UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Cassio Rodrigo Conti

INFORMAÇÃO HEURÍSTICA PARA AUMENTO DA VELOCIDADE DE CONVERGÊNCIA EM ALGORITMOS ACO PARA DOMÍNIOS CONTÍNUOS

Florianópolis

2011

Cassio Rodrigo Conti

INFORMAÇÃO HEURÍSTICA PARA AUMENTO DA VELOCIDADE DE CONVERGÊNCIA EM ALGORITMOS ACO PARA DOMÍNIOS CONTÍNUOS

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do Grau de Mestre em Ciência da Computação.
Orientador: Prof. Dr. Mauro Roisenberg

Florianópolis

2011

Catalogação na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

C762i Conti, Cassio Rodrigo
Informação heurística para aumento da velocidade de convergência em algoritmos ACO para domínios contínuos [dissertação] / Cassio Rodrigo Conti ; orientador, Mauro Roisenberg. - Florianópolis, SC, 2011.
94 p.: il., grafs., tabs.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Computação.

Inclui referências

1. Ciência da computação. 2. Otimização. 3. Formigas. 4. Inteligência de enxames. 5. Domínios contínuos. I. Roisenberg, Mauro. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 681

Cassio Rodrigo Conti

INFORMAÇÃO HEURÍSTICA PARA AUMENTO DA VELOCIDADE DE CONVERGÊNCIA EM ALGORITMOS ACO PARA DOMÍNIOS CONTÍNUOS

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 04 de março 2011.

Prof. Dr. Mario Antonio Ribeiro Dantas
Coordenador do Curso

Prof. Dr. Mauro Roisenberg
Orientador

Banca Examinadora:

Prof. Dr. Mauro Roisenberg
Presidente

Profa. Dra. Ana Lúcia Cetertich Bazzan

Prof. Dr. Anderson Luiz Fernandes Perez

Profa. Dra. Luciana de Oliveira Rech

Eu gostaria de dedicar este trabalho a todas as pessoas que de alguma forma estiveram envolvidas, me ajudando e incentivando. Mas, para não me estender muito, vou ser bem seletivo e dedicá-lo a três pessoas diretamente envolvidas, não apenas com este trabalho, mas com a minha vida:

Meu pai, Ari Conti, e minha mãe, Nadir Conti, que com muito suor, mas principalmente com muito amor e dedicação, me proporcionaram o estudo e o caráter sem o qual eu não teria chegado a lugar algum. "Tenho orgulho de vocês e tudo que eu faço na vida, penso sempre em vocês se orgulharem de mim. Vocês me deram tudo e sempre estaremos juntos em tudo que eu fizer."

Minha namorada, Jéssica Wildgrube Bertol, pela paciência nas horas que fiquei na frente do computador. E além desta paciência, quero agradecê-la pela motivação e apoio que sempre me dá. "Com você ao meu lado, tenho forças pra lutar contra qualquer dificuldade".

A vocês três todo o meu amor e o meu carinho para sempre.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela saúde, fé e perseverança que tem me dado. A meus amigos pelo incentivo na busca de novos conhecimentos, a todos os professores e professoras que muito contribuiram para a minha formação, dos quais tenho boas lembranças e o professor doutor, senhor Mauro Roisenberg, pelas orientações precisas e pela dedicação com a qual me guiou nestes dois anos.

Aprender é a única coisa de que a mente nunca se cansa, nunca tem medo e nunca se arrepende.

Leonardo da Vinci

RESUMO

Otimização por Colônia de Formigas (*Ant Colony Optimization* - ACO) é uma meta-heurística de otimização baseada no comportamento das formigas na busca por alimento. Esta meta-heurística foi originalmente desenvolvida para encontrar boas soluções em problemas de otimização combinatória discretos. Em domínios contínuos, a discretização do intervalo tem sido praticada para o uso de técnicas baseadas em ACO. Extensões do ACO para trabalhar diretamente com domínios contínuos têm surgido, entretanto as propostas mais similares à ideia clássica não usam a informação heurística chamada *visibilidade*, geralmente presente em algoritmos de ACO discreto. Neste trabalho é realizada uma revisão da ideia central do ACO mostrando a importância da visibilidade em domínios discretos e estendendo sua implementação em algoritmos ACO com domínio contínuo. Resultados de experimentos mostram a melhora na velocidade de convergência com o uso da heurística de visibilidade.

Palavras-chave: Otimização por colônia de formigas. Inteligência de enxames. Domínios Contínuos.

ABSTRACT

The Ant Colony Optimization (ACO) is a meta heuristic based on the foraging behavior of ants. It was originally designed to find good solutions to discrete combinatorial optimization problems. In continuous domains, the discretization of the interval has been applied for the use of techniques based on ACO. Extensions of the ACO to work directly with continuous domains have been proposed, but even those with close similarity with classical (discrete domain) ACO do not use the heuristic information called visibility commonly used in the original ACO algorithm. In this paper we show the importance of the visibility in ACO and propose two implementations of the visibility term for a continuous domain ACO algorithm with a more close similarity to classical ACO. Results of experiments show the improvement in convergence speed with the use of the visibility heuristic.
Keywords: ACO. Ant Colony Optimization. AS. Ant System. Swarm inteligence. Continuous Domain.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

A otimização é um campo da matemática aplicada cujos métodos são usados para resolver problemas quantitativos das mais diversas áreas (física, engenharia, biologia, economia, etc.) (OPTIMIZATION, 2011). Problemas onde o objetivo é maximizar ou minimizar funções através da escolha de valores para as variáveis que compõem uma solução do problema. Os problemas de otimização, em geral, podem ser divididos em 3 elementos (OPTIMIZATION, 2011): O primeiro é um valor numérico que é a saída da função objetivo, o qual se deseja maximizar ou minimizar, como por exemplo os custos de produção de uma empresa ou os lucros, o tempo de chegada de um veículo ao destino especificado ou o percurso total percorrido. O segundo elemento é um conjunto de variáveis onde seus valores podem ser manipulados de modo a alcançar o resultado desejado para a função objetivo, como por exemplo a quantidade de itens para serem comprados ou vendidos, a quantia de recursos a serem alocados para uma atividade, a rota a ser seguida por um veículo ou o mapa de cidades por onde precisa passar. O terceiro elemento é um conjunto de restrições para as variáveis, como por exemplo não alocar mais recursos do que os disponíveis ou não usar mais material do que o comprado, não visitar uma cidade que já foi visitada anteriormente. É uma área que cresce rapidamente com novas técnicas específicas para alguns problemas ou genéricas pra um certo grupo.

Problemas onde as variáveis possuem valores contínuos requerem técnicas diferentes de problemas onde as variáveis são discretas ou combinatoriais. Os problemas discretos são aqueles cujos valores para as variáveis estão dentro de um conjunto finito. Segundo Sergienko e Shylo (2006), a maioria dos problemas de otimização discreta são NP-difícil[1] o que torna praticamente impossível criar métodos eficientes para resolvê-los como métodos lineares. Além disso, existe a dificuldade computacional para resolver tais problemas devido, frequentemente, a suas amplas dimensões. Sergienko e Shylo (2006) também levanta que é impossível desenvolver métodos exatos aceitáveis para a maioria das classes de problemas. A complexidade para encontrar boas soluções cresce exponencialmente como cresce o número de elementos que compõe uma solução, devido ao fato de que com mais variáveis ou mais possíveis valores para as variáveis têm-se mais combinações para analisar.

Dentre os problemas de otimização discreta, alguns geralmente utilizados por autores para mostrar a efetividade de suas propostas são o Problema do Caixeiro Viajante (*Travelling Salesman Problem*) (LAPORTE, 1992)[2], o

[1] NP-difícil ou NP-complexo — *non-deterministic polynomial-time hard* (*NP-hard*).

[2] Os trabalhos referenciados para os 4 problemas clássicos citados neste parágrafo, são traba-

Problema da Mochila (*Knapsack Problem*) (IBARRA; KIM, 1975), o Problema da Cobertura de Conjuntos (*Set Covering Problem*) (CAPRARA; FISCHETTI; TOTH, 1999) e o Problema do Caminho Mínimo (*The Shortest-Path Problem*) (AHUJA et al., 1990). Novas técnicas de otimização que vão surgindo procuram expor sua eficiência realizando comparações com outras técnicas já publicadas. Para esta tarefa, utilizam-se dos recursos oferecidos por alguns centros/grupos de pesquisa que armazenam e disponibilizam instâncias para diversos problemas. Estes centros têm a finalidade de possibilitar a comparação de resultados e eficiência entre diferentes propostas aplicadas as mesmas instâncias de um problema. Assim um autor pode comparar seu trabalho com outros da literatura sem a necessidade de replicar o experimento já publicado.

Na busca de resolver problemas onde encontrar a solução ótima é inviável, surgiram as técnicas de otimização. Técnicas como *Simulated Annealing* (Têmpera Simulada) (KIRKPATRICK; GELATT; VECCHI, 1983), Algoritmos Genéticos (HOLLAND, 1992a) (HOLLAND, 1992b), Busca Tabu (GLOVER; LAGUNA, 1997), entre outras, foram propostas para resolver esta classe de problemas em uma menor quantidade de tempo. Estas técnicas não garantem que a melhor solução será encontrada, mas garantem ao menos uma boa solução em um tempo inferior a uma busca exaustiva. *Ant Colony Optimization* (ACO) também é uma meta-heurística de otimização. Foi proposta por (DORIGO, 1992) e se baseia no comportamento das formigas na busca por alimento na natureza. Esta técnica foi aplicada a vários problemas e mostrou ser um bom processo de otimização para problemas de domínio discreto (DORIGO; MANIEZZO; COLORNI, 1996) (DORIGO; GAMBARDELLA, 1997a) (DORIGO; CARO; GAMBARDELLA, 1999) (DORIGO; BONABEAU; THERAULAZ, 2000).

Na área de otimização, existem problemas onde os valores das variáveis estão em um domínio contínuo, ou seja, existem infinitos valores possíveis para cada variável do problema. Algoritmos de otimização discreta só podem ser aplicados para estes problemas se os intervalos de valores para as variáveis forem discretizados. Jalali, Afshar e Mariño (2006) propuseram um multi ACO para discretizar o espaço de busca contínuo de forma não homogênea a fim de focar a área em torno da solução ótima. Entretanto, a discretização do espaço de busca nem sempre é possível, especialmente se o intervalo de valores possíveis for grande ou se for necessária uma grande precisão na resposta (SOCHA, 2004).

Ao citar exemplos de problemas de otimização contínua têm-se problemas conhecidos por aparecem em diversas pesquisas na literatura, como por exemplo *Rosenbrock*, *Zakharov*, *Goldstein and Price*, *Martin and Gaddy*

lhos onde é possível encontrar uma boa definição do respectivo problema.

entre outros. A escolha dos valores dos pesos das conexões sinápticas em uma Rede Neural Artificial (RNA) também é um problema de otimização contínua e, nesse caso, existem diversas técnicas específicas para essa classe de problemas. Os problemas contínuos são aqueles onde tem-se uma função com diversas variáveis cujos valores estão restritos a um intervalo do domínio e a saída dessa função é o que se deseja maximizar ou minimizar. Entretanto o principal diferencial em relação aos problemas discretos está no fato de haver infinitos valores possíveis para as variáveis do problema, o que também inviabiliza uma busca exaustiva.

Métodos baseados na meta-heurística ACO aplicados diretamente para domínios contínuos foram propostos inicialmente por Bilchev e Parmee (1995) que sugeriram o *Continuos ACO* (CACO). Posteriormente, outros métodos como *Asynchronous Parallel Implementation* (API) (MONMARCHè; VENTURINI; SLIMANE, 2000) e *Continuous Interacting Ant Colony* (CIAC) (DRéO; SIARRY, 2002) também foram propostos em tentativas de extensões do ACO para domínios contínuos. Existem ainda outros trabalhos como o de Huang e Hao (2006), Kong e Tian (2006b), Madadgar e Afshar (2008) e Mathur et al. (2000) que estenderam alguma das propostas já citadas neste parágrafo, melhorando os métodos. De acordo com Socha e Dorigo (2006), estas propostas não mantém uma similaridade próxima com o ACO clássico ou não o seguem exatamente. Socha (2004) propôs uma extensão para domínios contínuos ($ACO_{\mathbb{R}}$) baseando-se nos conceitos originais do ACO. Socha e Dorigo (2006) testaram o algoritmo com problemas clássicos e mostraram melhora no desempenho do $ACO_{\mathbb{R}}$ em comparação com outros algoritmos baseados em ACO para domínios contínuos. Embora a implementação seja mais similar com o ACO clássico, ela não usa uma característica importante deste tipo de algoritmo; a *visibilidade*. Visibilidade é alguma informação do ambiente (informação heurística) que pode guiar as formigas para melhores regiões[3] do espaço de busca para melhorar a velocidade de convergência e o resultado.

