UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

MATHEUS GOMES ARAUJO

APLICAÇÃO DA ABORDAGEM DE COLÔNIA DE FORMIGAS PARA SOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE

FEIRA DE SANTANA

2012

UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

MATHEUS GOMES ARAUJO

APLICAÇÃO DA ABORDAGEM DE COLÔNIA DE FORMIGAS PARA SOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação da Universidade Estadual de Feira de Santana para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Matheus Giovanni Pires.

FEIRA DE SANTANA

2012

FOLHA DE APROVAÇÃO

Porque Dele, e por meio Dele, e para Ele são todas as coisas.
A Ele, pois, a glória eternamente. Amém!
Romanos 11:36

AGRADECIMENTOS

Ao Senhor dos exércitos por estar comigo sempre que eu preciso em todas as situações.
Ao professor Matheus Giovanni pelo compromisso e orientação durante todo o processo de construção desse trabalho, em que este procurou sempre e da melhor forma passar o seu conhecimento guiando a orientação até que ela chegasse ao resultado final.
A todos os outros professores que me ajudaram neste processo de aprendizagem, em especial aos professores Roberto Bittencourt, Thiago D'Martin Maia, Wagner Oliveira, Antonio Augusto, Ângelo Duarte, Fabiana Cristina, Márcia Prado, Gabriela Rezende, Michele Fúlvia, Edgar Silva, Anfranserai, professores do curso de Engenharia de Computação da UEFS.
Aos meus pais e ao meu irmão, por tudo que eles fizeram por mim durante todo o curso de Engenharia de Computação.
A todos os amigos da Igreja que sempre me deram apoio nessa caminhada, em especial Jaira Alves pelos conselhos nas horas difíceis.
Aos amigos e colegas do curso de Engenharia de Computação que me ajudaram nesta caminhada, em especial a Larissa Soares que foi minha amiga e se tornou no decorrer no curso como uma irmã, além de outros amigos como Elisson, Vinicius, Wesley, Wilsom, Diego, Linton, Leo, André, Pedro, Anderson, Daniel dos Anjos, Igo, Laue, Jomas, Carlos, Nathalia, Lorena, Raquel, Fernanda, Jairo, entre outros que lutaram juntamente comigo para chegar ao fim desta caminhada.

RESUMO

O termo "planejamento de rotas" está relacionado a um grande conjunto de problemas de fundamental importância para as diferentes áreas do conhecimento, em especial para a área de logística de transportes, seja ela relacionada ao transporte de mercadorias, pessoas ou informação. O objetivo de tais problemas é a determinação de rotas a serem seguidas que melhor representem a relação custo-benefício de determinada atividade. Este objetivo, em geral, é complexo e difícil de ser alcançado, devido às restrições operacionais que dependem de fatores como, por exemplo, a natureza e características dos itens a serem transportados ou da qualidade com que o serviço deve ser executado. Além disso, os custos envolvidos costumam ser elevados e sensíveis às diferentes variáveis associadas ao problema. Um problema específico de planejamento de rotas é o Problema do Caixeiro Viajante, que visa encontrar a menor rota entre um conjunto de cidades, partindo de uma cidade origem, visitando cada cidade uma única vez e retornando à cidade inicial. A maioria dos algoritmos atualmente disponíveis para solucionar tal problema não consegue encontrar, em tempos computacionais aceitáveis, a solução ótima para problemas de tamanho razoável. Neste contexto, este trabalho tem por objetivo avaliar o desempenho de uma abordagem baseada em Colônia de Formigas quando aplicada ao Problema do Caixeiro Viajante.

PALAVRAS-CHAVE: Planejamento de Rotas. Problema do Caixeiro Viajante. Colônia de Formigas.

ABSTRACT

The term "path planning" is related to many important areas, such as, logistics, transport of foods, people and information. The goal of such problems is find the best cost-benefit ratio or the closer it. So, this objective, in general, is complex and difficult to find due to operational constraints and the variables of the problem. A specific problem of path planning is the Traveling Salesman Problem, which aims to find the shortest path between a set of cities. The path should starting from a source city, visiting each city only once, and should returning to source city. The most of currently available algorithms for solving the Traveling Salesman Problem can not find an path in acceptable computational time. In this context, this work aims to evaluate the performance of an approach based on Ant Colony applied int the resolution the Traveling Salesman Problem.

KEYWORDS: Path Planning. Traveling Salesman Problem. Ant Colony.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um problema de otimização combinatória tem por objetivo determinar valores a um conjunto de variáveis de decisão, de tal modo que uma função dessas variáveis (função-objetivo) seja otimizada na presença de um conjunto de restrições. Problemas práticos de engenharia elétrica, engenharia aeroespacial, economia e pesquisa operacional constituem-se em problemas de otimização combinatória e frequentemente estão relacionados ao planejamento de rotas (ou caminhos) a serem percorridos para executar uma determinada tarefa.

Tal problema de planejamento de rotas consiste em determinar o melhor caminho, dentre os possíveis, a ser percorrido entre uma origem e um destino, como acontece com o roteamento de pacotes em redes de computadores, o roteamento de veículos em sistemas de transporte, o planejamento de produção em sistemas de manufatura, o roteamento de informações em redes de telecomunicações, e o planejamento de rotas para robôs, por exemplo. O interesse dos pesquisadores por problemas desta natureza objetiva repensar processos para trabalhar de forma mais eficiente, evitando o desperdício, seja de tempo ou de recursos materiais.

O Problema do Caixeiro Viajante (PCV) é um clássico deste tipo de problema e consiste na procura de uma rota que possua a menor distância, começando em uma cidade qualquer, entre várias, visitando cada cidade precisamente uma vez e regressando à cidade inicial (NILSSON, 1982).

Na literatura podem ser encontradas diversas abordagens para a solução do PCV, as quais utilizam algoritmos determinísticos, heurísticos e baseados em inteligência artificial (JOHNSON; MCGEOCH, 1997), (JOHNSON; MCGEOCH, 2002), (SAADATMAND-TARZJAN et al, 2007). A abordagem utilizada neste trabalho, baseada em Colônias de Formigas, que foi apresentada inicialmente por Dorigo et al. (1996), propõe um paradigma computacional para a resolução de problemas combinatórios baseado no comportamento das formigas. Este comportamento consiste no deslocamento de uma formiga entre a colônia e a fonte de alimento. A partir de experimentos realizados com formigas reais, observou-se que elas eram capazes de encontrar o menor caminho do formigueiro para uma fonte de alimento sem a necessidade de utilizar dados visuais.

A apresentação das etapas de desenvolvimento deste trabalho está organizada como descrito a seguir. No Capítulo 2 são apresentados os conceitos relacionados a cada um dos assuntos abordados, definindo inicialmente o termo otimização, descrevendo o que representa um problema de otimização combinatória, e apresentando com maiores detalhes o problema de planejamento de rotas, especificamente o PCV. Neste capítulo também são apresentadas as definições envolvendo a abordagem de Colônia de Formigas, a qual será aplicada na solução do Problema do Caixeiro Viajante.

O Capítulo 3 trata da utilização da abordagem baseada em Colônia de Formigas para a solução do PCV, modelando o problema para duas versões do algoritmo, o *Ant System* e o *Ant Colony System*.

No Capítulo 4 são mostrados três experimentos para avaliar o desempenho dos algoritmos *Ant System* e *Ant Colony System*, considerando diferentes tamanhos (quantidade de cidades) para o PCV. Os resultados são apresentados e discutidos.

Por fim, o Capítulo 5 apresenta as conclusões e discute possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem por objetivo apresentar os conceitos básicos para a compreensão de um problema de otimização combinatória, e também a abordagem de Colônia de Formigas aplicada na solução do problema do Caixeiro Viajante, o qual é o objeto de estudo deste trabalho.

2.1 Otimização Combinatória

Otimização é o processo de buscar a melhor solução dentre um conjunto de soluções disponíveis para um problema. Tipicamente, um problema de otimização possui três elementos constituintes: variáveis de decisão (parâmetros cujos valores definem uma solução para o problema), função objetivo (uma função das variáveis de decisão a ser minimizada ou maximizada) e restrições (um conjunto de funções que define o espaço de soluções aceitáveis). Como exemplo da utilização destes elementos, considera-se um sistema de produção. As variáveis de decisão podem representar as quantidades produzidas de determinados objetos; a função objetivo pode simbolizar o interesse em minimizar os custos na produção destes objetos; e as restrições podem estar relacionadas às limitações operacionais do processo de produção ou até mesmo às limitações físicas e tecnológicas.

Existe uma classe especial de problemas de otimização, comum nas áreas de pesquisa operacional e de engenharia, a qual possui como tarefa encontrar uma permutação ótima de algumas variáveis de decisão. Os problemas pertencentes a esta classe são conhecidos como problemas de otimização combinatória. Formalmente, um problema de otimização combinatória é definido por meio de um conjunto finito $N=\{1,...,n\}$, com pesos c_j associados a cada $j \in N$, e um conjunto F formado por subconjuntos viáveis de N. Deseja-se determinar elementos de F, tais que o somatório dos pesos associados sejam ótimos. Uma solução é viável quando os valores atribuídos às variáveis não violam nenhuma restrição. O conjunto F é também chamado de espaço de busca de soluções (WOLSEY, 1998).

Dentre os problemas de otimização combinatória existe o problema de planejamento de rotas, o qual consiste em definir o melhor caminho entre dois pontos de

um mapa, que deve ser percorrido por um determinado agente para cumprir uma tarefa. Dependendo do tipo da aplicação, o melhor caminho pode significar o menor ou o mais rápido, ou até mesmo uma combinação de vários critérios.

Considerando o problema do Caixeiro Viajante, um problema clássico de planejamento de rotas e objeto de estudo deste trabalho, o objetivo consiste em encontrar o caminho de menor custo (distância) que se inicie e termine em uma cidade, passando por todas as demais cidades, sem repeti-las.

Este problema pode ser facilmente modelado em um grafo, conforme ilustrado na Figura 1, onde cada nó do grafo representa uma cidade e os pesos das arestas representam a distância entre as cidades. Formalmente, o problema do caixeiro viajante pode ser definido como um grafo (N,E), onde N é o conjunto de cidades e E o conjunto de arestas entre as cidades. A distância entre uma cidade i e j é chamada de d_{ij}.

Para maiores detalhes sobre grafos ver (ZIVIANI, 2005).

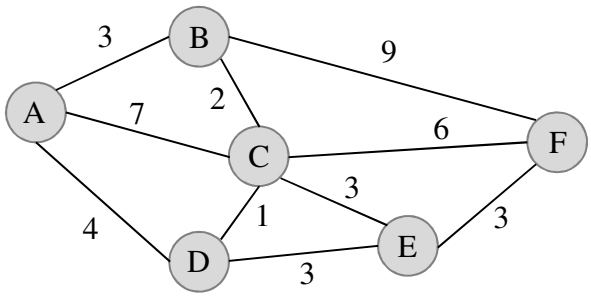

Figura 1: Exemplo de modelagem do problema do caixeiro viajante com 6 cidades. Fonte: Próprio Autor.

Uma característica comum dos problemas de otimização combinatória é o fato de que o conjunto de soluções viáveis, embora seja finito, pode ser muito grande. Tomando como exemplo o problema do Caixeiro Viajante, à medida que cresce o número de cidades, a complexidade do problema aumenta exponencialmente. Para encontrar o número R(n) de rotas para o caso de n cidades, basta fazer um raciocínio combinatório simples. No caso de 4 cidades, a primeira e última posição coincidem e são fixas, de modo que elas não afetam o cálculo; na segunda posição, coloca-se qualquer uma das três cidades restantes e, uma vez escolhida uma delas, escolhe-se qualquer uma das duas restantes na terceira posição; na quarta posição não há escolha, pois restou apenas uma cidade. Consequentemente, o número de rotas é 3x2x1=6. Generalizando para o caso de n cidades, como a primeira é fixa, o número total de escolhas é (n-1)x(n-2)x...x2x1, ou seja, R(n) = (n-1)!. Assim, se fossem atribuídos a n os valores 5, 10 e 25, ter-se-ia R(n) igual a 24, 362.880 e $6{,}2x10^{23}$ rotas, respectivamente.

A partir destes valores é possível perceber o aumento exponencial das alternativas a serem avaliadas no problema do Caixeiro Viajante, e consequentemente, a exigência de mais recursos computacionais, tais como memória e tempo de processamento.

Situações como esta, onde existe uma grande variedade de possíveis soluções, inviabilizam a utilização de estratégias de força bruta (testar todos os valores de solução possíveis), sendo necessário o uso de métodos específicos de otimização, por não haver uma solução simples e diretamente calculável para o problema.

2.2 Métodos de Solução para Problemas de Otimização Combinatória

Para algumas classes de problemas de otimização combinatória existem métodos exatos de resolução, chamados de Programação Matemática, que visam criar e solucionar modelos quantitativos que podem ser expressos matematicamente e que representam algum processo. Outras classes levaram à necessidade de métodos não-exatos, tais como os Heurísticos e os baseados em Inteligência Artificial, uma vez que sua resolução exata é computacionalmente intratável (ZIVIANI, 2005). Além disso, muitas aplicações não exigem uma solução exata, tornando aceitável o uso de métodos de solução de problemas que encontrem soluções aproximadas, mas com menor custo computacional.

Os métodos heurísticos utilizam informações que tem por objetivo indicar a direção mais promissora para alcançar o alvo de um determinado problema (POOLE, et al, 1998). Como estas informações são particulares para cada tipo de problema, é preciso ter conhecimento específico do mesmo, sendo que o sucesso da busca pela solução está fortemente baseado na experiência de um especialista na área do problema.

Já os métodos baseados em inteligência artificial visam representar um comportamento inteligente por meio de um programa de computador e oferecem alternativas para solucionar problemas combinatórios de forma eficiente, devido à possibilidade de percorrer espaços de busca não-lineares e extensos. Dentre estes métodos podem-se destacar os Algoritmos Genéticos e a Colônia de Formigas, sendo que apenas este último será discutido por ter sido utilizado neste trabalho.

2.3 Otimização por Colônia de Formigas

A abordagem baseada em Colônia de Formigas foi apresentada por Dorigo et al. (1996), propondo um paradigma computacional para a solução de problemas combinatórios baseado no comportamento encontrado ao observar formigas se movendo entre a colônia e a fonte de alimento. A fundamentação teórica do algoritmo descrito por Dorigo et al. (1996), se encontra na capacidade das formigas de gerar uma "trilha" entre o ninho e uma fonte de comida.

