônia é calculada a fim de medir a convergência da colônia. A Formiga Paraconsistente torna possível controlar a convergência da colônia, conforme foi proposto neste trabalho.

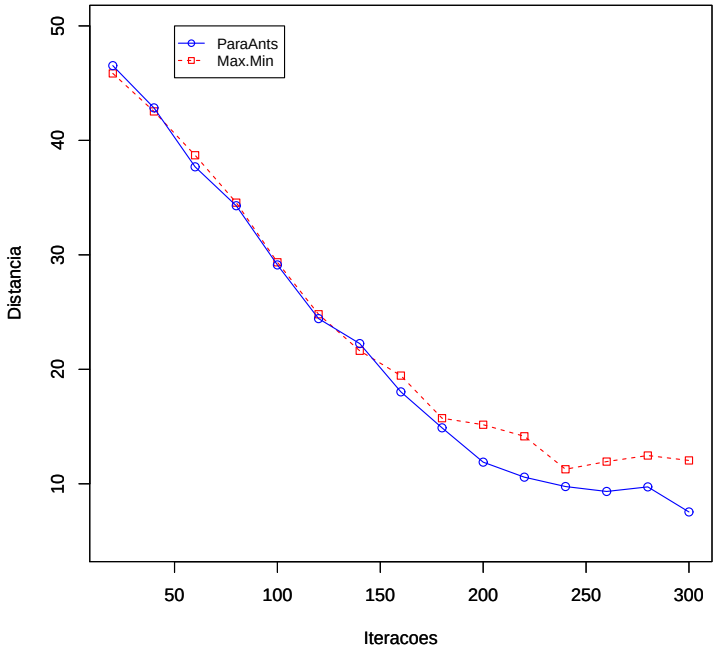


Figura 4.11: *Comparação da convergência dos algoritmos $\mathcal{MAX}-\mathcal{MIN}$ e das Formigas Paraconsistentes para a instância denominada eil76, da TSPLIB.*

O gráfico da Figura 4.11 corresponde ao experimento realizado para a instância *eil76* da

TSPLIB, (Reinelt , 1991). Para este experimento foram utilizadas 76 formigas, em 5 tentativas e 300 iterações. A Figura 4.11 demonstra a convergência das formigas durante as iterações. O eixo vertical do gráfico denota a distância média das soluções das formigas da colônia, conforme a Equação 4.7. A distância diminui mais acentuadamente usando a Formiga Paraconsistente em relação ao algoritmo $\mathcal{MAX} - \mathcal{MIN}$, conforme já era esperado.

4.3.2 Diversificação versus Intensificação na Formiga Paraconsistente

Os algoritmos para problemas de otimização lidam com objetivos contraditórios na sua implementação. É interessante que no início do processo o algoritmo experimente as mais diversas possibilidades no espaço de busca do problema. Uma analogia pode ser feita com relação a diversidade de caminhos definidos pela colônia de formigas assim que se inicia o processo de busca por alimento. Após um período inicial da busca e próximo do final do algoritmo é interessante que as formigas intensifiquem a busca em torno da melhor solução encontrada. A questão é, portanto, regular o processo aleatório no início do algoritmo e o processo determinístico em torno da melhor solução no final do algoritmo.

Dessa forma, a Formiga Paraconsistente apresenta um parâmetro a mais que pode ser utilizado a fim de determinar o instante, a partir do qual, as formigas começam a intensificar a busca em torno da melhor solução. A variação da especialidade (ε) tem essa função, regular o momento do processo de intensificação do algoritmo de Formigas Paraconsistentes.

4.4 Considerações Finais

Este capítulo apresenta o algoritmo da Formiga Paraconsistente, uma variação da metaheurística de colônia de formigas com a lógica não clássica denominada Lógica Paraconsistente. A lógica paraconsistente é uma extensão da lógica clássica que permite trabalhar com outros estados lógicos, além do estados verdade e falso.

Com a lógica paraconsistente pode-se tratar de forma mais realista problemas de tomada de decisão que envolvem incerteza, conhecimento parcial e inconsistência. As formigas da estratégia de otimização denominada Colônia de Formigas se depara com este tipo de problema de inconsistência e indecisão quando constroi soluções para o problema que está sendo atacado pela colônia. A Lógica Paraconsistente é embutida no algoritmo original da metaheurística de Colônia de Formigas, a fim de tentar capturar o processo de aprendizado realizado pela colônia de formigas durante a construção de soluções.

O algoritmo proposto, denominado Formiga Paraconsistente é aplicado a um *benchmark* de otimização combinatória, denominado Problema do Caixeiro Viajante e o resultado encontrado é melhor que o resultado obtido pela variante da ACO denominado $\mathcal{MM}$AS, para a instância do problema considerado.

Nos próximos capítulos são apresentados implementações da Formiga Paraconsistente para

problemas de otimização em espaço de busca discreto e em espaço de busca contínuo. Os algoritmos desenvolvidos são aplicados em problemas de otimização na área de Sistemas Elétricos de Potência e os resultados são apresentados e discutidos.

Capítulo 5

Aplicação da Formiga Paraconsistente em Espaço de Busca Discreto

Nesse capítulo, apresenta-se uma implementação da Formiga Paraconsistente para solução de um problema de otimização em espaço de busca discreto. Para isso, utiliza-se a Formiga Paraconsistente no problema de Restabelecimento de Sistema de Distribuição, que conforme será apresentado é um problema característico do domínio discreto.

5.1 Otimização em Espaço de Busca Discreto

Os Problemas de otimização (Cormem *et al.* , 2002) são problemas que se caracterizam por possuírem várias soluções possíveis, onde cada solução tem um valor associado. O objetivo da otimização é encontrar uma solução com o valor mínimo (ou o valor máximo), que é chamada de solução ótima. Quando o objetivo é encontrar uma solução com o menor valor associado, chamamos de problema de minimização. Quando é para encontrar a solução com o maior valor associado, tem-se um problema de maximização. Em alguns problemas podem existir inúmeras soluções ótimas.

Usando uma abordagem matemática, o problema de otimização pode ser definido da seguinte forma (Boyd e Vandenberghe , 2004):

Definição 1 *Dada a função* $f : S \to \mathbb{R}$, $X^* \in S$ *é mínimo de* f *se,* $f(X^*) \leq f(X) : \forall X \in S$. *Da mesma forma,* $X^* \in S$ *é máximo de* f *se,* $f(X^*) \geq f(X) : \forall X \in S$.

A função f é a função objetivo do problema de otimização, o domínio S é denominado *espaço de busca*, e os elementos de S são denominadas *soluções factíveis*. As soluções em S podem estar condicionadas a alguma restrição do problema. Uma solução factível X é um conjunto finito de variáveis de otimização $X = X_1, X_2, ..., X_n$. Uma solução factível X^* que minimiza (maximiza) a função objetivo f é denominada *solução ótima*.

É fácil observar que um problema de maximização sobre uma função objetivo f é equivalente a um problema de minimização sobre uma função objetivo $-f$.

Nos problemas de otimização em espaço de busca discreto, as variáveis de otimização $X_i, i = 1, 2, ...n$ são discretas, ou seja, pertencem a um conjunto contável $X_i \in D_i, i = 1, ..., n$.

5.1.1 Otimização Combinatória

Problemas de otimização combinatória são um subconjunto da classe de otimização em espaço de busca discreto. Esses podem ser expressos como o problema de encontrar uma permutação ou combinação de um conjunto finito de elementos.

Formalmente, um problema de otimização combinatória P pode ser definida por

Definição 2 $P = (S, \Omega, f)$

- S *é o espaço de busca definido sobre um conjunto de variáveis discretas;*
- Ω *é o conjunto de restrições entre as variáveis;*
- $f : S \rightarrow \mathbb{R}$ *é a função objetivo que deve ser minimizada (maximizada).*

Um exemplo de problema de otimização combinatória é o problema do Caixeiro Viajante, conforme já foi explicado na Seção 2.1. No Caixeiro Viajante a solução ótima é uma permutação de todas as cidades que devem ser visitadas, que defina o menor percurso, i. e., a menor soma das distâncias entre as cidades. A restrição para as soluções factíveis, nesse caso é que todas as cidades devem compor a solução e nenhuma cidade pode ser relacionada mais de uma vez na solução.

Na próxima seção é apresentado um problema de otimização em espaço de busca discreto no campo de Sistemas Elétricos de Potência, denominado problema de Restabelecimento de Sistemas de Distribuição. A Formiga Paraconsistente é adaptada para encontrar uma combinação de chaves que restabeleça o sistema de distribuição.

5.2 O problema de Restabelecimento de Sistemas de Distribuição

Nos dias atuais, com o aumento da demanda por energia elétrica, torna-se fundamental também a demanda por Sistemas Elétricos de Distribuição (SEDs) que sejam confiáveis e disponíveis (Liu , 2006; Rossi , 2000). Porém, é impossível garantir que um SED estará sempre disponível, devido à possibilidade de ocorrência de falhas em partes do sistema em algum momento. Uma falha em um SED corresponde à quebra da configuração atual desse sistema, podendo fazer com que algumas cargas (ou seja, unidades que demandam energia) tenham a sua alimentação interrompida. Logo, falhas causadas por danos físicos requerem que alguns componentes desses sistemas sejam reparados ou substituídos, para que estes retornem a um estado de distribuição de energia similar ao estado anterior à falha. Contudo, enquanto os componentes afetados por falhas não são substituídos, a arquitetura de um SED, para que esta tenha uma maior confiabilidade, possui ligações alternativas entre suas cargas, permitindo diversas configurações operacionais para a alimentação destas. Espera-se, então, que o caminho das linhas de

transmissão de um SED sejam reconfiguradas de modo que se retome o atendimento às cargas o mais rápido possível, ao menor custo. À tarefa de reconfiguração deste sistema para atingir esse objetivo é dada o nome de Serviço de Restabelecimento.

Com isso, algumas características devem ser respeitadas em um SED durante a atividade de correção de uma falha pelo serviço de restabelecimento. Tais características estão embutidas no problema de Restabelecimento de SEDs, e serão discutidas.

5.2.1 Restabelecimento de Sistemas Elétricos de Distribuição

O serviço de restabelecimento envolve a escolha de uma sequência de chaves (elementos capazes de fazer a interligação entre linhas de transmissão) a fim de estabelecer um caminho alternativo entre a fonte geradora e as cargas que tiveram o serviço de atendimento interrompido, após a ocorrência de uma falha. Para isso, os SEDs contam com linhas alternativas entre as cargas que possuem chaves normalmente abertas (desconectam uma linha de energia do sistema), sendo as primeiras a terem o seu estado alterado no caso de uma falha. Em situações normais, as cargas possuem chaves que estão normalmente fechadas (conectam uma linha de energia do sistema), sendo estas as primeiras que têm o seu estado alterado em caso de uma sobrecarga de uma linha do sistema.

Para o serviço de restabelecimento dos SEDs, devem ser observadas as seguintes características (Watanabe , 2005):

- Restrição radial da rede: devido à topologia dos SEDs ser similar a de uma árvore geradora de um grafo, o número de linhas que transferem energia, de modo que todas as cargas sejam atendidas, deve ser igual ao número de cargas menos um;
- Restrição de balanço de energia: as cargas totais do sistema não devem ultrapassar o limite máximo da fonte de energia correspondente;
- Restrição da capacidade da linha: o fluxo de energia por uma linha não deve ser maior que sua capacidade.

Tais características são obrigatórias nos SEDs que apresentam uma topologia radial. Uma característica desejável é que todas as cargas que tiveram o atendimento interrompido por uma falha sejam supridas após o serviço de restabelecimento. Porém, isso pode depender da configuração da rede. Todo o processo descrito acima é mostrado na Figura 5.1. A Figura 5.1a ilustra um SED em seu funcionamento normal, com algumas chaves normalmente abertas (NA_i), e as outras fechadas (NF_i), para que todas as cargas sejam supridas. A Figura 5.1b ilustra uma situação de falha. A Figura 5.1c ilustra uma possível solução para o restabelecimento do sistema, onde fechando apenas a chave NA_3, todas as cargas têm o seu atendimento garantido. Por fim, a Figura 5.1d ilustra uma possível solução, caso a linha com a chave NC_2 da Figura 5.1c tenha uma sobrecarga.

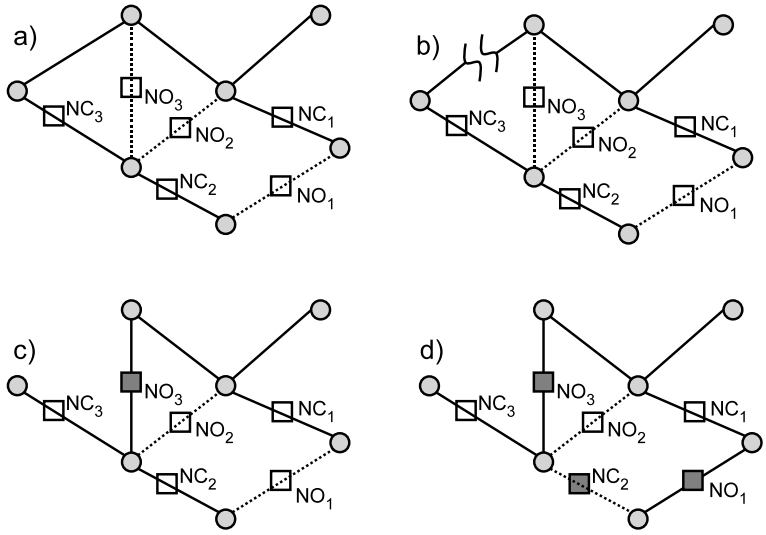


Figura 5.1: *Representação de um Sistema Elétrico de Distribuição (a), assim como situações de restabelecimento (b,c,d) em caso de falhas no sistema.*

Um fato que pode ser observado na Figura 5.1 é o número de chaves alteradas no caso de uma falha. Devido à topologia da rede, para que o seu aspecto radial seja mantido, o número de chaves que tem o seu estado alterado é sempre ímpar. Outro ponto fundamental para o restabelecimento é que para realizar esse processo, a sequência de chaves deve ser intercalada entre chave normalmente aberta e chave normalmente fechada. Assim, à medida que um SED vai aumentando a sua complexidade, o número de chaves desse sistema cresce consideravelmente. Logo, com um número maior de chaves, o número de sequências possíveis na tentativa de um restabelecimento torna-se muito grande, devido ao número de combinações de chaves possíveis para formar uma sequência que satisfaça às restrições do problema. Diante desta natureza multi-objetiva, todo esse processo poderá ser dispendioso, o que torna o restabelecimento de SEDs uma tarefa de computação difícil. Este é o ponto que torna interessante a aplicação de heurísticas de otimização para problemas combinatórios difíceis.

5.2.2 Restrições e objetivos do problema de restabelecimento de SED

Os trabalhos encontrados sobre o tema, descrevem diferentes cenários e estratégias para a solução de problemas relacionados ao restabelecimento de Sistema Elétrico de Distribuição. A complexidade dos Sistemas de Distribuição, além do tamanho da rede de distribuição, está relacionada a diversos objetivos e restrições do sistema como:

1. A carga tem que ser restaurada para o máximo de áreas afetadas pela falha.

2. Do ponto de vista econômico, deve haver o mínimo de perda de carga para a restauração do sistema.

3. A restauração do sistema deve manter o aspecto radial do sistema elétrico de distribuição.

4. Para restabelecimento do sistema pode haver variação de carga e voltagem nas linhas do sistema. Essas mudanças tem que respeitar os limites superior e inferior para os barramentos do sistema.

