Universidade Estadual de Campinas

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Departamento de Telemática

Aplicações de Meta-heurística Genética e Fuzzy no Sistema de Colônia de Formigas para o Problema do Caixeiro Viajante

Márcia Braga de Carvalho

Orientador: Prof. Dr. Akebo Yamakami

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para obtenção do título de **Mestre em Engenharia Elétrica**.

Banca Examinadora:

Prof. Dr. Anésio dos Santos Júnior	FEEC / UNICAMP
Prof. Dr. Takaaki Ohishi	FEEC / UNICAMP
Profa. Dra. Tatiane Regina Bonfim	FAC / Campinas

Campinas, SP

27 de Julho de 2007

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

C253a

Carvalho, Márcia Braga de
Aplicações de meta-heurística genética e fuzzy no sistema de colônia de formigas para problema do caixeiro viajante / Márcia Braga de carvalho. --Campinas, SP: [s.n.], 2007.

Orientador: Akebo Yamakami
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Algoritmo de formiga. 2. Algoritmos genéticos. 3. Conjuntos difusos. 4. Problema do caixeiro viajante. I. Yamakami, Akebo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Aplications of Genetic and Fuzzy Metaheuristic in the Ant Colony System for the Traveling Salesman Problem.
Palavras-chave em Inglês: Ant algorithms, Genetic algorithm, Fuzzy set theory, Traveling salesman problem
Área de concentração: Automação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Tatiane Regina Bonfim, Anésio dos Santos Júnior e Takaaki Ohishi
Data da defesa: 27/07/2007
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Márcia Braga de Carvalho

Data da Defesa: 27 de julho de 2007

Título da Tese: "Aplicação de Meta-heurística Genética e Fuzzy no Sistema de Colônia de Formigas para o Problema do Caixeiro Viajante"

Prof. Dr. Akebo Yamakami (Presidente): _____

Profa. Dra. Tatiane Regina Bonfim: _____

Prof. Dr. Anésio dos Santos Júnior: _____

Prof. Dr. Takaaki Ohishi: _____

Resumo

Dentre as várias técnicas heurísticas e exatas existentes para a resolução de problemas combinatórios, os algoritmos populacionais de otimização por colônia de formigas e genéticos têm se destacado devido à sua boa performance. Em especial os algoritmos de colônia de formigas são considerados atualmente como uma das técnicas mais bem sucedidas para a resolução de vários problemas combinatórios, dentre eles o problema do caixeiro viajante. Neste trabalho é apresentado um algoritmo híbrido que trabalha com as meta-heurísticas de sistema de colônia de formigas e genético conjuntamente aplicados no problema do caixeiro viajante simétrico. Além disso, apresentamos uma proposta para o algoritmo de formigas quando temos incertezas associadas aos parâmetros do problema. Os resultados obtidos com as metodologias propostas apresentam resultados satisfatórios para todas as instâncias utilizadas.

Palavras-chave: Otimização por Colônia de Formigas, Algoritmo Genético, Teoria dos Conjuntos Fuzzy, Problema do Caixeiro Viajante.

Abstract

Amongst the several existing heuristical and accurate techniques for the resolution of combinatorial problems, the population algorithms ant colony optimization and genetic have been detached due to their good performance. In special the ant colony algorithms are considered currently as one of the techniques most succeeded for the resolution of some combinatorial problems, amongst them the travelling salesman problem. In this work is presented a hybrid algorithm which works with the ant colony system and genetic metaheuristics jointly applied in the symmetric travelling salesman problem. Moreover, we presented a proposal for the ant algorithm when we have uncertainties associated to problem parameters. The results gotten with the methodology proposals present resulted satisfactory for all the used instances.

Keywords: Ant Colony Optimization, Genetic Algorithm, Fuzzy Set Theory, Traveling Salesman Problem.

Dedicatória

A minha avó Margarida Neres Braga (in memorian) e a minha tia Joelma Souza Braga (in memorian), que embora não estejam mais presentes, continuam dentro do meu coração.

Ao meu noivo Maurício, pelo amor, carinho e apoio nos momentos difíceis ao longo desta caminhada.

Agradecimentos

Agradeço a Deus em primeiro lugar, por tudo o que tem me proporcionado.

Ao meu orientador, o professor Akebo Yamakami, pela orientação, oportunidade e motivação nos momentos de dificuldade.

À minha amada mãe, Jussara, um exemplo vivo de uma verdadeira lutadora a quem eu devo tudo. A sua criação e educação me fizeram chegar até aqui.

Ao meu noivo Maurício, meu grande amor e companheiro. Agradeço por tudo que fez por mim nestes anos. Sem o seu carinho e compreensão teria sido muito mais difícil. Sua presença em minha vida é mais um presente de Deus.

À toda minha família, por sempre me apoiarem e torcerem muito por mim.

À todos os amigos da FEEC, especialmente as colegas que participaram diretamente comigo desta jornada: Ingrid, Polyane e Priscila.

Às minhas amigas da Bahia que sempre torceram por mim: Gleice, Ariana, Aline, Juliana e a querida Jucy.

Aos professores da banca: Anésio dos Santos Júnior, Takaaki Ohishi e Tatiane Regina Bonfim, por terem lido o meu trabalho e pelas valiosas correções e sugestões.

À todos os colegas que prestigiaram minha defesa.

À todos os professores e funcionários da Faculdade de Engenharia Elétrica - UNICAMP.

À CAPES, pelo apoio financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACO	-	*Optimization Colony System* - Otimização por Colônia de Formigas.
TSP	-	*Problem Traveling Salesman* - Problema do Caixeiro Viajante.
ACS	-	*Ant Colony System* - Sistema de Colônia de Formigas.
AS	-	*Ant System* - Sistema de Formiga.
AG	-	*Genetic Algorithm* - Algoritmo Genético.
MMAS	-	*Ant System Max-Min* - Sistema de Formigas Max-Min.
HAS	-	*Ant System Hibrid* - Sistema de Formiga Híbrido.

Capítulo 1

Introdução

O problema do caixeiro viajante pode ser definido como um dos problemas mais proeminentes da área de otimização combinatória. A relevância desse problema se deve a razões históricas, já que este foi um dos primeiros problemas de otimização para os quais se conjecturou a classificação *hard* [15] e o primeiro descrito no livro *Computers and Intractability* [17]. A prática também contribui para sua popularidade, já que inúmeras situações em ciência e engenharia podem ser formuladas como instâncias deste problema.

Embora este seja um problema simples de descrever, ele é muito difícil de resolver e em muitos casos encontrar a solução ótima global é uma tarefa praticamente impossível. Um dos motivos é que sua solução torna-se mais complexa à medida que se aumenta o número de nós do grafo, quando somente através de heurísticas é possível obter soluções aproximadas [24]. Conseqüentemente, surge a necessidade de contornar essa complexidade e uma das maneiras é explorar metodologias de computação natural.

Dentre os algoritmos inspirados nessa metodologia, podemos citar os Algoritmos Genéticos e os Algoritmos de Otimização por Colônia de Formigas.

A maioria dos novos algoritmos são baseados em processos biológicos ou naturais, como por exemplo, os algoritmos genéticos e os algoritmos baseados em colônia de formigas.

Os algoritmos genéticos são algoritmos populacionais baseados na evolução natural das espécies. Neste contexto, os indivíduos (soluções de um problema) são submetidos a processos de mutação, reprodução, competição e seleção. A reprodução permite a transmissão de informações genéticas dos pais para os filhos, enquanto a mutação modifica parte genética dos indivíduos com um único objetivo de aumentar a diversidade da população. Esse mecanismo é repetido por várias gerações, fazendo com que a cada geração somente indivíduos mais aptos sejam selecionados para fazerem parte da próxima geração.

O algoritmo de colônia de formigas foi proposto no início da década de 90, e baseou-se nas atividades cotidianas das colônias de formigas reais. Foi verificado que as formigas, à medida que se movimentam, liberam um composto químico conhecido como feromônio. Quanto maior essas quantidades de feromônio, mais propensas as formigas estarão de seguir esta trilha. Desta forma, o método de otimização por colônia de formigas mostrou ser bastante eficiente para a solução de diversos problemas combinatórios, destacando-se entre eles o problema do caixeiro viajante.

O problema do caixeiro viajante aparece em uma série de aplicações práticas como por exemplo, roteamento de veículos [2], problemas de perfuração [26], entre outros.

Em situações reais é comum encontrarmos incertezas nas informações associadas pois, por exemplo, se falarmos sobre a temperatura de um copo, para uma pessoa a temperatura pode ser extremamente agradável e para outra, muito quente ou fria; isso só vai depender do ponto de vista do observador. Desta forma, as tomadas de decisões baseiam-se no conhecimento individual, sem ter certeza sobre as informações [27].

Entretanto, essas incertezas podem ser modeladas e tratadas de uma forma mais robusta utilizando a teoria dos conjuntos nebulosos [33], [14] e [27].

O marco inicial da teoria *fuzzy* foi o artigo *Fuzzy Sets*, publicado em 1965 pelo matemático Lotfi Asker Zadeh [33], com a principal intenção de dar tratamento matemático a certos termos lingüísticos subjetivos. Este foi o primeiro passo no sentido de se programar e armazenar conceitos vagos em computadores, tornando possível a produção de cálculos com informações imprecisas, a exemplo do que faz o ser humano.

O tratamento de incertezas em problemas de grafos *fuzzy* ainda está na sua fase inicial, tanto na parte teórica quanto na prática. Isso se deve às diferentes abordagens que um problema de grafos *fuzzy* pode ter, pois os níveis de incertezas podem estar associadas tanto na estrutura (nós e/ ou arestas) quanto nos parâmetros (custo, capacidade, demanda) [32].

Motivação

Encontrar o menor caminho para o problema do caixeiro viajante envolve uma busca em um espaço que cresce de forma fatorial conforme se aumenta o número de nós.

O número de rotas possíveis em um grafo completo do problema do caixeiro viajante é da ordem $(n-1)!$ [18]. Assim, por exemplo, em um grafo completo com 20 nós é

possível obter 1,21 x 10^{17} rotas possíveis, tornando-se numa busca exaustiva inviável para encontrar o caminho com menor custo. Supondo que temos um computador bastante veloz, capaz de fazer 1 bilhão de adições por segundos, desta forma ele seria capaz de calcular 53 milhões de rotas por segundo. Contudo essa imensa velocidade é pouco diante das rotas que ele tem que analisar e isso pode levar até cerca de 73 anos. Cabe considerar que um grafo com 20 nós é bastante pequeno se comparado com muitos problemas reais existentes.

Diante disto, é proposta uma meta-heurística para encontrar soluções aproximadas satisfatórias, tentando contornar essa complexidade.

Dentre as meta-heurísticas existentes escolhemos a de otimização por colônia de formigas e o algoritmo genético, mais precisamente um novo algoritmo híbrido denominado ACS+AG-TSP (Ant Colony System + Genetic - Travelling Salesman Problem) que trabalha com os algoritmos de Sistema de Colônia de Formigas e Genético conjuntamente para resolver o problema do caixeiro viajante.

Quando se trabalha com grafos nos quais são associados custos *fuzzy* as suas arestas, sua complexidade aumenta. Desta forma propomos um algoritmo ACS-*Fuzzy* e verificarmos que é possível também resolver o problema do caixeiro viajante com o algoritmo de sistema de colônia de formigas, quando se tem incertezas nos parâmetros.

Objetivos

Neste trabalho é feito um estudo do algoritmo de colônia de formigas e do algoritmo genético e, a partir de testes com esses modelos, é sugerido um novo algoritmo. O algoritmo híbrido proposto nesta dissertação trabalha com as heurísticas de colônia de formigas e genético, cujo principal objetivo é encontrar a melhor solução para o problema do caixeiro viajante simétrico, visando contornar a complexidade do problema. Também é proposto um algoritmo ACS com parâmetros *fuzzy* para o problema do caixeiro viajante, baseado na Teoria da Possibilidade. Esse algoritmo tem por objetivo mostrar que é possível tratar o problema do caixeiro viajante com parâmetros incertos.

Organização do Trabalho

O trabalho está organizado da seguinte maneira: no Capítulo 2 é conceituado o problema do caixeiro viajante, são apresentadas sua formulação matemática, complexidade e alguns métodos de resolução. O Capítulo 3 faz uma apresentação sobre o algoritmo de otimização por colônia de formigas, sua idéia principal e seu funcionamento. O Capítulo 4 revê os princípios de algoritmos genéticos. A seguir, no Capítulo 5 é proposto um algoritmo híbrido que visa encontrar a melhor solução para o problema do caixeiro viajante. No Capítulo 6 são apresentados os experimentos e resultados obtidos para cada um dos métodos propostos. No Capítulo 7 é apresentada uma extensão do algoritmo de formigas para problemas com parâmetros incertos. Finalmente, no Capítulo 8 são feitas as considerações finais deste trabalho e algumas propostas para trabalhos futuros.

No final da dissertação estão incluídos anexos com os grafos utilizados neste trabalho.

Capítulo 2

O Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (TSP) é um problema clássico da Otimização Combinatória [22] e tem sido estudado por diversos pesquisadores de diferentes áreas. Isso se deve ao fato de possuir diversas aplicações práticas. O TSP pertence a uma classe de problemas não determinísticos de tempo não polinomial, classificado como NP- difícil [17]. Por isso, a resolução do TSP, utilizando métodos heurísticos, ganhou maior importância, principalmente quando é aplicado a problemas de grande porte.

Este capítulo tem por objetivo conceituar o problema do TSP, apresentando sua formulação matemática, complexidade e métodos de resolução utilizando o algoritmo de colônia de formigas e o algoritmo Genético, visto que neste trabalho o objeto de análise é um algoritmo híbrido ACS + AG aplicado ao TSP.

2.1 Descrição do problema do caixeiro viajante

Seja um conjunto de pontos representando n cidades. O problema do Caixeiro Viajante consiste na determinação de uma rota que inicia em uma cidade, passa por cada cidade do conjunto apenas uma vez, e retorna à cidade inicial da rota perfazendo uma distância total mínima. Esta rota é denominada ciclo Hamiltoniano de custo mínimo.

A representação deste problema pode ser feita através de um grafo completo $G = (V, A)$, onde V é o conjunto de vértices representando as cidades e A o conjunto de arcos ou arestas que conectam cada par de cidades $i, j \in V$. A cada aresta é atribuído um valor c_{ij}, que é a distância da cidade i para a cidade j. Para o TSP simétrico, essa distância c_{ij} é independente da direção dos arcos, ou seja $c_{ij} = c_{ji}$, e para o TSP assimétrico $c_{ij} \neq c_{ji}$.

2.2 Formulação Matemática

A formulação matemática para o problema é normalmente referenciada na literatura usando minimização na função objetivo, embora esta formulação seja a mesma tanto para *Min* (minimização), quanto para *Max* (maximização).

Função objetivo: $$Min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}.x_{ij}$$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, ..., n \tag{2.1}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, ..., n \tag{2.2}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \qquad \forall S \subset V, S \neq \phi \tag{2.3}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, ..., n \tag{2.4}$$

onde c_{ij} e x_{ij} são, respectivamente, os custos e as variáveis de decisão associados à designação do elemento i à posição j. A variável $x_{ij} = 1$ indica que a cidade j é visitada logo após a cidade i, caso contrário $x_{ij} = 0$.

A função objetivo representa a minimização do somatório das distâncias entre as cidades da rota, a variável n representa o número total de cidades no problema, S é um subconjunto do conjunto $V = \{1, 2, ..., n\}$, e o símbolo $|\ |$ denota a cardinalidade do conjunto. As restrições (2.1) e (2.2) garantem que cada cidade $i \in n$ será designada para exatamente uma cidade j. A restrição (2.3) garante a não existência de infactibilidade e a restrição (2.4) define x como uma variável binária.

2.3 Complexidade

Problemas como o do Caixeiro Viajante são extremamente complexos quando da tentativa de obtenção de uma solução ótima.

Os problemas de otimização combinatória podem ser classificados segundo sua complexidade em, P, NP, NP-difícil e NP-completo, sendo que o problema do caixeiro

viajante pertence a classe NP-difícil. Os problemas NP-difícil são caracterizados por haver redução polinomial a partir de todo o problema pertencente à classe NP.

A classe NP (*NonDeterministic Polinomial Time*) são aqueles problemas que podem ser resolvidos por algoritmos não-determinísticos em tempo polinomial.

Problemas P (*Polinomial time*) é formada pelo conjunto de todos os problemas que podem ser resolvidos por um algoritmo determinístico em tempo polinomial.

2.4 Métodos para Resolução do TSP

Embora o problema do caixeiro viajante possa ser descrito de maneira simples, ele pertence à classe dos problemas NP-difíceis, tornando-o intratável para obtenção da solução por métodos exatos para problemas de grande porte.

Atualmente existe um número muito grande de problemas que podem ser modelados como um problema do caixeiro viajante. Com isso, abordagens exatas e de aproximações como as heurísticas e metaheurísticas, têm sido buscadas para a resolução de instâncias de grande porte desse problema, tanto para o caso do TSP simétrico quanto para o assimétrico.

Técnicas que são freqüentemente utilizadas para resolver métodos exatos do TSP são *branch-and-bound* e *branch-and-cut*.

Os métodos heurísticos embora não garantam encontrar soluções necessariamente ótimas, eles permitem encontrar boa aproximação para problemas reais e com maior rapidez que os exatos. Por isso o problema do TSP tem sido utilizado nos últimos anos como base de comparação para melhorias em diversas técnicas de otimização, tais como Algoritmos Genéticos, Colônia de formigas, Busca Tabu, Busca Local, Redes Neurais, dentre outras.

A resolução do problema do caixeiro viajante, utilizando-se a heurística de Colônia de Formigas, pode ser encontrada em [9]. A idéia desta heurística foi inspirada na observação das colônias de formigas reais e baseia-se na capacidade das formigas encontrarem um caminho mínimo entre a colônia e a fonte de alimento. Neste artigo, são mostradas comparações entre a heurística de otimização de colônia de formigas e os métodos de otimização global, como: Simulated Annealing, Rede Elástica e Algoritmos Genéticos, sendo que os resultados da heurística de Colônia de Formigas foram melhores que os três métodos para as matrizes de custos consideradas. Além disso, a heurística proposta é aplicada com a técnica de melhoria *3-opt*, reduzindo os erros médios que se encontravam

nos grafos considerados.

Outros trabalhos semelhantes, que utilizam a heurística podem ser encontrados em [31], [12], [10] entre outros.

O trabalho de Pilat e White (2002) mostra a aplicação de um método híbrido de Algoritmos Genéticos aplicado ao método de Colônias de Formigas para o problema do Caixeiro Viajante simétrico. Os Algoritmos Genéticos são utilizados para determinar quais formigas artificiais servem para melhorar as soluções para o problema.