Neste trabalho pretende-se mostrar a importância da influência do termo visibilidade para um algoritmo de ACO clássico (domínio discreto), comparando testes onde a visibilidade é usada com outros testes onde apenas o feromônio guia a evolução do algoritmo, aplicado a problemas bem conhecidos da literatura. Tomando ciência da importância deste termo na aceleração da convergência do algoritmo em melhores resultados, busca-se estender o conceito para domínios contínuos no $ACO_{\mathbb{R}}$ de forma a mostrar que o mesmo comportamento ocorre para esta classe de problemas. Uma implementação da extensão é então descrita, beneficiando-se dos conceitos gerais do $ACO_{\mathbb{R}}$, e o desempenho do modelo proposto é discutido sobre um conjunto de ex-

[3] Regiões com bons valores para a função objetivo.

perimentos com problemas conhecidos, realizados para mostrar a melhora na velocidade de convergência quando este termo heurístico é usado.

A meta-heurística ACO para domínios contínuos foi escolhida por ser uma área relativamente nova e com resultados bem promissores em simulações realizadas previamente durante definição do tema de pesquisa deste trabalho. Identificou-se que haviam assuntos para serem explorados nesta área. Algumas limitações podem ser observadas se comparar o $ACO_{\mathbb{R}}$ com técnicas desenvolvidas especificamente para alguns problemas, como por exemplo, no treinamento de uma rede neural artificial com perceptrons multi camadas (*Multilayer Perceptron*) (MLP) de alimentação a frente (*feedforward*) onde o algoritmo de aprendizado por Retro Propagação (*Backpropagation*) (WERBOS, 1974), desenvolvido para uso com essa classe de problemas, apresenta melhor desempenho que esta extensão do $ACO_{\mathbb{R}}$, apesar deste último alcançar bons valores e com o diferencial de poder ser aplicado para uma maior variedade de problemas por ser mais genérico.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Propor uma extensão do $ACO_{\mathbb{R}}$ para domínios contínuos, que siga as ideias e os conceitos clássicos do ACO original para domínios discretos, utilizando todos os seus recursos a fim de alcançar melhores resultados e com maior velocidade de convergência.

1.1.2 Objetivos Específicos

Estudar os algoritmos baseados em ACO para problemas discretos, entendendo as características da técnica. Com base nesse estudo, mostrar a importância do termo de visibilidade neste tipo algoritmo (discreto), ou seja, como essa informação heurística melhora o resultado e a velocidade de convergência, de forma que posteriormente possa ser feita uma ligação com algoritmos baseados em ACO para problemas contínuos.

Estudar os algoritmos de ACO para problemas contínuos, procurando compará-los e definir o mais eficiente e mais similar a ideia original de ACO para problemas discretos. Propor heurísticas que associem um valor númerico a qualidade das escolhas dos agentes para serem usadas como termo de visibilidade para o algoritmo escolhido, $ACO_{\mathbb{R}}$, tomando o cuidado de que a interpretação de extensão proposta não se distancie da ideia original de ACO

proposta originalmente (DORIGO, 1992).

Por fim, testar o algoritmo com e sem as extensões propostas para o termo visibilidade, a fim de comprovar um comportamento similar ao que ocorre em domínios discretos quando a visibilidade é usada. Este comportamento buscado é o aumento na velocidade de convergência do ACO nos melhores valores do domínio.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em cinco capítulos, sendo esta introdução o primeiro deles. A revisão bibliográfica necessária para o entendimento da proposta de pesquisa é apresentada a seguir, no capítulo 2. No capítulo 3 é apresentada a proposta desenvolvida a partir desta pesquisa. Os testes e simulações realizadas são apresentadas no capítulo 4 juntamente com os respectivos resultados analisados. O último capítulo deste trabalho, capítulo 5, apresenta a conclusão com as considerações finais sobre a pesquisa realizada.

2 OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

2.1 PROBLEMAS DE OTIMIZAÇÃO EM DOMÍNIOS DISCRETOS

Problemas de otimização são aqueles onde é preciso encontrar a melhor combinação de valores para as variáveis do problema de modo a maximizar / minimizar um objetivo. Como por exemplo, otimizar uma quantidade (lucro, custo, receita, número de produtos, entre outros). Em geral, seria necessária a comparação e verificação de todos os valores possíveis das variáveis de decisão para encontrar os melhores valores, mas como mostra Sergienko e Shylo (2006), existem diversas técnicas para "podar" a árvore de busca ao perceber que, ramificações de determinado nó, apresentam valores piores, além de técnicas probabilísticas. Esta classe de problemas geralmente demanda quantidade de tempo proporcional ao tamanho da instância[1]. Os problemas de otimização combinatória possuem a qualidade da solução ligada a ordem em que os elementos que a compõem aparecem e assim a enumeração das combinações seria necessária para encontrar o valor ótimo. Por exemplo, se o problema possui k variáveis binárias, existem 2^k soluções diferentes possíveis. Se $k = 100$ então $2^{100} \approx 10^{30}$ soluções possíveis. Para um computador que verifique um trilhão (10^{12}) de soluções por segundo, têm-se $10^{30} \div 10^{12} = 10^{18}$ segundos, que por sua vez $10^{18} \approx 400$ milhões de séculos. Quanto mais elementos, mais combinações são possíveis e maior é o tempo de processamento necessário para encontrar a solução ótima em uma busca exaustiva. Algoritmos de otimização buscam a melhor resolução de um problema em situações onde a solução determinística demanda uma quantidade de tempo grande o suficiente para ser considerada inviável, encontrando-a em uma quantidade de tempo definida ou aceitável.

Métodos de otimização combinatória buscam diminuir o tempo de processamento para encontrar boas soluções através de técnicas e heurísticas, algumas desenvolvidas especificamente para um determinado problema, como por exemplo, uma técnica específica para acelerar a busca pela melhor solução em um problema onde é preciso encontrar o menor caminho entre dois pontos, é excluir do espaço de busca qualquer caminho que comece com uma sequência (caminho incompleto) que já apresenta percurso maior do que uma outra solução (caminho completo) anteriormente encontrada. Em outras palavras, supondo que os pontos a serem ligados sejam A e E, sabendo que o caminho $ABCDE$ tem tamanho 10 e que o caminho incompleto ACB já atingiu

[1] Tamanho da instância/problema é relacionado com a quantidade de variáveis e valores possíveis para as variáveis.

o tamanho 11, ignora-se do espaço de busca qualquer caminho que comece com *ACB*. Também existem métodos e heurísticas mais genéricas, podendo ser aplicadas a uma maior variedade de problemas. Métodos inspirados nas mais diversas áreas, como fenômenos físicos, evolução das espécies, comportamento dos seres vivos, entre outros. Geralmente estas técnicas não garantem o melhor resultado possível (valor ótimo) mas podem garantir uma "boa" solução dentro de um tempo viável.

Dentre os problemas de otimização, se destacam por sua popularidade em trabalhos na literatura: O Problema de Roteamento de Veículos, que consiste no atendimento de um conjunto de consumidores por meio de uma frota de veículos que partem de pontos chamados *depósitos* e cada veículo possui uma restrição relacionada a capacidade de atender as demandas dos clientes; O Problema do Menor Caminho, onde o objetivo é percorrer por entre dois pontos de um grafo (ou duas cidades de um mapa) de modo que o caminho interligando esses pontos seja o menor possível; O Problema da Mochila, que consiste em carregar uma mochila guardando objetos de diferentes pesos e valores sendo o objetivo preencher a mochila com o maior valor possível, não ultrapassando o peso máximo suportado por ela; O Problema da Alocação de Professores e Disciplinas, entre outros. Outro clássico exemplo de problema de otimização é o Problema do Caixeiro Viajante (TSP) — *Traveling Salesman Problem* — onde a ideia é que existe um vendedor (caixeiro) e um conjunto de cidades pelas quais ele deve visitar todas sem repetir nenhuma cidade. O caminho total percorrido deve ser o menor possível. Se existem 2 cidades, *A* e *B*, considerando que o caixeiro inicialmente está em *A*, tem-se apenas 1 solução possível que é a distancia entre *A* e *B*. Se existem 3 cidades, *A*, *B* e *C*, têm-se 2 caminhos possíveis, *ABC ACB*. 4 cidades, *A*, *B*, *C* e *D*, indicam 6 caminhos possíveis, *ABCD*, *ABDC*, *ACBD*, *ACDB*, *ADBC* e *ADCB*, ou seja, o fatorial do número de cidades subtraído 1 unidade é a quantidade de possíveis caminhos diferentes, conforme mostrado na Equação 2.1.

$$numeroCaminhosPossiveis = (numeroCidades - 1)! \tag{2.1}$$

Para 10 cidades temos $(10-1)! = 362880$ caminhos diferentes. Para 20 cidades temos $19! = 121645100408832000$ caminhos diferentes. Para obter o melhor resultado possível é necessário verificar todos os caminhos possíveis. Com muitas cidades no problema, isso torna-se inviável devido ao tempo demandado que cresce exponencialmente com número de cidades. Em situações como essa é que os algoritmos de otimização são extremamente necessários e com o intuito de auxiliar nessas situações é que foram desenvolvidos.

2.2 EXEMPLOS DE TÉCNICAS DE OTIMIZAÇÃO

Algoritmos Genéticos (AG) são técnicas que se baseiam na ideia de melhorar um conjunto de indivíduos, chamado *população*, que são soluções do problema. Essa melhora ocorre de acordo com conceitos de variabilidade genética e seleção natural. Os algoritmos genéticos foram propostos por Holland (1975) e, segundo Sergienko e Shylo (2006), vieram a ser aplicados em problemas de otimização discreta nos anos 80. Esta classe de algoritmos foi usada para resolver diversos problemas de otimização discreta (HOLLAND, 1992b) (HOLLAND, 1992a) e continua sendo ativamente desenvolvida.

Simulated Annealing (SA) (Têmpera Simulada ou Arrefecimento Simulado ou ainda Recozimento Simulado) é uma meta-heurística probabilística para problemas de otimização que se baseia nos conceitos da termodinâmica. O processo no qual a técnica se fundamenta é a atividade metalúrgica onde um material é levado a uma alta temperatura, no qual ocorre a fusão, e então é feito um resfriamento lento e controlado para redução dos defeitos no material ao final. O algoritmo procede de forma a encontrar soluções próximas a uma solução atual, permitindo "passos" distantes de acordo com uma *temperatura* T. Inicialmente T é alta, permitindo grandes mudanças e é diminuída com o passar das iterações, permitindo somente pequenas modificações na solução (KIRKPATRICK; GELATT; VECCHI, 1983) (CERNý, 1985).

Tabu Search (TS) (Busca Tabu ou Pesquisa Tabu) é uma heurística de otimização onde o algoritmo explora o espaço de busca através de uma busca local combinada com a memória da heurística. Estando em uma solução, o algoritmo tenta avançar a uma solução vizinha melhor do que a solução atual até que seja satisfeito um critério de parada. Ele evita a situação de visitar uma solução que já foi visitada em um passo anterior do algoritmo utilizando um método de memorização, a lista tabu. A solução vizinha é escolhida com a ajuda de uma busca local, que exclui da procura, um número n de soluções já visitadas, sendo n o tamanho da memória (lista tabu) (GLOVER, 1989) (GLOVER, 1990).

2.3 ALGORITMOS DE ACO

Ant Colony Optimization (ACO) é uma meta-heurística inspirada no comportamento das formigas na busca por alimentos. A técnica foi inicialmente aplicada para resolver problemas otimização combinatória quando foi introduzida por Marco Dorigo em sua tese de doutorado (DORIGO, 1992). Dorigo também propôs o *Ant System* (AS) que foi o primeiro exemplo de algoritmo baseado em ACO (DORIGO; MANIEZZO; COLORNI, 1991).

Os algoritmos baseados no ACO tentam explorar o seguinte comportamento: durante o processo de busca por comida, as formigas exploram aleatoriamente a área ao redor do ninho e depositam nas trilhas por onde passam uma substância química chamada feromônio que evapora com o passar do tempo. Madadgar e Afshar (2008) acrescentam que a quantidade de feromônio depositada pela formiga pode depender da qualidade e quantidade de comida. Uma formiga, ao cruzar por uma trilha de feromônio, tende probabilisticamente a seguir a trilha de maior concentração de feromônio (não obrigatoriamente). A Figura 1 exemplifica o comportamento das formigas em busca do alimento. Em 1A as formigas seguem uma trilha entre o alimento e o ninho. Em 1B um obstáculo é inserido sobre o caminho. Em 1C, metade das formigas vão pelo caminho mais longo e a outra metade pelo caminho mais curto. As formigas que foram pelo caminho mais curto, vão chegar primeiro até o alimento. Na volta ao ninho, algumas escolherão o caminho maior outras o menor. As que escolherem o menor chegarão antes ao ninho e o ciclo se repete de modo que o menor caminho passa a acumular mais feromônio. Com o passar do tempo, a concentração de feromônio no menor caminho será suficientemente superior que a do maior caminho fazendo que a maioria das formigas escolham o menor caminho. Além de pouco feromônio no maior caminho, este também evapora em função do tempo diminuindo ainda mais a probabilidade de que ele seja escolhido chegando ao exemplificado na Figura 1D. Os trabalhos de (DORIGO; MANIEZZO; COLORNI, 1996), (DORIGO; BONABEAU; THERAULAZ, 2000) e (DORIGO; STüTZLE, 2009) mostram mais detalhadamente como esse comportamento implica que, ao passar do tempo, os caminhos mais longos passam a ser raramente utilizados e os menores caminhos prevalecem entre o ninho e a fonte de alimento.