As formigas reais são capazes de encontrar o caminho mais curto do formigueiro para uma fonte de alimento sem a utilização de dados visuais. Enquanto caminham, as formigas depositam no solo uma substância denominada feromônio (designação genérica de substâncias secretadas pelas formigas que servem de meio de comunicação entre elas), e tem seu deslocamento baseado em trilhas de feromônios previamente depositados por outras formigas. Estas trilhas podem ser observadas por outras formigas e motivá-las em seguir determinado caminho, isto é, um movimento aleatório das formigas segue com maior probabilidade uma trilha de feromônio. Esta é uma maneira de como as trilhas são reforçadas e mais e mais formigas tendem a seguir aquela trilha (NETO; COELHO, 2004).

A dinâmica do comportamento das formigas é ilustrada na Figura 2 (NETO; COELHO, 2005). Com o objetivo de ir do ponto A (ninho) ao ponto B (fonte de alimento), as formigas inicialmente vão tanto para o caminho da esquerda quanto para o caminho da direita (Figura 2 (a)). Posteriormente, com a evaporação do feromônio, a trilha da esquerda se torna mais reforçada, pois um número maior de formigas passa por ele em um menor espaço de tempo (Figura 2 (b)). Após algum tempo, o caminho da esquerda conterá uma fração dominante de feromônio, o que levará as formigas a escolhê-lo com maior frequência (Figura 2 (c)).

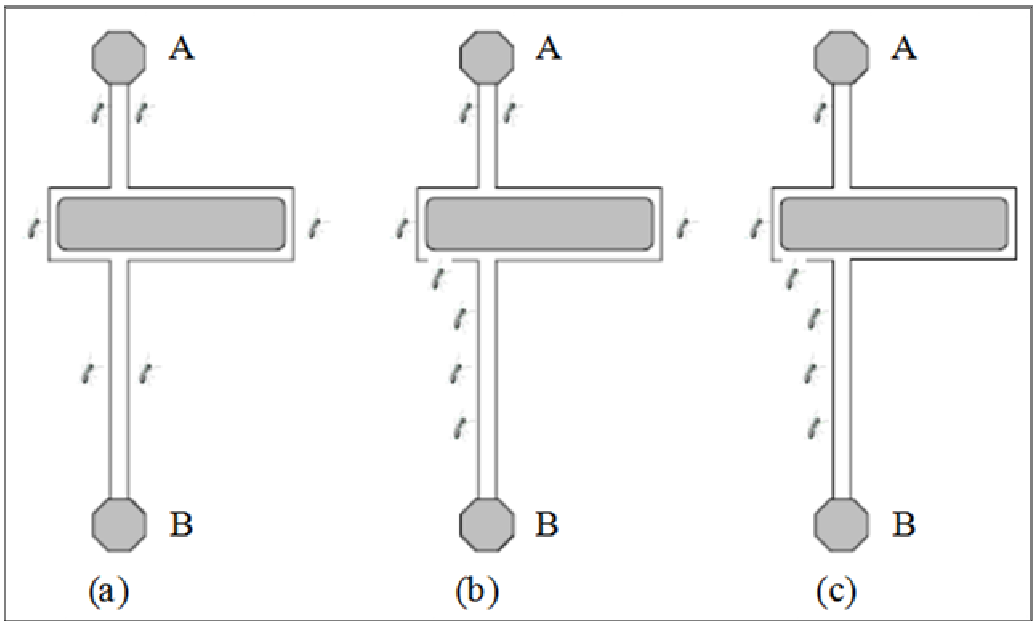

Figura 2: A descoberta do menor caminho através do feromônio. Fonte: (NETO; COELHO, 2005).

As formigas não seguem as trilhas perfeitamente, pois elas podem se perder da trilha ao longo do percurso. Este tipo de comportamento é importante para que seja possível a descoberta de outras fontes de alimento, assim, essa “aleatoriedade” possui um papel significativo no processo.

3 METODOLOGIA DO TRABALHO

Este capítulo descreve a modelagem do Problema do Caixeiro Viajante (PCV) através de duas abordagens do algoritmo de Colônia de Formigas, o *Ant System* e o *Ant Colony System*, e discute algumas considerações e parâmetros de ambas.

3.1 Modelagem do Problema

Conforme já descrito na Seção 2.1, o problema de planejamento de rotas consiste em definir o melhor caminho entre dois pontos de um mapa. Dependendo do tipo da aplicação, o melhor caminho pode significar o menor ou o mais rápido, ou até mesmo uma combinação de vários critérios. No caso do problema do Caixeiro Viajante, o objetivo consiste em encontrar o menor caminho entre duas cidades, em que a origem é igual ao destino, passando por todas as demais cidades uma única vez. Este problema pode ser facilmente modelado por um grafo, conforme ilustrado na Figura 1.

A utilização da abordagem baseada em Colônia de Formigas para a solução do problema do Caixeiro Viajante ocorre da seguinte forma. Inicialmente, cada formiga é colocada em uma cidade diferente, pois, cada uma delas irá construir uma solução para o problema, visitando uma cidade de cada vez, a partir de uma cidade *i*, como ilustrado na Figura 3.

Cada formiga irá calcular uma probabilidade de transição para escolher a próxima cidade *j* que deve ir. Considerando a hipótese de que a formiga saia da cidade 4 para a cidade 1 (ver Figura 3), a formiga depositará certa quantidade de feromônio nessa aresta percorrida. Assim, quanto mais formigas passarem por esta aresta, ou seja, indo da cidade 4 para a cidade 1, maior será a quantidade de feromônio depositado, e isto irá atrair mais formigas a passarem por ela.

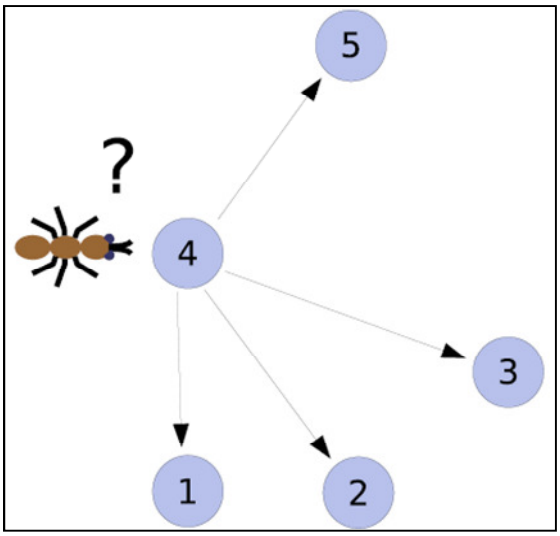

Figura 3: Formiga construindo um caminho. Fonte: (LACERDA, 2010).

3.2 Resolução do problema do Caixeiro Viajante pelo algoritmo Ant System

O *Ant System* (*AS*) foi o primeiro exemplo de algoritmo proposto por Dorigo et al. (1999) baseado no comportamento de uma colônia de formigas. A resolução do problema do Caixeiro Viajante pelo *AS* ocorre da seguinte maneira.

Na fase de construção da solução, cada formiga escolhe para qual cidade ir com uma probabilidade que é uma função dada pelo inverso da distância da cidade e pela quantidade de feromônio presente na aresta que se conecta a essa cidade. Para não permitir passos inválidos, a formiga tem uma memória de quais cidades ela já visitou. Assim, enquanto ela constrói uma solução, ela é forçada a não visitar uma mesma cidade duas vezes até que a solução esteja completa. Quando a formiga termina um caminho completo, ela deposita certa quantidade de feromônio em cada aresta (i,j) que ela visitou (DORIGO et al, 1996).

No *AS*, não há depósito de feromônio durante a fase de construção da solução. Ao dar um passo, a formiga calcula a probabilidade de todas as cidades que ela ainda não visitou e escolhe ir para aquela de maior probabilidade. Essa probabilidade é calculada pela Equação (1):

$$p_{ij}^{k} = \left\{ \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \right. \quad (1)$$

onde τ_{ij} é a quantidade de feromônio presente na aresta *ij* e η_{ij} é o inverso da distância d_{ij}, que representa a informação heurística do PCV usada pelo *AS*. α e β são parâmetros para indicar a importância do feromônio e da informação heurística, respectivamente. N_i^k é o conjunto das cidades ainda não visitadas pela formiga *k*.

O *AS* não realiza nenhuma ação global, nem busca local. Assim, a próxima fase, após a construção da solução, é a atualização do feromônio, que envolve tanto o incremento do feromônio quanto a sua evaporação, e é realizado por todas as formigas. A atualização é definida pela Equação (2):

$$\tau_{ij}(t+1) = (1-\sigma)\,\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

onde σ é um parâmetro, definido entre 0 e 1, para regular a taxa de evaporação do feromônio, *m* o número de formigas e $\Delta\tau_{ij}^{k}$ é definida pela Equação (3):

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{1}{L^k}, \text{ se a aresta ij faz parte do caminho k} \\ 0, \textit{ caso contrário} \end{cases} \tag{3}$$

onde L^k é o custo do caminho percorrido pela formiga *k*.

No anexo A é mostrado o pseudocódigo do *AS* para uma melhor compreensão do algoritmo.

3.3 Resolução do problema do Caixeiro Viajante pelo algoritmo Ant Colony System

O algoritmo *Ant Colony System* (*ACS*) foi proposto, como uma melhoria do *AS*, por Dorigo e Gambardella (1997). Em relação ao *AS*, o *ACS* possui um parâmetro adicional q_0, $(0 \le q_0 \le 1)$, que define a probabilidade de ser utilizada uma nova expressão para a decisão de qual aresta cada formiga deve seguir na fase de construção. A cada iteração é sorteado um número aleatório e caso este seja menor que q_0, a regra de decisão utilizada é $p_{ij}^k = \tau_{ij}\eta_{ij}^{\beta}$, caso contrário, a regra de decisão utilizada é a mesma usada no algoritmo *AS,* definida pela Equação (1).

Com valores de q_0 mais próximos de um, tem-se uma priorização da intensificação, enquanto que com valores de q_0 mais próximos de zero, prioriza-se a diversificação, por conta da regra de decisão do *AS* ser utilizada com maior frequência.

A intensificação é quando as formigas tendem a buscar o melhor caminho, enquanto na diversificação, as formigas tendem a buscar caminhos diferentes.

Outra diferença do *ACS* com relação ao *AS* é que, a cada iteração da fase de construção, as formigas atualizam o feromônio da aresta pela qual acabaram de atravessar, de acordo com a Equação (4):

$$\tau_{ij}(t+1) = (1-\sigma).\tau_{ij\,(t)} + \sigma.\tau_0 \tag{4}$$

onde σ é um parâmetro definido entre 0 e 1, τ_0 é o valor inicial dos feromônios das arestas. A esta ação dá-se o nome de atualização local de feromônio, tendo sido inserida no algoritmo para contrabalancear a regra de decisão gulosa desta abordagem.

Além da atualização local, o *ACS* compreende uma atualização global de feromônio, a qual é executada a cada iteração do algoritmo. Esta atualização é feita somente pela formiga que construiu a melhor solução até o momento. Todas as arestas do melhor caminho atual são atualizadas segundo a Equação (5):

$$\tau_{ij}(t+1) = (1-\sigma).\tau_{ij(t)} + \sigma.\Delta\tau_{ij}^{melhor} \tag{5}$$

Segundo Dorigo et al. (1996), outra diferença entre o *AS* e *ACS* é a lista candidata. A lista candidata é uma estrutura de dados usada pelo *ACS* para ajudar a resolver grandes problemas relacionados ao caixeiro viajante. A lista candidata é uma lista de cidades preferenciais e ainda não visitadas de uma determinada formiga. Ao invés da formiga analisar todas as possíveis cidades para visitar, dá-se prioridade às cidades que estão na lista candidata. Somente quando todas as cidades desta lista forem visitadas, as demais cidades serão analisadas. As cidades adicionadas nesta lista são ordenadas pela distância crescente e ela é percorrida sequencialmente.

No anexo B é mostrado o pseudocódigo do *ACS* para uma melhor compreensão do algoritmo.

3.4 Comparativo dos algoritmos *AS* e *ACS*

A partir do entendimento dos algoritmos apresentados, nota-se que a principal diferença entre o *AS* e o *ACS* é como a atualização do feromônio nas arestas é realizada. A Tabela 1 mostra a comparação entre os algoritmos.

Tabela 1: Comparação dos Algoritmos

Algoritmo	Expressão	Atualização dos Feromônios
AS	$\tau_{ij} = (1-\sigma)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$	Por todas as formigas ao realizar o caminho.
ACS	$\tau_{ij} = (1-\sigma).\tau_{ij} + \sigma.\tau_0$	Por todas as formigas em cada passo da construção do percurso.
	$\tau_{ij} = (1-\sigma).\tau_{ij} + \sigma.\Delta\tau_{ij}^{melhor}$	Apenas pela melhor formiga até agora ao completar o caminho.

3.5 Análise dos parâmetros dos algoritmos Ant System e Ant Colony System

Neste trabalho foram implementadas duas versões de algoritmos para a solução do problema do Caixeiro Viajante baseado em Colônia de Formigas – o *Ant System* e o *Ant Colony System*. O *AS* foi o primeiro algoritmo proposto por Dorigo et al. (1996), e o *ACS* foi proposto como uma melhoria do AS por Dorigo e Gambardella (1997). Embora o desempenho do *AS* não seja suficiente para competir com os melhores algoritmos propostos para resolver o PCV, ele é usado como ponto de partida para a formulação de outros algoritmos de otimização por colônia de formigas, pois este algoritmo produziu boas soluções apenas para conjuntos de dados entre 30 e 70 cidades (DORIGO et al., 1996).

Como se pode observar na Seção 3.2, há quatro parâmetros que devem ser ajustados para se obter um bom desempenho com o *AS*. Um parâmetro para controlar a importância do feromônio, outro para a informação heurística, um terceiro para regular a taxa de evaporação e um quarto para indicar o tamanho da população de formigas. O ajuste correto desses parâmetros é fundamental para se obter um bom equilíbrio entre intensificação e diversificação com o *AS*.

O parâmetro σ, por exemplo, deve ser ajustado para um valor menor que 1, pois, do contrário, haverá uma acumulação ilimitada do feromônio, prejudicando a diversificação (DORIGO et al., 1996). Dorigo et al. (1996) sugere um valor em torno de 0,5. O parâmetro α deve ser maior que 0. Quando o seu valor é alto, isso significa que o feromônio é muito importante e as formigas tendem a escolher arestas escolhidas por outras formigas no passado, o que tem um efeito de intensificação. Se o valor for baixo,

a construção se comporta de maneira semelhante a uma construção gulosa aleatória (DORIGO et al., 1996). No entanto, valores altos demais podem provocar a estagnação precoce do algoritmo. Dorigo et al. (1996) propõe que α deve ficar em torno de 1. Um valor muito baixo de β provocaria estagnação precoce do algoritmo e um valor alto demais o aproxima de uma construção gulosa. O ajuste deste parâmetro é importante para se obter uma boa diversificação. Dorigo et al. (1996) propõe que fique β em torno de 5. Geralmente, m = n, isto é, a quantidade de formigas igual à quantidade de cidades. Por fim, Dorigo et al. (1996) propõe inicializar o feromônio de todas as arestas por $\frac{1}{n.L_{nn}}$, onde L_{nn} é o custo de uma construção gulosa.