No trabalho de Merz e Freisleber (1997) é apresentada a aplicação de Algoritmos Genéticos para o problema do Caixeiro Viajante simétrico, com aplicação da técnica de Busca local *2-opt* para a rota encontrada, com o objetivo de encontrar melhores rotas para as próximas gerações. Os resultados encontrados no banco de dados do *TSPLIB* (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/) mostram que esta técnica é capaz de obter soluções na maioria dos casos citados.

Affenzeller e Wagner (2003) mostra uma técnica para determinar boas soluções para o TSP com o uso de Algoritmos Genéticos através das comparações entre o *fitness* de novas soluções com o *fitness* da população e dos *fitness* dos novos indivíduos e de seus pais. Os resultados para alguns problemas do *TSPLIB* são melhores do que os resultados encontrados com a aplicação de Algoritmos Genéticos da forma tradicional.

O trabalho de Sari, Sherali e Bhootra (2005) mostra uma nova formulação para o problema do Caixeiro Viajante assimétrico, criando restrições para a criação de sub-rotas. Os resultados para alguns problemas do *TSPLIB* mostram que esta nova formulação permite a obtenção de soluções ótimas em todos os casos testados.

Os Algoritmos Genéticos representam, atualmente, uma poderosa ferramenta para busca de soluções de problemas com alto nível de complexidade. Dentre alguns trabalhos que utilizam algoritmos genéticos para o TSP tem-se, [20] entre outros.

Capítulo 3

Otimização por Colônia de Formigas

Neste capítulo serão apresentadas algumas noções importantes para a compreensão da meta-heurística de Otimização por Colônia de Formigas (*Ant Colony Optimization - ACO*), seu funcionamento e o algoritmo estudado, o *Ant Colony System*, de modo a facilitar a compreensão deste trabalho. Para estudos mais aprofundados do assunto, consultar [3] e [13]. Este capítulo está organizado da seguinte maneira:

- Na seção 3.1 é feita uma breve introdução sobre a meta-heurística ACO e são apresentadas algumas diferenças entre o algoritmo inicial, o Sistema de Formigas (*Ant Colony - AS*) e o Sistema de Colônia de Formigas (*Ant Colony System - ACS*);
- Na seção 3.2 são apresentadas algumas características que conferem as semelhanças entre as formigas reais e as artificiais, bem como algumas idéias que foram transferidas das colônias de formigas reais para as formigas artificiais;
- Na seção 3.3 é descrito o processo de construção do algoritmo ACS e é apresentada a descrição detalhada de seu funcionamento;
- Na seção 3.4 são apresentadas algumas contribuições e aplicações da meta-heurística ACO;

3.1 Considerações Iniciais

Ant Colony Optimization é uma meta-heurística cuja fonte de inspiração é o comportamento de colônias de formigas, baseado na observância de que as formigas são capazes

de encontrar o menor caminho entre o ninho e a fonte de alimento. Esses comportamentos são explorados em colônias artificiais de formigas para obtenção de soluções aproximadas aos problemas de Otimização Combinatória, como por exemplo o problema do caixeiro viajante. O primeiro algoritmo de ACO, chamado Sistema de Formigas - AS, proposto inicialmente por Dorigo em meados dos anos 90, foi aplicado ao problema do Caixeiro Viajante. Posteriormente, Dorigo e Gambardella (1997), propuseram o algoritmo denominado *Ant Colony Systems - ACS* para melhorar o desempenho do AS. Segundo eles o ACS difere em dois aspectos importantes do AS.

1. Em ACS o feromônio é adicionado somente aos arcos que pertencem a melhor solução global. Já em AS esse feromônio é adicionado a todos os arcos da rede.

2. Em ACS, que cada vez que uma formiga usa o arco (i, j) para mover-se da cidade i, para a cidade j, remove-se uma quantidade do feromônio desse arco, segundo a equação 3.4, já em AS esse processo de evaporação do feromônio não ocorre.

3.2 Inteligência por Colônia: Formigas reais e artificiais

As idéias utilizadas na meta-heurística ACO provém da observação das colônias de formigas reais. Existem características que conferem semelhanças entre as formigas reais e as artificiais. Algumas dessas semelhanças serão abordadas ao longo deste texto.

A meta-heurística ACO prevê uma colônia de agentes independentes entre si, em sinergia na busca de boas soluções. Tal cooperação entre os agentes ocorrem pela leitura e escrita em um ambiente compartilhado, em busca do menor caminho entre uma origem (ninho) e um destino (alimento).

As formigas artificiais são agentes que se movem de cidade para cidade em um grafo do TSP. Podemos citar três idéias do comportamento natural das formigas que foram transferidas para a colônia de formigas artificiais.

- A preferência para trajetos com nível elevado de feromônio.

- A taxa de crescimento mais elevado da quantidade de feromônio em alguns trajetos mais curtos.

- A deposição do feromônio medeia uma comunicação entre as formigas.

A estas formigas artificiais foram dadas algumas potencialidades que não tem as contrapartes naturais, mas que foram observadas para serem bem sucedidas nas aplicações do TSP: as formigas artificiais são dotadas de uma memória denominada *lista tabu* para memorizar as cidades já visitadas.

Essa memória é esvaziada no começo de cada nova viagem, e é atualizada adicionando a cidade inicial corrente ao conjunto das cidades já visitadas. Essa memória é muito importante no TSP, pois evita que uma cidade seja visitada duas vezes, define o conjunto de cidades que uma formiga, localizada na cidade i, ainda tem para visitar e permite que a formiga calcule o comprimento do *tour* e retroceda o caminho para depositar o feromônio.

A seguir serão mostrados dois experimentos com formigas reais realizados por [8], que serviu de inspiração à criação do método de Otimização de Colônia de Formigas. Esta experiência consistiu na submissão de uma colônia de formigas *Iridomyrmex humilis* a uma fonte de alimento através de dois caminhos distintos. No primeiro caso, o caminho entre o ninho e a fonte de alimento é dividido em duas partes iguais, como pode ser visto na figura 3.1 abaixo.

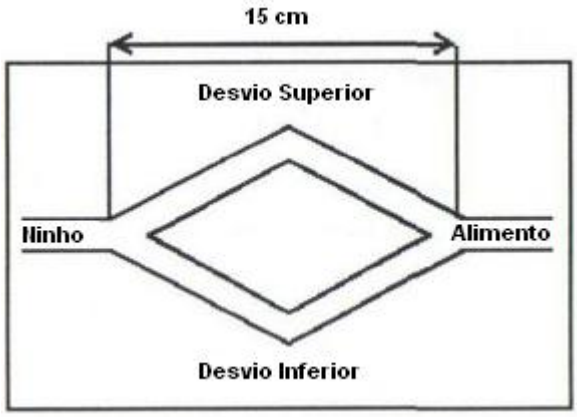

Figura 3.1: Caminho dividido igualmente no 1^o experimento

No inicio as formigas são liberadas e procuram a fonte de alimento através de um caminho aleatório, mas rapidamente converge para uma das pontes. A explicação para tal comportamento está relacionada ao feromônio depositado pelas formigas durante a sua locomoção. Entretanto, algumas formigas ainda fazem um caminho aleatório, para procurar novos caminhos que ainda possam ser descobertos. Na figura 3.2, é apresentado o percentual de formigas seguindo cada um dos caminhos em um espaço de tempo definido.

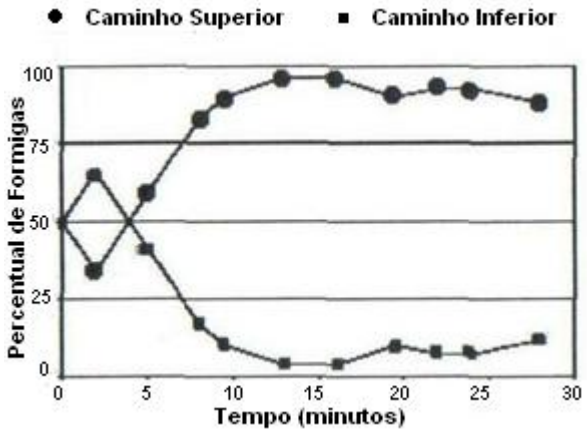

Figura 3.2: Relação entre percentual de formigas no caminho durante o tempo.

Já o segundo experimento tinha como objetivo verificar como as formigas se comportariam a mudanças no ambiente global, visto que isto normalmente ocorre entre o ninho e o formigueiro. Para isto foi desenvolvido um novo caminho entre o ninho e a fonte de alimento. Neste caminho existem duas partes mais longas e duas partes mais curtas, conforme figura 3.3. Inicialmente, quando apresentados os dois caminhos ao mesmo tempo, as formigas só podem percorrer os caminhos mais longos. Após a deposição de feromônio neste pior caminho e consequentemente fixação da trilha, são liberados os caminhos mais curtos para serem percorridos.

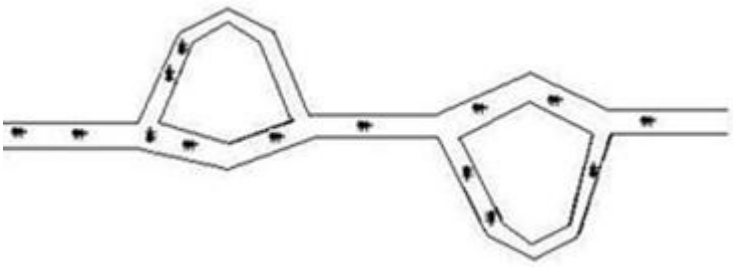

Figura 3.3: Caminho no 2^o experimento com trilhas longas e curtas.

Neste experimento, Deneubourg concluiu que as formigas não trocam de caminho imediatamente quando apresentado o novo(menor) caminho, porém quando este é encontrado, ele tende a ser cada vez mais utilizado e, com o passar do tempo, todas as formigas passam a percorrê-lo.

As formigas são insetos sociais que possuem um sistema complexo de organização e

divisão de tarefas, cuja principal função é garantir a sobrevivência do formigueiro. Elas são capazes de encontrar o menor percurso entre a fonte de alimento e o ninho, sem usar sugestões visuais ou qualquer outro tipo de controle centralizado [9].

Essas funcionalidades são obtidas por um processo de estigmergia e auto-organização em que agentes simples são capazes de emergir comportamentos complexos e direcionado a um objetivo. Por isso elas utilizam uma substância chamada *feromônio*, que é uma estrutura química de comunicação e sinalização.

Ao se deslocarem, as formigas tendem a seguir trilhas com maior nível dessa substância. Ao seguir tais caminhos, deposita-se mais feromônio, o que torna o caminho mais atraente na medida em que mais formigas escolhem a mesma trilha. Individualmente uma formiga pode não produzir boas soluções para o problema em questão, mas através da cooperação são capazes de encontrar boas soluções. Esse comportamento foi apresentado no experimento feito por [9] e pode ser visto na Figura 3.4.

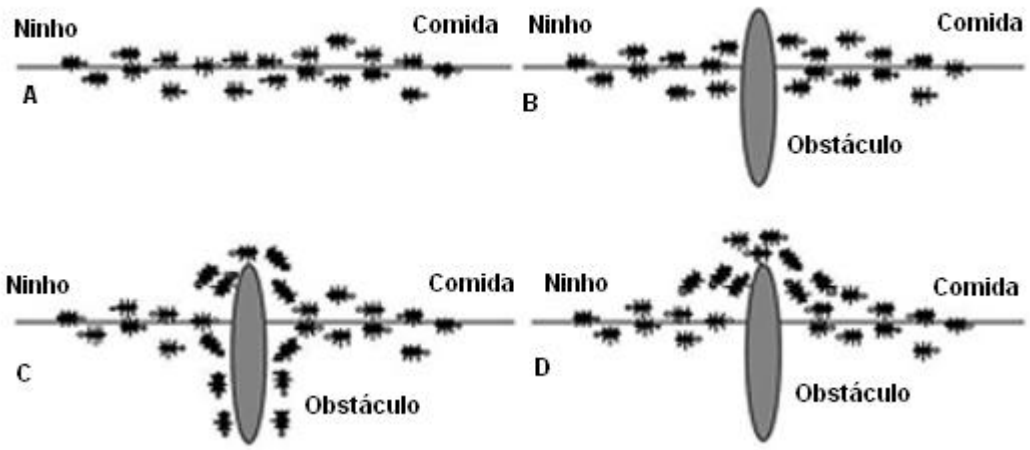

Figura 3.4: Comportamento das formigas após o surgimento de um obstáculo entre o ninho e a fonte de alimento.

Na figura 3.4(A), as formigas reais seguem um trajeto entre o ninho e a fonte de alimento. Na ilustração (B), um obstáculo aparece no caminho: as formigas escolhem de forma aleatória o caminho a direita ou a esquerda com a mesma probabilidade. Na ilustração (C), o feromônio é depositado no menor caminho mais rapidamente. Na ilustração (D), todas as formigas escolhem o caminho com maior nível de feromônio, que conseqüentemente é o caminho mais curto.

3.2.1 Estigmergia

O processo de estigmergia é um processo que explica a coordenação de tarefas na reconstrução de ninhos de cupins (para maiores detalhes consultar [3]). Um exemplo bastante evidente da estigmergia é a busca por alimentos realizada pelas formigas. A interação dos insetos para produzir um sistema auto-organizado pode ocorrer de duas formas distintas:

- Direta: Ocorre através do contato táctil entre os animais por suas antenas ou mandíbulas, do contato visual ou mesmo quimicamente pelos odores de outros insetos, como no caso das formigas, com o feromônio.
- Indireta: Esta forma de comunicação ocorre quando um indivíduo de uma população altera seu meio próximo, fazendo com que o ambiente local onde ele está seja modificado, e que outros indivíduos que estejam no mesmo meio, em um momento posterior, tenham suas decisões afetadas por esta modificação individual.

3.2.2 Auto-organização

A auto-organização é um mecanismo dinâmico no qual uma estrutura global bem definida surge a partir de iterações de componentes de baixo nível, no caso as formigas. Para que esse processo de auto-organização ocorra é necessário que algumas regras especifiquem as formas de interações dos componentes com o meio local onde estão localizados, sem referência a todo o sistema.

Este conjunto de regras podem ser resumidos em quatro grupos básicos (para maiores detalhes consultar [3]:

1. Auto-alimentação positiva(amplificação): A amplificação, que é preferência no caso das formigas, dos melhores caminhos faz com que estes se tornem caminhos preferenciais na busca.
2. Auto-alimentação negativa: Utilizado para contrabalançar o efeito da alimentação positiva, fazendo com que caminhos pouco utilizados sejam esquecidos com o passar do tempo.
3. Aleatoriedade: A auto-organização surge da amplificação das flutuações, que permitem a localização de novas soluções superando minímos locais.

4. Um único individuo pode produzir uma estrutura organizada através de uma trilha estável de feromônio. Entretanto, esta estrutura não será otimizada; para que um sistema auto-organizado encontre boas soluções é necessário que múltiplos indivíduos interajam entre si a partir dos seus resultados e através do ambiente comum.

3.3 Ant Colony System para o Problema do Caixeiro Viajante

Aqui é apresentado o esquema do algoritmo de sistema de colônia de formigas para o problema do caixeiro viajante proposto por [9].

A primeira meta-heurística de otimização por colônia de formigas foi o AS [11]. Posteriormente [9] propuseram o algoritmo ACS para o TSP. O sistema de colônia de formigas é uma forma de resolução de problemas que simulam o comportamento de um grupo de formigas na busca pelo menor caminho.

Em um sistema de colônia de formigas - ACS, duas tarefas são principais:

- A construção de uma representação do problema de uma forma versátil, normalmente uma estrutura na forma de um grafo e que permita uma regra probabilística de transição entre os nós baseada na trilha de feromônios e no valor de cada arco;
- O desenvolvimento de uma heurística para transição de nós que possa avaliar a qualidade dos caminhos tomados.

O problema do TSP busca o caminho que minimize o trajeto entre os diversos nós interligados através de arcos em uma estrutura semelhante a um grafo, como pode ser visto na Figura 3.5.

Para cada arco (i, j) do grafo, é definida uma variável τ_{ij}, conhecida como trilha artificial de feromônio. Inicialmente este τ_{ij} é igual para todos os arcos da rede. Cada formiga **k** constrói uma solução a partir de um dos nós do grafo de forma randômica. Assim ,a variável τ_{ij} é incrementada conforme o passar das formigas e decrementada a cada ciclo. A intensidade da trilha de feromônio definirá a utilidade da trilha para as formigas, isto é, quanto maior a quantidade de feromônio, melhor é esta trilha e as formigas tendem a utilizar esse percurso em detrimento dos outros.

A cada nó, a formiga artificial executa uma função estocástica para calcular a probabilidade de utilização dos arcos. Inicialmente a função probabilística foi desenvolvida

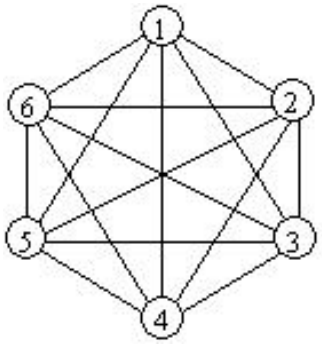

Figura 3.5: Estrutura de nós interligados por arcos utilizado no TSP.

relacionando-se apenas com o processo de estigmergia, ocasionado pelo acumulo de feromônio pelas trilhas. Desta forma, a função continha apenas um termo relacionado à deposição do feromônio, como apresentado na equação (3.1).

$$p_{ij}^{k} = [\tau_{ij}] \tag{3.1}$$

Esta abordagem, apesar de obter alguns resultados para alguns problemas específicos, não obtinha bons resultados para diversas instâncias do TSP e ainda não tinha uma grande confiabilidade dos resultados. Por isso decidiu-se pela implementação de um segundo termo que seria um valor heurístico relacionado à natureza do problema e que permitiria uma maior exploração. Esta nova fórmula é apresentada na equação (3.2).

$$u = \text{argmax}\{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}\}, \text{ se } \quad q \leq q_0; \tag{3.2}$$

onde q é uma variável aleatória uniformemente distribuída entre $[0, 1]$, q_0 é um parâmetro ajustável entre $(0 \leq q_0 \leq 1)$, e os parâmetros α e β controlam a intensidade do feromônio τ_{ij}, e a qualidade da aresta η_{ij}, respectivamente.