2.3.1 A ideia do ACO para o problema do menor caminho

O ponto central dos algoritmos baseado em ACO é o modelo de feromônio. Um modo de representar o comportamento das formigas na Figura 1 de forma mais matemática é através o modelo mostrado na Figura 2.

De acordo com a Figura 2 onde cada "trecho" do caminho (aresta do grafo) possui uma distância de $1d$, suponha que 4 formigas partam do ninho em direção ao alimento. Ao chegarem na bifurcação A, como até o momento não temos informações do ambiente, duas formigas vão por ABE e as outras duas por $ACDE$. Quando as formigas que foram por ABE chegarem ao alimento, as que foram por $ACDE$ terão recém-chegado em E. No caminho de volta, as duas formigas que vieram por ABE chegarão em E quando as que foram por $ACDE$ chegarem no alimento. Como ambos os caminhos ainda

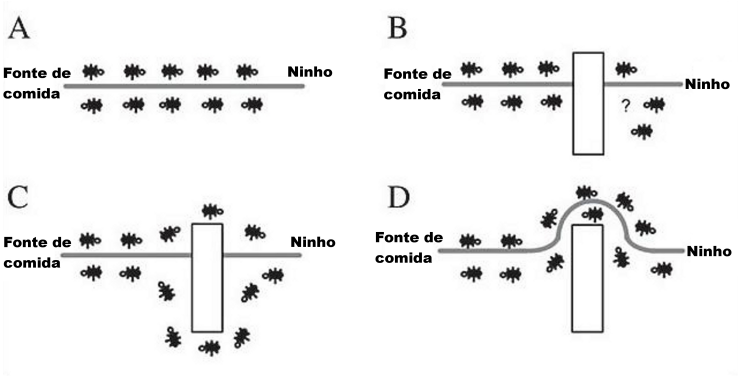

Figura 1: Comportamento das formigas durante a busca por alimento.
Fonte: Adaptada de Perretto e Lopes (2005)

têm a mesma quantidade de feromônio (pois passaram 2 formigas por cada um dos caminhos), cada uma vai por um caminho. Quando elas atingirem respectivamente os pontos B e D as duas formigas que antes pegaram o caminho $ACDE$ terão chegado em E no caminho de volta e cada uma vai por um caminho. Quando a primeira formiga que escolheu o caminho EBA chegar no ninho, a primeira formiga que seguiu o caminho $EDCA$ chegará em A também quando a segunda formiga que pegou EBA chegar em A. Quando mais $1d$ for percorrido, a primeira formiga a retornar ao ninho chegará em A para uma nova busca por comida e como nesse ponto 4 formigas já passaram por ABE e apenas 3 por $ACDE$, é mais provável que ela escolha o caminho ABE. Nesse mesmo momento haverão 2 formigas chegando ao ninho e a formiga que pegou o caminho $ACDE$ tanto para ir quanto para voltar estará chegando em A. Quando mais 1d for percorrido, supondo que a formiga que seguiu 2 vezes o caminho ABE escolheu o caminho com mais feromônio, ela agora está em B. A formiga que escolheu 2 vezes o caminho $ACDE$ terá recém chegando no ninho. As duas que foram por um caminho e voltaram por outro estarão em A e tendo que escolher se vão por ABE com quantidade de feromônio referente a 5 passagens de formigas ou $ACDE$ com 4 passagens.

Com apenas alguns passos, descritos no parágrafo anterior, o menor caminho começa a prevalecer sobre o maior. Ao longo de mais iterações e ainda considerando a evaporação que o feromônio sofre, haverá um momento onde o menor caminho será significativamente mais atrativo e é nessa ideia que o *Ant Colony Optimization* se baseia (DORIGO; GAMBARDELLA, 1997a) (DORIGO; CARO; GAMBARDELLA, 1999).

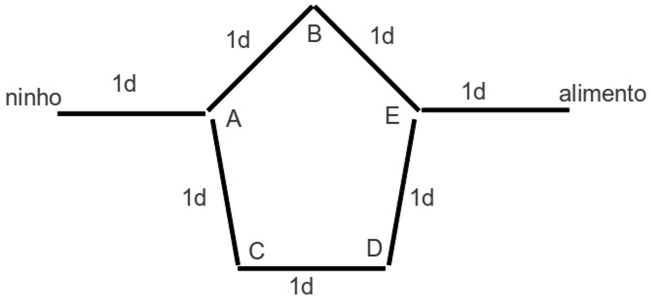

Figura 2: Exemplo de possíveis caminhos entre o ninho e a fonte de alimento.

2.3.2 Algoritmo

Implementações dessa meta-heurística cujo os passos foram mostrados na Subseção 2.3.1 seguem basicamente as instruções do Algoritmo 1. Para exemplificar a lógica deste algoritmo, o problema do menor caminho é usado de exemplo.

Algoritmo 1 ACO meta-heurística

```
f = inicializa_feromonio()
para cada iteração i faça
    Δf ← ∅
    para cada formiga j faça
        s_j = construir_solução(f)
        q_j = avaliar_solução(s_j)
        Δf = Δf + deposita_feromonio(s_j, q_j)
    fim para
    f = (1 − taxa_evaporação) * f + Δf
fim para
```

O método $inicializa\_feromonio()$ do Algoritmo 1 seta um valor inicial igual para todas as arestas dos caminhos. Um valor inicial é necessário para os cálculos durante a escolha que a formiga faz sobre qual caminho seguir. Se fosse usado o valor 0 (zero), seria necessário tratar casos especiais pois como a fórmula da probabilidade de escolha do caminho (mostrada na Seção 2.4, Equação 2.2) utiliza o valor do feromônio do caminho no dividendo e a soma dos feromônios de todos os caminhos possíveis como divisor, o que resultaria

em uma divisão de zero por zero na primeira iteração. Mesmo tratando o caso especial da primeira iteração, para a segunda iteração, somente os caminhos utilizados na primeira iteração teriam valor de feromônio maior que zero, isto resultaria que a probabilidade de escolha de caminhos ainda não utilizados fosse zero e esta não é uma característica desejada no início do algoritmo. Meyer (2004) lembra que para todas as meta-heurísticas, o equilíbrio entre o aprendizado baseado em soluções anteriores (intensificação) e a exploração do espaço de busca (diversificação), é de importância crucial para a eficácia do algoritmo.

A variação do feromônio é acumulada durante a iteração na variável Δf para que seja atribuída somente ao final da iteração. O motivo do feromônio não ser diretamente adicionado a variável f é explicado na seguinte situação: suponha a primeira iteração, onde todas as arestas possuem um mesmo valor inicial de feromônio (geralmente esse valor inicial é baixo), a primeira formiga constrói seu caminho (solução) e já atribui o feromônio a f, quando a segunda formiga for construir seu caminho, suas escolhas já serão influenciadas pelo feromônio depositado pela primeira formiga. Já no caso onde o feromônio só é atualizado ao final da iteração com a variação da substância sendo guardada em Δf, todas as formigas da primeira iteração construirão soluções sem serem influenciadas. Isso possibilita exploração maior do ambiente, já que todas as formigas poderão construir qualquer caminho com a mesma probabilidade na primeira iteração.

Cada formiga constrói um caminho (solução) s_j com base na matriz de feromônios f que guarda o "conhecimento" da colônia de formigas[2]. Os componentes da solução são escolhidos através de uma regra de probabilidade que varia entre diferentes algoritmos baseados em ACO. Esse caminho s_j é avaliado (q_j) e, dependendo do critério de atualização do feromônio, pode ser usado para definir um mesmo valor feromônio que será atribuído a todas as arestas de s_j ou valores diferentes para cada aresta, de modo a dar uma maior recompensa para os menores caminhos[3].

A taxa de evaporação (*taxa_evaporação*) é uma valor entre 0 e 1 que simula a evaporação do feromônio no ambiente em função do tempo. Os valores na matriz de feromônio f são submetidos a essa evaporação e os novos valores gerados pelas formigas durante a iteração (Δf) são agregados a matriz f. A taxa de evaporação é uma variável de grande influência no algoritmo pois permite que caminhos que não são mais usados sejam esquecidos com o tempo. Valores muito baixos para *taxa_evaporação* im-

[2] A matriz de feromônios f guarda os valores referentes a história da colônia. As arestas de caminhos mais visitados possuem maiores valores.

[3] Na Seção 2.4 são mostradas técnicas, tanto que atribuem diferentes valores de feromônio para diferentes caminhos, quanto técnicas onde a recompensa em feromônio é igual independente do quão boa é a solução.

plicam que caminhos, talvez não tão bons, demorem para serem esquecidos e também leva a uma rápida estagnação[4], visto que o feromônio acaba sendo acumulado nos mesmos caminhos iniciais. Por outro lado, valores muito altos para *taxa_evaporação*, apesar de guiarem o algoritmo para uma maior exploração do ambiente, não permitem que seja aproveitado o "conhecimento" da colônia, já que caminhos anteriores são esquecidos muito rápido e talvez deixe de explorar uma boa área do grafo encontrada em uma iteração passada.

2.4 ANT SYSTEM

Ant System foi a primeira implementação de *Ant Colony Optimization* proposta por Dorigo (1992) e foi utilizada para o problema do Caixeiro Viajante, onde a ideia é que cada formiga construa uma solução (sequência de cidades) com base no feromônio depositado no ambiente — que no caso do caixeiro viajante, o feromônio é depositado nas arestas entre as cidades — e do mesmo modo como o exemplo do menor caminho da Subseção 2.3.1, o menor caminho também emerja para o caixeiro viajante.

Dorigo publicou outros trabalhos, de forma mais detalhada, em Dorigo, Maniezzo e Colorni (1996) onde as características do *Ant System* foram melhor identificadas, estudadas e comparadas. Os 3 modos de exploração das características propostos para o AS são:

ant-cycle A definição do valor de atualização do feromônio, ou seja, a quantidade de feromônio que será depositada no caminho, é feita só no final da construção do caminho e as arestas que compõe essa solução recebem a mesma quantidade de feromônio, esta que é inversamente proporcional ao tamanho do caminho total.

ant-density A definição do valor de atualização do feromônio é feita durante a construção do caminho e cada aresta da solução recebe a mesma quantidade da substância baseada em um parâmetro do algoritmo, independente da qualidade da solução.

ant-quantity A definição do valor de atualização também é feita durante a construção do caminho, mas cada aresta recebe quantidade de feromônio inversamente proporcional ao seu próprio comprimento (comprimento da aresta escolhida pela formiga), não considerando o percurso total final do caminho.

[4] A estagnação ocorre quando há muito feromônio acumulado em um caminho e pouco nos outros o que acarreta que esse caminho seja quase sempre escolhido pelas formigas durante a construção da solução e isso sugere que dificilmente novos caminhos serão explorados.

2.4.1 Algoritmo

Uma forma de representar a matriz de feromônio utilizada pela formiga na construção do caminho (solução) é mostrada na Tabela 1. Para este exemplo têm-se 5 cidades em um grafo não direcionado[5]. Note que a tabela é simétrica, pois o valor da coluna *x* com a linha *y* é igual ao valor da coluna *y* com linha *x*, ou seja, estando na cidade *x* a quantia de feromônio referente a escolha da próxima cidade ser *y* é a mesma que se estivesse na cidade *y* e analisasse a quantia de feromônio referente ao caminho que leva a cidade *x*. Entretanto, o AS também poderia ser aplicado a um problema com grafos direcionados, onde haveriam diferentes valores de feromônio se comparasse *x*—*y* com *y*—*x*.

Tabela 1: Possível representação do feromônio no ambiente.

cidade	**1**	**2**	**3**	**4**	**5**
1	0	0,2	0,8	0,3	0,1
2	0,2	0	0,2	0,8	0,9
3	0,8	0,2	0	0,1	0,7
4	0,3	0,8	0,1	0	0,3
5	0,1	0,9	0,7	0,3	0

Com base na Tabela 1 e no grafo completo[6] da Figura 3 — cada círculo da Figura é um vértice que representa uma cidade no problema do caixeiro viajante, os números dentro dos vértices são seus identificadores (cidade 1, cidade 2, ..., cidade 5) e os números próximos as arestas que ligam os vértices representam a distância entre as cidades — a formiga, estando em uma cidade *i*, escolhe a próxima cidade *j* dentre todas as *n* possíveis cidades com base na Equação 2.2 quando só o feromônio é usado na construção ou quando não há informação heurística[7] (*visibilidade* não é usada) (τ_{ij} armazena o valor de feromônio depositado na aresta entre os vértices *i* e *j* — coluna *i* e linha *j* na tabela de feromônio). Ou seja, a formiga estando na cidade 1 tem a possibilidade de ir para a cidade 2 com probabilidade $p_{12} = 0{,}142857143$, ir a 3 com $p_{13} = 0{,}571428571$ e ainda $p_{14} = 0{,}214285714$ e $p_{15} = 0{,}071428571$. Para o problema do caixeiro existe ainda a restrição em relação as cidades já

[5]Em um grafo não direcionado a mesma aresta pode ser usada no sentido *x*—*y* quanto no sentido *y*—*x*

[6]Um grafo completo é um grafo onde existem arestas entre todos os seus vértices.