5. A restauração é basicamente realizada pela transferência de carga de um sistema energizado para o sistema que está fora de área via mudança do estado de ligado para desligado e vice-versa para diferentes chaves no sistema de distribuição. Algumas dessas chaves são operadas manualmente e outras automaticamente. O tempo de resposta de chaves automáticas e chaves manuais são diferentes e devem ser considerados na solução de problemas reais.

6. A sequência de operações das chaves para o restabelecimento do sistema é importante para garantir a confiabilidade dos Sistemas Elétricos.

7. Em qualquer sistema de distribuição, há sempre algumas cargas que tem prioridade mais alta para o restabelecimento do sistemas (hospitais, indústrias, etc.). No caso de um restabelecimento parcial do sistema, esses clientes com prioridade maior tem que participar da solução encontrada para o restabelecimento.

8. O tempo de execução do algoritmo para restabelecimento do sistema precisa encontrar soluções o mais rápido possível para ser útil em situações reais.

9. O Sistema Elétrico pode conter geração distribuída, com o objetivo de garantir a manutenção dos sistemas em locais críticos como indústrias, hospitais, etc. O algoritmo de restabelecimento de sistemas deve levar isso em consideração na solução do problema.

10. O problema pode ser visto do ponto de vista do planejamento da colocação de chaves seccionadoras para aumentar a confiabilidade dos sistemas elétricos.

11. O restabelecimento deve balancear a carga nos transformadores e nos alimentadores do sistema elétrico.

A próxima seção apresenta uma revisão bibliográfica das estratégias utilizadas para resolução do problema de restabelecimento de sistemas elétricos de distribuição.

5.2.3 Revisão de Métodos Heurísticos e Meta-Heurísticos

Em (Watanabe *et al.* , 2006), o autor propõe um sistema híbrido para resolução do problema de restabelecimento de sistemas de distribuição. O autor já havia proposto uma solução baseada em algoritmos genéticos, aplicados em duas etapas: uma que escolhe as chaves para o restabelecimento do sistema e outra que escolhe a sequência de chaves que minimizam a potência não atendida durante o processo de restabelecimento do sistema. Essa primeira versão demonstrou-se aplicável para sistemas pequenos. O sistema híbrido proposto se adapta ao algoritmo de duas

etapas original e inclui três estratégias para melhorar o algoritmo e torná-lo aplicável a sistemas maiores. As estratégias são: busca local, algoritmo guloso e algoritmos de fluxo máximo eficientes.

Uma solução baseada no algoritmo genético denominado (NSGA-II, *nondominated sorting genetic algorithm*) é apresentada em (Kumar *et al.* , 2008). Vários objetivos e restrições que podem estar envolvidos no restabelecimento de Sistemas de Distribuição são discutidos nesse trabalho. Para poder combinar os vários objetivos e restrições, a estratégia NSGA-II usa o conceito de dominância, que define quando uma função objetivo domina outra função. Quando comparando duas soluções u e v, diz-se que a solução u domina a solução v, se u é tão boa quanto v em todas as funções objetivos e, além disso, u é melhor que v em pelo menos uma função objetivo. A seleção, no algoritmo genético, é feita calculando o número nd de funções objetivo das soluções que dominam as funções da solução selecionada. As soluções com $nd = 0$ são aquelas melhor classificadas na seleção.

Em (Xianchao e Taylor , 2010) o objetivo de redução minimizada de cargas controláveis é incorporado no problema de restabelecimento de sistemas de distribuição para utilização do algoritmo genético NSGA-II. Em (Xianchao *et al.* , 2010), o autor usa novamente NSGA-II como estratégia para resolver um problema de restabelecimento levando em consideração outros critérios ainda de otimização para geração distribuída: como a capacidade de usar o processo *Black-Start* (capacidade de restaurar o sistema sem necessitar de uma fonte externa) para restaurar o sistema e o uso privilegiado de fontes de energia renováveis para o processo de restabelecimento.

A indústria de distribuição de energia tem duas preocupações principais: a satisfação dos clientes e a confiabilidade do serviço, conforme citado em (Toune *et al.* , 2002). Vários estudos sugerem que a satisfação do cliente está relacionada a frequência em que o serviço de energia é interrompido e a duração das interrupções. Desenvolver estratégias efetivas para restabelecimento de sistemas é uma forma de melhorar a confiabilidade e dessa forma aumentar a satisfação dos clientes. O problema de restabelecimento é um problema NP-Completo e nesse artigo o autor investiga a aplicabilidade de quatro estratégias de algoritmos heurísticos que são atualmente utilizados na resolução desse problema. As estratégias investigadas são: Algoritmos Genéticos, *Simulated Annealing* Paralelo, Pesquisa Tabu e Pesquisa Tabu Reativa. O que diferencia essas duas últimas estratégias é que a Pesquisa Tabu simples usa uma lista tabu fixa e a Pesquisa Tabu Reativa usa uma lista tabu variável, onde o tamanho pode crescer ou diminuir em função de uma taxa de modificação do comprimento da lista.

Uma alternativa para resolução do problema de Restauração de Sistema de Distribuição usando Programação Dinâmica com redução de estados é apresentada em (Perez-Guerrero *et al.* , 2008). Os estados representam o tempo e a seleção de cargas que devem ser energizadas. Para aumentar a velocidade da computação do método de Programação Dinâmica o autor propõe um esquema de redução de estado usando um critério de agrupamento. A redução de estados é realizada pelo agrupamento de estados que estão próximos e selecionando o melhor estado do

grupo.

Em (Tian *et al.* , 2009) apresenta-se um método baseado na Otimização por Enxame de Partículas Binário (BPSO) para resolver o problema de Restabelecimento, baseado na informação de tipo, capacidade e prioridade das cargas, numa rede de distribuição com geradores distribuídos. O tamanho dos Sistemas de Potência tem causado dificuldades e alto custo para operação desses sistemas. Para melhorar aspectos de operação e segurança tem sido utilizadas tecnologias de geração distribuída nesses sistemas.

5.2.4 Revisão de Otimização de Colônia de Formigas

A soluções baseadas em Algoritmos Genéticos e Enxame de Partículas dependem de bons populações iniciais para aproximar a busca da solução ótima e dependendo da estratégia adotada, o processo encontra muita solução inviável durante o processo de busca de soluções. Em (Lu *et al.* , 2009), uma alternativa para o problema usando ACO adaptado é apresentada. Para encontrar soluções viáveis é utilizada uma estratégia de pesquisa baseada em Árvore Geradora Estocástica. E a forma de seleção do caminho e atualização de feromônio do MMAS é adaptado para melhorar a estratégia ACO ao problema de restabelecimento de sistemas de distribuição.

Em (Falaghi *et al.* , 2009), diferente dos trabalhos que tratam questões relacionadas ao restabelecimento de sistemas, apresenta-se uma metodologia, baseada em lógica Fuzzy e ACO, para colocação de chaves seccionadoras em redes de distribuição a fim de melhorar a confiabilidade do sistema com consideração de aspectos econômicos do projeto proposto por esse sistema.

Uma solução para problema de restabelecimento de sistemas usando o *framework Hypecube Ant Colony System* para otimizar a sequência de chaves que minimiza a energia não suprida no processo de restabelecimento é apresentada em (Watanabe , 2005). Além dos objetivos e restrições do problema original, é tratado nesse trabalho a sequência ótima de operações com chaves para restabelecimento do sistema após a ocorrência de uma falha.

Em (Ahuja *et al.* , 2008) apresenta-se um híbrido de uma sistema imunológico artificial e da otimização por colônia de formigas para resolver o problema de restabelecimento de sistemas. O sistema imunológico artificial é usado para explorar o espaço de pesquisa através de uma operador chamado hipermutação. Para melhorar a convergência, o autor propõe uma implementação da hipermutação baseada em feromônios. O ACO é usado como algoritmo de otimização. A característica de otimização multi-objetivo do problema de otimização é tratada usando o conceito de dominância, assim como o NSGA-II.

5.3 Formigas Paraconsistentes em Espaço de Busca Discreto

A fim de se mapear o problema de restabelecimento de SEDs para o modelo de feromônio da ACO, existem algumas características do algoritmo que são essenciais para o seu funcionamento. Entre elas estão as formigas, a informação heurística, feromônio e a forma como as

soluções são construídas pelas formigas da colônia.

5.3.1 Tipificação das Formigas

As formigas da colônia serão o núcleo da técnica proposta, pois são as responsáveis pela construção de uma solução para o SED em caso de ocorrência de uma falha. Assim, irão adotar o seguinte processo de construção de soluções: na elaboração de uma solução, cada formiga vai possuir uma sequência de chaves $S = \{C_i, C_{i+1}, ..., C_n\}$. Entre estas chaves, estão embutidas os dois tipos de chaves já comentadas, as normalmente abertas (NA) e as normalmente fechadas (NF). Para que a característica radial da rede seja sempre satisfeita, é obrigatória a seguinte ordem para o restabelecimento: fechamento de uma chave NA e verificação da condição de parada; caso haja sobrecarga, abre-se uma chave NF, fecha-se uma NA e verifica-se novamente a condição de parada; esse processo segue até que se encontre uma solução. Diante disso, o conjunto S pode ser ordenado da seguinte forma: para todo i ímpar do conjunto, o elemento referenciado corresponderá a uma chave NA que poderá ser fechada; e para todo i par do conjunto, o elemento referenciado corresponderá a uma chave NF, que poderá ser aberta para a construção de uma solução. Seguindo este raciocínio,observa-se que a cardinalidade do conjunto S será sempre ímpar, pois é obrigatória a intercalação de uma sequência par (abertura e fechamento de chaves) ao primeiro fechamento, caso uma única alteração não seja suficiente para encontrar uma resposta para o problema.

5.3.2 Informação Heurística

Como as formigas detêm um processo de construção estocástico e devido à baixa influência das trilhas iniciais de feromônio, guiá-las para uma possível solução factível é algo indispensável no início da otimização.

Um ponto importante que se deve ser observado para este problema é que não há nenhuma informação mensurável entre as ligações de cargas que sirva para uma escolha de um percurso pelas formigas quando as trilhas de feromônio têm pouca ou nenhuma influência, o que difere, por exemplo, do Problema do Caixeiro Viajante, que fornece essa informação. Isso acaba gerando uma dificuldade para se estabelecer uma função heurística que seja interessante para este problema. Assim, a informação heurística cabível para o problema do SED está relacionada às chaves que devem ser escolhidas para o seu restabelecimento: quais chaves NA devem ser fechadas e quais chaves NF devem ser abertas.

Para as chaves NA que devem ser fechadas, a melhor heurística é proceder com o fechamento daquelas mais próximas à região de falha. Isso, pelo motivo de satisfabilidade: quanto mais próximo à falha uma chave NA for fechada, maior será a chance de se ter um sistema com 100% das cargas restabelecidas. Se por um acaso não houver mais chaves NA próximas à região de falha, certamente o sistema não será restabelecido em sua totalidade.

Para as chaves NF, usa-se como função heurística a distância da chave NF para a fonte de geração, considerando-se para isso a Equação 5.1.

$$\eta_{NF}[i] = \max_{i}\{s_i\} - s_i + 1 \tag{5.1}$$

Onde:

- s_i representa o número de linhas da chave até a fonte de geração.
- i representa o índice da chave NF.

5.3.3 A Representação do Feromônio

O SED não possui caminhos entre as cargas nos quais seja possível fazer o depósito de feromônio, pois não se busca a otimização de caminhos, mas a otimização de trocas de chaves. Logo, os níveis de feromônio estão relacionados a todas as chaves que este SED possui, tanto as NA, quanto as NF. Formalmente, teremos $\tau_{NF}[i]$ relacionado à quantidade de feromônio presente na chave fechada NF de índice i; e $\tau_{NA}[j]$ relacionado à quantidade de feromônio depositado na chave aberta NA de índice j.

Assim, quanto mais feromônio possui uma chave do SED, seja ela qual for (NA ou NF), tem-se a indicação de que várias formigas estão utilizando-a no restabelecimento e, com isso, alterando seus níveis de feromônio. Desse modo, quanto maior a quantidade de feromônio de uma chave, maior a sua probabilidade de escolha na função de probabilidade do algoritmo que as formigas utilizam para construir suas soluções para o problema de restabelecimento.

5.3.4 Construção das Soluções

Na ocorrência de uma falha, considerando que somente uma linha foi cortada, uma chave NA adjacente a região da falha deve ser mudada a fim de restabelecer o sistema.

Verifica-se então se existe sobrecarga em alguma linha do sistema. Se não há sobrecarga, uma solução ótima foi encontrada. Se por outro lado, depois da mudança na chave NA, o sistema apresenta sobrecarga em alguma linha duas novas linhas tem que ter seus estados mudados. Uma NF para resolver a sobrecarga na linha e uma NA para restabelecer atender todas as cargas do sistema.

Para a escolha da NF que deve ser aberta a fim de eliminar alguma sobrecarga, utiliza-se a lógica paraconsistente de modo similar aquele utilizado para solução do problema do Caixeiro Viajante. Nessa abordagem, no entanto, a escolha é feita considerando a quantidade de feromônio nas chaves NF. Se a diferença entre o maior nível de feromônio e o segundo maior nível de feromônio nas chaves NF é suficiente para tomar uma decisão assertiva, considerando a lógica paraconsistente, a chave com maior nível de feromônio é escolhida. Caso contrário, o método probabilístico original da metaheurística ACO é utilizado.

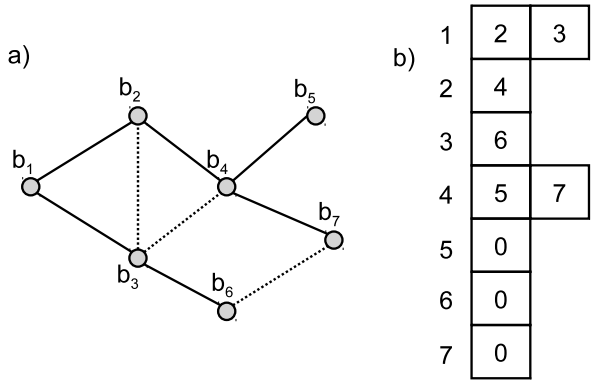


Figura 5.2: *Representação da árvore geradora para o SED, sendo (a) o próprio SED e (b) a sua árvore geradora*

Estes passos são executados por cada formiga da Colônia, durante o processo de construção de soluções.

5.3.5 Estrutura de Dados Adicionais

A intenção de utilizar um algoritmo de otimização em um problema real tem a finalidade de obter uma resposta rápida e interessante para o problema em questão. Além da utilização exclusiva de um algoritmo para auxiliar na resolução de um problema, pode-se recorrer a algumas estruturas de dados que modelem o problema a ser tratado. Isto torna mais natural e específica a representação deste para a estrutura que o algoritmo necessita para sua execução, melhorando ainda mais o seu desempenho.