A equação (3.2) foi normalizada dividindo-se os termos desta equação pelo somatório de todas as probabilidades possíveis. Assim, obteve-se uma fórmula probabilística onde cada formiga k constrói o seu caminho movendo-se através de uma sequência de locais visinhos, onde os movimentos são selecionados segundo uma distribuição de probabilidades dada pela equação (3.3):

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}} \tag{3.3}$$

onde:

p_{ij}^k : é a probabilidade da formiga k, que se encontra na cidade i, escolher o nó j como próximo nó a ser visitado;
τ_{ij} : quantidade de feromônio existente no arco (i, j). Inicialmente, adota-se um mesmo valor τ_0 para todos os arcos da rede;
η_{ij} : função heurística que representa a atratividade do arco (i, j). No caso do problema do caixeiro viajante, adota-se o inverso $1/d_{ij}$ do valor da distância entre os nós i e j;
J_i^k : conjunto de pontos ainda não visitados pela formiga k, que se encontra atualmente no ponto i;
α : é um parâmetro que pondera a importância relativa da trilha de feromônio, τ_{ij} na decisão de movimentação da formiga.
β : valor heuristicamente escolhido, que pondera a influência relativa da distância η_{ij} entre os nós i e j no processo de decisão.

Podemos observar que se $\alpha = 0$, as formigas seguirão a heurística do vizinho mais próximo para dar o próximo passo, enquanto que se $\beta = 0$, as formigas selecionarão um caminho com maior nível de feromônio e pode ocorrer uma estagnação com o passar das gerações, levando o algoritmo a pontos sub-ótimos. No caso do ACS o valor de α é igual a 1.

A equação (3.3) mostra que a preferência da formiga por um determinado caminho é maior para os caminhos com maior nível de feromônio e com menor distância. A atualização do feromônio é realizada local e globalmente.

A atualização local realizada de acordo com a euqação3.4 é para evitar uma aresta muito forte que está sendo escolhida por todas as formigas, e ela é motivada pela evaporação do feromônio no caminho por onde cada formiga k passar. Esse processo aumenta o poder de exploração das formigas.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0, \qquad \tau_0 = (n.L_{nn})^{-1} \tag{3.4}$$

onde n é o número de nós do grafo, L_{nn} um valor heurístico de menor caminho encontrado por alguma formiga e ρ é um parâmetro que determina a velocidade da evaporação do feromônio.

Já na atualização global é pretendido recompensar as arestas que pertencem a rotas mais curtas. Uma vez que as formigas artificiais terminam seus trajetos, a melhor formiga deposita feromônio em arestas visitadas que pertencem a seu trajeto e as outras arestas pertencem inalteradas. Essa atualização é dada pela equação (3.5). Assim, a cada arco (i, j) da rede, adiciona-se uma quantidade de feromônio proporcional ao tamanho da rota obtida, ou seja, quanto mais curta a rota, maior será a quantidade de feromônio depositada.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \triangle\tau_{ij}, \qquad \rho \in [0, 1] \tag{3.5}$$

onde,

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L^k}, & \text{se } (i, j) \text{ usado} \\ 0, & \text{caso contrário} \end{cases} \tag{3.6}$$

O primeiro termo das equações (3.4) e (3.5) é responsável pela evaporação do feromônio. O parâmetro ρ é utilizado para que os caminhos que são menos freqüentados sejam esquecidos com o passar do tempo. O segundo termo da equação (3.5) é responsável por aumentar a concentração de feromônio apenas nos arcos visitados pela melhor formiga, onde L^k é a distância total percorrida na rota construída pela melhor formiga da iteração. Quanto menor a rota, maior a quantidade de feromônio depositada.

Esse procedimento se repete até que um número máximo de iterações tenha sido alcançado ou caso não se verifique mais melhorias nas soluções encontradas.

3.3.1 Algoritmo ACS para o TSP

Algoritmo 3.3.1: Algoritmo ACS - TSP

Inicialização:

Para cada aresta (i, j) do grafo, estabelece-se um nível inicial de feromônio.

Para cada formiga k, escolhe-se um nó aleatório para iniciar o percurso.

Loop

para t indo de 1 até um número máximo de iterações **faça**,

para k indo de 1 até m **faça**, (m = número de formigas)

Cada formiga k constrói um caminho selecionando a próxima cidade a ser visitada segundo a regra determinística ou probabilística influenciada pelo nível de feromônio τ_{ij} e da métrica heurística de afinidade η_{ij},

$$j = \text{argmax}\{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}\}, \quad \text{se} \quad q \leq q_0;$$

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)] \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}(t)] \cdot [\eta_{il}]^{\beta}}, \quad \text{se} \quad q > q_0;$$

Após cada transição da formiga k, aplique a regra de atualização local, motivada pela evaporação do feromônio,

$$\tau_{ij}(t) \leftarrow (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0$$

fim-do-para

Para cada solução, calcule a distância $L_k(t)$ do caminho descoberto pela formiga k,

Se $L_k(t) < L^*$ **então** $S^* \leftarrow S_k(t)$;

Para cada aresta (i, j), atualize o feromônio $\tau_{ij}(t)$, $\forall (i, j) \in L^k$, de acordo com o processo de deposição e evaporação de feromônio.

$$\tau_{ij}(t) \leftarrow (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t), \text{ onde } \Delta\tau_{ij}(t) = 1/L^k$$

fim-do-para

fim-do-Loop

Retorne a melhor solução S^*.

3.4 Algumas contribuições importantes e suas aplicações

Pesquisas sobre algoritmos de ACO, e sobre algoritmos de formigas, produziram um grande número de aplicações bem sucedidas aos problemas de otimização combinatorial, tais como problemas de ordenação seqüencial [16], roteamento de veículos [5], entre outros. Também foram propostos diversos algoritmos baseado no algoritmo original; na Tabela 3.1 encontram-se alguns deles, para maiores detalhes consultar [13].

Problema	**Nome do Algoritmo**	**Autores**	**Ano**
Caixeiro Viajante	*AS*	Dorigo,et.al	1991
	Ant-Q	Gambardella, et.al.	1995
	ACS	Dorigo e Gambardella	1996
	MMAS	Stützle e Hoss	1997
	AS_{rank}	Bullnheimer	1997
Atribuição Quadrática	*AS-QAP*	Maniezzo,et.al	1994
	HAS-QAP	Gambardella	1997
	MMAS-QAP	Stützle e Hoss	1998
	ANTS-QAP	Maniezzo	1998
Escalonamento	*AS-FSP*	Stützle	1997
Roteamento de Veiculos	*AS-VRP*	Bullnheimer,et.al.	1999
Roteamento de rede orientada à conexão	*ANT-NET*	Di Caro e Dorigo	1997
Ordenação Seqüencial	*HAS-SOP*	Dorigo e Gambardella	2000
Atribuição de freqüência	*ANTS-FAP*	Maniezzo,et.al.	1999
Cobertura	*ACS-SCP*	Hadji,et.al.	2000

Tabela 3.1: Algumas aplicações de algoritmos ACO em problemas de otimização combinatória

O ANTS (*Approximated Non-Deterministic Tree Search*) é uma extensão proposta por [21], do ACO original. Trata-se de uma hibridação entre o ACO e os algoritmos de busca em árvores. As principais diferenças implementadas foram os parâmetros de atratividade e o mecanismo de incremento da trilha de feromônio.

O Max-Min foi proposto por [30]. Neste algoritmo foi implementada uma forma agressiva de incremento de feromônio, na qual apenas o melhor caminho obtido em um ciclo teria sua trilha de feromônio acrescida. O Max-Min utiliza métodos de busca local, utilizando para isso um método chamado de correlação distância-*fitness*.

A primeira alteração do AS para a resolução do problema do caixeiro viajante foi proposto por [9]. O algoritmo chamado ACS mostrou ser eficiente e versátil, obtendo uma solução para o problemas do caixeiro viajante, superior na média às soluções encontradas por redes neurais, algoritmos genéticos e algoritmos de *simulated annealing*.

Finalmente em 1999, Dorigo e Stützle consagraram a família de algoritmos ACO como um conjunto de métodos eficientes para a resolução de vários problemas de otimização combinatória *NP*-hard, tanto estáticos quanto dinâmicos, relacionando diversas implementações, referências e aplicações eficientes.

Capítulo 4

Algoritmo Genético

Neste capítulo são fornecidos alguns conceitos básicos sobre os algoritmos genéticos (seção 4.1), bem como seus operadores, alguns mecanismos de seleção, e sua estrutura básica. A intenção aqui não é de oferecer um texto completo sobre os algoritmos genéticos, mas apenas disponibilizar algumas informações sobre os principais conceitos utilizados nesta dissertação, de modo a facilitar a compreensão do método proposto. Para um estudo mais aprofundado sobre os algoritmos genéticos, consultar [19] e [24].

4.1 Conceitos Básicos

Os algoritmos genéticos foram introduzidos nos anos 60 por John Holland e consolidado em 1975 com a finalidade de formalizar matematicamente e explicar os processos de adaptação em sistemas naturais para, posteriormente, desenvolver sistemas artificiais simulados em computador que retenham os mesmos mecanismos originais encontrados em sistemas naturais.

A terminologia utilizada no desenvolvimento dos algoritmos genéticos foi baseada na teoria da evolução natural e da genética proposta por Darwin em 1859.

Uma das principais características dos AG's é de não encontrarem uma única solução (indivíduo), mas uma população deles, caracterizando-se como método estocástico de busca e seleção. Segundo [24], para percorrer este espaço de busca é necessário manter equilíbrio entre dois pontos conflitantes:

- **Explotação:** consiste no aproveitamento das melhores soluções;
- **Exploração:** consiste na exploração do espaço de busca.

Diversos estudos comprovaram que os algoritmos genéticos mantém um equilíbrio notável nos pontos de contraste apresentados anteriormente e, portanto, conseguem fazer o aproveitamento de melhores soluções e exploração do espaço de busca. Embora sejam identificadas etapas não-determinísticas em seu desenvolvimento, os algoritmos genéticos não são métodos de busca totalmente aleatórios, e sim métodos que relacionam variações aleatórias com seleção polarizada pelos valores de adequação (*fitness*) atribuídos a cada indivíduo.

Os AG's são capazes de desenvolver soluções para problemas do mundo real, tais como problemas de busca, de otimização entre outros problemas complexos de engenharia. Embora eles não garantam eficiência total na obtenção da solução, geralmente garantem uma boa aproximação. Eles utilizam os conceitos da evolução biológica, tais como: genes, cromossomos, cruzamento, mutação e seleção. Desta forma as espécies evoluem aleatoriamente via operadores, estando sujeitas à seleção natural. Assim os indivíduos mais adaptados sobrevivem e se reproduzem, propagando o seu material genético para as próximas gerações.

A seguir, apresentamos uma descrição mais detalhada das partes constituintes de um algoritmo genético.

Codificação: A escolha da representação (codificação) dos cromossomos é uma etapa bastante importante na construção do algoritmo genético, pois deve permitir uma representação da solução completa do problema, sem dificultar a medida da função de *fitness* ou a aplicação dos operadores genéticos. Os algoritmos genéticos clássicos adotam a codificação binária, isto é, cada gene (bit) pode assumir valores 0 ou 1.

A motivação para o uso dessa codificação vem da teoria dos esquemas [19]. Holland argumenta que seria benéfico para o desempenho do algoritmo maximizar o paralelismo implícito inerente ao algoritmo genético. Entretanto esta representação não é a mais adequada a diversos problemas de otimização, tais como o problema do caixeiro viajante, pois tem se mostrado ineficiente. Contudo outros tipos de codificação podem ser utilizadas, como por exemplo, codificação inteira, real e mista.

Cromossomo: É a estrutura nucleoprotéica dentro da célula que armazena o DNA dos seres vivos. Dentro de cada cromossomo, o DNA fica como uma fita enrolada em espiral. Em algoritmos genéticos, um cromossomo haplóide (apenas uma cadeia de DNA representa o cromossomo) geralmente corresponde a uma cadeia de bits que representa

um candidato à solução do problema (vide Figura 4.1).

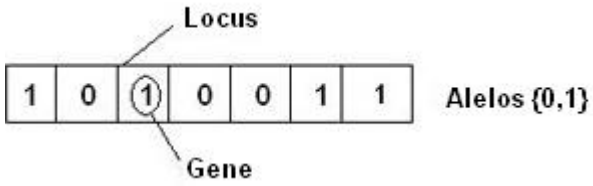

Figura 4.1: Exemplo de cromossomo com codificação binária.

Gene: Seqüência pequena de bases hidrogenadas capaz de definir uma característica do ser vivo. Em AGs, é um parâmetro codificado no cromossomo, ou um elemento do vetor que representa o cromossomo.

Alelo: Representa uma das formas alternativas de um gene, nos AG's representa os valores que o gene pode assumir. Por exemplo, um gene que representa o parâmetro cor de um objeto poderia ter o alelo de cor azul, preto, verde, etc.

Individuo: Formado pelo cromossomo e sua aptidão.

População: Soluções que representam os cromossomos, ou conjunto de indivíduos que evoluem por meio de operadores genéticos.

Fenótipo: É a manifestação do genótipo no comportamento, fisiologia e morfologia do indivíduo, como um produto de sua interação com o ambiente.

Genótipo: Estrutura de dados que representa uma solução candidata à um determinado problema.

Função de Avaliação (*fitness*): A função de avaliação associa a cada indivíduo da população uma medida de aptidão. Ela deve ser escolhida de forma a medir o desempenho de cada indivíduo como solução do problema.

4.1.1 Operadores Genéticos

Os operadores genéticos freqüentemente utilizados nos AG's clássicos são o *crossover* e a mutação. Estes operadores representam o núcleo de um AG. O objetivo deles é produzir novos cromossomos que possuam propriedades genéticas superiores às encontradas nos pais. A seguir apresentamos alguns desses operadores e seus principais aspectos.

***Crossover*:** consiste na troca aleatória de material genético entre dois cromossomos (strings - séries) selecionados. Entretanto existem alguns tipos de crossover, tais como:

- *Crossover de um ponto:* escolhe-se dois indivíduos da população (pais), seleciona-se aleatoriamente o ponto de corte e efetua a troca de modo a gerar dois filhos (ver figura 4.2).

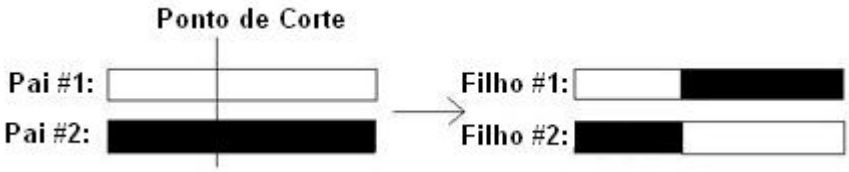

Figura 4.2: Exemplo de *crossover* de 1 ponto

- *Crossover de dois pontos:* seleciona-se dois indivíduos da população (pais), escolhe-se aleatoriamente dois pontos de cortes e efetua-se a troca de modo a gerar dois filhos (ver figura 4.3).

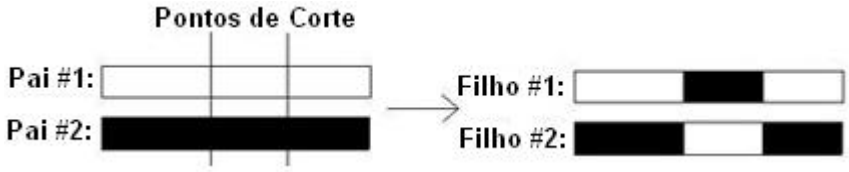

Figura 4.3: Exemplo de *crossover* de 2 pontos

- *Crossover Uniforme:* para cada gene no primeiro filho é decidido (com alguma probabilidade fixa p) qual pai vai contribuir com seu valor para aquela posição (ver figura 4.4).

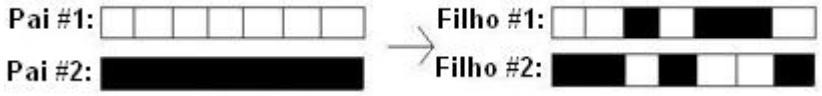

Figura 4.4: Exemplo de *crossover* Uniforme

- *Crossover OX:* escolhe-se dois indivíduos pais da população para gerar dois filhos. Primeiro copia-se um trecho de um dos indivíduos pais para um filho e o restante do cromossomo do novo indivíduo filho é preenchido com as informações do outro indivíduo pai, na mesma sequência para evitar valores repetidos nos genes (figura 4.5).

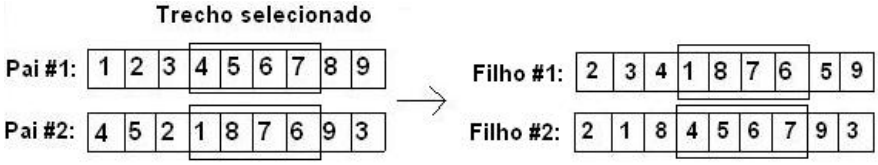

Figura 4.5: Exemplo de *crossover* OX

Mutação: geração de um novo indivíduo (filho) partindo de um único pai. O operador de mutação modifica aleatoriamente um ou mais genes de um cromossomo. A probabilidade de ocorrência de mutação em um gene normalmente é pequena. A idéia é apenas de criar uma variabilidade extra na população, mas sem destruir o processo já obtido com a busca. Existem alguns operadores de mutação, entretanto nos restringiremos apenas aos operadores abaixo.

- *Mutação Inversiva:* é bastante utilizada e pode ser feita através de uma simples troca de posição entre genes de um mesmo cromossomo. Este tipo de mutação, usando codificação inteira, é muito comum em problemas tradicionais como o problema do Caixeiro Viajante (ver figura 4.6).

- *Mutação Uniforme:* é bastante utilizado em problemas com codificação em ponto flutuante, assim como a mutação gaussiana. Este operador seleciona aleatoriamente um componente $k \in \{1, 2, ..., n\}$ do cromossomo $\mathbf{x} = [x_1, ..., x_p]$, e altera o gene na posição k, de tal forma que o indivíduo criado após mutação seja **x'** =

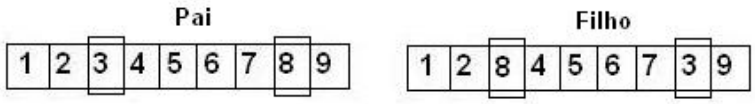

Figura 4.6: Exemplo de mutação inversiva

$[x_1, ..., x'_k, ...x_p]$, onde x'_k é um número aleatório (com distribuição de probabilidade uniforme) no intervalo $[x_{kmin}, x_{kmax}]$, onde x_{kmin} e x_{kmax} são, respectivamente, os limites inferior e superior do gene na posição k.

4.1.2 Mecanismos de Seleção

O processo de seleção num AG pode ser definido como sendo uma escolha probabilística de cromossomos de uma população tendo como base as suas aptidões (*fitness*), ou seja, quanto maior for a aptidão de um indivíduo, maior será a chance dele ser selecionado para a próxima geração. Existem diferentes estratégias de seleção, abaixo citaremos algumas dessas.