Com relação aos melhores parâmetros para o *ACS*, Dorigo et al. (1996) propõe que m seja igual a 10, baseado em observações de experimentos (DORIGO; GAMBARDELLA, 1997). No entanto, ao realizar experimentos para o PCV com este valor, pode-se observar que os resultados não foram satisfatórios. Sendo assim, foi necessária a realização de testes variando o valor de m. O melhor valor para este parâmetro, a partir dos experimentos realizados, foi igual a 20.

Já o valor σ deve ser igual a 0,1 e β deve ser igual a 2, fazendo com que a construção gulosa seja priorizada. Ainda, para favorecer a intensificação, Dorigo et al. (1996) sugere que q_0 deve ser igual a 0,9. Também foi sugerido que a inicialização dos feromônios das arestas seja igual a $\frac{1}{n.L_{nn}}$. Por fim, para a lista candidata L, o valor proposto por Dorigo et al. (1996) foi de 15 cidades.

Considerando os estudos encontrados na literatura sobre a definição dos parâmetros do *AS* e do *ACS*, e também a partir de testes preliminares, os valores dos parâmetros utilizados neste trabalho para o *AS* e o *ACS* estão descritos nas Tabela 2 e Tabela 3, respectivamente.

Tabela 2: Valores dos parâmetros para o AS.

Parâmetro	*Valor*
A	1
B	5
σ	0,5
N	Número de cidades

Tabela 3: Valores dos parâmetros para o ACS.

Parâmetro	*Valor*
B	2
σ	0,1
q_0	0,9
M	20
τ_0	$(n*L_{nn})^{-1}$
L	15

4 ANÁLISE DOS RESULTADOS

Para a realização dos experimentos foram utilizados conjuntos de dados que caracterizam o problema do caixeiro viajante simétrico, ou seja, a distância entre duas cidades i e j é igual à distância entre as cidades j e i ($d_{ij}= d_{ji}$). A escolha dos conjuntos de dados foi determinada pela disponibilidade de resultados obtidos por outros trabalhos encontrados na literatura, quando estes se utilizaram de um determinado conjunto de dados. Os conjuntos de dados e seus respectivos percursos ótimos foram obtidos do repositório *TSPLIB* (REINELT, 2010). As bases de dados utilizadas para os experimentos foram: *Oliver30*, *Eil50*, *Eil75*, *kroA100*, *eil101*, *st70*, *kroA150*, *pr107*, *lin105* e *berlin52*.Cada conjunto de dados possui juntamente com seu nome, o número de cidades especificado, logo o número 30 na base de dados *Oliver30* se refere à quantidade de cidades que este conjunto denominado *Oliver* possui.

4.1 Experimento 1

O *Ant System* (*AS*) foi testado com o conjunto de dados *Oliver30*, sendo este um conjunto referente a um problema do caixeiro viajante simétrico com 30 cidades. Este problema possui como solução ótima o valor de 420.

A Tabela 4 mostra a comparação dos resultados obtidos pelo *AS* e por outros dois algoritmos, *Busca Tabu* (BT) e *Simulated Annealing* (*SA*). Os resultados dos algoritmos BT e SA foram obtidos de (DORIGO et al, 1999). As colunas Melhor, Média e Desvio Padrão se referem ao melhor, a média e o desvio padrão de 10 resultados obtidos de cada algoritmo, ou seja, a partir de dez execuções.

Tabela 4: Resultados obtidos pelo AS, BT e SA para o conjunto Oliver30.

	Melhor	Média	Desvio padrão
AS	424	425,82	2,5
BT	420	420,6	1,5
SA	422	459,8	25,1

A partir dos resultados ilustrados na Tabela 4, pode-se constatar que o melhor resultado para o problema *Oliver30* foi obtido pela *Busca Tabu*. O segundo melhor resultado foi obtido pelo *Simulated Annealing*, porém, o desvio padrão apresentado foi o maior entre todas as abordagens, o que significa que a variação das respostas é muito alta.

A Figura 4 mostra a evolução do melhor percurso encontrado pelas formigas durante a execução do algoritmo *AS*. Pode ser observado que no início, o melhor percurso está em torno de 500 e ao decorrer das iterações as formigas mantêm o melhor percurso encontrado. Segundo Dorigo et al. (1996), apesar da solução convergir rapidamente, a população mantém uma grande diversidade, e isto pode ser observado na Figura 5, que mostra a evolução do desvio padrão para cada iteração. Em cada iteração cada formiga encontra uma solução diferente.

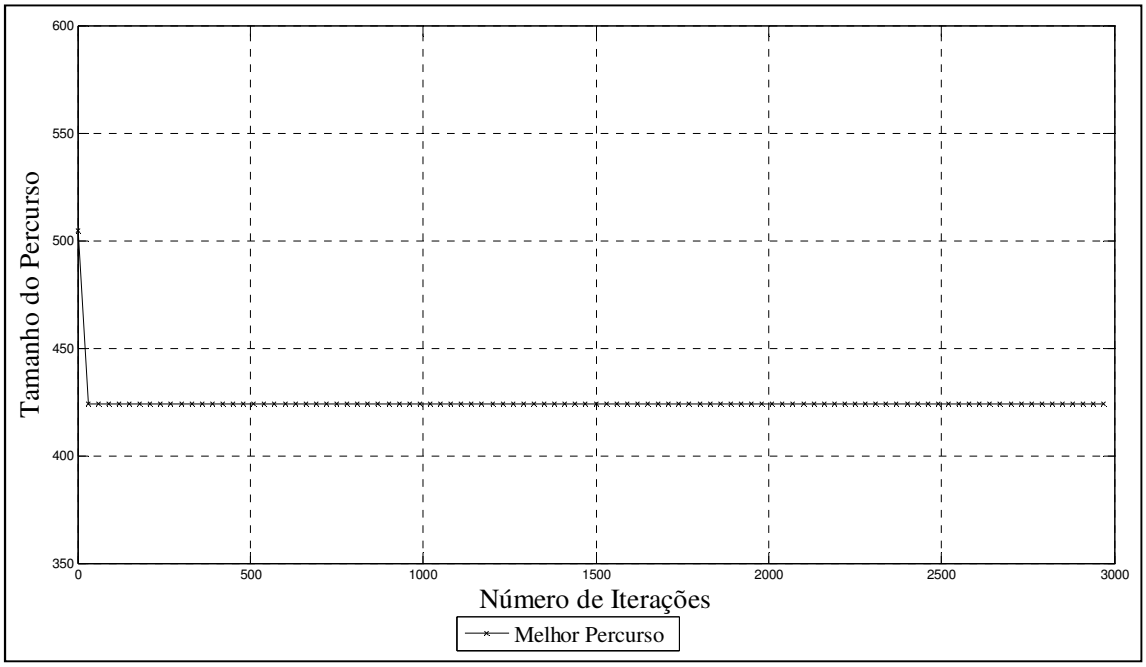

Figura 4: Evolução do melhor percurso para o problema *Oliver30*. Fonte: Próprio Autor.

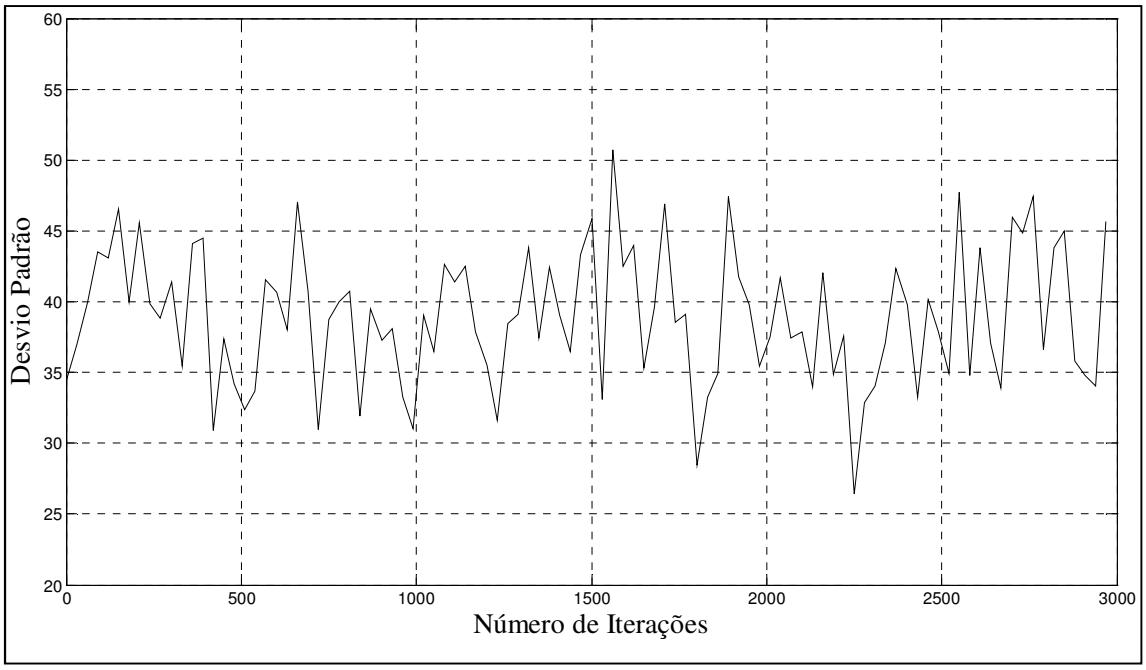

Figura 5: Evolução do desvio padrão dos menores caminhos encontrados pelas formigas. Fonte: Próprio Autor.

4.2 Experimento 2

O *Ant Colony System* foi comparado com três abordagens: *Algoritmos Genéticos* (*AG*), S*imulated Annealing* (*SA*) e *Programação Evolutiva* (*PE*). Nesta comparação foram utilizados três conjuntos de dados: *Eil50*, *Eil75* e *KroA100*. O *Eil50*, *Eil75* e *KroA100*, como já foi explicado no experimento1, também caracterizam um problema do caixeiro viajante simétrico com 50, 75 e 100 cidades, respectivamente. Os resultados dos algoritmos AG, SA e PE foram obtidos de (DORIGO et al, 1999), no entanto, o valor ótimo para os conjuntos *Eil50, Eil75* e *KroA100* não estão disponíveis. Para o conjunto *KroA100* foi possível encontrar o valor ótimo na TSPLIB (REINELT, 2010), que é 21282.

Os resultados descritos na Tabela 5 ilustram para cada abordagem duas colunas. A primeira apresenta o melhor resultado encontrado entre 10 execuções e a segunda coluna apresenta o número de iterações executadas pelo algoritmo. Os resultados que não estão disponíveis estão descritos como N/A.

Tabela 5: Resultados obtidos pelo ACS, AG, PE e SA para os conjuntos Eil50, Eil75 e KroA100.

	ACS Melhor	*ACS Iteração*	*AG Melhor*	*AG Iteração*	*PE Melhor*	*PE Iteração*	*SA Melhor*	*SA Iteração*
Eil50	427	2000	428	25000	426	100000	443	68512
Eil75	542	4000	545	80000	542	325000	580	173250
KroA100	21320	6000	21761	103000	N/A	N/A	N/A	N/A

A partir dos resultados ilustrados na Tabela 5, pode-se constatar que os melhores resultados encontrados são do *ACS*, exceto para o conjunto *Eil50*, onde a abordagem *PE* obteve melhor desempenho. Para o conjunto *Eil75*, as abordagens *ACS* e *PE* obtiveram a mesma solução, porém, o *ACS* encontrou a solução em um número de iterações muito menor em relação à abordagem *PE*. Isto acontece não somente para este conjunto, mas também para os conjuntos *Eil50* e *KroA100*.

A Tabela 6 exibe informações de uma execução do ACS para as bases de dados *Eil75, Eil50* e *KroA100*. Esta execução é a melhor entre 10 realizadas. As colunas melhor, pior e média se referem ao melhor, ao pior e a média do percurso encontrado pelo ACS. A coluna iteração descreve o número de iterações realizadas pelo algoritmo.

Tabela 6: Resultados obtidos pelo ACS para os conjuntos Eil50, Eil75 e KroA100.

	Melhor	*Pior*	*Média*	*Iteração*
Eil50	427	449	436.14	2000
Eil75	542	556	550.19	4000
KroA100	21320	21911	21560.63	6000

As Figuras 6, 7, 8 apresentam a evolução dos melhores, piores e média dos percursos encontrados pelas formigas utilizando as base de dados *Eil50*, *Eil75* e *KroA100*. Ao observar a evolução dos gráficos percebe-se que os melhores percursos encontrados pelo algoritmo *ACS* tendem a ser próximos aos valores apresentados pela literatura para essas bases. Outro fato relevante é que esses resultados se mantêm a partir de uma determinada iteração até o fim da execução do algoritmo.

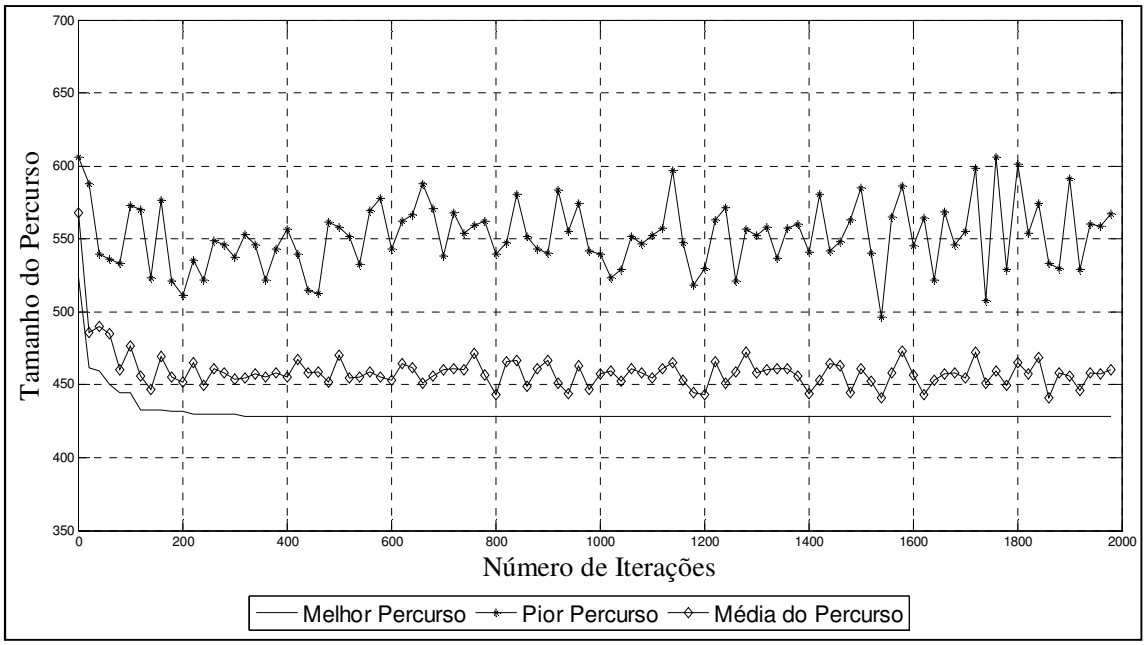

Figura 6: Resultados do melhor, pior e média dos caminhos encontrados para a base de dados *Eil50*. Fonte: Próprio Autor.