Assim sendo, este trabalho buscou utilizar de estruturas de dados adicionais para representação do problema do SED para o modelo de feromônio da ACO. Para isso, foram utilizadas duas estruturas de dados, que são: Árvore Geradora e Árvore Reversa.

Árvore Geradora

A forma que a Árvore Geradora foi concebida para o sistema, com base em (Silva *et al.* , 2010), pode ser observada na Figura 5.2.

Pela Figura 5.2, podem ser observadas as seguintes características: a Figura 5.2a representa uma instância de um SED qualquer; a Figura 5.2b modela a Árvore Geradora para o SED apresentado. Na estrutura da Árvore Geradora, tem-se, para cada linha desta, as cargas do sistema, sendo a primeira a fonte de geração (representada pela carga de número 1); em seguida, são representadas as chaves filhas, que são as chaves nas quais existem caminhos de conexão entre ambas. A carga que possui ligação para uma outra de número zero (0), indica que esta é uma carga terminal do sistema.

Dessa forma, através da utilização desta estrutura, o algoritmo da ACO terá a capacidade de descobrir rapidamente os caminhos da raiz (fonte de geração) até as folhas (cargas terminais) desta árvore, através de uma busca em largura na árvore. Logo, esta estrutura será elementar

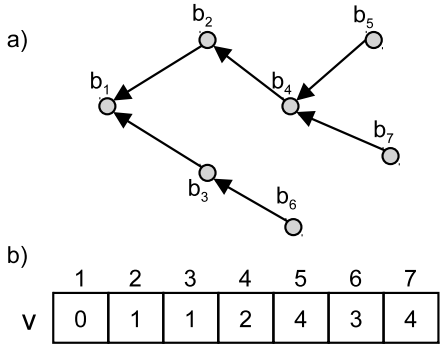


Figura 5.3: *Representação da árvore reversa para o SED, no qual (a) mostra sua representação gráfica e (b) sua representação estrutural.*

durante a construção de uma solução para uma formiga, indicando se as soluções elaboradas satisfazem à condição de Árvore Geradora.

Árvore Reversa

A Árvore Reversa constitui uma estrutura muito similar à Árvore Geradora. Porém, ao invés de indicar um caminho da raiz às folhas de uma árvore, trabalha no sentido oposto, partindo de uma das folhas (ou outro ponto qualquer), visando chegar à raiz desta estrutura de árvore. De uma maneira formal, se cada barra é identificada por um valor inteiro no intervalo $[1..n]$, então $v[i]$, $i \in \{1,...,n\}$ armazena o inteiro que identifica o pai de i na Árvore Geradora. Para um melhor entendimento, a Figura 5.3 ilustra esta estrutura.

Pela Figura 5.3, são observadas as seguintes características: a Figura 5.3 mostra, graficamente, a Árvore Reversa do SED na Figura 5.3a; pela Figura 5.3b, é obtida esta mesma árvore, porém pela forma estrutural utilizada pelo algoritmo. Novamente, a carga que tiver uma representação de ligação de carga pai igual ao número zero (0) indicará uma característica diferente das demais, correspondendo, neste caso, à raiz da árvore, ou seja, à fonte de geração.

Por fim, a utilização desta estrutura será importante na ocorrência de uma falha, pois permite ao algoritmo percorrer de forma eficiente as soluções construídas até a fonte de geração. Desse modo, durante este percurso, é possível verificar se alguma linha da solução teve sobrecarga, o que seria necessário para a escolha de um caminho alternativo pelas formigas do algoritmo.

5.4 Aplicação da Formiga Paraconsistente no Problema de Restabelecimento de Sistemas de Distribuição

Para experimentar a solução do problema de restabelecimento de sistemas elétricos de potência sugerido nesse capítulo, foi utilizada uma instância de um SED denominada *"Distribution_System_01"* descrita em (Ramirez-Rosado e Bernal-Agustin , 1998), que é composta por uma unidade geradora, 201 cargas consumidoras, 39 linhas alternativas (chaves NA) e 37 chaves

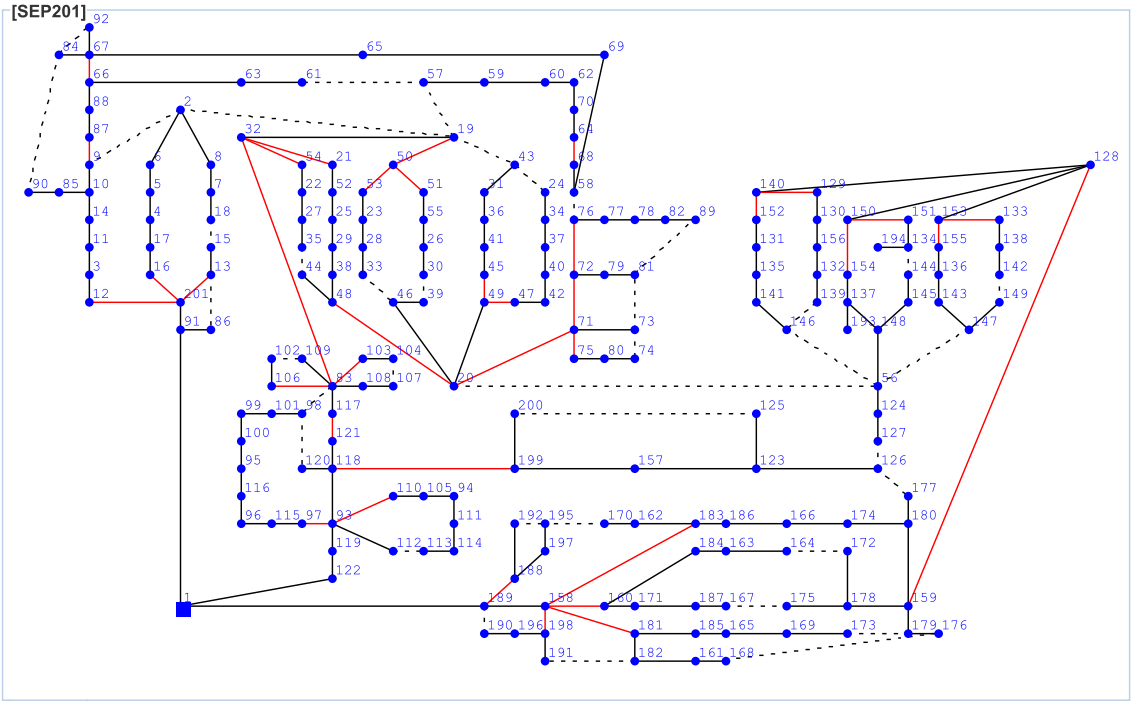


Figura 5.4: *Sistema Elétrico de Distribuição utilizado como exemplo para o algoritmo proposto. As linhas pontilhadas representam linhas com chaves NA e as linhas vermelhas representam linhas com chaves NF.*

normalmente fechadas (NF). A ilustração deste sistema pode ser observada na Figura 5.4.

5.4.1 Configurações Básicas e resultados

As configurações básicas utilizadas para execução do algoritmo proposto estão apresentadas na Tabela 5.1. Esses parâmetros foram definidos empiricamente. O parâmetro δ, que determina a variação da especialidade conforme definido no Capítulo 4, foi ajustado para um valor alto, a fim de que as decisões, considerando a lógica paraconsistente, fossem utilizadas principalmente nas últimas iterações de cada tentativa do algoritmo proposto.

A Tabela 5.2 apresenta os resultados obtidos para várias situações de falhas no sistema,

Tabela 5.1: *Tabela dos parâmetros utilizados para o problema de restabelecimento de SED*

Parâmetro	Valor
α	2
β	1
ρ	0.01
δ	4.0
τ_{max}	0.999
τ_{min}	0.001
Máx. Tentativas	2
Número Iterações	10
Número de Formigas	30

Tabela 5.2: *Resultados obtidos para o problema de restabelecimento de SED*

# teste	Linha da Falha	P %	NA	NF
1	1-91	82.6	19-57, 20-56	20-48
2	1-91	82.6	58-76, 19-57, 20-56	72-76, 20-48
3	1-122	54.5	126-177, 58-76	117-121
4	1-122	54.5	2-19, 126-177	117-121
5	1-189	62.9	126-177, 58-76	117-121
6	12-201	86.8	2-9	
7	19-32	93.9	19-43	
8	56-148	99.2	126-127	

escolhidas aleatoriamente. A primeira coluna mostra uma situação de falha. A segunda coluna indica o percentual de cargas atendidas com a situação de falha. As colunas 3 e 4 apresentam a melhor sequência de chaves normalmente aberta (NA) e normalmente fechada (NF) encontrada pelo algoritmo, respectivamente, sendo possível uma situação de falha possuir várias soluções.

As soluções apresentadas na Tabela 5.2 estão ilustradas nas Figuras 5.5, 5.6, 5.7 e 5.8.

Pode-se observar que o algoritmo proposto obteve soluções válidas para respectiva falha, sendo, na maioria das vezes, em um intervalo médio de tempo menor do que 1 segundo. As únicas exceções estão a cerca das falhas $1-122$, $1-189$ e $1-91$. Para esta última, a otimização pelo algoritmo da ACO foi a mais difícil de obter um desempenho satisfatório, tendo até mesmo encontrado soluções não tão qualificadas para o problema, em alguns casos (5 alterações de chaves). As soluções encontradas por outros algoritmos, como em (Lambert-Torres *et al.* , 2009) são as mesmas, a diferença está no tempo de computação que ficou em torno de um segundo. Apresenta-se na Seção 5.4.2 uma análise de complexidades desse algoritmo.

Uma característica que deve ser observada concerne à heurística utilizada para escolha das chaves NA. Como a preferência é por aquelas chaves mais próximas à região de falha, as situações que necessitam de apenas uma alteração de chave para satisfazer o problema tiveram os melhores resultados. Isso ocorre, pois, para estes casos, era necessário somente o tempo para calcular tal lista de chaves para que o algoritmo pudesse fazer a escolha por uma destas que são a solução para o problema.

Assim, como pode ser observado nos resultados, a algoritmo de Formiga Paraconsistente apresentou um desempenho satisfatório sobre a instância utilizada, visto a complexidade do problema tratado, sendo uma estratégia fortemente indicada para resolução de problemas desta natureza.

5.4.2 Análise da Complexidade da Solução

A complexidade da solução de Formigas Paraconsistentes para o problema de SED, usando a notação-O de complexidade, descrita em (Cormem *et al.* , 2002), pode ser dada pela Equação 5.2.

$$O(n_f.h.n_i.n_t) \tag{5.2}$$

Onde:

- n_f - é o número de formigas
- h - é o tamanho da árvore geradora
- n_i - representa o número de iterações
- n_t - representa o número de tentativas

Por exemplo, considerando os parâmetros utilizados para resolução do problema de restabelecimento de SED deste trabalho, com $n_f = 30$, $h = 100$, $n_i = 10$ e $n_t = 2$. esse algoritmo realiza em torno de 60 mil operações para encontrar uma solução para o problema, o que é realizado em menos de um segundo de processamento na maioria das arquiteturas atuais.

5.5 Considerações Finais

Nesse capítulo é introduzida uma formulação da Formiga Paraconsistente para problemas de otimização em espaço de busca discreto. Uma implementação para o problema de Restabelecimento de Sistemas de Distribuição, usando Formigas Paraconsistentes é apresentada e discutida nesse capítulo.

A Colônia de Formigas Paraconsistente é adaptada para escolher uma sequência de chaves, após a ocorrência de uma falha, de modo a restabelecer todas as possíveis cargas do sistema, respeitando as restrições impostas neste problema de espaço de busca discreto.

A resolução do problema de restabelecimento de SED, usando essa técnica apresentou resultados eficientes, confirmando o algoritmo de Formigas Paraconsistentes como uma promissora alternativa para aplicação em problemas combinatórios.

No próximo capítulo, é apresentada e discutida uma abordagem da Formiga Paraconsistente aplicada em problemas de otimização em espaço de busca contínuo, a fim de testar e avaliar o modelo proposto. Utiliza-se especificamente, para teste e avaliação, o problema de Despacho Econômico de Carga que envolve a otimização no domínio contínuo, que não é a aplicação tradicional da metaheurística ACO. Conforme já foi discutido no capítulo 2, a metaheurística ACO foi originalmente concebida para problemas de otimização no domínio discreto.

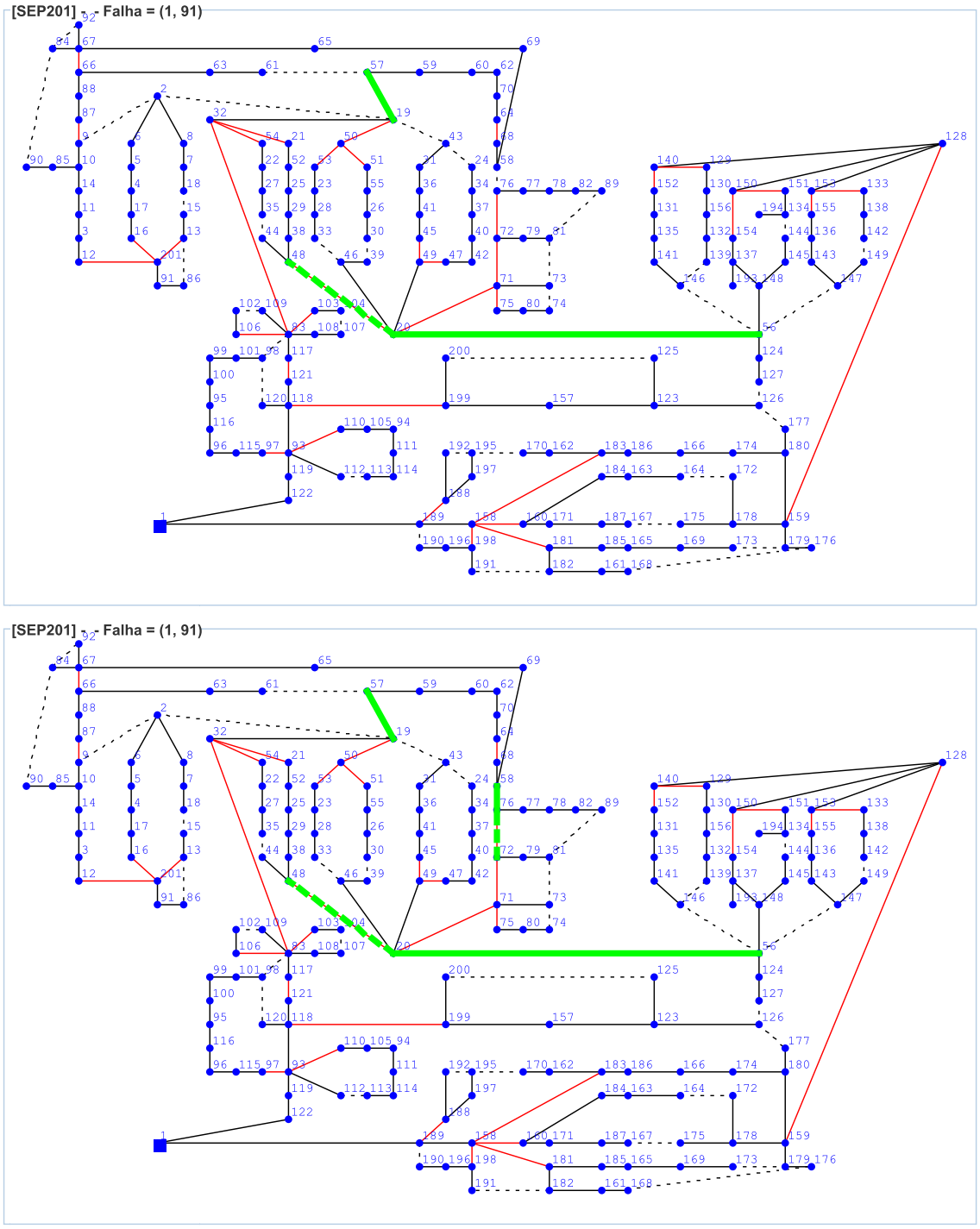


Figura 5.5: *Ilustração das soluções encontradas pelo algoritmo híbrido de ACO e lógica paraconsistente para restabelecimento de SED, testes 1 e 2. A linha verde em cada figura indica as linhas das chaves que tiveram seus estados (aberta ou fechada) alterados na solução encontrada.*