[7]A heurística é uma informação do ambiente do problema que no ACO pode ser usada para influenciar a construção de uma solução. Para o problema do caixeiro, o valor das arestas (distância entre as cidades) geralmente é utilizado como informação heurística.

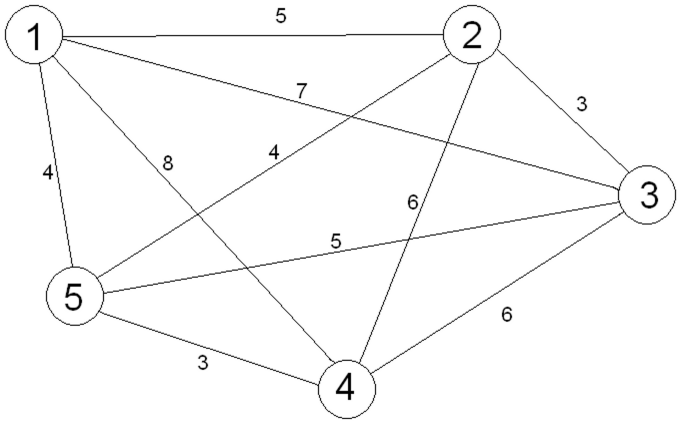

Figura 3: Exemplo de grafo representando o problemas do caixeiro viajante com 5 cidades.

visitadas. O caixeiro só pode passar uma vez por cada cidade por isso j deve pertencer ao conjunto *naoVisitados*.

$$p_{ij} = \frac{\tau_{ij}}{\sum_{k=1}^{n} \tau_{ik}}, \quad \text{onde } j \text{ e } k \in naoVisitados \tag{2.2}$$

Quando há informação heurística para ajudar na escolha da próxima cidade, a Equação 2.3 é utilizada. O termo heurístico η que influencia nos valores de escolha durante a construção da solução é chamado *visibilidade*, ou seja, a visibilidade é um termo de algoritmos ACO que é qualquer informação heurística do ambiente de domínio que possa ser utilizada pelas formigas para saber se um determinado passo é melhor ou pior que outro, associando um valor númerico referente a qualidade desta escolha, mesmo que esse valor seja apenas uma estimativa, não representando a qualidade real da escolha. Os comportamentos de τ e η seguem a seguinte ideia: η diz ao algoritmo que a cidade mais próxima deve ser preferencialmente escolhida, enquanto τ diz ao algoritmo que, se uma aresta tem bastante tráfego, então essa aresta deve ser escolhida com maior probabilidade.

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k=1}^{n} \left(\tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta} \right)}, \quad \text{onde } j \text{ e } k \in naoVisitados \tag{2.3}$$

α e β são exponentes que definem a importância do feromônio e da visibilidade respectivamente. Como os valores de feromônio e visibilidade geralmente estão entre 0 e 1, valores altos de expoente implicam em um menor valor de resultado para aquele termo, consequentemente menor importância. Portanto, se for setado um valor muito baixo para β e alto para α, a visibilidade ganha mais importância e a busca se torna uma busca gulosa[8] onde o "conhecimento" da colônia não é explorado. A Equação 2.4 mostra um exemplo de como pode ser calculado o valor da visibilidade η_{ij} onde a_{ij} é o valor referente ao tamanho da aresta que liga o vértice *i* ao vértice *j* (distância entre as cidades *i* e *j*).

$$\eta_{ij} = \frac{1}{a_{ij}} \tag{2.4}$$

Escolhida a cidade, o processo se repete (agora considerando a nova cidade em que a formiga se encontra e atualizando o conjunto de cidades não visitadas) para que a formiga escolha a próxima. O passo é repetido até que todas as *n* cidades sejam visitadas.

Ao final da construção da solução, a definição do valor de feromônio ao caminho construído pela formiga pode ser feito de 3 modos diferentes como mostrado no início desta seção. Por exemplo, considerando os caminhos $c_1 = 1$–3–5–2–4 e $c_2 = 1$–5–4–2–3, têm-se:

ant-cycle Como os valores totais percorridos dos dois caminhos são respectivamente 22 e 16, um exemplo de atribuição do feromônio seria adicionar $1/22\,(\approx 0{,}05)$ na matriz de feromônio nas arestas que compõem o caminho c_1 e $1/16\,(\approx 0{,}06)$ nas arestas que compõem c_2. Apesar de ambos usarem a aresta entre 2 e 4, a formiga que construiu c_2 depositará mais feromônio nessa conexão do que a formiga que construiu c_1, pois o caminho total final foi menor.

ant-density Nesse modelo, cada uma das arestas que compõem as soluções c_1 e c_2 recebem um valor previamente definido (parâmetro do algoritmo), por exemplo 0,05. Como a aresta entre 2 e 4 foi usada duas vezes, ela acumulará o feromônio e terá 0,1 para ser adicionado a matriz.

ant-quantity O caminho c_1 utiliza as arestas entre 1–3, 3–5, 5–2 e 2–4 que possuem comprimentos 7, 5, 4 e 6 respectivamente. Um exemplo seria depositar o valor $1/7\,(\approx 0{,}14)$ na aresta 1–3, $1/5\,(0{,}2)$ em 3–5,

[8] A busca gulosa (*greedy search*) é uma técnica de busca que visa o próximo aparentemente melhor. Para problemas como o do menor caminho, seta como próximo vértice o nó mais próximo (menor valor de aresta com o vértice atual) (CORMEN et al., 2009).

$1/4\,(0{,}25)$ em 5–2 e $1/6\,(\approx 0{,}17)$ em 2–4. O mesmo processo para as arestas que compõe c_2.

É importante ressaltar que os valores definidos para feromônio durante a iteração são armazenados na matriz auxiliar Δf e só são adicionados a matriz principal de feromônio f ao final da iteração quando ocorre também a evaporação. A Equação 2.5 mostra como é feita a atualização de feromônio ao final da iteração no AS. O feromônio de cada aresta ij sofre a evaporação e é incrementado com os valores acumulados em Δf durante a atual iteração t.

$$\tau_{ij}(t+1) = (1 - taxa\_evaporação)\,\tau_{ij}(t) + \Delta f_{ij} \qquad \text{, sendo } \tau_{ij} \in f \quad (2.5)$$

Diferentes algoritmos baseados em ACO, tais como *Ant Colony System* (ACS) (DORIGO; GAMBARDELLA, 1997b) ou *Max-Min Ant System* (MMAS) (STüTZLE; HOOS, 2000), possuem diferentes técnicas para atualização do feromônio. Além do uso de diversas soluções construídas durante a iteração, algumas outras técnicas usam somente a melhor solução da iteração e outras somente a melhor global para atualizarem o feromônio.

Dorigo, Maniezzo e Colorni (1996) realizaram simulações com diversos valores dos parâmetros para as três implementações do algoritmo com o objetivo de colher dados estatísticos que dissessem quais os melhores valores para os parâmetros. Conhecendo os melhores valores dos parâmetros para cada algoritmo, os métodos foram comparados e concluiu-se que a implementação *ant-cycle* obteve os melhores resultados.

Ant System também foi comparado com outras técnicas como *Simulated Annealing* e *Tabu Search* e os resultados foram melhores que os do *Simulated Annealing* e tão bons quanto os do *Tabu Search* para o problema do caixeiro viajante.

2.4.2 Importância da visibilidade

O uso da visibilidade na meta-heurística ACO tem grande influência no resultado do algoritmo e, para provar essa afirmação, foram realizados testes onde o objetivo era encontrar o menor caminho em instâncias para o problema do caixeiro viajante. A Tabela 2 mostra os valores do menor percurso encontrado por uma implementação de AS quando foi usada visibilidade e quando não foi. Os valores entre colchetes indicam (em média) em qual iteração o melhor valor foi encontrado.

A implementação utilizada foi desenvolvida durante este trabalho e foi

Tabela 2: Importância do uso da visibilidade em problemas discretos.

Nome do problema	Sem visibilidade	Com visibilidade
oliver30	898,72 [559,58]	416,28 [373,92]
eil51	1208,58 [536,28]	443,14 [373,68]
a280	28938,58 [575,34]	2892,92 [490,2]

baseada na descrição de Dorigo, Maniezzo e Colorni (1996).

Todos os valores da Tabela 2 são médias de 50 simulações independentes, limitadas a um máximo de 1000 iterações cada simulação. Os problemas são instâncias encontradas na TSPLIB[9] e os parâmetros usados no algoritmo são apresentados na Tabela 3 onde *numeroDeFormigas* é o número de formigas do algoritmo, o qual foi usado com o valor igual ao número de cidades do problema. Os demais parâmetros são aqueles já descritos na Subseção 2.4.1, onde α, β e *taxa_evaporacao* são, respectivamente, as importâncias do feromônio, da visibilidade e a velocidade de evaporação do feromônio no ambiente. Os valores para todos estes parâmetros (Tabela 3) são os mesmos sugeridos por Dorigo, Maniezzo e Colorni (1996), que por sua vez os obtiveram através de simulações específicas com o objetivo de encontrar os melhores valores destes parâmetros.

Tabela 3: Parâmetros usados no AS ACO de domínio discreto.

Parâmetro do Algoritmo	Valor
α	1
β	5
taxa_evaporacao	0,5
numeroDeFormigas	Número de cidades do problema

Dentre os 3 problemas usados para mostrar a importância da visibilidade, o problema *oliver30* foi o que teve a melhora menos significativa, e mesmo assim o percurso encontrado foi inferior a metade do que sem o uso da visibilidade. Assim, nota-se que quanto maior a complexidade[10] do problema, maior é a importância da visibilidade no algoritmo. O número de iterações até atingir o melhor valor também diminui significativamente, ou seja, o algoritmo converge para uma solução melhor em um tempo inferior quando a visibilidade é usada.

[9]TSPLIB é uma biblioteca de instâncias para o problema do Caixeiro Viajante. Pode ser acessada em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[10]A complexidade é diretamente relacionada com o número de cidades que compõe o problema. *oliver30* tem 30 cidades, *eil51* 51 cidades e *a280* 280 cidades.

Sem o uso da visibilidade, as formigas exploram aleatoriamente várias regiões do ambiente nas primeiras iterações e frequentemente caem em regiões com mínimos locais[11]. A visibilidade guia o algoritmo a boas regiões[12] nas primeiras iterações pois não há informação de feromônio do ambiente. Sendo guiadas para uma boa região, as formigas exploram essa área e acabam encontrando menores tamanhos de percurso em menos tempo do que quando não há o uso da visibilidade.

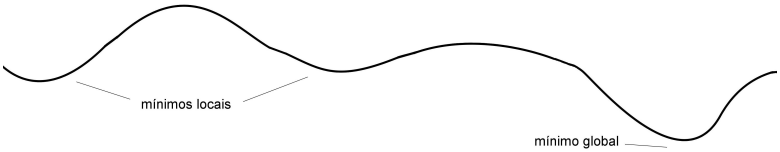

Figura 4: Mínimos locais e mínimo global.

Como citado na seção anterior, existem outras propostas de algoritmos baseados no comportamento das formigas para resolver problemas discretos. Um exemplo é o *Max-Min Ant System* proposto por Stützle e Hoos (2000) onde são definidos valores máximo e mínimo de feromônio que pode conter em uma aresta do caminho. Nesse algoritmo ele tenta evitar a convergência impedindo que um caminho acumule demasiada quantia de feromônio e evitando que outros não usados sejam totalmente esquecidos. No trabalho de Stützle e Hoos (2002) é explorada a técnica usando o problema do caixeiro viajante como estudo de caso.

2.5 PROBLEMAS DE OTIMIZAÇÃO EM DOMÍNIOS CONTÍNUOS

Os problemas apresentados até aqui possuem seus domínios discretos. Exemplos: o item pertence ou não a mochila; a cidade D é visitada após a cidade F; o menor caminho não passa pelo nó C; entre outros. Alguns problemas possuem o domínio dos valores que as variáveis podem assumir

[11] Um mínimo local é um ponto do domínio onde a função objetivo retorna menor valor do que para seus vizinhos, mas existe algum outro ponto do domínio com melhor resultado da função objetivo. Ver Figura 4.