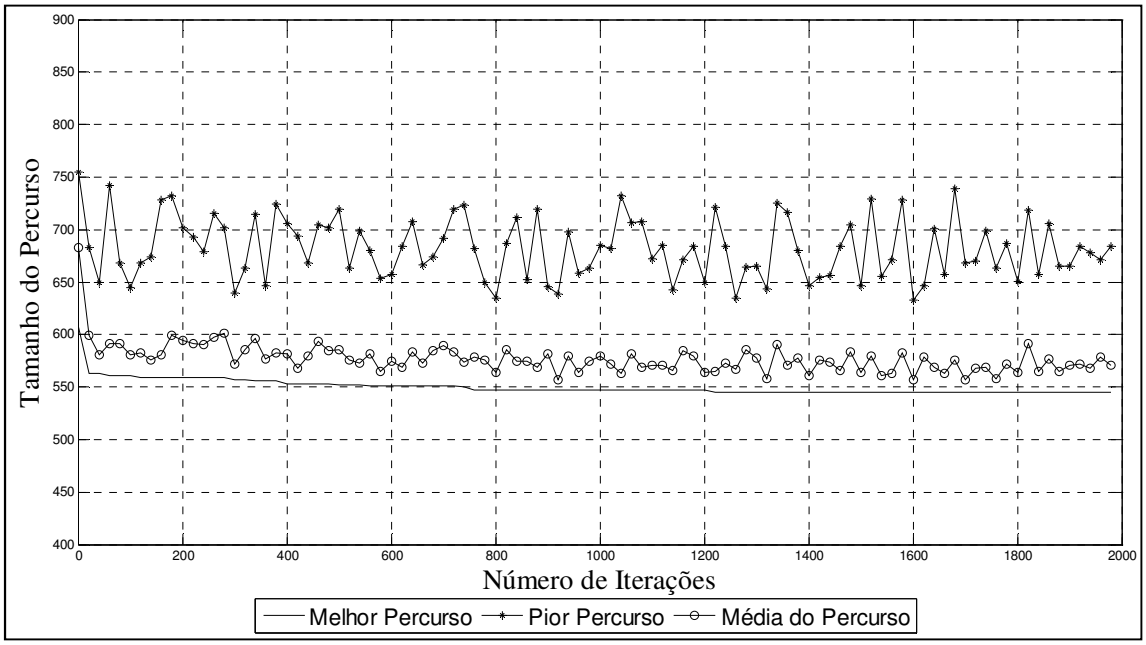

Figura 7: Resultados do melhor, pior e média dos caminhos encontrados para a base de dados *Eil75*. Fonte: Próprio Autor.

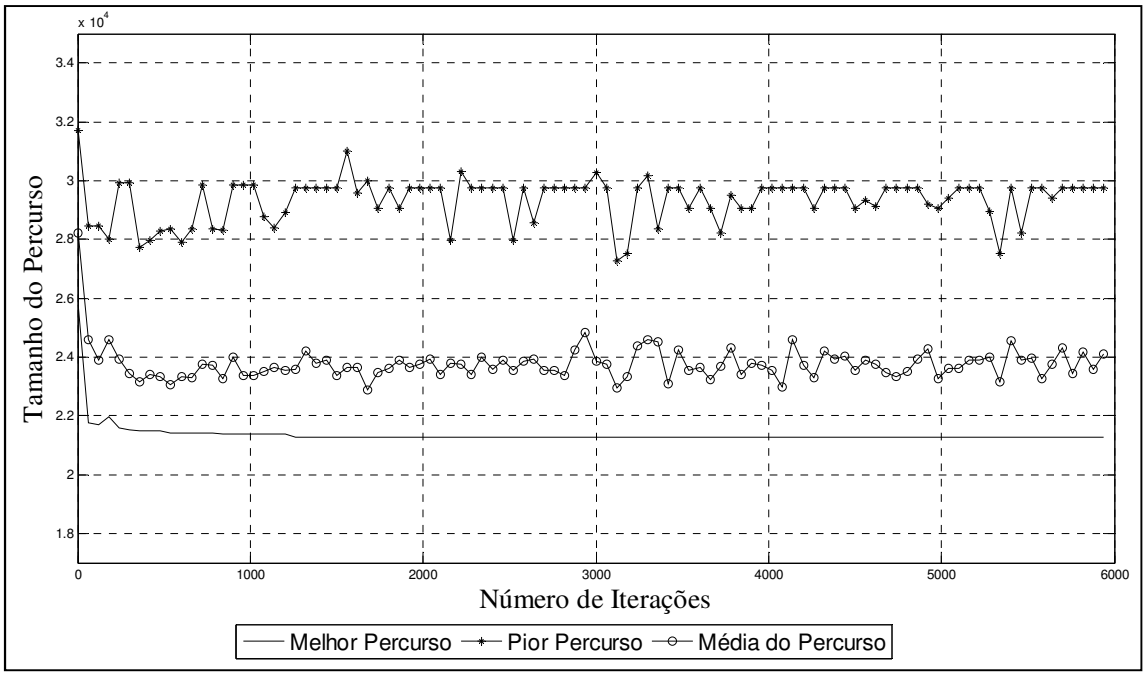

Figura 8: Resultados do melhor, pior e média dos caminhos encontrados para a base de dados *KroA100*. Fonte: Próprio Autor.

4.3 Experimento 3

O *ACS* foi testado também para outras bases de dados, sendo que para estas não foram encontrados na literatura trabalhos que as utilizassem. Sendo assim, não foi possível realizar a comparação dos resultados, como feito nos casos anteriores. As bases de dados utilizadas, as quais foram obtidas da biblioteca *TSPLIB* (REINELT, 2010), foram *berlin52*, *st70*, *eil101*, *lin105*, *pr107* e *kroA150.* Estas bases caracterizam um problema do caixeiro viajante simétrico com 52, 70, 101, 105, 107 e 150 cidades, respectivamente.

A definição dos valores dos parâmetros do ACS, tais como, tamanho da lista candidata e número de formigas, foram definidos a partir da realização de experimentos, os quais variavam os valores destes parâmetros.

Para cada base de dados foram realizadas 15 execuções do ACS, variando o número de iterações, o número de formigas e o tamanho da lista candidata. As tabelas apresentadas a seguir mostram o valor do melhor e do pior caminho encontrado pelo ACS, o valor médio dos caminhos e o número de iterações executados pelo algoritmo. Já as figuras ilustram a evolução do melhor e do pior caminho, assim como o valor médio dos caminhos encontrados pelo ACS. Estes valores (tanto os descritos nas tabelas

quanto os ilustrados nas figuras) se referem ao melhor desempenho do ACS entre as 15 execuções efetuadas.

Os resultados obtidos utilizando a base de dados *berlin52* são mostrados na Tabela 7. O valor ótimo para este problema é 7542. O melhor valor encontrado para o número de formigas para este problema foi 30 e o melhor valor para o tamanho da lista candidata foi 20.

Tabela 7: Resultados obtidos pelo ACS para o conjunto de dados *berlin52*.

Melhor	*Pior*	*Média*	*Iterações*
7544	7717	7601,806	3000
7544	7717	7601,806	5000
7544	7750	7581,043	7000
7544	7717	7555,854	8500

As Figuras 9, 10, 11 e 12 ilustram a evolução do algoritmo variando o número de iterações em 3000, 5000, 7000 e 8500, respectivamente.

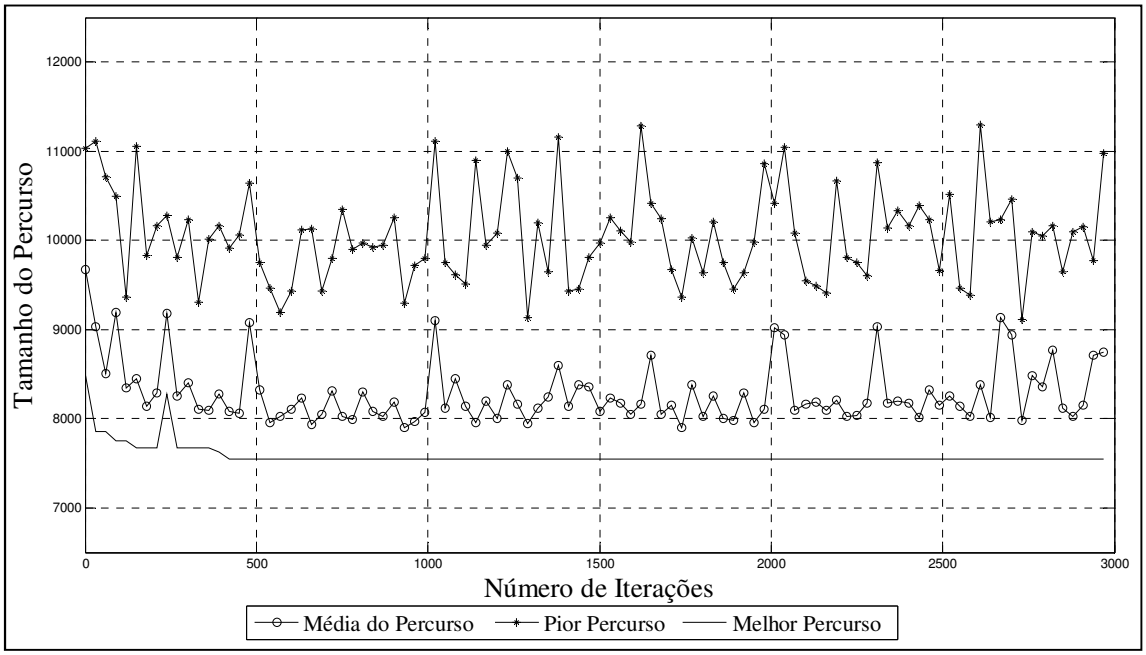

Figura 9: Evolução dos caminhos encontrados pelo ACS usando Berlin52 com 3000 iterações. Fonte: Próprio Autor.

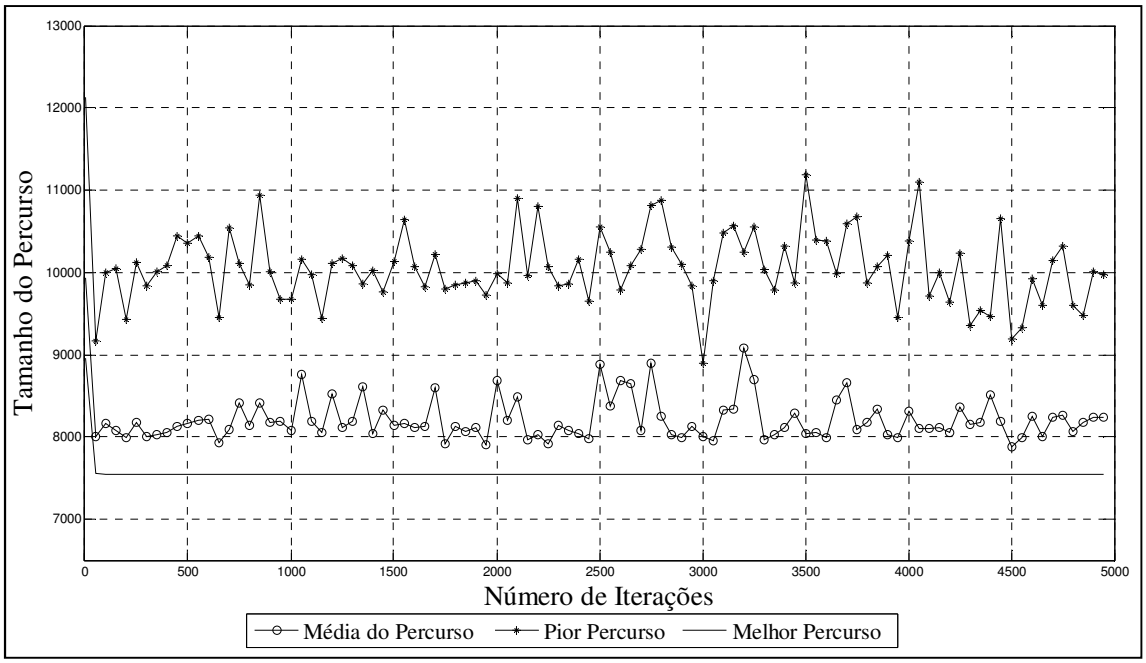

Figura 10: Evolução dos caminhos encontrados pelo ACS usando Berlin52 com 5000 iterações. Fonte: Próprio Autor.