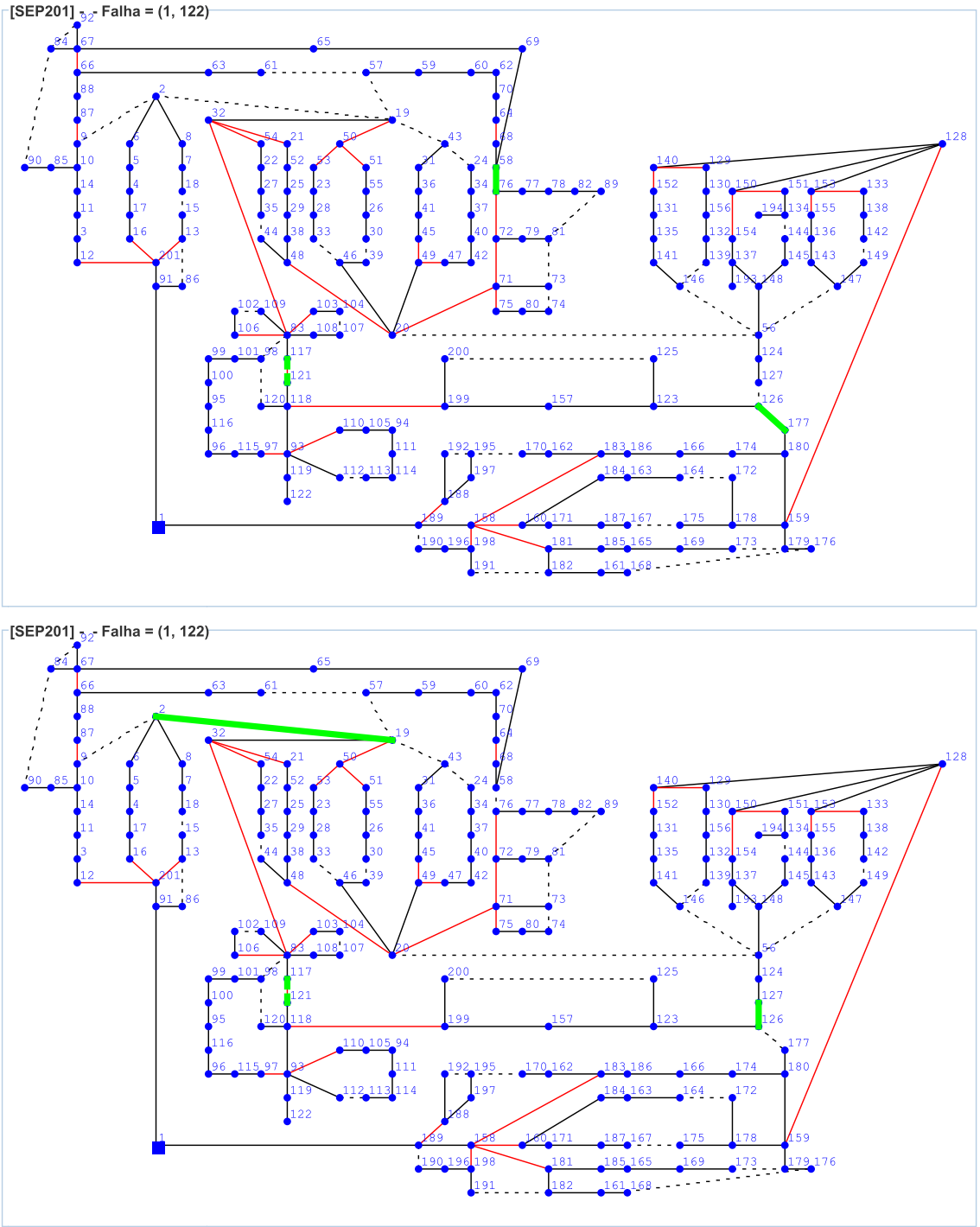


Figura 5.6: *Ilustração das soluções encontradas pelo algoritmo híbrido de ACO e lógica paraconsistente para restabelecimento de SED, testes 3 e 4.*

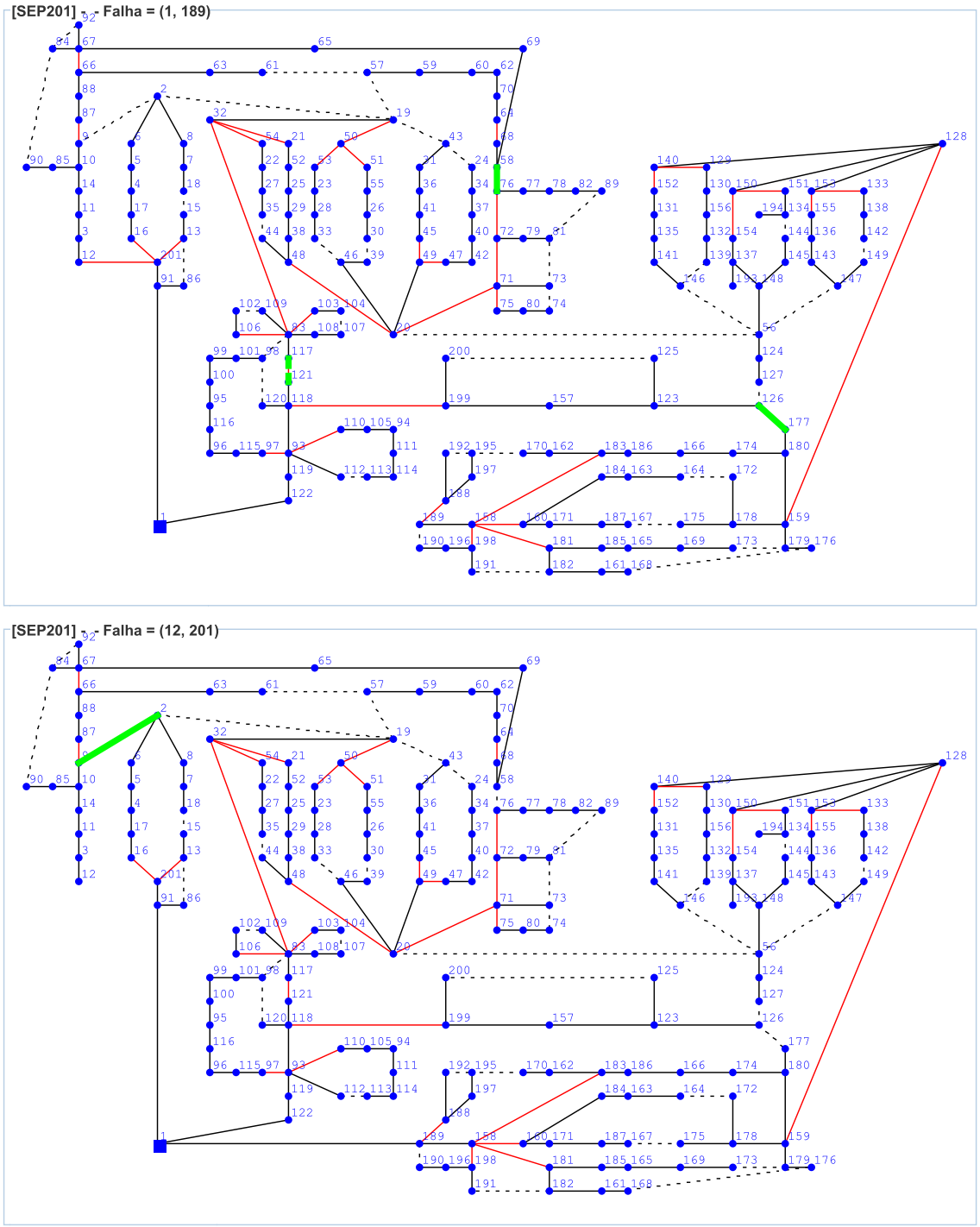


Figura 5.7: *Ilustração das soluções encontradas pelo algoritmo híbrido de ACO e lógica paraconsistente para restabelecimento de SED, testes 5 e 6.*

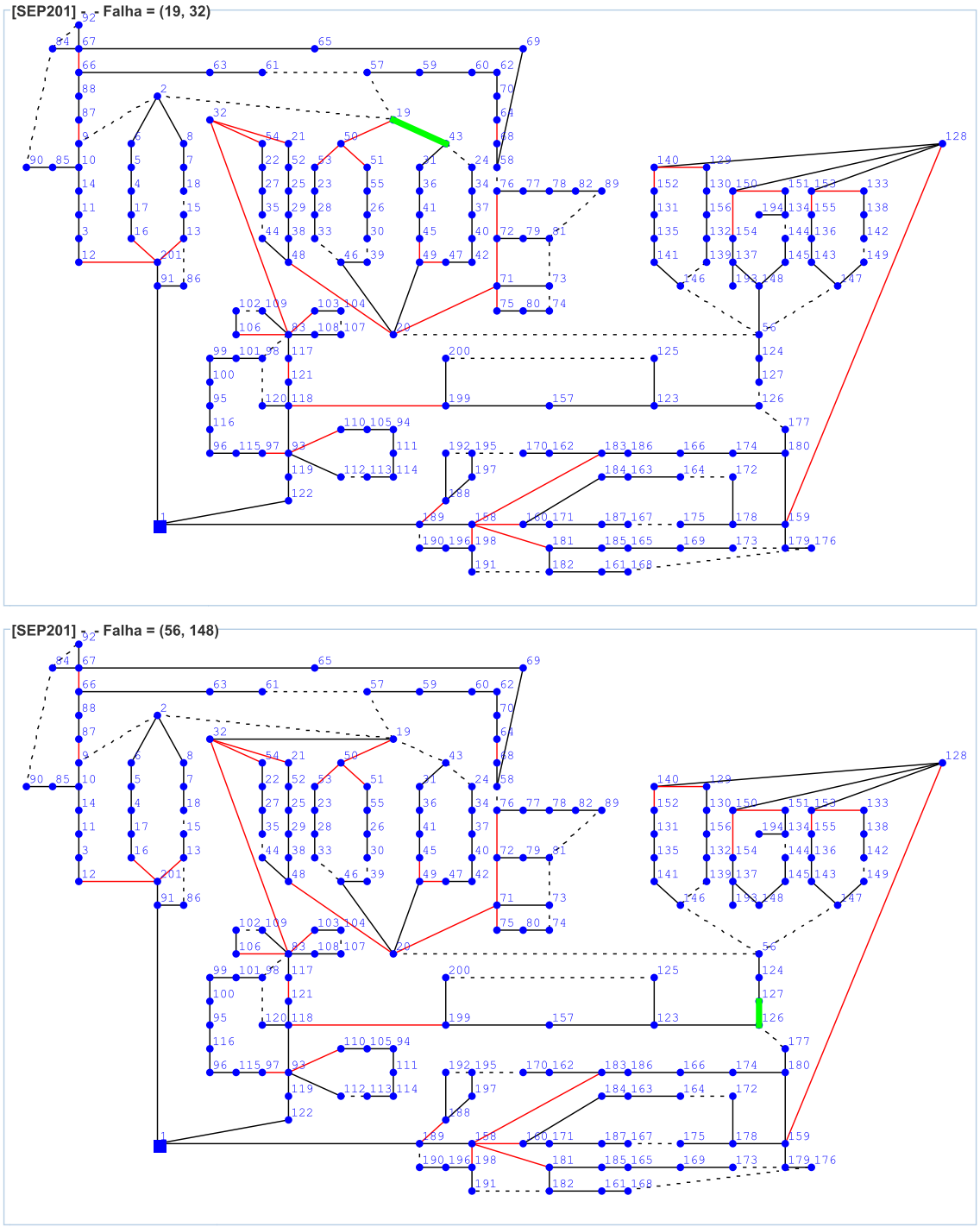


Figura 5.8: *Ilustração das soluções encontradas pelo algoritmo híbrido de ACO e lógica paraconsistente para restabelecimento de SED, testes 7 e 8.*

Capítulo 6

Aplicação da Formiga Paraconsistente em Espaço de Busca Contínuo

Nesse capítulo, apresenta-se uma implementação da Formiga Paraconsistente para solução de um problema de otimização em espaço de busca contínuo. Para isso, utiliza-se a Formiga Paraconsistente para outro problema interessante no campo da Engenharia Elétrica denominado Despacho Econômico de Carga que, diferente do problema de otimização combinatória apresentado no Capítulo 5, está relacionado a um problema de otimização no domínio contínuo.

6.1 Otimização em Espaço de Busca Contínuo

O problema de Otimização em Espaço de Busca Contínuo, tem uma formulação similar ao problema de Otimização em Espaço de Busca Discreto, conforme apresentado na Seção 5.1. Mas diferente do formulação discreta, nos problemas de otimização no domínio contínuo, as variáveis de otimização $X_i, i = 1,2,...n$ são contínuas, ou seja, pertencem ao conjunto real, $X_i \in \mathbb{R}, i = 1,...,n$.

A principal diferença da otimização contínua para a otimização combinatória é o fato que a espaço de busca não é finito. Cada uma das variáveis de otimização do problema contínuo podem assumir um número infinito de valores.

Diversos problemas do mundo real podem ser interpretados como um problema de otimização no espaço de busca contínuo. Um exemplo é a escolha dos valores dos parâmetros para processos industriais, ou ainda um algoritmo para treinar uma rede neural que é usada para diagnóstico médico.

Na próxima Seção é apresentado um problema de otimização em espaço de busca contínuo no campo de Sistemas Elétricos de Potência, denominado Despacho Econômico de Carga. A Formiga Paraconsistente é adaptada para encontrar as potências, que devem ser geradas nas diversas unidades geradoras, a fim de minimizar o custo total de produção.

6.2 O Problema do Despacho Econômico

O Problema do Despacho Econômico em Sistemas Elétricos de Potência visa utilizar de forma ótima as unidades geradoras de um Sistema Elétrico (Chowdhury e Rahman , 1990; Coelho e Mariani , 2006; da Silva , 2007; Musirin *et al.* , 2009). O objetivo é minimizar o custo de produção de energia elétrica, levando-se em consideração algumas restrições de operação do sistema.

A função objetivo f a ser minimizada pelo algoritmo de otimização pode ser formalmente apresentada pela Equação 6.1.

$$f = \sum_{i}^{n} F_i(P_i) \tag{6.1}$$

Onde:

- F_i representa o custo de geração da Unidade Geradora i;
- P_i representa a potência elétrica gerada pela Unidade Geradora i;
- n é o número de unidade Geradoras.