[12] Boas regiões são caminhos do grafo com menor tamanho de percurso.

dentro de um intervalo contínuo. Exemplo: encontrar os valores de x e y que maximizem z na Equação 2.6, onde x e y podem assumir qualquer valor real entre -100 e 100.

$$z = \cos x \cdot \cos y \cdot \exp\left(-((x-\pi)^2 + (y-\pi)^2)\right) \tag{2.6}$$

O valor ótimo da Equação 2.6 é $z = 1$ sendo que os valores para as variáveis alcançarem o melhor global são $x = \pi$ e $y = \pi$ (ponto cinza da Figura 5B – valor 3,1415...). Os algoritmos tradicionais de *Ant Colony Optimization* só lidam com esse tipo de problema se os valores das variáveis puderem ser discretizados em um conjunto finito de possíveis valores, como no exemplo da Figura 5A, que mostra uma possível discretização de um intervalo de -5 até 5 em 11 possíveis valores. Supondo que esse ambiente exemplo (Figura 5A) seja o proposto para resolver a Equação 2.6, então o valor mais próximo possível do ótimo para x e y é 3, o que resultaria um valor de $z = 0{,}942$ que é distante 0,058 do valor ótimo. Assim, buscando resolver esse tipo de problema, os algoritmos de ACO foram extendidos para lidar diretamente com valores contínuos.

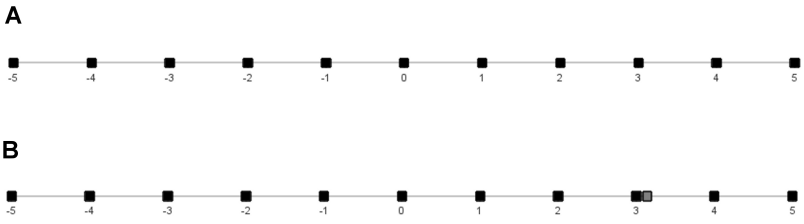

Figura 5: Exemplo de discretização do intervalo $[-5,5]$. Em *A* o intervalo está discretizado em 11 diferentes valores. Em *B* o ponto cinza indica a posição do valor ótimo para a Equação 2.6.

Socha e Dorigo (2006) lembram que discretizar variáveis contínuas nem sempre é possível, pois o intervalo pode ser muito amplo ou a resposta pode exigir uma grande precisão. Se a solução ótima for incorporada nos espaços ignorados entre os valores possíveis, ela não terá chance de ser escolhida. Assim, novas técnicas baseadas em *ACO* foram propostas de modo a resolver esta classe de problemas.

A primeira implementação de *ACO* aplicada diretamente para domínios contínuos, o *Continuous ACO* (CACO), foi proposta por Bilchev e Parmee (1995) onde é setado um valor inicial (ninho) em algum lugar no domínio e

a partir desse ponto as formigas andam para algumas direções pré-definidas dentro de um raio R a cada passo. As melhores soluções são salvas como um conjunto de vetores que iniciam no ninho. Ao escolher um vetor, a formiga continua a busca a partir do ponto ao qual o vetor o levou realizando alguns passos aleatórios. Novamente o conjunto de vetores é atualizado com os melhores encontrados. Existem outras ideias para esta classe de problemas, como a propostas por Zhao, Wang e Xie (2008), além de Jalali, Afshar e Mariño (2006) que utilizaram um sistema multi colônia com discretização heterogênea e troca de informação entre as colônias para que fosse possível encontrar a solução ótima em um domínio contínuo, ou Monmarchè, Venturini e Slimane (2000) que propuseram o *Asynchronous Parallel Implementation* (API), onde as formigas constroem suas soluções independentemente, partindo do mesmo ninho (que é movido periodicamente). No *Continuous Interacting Ant Colony* (CIAC) Dréo e Siarry (2002) as formigas se comunicam através do feromônio e também por comunicação direta. As formigas são atraídos pelo feromônio e guiados pela comunicação com outras formigas. Socha e Dorigo (2006) dizem que estas técnicas não seguem os conceitos clássicos de ACO, pois algumas não utilizam o conhecimento da colônia e soluções construídas incrementalmente, ou adicionam alguma característica que difere da ideia original. Socha propõe um método mais similar aos conceitos introduzidos por Dorigo.

O algoritmo $ACO_{\mathbb{R}}$ proposto por Socha (2004) está entre os modelos mais referenciados em trabalhos nessa área. A implementação se mostrou uma boa técnica em diversos problemas contínuos de otimização. Socha ainda aplicou seu algoritmo em outros trabalhos juntamente com Dorigo e outros pesquisadores e sua ideia para representação do feromônio pode ser considerada o estado da arte para este domínio (SOCHA; BLUM, 2006) (SOCHA; DORIGO, 2007) (SOCHA; DORIGO, 2006).

Existem ainda outros trabalhos nessa área como Mathur et al. (2000) que se baseia na pesquisa de Bilchev e Parmee (1995), os trabalhos de Kong e Tian (2005), Kong e Tian (2006a), Kong e Tian (2006b), Huang e Hao (2006) e Madadgar e Afshar (2008) que se baseiam no proposto por Socha (2004). O conceito para a representação do feromônio proposto por Socha é o modelo no qual a presente pesquisa se baseia e mais detalhes sobre a implementação proposta é apresentada na Seção 2.6.

2.6 $ACO_{\mathbb{R}}$

Proposto por Socha (2004), $ACO_{\mathbb{R}}$ é a versão de ACO para domínios contínuos que mais se aproxima da ideia original proposta por Dorigo (1992)

para problemas discretos.

2.6.1 Representação do Feromônio

A ideia principal do $ACO_{\mathbb{R}}$ é o uso de uma função de densidade de probabilidade onde um valor para a variável contínua é escolhido a partir de uma amostragem probabilística do domínio. A Figura 6 representa a proposta de feromônio onde, em 6a é mostrada a probabilidade de cada componente em um ambiente discreto com base na matriz de feromônio de um ACO clássico e, a medida que o número de componentes vai aumentando, distribuição discreta se aproxima da Figura 6b, que mostra a mesma distribuição de probabilidade mas considerando o domínio contínuo.

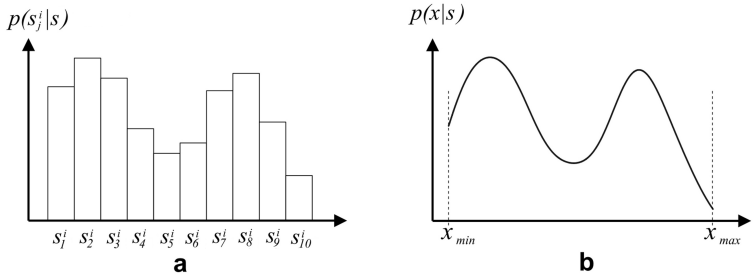

Figura 6: Distribuição de probabilidade. Em **a** a distribuição é discreta entre os componentes do domínio. Em **b** a distribuição é contínua por todo o intervalo.

Adaptado de (SOCHA; DORIGO, 2006).

A distribuição de probabilidade representa o feromônio no ambiente com base nas informações da matriz de feromônio em um algoritmo de ACO discreto e do *arquivo população* de um ACO contínuo. Os algoritmos de ACO discretos acumulam mais feromônio nos melhores caminhos construídos pelas formigas, ou seja, o feromônio é acumulado em soluções já construídas em passos anteriores. O ACO contínuo se assemelha a esse fato pois os valores representados com maior probabilidade são referentes as melhores soluções encontradas pelas formigas. O arquivo população no ambiente contínuo é um conjunto de soluções que, juntas, representam a ideia da Figura 6.

A função de densidade de probabilidade usada nas amostragens do algoritmo é a função gaussiana (distribuição normal), entretanto ela não é capaz de descrever a situação onde há duas áreas promissores disjuntas no

domínio. Assim, a escolha do valor de uma variável ocorre com apenas um valor usado como média e um desvio padrão em cada amostragem. Os valores são gerados probabilisticamente como no exemplo da Figura 7 onde os valores mais próximos da média μ tem maior probabilidade de serem amostrados e essa probabilidade diminui a medida que se afasta de μ. Contudo, como esse valor que será usado como média é probabilisticamente escolhido da população, a representação da Figura 6 é válida. O valor usado como média para a distribuição de probabilidade é o valor escolhido para a variável de uma solução já construída pelas formigas em passos anteriores[13]. O arquivo população se comporta de modo que quanto mais uniforme é o conjunto de soluções, mais concentrado é o feromônio em um valor. Por outro lado, quanto mais dispersas as soluções construídas pelas formigas, mais igualmente distribuído é o feromônio e mais uniforme é a escolha de valores para as variáveis na construção de novos caminhos. Esse comportamento é similar ao que ocorre em algoritmos de ACO discreto.

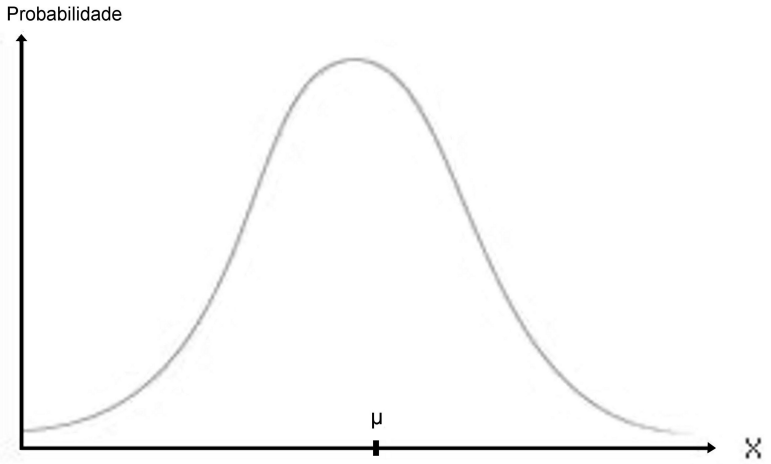

Figura 7: Exemplo de distribuição de probabilidade de uma função gaussiana (normal).

Algumas linguagens de programação dispõem em suas bibliotecas métodos para gerar valores com distribuição normal. Dentre elas, algumas utilizam o método proposto por (BOX; MULLER, 1958) que pode ser implementado a partir de um método de geração de número aleatórios com probabilidade

[13] Mais detalhes sobre o processo de amostragem são apresentados na Subseção 2.6.2.

uniforme.

A atualização de feromônio ocorre ao final de cada iteração. Novas soluções são adicionadas ao arquivo população e as piores são removidas de forma que há um número fixo de soluções que são mantidas. Algumas soluções permanecem no arquivo até que as formigas encontrem um número suficiente de soluções melhores para que as antigas sejam removidas.

Nas primeiras iterações do algoritmo é ideal que o valor de desvio padrão seja um valor alto para permitir que sejam amostrados valores distantes da média μ, para isso é necessário que o arquivo população seja o mais heterogêneo possível. Com o passar das iterações esse desvio padrão deve diminuir quando a média μ se aproximar de uma boa região para que os valores próximos possam ser explorados, para isso, o arquivo população precisa passar a ser mais homogêneo. É esse o comportamento que emerge no algoritmo.

2.6.2 Passos do Algoritmo

O valor utilizado como média μ é o valor s_j^i com $j = 1,2,\ldots,k$ onde k é o número de soluções mantidas pelo algoritmo durante a execução (número de soluções no arquivo população) e $i = 1,2,\ldots,n$ sendo n o número de variáveis do problema. Assim, pode-se ter mais de uma média para cada variável.

Cada solução s_j ($j = 1,2,\ldots,k$) possui um valor de média para cada uma de suas n variáveis. O arquivo população com essas soluções que o algoritmo mantém durante a simulação pode ter 1 ou mais soluções[14] ($k \geq 1$). A Tabela 4 mostra a estrutura do arquivo de soluções.

Tabela 4: Estrutura do arquivo de soluções.

s_1	s_1^1	s_1^2	$\ldots$	s_1^i	$\ldots$	s_1^n
s_2	s_2^1	s_2^2	$\ldots$	s_2^i	$\ldots$	s_2^n
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
s_j	s_j^1	s_j^2	$\ldots$	s_j^i	$\ldots$	s_j^n
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
s_k	s_k^1	s_k^2	$\ldots$	s_k^i	$\ldots$	s_k^n

Inicialmente, k soluções são geradas com valores aleatórios dentro do

[14] k é um parâmetro do algoritmo cujo valor corresponde ao número de soluções que são mantidas no arquivo.

domínio das variáveis e adicionadas ao arquivo de soluções (população). O arquivo é então ordenado de forma que a primeira solução seja a com maior valor da função objetivo em problemas de maximização[15] ou com menor valor da função objetivo em problemas de minimização[16].

O vetor população (arquivo de soluções) funciona para o algoritmo como um conjunto de probabilidades que soma diversas funções gaussianas de uma dimensão em uma nova distribuição, como representado na Figura 8 para cinco soluções e uma variável a ser amostrada.