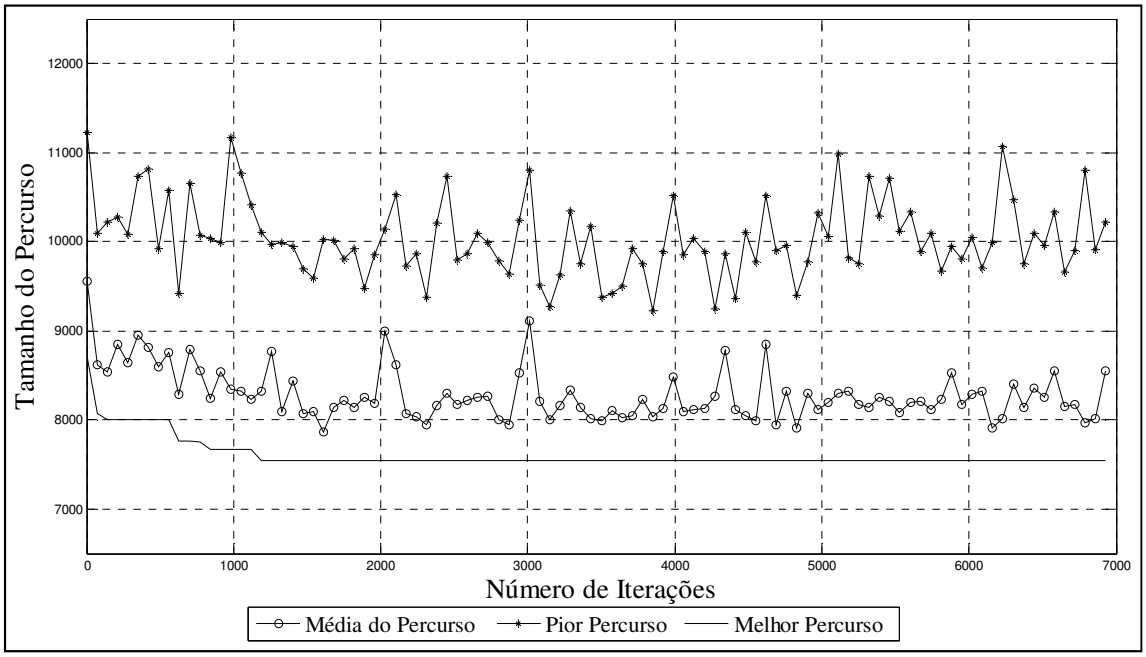

Figura 11: Evolução dos caminhos encontrados pelo ACS usando Berlin52 com 7000 iterações. Fonte: Próprio Autor.

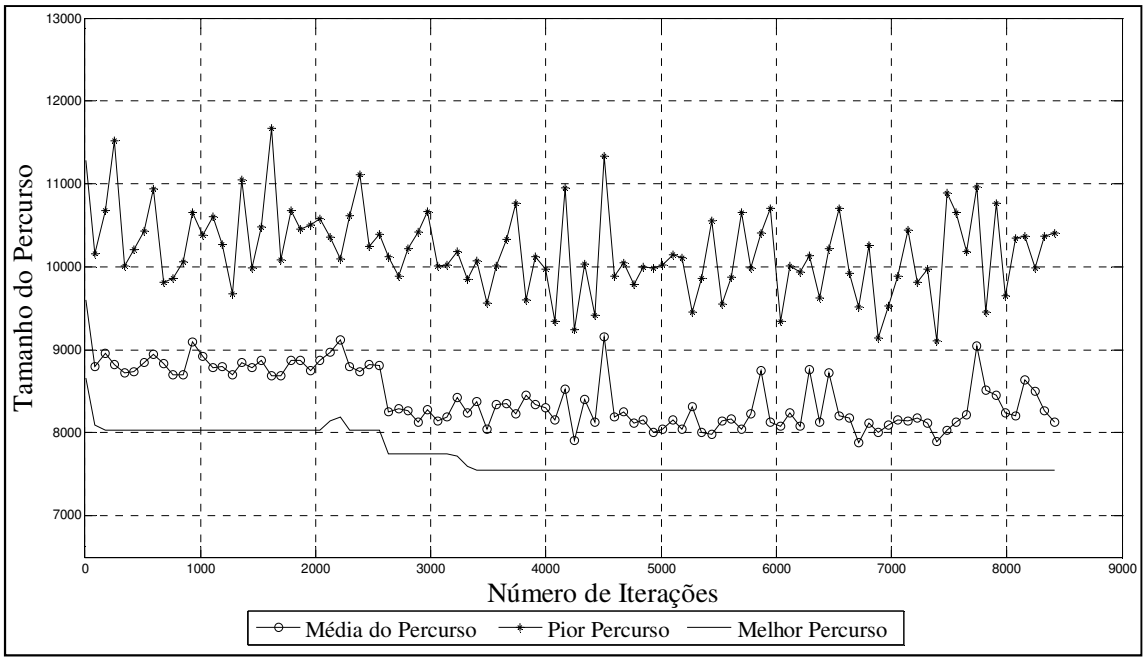

Figura 12: Evolução dos caminhos encontrados pelo ACS usando Berlin52 com 8500 iterações. Fonte: Próprio Autor.

A partir dos resultados descritos na Tabela 7 e dos gráficos apresentados nas Figuras 9, 10, 11 e 12, pode-se notar que o ACS obteve como resposta um valor bem próximo do valor ótimo do problema, e que o aumento do número de iterações não permitiu ao ACS a obtenção de melhores resultados.

Os resultados obtidos usando a base de dados *st70* são mostrados na Tabela 8. O valor ótimo para este problema é 675. O melhor valor encontrado para o número de formigas para este problema foi 30 e o melhor valor para o tamanho da lista candidata foi 40.

Tabela 8: Resultados obtidos pelo ACS para o conjunto de dados st70.

Melhor	Pior	Média	Iterações
677	697	686,241	3000
677	693	684,705	5000
677	687	682,888	7000
677	689	681,889	8500

As Figuras 13, 14, 15 e 16 ilustram a evolução do algoritmo variando o número de iterações em 3000, 5000, 7000 e 8500, respectivamente.

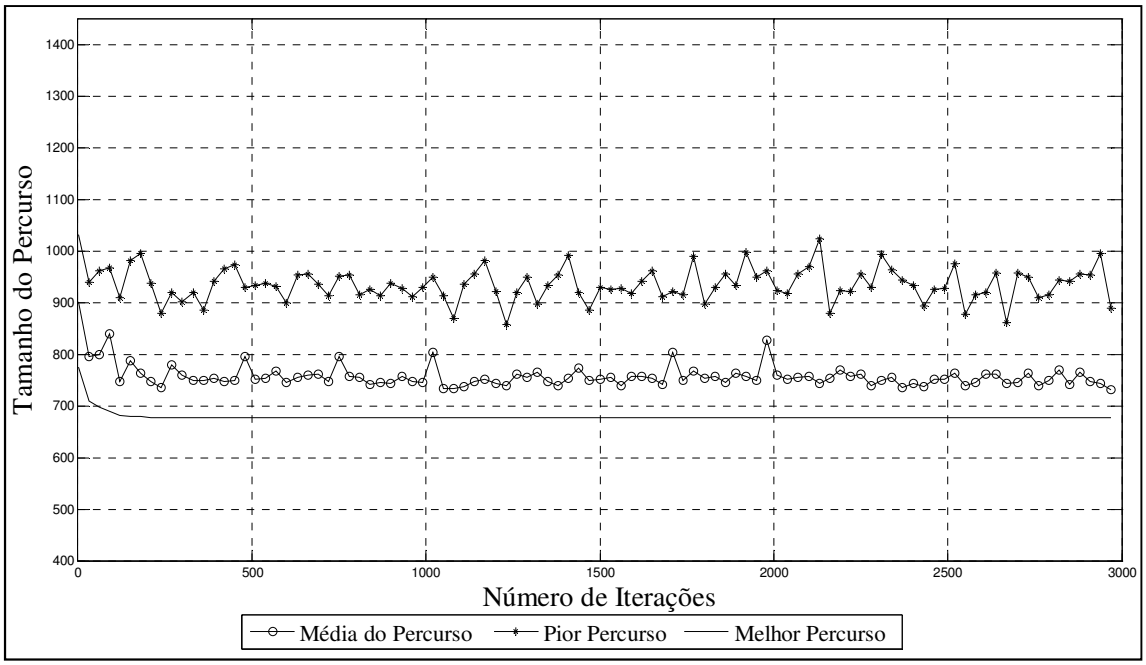

Figura 13: Evolução dos caminhos encontrados pelo ACS usando st70 com 3000 iterações. Fonte: Próprio Autor.

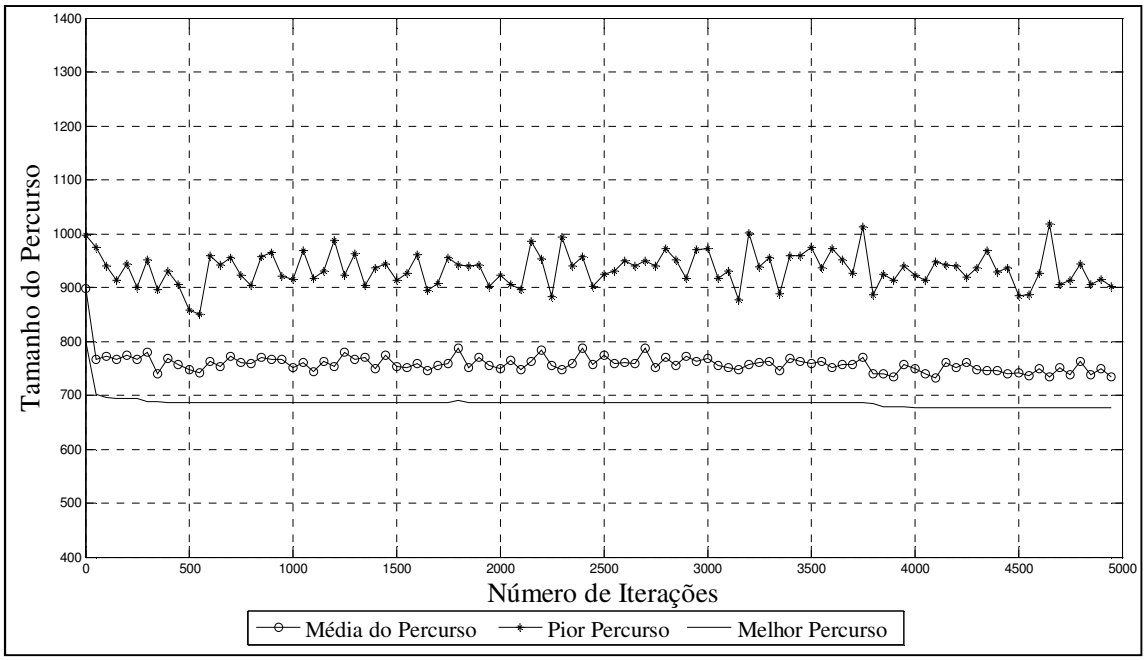

Figura 14: Evolução dos caminhos encontrados pelo ACS usando st70 com 5000 iterações. Fonte: Próprio Autor.

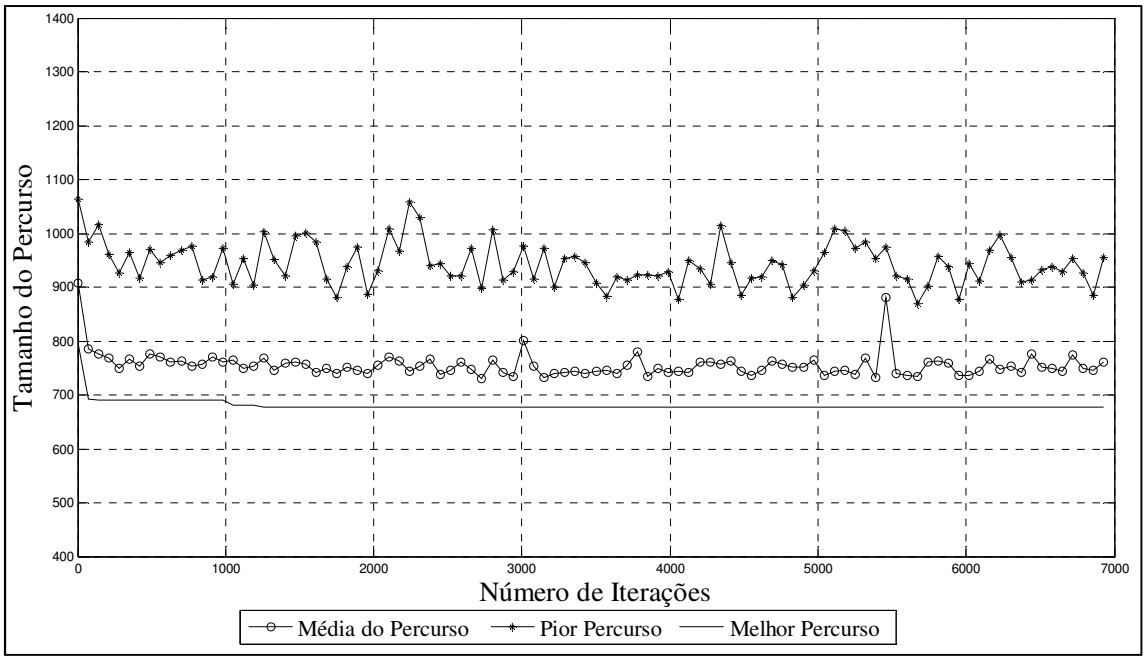

Figura 15: Evolução dos caminhos encontrados pelo ACS usando st70 com 7000 iterações. Fonte: Próprio Autor.

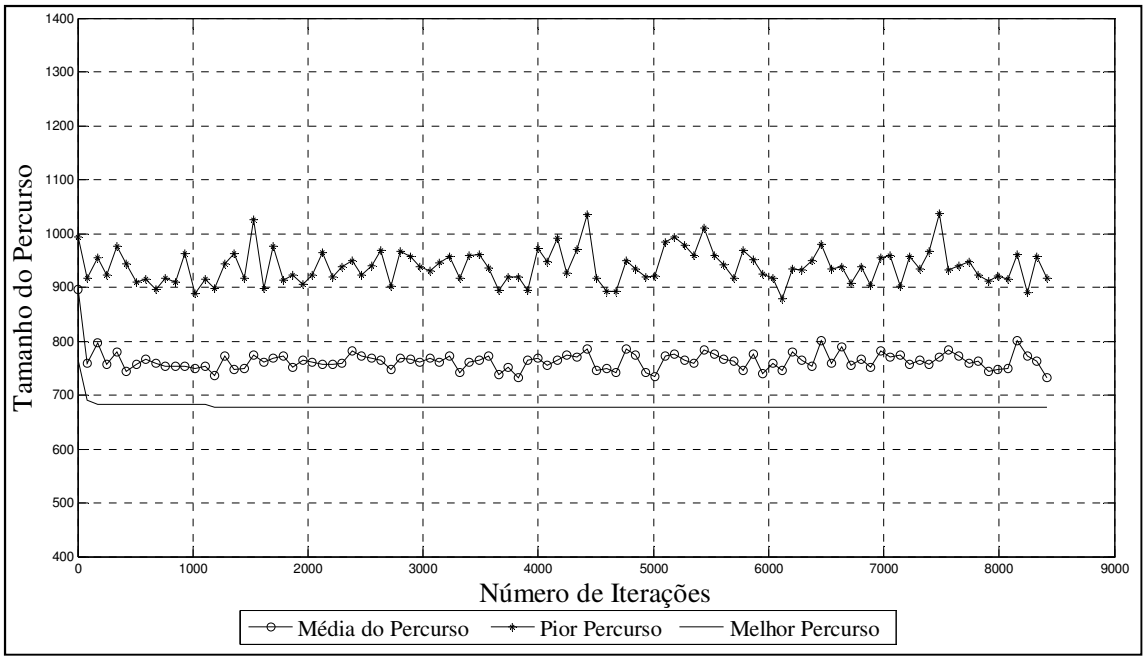

Figura 16: Evolução dos caminhos encontrados pelo ACS usando st70 com 8500 iterações. Fonte: Próprio Autor.