F_i é a função de custo de combustível para a unidade geradora i e é dada pela Equação 6.2.

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i \tag{6.2}$$

Onde a_i, b_i e c_i são restrições das características do gerador i.

Se considerarmos o efeito de Ponto de Válvula, dados pelos parâmetros e_i e f_i, a função custo é modificada para a Equação 6.3.

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i + |e_i sen(f_i(P_i^{min} - P_i))| \tag{6.3}$$

Algumas das restrições desse problema estão representadas nas Equações 6.4 e 6.5.

$$\sum_{i}^{n} P_i - P_L - P_D = 0 \tag{6.4}$$

$$P_i^{min} < P_i < P_i^{max} \tag{6.5}$$

A Equação 6.4 representa a restrição de igualdade entre o suprimento e a demanda de potência. Nessa equação P_D representa a demanda da carga total e P_L são as perdas de transmissão.

A Equação 6.5 representa a restrição dos limites mínimos e máximos da capacidade de geração de potência das Unidades Geradoras.

6.2.1 Comparação com o problema Anterior

Diferente do problema de Restabelecimento de Sistemas Elétricos de Distribuição, que envolve a seleção de um conjunto discreto de chaves para sua solução, no problema de Despacho Econômico, o objetivo é definir um conjunto de valores reais (no domínio contínuo), que defina um balanceamento ótimo de produção das Unidades Geradoras a fim minimizar o custo de produção.

Conforme já foi discutido no capítulo 2, o algoritmo original da metaheurística ACO não está preparado para ser utilizado nessa classe de problemas. Nos problemas do domínio discreto, a solução envolve encontrar combinações ou permutações de um conjunto discreto de componentes do problema. Por exemplo, no caso do problema do caixeiro viajante, a solução envolve encontrar a permutação das cidades (que definem um roteiro se a última cidade é ligada a primeira), que totalize a menor distância percorrida. O problema de domínio contínuo envolve encontrar valores reais que aplicados a função de avaliação, retorne o menor (ou maior) valor possível.

6.2.2 Revisão de Métodos Heurísticos e Metaheurísticos

Em (Chowdhury e Rahman , 1990), o autor faz um revisão sobre as publicações relacionadas ao tema de despacho econômico. Esse trabalho é posterior ao trabalho de Happ (Happ , 1977) e o grupo de trabalho IEEE (Happ *et al.* , 1981), que revisou o tema num período anterior, de 1920 até 1979. A contribuição desse trabalho está relacionado a apresentação e discussão dos artigos publicados entre 1977 até 1988 para quatro áreas relacionadas ao despacho econômico identificadas nesse trabalho: fluxo de potência ótimo, despacho econômico em relação ao controle automático de geração, despacho dinâmico e despacho econômico com fontes de geração não convencionais.

No trabalho (Sinha *et al.* , 2003) são apresentadas técnicas de programação evolucionária como uma alternativa para tratar problemas de programação não linear. Neste trabalho são comparadas diversas variações do algoritmo de programação evolucionária para problemas de diversas dimensões de despacho econômico de carga. As variações de programação evolucionária consideradas diferem com relação aos critérios de mutação adotados como: mutação baseada em distribuição gaussiana, mutação Cauch, mutação baseada na média ponderada das distribuições anteriores. O objetivo desse trabalho é comparar a performance e a convergência dessas diversas variações de Programação Evolucionária quando aplicados a Despacho Econômico de Carga.

Os autores de (Coelho e Mariani , 2006) apresentam uma solução híbrida de estratégias evolutivas (para etapa de busca global de soluções) e o método quase newton (para etapa de busca local) aplicados ao problema do despacho econômico. Os resultados obtidos são aplicados em três problemas considerando ponto de válvula. Os resultados são comparados com resultados de outras estratégias para o mesmo problema.

O trabalho (Chakraborty *et al.* , 2011) implementa uma estratégia que explora um mecanismo híbrido envolvendo um enxame de partículas modificado para se mover considerando a teoria de mecânica quântica, onde a posição e a velocidade das partículas são alteradas das mais diversas formas, a fim de explorar melhor o espaço de busca. Os resultados são promissores se comparados com outras técnicas correspondentes.

Em (Affijulla e Chauhan , 2011) apresenta-se uma abordagem para o problema de Despacho Econômico baseado num Algoritmo de Pesquisa Gravitacional. A ideia dessa estratégia é realizar a pesquisa no espaço de busca do problema considerando um modelo, simples de aplicar, baseado nas leis de movimento e leis gravitacionais de Newton. O algoritmo tem potencial para resolução de problemas de otimização complexos.

6.2.3 Revisão da Otimização por Colônia de Formigas

O trabalho (Hou *et al.* , 2002) apresenta uma variação da metaheurística ACO denominada ACO generalizada (GACO - Generalized Ant Colony Optimization) para resolver problemas de otimização complexos, não convexos e não lineares. O autor estuda as propriedades de convergência do GACO e aplica a variante proposta do ACO para teste e avaliação do método proposto no problema de Despacho Econômico. Os resultados apresentam ser versáteis, robustos e eficientes.

No artigo (El-Wahed *et al.* , 2008) apresenta-se uma solução para o problema de despacho econômico através de um híbrido fracamente acoplado de Colônia de Formigas com uma versão do Algoritmo Simplex modificado. A Colônia é usada para produzir a soluções globais e iniciais para o problema e o método Simplex modificado é usado para realizar busca local para as soluções encontradas pelas Colônias de Formigas. As soluções são comparadas com outros trabalhos relacionados.

6.3 Formigas Paraconsistentes em Espaço de Busca Contínuo

Diversas pesquisas foram realizadas na tentativas de adaptar a metaheurística ACO a fim de tornar possível a sua aplicação aos problemas de otimização no domínio contínuo, como em (Bilchev e Parmee , 1995; Dréo e Siarry , 2002; Leguizamón e Coello , 2010; Mathur *et al.* , 2000; Socha , 2010; Socha e Dorigo , 2008).

No trabalho de (Bilchev e Parmee , 1995), denominado *Continuous ACO* (CACO), as formigas começam o processo de busca num ninho situado em algum ponto do espaço de busca. As boas soluções da colônia são armazenadas com um conjunto de vetores. As formigas, a cada iteração escolhem probabilísticamente um dos vetores, para prosseguir a busca a partir desse.

Em (Dréo e Siarry , 2002), denominado *Continuous Interacting Ant Colony*(CIAC), está apresentado uma outra extensão da ACO. No CIAC as formigas se comunicam de duas formas:

através do depósito de feromônio no espaco de busca e também através de comunicação direta. As formigas se movem no espaço de pesquisa guiadas pela comunicação com as outras formigas e também graças a atração exercida pelo feromônio deixado no caminho.

O trabalho de (Socha , 2010; Socha e Dorigo , 2008), denominado *ACO extended to continuous* ($\text{ACO}_{\mathbb{R}}$), apresenta a extensão que é considerada na implementação do híbrido discutido nesse capítulo. A ideia fundamental do $\text{ACO}_{\mathbb{R}}$ é considerar uma função de densidade de probabilidade (PDF), ao invés da função de probabilidade discreta usada no método ACO original, conforme ilustrado na Figura 6.1. A Figura 6.1a, ilustra uma função de distribuição de probabilidade discreta $P_d(c_{ij}|s^p)$ para um conjunto finito de componentes $c_{i1}, c_{i2}, ..., c_{i10}$. A Figura 6.1b, representa a função de densidade de probabilidade contínua $P_c(x|s^p)$ com o intervalo de $x \in [x_{min}, x_{max}]$. Nas duas Figuras o eixo y indica a probabilidade de p. As duas distribuições satisfazem $\sum_j p(c_{ij}|s^p) = \int_{x_{min}}^{x_{max}} p(x|s^p)dx = 1$.

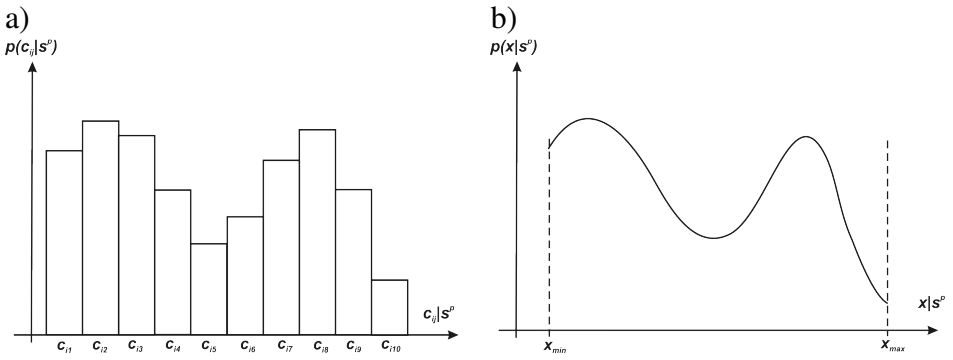


Figura 6.1: *Ilustração das funções de distribuição de probabilidades. a) Distribuição discreta, b) Distribuição contínua*

No ACO para problemas discretos, o feromônio está associado a um conjunto discreto de componentes do problema, o que torna possivel a sua representação através de uma tabela de feromônio. No caso de problemas contínuos isso não é possível. Para tanto, o $\text{ACO}_{\mathbb{R}}$ usa um *Arquivo de Soluções* como uma forma de descrever a distribuição do feromônio no espaço de pesquisa de soluções. Cada linha do arquivo contém uma solução completa do problema. Cada coluna representa uma variável diferente da função a ser otimizada. Dessa forma, o modelo de tabela de feromônio do ACO para problemas discretos pode ser visto como uma memória implícita do histórico da pesquisa. O novo modelo para representar feromônio do $\text{ACO}_{\mathbb{R}}$ é uma memória explícita do histórico da pesquisa (Socha e Dorigo , 2008).

A Figura 6.2 mostra a estrutura do *Arquivo de Soluções*. O Arquivo de soluções representado na Figura contém k soluções para um problema de otimização contínuo com n variáveis. $\text{ACO}_{\mathbb{R}}$ mantém o histórico do processo de pesquisa nas k linhas do Arquivo de Soluções A. O valor da j-ésima variável da i-ésima solução no Arquivo de Soluções A é denotado por s_i^j. Na figura o valor $f(s_i)$ representa o valor da função de otimização para a solução s_i. A coluna w_i representa o peso associado a cada solução. Na próxima seção é apresentado como esse valor é calculado.

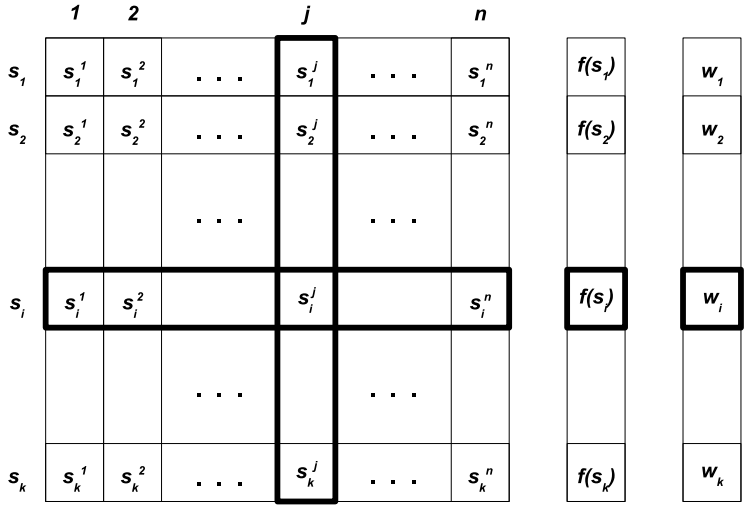


Figura 6.2: *Estrutura do Arquivo de Soluções*

Resumidamente, o algoritmo $\text{ACO}_{\mathbb{R}}$ trabalha da seguinte forma. Primeiro, o Arquivo de Soluções é inicializado com k soluções aleatórias. Esse conjunto de soluções aleatórias é ordenado de acordo com a sua qualidade (i.e., o valor da função objetivo $f(s_i)$). Então, a cada iteração, um conjunto de m soluções, na vizinhança das melhores soluções de A é construído de forma probabilística pelas formigas da colônia (esse passo está detalhado na seção 6.3.1). Estas soluções podem ser melhoradas através de algoritmos de busca local ou técnica de gradiente. O conjunto de $k+m$ soluções é reordenado de acordo com a sua qualidade. Desse conjunto ordenado, as m piores soluções são eliminadas do Arquivo de Soluções A. Esse processo é repetido nas próximas iterações até que a condição de término do processo é alcançada (máximo número de iterações, convergência da colônia ou outro critério que seja adotado para determinar o fim da busca por soluções).

A seção 6.3.1 detalha o processo de construção probabilística do algoritmo $\text{ACO}_{\mathbb{R}}$.

6.3.1 Construção de Soluções Probabilísticas

O processo de construção de novas soluções é realizado de maneira incremental. Primeiro é escolhida uma solução de A na vizinhança das melhores soluções de forma probabilística. A probabilidade p_i de uma formiga escolher a solução i é dada pela Equação 6.6.

$$p_i = \frac{w_i}{\sum_{r=1}^{k} w_r} \tag{6.6}$$

Onde w_i é o peso associado com a solução i. O peso associado a cada solução pode ser calculado de várias formas. Para esse implementação será adotado a função Gaussiana da Equação 6.7, onde $g(\mu,\sigma) = g(1,qk)$, conforme sugerido no trabalho de (Socha , 2010).

$$w_i = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(i-1)^2}{2(qk)^2}} \tag{6.7}$$

Onde q é um parâmetro do algoritmo e k é o tamanho do Arquivo A. O parâmetro q tem a função de determinar a localidade no processo de pesquisa. É similar ao parâmetro de vizinhança no algoritmo ACO para problemas discretos. Quanto menor o valor de q, maior a probabilidade de escolher a melhor solução do Arquivo de Soluções na próxima iteração. Para a média μ é adotado o valor 1 a fim de garantir que as melhores soluções, no início do arquivo A, tenham uma maior probabilidade de serem escolhidas nesse processo de construção de soluções.

Pode-se observar a semelhança da Equação 2.1 para problemas discretos, com a Equação 6.6 para problemas contínuos. É a partir dessa equação de probabilidade que o algoritmo vai capturando a experiência da colônia no processo de busca de soluções. Para implementação do Algoritmo de Formiga Paraconsistente, a aplicação dessa função estará condicionada ao resultado de um teste, usando a lógica paraconsistente, conforme é apresentado na Seção 6.3.2.

Outro detalhe importante é que essa distribuição é fixa, pois está relacionada a posição relativa das soluções no Arquivo de Soluções. Do início ao final do processo de construção de soluções o sorteio probabilístico será sempre o mesmo. A implementação do híbrido, proposto nesse trabalho, irá privilegiar a escolha da melhor solução do Arquivo A, nas últimas iterações, conforme é explicado detalhadamente na próxima Seção.