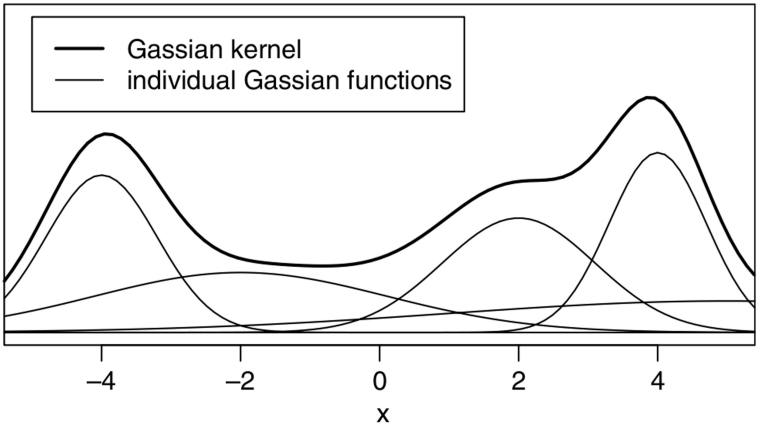

Figura 8: Exemplo de cinco funções Gaussianas e a combinação de probabilidades resultante.
Adaptado de (SOCHA; DORIGO, 2006).

Cada função gaussiana individual tem diferente importância na combinação resultante das probabilidades. Melhores soluções têm maior influência e a probabilidade resultante é definida pela Equação 2.7 proposta por Socha e Dorigo (2006). Na equação, G^i é a combinação de probabilidades para a variável i do problema. ω_j é um valor atribuído a solução j do arquivo, onde as soluções melhor classificadas têm maior valor ω ($\omega_1 \geq \omega_2 \geq \ldots \geq \omega_k$). $g^i_j(x)$ é a gaussiana individual e σ^i_j é o valor definido como desvio padrão para i-ésima variável da solução s_j.

[15] $f(s_1) \geq f(s_2) \geq \ldots \geq f(s_k)$.
[16] $f(s_1) \leq f(s_2) \leq \ldots \leq f(s_k)$.

$$G^i(x) = \sum_{j=1}^{k} \omega_j g_j^i(x) = \sum_{j=1}^{k} \omega_j \frac{1}{\sigma_j^i \sqrt{2\pi}} e^{-\frac{\left(x-\mu_j^i\right)^2}{2\sigma_j^{i2}}} \tag{2.7}$$

Como dito na Subseção 2.6.1, a função gaussiana não suporta uma amostragem onde existem mais que uma área disjunta com alta probabilidade de serem amostradas e então o processo é feito em partes. Assim, a cada iteração, cada formiga, no processo de construção de uma nova solução X, primeiramente escolhe uma solução s_l do arquivo com probabilidade p_l definida na Equação 2.8 e essa solução escolhida, com seus n valores para as n dimensões do problema, será usada para as médias na amostragem de uma nova solução X.

$$p_l = \frac{\omega_l}{\sum_{j=1}^{k} \omega_j}, \quad l \in [1,2,\ldots,k] \tag{2.8}$$

Na Equação 2.8, p_l é a probabilidade da solução s_l ser escolhida e ω_j é calculado para cada solução do arquivo[17] segundo a Equação 2.9, onde q é uma variável do algoritmo onde baixos valores de q implicam que as melhores soluções terão maior probabilidade de serem escolhidas. Por outro lado, altos valores de q implicam que as soluções serão escolhidas com probabilidades mais uniformes. Altos valores para q também implicam em uma busca mais diversificada e é mais confiável que a solução encontrada no final seja a solução ótima. Entretanto, o alto valor também implica em uma menor velocidade de convergência para o algoritmo. Uma análise mais detalhada sobre o parâmetro q é realizada na Subseção 4.1.1 quando alguns valores de q são propostos para as simulações.

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(j-1)^2}{2q^2k^2}} \tag{2.9}$$

A solução escolhida s_l terá seus valores s_l^i com $i = 1,2,\ldots,n$ utilizados como média para cada uma das n variáveis da nova solução $X_i \in X$ que será amostrada pela formiga. Para a amostragem, também é necessária a definição de um valor para o desvio padrão da função gaussiana. A Equação 2.10 mostra como esse valor é obtido (SOCHA; DORIGO, 2006).

[17] Lembrando que $l = 1$ para a melhor solução do arquivo, $l = 2$ para a segunda melhor, até $l = k$ para a pior solução do arquivo.

$$\sigma_l^i = \xi \sum_{j=1}^{k} \frac{\left| s_j^i - s_l^i \right|}{k-1} \tag{2.10}$$

Na Equação 2.10 σ_l^i é o valor do desvio padrão que será utilizado com a média s_l^i e ξ é um parâmetro do algoritmo que limita o tamanho do desvio padrão e funciona de maneira semelhante a taxa de evaporação no ACO clássico pois altos valores de ξ implicam uma convergência mais lenta do que com baixos valores de ξ. Este processo se repete para cada variável $i = 1, \ldots, n$ da nova solução, pois cada uma tem seu próprio valor de desvio padrão que é usado junto com o valor de média s_l^i na amostragem de X_i.

Como inicialmente o vetor população é preenchido com soluções criadas aleatoriamente, é provável que os valores definidos pela Equação 2.10 sejam mais altos nas primeira iterações do algoritmo. A medida que o algoritmo encontra uma boa região, as soluções amostradas dentro dessa região, além de possuirem valores próximos, terão melhores resultados da função objetivo e farão parte das k soluções que compõe o arquivo. Como o vetor população só mantém as k melhores soluções, as demais são removidas e então, por ter soluções mais semelhantes no arquivo, o cálculo do desvio padrão passará a retornar menores valores. A Figura 9 exemplifica esse comportamento descrito.

A Figura 9 mostra um exemplo com um arquivo população contendo 5 soluções que convergem com o passar das iterações. Em 9A a população está distribuída como no início no algoritmo, quando as soluções são definidas aleatoriamente dentro do domínio. Em 9B a população, com o passar das iterações, vai convergindo em direção a melhor região. Esse comportamento faz com que o valor de desvio padrão seja maior nas primeiras iterações e vá diminuindo a medida que a população se torna mais homogênea.

Ao final da iteração, quando as m formigas construíram suas soluções, estas m soluções são adicionadas a população e o mesmo número m de soluções é removido. As soluções removidas são as com piores valores para a função objetivo. Note que a população sempre mantém o mesmo número k de soluções em seu vetor.

O ciclo se repete para a próxima iteração até que um número especificado de iterações seja atingido ou, quando se conhece o valor ótimo, até que certa precisão com um erro admitido seja alcançada.

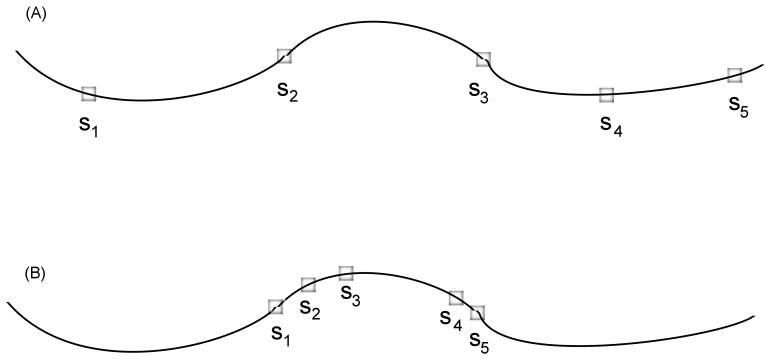

Figura 9: Exemplo de arquivo população com 5 soluções convergindo com o passar das iterações. Em (A) a população está distribuída como no início no algoritmo. Em (B) a população, com o passar das iterações vai convergindo em direção a melhor região.

2.7 CONSIDERAÇÕES FINAIS SOBRE A TÉCNICA ESCOLHIDA

Dadas as diversas técnicas de otimização, a opção pela escolha de uma ligada a otimização por colonia de formigas se deu devido ao fato das técnicas baseadas em ACO para domínio contínuo não explorarem uma característica de grande importância que é geralmente encontrada em técnicas de ACO para o domínio discreto. Essa característica é a visibilidade e sua importância foi provada no experimento apresentado na Subseção 2.4.2.

Dentre os algoritmos de ACO para domínio contínuo, o $ACO_{\mathbb{R}}$ foi escolhido pela maior semelhança com a ideia original de otimização por colonia de formigas proposta por Dorigo (1992). Além disso, Socha e Dorigo (2006) realizam uma boa comparação do $ACO_{\mathbb{R}}$ com outras técnicas baseadas em ACO para domínio contínuo e apresentaram bons resultados, o que poderia caracterizar a técnica como o estado da arte para este domínio e nesta meta-heurística.

Com base nas considerações levantadas até aqui, no próximo capítulo é interpretada o que seria a visibilidade no domínio contínuo de uma maneira similar ao que é no domínio discreto. Além disso, busca-se melhores resultados com significância semelhante a apresentada no ACO discreto quando a visibilidade é usada e quando não é. Estes resultados da comparação são

apresentados no Capítulo 4 onde também é feita uma análise em relação ao tempo de execução, pois não é vantajoso obter melhores resultados se o tempo demandado para isso for relativamente superior. A ideia é que os melhor resultados sejam encontrados em um tempo de execução equivalente ou menor, além da aceleração da velocidade de convergência do algoritmo, no que diz respeito ao número de iterações, em boas soluções dos problemas que serão testados.

3 PROPOSTAS DE TERMO VISIBILIDADE NA OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS EM DOMÍNIOS CONTÍNUOS

Partindo da ideia de que a visibilidade acelera a convergência de algoritmos baseados em ACO em boas regiões para domínios contínuos, assim como faz para domínios discretos, neste capítulo são sugeridas duas versões de termos heurísticos para serem utilizadas como visibilidade em ACO para problemas de domínio contínuo. A primeira proposta de visibilidade possui uma fórmula com grande similaridade com o clássico apresentado por Dorigo, Maniezzo e Colorni (1996) no *Ant System* e que foi apresentado na Seção 2.4. A segunda proposta também usa o conceito de que a visibilidade guia o algoritmo para boas regiões, mas a ideia é utilizada de forma mais abstrata.

A ideia do uso de ambas as visibilidades é que o comportamento da Figura 10 seja aplicado ao algoritmo, onde as soluções s_j, $j = 1,2,3$, são as soluções geradas pelas formigas a partir da amostragem e s'_j são as soluções vizinhas encontradas pela busca local[1] na vizinhança de s_j. A Figura 10a mostra um exemplo com as soluções s_j e suas respectivas buscas locais s'_j que são possíveis soluções a serem escolhidas para servirem de base na amostragem de uma nova solução X. Em 10b tem-se as probabilidades das soluções serem escolhidas com base somente na solução gerada pela formiga (s_j). É possível notar que s_3 apresenta maior chance de ser escolhida por ser a melhor das 3 soluções, apesar de já estar próxima de um máximo local e não oferecer a possibilidade de que um valor significativamente melhor seja encontrado na nova amostragem (área não promissora). Entretanto, as soluções s_1 e s_2, apesar de possuírem valores não tão bons quanto s_3 para a função objetivo, estão em áreas mais promissoras com possibilidade de encontrar um ponto de melhor valor, mas estas soluções recebem poucas chances de serem escolhidas pela distribuição de probabilidade original do $ACO_{\mathbb{R}}$. Em 10c as novas probabilidades, com a modificação da equação incluindo a influência das soluções resultantes da busca local (s'_j), dado o fato de uma determinada solução s_j possuir um vizinho s'_j significativamente melhor, aumentado a probabilidade de soluções promissoras serem usadas em uma nova amostragem.

Ao realizar simulações com o $ACO_{\mathbb{R}}$ original, quando testados diversos parâmetros para o algoritmo a fim de obter ideias para uma melhor exploração da técnica, foi possível notar que diminuir o valor do desvio padrão leva a melhores resultados, mas por outro lado, na maioria das simulações, o algoritmo atinge o número máximo de iterações sem alcançar a precisão especificada. Isso se deve ao fato de que, com maiores valores de desvio padrão,

[1] O algoritmo para busca local é apresentado na Seção 3.4.

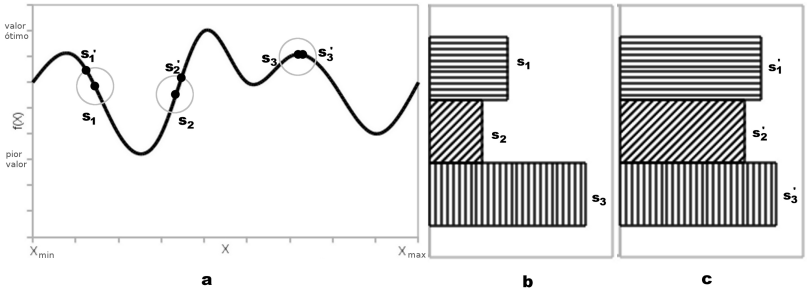

Figura 10: Busca na vizinhança.

valores mais distantes da média podem ser amostrados com maior probabilidade e consequentemente o algoritmo acaba sempre atingindo a precisão necessária mesmo que leve mais iterações até realizar a tarefa. Com menores valores para o desvio padrão, o algoritmo amostra novos valores próximos da média o que muitas vezes impede que ele explore regiões distantes da distribuição aleatória inicial, mas faz com que, ao estar em uma boa região, explore seus vizinhos. Assim, ele explora mais a área em que está mas deixa de explorar novos territórios. Desta forma, pode-se dizer que baixos valores de desvio padrão só são desejáveis quando as soluções no arquivo população estão na região do máximo global, o que nem sempre é fácil identificar.