A partir dos resultados descritos na Tabela 8 e dos gráficos apresentados nas Figuras 13, 14, 15 e 16, pode-se notar que o ACS obteve como resposta um valor bem próximo do valor ótimo do problema, e que o aumento do número de iterações não permitiu ao ACS a obtenção de melhores resultados.

Os resultados obtidos utilizando a base de dados *eil101* são apresentados na Tabela 9. O valor ótimo para este problema é 629. O melhor valor encontrado para o número de formigas para este problema foi 70 e o melhor valor para o tamanho da lista candidata foi 60.

Tabela 9: Resultados obtidos pelo ACS para o conjunto de dados eil101.

Melhor	Pior	Média	Iterações
641	663	654,102	3000
641	664	651,426	5000
640	667	651,854	7000

As Figuras 17, 18, e 19 ilustram a evolução do algoritmo variando o número de iterações em 3000, 5000 e 7000, respectivamente.

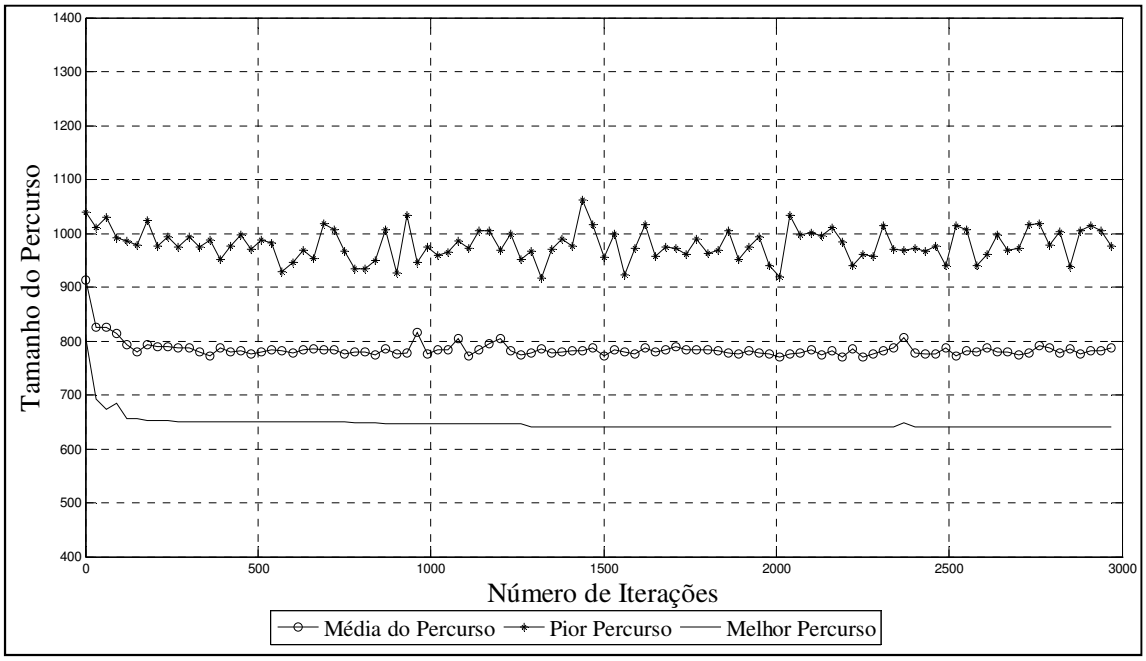

Figura 17: Evolução dos caminhos encontrados pelo ACS usando eil101 com 3000 iterações. Fonte: Próprio Autor.

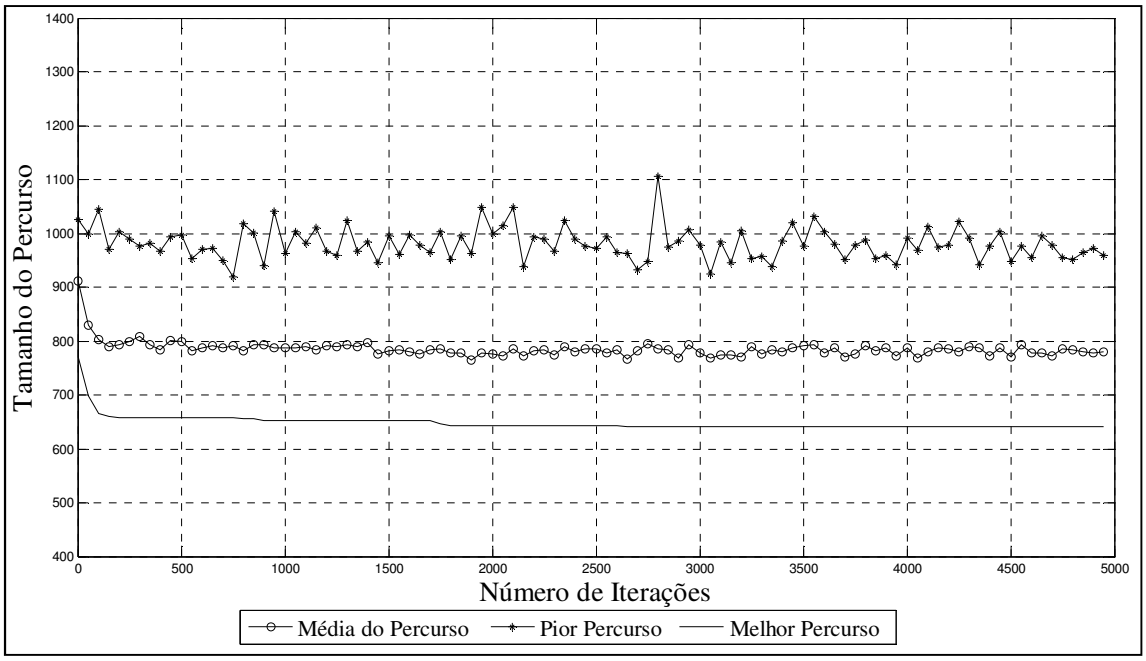

Figura 18: Evolução dos caminhos encontrados pelo ACS usando eil101 com 5000 iterações. Fonte: Próprio Autor.

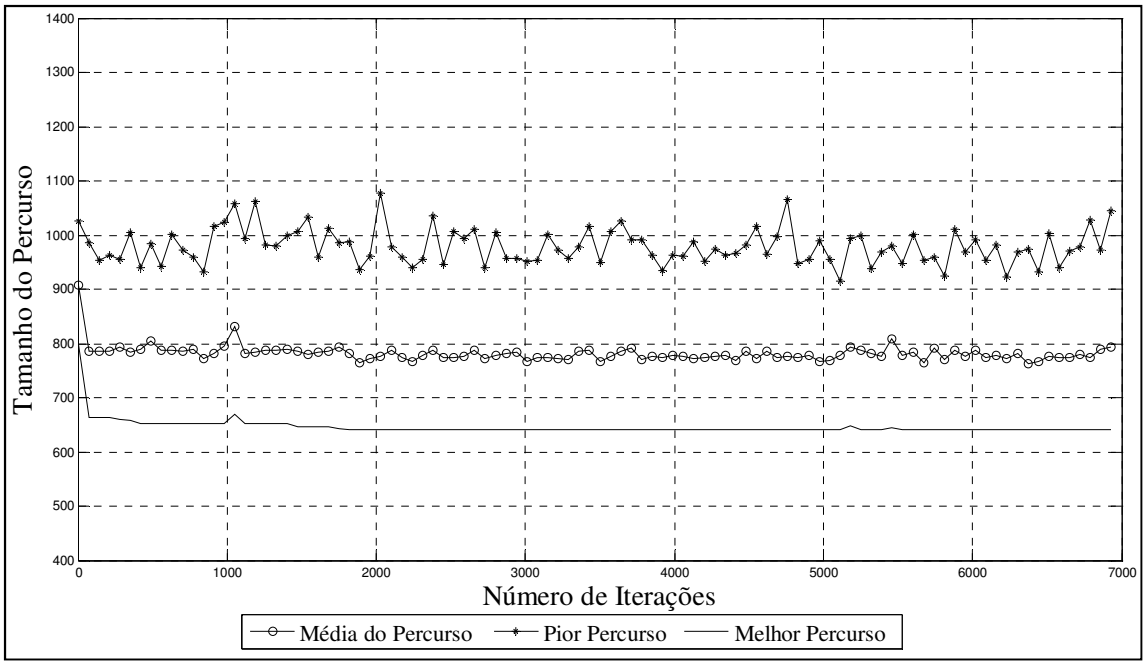

Figura 19: Evolução dos caminhos encontrados pelo ACS usando eil101 com 7000 iterações. Fonte: Próprio Autor.

A partir dos resultados descritos na Tabela 9 e dos gráficos apresentados nas Figuras 17, 18, e 19, pode-se notar que o ACS obteve como resposta um valor próximo do valor ótimo do problema. Além disso, o experimento com 7000 iterações foi o que obteve o melhor resultado.

Os resultados obtidos usando a base de dados *st70* são mostrados na Tabela 10. O valor ótimo para este problema é 14379. O melhor valor encontrado para o número de formigas para este problema foi 80 e o melhor valor para o tamanho da lista candidata foi 70.

Tabela 10: Resultados obtidos pelo ACS para o conjunto de dados *lin105*.

Melhor	Pior	Média	Iterações
14383	14627	14466,613	3000
14383	14627	14469,090	5000
14383	14520	14423,746	7000
14383	14510	14429,375	8500

As Figuras 20, 21, 22 e 23 ilustram a evolução do algoritmo variando o número de iterações em 3000, 5000, 7000 e 8500, respectivamente.

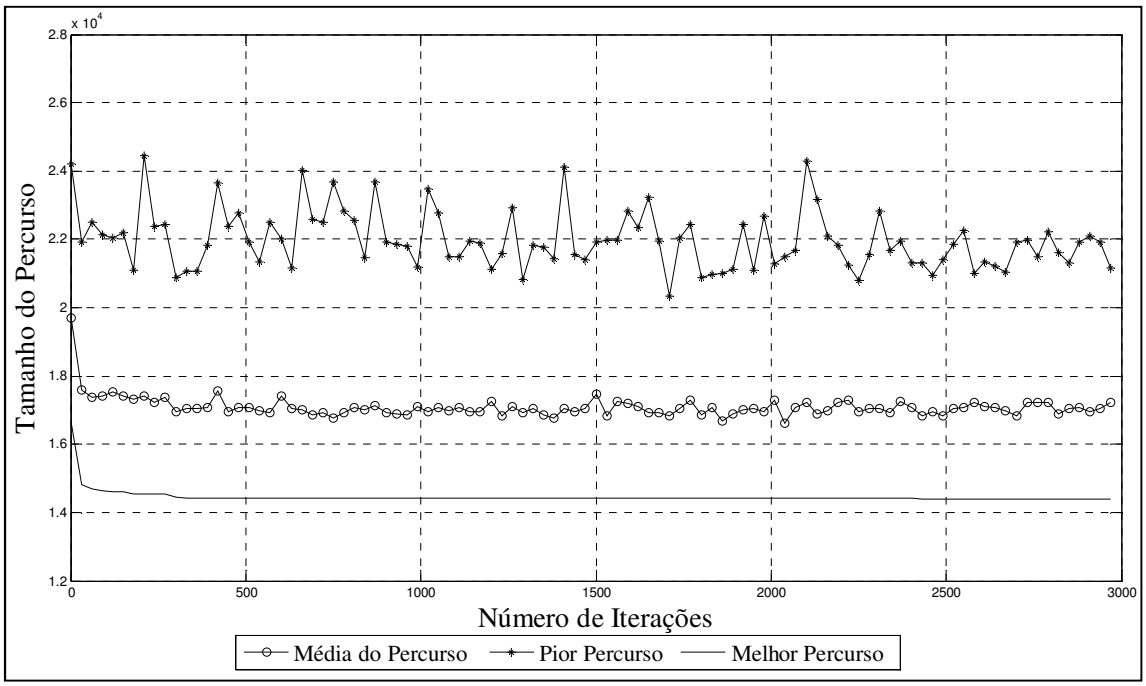

Figura 20: Evolução dos caminhos encontrados pelo ACS usando lin105 com 3000 iterações. Fonte: Próprio Autor.

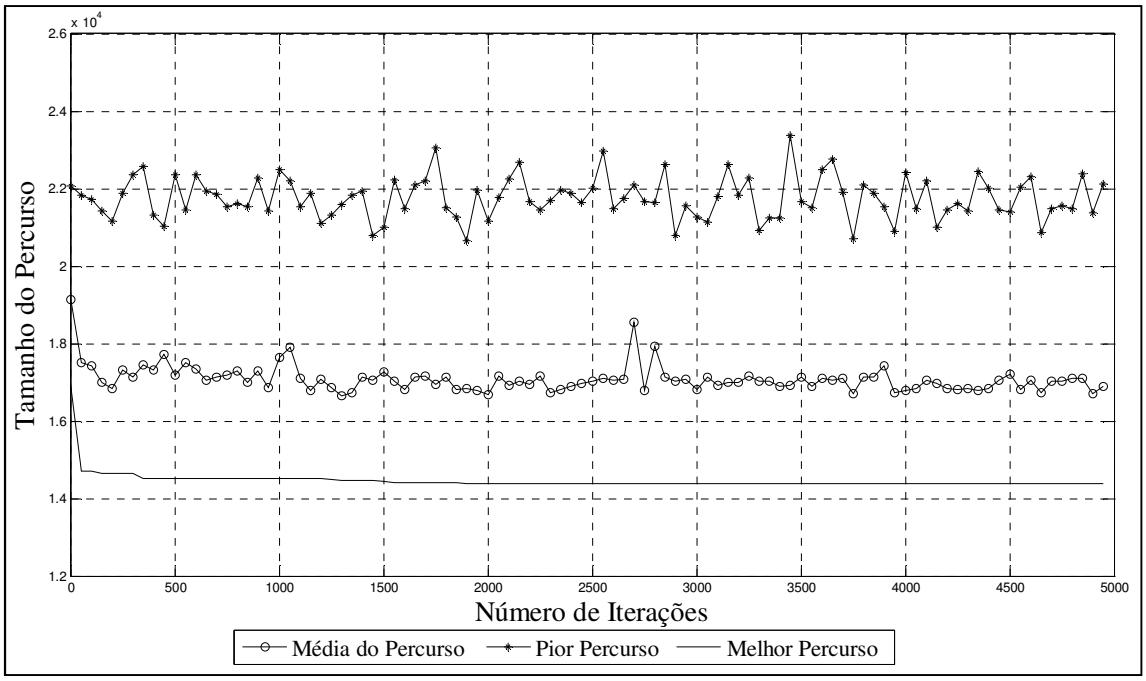

Figura 21: Evolução dos caminhos encontrados pelo ACS usando lin105 com 5000 iterações. Fonte: Próprio Autor.