- *Roleta:* muito utilizado no AG clássico, esse método de seleção atribui a cada indivíduo da população uma probabilidade de passar para a próxima geração proporcional a seu valor de *fitness*, ou seja, quanto maior o *fitness* de um indivíduo, maior a probabilidade dele passar para a próxima geração. Entretanto, a grande desvantagem desse método é a possibilidade da perda do melhor indivíduo da população atual, ou seja, que ele não passe para a próxima geração. A figura 4.7 mostra a idéia desse mecanismo de seleção.

Indivíduo	Cromossomo	Fitness	% do total
a	11001	1	10
b	10101	3	30
c	10001	2	20
d	11110	4	40

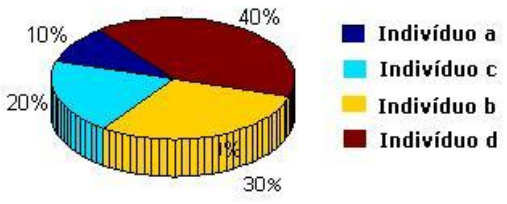

Figura 4.7: Exemplo de seleção por Roleta

- *Elitista:* são selecionados uma porcentagem dos melhores indivíduos da população atual.
- *Bi-classista:* são selecionados os $P\%$ melhores indivíduos e os $(1 - P)\%$ piores indivíduos.
- *Ranking:* esta estratégia utiliza as posições dos indivíduos quando ordenados de acordo com o *fitness* para determinar a probabilidade de seleção através de mapeamentos lineares ou não-lineares.
- *Torneio:* duas soluções são indicadas aleatoriamente e a melhor é selecionada se um número aleatório $r \in [0, 1]$ for menor que o parâmetro $p \in [0, 1]$ previamente determinado; caso contrário, a outra solução é escolhida.

4.1.3 Algoritmo Genético

A estrutura de um AG é baseada nos conceitos de reprodução celular e evolução natural, podendo ser descrito em quatro elementos básicos: uma população de soluções, uma função de desempenho, operadores genéticos e processo de seleção. A estrutura de um AG simples, facilmente encontrado na literatura, é mostrada abaixo.

Algoritmo 4.1.3: Algoritmo Genético Clássico

```
Início
  t = 0;
  Inicializa_População (P, t);
  Avalia (P, t);
  enquanto t ≠ d;          % critério de parada não atingido;
    faça
      t = t + 1;
      seleção_dos_pais (P, t);
      reprodução (crossover) (P, t);
      mutação (P, t);
      avaliação (P, t)';
    Fim
Fim
```

Capítulo 5

Algoritmo Híbrido

Neste capítulo é apresentado um algoritmo híbrido de sistema de colônia de formigas e genético para resolver o problema do caixeiro viajante. Esse novo algoritmo desenvolvido neste trabalho denominamos de ACS+AG-TSP, pois ele trabalha conjuntamente com as meta-heurísticas de ACS e AG, com o objetivo de obter melhores resultados que os algoritmos ACS e AG individualmente.

Inicialmente é feita uma breve introdução justificando a utilização das meta-heurísticas de ACS e AG, posteriormente apresentamos a estrutura do algoritmo híbrido e seu funcionamento.

5.1 Introdução

Modelos baseados em sistemas naturais objetivando resolver problemas de explosão combinatória demonstrou ser uma estratégia bastante eficiente. A meta-heurística de Colônia de Formigas, que surgiu a partir da observação de que formigas artificiais são capazes de encontrar o menor caminho entre o formigueiro e fonte de alimento, têm sido aplicada com sucesso em diversos problemas, entre eles destacamos o problema do caixeiro viajante [12], pois tem apresentado resultados bastante satisfatórios.

Os algoritmos genéticos são métodos heurísticos, não específicos, utilizados principalmente na exploração do espaço de soluções de problemas de otimização discreta. Eles são geralmente aplicados em problemas cuja complexidade dificulta, ou até mesmo impede a utilização de métodos exatos, já que são capazes de fornecer soluções quase ótimas em tempo de execução aceitável.

O problema do caixeiro viajante é um problema que envolve uma busca em um espaço que cresce fatorialmente conforme se aumenta o número de vértices do grafo, e portanto, ele é intratável por meio de buscas exaustivas ou por métodos exatos.

Baseando-se em técnicas da computação natural, um algoritmo híbrido denominado ACS+AG-TSP, que trabalha conjuntamente com as meta-heurísticas de ACS e AG, é proposto para resolver o problema do caixeiro viajante, cujo objetivo é explorar o espaço de busca de soluções de maneira eficiente, visando encontrar boas soluções de forma a contornar a questão da complexidade.

5.2 Implementação de um Algoritmo Híbrido ACS+AG-TSP

Dado $G : (V, A)$ um grafo onde V é um conjunto de vértices e A é um conjunto de arcos ou arestas associados a um custo c_{ij}, deseja-se encontrar a rota de menor custo, passando por cada vértice uma única vez.

Devido a característica inerente a este tipo de problema, e considerando as similaridades e diferentes características entre o algoritmos ACS e o AG, foi desenvolvido um novo algoritmo híbrido que combina a capacidade de busca das duas meta-heurísticas, para ser mais rápido, melhorar a capacidade de busca e tentar contornar a complexidade do problema.

Na abordagem ACS+AG-TSP há um processo de cooperação entre os algoritmos ACS e o AG. Esse processo se dá através de uma migração das soluções encontradas. A migração fornece uma capacidade adicional de intensificação para ambos os algoritmos. Neste aspecto, o ACS sustenta o AG, exportando soluções potenciais para sua população, e o AG sustenta o ACS reforçando alternativas de buscas potenciais para as formigas artificiais.

A estratégia de migração implementada é a seguinte: sempre que o algoritmo de formigas, o ACS, não consegue sair de um ponto ótimo local, ele exporta uma quantidade pré-determinada de soluções (melhores caminhos encontrados pelas formigas) encontradas até o momento, para formar a população inicial do algoritmo genético. Neste momento as iterações do ACS são interrompidas e o AG tem início para encontrar uma melhor solução que a do ACS. Uma vez que essa solução é encontrada, o algoritmo ACS adiciona feromônio nesse novo caminho, reforçando as alternativas de buscas para as formigas

artificiais.

5.2.1 Estrutura do Algoritmo Híbrido ACS+AG-TSP

O algoritmo híbrido proposto neste trabalho combina as meta-heurísticas de colônia de formigas e o algoritmo genético. O algoritmo de formigas utilizado na implementação do algoritmo híbrido é apresentado no capítulo 3, na subseção 3.3.1, e o algoritmo genético utilizado será descrito abaixo:

Representação: Cada cromossomo representa um grafo e seus genes representam os nós associados aos custos das arestas que ligam esses nós. No algoritmo genético desenvolvido para o algoritmo híbrido, cada cromossomo $C_i = (c_{i1}, c_{i2}, c_{i3}, ..., c_{in})$ representa um circuito (*tour* do caixeiro viajante) e n representa o número de cidades (Figura 5.1).

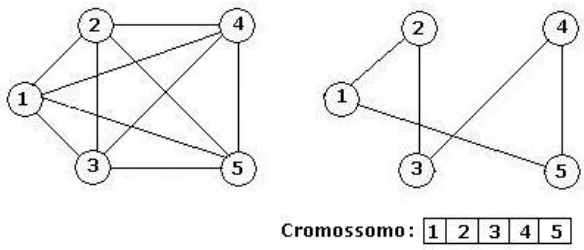

Figura 5.1: Exemplo de um Cromossomo representando um *tour* do TSP

Inicialização: A população inicial do algoritmo genético é dada pelos caminhos percorridos pelas formigas no algoritmo ACS quando este não obtém melhorias.

Função de Avaliação (*Fitness*): a função de avaliação associa a cada indivíduo da população uma medida de aptidão. Ela deve ser escolhida de forma a medir o desempenho de cada indivíduo como solução do problema. Neste algoritmo adotou-se um *fitness* igual ao custo do circuito, ou seja, o custo de cada viagem.

$$fitness(C_i) = \sum_{i=1}^{n} c_i$$

Variabilidade Genética: Segundo Michalewicz (1996), a variabilidade de uma população cai conforme o *fitness* aumenta. Isso ocorre porque a chance dos indivíduos gerarem descendentes é maior para aqueles que possuem *fitness* maiores. Com isso, o algoritmo utiliza os seguintes mecanismos para injetar variabilidade na população:

Seleção Bi-Classista: é preservada P% dos melhores indivíduos e (100 - P%) dos piores indivíduos da população.

Imigração: em cada geração, é injetada uma porcentagem de novos indivíduos, gerados de forma aleatória.

Crossover OX: O *crossover* é feito escolhendo-se dois indivíduos pais $\#P1$ e $\#P2$ da população para gerar dois indivíduos filhos $\#F1$ e $\#F2$. Primeiro é sorteado aleatoriamente um trecho de cada um dos pais; depois é copiado o trecho do pai $\#P1$ para o filho $\#F2$ e o do pai $\#P2$ para o filho $\#F1$; o restante do cromossomo do filho $\#F1$ e $\#F2$ é preenchido com as informações do outro pai $\#P2$ e $\#P1$ respectivamente, na mesma seqüência para evitar valores repetidos nos genes (Figura 5.2).

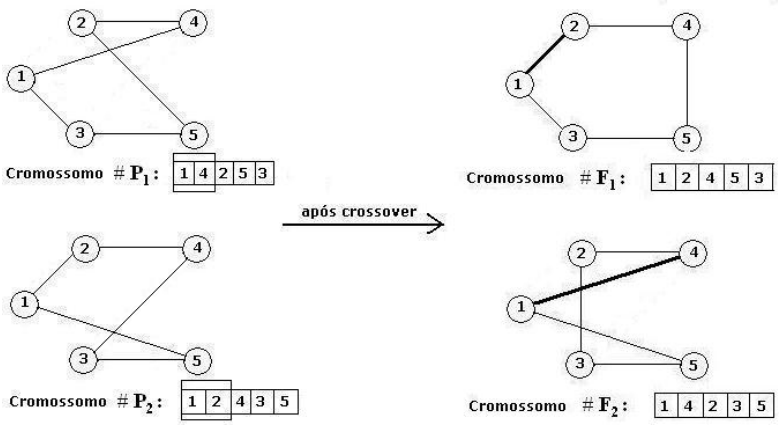

Figura 5.2: Exemplo de *crossover* entre dois pais $\#P1$ e $\#P2$

Mutação Inversiva: O operador de mutação utilizado foi a mutação inversiva, onde é escolhido aleatoriamente um indivíduo da população, seleciona dois pontos aleatórios do cromossomo e seus genes são permutados (Figura 5.3).

Figura 5.3: Exemplo de mutação em um *tour* do TSP

5.2.2 Algoritmo ACS+AG-TSP

Algoritmo 3.2.1: Algoritmo Híbrido ACS+AG-TSP

pop ← População recebida pelo ACS; L* ← menor distância encontrada pelo ACS
cgen ← *fitness* do AG S* ← melhor solução (ciclo) encontrada

Inicialização 1: Algoritmo ACS

Para cada aresta (i, j) do grafo, estabeleça um nível inicial de feromônio τ_{ij};
t ← 1;
enquanto t ≤ iteração, **faça**
  para cada formiga k, **faça**
    escolhe-se um nó aleatório para iniciar o percurso.
    construa uma solução aplicando uma regra construtiva probabilística influenciada pelo nível de feromônio das arestas e uma métrica heurística de afinidade à aresta;
    após ter completado o ciclo, atualize o feromônio motivado pela evaporação;
    calcule a distância $L_k(t)$ do caminho.
      se $L_k(t) < L^*$ **então** $S^* \leftarrow S_k(t)$ e $L^* \leftarrow L_k(t)$
      atualize o nível de feromônio de acordo com uma regra de atualização;
  fim-para
      se L^* não melhorar por n (estagnação) iteração **então**,
      Inicialização 2: Algoritmo Genético
        pop ← S;
        i ← 1;
        enquanto i ≤ geração ou L^* ≤ cgen, **faça**
          pop' ← seleção(pop);
          pop ← crossover (pop');
          pop ← mutação (pop);
          calcula *fitness* de pop;
          atualiza pop;
          i ← i + 1;
          se L^* > cgen, **pare**
            atualize o nível de feromônio de acordo com uma regra de atualização;
        fim-enquanto
  fim-enquanto
  t ← t+1;
Retorne a melhor solução.

Capítulo 6

Experimentos Computacionais

Neste capítulo são apresentados e discutidos os resultados obtidos com a aplicação dos algoritmos de colônia de formigas e genéticos no problema do caixeiro viajante. Foram utilizados alguns grafos para avaliar o desempenho do algoritmo genético, do algoritmo de formigas e do algoritmo híbrido proposto, o ACS+AG-TSP.

Para cada grafo foram realizados 10 experimentos, obtendo o melhor valor dentre os experimentos, sua média e o pior valor. Todas as instâncias utilizadas foram retiradas da base de dados *TSPLIB*[1], que é um repertório público de instâncias para o problema do caixeiro viajante na Internet onde, além das instâncias, consta também o melhor valor conhecido para algumas delas.

Todos os grafos utilizados nos testes são completos e simétricos, e o tamanho das instâncias variam de 51 a 226 cidades. Na tabela 6.1 encontram-se as descrições dos problemas simétricos do caixeiro viajante utilizados neste trabalho.

Devido ao fato do algoritmo híbrido ACS+AG-TSP trabalhar com dois algoritmos ACS e AG, ele consome mais tempo de processamento que os algoritmos ACS e AG para realizar cada experimento, mediante a igualdade dos parâmetros. Desta forma, a fim de obter uma comparação mais precisa, cada experimento do ACS e do AG foi testado utilizando o mesmo tempo de processamento consumido pelo ACS+AG-TSP. Por meio de gráficos gerados para cada instância é possível avaliar o desempenho do algoritmo híbrido proposto e saber os melhores resultados encontrados ao longo dos experimentos. No Apêndice A encontram-se alguns grafos obtidos pelos algoritmos.

Para a implementação dos algoritmos e realização dos testes foi utilizada a ferramenta

[1]http://www.iwr.uni-heidelberg.de/groups/comopt/soft/TSPLIB95/TSPLIB.html

MatLab® 7.0 na seguinte plataforma: Processador Pentium IV com 3.0Ghz e 1024 de memória RAM.

Instância	Número de cidades	Descrição do problema
eil51	51	problema de 51 cidades (Christofides/Eilon)
brazil58	58	problema de 58 cidades no Brazil (Ferreira)
eil76	76	problema de 76 cidades (Christofides/Eilon)
kroA100	100	problema de 100 cidades A (Krolak/Felts/Nelson)
bier127	127	problema de 127 cidades em Augsburg (Reinelt)
pr226	226	problema de 226 cidades (Padberg/Rinaldi)

Tabela 6.1: Problemas simétricos do Caixeiro Viajante utilizados neste trabalho

6.1 Parâmetros do Algoritmo Híbrido ACS+AG-TSP

Os seguintes parâmetros foram calibrados e utilizados para todos os experimentos:

1. **ACS**

   Número de formigas: 20;
   $\alpha = 1$, $\beta = 2$, $\rho = 0,1$, $q_0 = 0,9$;
   Critério de parada: número de iterações ou encontrar a melhor solução conhecida;

2. **AG**

   Tamanho da população de cromossomos: 60;
   Taxa de crossover: 60%;
   Taxa de mutação: 5%;
   Critério de parada: número de gerações ou encontrar melhor solução que a encontrada pelo ACS;

6.2 Resultados

6.2.1 Instância eil51

O grafo eil51 representa todas as conexões entre as 51 cidades de Christofides/Eilon, com as respectivas distâncias geográficas entre elas. Este é um grafo completo com 51 nós e 1275 arestas, que possui em torno de 3,04 x 10^{64} soluções possíveis.

Para avaliar o desempenho do algoritmo híbrido proposto neste trabalho, foi necessário implementar e análisar os resultados (Tabela 6.2) obtidos pelos algoritmos ACS e o AG.

Os resultados obtidos pelas heurísticas estão exibidos na Tabela 6.2. Em negrito são indicados os resultados dos algoritmos que atingiram a melhor solução conhecida e, em parênteses, encontra-se o erro encontrado por cada algoritmo. A convergência do algoritmo híbrido e a comparação do desempenho dos melhores resultados obtidos pelos algoritmos ACS+AG-TSP, ACS e AG, estão exibidos respectivamente nos gráficos das Figuras 6.1 e 6.2.

Algoritmo	Melhor sol. conhecida	Melhor sol. encontrada	Média das soluções	Pior sol. encontrada
ACS	**426**	**426**(0,0%)	426,8(0,18%)	430(0,94%)
AG		429(0,7%)	435(2,11%)	440(3,28%)
ACS+AG-TSP		**426**(0,0%)	426,4(0,09%)	427(0,23%)

Tabela 6.2: Resultados obtidos pelo ACS, AG e ACS+AG-TSP para o grafo eil51

Tanto o algoritmo híbrido quanto o algoritmo de formigas conseguiram encontrar a melhor solução conhecida, enquanto que o algoritmo genético obteve uma solução com erro de 0,7% da solução ótima. Entretanto, o algoritmo ACS+AG-TSP obteve um desempenho melhor que os algoritmos ACS e AG, pois sua média das soluções está próximo da solução ótima; isso pode ser observado também nos erros médios dos algoritmos, conforme mostra a Tabela 6.2. A média é de grande importância na análise dos resultados pois mostra a capacidade geral do algoritmo, quanto mais próximo este número estiver do valor ótimo melhor a performance do algoritmo.