Depois que a formiga escolheu uma solução de A, ela pode construir uma nova solução. A formiga trata individualmente cada variável $j = 1,2,...n$ do problema. Cada formiga toma o valor s_i^j da solução i e variável j e sorteia valores aleatórios em torno da vizinhança desse valor. Para esse sorteio usa-se uma função de densidade de probabilidade (PDF). Várias funções podem ser utilizadas, mas nesse trabalho, assim como no trabalho de Socha (Socha e Dorigo , 2008), a função de densidade utilizada é a função Gaussiana dada pela Equação 6.8.

$$g(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{6.8}$$

Os valores da média μ e do desvio σ são dados pela Equação 6.9 e pela Equação 6.10, respectivamente.

$$\mu = s_i^j \tag{6.9}$$

$$\sigma = \xi \sum_{r=1}^{k} \frac{|s_r^j - s_i^j|}{k-1} \tag{6.10}$$

Ou seja, para a média μ é adotado o valor da variável j da solução i que está sendo ajustada pela formiga na iteração. E para o desvio σ é adotado a média do módulo das diferenças da mesma variável (mesma coluna j, conforme ilustrado na Figura 6.2), nas outras soluções de A. Um parâmetro extra é a variável ξ que é utilizada para ajustar a velocidade da convergência da

```
1               (...)
2               if (decideParaconsistente()) {
3                   i = 1;
4               } else {
5                   i = escolhaProximo();
6               }
7               (...)
```

Figura 6.3: *Pseudo-código da modificação do algoritmo para $ACO_{\mathbb{R}}$*

```
1     boolean decideParaconsistente() {
2         varepson = 1 - Math.pow(iteracao / maxIteracoes, delta);
3         x = f(1);
4         y = f(2);
5         z = f(k);
6         mu = 1;
7         lambda = Math.abs(x - y) / Math.abs(x - z);
8         if (diagnostico(mu, lambda, varepson) == 1) {
9            return true;
10        }
11        return false;
12    }
```

Figura 6.4: *Pseudo-código da função para decisão paraconsistente*

colônia. O parâmetro ξ tem um efeito similar a taxa de evaporação do feromônio para do ACO para problemas discretos. Quanto maior o valor de ξ, menor a velocidade da convergência do algoritmo.

6.3.2 A Colônia de Formigas Paraconsistentes

A híbrido do $ACO_{\mathbb{R}}$ com a lógica paraconsistente é construído com a mudança na forma como as formigas escolhem as soluções do Arquivo de Soluções. A modificação ocorre no trecho do programa indicado na Figura 6.3.

Conforme representado no fragmento de código da Figura 6.3, é feito um teste para verificar se a formiga, na iteração corrente pode escolher de forma paraconsistente. Se o resultado for verdadeiro, a formiga escolhe a melhor solução do Arquivo de Soluções ($i = 1$). A partir dessa melhor solução a formiga tenta ajustar suas variáveis visando melhorar ainda mais essa solução.

A função que determina se a decisão paraconsistente é assertiva está representada no pseudo-código da Figura 6.4. Como não existe o feromônio representado de forma explícita no algoritmo $ACO_{\mathbb{R}}$, para essa implementação está sendo adotado o valor da função objetivo da melhor solução $f(s_1)$ do Arquivo de Soluções A, a segunda melhor solução $f(s_2)$ e a pior solução $f(s_k)$ para a tomada de decisão. O valor do grau de crença é adotado como $\mu = 1$, como nos exemplos anteriores, e o grau de descrença é dado por $\lambda = \frac{|f(s_1)-f(s_2)|}{|f(s_1)-f(s_k)|}$.

6.4 Aplicação da Formiga Paraconsistente no Problema de Despacho Econômico de Carga

Para a aplicação das Formigas Paraconsistentes no problema do despacho econômico ainda é necessário implementar alguma forma de tratar as restrições de igualdade, dada pela Equação 6.4 e de limites de capacidades das unidades de geração de energia, dada pela Equação 6.5.

6.4.1 Tratamento das restrições do problema

Nesse trabalho usa-se um critério similar ao de Coelho (Coelho e Mariani , 2006) para essas restrições. No caso da restrição de capacidade das unidades usa-se um ajuste aleatório dado pelo parâmetro w da Equação 6.11.

$$s'^{j}_{i} = \begin{cases} s^{j}_{i} - w.rand[0,1](P^{max}_{i} - P^{min}_{i}) & \text{se} \quad s^{j}_{i} > P^{max}_{i} \\ s^{j}_{i} + w.rand[0,1](P^{max}_{i} - P^{min}_{i}) & \text{se} \quad s^{j}_{i} < P^{min}_{i} \end{cases} \tag{6.11}$$

Onde $w = 0.05$ e $rand[0,1]$ é um gerador de número aleatório no intervalo de 0 a 1, inclusive. Este ajuste garante que toda solução construída é factível.

As soluções que não satisfazem a restrição de demanda da Equação 6.4, são penalizadas através da Equação 6.12.

$$f' = \begin{cases} f + q_1|\sum_i^n P_i - P_L - P_D| & \text{se} \quad f < |\sum_i^n P_i - P_L - P_D| \\ f + q_2|\sum_i^n P_i - P_L - P_D| & \text{se} \quad f > |\sum_i^n P_i - P_L - P_D| \end{cases} \tag{6.12}$$

Onde q_1 e q_2 são constantes positivas que determinam a penalidade no caso da solução não atender a demanda (q_1) ou ultrapassar a demanda de energia (q_2). As penalidades podem ser diferentes, por exemplo, $q_1 > q_2$ para garantir que ao menos a demanda de energia seja garantida na solução. Para fins de classificação das soluções no Arquivo de Soluções, considera-se a função de custo com a penalidade da Equação 6.12.

6.4.2 Testes e resultados

Para os testes do algoritmo de Formigas Paraconsistentes para o domínio contínuo, são utilizadas duas instâncias do problema de despacho econômico, descritos em (Sinha *et al.* , 2003; Walters e Sheble , 1993).

Caso 1 - Instância com 3 unidades geradoras

Os dados da instância com três unidades geradoras estão na Tabela 6.1.

Os Parâmetros utilizados com essa instância estão definidos na Tabela 6.2. A demanda a ser atendida nesse problema é de $P_D = 850MW$, não são consideradas perdas na geração $P_L = 0$ e os parâmetros de penalidade são $q_1 = 1500$ e $q_2 = 100$.

Tabela 6.1: *Dados da instância para 3 unidades geradoras.* P_i^{min} *e* P_i^{max} *em MW*

Unidade Geradora i	P_i^{min}	P_i^{max}	a	b	c	e	f
1	100	600	0.001562	7.92	561	300	0.0315
2	100	400	0.001940	7.85	310	200	0.042
3	50	200	0.004820	7.97	78	150	0.063

Tabela 6.2: *Parâmetros utilizados para Instância de 3 Unidades Geradoras*

Parâmetro	Valor
k	50
m	2
ξ	0.01
q	0.01
δ	32.0
Max. Iterações	100
Max. Tentativas	1000

O melhor resultado encontrado pelo algoritmo de Formigas Paraconsistentes está apresentado na Tabela 6.3.

Um comparativo com os resultados do trabalho de (Coelho e Mariani , 2006) estão apresentados na Tabela 6.4. Nessa Tabela *QN* se refere a abordagem denominada Quase-Newton e *CES* se refere a Estratégia Evolutiva, um algoritmo genético adaptado para problemas de domínio contínuo. $CES-QN(1)$ é um híbrido que aplica a *CES* ao problema de otimização e a melhor solução, a cada geração, é utilizada como solução inicial para o método *QN*. $CES-QN(2)$ é o contrário, a melhor solução de *QN* é usada para gerar a população inicial de *CES*. Para esse problema em questão a estratégia de formigas paraconsistentes obteve um resultado melhor do que as outras estratégias comparadas. A Figura 6.5 apresenta o gráfico de convergência do algoritmo de formigas paraconsistentes para essa instância do problema.

Caso 2 - Instância com 13 unidades geradoras

A Tabela 6.5 apresenta os dados para a instância com 13 unidades geradoras do segundo caso de teste.

Para essa instância, os Parâmetros ajustados do algoritmo de formigas paraconsistentes estão

Tabela 6.3: *Melhor solução encontrada pela Colônia de Formigas Paraconsistentes para o problema com 3 unidades geradoras*

Unidade Geradora (i)	Geração (P_i)
1	299.76485
2	400.47236
3	149.76293
$\sum_i^n P_i$	850.00014

Tabela 6.4: *Comparativo com o trabalho de (Coelho e Mariani , 2006)*

Método	Custo ($/h)
QN	8234.58
CES	8234.08
CES-QN (1) e CES-QN (2)	8234.07
Formiga Paraconsistente	**8233.85**

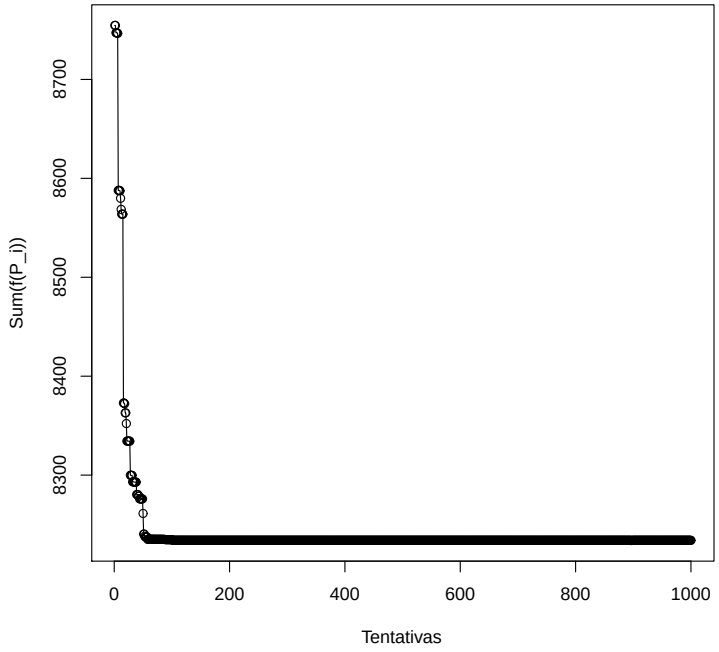


Figura 6.5: *Convergência do algoritmo de $ACO_{\mathbb{R}}$ Paraconsistente para Instância de 3 Unidades de Geração*

Tabela 6.5: *Dados da instância para 13 unidades geradoras. P_i^{min} e P_i^{max} em MW*

Unidade Geradora *i*	P_i^{min}	P_i^{max}	a	b	c	e	f
1	0	680	0.00028	8.10	550	300	0.035
2	0	360	0.00056	8.10	309	200	0.042
3	0	360	0.00056	8.10	307	150	0.042
4	60	180	0.00324	7.74	240	150	0.063
5	60	180	0.00324	7.74	240	150	0.063
6	60	180	0.00324	7.74	240	150	0.063
7	60	180	0.00324	7.74	240	150	0.063
8	60	180	0.00324	7.74	240	150	0.063
9	60	180	0.00324	7.74	240	150	0.063
10	40	120	0.00284	8.60	126	100	0.084
11	40	120	0.00284	8.60	126	100	0.084
12	55	120	0.00284	8.60	126	100	0.084
13	55	120	0.00284	8.60	126	100	0.084

Tabela 6.6: *Parâmetros utilizados para Instância de 13 Unidades Geradoras*

Parâmetro	Valor
k	50
m	2
ξ	0.25
q	0.1
δ	8.0
Max. Iterações	100
Max. Tentativas	1000

Tabela 6.7: *Melhor solução encontrada pela Colônia de Formigas Paraconsistentes para o problema com 13 unidades geradoras*

Unidade Geradora (i)	Geração (P_i)
1	449.09980
2	150.07779
3	300.30645
4	109.88277
5	110.32062
6	110.08205
7	110.07527
8	109.98648
9	110.12568
10	40.26242
11	77.48379
12	55.75505
13	66.54404
$\sum_i^n P_i$	1800.00219

definidos na Tabela 6.6. A demanda a ser atendida nesse problema é de $P_D = 1800MW$, não são consideradas perdas na geração $P_L = 0$ e os parâmetros de penalidade são $q_1 = 500$ e $q_2 = 50$.

O melhor resultado encontrado para o Caso 2 está apresentado na Tabela 6.7.

Um comparativo com os resultados do trabalho de (Coelho e Mariani , 2006) estão apresentados na Tabela 6.8. Para esse problema em questão a estratégia de formigas paraconsistentes obteve um resultado um pouco pior do que as outras estratégias comparadas. Ainda que no algoritmo de formigas paraconsistentes nenhuma estratégia de otimização local seja empregada. A Figura 6.6 apresenta o gráfico de convergência do algoritmo de formigas paraconsistentes para essa instância do problema.

6.4.3 Análise da Complexidade da Solução

A complexidade da solução de formigas paraconsistentes para o problema de Despacho Econômico, usando a notação-O de complexidade, descrita em (Cormem *et al.* , 2002), pode ser dada pela Equação 6.13.

Tabela 6.8: *Comparativo com o trabalho de (Coelho e Mariani , 2006)*

Método	Custo ($/h)
QN	17988.199
CES	18090.446
CES-QN (2)	**17964.878**
Formiga Paraconsistente	18090.70

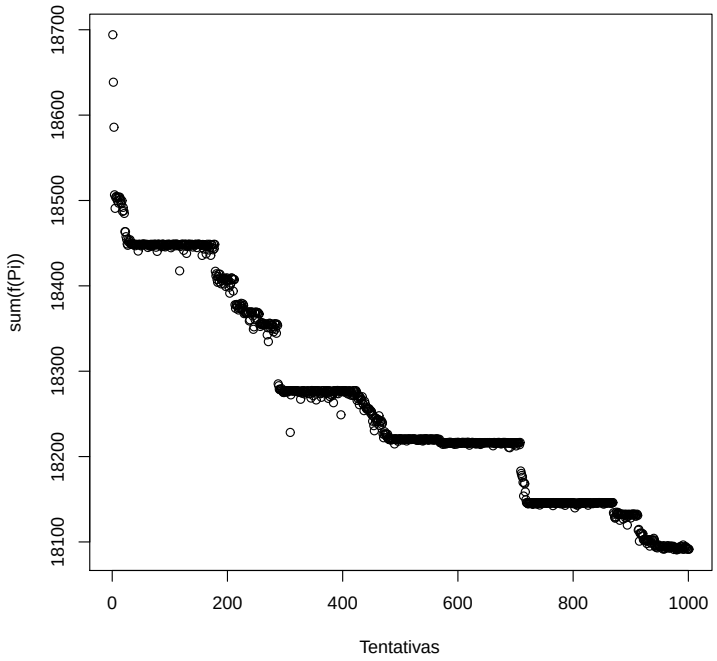


Figura 6.6: *Convergência do algoritmo de $ACO_{\mathbb{R}}$ Paraconsistente para Instância de 13 Unidades de Geração*

$$O(m.k.n.n_i.n_t) \tag{6.13}$$

Onde:

- m - é o número de formigas
- k - é o tamanho do Arquivo de Soluções
- n - é o número de variáveis na função de otimização.
- n_i - representa o número de iterações
- n_t - representa o número de tentativas

Por exemplo, considerando os parâmetros utilizados na solução para o problema com 13 unidades geradoras, onde $m = 2$, $k = 50$, $n = 13$, $n_i = 100$ e $n_t = 1000$, são realizadas em torno de 130 milhões de operações para encontrar a solução do problema de despacho econômico de carga em questão. Num computador com processador Core i5, com 4Gb de memória, num código implementado em linguagem JAVA, cada execução, considerando esses parâmetros foi realizada em torno de 20 segundos.