A ideia de utilizar uma busca local na vizinhança da solução vem da observação apresentada no parágrafo anterior. Assim, um valor é definido para o desvio padrão de modo que permita explorar diversas áreas e, para cada solução amostrada, uma busca em sua vizinhança em um raio fixo é aplicada como heurística de visibilidade para o algoritmo nas propostas de extensão apresentadas nesta pesquisa. Pretende-se assim, aproveitar as características desejáveis de quando são usado altos valores de desvio padrão, que é a exploração de mais regiões, e também quando são usados baixos valores, que é a exploração na vizinhança das soluções que estão no arquivo.

O comportamento normal do algoritmo já faz com que, ao passar das iterações, o desvio padrão vá diminuindo devido a homogeneização do arquivo população, pois o desvio padrão proposto por Socha e Dorigo (2006), apresentado Equação 2.10 da Subseção 2.6.2, se baseia na distância entre as soluções do arquivo no domínio para definir o valor, de modo que quanto mais heterogêneo o arquivo população, maior é o valor do desvio padrão[2]. Porém,

[2]Vale lembrar que a equação do desvio padrão também tem um parâmetro que é definido para toda a execução do algoritmo e que é multiplicado a essa diferença entre as soluções, podendo

o uso de uma busca local permite que uma exploração da vizinhança seja realizada nos primeiros passos do algoritmo, antes que as soluções da população convirjam para uma mesma região. Com esse procedimento é possível acelerar a velocidade de convergência do algoritmo, encontrando as melhores soluções com uma quantidade significativamente menor de iterações.

Com exceção do modo de uso da heurística de visibilidade na geração de uma nova solução, os demais passos do algoritmo são os mesmos para ambas as propostas. Nas seções a seguir são apresentados os passos comuns as propostas e suas características individuais.

3.1 INICIALIZAÇÃO DO ALGORITMO

A inicialização do algoritmo é feita do mesmo modo que ocorre no $ACO_{\mathbb{R}}$ original. São geradas k soluções, onde para cada solução, são gerados valores aleatórios para as variáveis que compõem a solução. Estes valores são gerados com probabilidade uniforme dentro do domínio de possíveis valores referente a cada variável.

As k soluções são armazenadas em um vetor chamado de arquivo população. Dentro desse vetor as soluções são mantidas ordenadas de forma decrescente em relação a qualidade da solução, ou seja, s_1 é a melhor solução e ocupa a primeira posição do vetor, enquanto s_k é a pior das k soluções e ocupa a última posição do vetor. A qualidade de uma solução s_j é a saída da função objetivo (que se deseja otimizar) utilizando como valores das variáveis do problema os valores atribuídos nas variáveis da solução.

3.2 PRIMEIRA PROPOSTA PARA VISIBILIDADE

A primeira proposta para a heurística usada como termo visibilidade do ACO para domínios contínuos se baseia na semelhança com a visibilidade proposta no Algoritmo *Ant System* (AS) de ACO para domínios discretos proposta por Dorigo (1992).

O cálculo de probabilidade da solução s_l a ser escolhida difere da forma que é feito por Socha (2004) (visto na Equação 2.8) e é apresentado na Equação 3.1 onde $f(s_j)$ é a saída da função objetivo para para a solução s_j. Da mesma forma, $f(s'_j)$ é a saída da função objetivo para para a solução s'_j.

assim limitar o valor.

$$p_l = \frac{\omega_l^{\alpha} \cdot \left[f\left(s'_l\right) - f\left(s_l\right) \right]^{\beta}}{\sum_{j=1}^{k} \omega_j^{\alpha} \cdot \left[f\left(s'_j\right) - f\left(s_j\right) \right]^{\beta}} \tag{3.1}$$

Assim como ocorre no ACO clássico, α e β controlam a importância do feromônio e da visibilidade respectivamente. No que diz respeito à heurística que representa a visibilidade, o fato de usar a diferença entre $f(s_j)$ e $f(s'_j)$ se baseia na ideia de que, se há um vizinho s'_j muito melhor que s_j então essa é uma área promissora e mais desejável de ser explorada. Por outro lado, se s'_j for equivalente a s_j, então provavelmente a solução já atingiu ou está próxima de atingir um ponto de máximo local ou global. A intenção é que se dê mais chances a áreas mais promissoras, como foi ressaltado na descrição da Figura 10.

Após escolhida a solução s_l, os seus valores para cada dimensão do problema servirão de média na amostragem dos n valores de uma nova solução X. Diferentemente do $\text{ACO}_{\mathbb{R}}$ original, a amostragem nessa proposta é unilateral, ou seja, ou será amostrado um valor maior que a média, ou será amostrado um valor menor. Para cada novo valor X_i com média s_l^i, considerando o desvio padrão obtido através do cálculo da Equação 2.10, a amostragem será unilateral a esquerda ($X_i \leq s_l^i$) se $s_l^{i'} \leq s_l^i$ ou unilateral a direita ($X_i \geq s_l^i$) se $s_l^{i'} > s_l^i$. Assim, pretende-se forçar que o novo valor seja amostrado seja na direção da solução encontrada pela busca local (s'_l).

Cada solução s_j ($j = 1,2,\ldots,k$) é organizada em uma estrutura de forma que, além dos valores s_j^i ($i = 1,2,\ldots,n$), guarde também os valores da solução s'_j como sendo informação extra, e que será usada a cada novo processo de construção na escolha de s_l para amostragem de uma nova solução X; em outras palavras $s'_j \subset s_j$. Assim, ao final da amostragem unilateral de todas as variáveis de X, a busca local é aplicada gerando uma solução X' que é um vizinho melhor ou igual a X (assim como s'_j é para s_j) e que é armazenado como informação extra da solução X. Essa nova "dupla-solução" X é adicionada a um vetor temporário enquanto outras formigas geram suas soluções. Ao final da iteração, quanto todas as m formigas construíram suas soluções, essas m soluções são movidas para o vetor população[3] e o mesmo número m de piores soluções são excluídas, de forma que o arquivo população sempre mantém somente as k melhores.

[3] Lembrando que o vetor de população sempre se mantém ordenado pela qualidade da solução.

3.3 SEGUNDA PROPOSTA PARA VISIBILIDADE

A segunda proposta da visibilidade insere a influência da visibilidade no algoritmo de um modo mais abstrato. O processo de construção de uma nova solução ocorre como definido no $ACO_{\mathbb{R}}$ original. Não existem mudanças na probabilidade de escolha (usa a probabilidade definida na Equação 2.8), nem na forma de amostragem (usa distribuição gaussiana normal bilateral).

Ao final do processo de construção, após serem atribuídos os valores das n variáveis de X, a busca local é executada retornando uma solução X', como acontece para a primeira proposta de visibilidade. A diferença é que essa nova solução X' é considerada uma solução normal e separada de X, elas não fazem parte da mesma estrutura. Ou seja, para cada formiga, em cada iteração, duas novas soluções são construídas, X e X'.

As soluções geradas pelas formigas são mantidas em um vetor auxiliar até que todas as formigas tenham executado a tarefa. Ao final da iteração, as $2m$ novas soluções geradas pelas m formigas são movidas deste vetor auxiliar para o vetor população (que estava com k soluções armazenadas), onde todas as $k+2m$ soluções são ordenadas e as $2m$ piores soluções são descartadas do arquivo. Com esse comportamento, garante-se que o arquivo população sempre mantém as k melhores soluções armazenadas.

Como as soluções resultantes da busca local são inseridas na população, seria como se o fator visibilidade ficasse implícito no algoritmo no momento em que X' tornou-se uma das soluções s_j, $j = 1,2,\ldots,k$. Se as soluções resultantes da busca local forem melhores que as construídas pelas formigas[4] elas terão maiores chances de serem escolhidas para servirem de base para uma nova solução que será construída em uma iteração futura, pois estarão melhor classificadas no arquivo população.

3.4 ALGORITMO PARA VISIBILIDADE

A implementação proposta neste trabalho para ser usada como busca local na vizinhança de uma solução é apresentada no Algoritmo 2. Este mesmo método é usado nas duas propostas de visibilidade apresentadas neste capítulo, sendo responsável por encontrar uma solução X' melhor ou igual[5] a uma solução X passada como entrada do algoritmo.

No Algoritmo 2, X é a solução gerada pela formiga através da amos-

[4] As soluções s'_j sempre serão melhores ou iguais a s_j, pois se não houver um vizinho melhor, s'_j será igual a s_j.

[5] Caso a solução X seja um máximo global ou máximo local em uma área maior que a coberta pela busca local, a solução X' será igual a X.

Algoritmo 2 Cálculo do Termo Visibilidade - Busca Local na Vizinhança.

$X' \leftarrow X$
$\Delta x \leftarrow 1$
enquanto $\Delta x \geq 0{,}0001$ **faça**
  $i \leftarrow 1$
  enquanto $i \leq n$ **faça**
    $X_{passoFrente} \leftarrow X'$
    $X_{passoAtras} \leftarrow X'$
    $X_{passoFrente}[i] \leftarrow X'[i] + \Delta x$
    $X_{passoAtras}[i] \leftarrow X'[i] - \Delta x$
    $X' \leftarrow melhorDe(X', X_{passoFrente}, X_{passoAtras})$
    $i \leftarrow i + 1$
  fim enquanto
  $\Delta x \leftarrow \Delta x \div 10$
fim enquanto

tragem durante a etapa de construção da solução e é utilizada como entrada no algoritmo da busca local. A variável Δx é o valor referente à diferença, somada (passo à frente) e subtraída (passo atrás), para cada variável da solução original que será analisada na busca local. O valor de Δx começa em 1 e a cada ciclo é dividido por 10 até que seja menor que 10^{-4}. O valor de Δx poderia ser variado em mais valores intermediários (uma divisão por um número menor que 10) ou um diferente intervalo para os possíveis valores (começar com um valor maior que 1 e/ou terminar com um valor menor que 10^{-4}), entretanto isso significaria uma maior quantidade de ciclos do algoritmo, portanto essa escolha precisa ser balanceada dependendo da necessidade. Contudo, para os problemas testados, estes valores definidos através de simulações de teste resultaram em uma melhora na saída, sem que fosse demandado aumento na quantidade de tempo de execução. Assim, com base nessa definição por simulações prévias, para cada variável da solução, são analisados valores vizinhos distantes até um máximo de 1,1111 do valor original, pois Δx começa em 1 e vai até 0,0001. Encontrar um algoritmo diferente para buscar o melhor vizinho, que retorne bons valores com menos instruções, é citado na conclusão (Capítulo 5), na parte de perspectiva para trabalhos futuros. $X_{passoFrente}$ é uma busca local para a solução X através de valores somados aos valores gerados pela formiga para uma variável, enquanto que $X_{passoAtras}$ é uma busca local através de valores subtraídos dos gerados pela formiga na construção. $X'[i]$ é a i-ésima variável da solução X'. O método *melhorDe* retorna a melhor dentre as 3 soluções passadas como parâmetro, ou seja, aquela com maior valor da função objetivo.

Nas primeiras iterações do Algoritmo 2, a variável Δx possui seu maior valor, o que faz que inicialmente a busca local procure em uma vizinhança relativamente grande. A cada ciclo, a variável Δx diminui, realizando um ajuste fino na busca pelo melhor vizinho. Ao final deste algoritmo, X' será a solução com o melhor valor objetivo para a área onde realizou-se a busca local com uma diferença de até 1,1111 para mais ou para menos no valor de cada variável da solução se comparada com a original X. O valor de retorno da função objetivo para X' depende de quão promissora é a área em que X se encontra.

Como dito na Seção 3.2, na primeira proposta de visibilidade X' é usado como informação extra na estrutura da solução X (as duas soluções estão em um único objeto), que ao final da iteração, é adicionada ao arquivo população. Situação diferente do que acontece na segunda proposta de visibilidade, explicado na Seção 3.3, onde X' é considerada uma solução independente e ambas X e X' são adicionadas individualmente ao arquivo população ao final da iteração, quando todas as formigas tiverem construído suas soluções. Assim, a primeira proposta de visibilidade gera m novas soluções a cada iteração (uma solução provinda de cada uma das m formigas), enquanto que a segunda proposta de visibilidade gera $2m$ novas soluções a cada iteração (duas soluções geradas por cada uma das m formigas). Contudo, ainda no final de cada iteração, todas essas soluções são ordenadas pela qualidade e as m piores (visibilidade 1) ou as $2m$ piores (visibilidade 2) são descartadas, permanecendo somente as k primeiras/melhores para a próxima iteração.

3.5 TEMPO DE EXECUÇÃO DO ALGORITMO

Apesar de ambas as propostas de visibilidade realizarem mais passos por iteração do que o $ACO_{\mathbb{R}}$ original, o fato de que a solução ótima é encontrada em uma quantidade significativamente menor de iterações, faz com que as extensões com as propostas de visibilidade tenham um tempo de execução menor do que o do algoritmo $ACO_{\mathbb{R}}$ original.