Esta diferença nos resultados dos algoritmos pode ser explicada pelo fato do algoritmo ACS+AG-TSP trabalhar com um processo de cooperação entre os algoritmos ACS e AG,

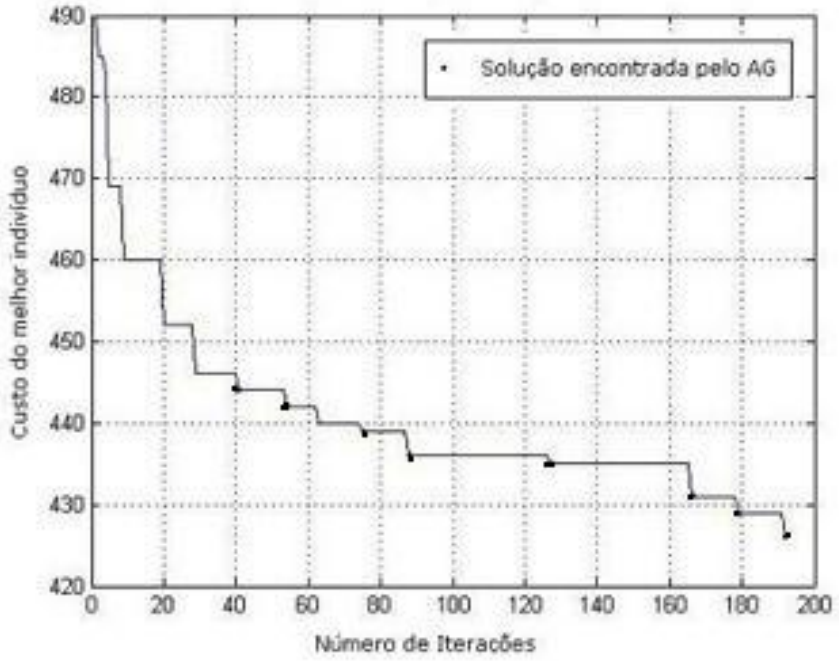

Figura 6.1: Gráfico de Convergência obtido pelo ACS+AG-TSP para o grafo eil51

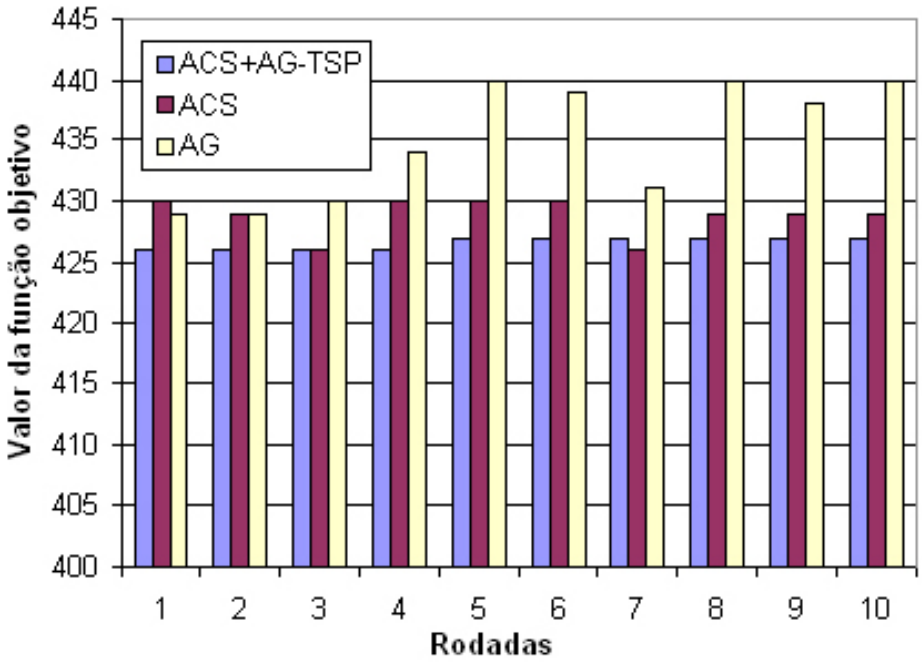

Figura 6.2: Gráfico de comparação do desempenho dos melhores resultados obtidos pelo ACS+AG-TSP, ACS e AG para a instância eil51

o que faz com que ele consiga manter a diversidade dos caminhos, permitindo-lhe obter melhor desempenho nos problemas.

Quanto a convergência do algoritmo híbrido ACS+AG-TSP, a Figura 6.1 mostra a evolução do custo do melhor indivíduo, evidenciando o processo de cooperação entre os algoritmos ACS e AG, e a contribuição do algoritmo genético no algoritmo de formigas, ajudando-o a sair de mínimos locais.

Na Figura 6.2 encontram-se os resultados detalhados de todas as rodadas realizadas com essa instância. Nela fica evidente a superioridade do algoritmo proposto ao longo dos experimentos, pois ele conseguiu encontrar soluções melhores que a dos outros algoritmos em 90% dos resultados. Os resultados da técnica proposta neste trabalho mostraram-se superiores aos algoritmos ACS e AG quando aplicados individualmente.

O tempo médio consumido em cada rodada para os algoritmos foi de 25 segundos.

6.2.2 Instância brazil58

Este grafo foi retirado da base de dados do *TSPLIB* e representa todas as possíveis conexões entre as 58 cidades no Brazil - Ferreira, com suas respectivas distâncias geográficas. Assim como todos os outros grafos aqui estudados, este é um grafo completo formado por 58 cidades e 1653 arestas, que possui em torno de $4,05 \text{ x } 10^{76}$ candidatos a solução. A principal característica deste grafo é a existência de poucos pontos de mínimos locais. Esse fato influência diretamente na qualidade das soluções e é favorável ao algoritmo genético.

Os resultados obtidos com esta instância estão exibidos na Tabela 6.3

Algoritmo	Melhor sol. conhecida	Melhor sol. encontrada	Média das soluções	Pior sol. encontrada
ACS	**25395**	**25395**(0,0%)	25517(0,48%)	25643(0,97%)
AG		**25395**(0,0%)	25578(0,78%)	25966(2,24%)
ACS+AG-TSP		**25395**(0,0%)	25398(0,01%)	25405(0,04%)

Tabela 6.3: Resultados obtidos pelo ACS, AG e ACS+AG-TSP para o grafo brazil58

Ambos os algoritmos conseguiram encontrar boas soluções, pois todos encontraram a melhor solução conhecida, entretanto, por uma pequena diferença, o ACS+AG-TSP obteve

um melhor desempenho. Enquanto ele conseguiu encontrar a média das soluções com um erro médio bem pequeno, de $0,01\%$, o ACS encontrou uma média com erro de $0,48\%$ e o AG com $0,78\%$.

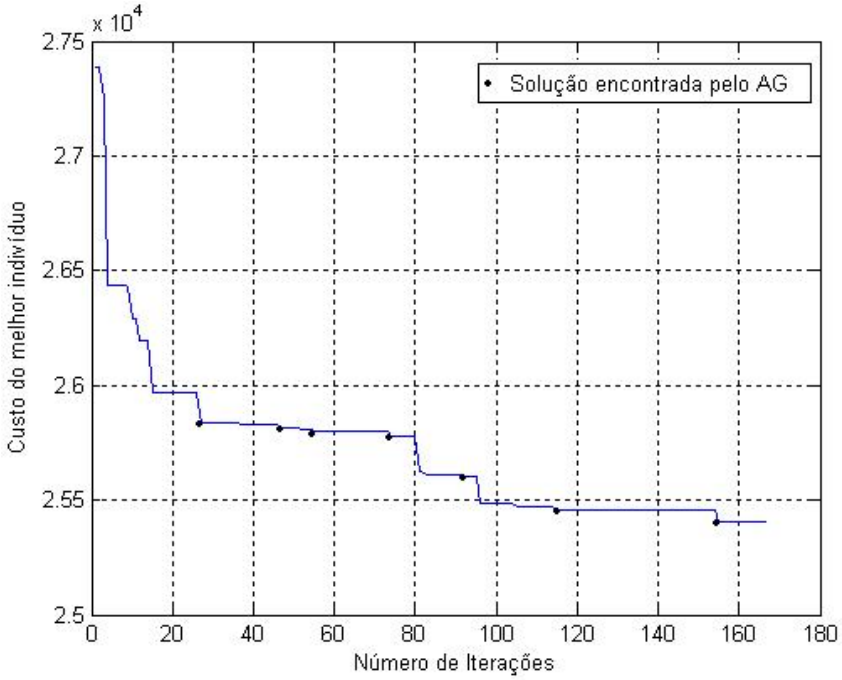

Figura 6.3: Gráfico de Convergência obtido pelo ACS+AG-TSP para o grafo brazil58

Em relação a convergência do algoritmo híbrido (Figura 6.3) podemos observar que ele conseguiu encontrar a solução ótima com menos avaliações, apesar do grafo ser um pouco maior que o anterior. O tempo médio consumido pelo algoritmo para realizar cada experimento foi de 10 segundos. Neste tempo, o algoritmo híbrido ACS+AG-TSP executou 500 gerações do AG e 300 iterações do ACS. Na figura 6.3 temos uma melhor visualização da convergência do algoritmo para esta instância (cabe ressaltar que este gráfico foi escolhido, pois nele temos uma melhor visualização da convergência do algoritmo para esta instância, apesar de dentre os 10 experimentos obtivemos convergências bem mais rápidas que a apresentada).

Na Figura 6.4, é possível observar que a solução ótima é encontrada várias vezes pelos algoritmos ACS+AG-TSP, ACS e AG, entretanto é possível notar que o algoritmo genético encontrou alguns valores distantes da solução ótima e que o algoritmo ACS+AG-TSP obteve melhor desempenho no conjunto dos 10 experimentos.

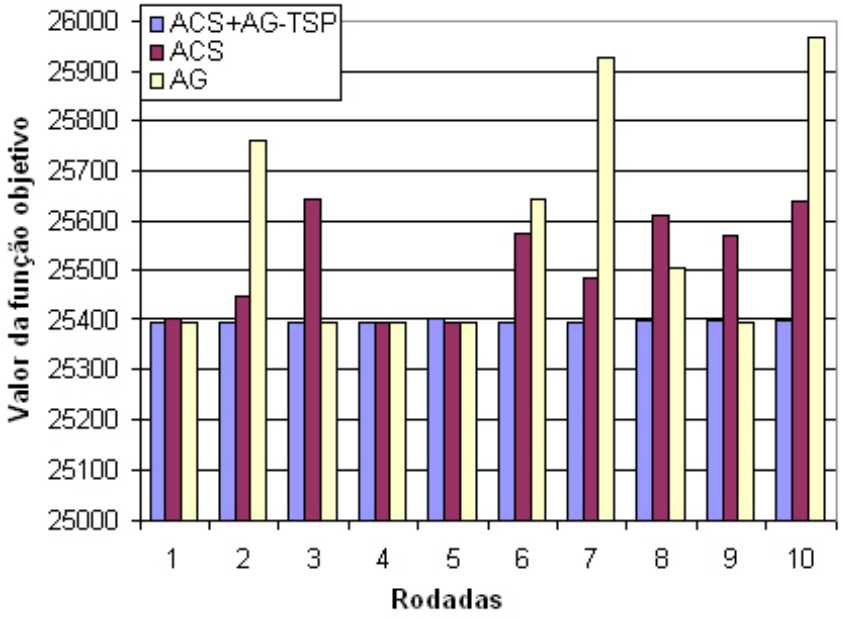

Figura 6.4: Gráfico de comparação do desempenho dos melhores resultados obtidos pelo ACS+AG-TSP, ACS e AG para a instância brazil58

6.2.3 Instância eil76

A instância eil76 é um grafo completo com 76 nós e 2850 arestas e também foi retirada da base de dados *TSPLIB*. Ela representa todas as ligações possíveis entre as 76 cidades de Christofides/Eilon e possui em torno de $2,48 \times 10^{109}$ soluções possíveis. Os resultados obtidos pelos algoritmos estão exibidos na Tabela 6.4.

Algoritmo	Melhor sol. conhecida	Melhor sol. encontrada	Média das soluções	Pior sol. encontrada
ACS	**538**	543(0,9%)	547,3(1,72%)	549(2,04%)
AG		551(2,42%)	564,7(4,96%)	573(6,5%)
ACS+AG-TSP		**538**(0,0%)	540(0,37%)	543(0,9%)

Tabela 6.4: Resultados obtidos pelo ACS, AG e ACS+AG-TSP para o grafo Eil76

Pela análise da Tabela 6.4 é possível concluir que, para os experimentos realizados com este grafo, o algoritmo ACS+AG-TSP obteve as melhores soluções, encontrando a

solução ótima e erro médio das soluções superior aos algoritmos ACS e AG.

A convergência do algoritmo híbrido e o comportamento das melhores soluções estão exibidos nos gráficos das Figuras 6.5 e 6.6, respectivamente.

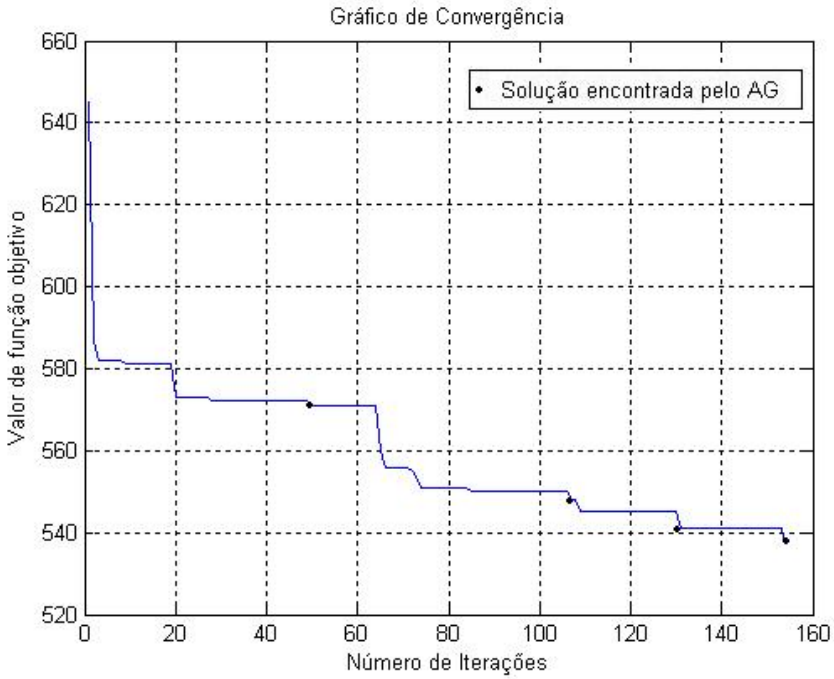

Figura 6.5: Gráfico de Convergência obtido pelo ACS+AG-TSP para o grafo Eil76

No gráfico da Figura 6.5 vemos que o algoritmo de formigas conseguiu convergir mais rápido para a solução ótima e que a solução ótima foi encontrada pelo algoritmo genético.

Em comparação com os melhores resultados por experimento, ficou evidente no histograma da Figura 6.6, que novamente o algoritmo proposto neste trabalho conseguiu encontrar melhores soluções que os algoritmos ACS e AG quando aplicados individualmente, pois ele conseguiu encontrar a solução ótima e um valor médio bem mais próximo da solução que os outros dois algoritmos.

O tempo médio consumido em cada rodada pelos algoritmos foi de 1 minuto e 35 segundos. Neste tempo, o ACS executou 300 iterações e o AG 600 gerações no algoritmo ACS+AG-TSP.

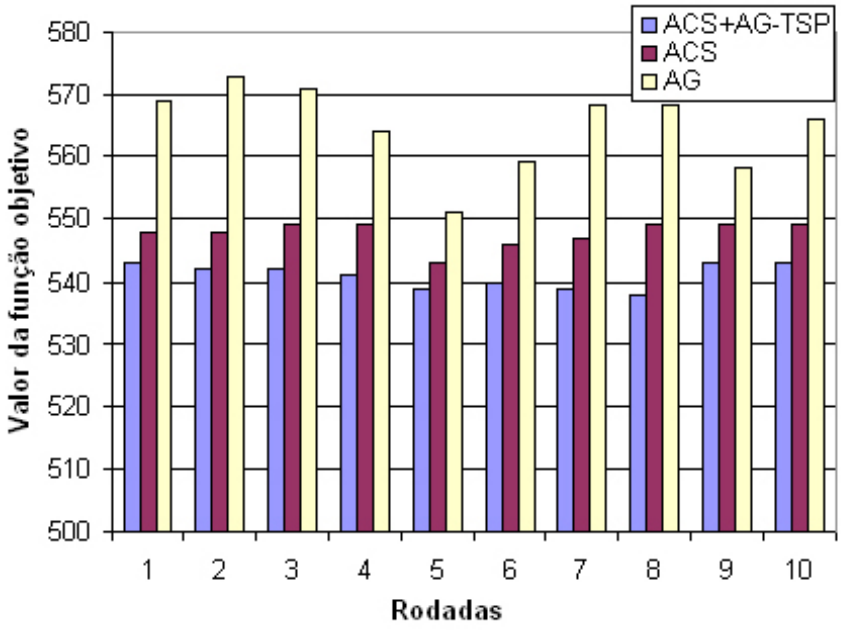

Figura 6.6: Gráfico de comparação do desempenho dos melhores resultados obtidos pelo ACS+AG-TSP, ACS e AG para a instância Eil76

6.2.4 Instância kroA100

Também retirado do *TSPLIB*, este é um grafo completo composto de 100 nós e 4950 arestas. Ele possui em torno de $9,33 \times 10^{155}$ soluções possíveis.

Na Tabela 6.5 estão os resultados obtidos para esta instância. Fazendo uma análise desses resultados pode-se perceber que ambos os algoritmos (ACS, AG e ACS+AG-TSP) conseguiram boas soluções, encontrando valores bem próximos do valor ótimo com erros abaixo de 1%. Entretanto apenas o algoritmo híbrido conseguiu encontrar a melhor solução conhecida, mostrando-se ser mais eficiente.