6.5 Considerações Finais

Nesse capítulo é apresentada uma adaptação do algoritmo da Formiga paraconsistente, para aplicação em problemas de otimização no domínio contínuo. Para tanto, foi utilizado a variação do algoritmo de colônia de formigas denominado $ACO_{\mathbb{R}}$. Nessa estratégia, as soluções são armazenadas numa estrutura de dados denominada Arquivo de Soluções, onde as formigas evoluem as soluções, considerando a experiência da colônia. Uma interpretação da Lógica Paraconsistente é usada, assim como nas implementações anteriores, para tentar absorver a experiência da colônia e forçar a convergência ao final do processo de busca, para a melhor solução encontrada.

O algoritmo é aplicado num problema de Sistemas Elétricos de Potência, denominado Despacho Econômico de Carga. Os resultados são bem satisfatórios, visto que nenhum mecanismo de busca local é utilizado a fim de melhorar os resultados da estratégia de otimização.

No próximo capítulo são apresentadas as conclusões do trabalho desenvolvido nessa tese. Os principais resultados alcançados e algumas sugestões para trabalhos futuros também são discutidos nesse capítulo.

Capítulo 7

Conclusões

Problemas de otimização, seja no domínio discreto ou no domínio contínuo, são encontrados nas mais diversas áreas de conhecimento. Essa diversidade talvez justifique a importância dessa classe de problemas e a quantidade de pesquisa que vem sendo realizadas, nos últimos tempos, relacionadas a esse tema.

A metaheurística de Colônia de Formigas é uma estratégia relativamente nova, apresentada numa área de inteligência artificial que busca desenvolver algoritmos inspirados em sistemas biológicos. Originalmente concebida para resolver problemas de otimização no domínio discreto, a metaheurística de Colônia de Formigas foi logo adaptada para problemas do domínio contínuo.

No processo de tomada de decisão das formigas, durante a busca por soluções pela Colônia de Formigas, existem muitas situações de inconsistência e incerteza, que não são tratados. Por exemplo, no início do processo de busca é natural que a formiga faça uma escolha aleatória entre os caminhos de solução mais prováveis. Considerando-se que a experiência da colônia é determinada pela quantidade de feromônio deixado no espaço de busca, é natural imaginar que no final do processo, as diferenças nas quantidades de feromônio deixadas no caminho devem ter uma avaliação diferente das diferenças observadas no início da busca por soluções.

Em inteligência artificial desenvolve-se novos sistemas de tomada de decisão, que incorporam situações de indecisão e inconsistência que não são admitidos na lógica clássica. Dentre essas lógicas não-clássicas destaca-se a Lógica Paraconsistente. A Lógica Paraconsistente admite outros estados lógicos além do estado verdade e estado falso, como quase-verdade, quase-falso, inconsistente, paracompleto e indecisão. Diversas interpretações podem ser adotadas para a Lógica Paraconsistente, com valores que determinam graus de crença e de descrença sobre as proposições que são avaliadas usando essa lógica.

Este trabalho apresenta uma proposição do algoritmo de Colônia de Formigas Paraconsistente, que é uma estratégia híbrida da interpretação da lógica paraconsistente com três variáveis (grau de crença, grau de descrença e especialidade), com a metaheurística de Colônia de Formigas. O sistema híbrido é normalmente concebido para potencializar as vantagens de cada um dos métodos que estão sendo combinados. No caso do híbrido desse trabalho, a ideia é complementar o potencial de tratar problemas de indecisão e inconsistência, da Lógica Paraconsistente, com o potencial para resolver problemas de computação dificil, do algoritmo de Colônia de For-

miga. Como efeito colateral do sistema híbrido, acrescenta-se a estratégia, a possibilidade de controlar o processo de convergência da colônia para uma solução do problema.

A Formiga Paraconsistente pode ser usada tanto para problemas de otimização no domínio discreto como problemas de otimização no domínio continuo. No domínio discreto, as variáveis do problema de otimização são variáveis discretas e contáveis. Para exemplificar isso, utiliza-se nesse trabalho, um problema característico de otimização em espaço de busca discreto, na área de Sistemas Elétricos de Potência, denominado problema de Restabelecimento de Sistemas de Distribuição. O resultados alcançados com a aplicação da Formiga Paraconsistente para o Problema de Restabelecimento de SED demonstra que o algoritmo híbrido proposto é uma abordagem promissora para essa classe de problema.

Diferente da otimização no domínio discreto, no domínio contínuo, as variáveis do problema de otimização são contínuas. Portanto, na otimização em domínio continuo, o espaço de busca é infinito. Para aplicar e avaliar o algoritmo proposto, a Formiga Paraconsistente é usada para um problema característico do domínio contínuo, na área de Sistemas Elétricos de Potência, denominado problema do Despacho Econômico de Carga.

Os testes realizados com problemas diversos, demonstra que a Formiga Paraconsistente é uma nova e interessante estratégia para resolução de problemas, tanto no domínio discreto, como no domínio contínuo.

7.1 Contribuições desse Trabalho

A principal contribuição dessa pesquisa é a proposição do algoritmo da Colônia de Formigas Paraconsistente, que é o sistema híbrido da Lógica Paraconsistente com o metaheurística, bioinpirada, denominada Colônia de Formigas. Pode-se relacionar ainda as seguintes contribuições desse trabalho:

- Aplicação de uma interpretação da Lógica Paraconsistente no desenvolvimento do algoritmo de Colônia de Formigas Paraconsistente;
- Desenvolvimento de uma nova variação da metaheurística de Colônia de Formigas, com um parâmetro adicional para controle da convergência da Colônia;
- Aplicação do algoritmo proposto num problema de otimização em espaço de busca discreto, exemplificado no problema de Restabelecimento de Sistemas de Distribuição;
- Aplicação da Formiga Paraconsistente num problema de otimização em espaço de busca contínuo, exemplificado para o problema de Despacho Econômico de Carga.

7.2 Sugestões para Trabalhos Futuros

Para trabalhos futuros sugere-se o desenvolvimento de versões mais eficientes e otimizadas do algoritmo de Colônia de Formigas Paraconsistente, com o enfoque no ajuste dos parâmetros a fim de determinar, de uma forma geral, a configuração mais apropriada a cada tipo de problema de otimização. A nova proposta inclui um parâmetro de variação de especialidade que também deve ser ajustado, aumentando a complexidade do ajuste de parâmetros do algoritmo de colônia de formigas original. Sugere-se a aplicação de testes de ajustes paramétricos afim de encontrar a melhor configuração para os problemas aplicáveis ao algoritmo de Formiga Paraconsistente.

Novas pesquisas devem ser realizadas afim de entender melhor a interação dos parâmetros do algoritmo e em especial, os parâmetros relacionados a lógica paraconsistente. Pesquisas e testes também devem ser realizadas com o intuito de interpretar como esses parâmetros influenciam a performance da Formiga Paraconsistente. O momento em que a formiga deve realizar decisões paraconsistente ainda não é bem entendida nesse algoritmo. Então, novos trabalhos devem ser realizados para entender melhor como ocorre a interação entre a lógica paraconsistente e o método probabilístico original.

Em nenhuma das implementações foram considerados algoritmos de busca local. Sugere-se então, o desenvolvimento de testes com algoritmos de busca local que possam melhorar a eficiência da Formiga Paraconsistente.

Sugere-se ainda o desenvolvimento de outras formas de integração da Lógica Paraconsistente e da Colônia de Formigas, com a avaliação dos resultados alcançados. Os outros estados da lógica paraconsistente devem ser explorados a fim de avaliar a influência que podem ter no processo decisório da colônia na busca de soluções ótimas para os problemas.

A aplicação do método proposto em outros problemas de otimização no domínio contínuo e no domínio discreto, também deve ser explorada em outros trabalhos.

Apêndice A

Publicações Associadas ao Trabalho e Realizadas no Período

Capítulos de Livros Publicados

- SILVA, L. E., LAMBERT-TORRES, G., SALGADO, R. M., OLIVEIRA, H. C. B. Hybrid System Based in Ant Colony and Paraconsistent Logic In: Behavior in Insects & Computer Applications ed.Oxford : Nova Science Publishers Inc., 2010

Artigos Completos Publicados em Periódicos

- SILVA, L. E., MARTINS, H. G., COUTINHO, M. P., LAMBERT-TORRES, G., SILVA, L. E. B. The Convergence Control to the ACO Metaheuristic Using Annotaded Paraconsistent Logic. Lecture Notes in Computer Science. v.5821, p.382 - 391, 2009.

- SILVEIRA, T., OLIVEIRA, H. C. B., SILVA, L. E., SALGADO, R. M. Controle de inércia não monotônico na otimização por enxame de partículas. Scientia (Unisinos). , v.20, p.69 - 82, 2009.

Trabalhos Publicados em Anais de Evento (Completo)

- SILVA, L. E., LAMBERT-TORRES, G., MARTINS, H. G., COUTINHO, M. P., SILVA, L. E. B., NETO, J. C. An Application of ACO in System Reconfiguration In: 2010 IEEE PES Transmission and Distribution, 2010, New Orleans. 2010 IEEE PES Transmission and Distribution. , 2010.

- SILVEIRA, T. ; OLIVEIRA, H. C. B. ; SILVA, L. E. ; SALGADO, R. M. ; MATEUS, G. R. . Cosine Function applied to the Inertia Control in the Particle Swarm Optimization. In: 2010 IEEE Congress on Evolutionary Computation, 2010, Barcelona. 2010 IEEE Congress on Evolutionary Computation. New York : IEEE Computer Society Press, 2010.

- SILVEIRA, T. ; SILVA, L. E. ; LAMBERT-TORRES, G. ; OLIVEIRA, H. C. B. . Restabelecimento de Sistemas Elétricos de Potência com a Otimização por Colônia de Formigas. In: XLII Simpósio Brasileiro de Pesquisa Operacional - XLII SBPO 2010, 2010, Bento Gonçalves. XLII SBPO 2010, 2010.

- SILVEIRA, T., OLIVEIRA, H. C. B., SILVA, L. E. Controle de Inércia para Fuga de Mínimos Locais de Funções Não-Lineares na Otimização por Enxame de Partículas In: XXIX Congresso da Sociedade Brasileira da Computação, 2009, Bento Gonçalves. XXIX Congresso da Sociedade Brasileira da Computação. , 2009.

- OLIVEIRA, H. C. B. ; MARTINEZ, M. R. ; SALGADO, R. M. ; SILVA, L. E. ; FIGUEIREDO, T. F. . Clusterização em Redes Sociais através do Simulated Annealing Não-Monotônico. In: XLII Simpósio Brasileiro de Pesquisa Operacional, 2010, Bento Gonçalves. XLII Simpósio Brasileiro de Pesquisa Operacional. Rio de Janeiro : SOBRAPO Press, 2010.

- SILVA, L. E. B., FARIA NETO, A., LAMBERT-TORRES, G., SILVA, L. E., MARTINS, H. G. Algoritmo para Fusão de Imagens usando Colônia de Formigas In: VI Congress of Logic Applied to Technology, 2007, Santos. VI Congress of Logic Applied to Technology. , 2007.

- SILVA, L. E., LAMBERT-TORRES, G., VIEIRA, W. S., SANTOS, C. R., CARMINATI, R. A., MARTINS, H. G. Rede Neural Artificial Aplicada em um Reconhecimento Automático de Voz Independente do Locutor In: VI Congress of Logic Applied to Technology, 2007, Santos. VI Congress of Logic Applied to Technology. , 2007.

Referências Bibliográficas

AAA (1987) *A Logic Programming System Based on a Six-Valued Logic*, Madri, Espanha. AAAI/Xerox Second Intl. Symp. on Knowledge Eng. Citado na pág. 18

Abe (1992) J. M. Abe. *Fundamentals of Annotated Logic*. Tese de Doutorado, FFLCH/USP, SP, Brazil. Citado na pág. 15, 18

Affijulla e Chauhan (2011) S. Affijulla e S. Chauhan. A new intelligence solution for power system economic load dispatch. Em *Environment and Electrical Engineering (EEEIC), 2011 10th International Conference on*, páginas 1 –5. doi: 10.1109/EEEIC.2011.5874614. Citado na pág. 67

Ahuja *et al.* **(2008)** A. Ahuja, S. Das e A. Pahwa. An ais-aco hybrid approach for multi-objective distribution system reconfiguration. Em *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, página 1. doi: 10.1109/PES.2008.4596330. Citado na pág. 52

Bilchev e Parmee (1995) George Bilchev e Ian C. Parmee. The ant colony metaphor for searching continuous design spaces. Em *Selected Papers from AISB Workshop on Evolutionary Computing*, páginas 25–39, London, UK, UK. Springer-Verlag. ISBN 3-540-60469-3. URL http://dl.acm.org/citation.cfm?id=648160.749861. Citado na pág. 67

Boyd e Vandenberghe (2004) S. Boyd e L. Vandenberghe. *Convex Optimization*. Cambridge University Press. Citado na pág. 46

Bullnheimer *et al.* **(1999)** B. Bullnheimer, R. F. Hartl e C. Strauss. A new rank based version of the ant system: A computational study. *Central European Journal for Operations Research and Economics*, 7(1):25–38. Citado na pág. 13

Chakraborty *et al.* **(2011)** S. Chakraborty, T. Senjyu, A. Yona, A.Y. Saber e T. Funabashi. Solving economic load dispatch problem with valve-point effects using a hybrid quantum mechanics inspired particle swarm optimisation. *Generation, Transmission Distribution, IET*, 5(10):1042 –1052. ISSN 1751-8687. doi: 10.1049/iet-gtd.2011.0038. Citado na pág. 67

Chowdhury e Rahman (1990) B.H. Chowdhury e S. Rahman. A review of recent advances in economic dispatch. *Power Systems, IEEE Transactions on*, 5(4):1248 –1259. ISSN 0885-8950. doi: 10.1109/59.99376. Citado na pág. 65, 66

Coelho e Mariani (2006) Leandro S. Coelho e Viviana Cocco Mariani. Otimização de despacho econômico com ponto de válvula usando estratégia evolutiva e método quase-newton. *Learning an Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais*. Citado na pág. viii, 65, 66, 72, 73, 74, 75, 76

Cormem *et al.* **(2002)** Thomas H. Cormem, Charles E. Leiserson, Ronald L. Rivest e Clifford Stein. *Algoritmos Teoria e PrÃ¡tica*. Campus, Rio de Janeiro. Citado na pág. 6, 46, 58, 75

Da Costa (1990) N. C. A. Da Costa. *The Philosophic Importance of Paraconsistent Logic*, volume 11. Bol. Soc. Paranaense of Mathematic. Citado na pág. 16

Da Costa (1974) N. C. A. Da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 11. Citado na pág. 16

Da Costa e Abe (1999) N. C. A. Da Costa e J. M. Abe. *Lógica Paraconsistente Aplicada*. Editora Atlas. Citado na pág. 16