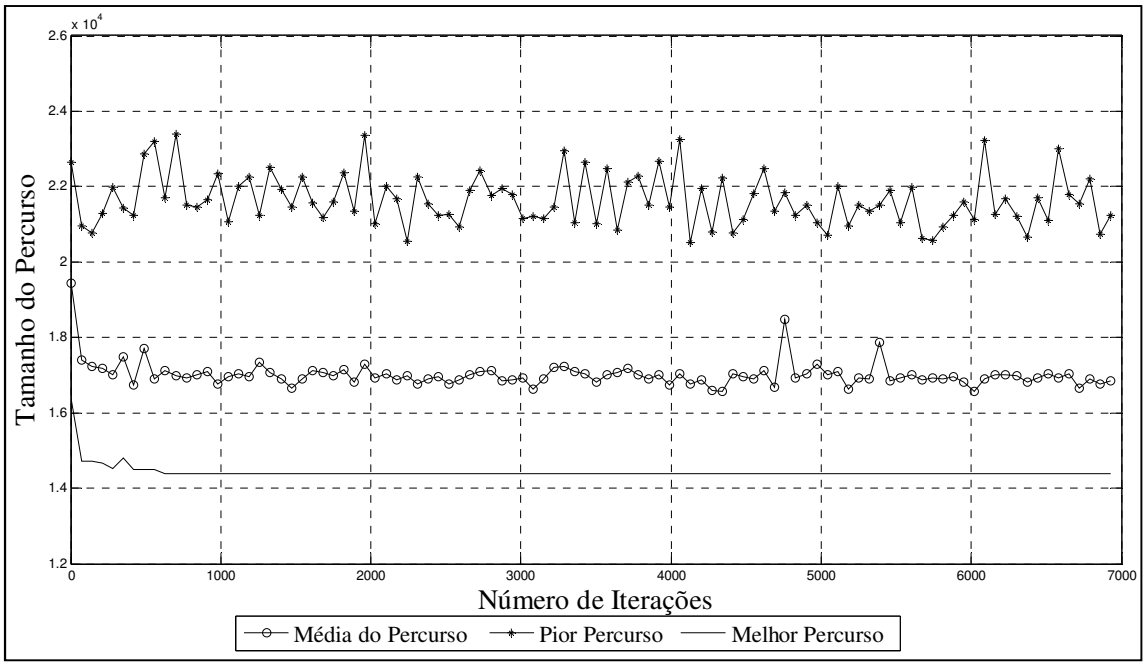

Figura 22: Evolução dos caminhos encontrados pelo ACS usando lin105 com 7000 iterações. Fonte: Próprio Autor.

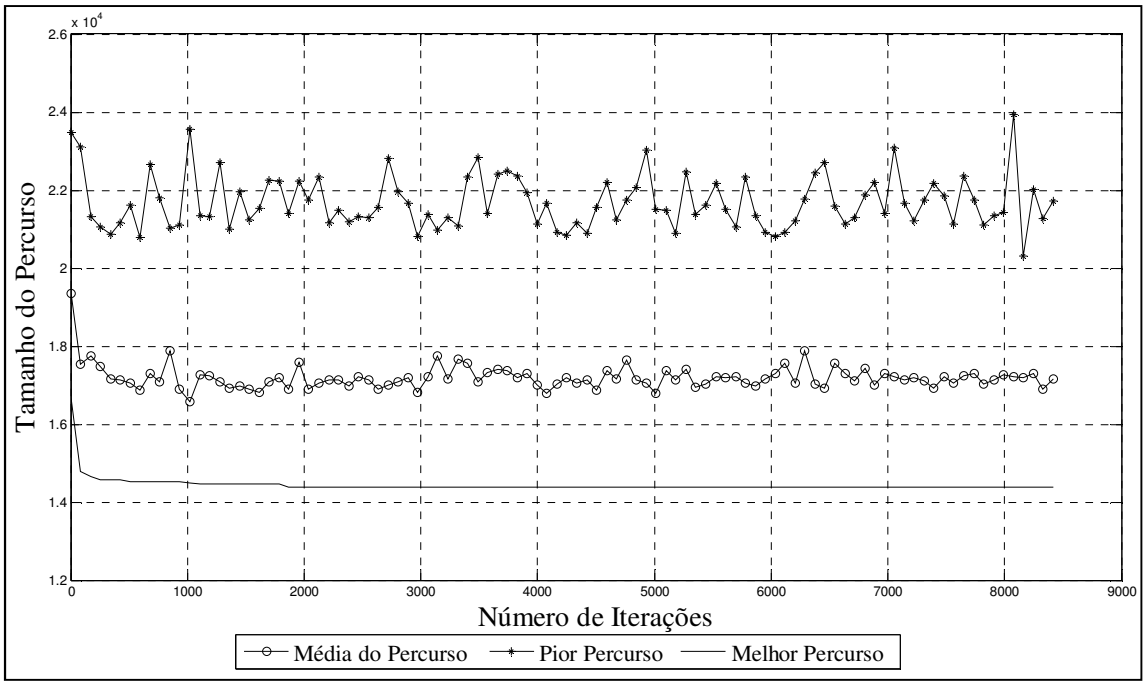

Figura 23: Evolução dos caminhos encontrados pelo ACS usando lin105 com 8500 iterações. Fonte: Próprio Autor.

A partir dos resultados descritos na Tabela 10 e dos gráficos apresentados nas Figuras 20, 21, 22 e 23, pode-se notar que o ACS obteve como resposta um valor próximo do valor ótimo do problema. Além disso, com o aumento do número de iterações, o algoritmo não encontrou melhores resultados.

Os resultados obtidos usando a base de dados *pr107* são mostrados na Tabela 11. O valor ótimo para este problema é 44303. O melhor valor encontrado para o número de formigas para este problema foi 70 e o melhor valor para o tamanho da lista candidata foi 60.

Tabela 11: Resultados obtidos pelo ACS para o conjunto de dados pr107.

Melhor	Pior	Média	Iterações
44346	44755	44592,251	3000
44303	44721	44498,440	5000
44303	44812	44531,128	7000

As Figuras 24, 25 e 26 ilustram a evolução do algoritmo variando o número de iterações em 3000, 5000 e 7000, respectivamente.

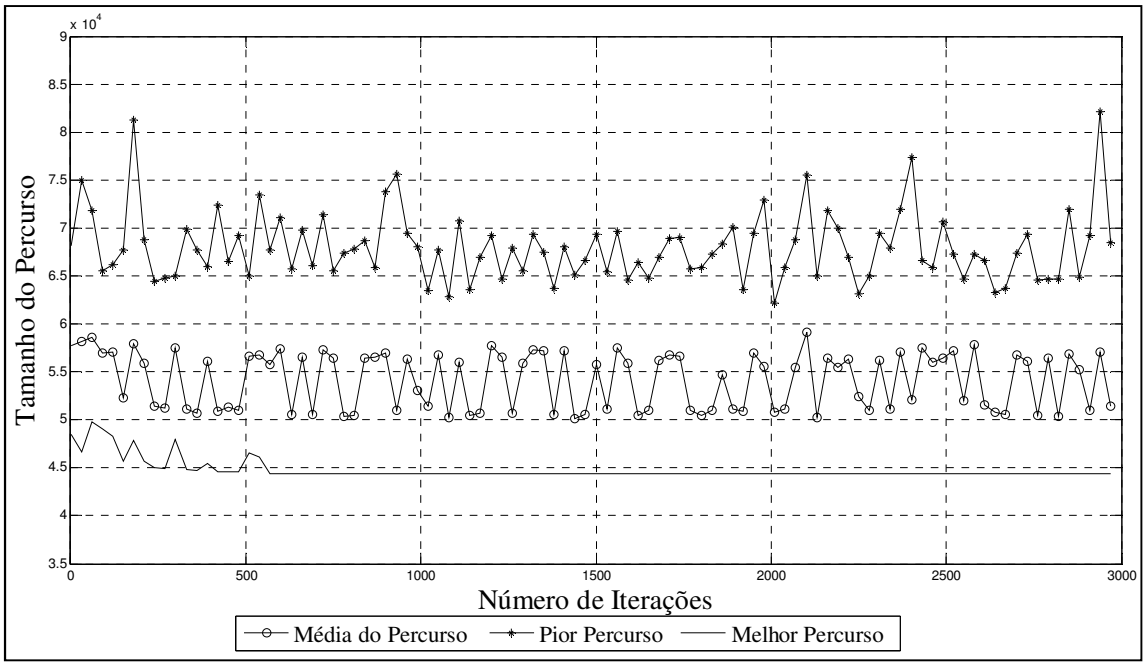

Figura 24: Evolução dos caminhos encontrados pelo ACS usando pr107 com 3000 iterações. Fonte: Próprio Autor.

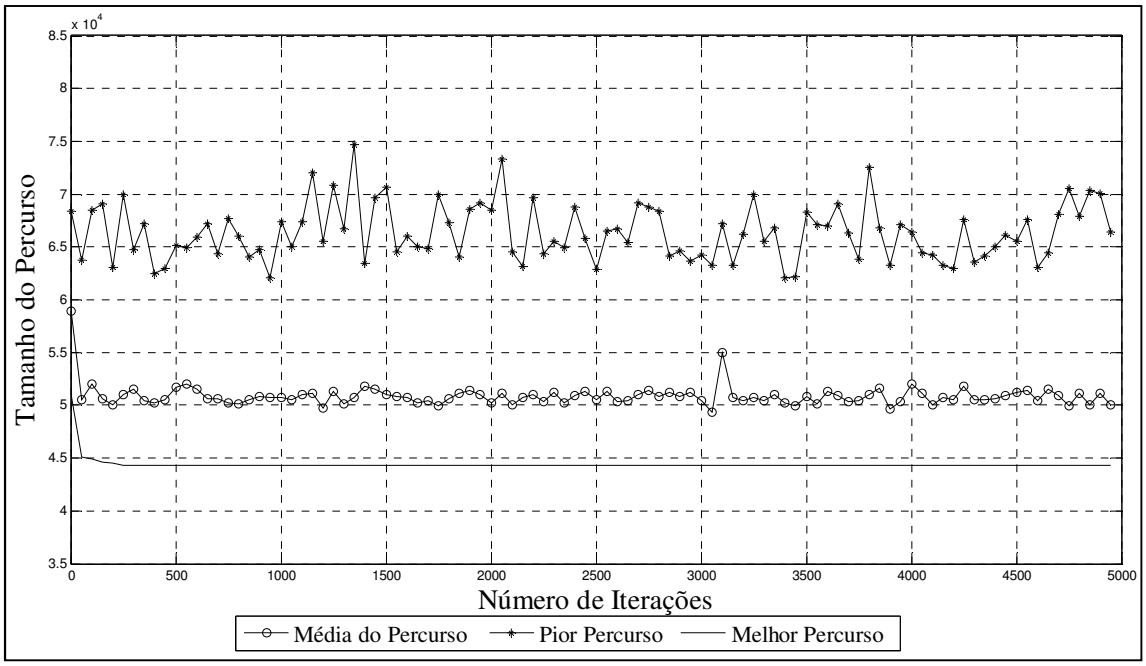

Figura 25: Evolução dos caminhos encontrados pelo ACS usando pr107 com 5000 iterações. Fonte: Próprio Autor.

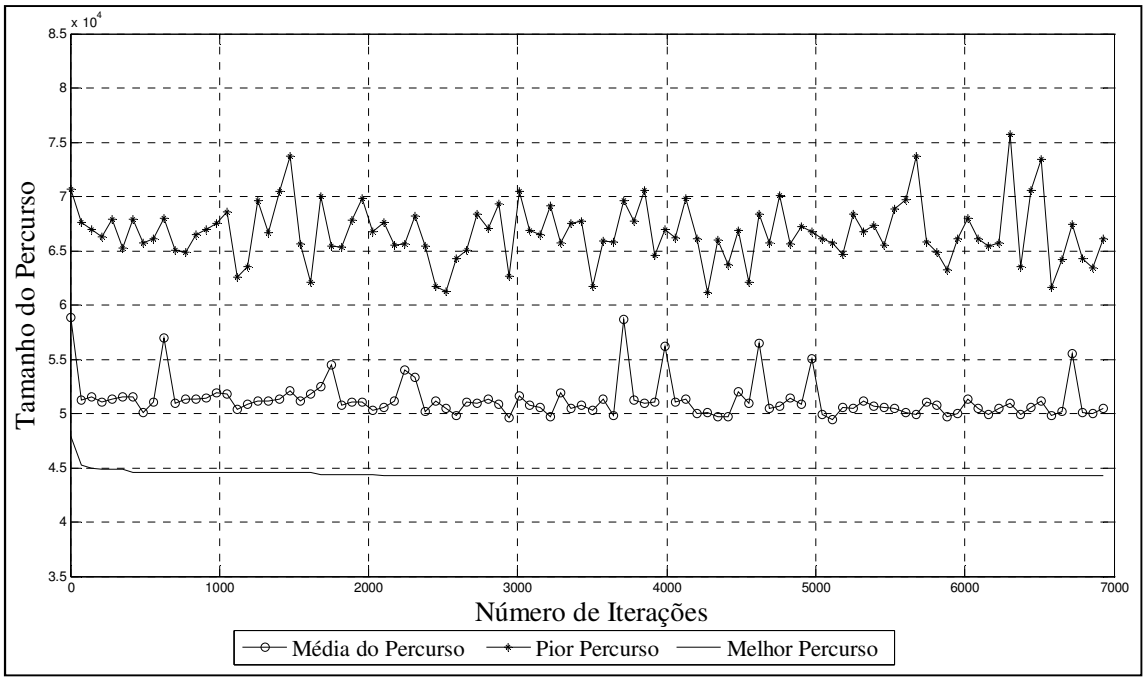

Figura 26: Evolução dos caminhos encontrados pelo ACS usando pr107 com 7000 iterações. Fonte: Próprio Autor.