Algoritmo	Melhor sol. conhecida	Melhor sol. encontrada	Média das soluções	Pior sol. encontrada
ACS	**21282**	21320(0,17%)	21471(0,88%)	21597(1,48%)
AG		21395(0,53%)	21679(1,86%)	21970(3,23%)
ACS+AG-TSP		**21282**(0,0%)	21291(0,04%)	21307(0,12%)

Tabela 6.5: Resultados obtidos pelo ACS, AG e ACS+AG-TSP para o grafo kroA100

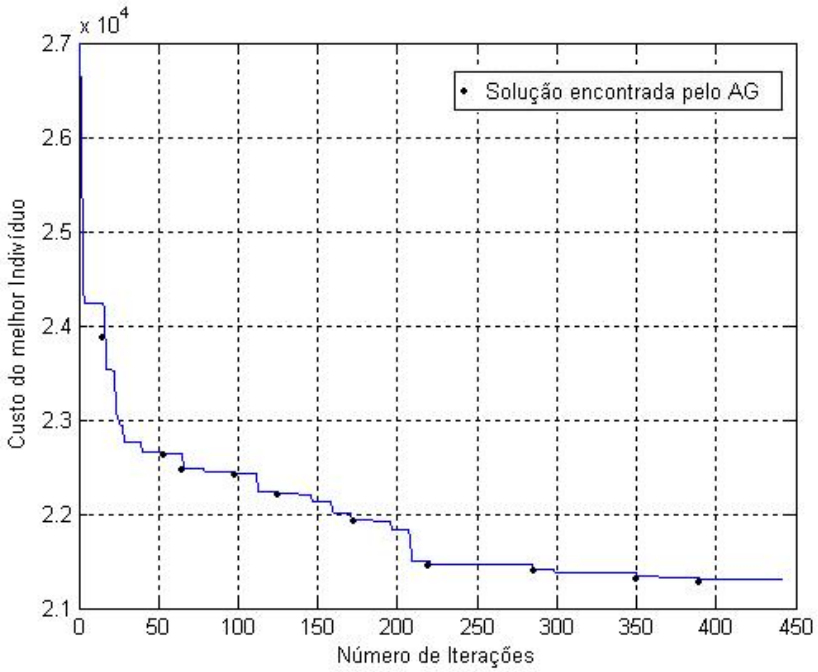

Figura 6.7: Gráfico de Convergência obtido pelo ACS+AG-TSP para o grafo kroA100

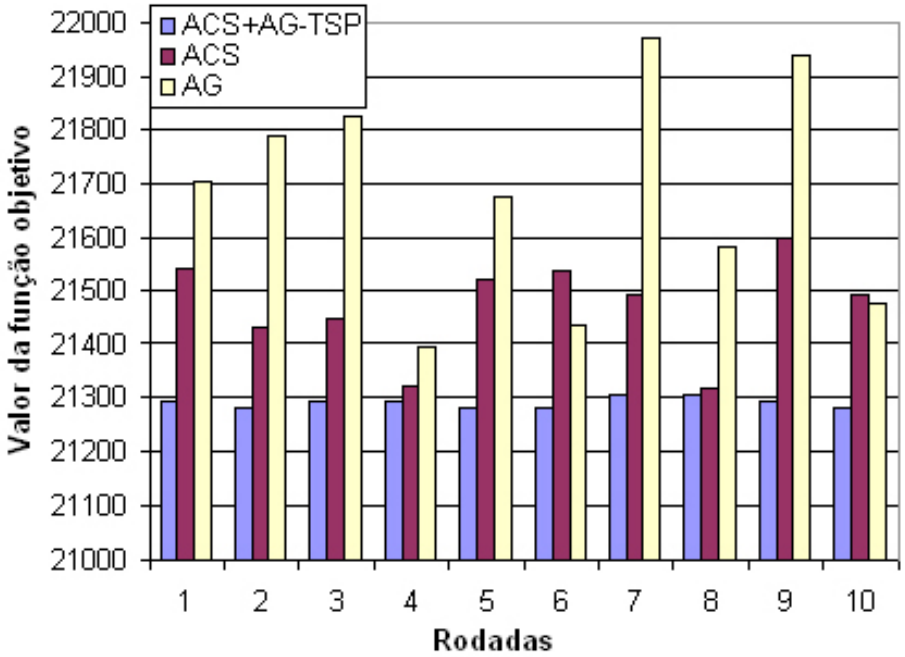

Figura 6.8: Gráfico de comparação do desempenho dos melhores resultados obtidos pelo ACS+AG-TSP, ACS e AG para a instância kroA100

Em relação a convergência do algoritmo híbrido (Figura 6.7), ficou evidente a cooperação entre os algoritmos ACS e AG. A melhor solução conhecida foi encontrada pelo algoritmo de formigas.

No gráfico da Figura 6.8 encontram-se os resultados do desempenho dos algoritmos nas 10 rodadas. Nele podemos ver que os melhores resultados dentre todos os testes foram do algoritmo híbrido, encontrando o melhor valor conhecido várias vezes ao longo dos experimentos, e que os resultados do AG ao longo das rodadas foram os piores.

O tempo consumido em cada rodada, para cada um dos algoritmos, foi em média de 4 minutos e 40 segundos. Neste tempo o algoritmo ACS+AG-TSP executou 500 gerações do AG e 300 iterações do ACS.

6.2.5 Instância bier127

O grafo bier127 é um grafo completo com 127 nós representando todas as conexões da região de Augsburg na Alemanha. Ele possui em torno de 2,37 x 10^{211} possíveis soluções.

Na 6.9 (a) e (b) encontram-se os gráficos de convergência dos algoritmos ACS e ACS+AG-TSP, respectivamente. Nele percebemos que o algoritmo ACS+AG-TSP obteve um desempenho melhor que o algoritmo ACS, conseguindo encontrar a solução ótima, enquanto que o algoritmo ACS apresentou dificuldades de convergência.

Em relação aos resultados (Tabela 6.6), o método proposto foi mais eficiente e conseguiu encontrar bons resultados. O algoritmo ACS obteve resultados melhores que o AG, já que o melhor valor encontrado pelo ACS foi com erro de 0,78% do valor ótimo, enquanto que no AG esse erro foi superior a 2%. A superioridade nos valores encontrados pelo algoritmo híbrido pode ser visto na média das soluções e até mesmo no pior valor encontrado pelos algoritmos, pois os erros do algoritmo ACS+AG-TSP são todos inferiores a 1%.

Algoritmo	Melhor sol. conhecida	Melhor sol. encontrada	Média das soluções	Pior sol. encontrada
ACS	**118282**	119206(0,78%)	120418,8(1,8%)	121332(2,58%)
AG		120884(2,19%)	124816,7(5,52%)	126240(6,72%)
ACS+AG-TSP		**118282**(0,0%)	118958(0,57%)	119458(0,99%)

Tabela 6.6: Resultados obtidos pelo ACS, AG e ACS+AG-TSP para o grafo bier127

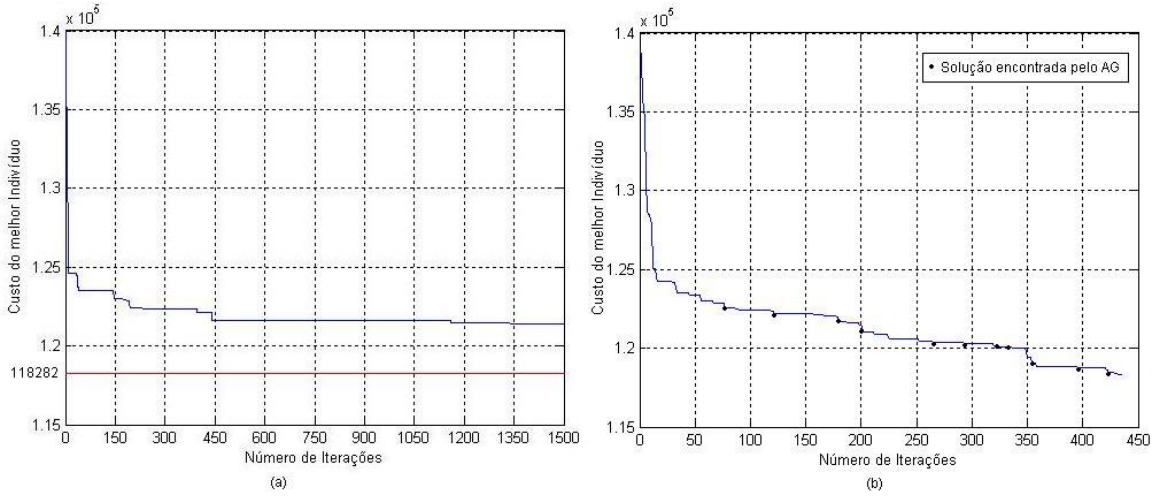

Figura 6.9: (a) Gráfico de Convergência obtido pelo algoritmo ACS e (b) pelo algoritmo ACS+AG-TSP para o grafo bier127

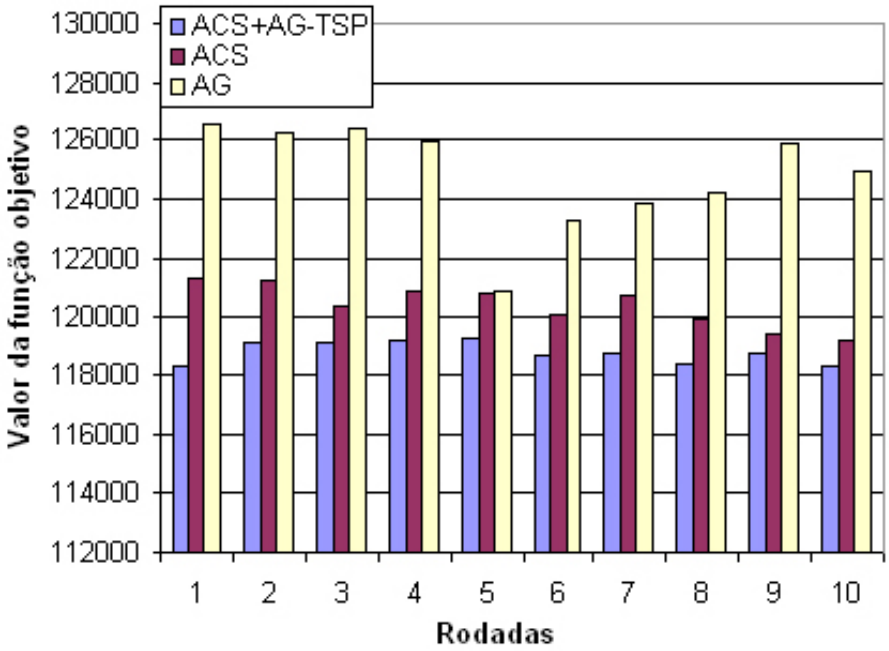

Figura 6.10: Gráfico de comparação do desempenho dos melhores resultados obtidos pelo ACS+AG-TSP, ACS e AG para a instância bier127

Em relação a Figura 6.9 (b), percebe-se que houve um processo de cooperação grande entre os algoritmos, chegando rapidamente a solução ótima. Na Figura 6.10 encontram-se os melhores valores encontrados pelos algoritmos ao longo dos experimentos. Os melhores valores encontrados em todos os experimentos foram os do algoritmo ACS+AG-TSP.

Por se tratar de um grafo maior, o tempo médio consumido para a realização de cada experimento foi de aproximadamente 19 minutos. Nesse tempo, enquanto que o ACS realizou 700 iterações o AG realizou 600 gerações no algoritmo proposto (ACS+AG-TSP).

6.2.6 Instância pr226

O grafo pr226, também disponível no repertório *TSPLIB*, representa todas as ligações possíveis entre 226 cidades de (Padberg/Rinaldi). Este é um grafo completo, bem maior que os grafos anteriores, que possui 25425 arestas e em torno de 1,25 x 10^{433} soluções possíveis. Os resultados obtidos com esta instância estão exibidos na Tabela 6.7.

Algoritmo	Melhor sol. conhecida	Melhor sol. encontrada	Média das soluções	Pior sol. encontrada
ACS	**80369**	81243(1,08%)	81997(2,02%)	82490(2,63%)
AG		81904(1,9%)	83680,5(4,12%)	84782(5,49%)
ACS+AG-TSP		**80369**(0,0%)	80505,9(0,17%)	80644(0,34%)

Tabela 6.7: Resultados obtidos pelo ACS, AG e ACS+AG-TSP para o grafo pr226

Analisando os resultados da Tabela 6.7, verifica-se que mais uma vez os resultados encontrados pelo algoritmo ACS+AG-TSP foram melhores que os dos outros dois algoritmos, comprovando a eficiência e superioridade do método proposto.

Em relação a convergência do algoritmo híbrido para o grafo pr226, percebe-se na Figura 6.11 que a convergência foi rápida e que houve uma cooperação grande entre os algoritmos. O tempo médio consumido pelos algoritmos para realizar cada experimento foi de 21 minutos. Nesse tempo, o algoritmo ACS+AG-TSP executou 700 iterações do ACS e 600 gerações do AG.

Na Figura 6.12 estão os erros encontrados pelo algoritmo ACS+AG-TSP, em todos os grafos utilizados neste trabalho.

Os erros apresentados na Figura 6.12 mostram resultados satisfatórios para os problemas utilizados, pois os erros médios são todos inferiores a 2%.

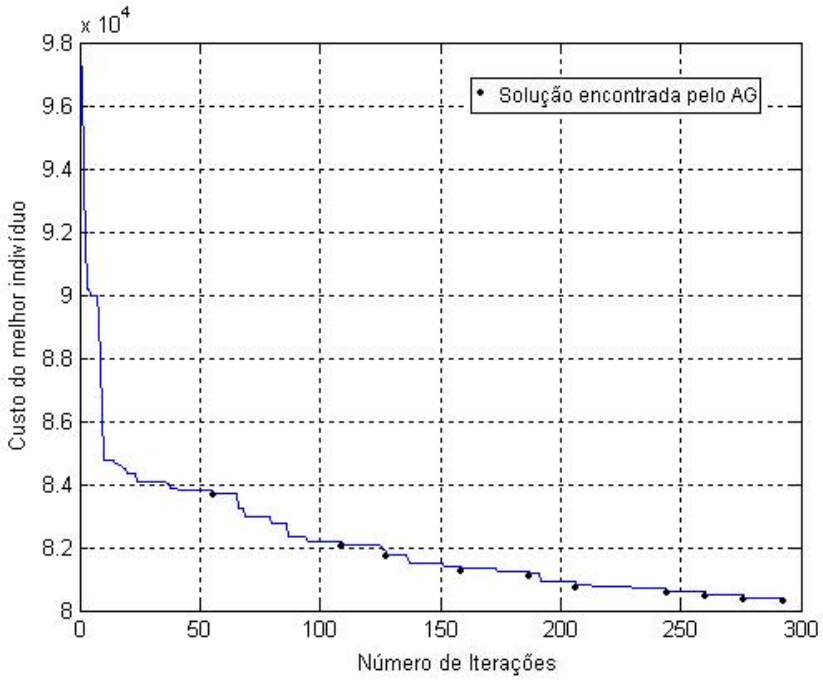

Figura 6.11: Gráfico de Convergência obtido pelo ACS+AG-TSP para o grafo pr226

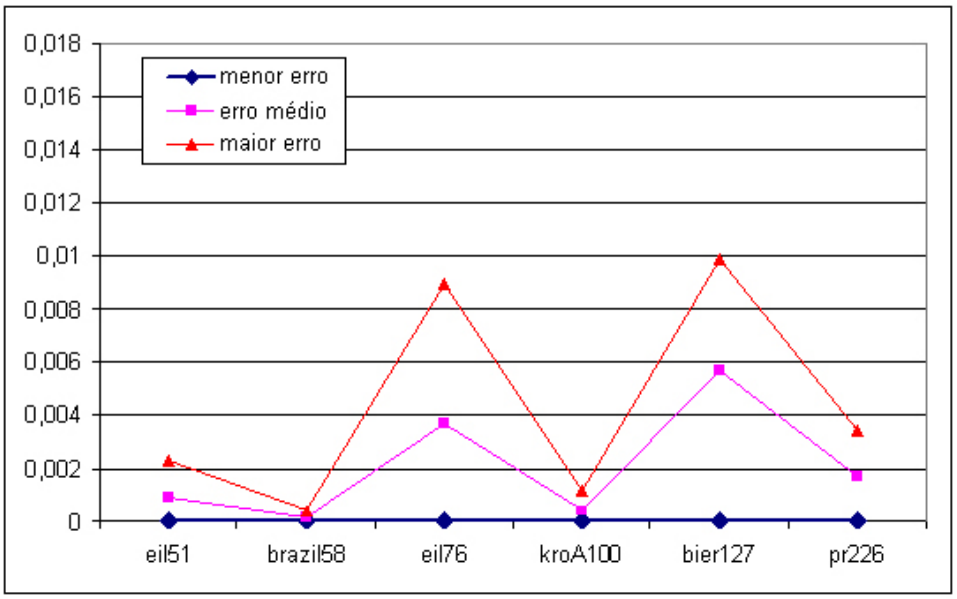

Figura 6.12: Erros médios encontrados pelo algoritmo ACS+AG-TSP

Capítulo 7

Extensão do *Ant Colony System* para o Problema do Caixeiro Viajante com Parâmetros Incertos

Neste capítulo apresentamos uma extensão do algoritmo de formigas para um grafo com estrutura *crisp* e parâmetros representados por números *fuzzy*. Inicialmente são fornecidas algumas noções básicas da teoria fuzzy e da teoria da possibilidade. Para maiores esclarecimentos sobre a teoria *fuzzy* consultar [14] e [27].

No capítulo 3 foi feito um estudo do algoritmo ACS com parâmetros bem definidos. Neste capítulo, o principal objetivo é fazer um estudo sobre o algoritmo ACS com parâmetros incertos, já que em situações reais é comum encontrarmos incertezas associadas às informações dos problemas.

Ao trabalhar com informações incertas, uma unidade deixa de ser representada por um número e passa a ser representada por um conjunto de valores. Considere, por exemplo, um grafo G(N,A), onde N é o conjunto de vértices representando as cidades, A é o conjunto de arcos que conectam cada par de cidades com custos entre a cidade (nó) i e a cidade j dado pelo tempo de translado que uma pessoa leva para atravessar essas cidades. Esse tempo pode ser considerado um parâmetro sujeito a incertezas, devido ao congestionamento, condições da rodovia, condições climáticas e até mesmo disposição do motorista. Desta forma, o uso da teoria clássica torna-se inviável pela sua ineficiência no tratamento de informações imprecisas. Entretanto, essas incertezas podem ser modeladas e tratadas de uma forma mais robusta utilizando a teoria dos conjuntos nebulosos ([33],

[14] e [27]).

Em problemas de grafos, as incertezas podem estar presentes em diversos níveis: na estrutura do grafo (quando não é possível determinar com certeza se uma aresta está conectada a um nó, ou até mesmo, se aquela aresta existe), nos parâmetros (quando não se sabe exatamente quanto vale o peso das arestas) e os dois casos ocorrendo ao mesmo tempo.

Neste capítulo vamos analisar e propor um algoritmo do tipo ACS-Fuzzy aplicando-o em alguns exemplos do TSP.

7.1 Conjuntos fuzzy

Um conjunto *fuzzy* pode ser visto como uma generalização de um conjunto clássico, possibilitando atribuir um grau de pertinência para cada elemento, e portanto, a função de pertinência do conjunto *fuzzy* mapeia cada elemento do universo de discurso a seu espaço de escala.

$$\mu_A(x) = \begin{cases} 1, & \text{se } x \in A \\ 0, & \text{se } x \notin A \end{cases} \tag{7.1}$$

Um subconjunto *fuzzy* A em X é definido por uma função de pertinência μ_A que associa cada ponto de X a um número real no intervalo $[0, 1]$, com o valor de μ_A em x representando o grau de pertinência de x em A. Desta forma, quanto mais próximo o valor de $\mu_A(x)$ estiver da unidade, maior será o grau de pertinência de x em A (Figura 7.1).

Definição 7.1 *Um conjunto fuzzy A é dito ser normal se sua função de pertinência* (μ_A) *possui pelo menos um valor tal que* $\mu_A(x) = 1$. *Caso contrário o conjunto é denominado subnormal.*

Definição 7.2 *O suporte de um conjunto fuzzy A, denotado por Supp*(A)*, é formado pelos elementos que possuem valores de pertinência não nulos, dados por:*

$$Supp(A) = \{x \in X \mid \mu_A(x) > 0\}$$

Definição 7.3 *Um conjunto fuzzy é dito convexo se sua função de pertinência é tal que*

$$\mu_A[\lambda x_1 + (1-\lambda)x_2] \geq \lambda\mu_A(x_1) + (\lambda - 1)\mu_A(x_2),$$

onde quaisquer $x_1, x_2 \in Supp(A)$ e $\lambda \in [0,1]$.