Da Costa *et al.* **(1990)** N. C. A. Da Costa, L. J. Hensche e V. S. Subrahmanian. *Automatic Theorem Proving in Paraconsistent Logics: Theory and Implementation*, volume 3 of *Estudos Avançados, Coleção Documentos*. USP, SP, Brasil. Citado na pág. 15, 17, 18, 22

Da Costa *et al.* **(1991)** N. C. A. Da Costa, V. S. Subrahmanian e C. Vago. The paraconsistent logic $\mathrm{p}\tau$. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 37:139–148. Citado na pág. 17

da Silva (2007) João Ângelo Bosco da Silva. Despacho econômico e ambiental nebuloso em sistemas de energia elétrica. Dissertação de Mestrado, Unicamp. Citado na pág. 65

da Silva Filho (1999) João Inácio da Silva Filho. *Métodos de interpretação da Lógica Paraconsistente Anotada com anotação com dois valores (LPA2v) com construção de algoritmos e implementação de circuitos eletrônicos*. Tese de Doutorado, EPUSP, São Paulo. Citado na pág. 16, 18

da Silva Filho *et al.* **(2008)** João Inácio da Silva Filho, Jair Minoro Abe e Germano Lambert-Torres. *Inteligência Artificial com as redes de análise paraconsistentes: teoria e aplicações*. LTC, Rio de Janeiro. Citado na pág. 16, 18

Deneubourg *et al.* **(1990)** Deneubourg, S. Aron, S. Goss e J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168. ISSN 0892-7553. doi: 10.1007/BF01417909. URL http://dx.doi.org/10.1007/BF01417909. Citado na pág. 9

Dorigo (1992) M. Dorigo. *Optimization, Learning and Natural Algorithms (in Italian)*. Tese de Doutorado, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy. Citado na pág. 10, 11, 13

Dorigo e Gambardella (1997) M. Dorigo e L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66. doi: http://dx.doi.org/10.1109/4235.585892. Citado na pág. 13, 41

Dorigo e Socha (2006) M. Dorigo e K. Socha. An introduction to ant colony optimization. Relatório técnico, IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Université Libre de Bruxelles, Belgium. Citado na pág. 11

Dorigo e Stützle (2002) M. Dorigo e T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications and advances. Em Fred Glover e Gary Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 9, páginas 251–285. Kluwer Academic Publishers, Boston, MA. doi: http://dx.doi.org/10.1007/0-306-48056-5_9. Citado na pág. 8

Dorigo *et al.* **(1996)** M. Dorigo, V. Maniezzo e A. Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41. doi: http://dx.doi.org/10.1109/3477.484436. Citado na pág. 11

Dorigo *et al.* **(2006a)** M. Dorigo, M. Birattari e T. Stützle. Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1(4):28–39. Citado na pág. 8

Dorigo *et al.* **(2006b)** M. Dorigo, M. Birattari e T. Stützle. Ant colony optimization. Artificial ants as a computational intelligence technique. Relatório Técnico TR/IRIDIA/2006-023, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium. Citado na pág. 13

Dréo e Siarry (2002) Johann Dréo e Patrick Siarry. A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. Em *Proceedings of the Third International Workshop on Ant Algorithms*, ANTS '02, páginas 216–221, London, UK, UK. Springer-Verlag. ISBN 3-540-44146-8. URL http://dl.acm.org/citation.cfm?id=646686.702805. Citado na pág. 67

El-Wahed *et al.* **(2008)** Waile F. Abd El-Wahed, A. A. Mousa e M. A. Elsisy. Solving economic emissions load dispatch problem by using hybrid aco-msm approach. *The Online Journal on Power and Energy Engineering (OJPEE)*, 1(1). Citado na pág. 67

Falaghi *et al.* **(2009)** H. Falaghi, M.-R. Haghifam e C. Singh. Ant colony optimization-based method for placement of sectionalizing switches in distribution networks using a fuzzy multiobjective approach. *Power Delivery, IEEE Transactions on*, 24(1):268 –276. ISSN 0885-8977. doi: 10.1109/TPWRD.2008.2005656. Citado na pág. 52

Goss *et al.* **(1989)** S. Goss, S. Aron, J. L. Deneubourg e J. M. Pateels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(76):579–581. doi: 10.1007/BF00462870. URL http://dx.doi.org/10.1007/BF00462870. Citado na pág. 9

Grassé (1959) P. P. Grassé. La reconstruction du nit el les coordinations interindividuelles chez belliconsitermes natalensis et cubtermes sp. la théorie de la stigmergie: Essai d'interprétation du comportemente des termites constructeus. *Insectes Sociaus*, 6:41–81. Citado na pág. 5

Happ (1977) H.H. Happ. Optimal power dispatch 2014;a comprehensive survey. *Power Apparatus and Systems, IEEE Transactions on*, 96(3):841 – 854. ISSN 0018-9510. doi: 10.1109/T-PAS.1977.32397. Citado na pág. 66

Happ *et al.* **(1981)** H.H. Happ, J.F. Aldrich, P.T. Chan, M.E. El-Hawary, C.R. Gagnon, T. Kennedy, E.F. Koncel, J.W. Lamont, H.D. Limmer, S. Riddington, K.J. Slater, W.O. Stadlin e B.F. Wollenberg. Description and bibliography of major economy-security functions part ii-bibliography (1959 - 1972). *Power Apparatus and Systems, IEEE Transactions on*, PAS-100 (1):215 –223. ISSN 0018-9510. doi: 10.1109/TPAS.1981.316791. Citado na pág. 66

Hou *et al.* **(2002)** Yun-He Hou, Yao-Wu Wu, Li-Juan Lu e Xin-Yin Xiong. Generalized ant colony optimization for economic dispatch of power systems. Em *Power System Technology, 2002. Proceedings. PowerCon 2002. International Conference on*, volume 1, páginas 225 – 229 vol.1. doi: 10.1109/ICPST.2002.1053539. Citado na pág. 67

Hughes (1983) Thomas P. Hughes. *Networks of Power: Electrification in Western Society, 1880-1830*. John Hopkins University Press, Baltimore. Citado na pág. 2

IEE (1987) *On the Semantics of Quantitative Logic Programs*, Washington DC. IEEE Symposium on Logic Programming, Computer Society Press. Citado na pág. 15, 18

Jünger *et al.* **(1997)** M. Jünger, G. Reinelt e G. Rinaldi. *The Traveling Salesman Problem: a Bibliography*. Annotated Bibliography in Combinatorial Optimization, Willey. Citado na pág. 6

Kleene (1952) S.C. Kleene. *Introduction to Metamathematics*. Bibliotheca Mathematica. North-Holland. Citado na pág. 15

Kumar *et al.* **(2008)** Y. Kumar, B. Das e J. Sharma. Multiobjective, multiconstraint service restoration of electric power distribution system with priority customers. *Power Delivery, IEEE Transactions on*, 23(1):261 –270. ISSN 0885-8977. doi: 10.1109/TPWRD.2007.905412. Citado na pág. 51

Lambert-Torres *et al.* **(2009)** G. Lambert-Torres, H.G. Martins, M.P. Coutinho, C.P. Salomon, F.M. Matsunaga e R.A. Carminati. Comparison between pso and ga in system restoration solution. Em *Intelligent System Applications to Power Systems, 2009. ISAP '09. 15th International Conference on*, páginas 1 –6. doi: 10.1109/ISAP.2009.5352885. Citado na pág. 58

Leguizamón e Coello (2010) Guillermo Leguizamón e Carlos A. Coello Coello. An alternative aco<inf>r</inf> algorithm for continuous optimization problems. Em *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, páginas 48–59, Berlin, Heidelberg. Springer-Verlag. ISBN 3-642-15460-3. URL http://dl.acm.org/citation.cfm?id=1884958.1884964. Citado na pág. 67

Liu (2006) X. Liu, Y. e Gu. Reconfiguration of network skeleton based on discrete particle-swarm optimization for black-start restoration. Em *EEE Power Engineering Society General Meeting*, Montreal, Canada. IEEE Power Engineering Society General Meeting. Citado na pág. 47

Lu *et al.* **(2009)** Zhigang Lu, Ying Wen e Lijun Yang. An improved aco algorithm for service restoration in power distribution systems. Em *Power and Energy Engineering Conference, 2009. APPEEC 2009. Asia-Pacific*, páginas 1 –4. doi: 10.1109/APPEEC.2009.4918137. Citado na pág. 52

Martins (2003) H. G. Martins. *The Four-Valued Annotated Paraconsistent Logic - 4vAPL Applied in a Case Based Reasoning System for Restoration of Electrical Substations*. Tese de Doutorado, UNIFEI - Universidade Federal de Itajubá, Brazil. Citado na pág. 23, 24, 26

Martins *et al.* **(2007)** H. G. Martins, G. Lambert-Torres e L. F. Pontin. *Annotated Paraconsistent Logic*. Ed. Communicar. Citado na pág. 23

Mathur *et al.* **(2000)** Mohit Mathur, Sachin B. Karale, Sidhartha Priye, V. K. Jayaraman e B. D. Kulkarni. Ant colony approach to continuous function optimization. *Industrial & Engineering Chemistry Research*, 39(10):3814–3822. doi: 10.1021/ie990700g. URL http://pubs.acs.org/doi/abs/10.1021/ie990700g. Citado na pág. 67

Musirin *et al.* **(2009)** Ismail Musirin, Nur Hasima Faezaa Ismail, Mohd. Rozely Kalil, Muhammad Khayat Idris, titik Khawa Abdul Rhman e Mohd Rafi Adzman. Ant colony optimization (aco) technique in economic power dispatch problem. Em Ping-Kong Alexander Wai, Xu Huang e Sio-Iong Ao, editors, *Trends in Communication Technologies and Engineering Science*, volume 3 of *Lecture Notes in Electrical Engineering*, páginas 191–203. Springer Netherlands. Citado na pág. 65

Pearl (1993) J. Pearl. Belief networks revisited. *Artificial Intelligence*, 59:49–56. Citado na pág. 15

Perez-Guerrero *et al.* (2008) R. Perez-Guerrero, G.T. Heydt, N.J. Jack, B.K. Keel e A.R. Castelhano. Optimal restoration of distribution systems using dynamic programming. *Power Delivery, IEEE Transactions on*, 23(3):1589 –1596. ISSN 0885-8977. doi: 10.1109/TPWRD.2007.916112. Citado na pág. 51

Ramirez-Rosado e Bernal-Agustin (1998) I. J. Ramirez-Rosado e J. L. Bernal-Agustin. Genetic algorithms applied to the design of large power distribution system. *IEEE Transactions on Power System*. Citado na pág. 56

Reinelt (1991) G. Reinelt. Tsplib: A traveling salesman problem library, 1991. URL http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. Citado na pág. 8, 41, 44

Rossi (2000) R. Rossi. *Systemic Hierarchic Classifier for Electrical Nets High-Voltage*. Tese de Doutorado, UNIFEI - Universidade Federal de Itajubá, Brazil. Citado na pág. 47

Russell e Norvig (2004) S. Russell e P. Norvig. *Inteligência Artificial*. Elsevier, Rio de Janeiro, 2 ed. Citado na pág. 16

SBC (1996) *Representação do Conhecimento Incerto*, Pernamburco, Brasil. SBC - Sociedade Brasileira de Computacao. Citado na pág. 15

Silva *et al.* (2009) L. E. Silva, H. G. Martins, M. P. Coutinho, G. Lambert-Torres e L. E. B. Silva. The convergence control to the aco metaheuristic using annotated paraconsistent logic. Em *ISICA2009*, Lectures Notes on Computer Science, páginas 382–391. Citado na pág. 33

Silva *et al.* (2010) L. E. Silva, G. Lambert-Torres, H. G. Martins, M. P. Coutinho, L. E. B. Silva e J. C. Neto. An application of aco in system reconfiguration. Em *IEEE PES Transmission and Distribution*. Citado na pág. 55

Sinha *et al.* (2003) N. Sinha, R. Chakrabarti e P.K. Chattopadhyay. Evolutionary programming techniques for economic load dispatch. *Evolutionary Computation, IEEE Transactions on*, 7 (1):83 – 94. ISSN 1089-778X. doi: 10.1109/TEVC.2002.806788. Citado na pág. 66, 72

Socha (2010) Krzysztof Socha. *Ant Colony Optimization for Continuous and Mixed-Variable Domains*. Tese de Doutorado, Université Libre de Bruxelles. Citado na pág. 67, 68, 69

Socha e Dorigo (2008) Krzysztof Socha e Marco Dorigo. Ant colony optimization for continous domais. *European Journal of Operational Research*. Citado na pág. 67, 68, 70

Stützle e Hoos (2000) Thomas Stützle e Holger Hoos. Max min - ant system. *Future Generation Computer Systems*, 16(8):889–914. ISSN 0167-739X. doi: DOI: 10.1016/S0167-739X(00)00043-1. URL http://www.sciencedirect.com/science/article/B6V06-40CS1PY-4/2/611ad48ab1e55d1f091f590dc54ea94d. Citado na pág. 5, 13, 41, 43

Tian *et al.* (2009) Ye Tian, Tao Lin, Man Zhang e Xialing Xu. A new strategy of distribution system service restoration using distributed generation. Em *Sustainable Power Generation and Supply, 2009. SUPERGEN '09. International Conference on*, páginas 1 –5. doi: 10.1109/SUPERGEN.2009.5348228. Citado na pág. 52

Toune *et al.* **(2002)** S. Toune, H. Fudo, T. Genji, Y. Fukuyama e Y. Nakanishi. Comparative study of modern heuristic algorithms to service restoration in distribution systems. *Power Delivery, IEEE Transactions on*, 17(1):173 –181. ISSN 0885-8977. doi: 10.1109/61.974205. Citado na pág. 51

Walters e Sheble (1993) D.C. Walters e G.B. Sheble. Genetic algorithm solution of economic dispatch with valve point loading. *Power Systems, IEEE Transactions on*, 8(3):1325 –1332. ISSN 0885-8950. doi: 10.1109/59.260861. Citado na pág. 72

Watanabe (2005) I. Watanabe. An aco algorithm for service restoration in power distribution systems. Em *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, páginas 2864 – 2871 Vol. 3. doi: 10.1109/CEC.2005.1555054. Citado na pág. 48, 52

Watanabe *et al.* **(2006)** I. Watanabe, I. Kurihara e Y. Nakachi. A hybrid genetic algorithm for service restoration problems in power distribution systems. Em *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, páginas 3250 –3257. doi: 10.1109/CEC.2006.1688722. Citado na pág. 50

Werbos (2011) P.J. Werbos. Computational intelligence for the smart grid-history, challenges, and opportunities. *Computational Intelligence Magazine, IEEE*, 6(3):14 –21. ISSN 1556-603X. doi: 10.1109/MCI.2011.941587. Citado na pág. 2

Xianchao e Taylor (2010) Huang Xianchao e G.A. Taylor. Service restoration of distribution systems based on nsga-ii. Em *Universities Power Engineering Conference (UPEC), 2010 45th International*, páginas 1 –6. Citado na pág. 51

Xianchao *et al.* **(2010)** Huang Xianchao, Zhang Lizi e G.A. Taylor. Service restoration of distribution system with distributed generation. Em *Power System Technology (POWERCON), 2010 International Conference on*, páginas 1 –5. doi: 10.1109/POWERCON.2010.5666388. Citado na pág. 51

Índice Remissivo