A partir dos resultados descritos na Tabela 11 e dos gráficos apresentados nas Figuras 24, 25, e 26, pode-se notar que o ACS obteve como resposta o ótimo do problema, com exceção do experimento com 3000 iterações.

Os resultados obtidos usando a base de dados *kroA150* são mostrados na Tabela 12. O valor ótimo para este problema é 26524. O melhor valor encontrado para o número de formigas para este problema foi 50 e o melhor valor para o tamanho da lista candidata foi 40.

Tabela 12: Resultados obtidos pelo ACS para o conjunto de dados kroA150.

Melhor	Pior	Média	Iterações
26707	27605	27274,705	3000
26728	27678	27201,961	5000

As Figuras 27 e 28 ilustram a evolução do algoritmo variando o número de iterações em 3000 e 5000, respectivamente.

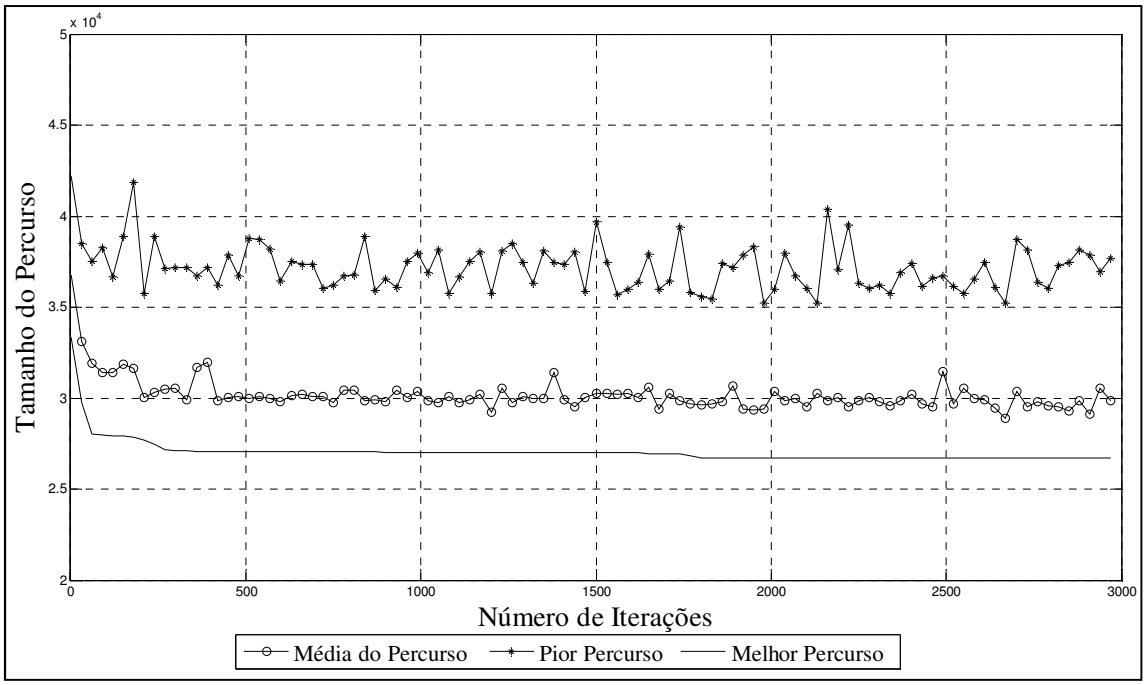

Figura 27: Evolução dos caminhos encontrados pelo ACS usando kroA150 com 3000 iterações. Fonte: Próprio Autor.

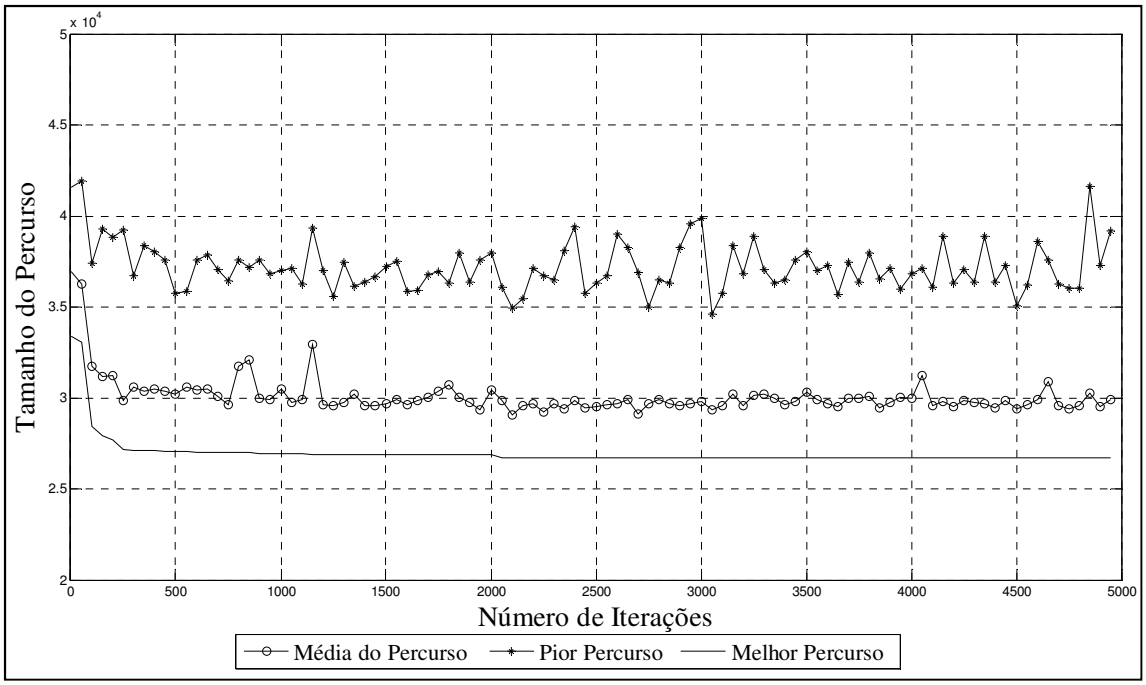

Figura 28: Evolução dos caminhos encontrados pelo ACS usando kroA150 com 5000 iterações. Fonte: Próprio Autor.

A partir dos resultados descritos na Tabela 12 e dos gráficos apresentados nas Figuras 27, e 28, pode-se notar que os resultados obtidos pelo ACS estão distantes do valor ótimo do problema.

5 CONCLUSÃO

Este trabalho teve como objetivo estudar a aplicação de algoritmos inspirados no comportamento de formigas para a solução do problema do Caixeiro Viajante simétrico. Dois algoritmos foram implementados e testados, o *Ant System* e o *Ant Colony System.*

O algoritmo *Ant System* foi testado com a base de dados Oliver30 e o seu desempenho foi comparado com outros algoritmos, tais como, Algoritmos Genéticos, Busca Tabu e *Simulated Annealing*. O Busca Tabu obteve o melhor desempenho. O *Simulated Annealing* obteve a segunda melhor resposta, porém, o seu desvio padrão foi muito alto, o que demonstrou a instabilidade do algoritmo. Por outro lado, o *Ant System*, com a terceira melhor resposta, apresentou um desvio padrão menor do que o *Simulated Annealing*, ou seja, possui melhor estabilidade.

O algoritmo *Ant Colony System* foi testado com várias bases de dados e comparado com apenas alguns outros algoritmos, pois não foram encontrados maiores informações sobre resultados de outras abordagens. O *Ant Colony System* se mostrou um algoritmo robusto, pois encontrou soluções bem próximas do valor ótimo dos problemas e, em alguns casos, conseguiu obter o ótimo. No entanto, para a base de dados *kroA150*, os resultados não foram satisfatórios. Para este caso, é necessária a realização de mais experimentos, com o objetivo de obter melhores resultados.

Uma das dificuldades encontradas na utilização dos algoritmos *Ant System* e *Ant Colony Sytem* foram a definição dos valores de seus parâmetros. Neste trabalho, vários experimentos foram realizados para determinar estes valores.

Para trabalhos futuros pretende-se aplicar o algoritmo *Ant Colony System* para o planejamento de rotas de robôs móveis autônomos. Será necessário realizar pequenas adaptações no algoritmo, para que este possa ser usado no planejamento de rotas.

.

6 REFERÊNCIAS BIBLIOGRÁFICAS

Dorigo M.; Maniezzo V.; Colorni A. The Ant System: Optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics**-Part B, vol.26, no.1, pp.29-41, 1996.

Dorigo M.; Gambardella L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. **IEEE Transactions on Evolutionary Computation**, vol.1, no.1, pp.53-66, 1997.

Dorigo, M.; Bonabeau E.; Theraulaz, G. **Swarm Intelligence From Natural to Artificial Systems**. Oxford University Press, 1ª Edição, 1999.

Johnson D.S.; McGeoch L. A. **The traveling salesman problem: a case study in local optimization.** Local search in Combinatorial Optimization. New York, Wiley, pp.215-310, 1997.

Johnson D.S.; McGeoch L.A. **Experimental Analysis of Heuristics for the STSP.** The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, pp. 369-443, 2002.

Lacerda, E. G. M. D. **Metaheurísticas**. Notas de aula. Disponível em <http://www.dca.ufrn.br/~estefane/metaheuristicas/ant.pdf>. Acesso em: 27 de julho de 2010.

Neto, R. F. T.; Coelho, L. D. S. **Planejamento de rotas para robôs de inspeção usando uma nova abordagem de Swarm Intelligence**. Workshop em Nanotecnologia e computação inspirada na biologia, vol.1, 2004.

Neto, R. F. T.; Coelho, L. D. S. **Planejamento de rotas para robôs de inspeção usando um algoritmo híbrido de colônia de formigas e algoritmo cultural**. Simpósio Brasileiro de Automação Inteligente, 2005.

NILSSON, N. J. **Principals of Artificial Intelligence**. New York: edição de Birkhauser, 1982.

Poole, D.; Mackworth, A.; Goebel, R. **Computational Intelligence: A Logical Approach**, Oxford University Press, New York, 1998.

Reinelt G. **TSPLIB.** Disponível em: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>. Acesso em: 10 de outubro de 2010.

Saadatmand, T. M.; Khademi, M.; Akbarzadeh-T, M.-R; Moghaddam, H. A. A Novel Constructive-Optimizer Neural Network for the Traveling Salesman Problem. **IEEE Transactions on Systems, Man and Cybernetics**, Part B (Cybernetics) vol.37, no.4, pp.754-770, 2007.

Wolsey, L. A. **Integer Programming**, Wiley-Interscience, 1998.

Ziviani, N. **Projeto de Algoritmos: com implementações em Pascal e C**, 2ª edição, Thomson Learning, São Paulo – SP, 2005.

ANEXO A – PSEUDOCÓDIGO DO ANT SYSTEM

Para toda aresta(i,j) **Faça**

$$\tau_{ij}(0) = \tau_0$$

Fim Para

Para k de 1 **até** m **Faça**

Coloque as formigas k sobre as cidades randomicamente

Fim Para

Faça T^+ ser o percurso mais curto do percurso encontrado e L^+ seu comprimento

Para t =1 **até** tmax **Faça**

Para k =1 **até** m **Faça**

Construa o percurso $T^k(t)$ aplicando n-1 vezes os seguintes passos

Escolha a próxima cidade j com a probabilidade

$$\rho_{ij}^k = \left\{ \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \right.$$

onde i é a cidade atual

Fim Para

Para k de 1 **até** m **Faça**

calcule o tamanho L^k do percurso T^k produzido pela formiga k

Fim Para

Se um melhor passeio é encontrado

Atualize T^+ e L^+

Fim Se

Para toda aresta(i,j) **Faça**

atualize a trilha de feromonio aplicando a regra

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

onde

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{L^k}, \text{ se a aresta ij faz parte do caminho k} \\ 0, \textit{ caso contrário} \end{cases}$$

Fim Para

Para toda aresta(i,j) **Faça**

$$\tau_{ij}(t+1) = \tau_{ij}(t)$$

Fim Para

Fim Para

τ_{ij} : Feromônio, τ_0 : Feromônio Inicial, m: Número de Formigas, k: Formiga, η_{ij} : Inverso da Distância, N_i^k : Cidades não visitadas, ρ : Probabilidade.

ANEXO B–PSEUDOCÓDIGO DO ANT COLONY SYSTEM

Para toda aresta(i,j) **Faça**

$\tau_{ij}(0) = \tau_0$

Fim Para

Para k de 1 **até** m **Faça**

Coloque as formigas k sobre as cidades randomicamente

Fim Para

Faça T^+ ser o percurso mais curto do percurso encontrado e L^+ seu comprimento

Para t =1 **até** t^{max} **Faça**

Para k =1 **até** m **Faça**

Construa o percurso $T^k(t)$ aplicando n-1 vezes os seguintes passos

Se existe pelo menos uma cidade j ϵ lista candidata **então**

Escolha a próxima cidade j. j ϵ J^k_i entre as cidades da lista candidata como segue

$$j = \begin{cases} \arg\max_{u \in J^k_i} \{[\tau_{uj}]\ [\eta_{uj}]^{\beta}\} & if\ \ q \leq q_0 \\ J\ \ if\ \ q > q_0 \end{cases}$$

Onde J ϵ J^k_i é escolhida de acordo com a probabilidade

$$\rho^k_{ij} = \begin{cases} \dfrac{[\tau_{ij}][\eta_{ij}]^{\beta}}{\sum_{l \in J^k_i} [\tau_{il}]\ [\eta_{il}]^{\beta}} \end{cases}$$

onde i é a cidade atual

Senão

escolha a cidade j mais próxima, J ϵ J^k_i

Fim Se

A formiga k aplica a regra de atualização local do feromônio

$$\tau_{ij} = (1-\sigma).\tau_{ij} + \sigma.\tau_0$$

Fim Para

Para k de 1 **até** m **Faça**

calcule o tamanho L^k do percurso T^k produzido pela formiga k

Fim Para

Se um melhor passeio é encontrado

Atualize T^+ e L^+

Fim Se

Para toda aresta(i,j) **Faça**

atualize a trilha de feromonio aplicando a regra

$$\tau_{ij} = (1-\sigma).\tau_{ij} + \rho.\Delta\tau^{melhor}_{ij}$$

onde

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L^+}, & \text{se a aresta ij faz parte do melhor caminho} \end{cases}$$

Fim Para

Para toda aresta(i,j) **Faça**

$$\tau_{ij}(t+1) = \tau_{ij}(t)$$

Fim Para

Fim Para

τ_{ij} : Feromônio, τ_0 : Feromônio Inicial, m: Número de Formigas, k: Formiga, η_{ij} : Inverso da Distância, J^k_i: Cidades não visitadas, ρ : Probabilidade.