7.2 Números *fuzzy*

A lógica *fuzzy* é um meio de aproximar a precisão matemática clássica e a *imprecisão* do mundo real. A teoria *fuzzy* consegue manipular e operar quantidades exatas e inexatas (quantificadores através de valores lingüísticos). Existe uma grande variedade de tipos de números *fuzzy* [27]. Entretanto, neste trabalho, vamos tratar apenas números *fuzzy* triangular.

Definição 7.4 *Um número fuzzy A é um conjunto fuzzy no espaço dos números reais R com função de pertinência $\mu_A : R \to [0,1]$.*

Exemplo 7.5 *Um número fuzzy triangular, é dado por:*

$$a = [a_i, a_m, a_s]$$

onde, a_m é o valor modal do número *fuzzy* triangular (valor máximo da função de pertinência associada), a_i é o valor inferior e a_s é o valor superior. Sua função de pertinência é definida de acordo com a Equação 7.2.

$$\mu_A(x) = \begin{cases} \frac{x-a_i}{a_m-a_i}, & \text{se } x \in [a_i, a_m] \\ \frac{a_s-x}{a_s-a_m}, & \text{se } x \in [a_m, a_s] \\ 0, & \text{caso contrário} \end{cases} \tag{7.2}$$

A representação gráfica de um número *fuzzy* triangular do tipo $a = [a_i, a_m, a_s]$ é apresentada pela Figura 7.1.

7.2.1 Operações com números *fuzzy* triangulares

Para as operações definidas abaixo, considere $a = [a_i, a_m, a_s]$ e $b = [b_i, b_m, b_s]$ números *fuzzy* quaisquer:

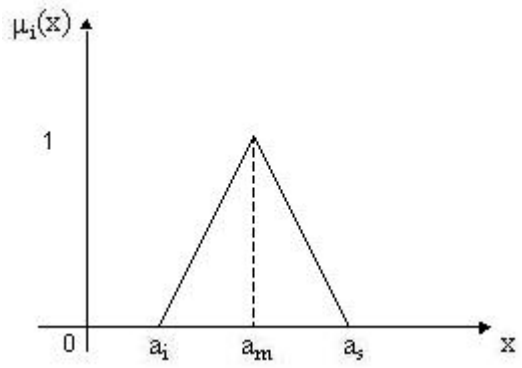

Figura 7.1: Número *fuzzy* triangular

- Adição: $a \oplus b = [a_i + b_i, a_m + b_m, a_s + b_s]$;
- Subtração: $a \ominus b = [a_i - b_s, a_m - b_m, a_s - b_i]$;
- Multiplicação: $a \otimes b = [Min\{a_i b_i, a_i b_s, a_s b_i, a_s b_s\}, a_m b_m, Max\{a_i b_i, a_i b_s, a_s b_i, a_s b_s\}]$;
- Divisão: $\frac{a}{b} = [Min\{\frac{a_i}{b_i}, \frac{a_i}{b_s}, \frac{a_s}{b_i}, \frac{a_s}{b_s}\}, \frac{a_m}{b_m}, Max\{\frac{a_i}{b_i}, \frac{a_i}{b_s}, \frac{a_s}{b_i}, \frac{a_s}{b_s}\}]$;
- Inversa: $a^{-1} = [Min\{\frac{1}{a_i}, \frac{1}{a_s}\}, \frac{1}{a_m}, Max\{\frac{1}{a_i}, \frac{1}{a_s}\}]$, (note que $\frac{a}{b} = a.b^{-1}$)

7.3 Teoria da Possibilidade

Seja um grafo $G(N, M)$, no qual N denota o conjunto de nós, e M o conjunto de arcos que têm associado um custo representado por um número *fuzzy* $\widetilde{c}_{ij} \in \mathbb{R}^n$. Sejam dois números *fuzzy* t^1 e t^2, $t^1 \neq t^2$. Podemos dizer que t^1 tem um grau de possibilidade de ser menor do que t^2 dado por [32]:

$$\widetilde{w} = Poss(\sum_{ij \in t^1} \widetilde{c}_{ij} \leq \sum_{ij \in t^2} \widetilde{c}_{ij}) = sup \min_{(u \leq v)} \{\mu_{t^1}(u), \mu_{t^2}(v)\} \tag{7.3}$$

Para encontrar uma solução *fuzzy* utilizando a teoria de possibilidade, é necessario encontrar todas as soluções que possuem algum grau de possibilidade de ser a solução ótima e comparar estas soluções para obter o grau de possibilidade de cada uma [25]. O grau de possibilidade é dado pela fórmula:

$$D_t = \min_{t^k \in t} \{Poss(\sum_{ij \in t^k} \tilde{c}_{ij} \leq \sum_{ij \in t} \tilde{c}_{ij})\} \tag{7.4}$$

Isto torna o problema de difícil solução, pois além de ter que enumerar todas as soluções, a comparação entre elas torna o problema NP- difícil.

Exemplo 7.6 *Considerando dois números fuzzy $\tilde{a}$ e $\tilde{b}$, com funções de pertinência $\mu_{\tilde{a}}(x)$ e $\mu_{\tilde{b}}(x)$ (Figura 7.2), a possibilidade $Poss(\tilde{a}, \tilde{b})$ é apresentada na Figura 7.3 [4]. Sendo definida por $Poss(\tilde{b} \leq \tilde{a}) = \sup_{(x \in X)} \{\min\{\mu_{\tilde{a}}(x), \mu_{\tilde{b}}(x)\}\}$*

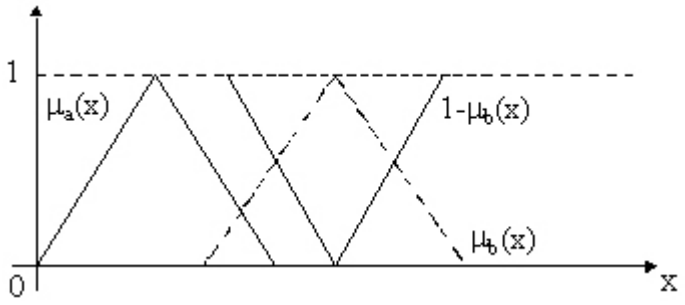

Figura 7.2: Funções de pertinência $\mu_{\tilde{a}}(x)$ e $\mu_{\tilde{b}}(x)$

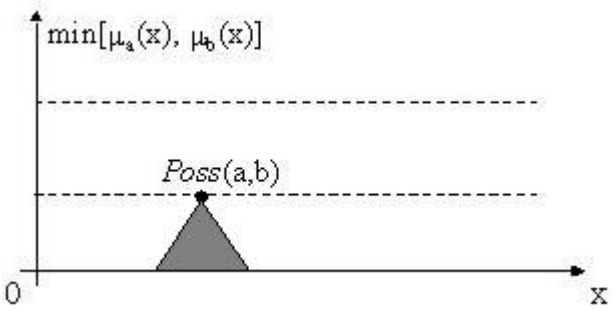

Figura 7.3: Possibilidade $Poss(\tilde{a}, \tilde{b})$

Para se responder a pergunta de qual é o maior ou menor número entre vários números *fuzzy* é preciso avaliar o grau de possibilidade do número *fuzzy* ser maior ou menor que outro número *fuzzy*. Para verificar se $a \geq b$, devemos analisar a Equação 7.3.

Daí pode-se concluir, pela Equação 7.5, que para $a = [a_i, a_m, a_s]$ e $b = [b_i, b_m, b_s]$:

$$\begin{cases} Poss(a \geq b) = 1, & \text{se e somente se } a_m \geq b_m \\ Poss(a \geq b) = 0, & \text{se } a_s < b_i \\ Poss(b \geq a) = \text{hgt}(a \cap b) \end{cases} \tag{7.5}$$

De acordo com a Equação 7.5, temos a seguinte definição:

Definição 7.7 *$hgt(a \cap b)$ representa a altura da intersecção dos números fuzzy a e b. Para compararmos os números a e b, é preciso calcular a $Poss(a \geq b)$ e a $Poss(b \geq a)$.*

7.4 Estrutura do Algoritmo ACS com Incertezas

Seja $G = (N, A)$ um grafo conexo com m nós e n arestas com custo *fuzzy* associado à cada aresta (a_i, a_m, a_s). O problema consiste em encontrar um caminho de menor custo entre todos os nós (Figura 7.4).

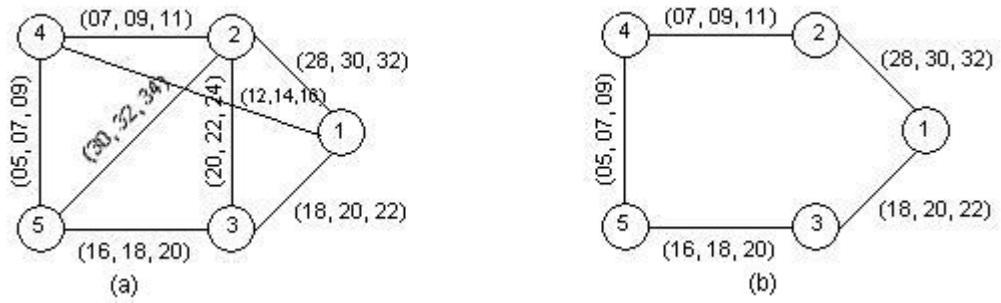

Figura 7.4: Exemplo de um problema do TSP com custos fuzzy

Neste grafo com parâmetros *fuzzy* (Figura 7.4 (a)) foi obtido um caminho mínimo do TSP (Figura 7.4 (b)) com custo *fuzzy* igual a $(74, 84, 94)$ que está associado a função de pertinência representado pela Figura 7.5. Como pode ser observado na função de pertinência da Figura 7.5, os *números fuzzy* estão representados por uma *função triangular* (ver Exemplo 7.5 na seção 7.2) e podem ser definidos por *números triangulares fuzzy* do tipo (a_i, a_m, a_s), onde a_i representa o custo inferior, a_m o custo modal e a_s o custo superior.

Nossa proposta aqui é trabalhar com a teoria da possibilidade [34] para tratar a solução do problema do caixeiro viajante com o algoritmo de formigas. O que difere o algoritmo ACS do ACS-*Fuzzy* é basicamente a forma de escolha do próximo passo da formiga, pois no ACS esse passo é dado de acordo com as fórmulas determinísticas e probabilísticas (ver equações 3.2, 3.3 na seção 3.3) e no ACS-Fuzzy, além dessas fórmulas, utilizamos o conceito da teoria da possibilidade (seção 7.3). Para entendermos melhor, considere o exemplo abaixo:

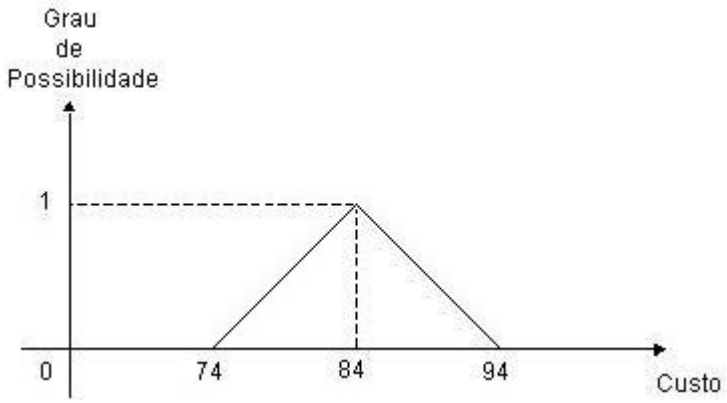

Figura 7.5: Função de pertinência associado ao custo mínimo do grafo da Figura 7.4

Exemplo 7.8 *Suponha que uma formiga iniciou o percurso no nó* 2 *(Figura 7.6). Qual será o próximo passo a ser dado pela formiga?*

Como esse movimento deve levar em conta a quantidade de feromônio existente na aresta e seu comprimento, já que a preferência da formiga por um determinado caminho é maior para os caminhos com maior nível de feromônio e com menor distância, devemos calcular esses níveis e decidir qual será o próximo passo.

Supondo que calculamos as probabilidades (níveis) dos nós (de acordo com a equação 3.3), a maior foi a do nó ♯1 (Figura 7.6).

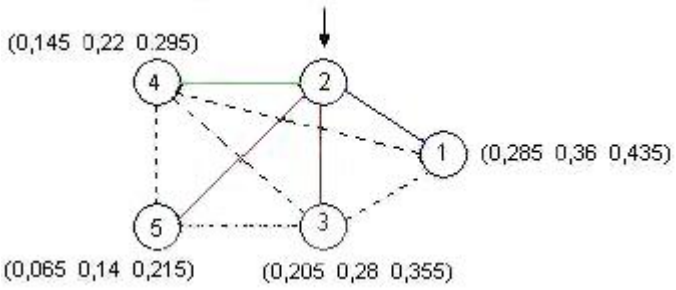

Figura 7.6: Probabilidade dos nós

Associando essas probabilidades a uma função de pertinência (Figura 7.7), temos que: com base na observação da Figura 7.7, percebemos que não existe possibilidade do nó (5) ser maior que a solução encontrada pelo nó ♯1, pois o maior valor possível que o nó (5) pode assumir (limitante superior) é menor que o menor valor que ♯1 pode obter (limitante

inferior). Já para os nós (3) e (4) existem possibilidades de serem maiores que o valor encontrado, pois existe pelo menos um valor em (3) e em (4) que podem ser maiores ou iguais ao valor limitante inferior de ♯1. Podemos dizer então que existem vários *graus de possibilidade* de um nó pertencer à solução do problema. Desta forma, de acordo com um limitante $0 \leq \delta \leq 1$, o nó que obtiver maior grau de pertinência e que o valor da pertinência for maior que esse limitante, será o nó escolhido para o próximo passo. Nesse exemplo seria o nó (3). Caso contrário o próximo passo seria para o nó #1 (nó com maior valor de probabilidade).

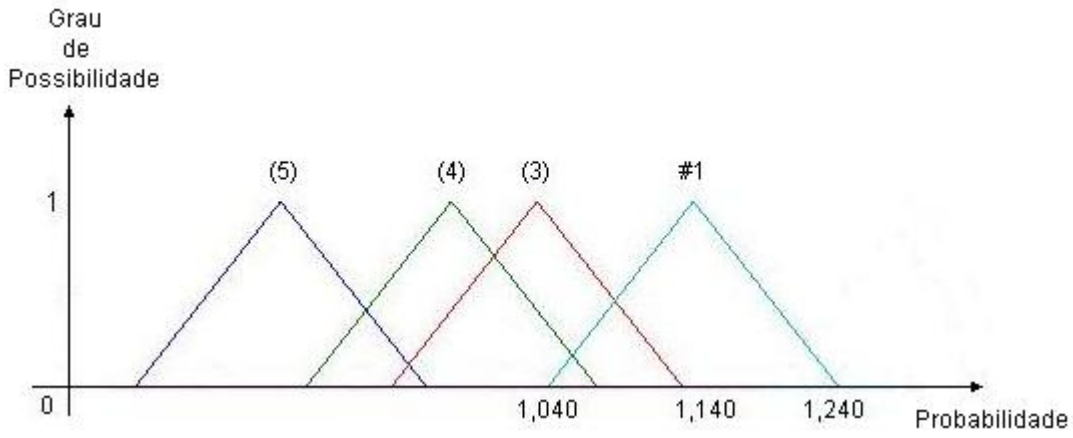

Figura 7.7: Comparação das probabilidades *fuzzy*

7.5 Resultados

Aqui são apresentados alguns resultado obtidos com o algoritmo ACS-*Fuzzy*. Para realização dos testes foram utilizadas duas instâncias retiradas da base de dados do *TSPLIB*.

Para cada grafo foram realizados 10 experimentos, sendo que em cada um deles obtivemos como resultado a solução composta pelo menor caminho e maior Grau de Possibilidade.

Os testes também foram realizados em MatLab® 7.0 com a plataforma: Processador Pentium IV com 3.0Ghz e 1024 de memória RAM.

Os seguintes parâmetros foram ajustados e utilizados nos experimentos:

- Numero de Iterações: 200;

- Número de formigas: 20;

- $\beta = 2, \alpha = 1, q_0 = 0,9$, $\rho = 0,1$ e $\delta = 0,2$.

7.5.1 Instância eil51

Este é um grafo completo que representa todas as conexões entre 51 cidades de Christofides/Eilon. As distâncias entre elas são representadas por números triangulares *fuzzy* do tipo (a_i, a_m, a_s), sendo que a_i representa o limitante inferior com desvio de 51 unidades para menos, a_m o valor modal da distância e a_s o limitante superior com desvio de 51 unidades a mais do valor modal. Os resultados obtidos pelo algoritmo estão exibidos na Tabela 7.1.

Algoritmo	Número de Experimentos	Grau de Possibilidade	Custo *Fuzzy*
ACS-*Fuzzy*	1	0,9706	[378 429 480]
	2	0,9804	[377 428 479]
	3	0,9510	[380 431 482]
	4	0,9902	[376 427 478]
	5	**1,0000**	[**375 426 477**]
	6	0,9608	[379 430 481]
	7	0,9706	[378 429 480]
	8	0,9510	[380 431 482]
	9	0,9706	[378 429 480]
	10	0,9706	[378 429 480]

Tabela 7.1: Resultados obtidos pelo ACS-*Fuzzy* para a instância eil51

Em relação aos resultados pode-se dizer que o algoritmo conseguiu encontrar boas soluções e que algumas delas possuem alto grau de possibilidade de serem melhores que a solução ótima (em negrito).

O tempo médio consumido para a realização de cada rodada foi de 3 minutos.

Na Figura 7.8 encontra-se alguns caminhos obtidos pelo algoritmo ACS-*Fuzzy*.

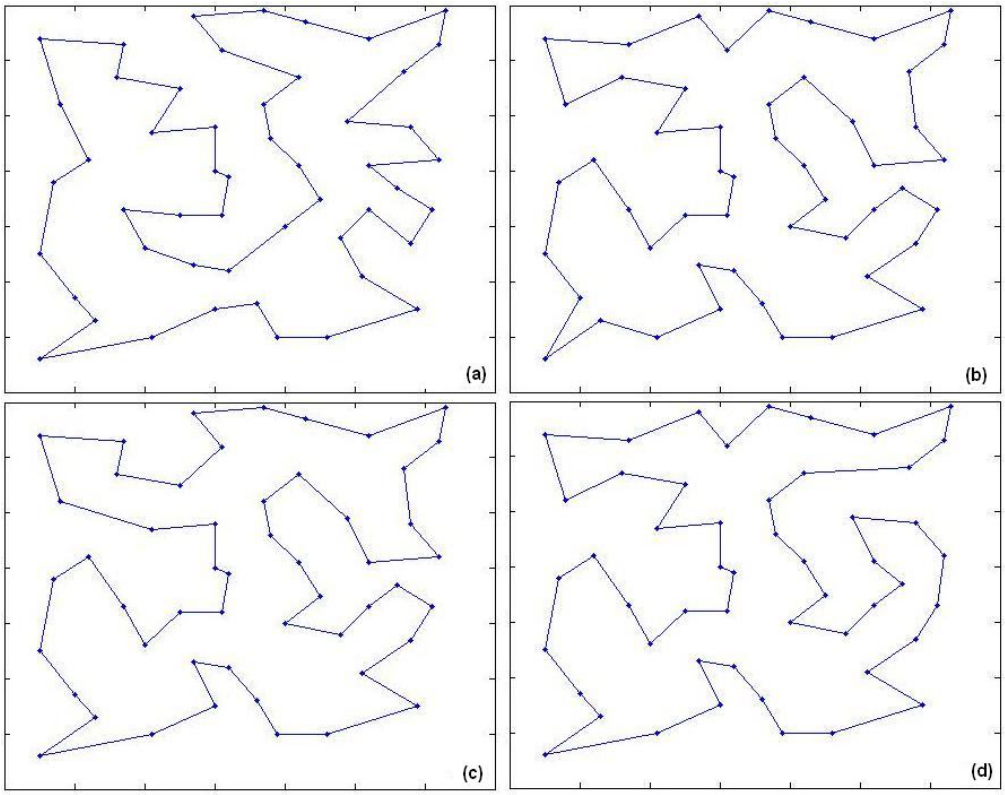

Figura 7.8: (a) Caminho ótimo com grau de possibilidade 1, (b) caminho com grau de possibilidade 0,9902, (c) caminho com grau de 0,9804 e (d) caminho com grau de possibilidade 0,9706 de ser melhor que a solução ótima

7.5.2 Instância eil76

Assim como o grafo eil51, os custos são representados por números triangulares *fuzzy* do tipo (a_i, a_m, a_s), onde o valor modal c_m corresponde a distância de uma cidade a outra e os limitantes inferiores a_i e superiores a_s a um desvio de 76 unidades.

Os resultados obtidos pelo algoritmo ACS-*Fuzzy* para esta instância estão exibidos na Tabela 7.2.

Para este grafo o algoritmo ACS-*Fuzzy* não conseguiu encontrar a solução ótima. Entretanto os resultados encontrados foram satisfatórios pois possuem alto grau de possibilidade de serem melhores que a melhor solução conhecida ([462 538 614]). Os graus de possibilidades encontrados por cada experimento foi obtido comparando os resultados encontrados com a solução ótima ([462 538 614]).

Algoritmo	Número de Experimentos	Grau de Possibilidade	Custo *Fuzzy*
ACS-*Fuzzy*	1	0,9803	[465 541 617]
	2	0,9408	[471 547 623]
	3	0,9276	[473 549 625]
	4	0,9342	[472 548 624]
	5	0,9276	[473 549 625]
	6	0,9605	[468 544 620]
	7	0,9539	[469 545 621]
	8	0,9474	[470 546 622]
	9	0,9539	[469 545 621]
	10	0,9474	[470 546 622]

Tabela 7.2: Resultados obtidos pelo ACS-*Fuzzy* para a instância eil76

O tempo médio consumido para a realização de cada experimento foi de 9 minutos.

Capítulo 8

Considerações Finais e Perspectivas Futuras

Neste trabalho foram apresentadas algumas técnicas heurísticas mais utilizadas na resolução do Problema do Caixeiro Viajante. Este problema consiste em encontrar um caminho mínimo entre duas cidades, e é bastante estudado dentro da teoria de grafos, pois o mesmo possui diversas aplicações práticas. Particularmente, o enfoque deste trabalho se concentrou nas meta-heurísticas de Otimização por Colônia de Formigas e no Algoritmo Genético.

Baseadas em técnicas da computação natural, um algoritmo híbrido que trabalha conjuntamente com as meta-heurísticas de sistema de colônia de formigas e genético foi proposto para resolver o problema do caixeiro viajante, cujo principal objetivo é explorar o espaço de busca de soluções de forma eficiente, visando encontrar uma solução aproximada, de forma a contornar a questão da complexidade.

Enquanto que o algoritmo de sistema de colônia de formigas tem dificuldades em sair de pontos mínimos locais, o algoritmo híbrido consegue explorar com maior facilidade o espaço de busca, obtendo melhores resultados que os algoritmos ACS e o AG individualmente.

Em muitos casos reais, as informações associadas aos problemas, tais como: custo, capacidade, tempo, etc; possuem valores incertos, fazendo com que a teoria clássica dos conjuntos torne-se ineficaz para representá-los. Assim é necessário utilizar uma representação robusta, que pode ser obtida por meio da teoria dos conjuntos *fuzzy*.

Desta forma foi proposta uma extensão do algoritmo de sistema de colônia de formigas

aplicado ao problema do caixeiro viajante em grafos que possuem incertezas associadas às arestas, e a Teoria da Possibilidade foi utilizada para encontrar a solução composta pelos nós que possuem alto Grau de Possibilidade de pertencerem à melhor solução.

Os resultados obtidos são bastantes satisfatórios, comprovando que a implementação de técnicas da computação natural é realmente promissora para resolver problemas de explosão combinatória, como é o caso do problema abordado neste trabalho.

Perspectivas Futuras

Algumas sugestões para trabalhos futuros são as seguintes:

• A implementação de uma busca local no algoritmo genético, logo após a mutação dos indivíduos, pode fazer com que o processo de explotação do espaço de busca seja otimizado, possibilitando obter melhores resultados. Também pode-se utilizar novas técnicas de seleção e novos operadores.

• Aplicar heurísticas de melhorias no algoritmo de sistema de colônia de formigas para o problema do caixeiro viajante, tais como 2-opt, 3-opt ou outra técnica similar.

• A utilização de outras técnicas da computação evolutiva como sistema imunológico artificial, algoritmos co-evolutivos, ou ainda a implementação de algoritmos meméticos podem ser eficientes para resolver o problema estudado.

• Devido aos bons resultados obtidos pela aplicação da meta-heurística híbrida para resolver o problema do caixeiro viajante, pode-se propor a implementação da mesma para resolver outros problemas reais, tais como roteamento de veículos e similares.

• Implementação de algoritmos híbridos para tratar problemas de caixeiro viajante com incertezas.

Apêndice A

GRAFOS DE ALGUMAS INSTÂNCIAS DO *TSPLIB* OBTIDOS COM OS ALGORITMOS ACS, AG E ACS+AG-TSP

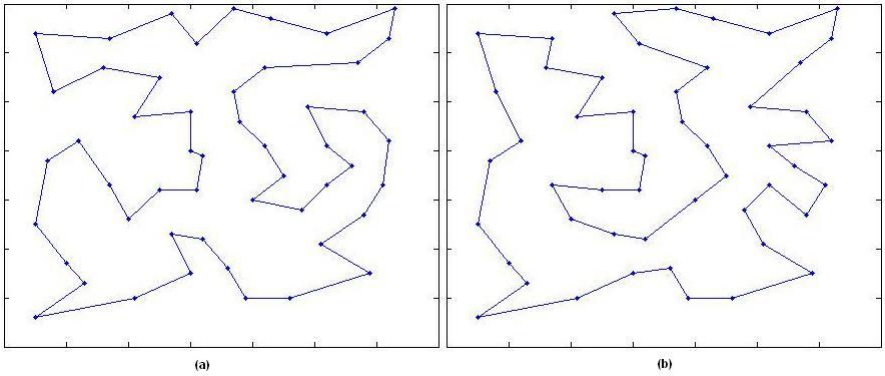

Figura A.1: (a) Solução para o grafo eil51 com o AG com erro de $0,7\%$ e (b) com os algoritmos ACS e ACS+AG-TSP solução ótima encontrada 426

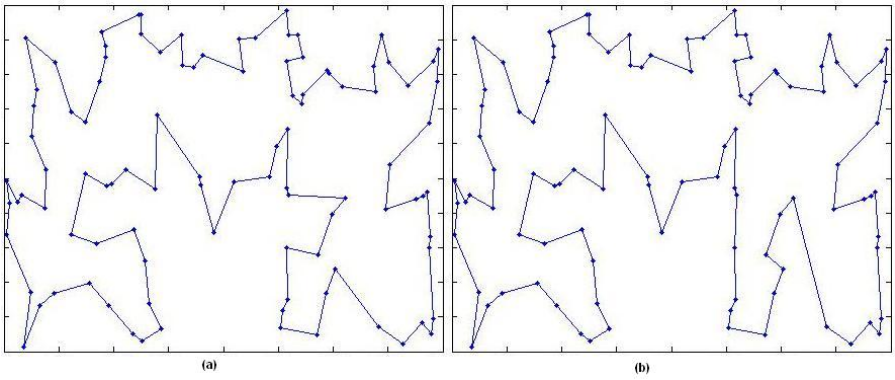

Figura A.2: (a) Solução para o grafo kroA100 com o ACS com erro de $0,17\%$ e (b) com o algoritmo ACS+AG-TSP solução ótima encontrada 21282

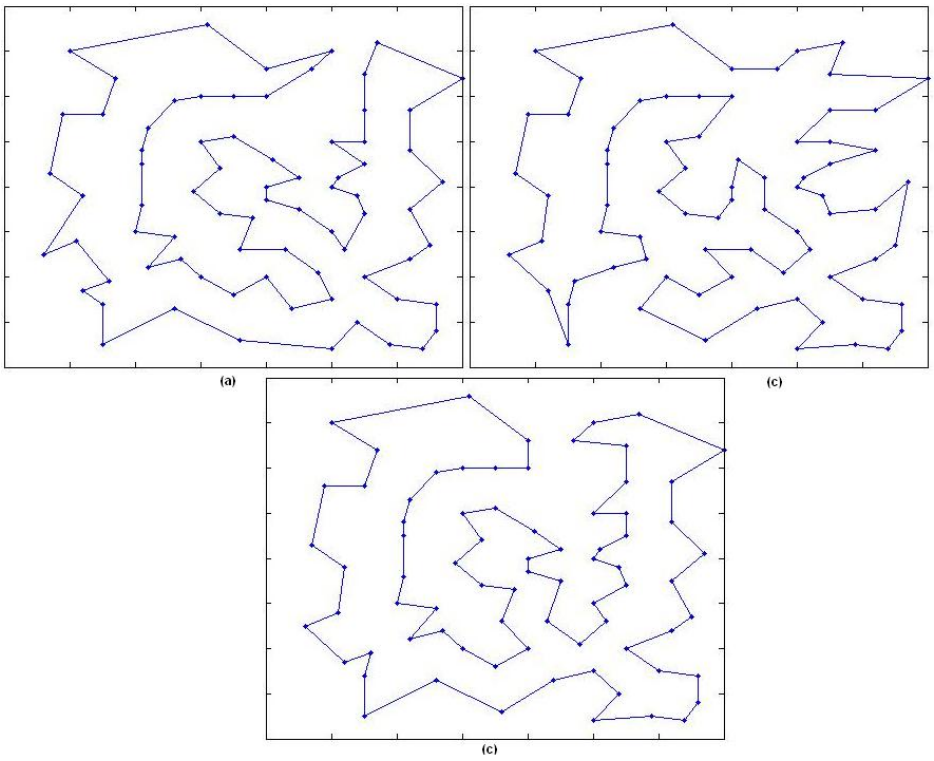

Figura A.3: Solução para o grafo eil76, (a) com o AG (erro de $2,42\%$) , (b) com o ACS (erro de $0,9\%$) e (c) com o algoritmo proposto neste trabalho (solução ótima encontrada 538)

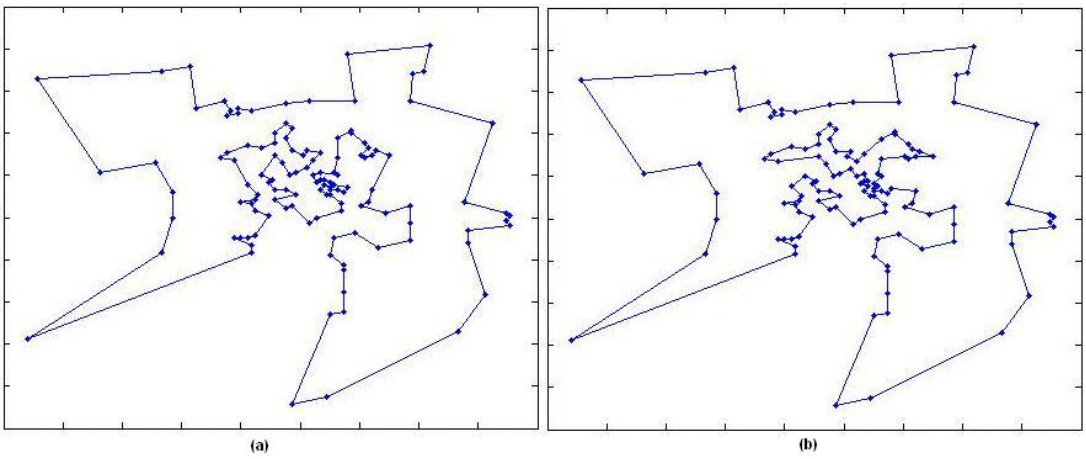

Figura A.4: (a) Solução para o grafo bier127 com o ACS com erro de $0,78\%$ e (b) com o algoritmo híbrido solução ótima encontrada 118282

Referências Bibliográficas

[1] Ahuja, A. K., Magnati, T. L. and Orlin, J. B., *Network flows: Theory, Algorithms, and Applications*. 1 edn, Prentice-Hall, (1993).

[2] Affenzeller, M., Wagner, S., A *Self-Adaptive Model for Selective Pressure Handling Within the of Genetic Algorithms*. In: Computer Aided Systems Theory, Eurocast, Spriger Verlag, **2809**, n.1, (2003) 384–393.

[3] Bonabeau, E., Dorigo, M., and Theraulaz, G., *From Natural to Artificial System - Swarm Intelligence*. Oxford University Press, (1999).

[4] Bonfim T. R., *Escalonamento memético e neuro-memético de tarefas*. PhD thesis, Universidade Estadual de Campinas, (2006).

[5] Bullnheimer, Hartz, R. F., and Strauss, C., *An improved Ant System algorithm for theVehicle Routing Problem*, Annals of Operations Research. To appear, (1999).

[6] Costa, D., and Hertz, A., *Ants can colour graphs*, Journal of the Operational Research Society, **48**, (1997).

[7] Darwin, C. *On The Origin of Species*, 1 st edition,Harward University Press. MA, (1859).

[8] Deneubourg, J. L., S. Aron, S. Goss, and J. M. Pasteels. *The Self-Organizing Exploratory Pattern of the Argentine Ant*. J. Insect Behavior 3 (1990), 159- 168.

[9] Dorigo, M., and Gambardella, L. M., *Ant Colonies for the Traveling Salesman Problem*, BioSystems **43**, n.2, (1997), 73–81.

[10] Dorigo, M., Di Caro, G., *The Ant Colony Optimization Meta-Heuristic*. In Corne, D., Dorigo, M., and Glover F. (Eds.) New Ideas in Optimization, New York: McGraw-Hill, (1999), 11–32.

[11] Dorigo, M. *Optimization, Learning and Natural Algorithms*. Ph.D.Thesis, Politecnico di Milano, Italy, (1992).

[12] Dorigo, M., and Stützle, T., *ACO Algorithms for the Traveling Salesman Problem*, In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, Evolutionary Algorithms in Engineering and Computer Science, Wiley, (1999).

[13] Dorigo, M., and Stützle, T., *Ant Colony Optimization*. Cambridge, MA: MIT Press, (2004).

[14] Dubois, D. and Prade, P., *Fuzzy Sets and Systems:Theory and Applications.*Academic Press, (1980).

[15] Edmonds, J., *Paths, Trees, and Flowers*. Canadian Journal of Mathematics, n.17, (1965), 449–467.

[16] Gambardella, L. M., and Dorigo, M., *Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS J Comput, **42**, n.12, (2000), 237–255.

[17] Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA: W. H. Freeman (1979).

[18] Goldbarg, M. C. end Luna, H. P. *Otimização Combinatória e Programação Linear: Modelos e Algoritmos*. 1 edn, Campus, RJ, Brasil, (2000).

[19] Holland, J. H., *Adaptation in Natural and Artificial Systems*, 2 edn, MIT Press, MA, USA, (1992).

[20] Homaifar, S. Guan, and G. E. Liepins., *A New Approach to the Traveling Salesman by Genetic Algorithms*. In: Proceeding of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, (1993), 460-466.

[21] Maniezzo, V.,*Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem*. INFORMS J Comput, **11**, n.4, (1999), 358–369.

[22] Melamed, I. I., Sergeev, S. I., Sigal, I. Kh. *The Traveling Salesman Problem*. Surveys. [S.l.]: Plenum Publishing Corporation, (1990), 1147–1173.

[23] Merz, P., Freisleber, B., *Genetic Local Search for the TSP: New results*. In: IEEE International Conference on Evolutionary, IEEE press, Indaianapolis, (1997), 159-164.

[24] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, (1996).

[25] Okada,S., *Interactions among paths in fuzzy shortest path problems*, in 'Joint 9th IFSA World Congress and 20th NAFIPS International Conference', Vol.1, Vancouver, BC, Canada, (2001), 41–46.

[26] Onwubolu, G. C., Clerc, M., *Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization*. International Journal of Production Research **42**, n. 3, (2004), 473–491.

[27] Pedrycz, W., and Gomide, F., *An Introduction to Fuzzy Sets: Analysis and Design*. A Bradford Book, (1998).

[28] Pilat, M. L., and White, T., *Using Genetic Algorithms to Optimize ACS-TSP*. Proceedings of the Third International Workshop on Ant Algorithms, ANTS, Springer-Verlag, **2463**, (2002), 282–287.

[29] SARI, S. C., SHERALI, H.D., BHOOTRA, A., *New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints*. Operations Research Letters, **33**, n. 1, (2005), 62–70.

[30] Stützle, H., Hoss, H., *The Max-Min Ant System and Local Search for the Travelling Salesman Problem*. Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97), (1997), 309–314.

[31] Stützle, H., Hoss, H., *Max-Min Ant System*. Future Generation Computer Systems, **16**, n. 8, (2000), 889–914.

[32] Takahaski, M. T., *Contribuiçõesao Estudo de Grafos Fuzzy: Teoria e Algoritmos*. PhD thesis, Universidade Estadual de Campinas, (2004).

[33] Zadeh, L., *Fuzzy sets. Information and Control*, **8**, (1965), 338–353.

[34] Zadeh,L.A.(1978),*Fuzzy sets as a basis for a theory of possibility*, Fuzzy Sets and Systems, negr1, V.1, 3–28.