No capitulo 4 são realizadas as simulações para os dois conjuntos de testes (experimentos 1 e 2) aplicados para o $ACO_{\mathbb{R}}$ original e as propostas de visibilidade apresentadas neste capítulo. Na Seção 4.5 os tempos de execução das simulações são apresentados em tabelas, mostrando as diferenças entre os três modelos de algoritmos baseados ACO para domínios contínuos abordados neste trabalho, para provar a afirmação da diminuição do tempo total de execução do algoritmo.

4 TESTES E RESULTADOS

Neste capítulo são preparados e executados os experimentos necessários para mostrar a influência da visibilidade na velocidade de convergência e qualidade das soluções encontradas em algoritmos baseados em ACO para domínios contínuos. Parte-se da ideia de que haja influência da visibilidade em domínios contínuos como há nos domínios discretos e que foi introduzida na Subseção 2.4.2. As duas extensões propostas no capítulo anterior são comparadas entre elas e com o $ACO_{\mathbb{R}}$ original utilizando nos algoritmos os parâmetros definidos para os dois conjuntos de experimentos (Tabela 5) e as devidas análises são apresentadas neste capítulo após as tabelas de comparações.

4.1 DEFINIÇÃO DOS PARÂMETROS DO ALGORITMO

As simulações para comparar o $ACO_{\mathbb{R}}$ original com as extensões propostas neste trabalho foram divididas em dois conjuntos de testes. Essa divisão em 2 experimentos se justifica pelo fato de que havia interesse em usar os mesmos parâmetros usados por Socha e Dorigo (2006) para comparar as técnicas, e também haviam outros conjuntos de parâmetros, encontrados em simulações prévias de teste realizadas, que retornaram melhores valores de saída. Portanto, optou-se para a realização dos dois conjuntos de experimentos (parâmetros definidos por Socha e Dorigo (2006) e parâmetros definidos neste trabalho).

No primeiro conjunto de testes (experimento 1), o $ACO_{\mathbb{R}}$ original é comparado com as extensões aqui propostas usando os valores de parâmetros do algoritmo sugeridos por Socha e Dorigo (2006). Estes valores são informados na coluna "Valor no experimento 1" da Tabela 5 e os autores os escolheram por existirem alguns trabalhos na literatura com os quais o $ACO_{\mathbb{R}}$ foi comparado e esses parâmetros tornariam viável essa comparação entre as técnicas sem que fosse necessário reimplementar os trabalhos da literatura, usando assim os valores ali informados para a comparação.

Já para o segundo conjunto de testes (experimento 2), foram realizados diversos testes variando os parâmetros a fim de encontrar aqueles com melhores saídas. No caso do erro admitido ε (precisão), este parâmetro é apenas um objetivo a se alcançar, visto que quanto maior esse valor, mais rápido o algoritmo converge, entretanto pior é o valor encontrado. Assim, optou-se por manter o valor de ε assim como o do experimento 1. De maneira semelhante, o número de formigas m e a quantidade de soluções mantidas no

Tabela 5: Parâmetros usados nos conjuntos de testes para o ACO com domínio contínuo.

Parâmetro	Valor no experimento 1	Valor no experimento 2
m	2	2
k	50	50
ξ	0,85	0,65
q	0,0001	0,0001
ε	10^{-10}	10^{-10}

arquivo população k, são parâmetros que quanto maior seus valores, menor é o número de iterações necessárias para o algoritmo convergir, entretanto, maior é o tempo de execução, dado que mais instruções são realizados. Por esse motivo, assim como aconteceu em relação a ε, decidiu-se manter os valores de m e k como no experimento 1.

Os parâmetros restantes ξ e q são triviais para o algoritmo, implicando diretamente no desempenho sendo que para estes parâmetros, diferentemente do que acontece aos parâmetros já citados, existe um valor ótimo para eles. O parâmetro ξ (da Equação 2.10 usada no processo de construção de novas soluções) se comporta de modo que baixos valores impliquem que sejam amostrados valores próximos da média, enquanto que altos valores implicam em maior probabilidade de serem gerados valores distantes. Partindo do valor sugerido por Socha e Dorigo (2006) ($\xi = 0{,}85$) foram testados os valores $\xi = 0{,}9$, $\xi = 0{,}95$ e $\xi = 1$, entretanto os resultados pioravam a medida que o valor aumentava. Testou-se então valores menores como $\xi = 0{,}8$, $\xi = 0{,}75$, $\xi = 0{,}7$, $\xi = 0{,}65$, $\xi = 0{,}6$ e $\xi = 0{,}55$ que encontravam a precisão em menos iterações conforme o valor de ξ era reduzido. Entretanto, com o uso de valores menores que $\xi = 0{,}7$ surgiu um comportamento indesejado que é o fato de que algumas das simulações atingiam o número máximo de iterações sem que o valor de precisão fosse alcançado. Até o valor de $\xi = 0{,}65$ julgou-se tolerável esse "efeito colateral", visto que o $\text{ACO}_{\mathbb{R}}$ e o $\text{ACO}_{\mathbb{R}}$v2 (segunda proposta de visibilidade) conseguiram resolver todos os seus problemas dentro do limite de iterações e alcançando valores muito bons. Valores menores para ξ implicaram que a maioria dos problemas testados não conseguissem ser resolvidos dentro do limite e então fixou-se como o melhor valor encontrado sendo $\xi = 0{,}65$.

Pensando no comportamento induzido pela fórmula que utiliza o ξ (Equação 2.10), o ideal seria altos valores de desvio padrão nas primeiras iterações, para que as formigas explorem uma maior área e então o desvio padrão vai diminuindo para que seja realizada uma exploração mais localizada a medida que as formigas tenham encontrado uma boa região. Esse

comportamento já emerge naturalmente do arquivo população, pois o cálculo do desvio padrão também leva em conta, além do valor ξ, a distância euclidiana entre as soluções no arquivo. Assim, a medida que as soluções passam a se concentrar em um determinado pico do domínio, os valores de desvio padrão diminuem. Com base nessa ideia, também foi testado o uso de ξ variando semelhante a temperatura de Boltzmann, como é usado em alguns algoritmos de *Simulated Annealing* (TSALLIS; STARIOLO, 1996). O valor de ξ foi setado inicialmente com valores altos e diminuía a medida que o número de iterações aumentava, sendo que diversos testes com diferentes intervalos foram usados, sempre com os valores variando entre $1 \geq \xi \geq 0{,}5$. Entretanto os resultados encontrados não eram melhores do que quando usava-se valores fixos como $\xi = 0{,}85$ ou $\xi = 0{,}65$ e então essa ideia foi descartada.

Já em relação ao parâmetro q (Equação 2.9), baixos valores para q implicam que as melhores soluções serão preferencialmente escolhidas enquanto que altos valores implicam que as soluções do arquivo serão escolhidas de forma mais uniforme. Foram testados valores de q maiores que 0,0001, pois como será mostrado na Subseção 4.1.1 a seguir, $q = 0{,}0001$ implica que somente a melhor solução do arquivo tem chances de ser escolhida. Para o $\text{ACO}_{\mathbb{R}}$ e o $\text{ACO}_{\mathbb{R}}$v2 os resultados pioravam a media que se aumentava o valor de q então decidiu-se manter o valor como sugerido por Socha e Dorigo (2006). Já para o $\text{ACO}_{\mathbb{R}}$v1 foi utilizado um valor diferente de q e o motivo disso será explicado na Subseção 4.1.1.

Com base nos testes descritos nos parágrafos anteriores, definiu-se os valores informados na coluna "Valor no experimento 2" da Tabela 5 como sendo os utilizados para o segundo conjunto de testes.

A primeira proposta de visibilidade sugerida neste trabalho, possui em sua fórmula (Equação 3.1) dois outros parâmetros que não existem no $\text{ACO}_{\mathbb{R}}$ original e nem na segunda proposta de visibilidade. Esses parâmetros são α e β, cujos valores são apresentados na Tabela 6, e controlam respectivamente, a importância do feromônio e da visibilidade na escolha da solução a ser usada como base no processo de amostragem durante a construção de uma nova solução. Os mesmos valores de α e β são usados nos dois experimentos e esses valores foram obtidos por apresentarem bons resultados para os problemas testados, após simulações de teste com diferentes valores para os parâmetros. Nem todos os problemas apresentaram seu melhor comportamento com os mesmos parâmetros, mas estes parâmetros, de maneira geral, resultaram em boas saídas para todos os problemas. Estes dois parâmetros são similares aos α e β do ACO discreto *Ant System*, apresentado na Equação 2.3, Seção 2.4.

Tabela 6: Definição dos valores de α e β para as simulações.

Parâmetro	Valor
α	1
β	5

4.1.1 Caso Especial do Parâmetro q

O parâmetro q do algoritmo (Equação 2.9) comporta-se de modo que baixos valores para q implicam que as melhores soluções do arquivo população serão escolhidas com maior probabilidade, enquanto que altos valores para q implicam em uma distribuição de probabilidade mais uniforme entre as soluções do arquivo população. Somente uma solução será escolhida (s_l) para servir de média na amostragem de valores no processo de construção de uma nova solução X pela formiga.

Usando os valores comuns definidos em ambos os conjuntos de testes (Tabela 5), temos $q = 0{,}0001$ e $k = 50$ que aplicados à Equação 2.9, que é responsável por definir o valor de ω (valor referente a chance da solução ser escolhida), retorna os valores apresentados na Tabela 7 para as 3 primeiras soluções do arquivo população. Consequentemente, com base na Equação 2.8 que define o valor da probabilidade com que a solução s_j será escolhida a partir do seu valor ω_j, podemos afirmar que com o uso de $q = 0{,}0001$ sempre será escolhida a primeira solução do arquivo (a melhor).

Tabela 7: Valores de ω quando $q = 0{,}0001$ e $k = 50$.

Solução	ω
s_1	79,78846
s_2	0
s_3	0

Para a primeira proposta de visibilidade apresentada neste trabalho, pode-se dizer que a visibilidade não influencia na escolha da solução se for usado o valor $q = 0{,}0001$, pois no dividendo da Equação 3.1 (que define a probabilidade da solução ser escolhida na primeira proposta de visibilidade), quando $l \geq 2$, temos $\omega_l = 0$ e consequentemente retornaria $p_l = 0$ devido a multiplicação na equação. Em outras palavras, sempre a primeira solução seria escolhida mesmo que s_1 fosse um máximo local (não tivesse um vizinho bom) e s_2 fosse uma solução promissora (com um vizinho s'_2 muito melhor). Assim, para a primeira proposta de visibilidade define-se o uso de $q = 0{,}03$,

por ser um valor que possibilita uma distribuição de probabilidade mais uniforme para as soluções, mesmo as primeiras continuarem tendo maior valor de ω. O valor de ω_l para $l = 2,3,\ldots,k$ é maior que zero (o que não anulará o vizinho) e dependendo da qualidade do vizinho encontrado, estas soluções ($l = 2,3,\ldots,n$) alcançam um valor suficientemente significativo para poderem ser escolhidas na Equação 3.1.

Tabela 8: Valores de probabilidade das soluções serem escolhidas quando $q = 0{,}0001$ e $q = 0{,}03$ com base na Equação 2.8.

Solução	**Probabilidade de s_l se $q = 0{,}0001$**	**Probabilidade de s_l se $q = 0{,}03$**
s_1	100%	42,0173%
s_2	0%	33,6448%
s_3	0%	17,2738%

A Tabela 8 mostra a probabilidade das 3 primeiras soluções do arquivo população serem escolhidas se for usado $q = 0{,}0001$ ou $q = 0{,}03$ quando é considerada a equação de probabilidade original do $ACO_{\mathbb{R}}$ (Equação 2.8). É possível notar que o uso de $q = 0{,}0001$ não é desejável na primeira proposta de visibilidade (que usa a Equação 3.1), pois o vizinho não influenciará no valor de probabilidade. A Tabela 9 contém os valores de uma simulação de teste para a primeira proposta de visibilidade, onde prova o que já foi dito sobre o uso de $q = 0{,}0001$. Como em cada simulação as soluções são inicialmente setadas aleatoriamente, cada simulação pode resultar em valores diferentes de probabilidade para cada solução no que diz respeito ao uso de $q = 0{,}03$, dependendo de quão promissora é a solução. Esse valor $q = 0{,}03$ foi obtido após uma série de simulações de teste para definir o valor q com o comportamento mais desejado, comportamento este que é o aumento na chance de uma solução que não é a melhor, mas é promissora.

Tabela 9: Exemplo de simulação com a probabilidade das soluções serem escolhidas quando $q = 0{,}0001$ e $q = 0{,}03$ com base na Equação 3.1.

Solução	**Probabilidade de s_l se $q = 0{,}0001$**	**Probabilidade de s_l se $q = 0{,}03$**
s_1	100%	20,9498%
s_2	0%	11,7434%
s_3	0%	43,7658%

Pode-se entender pela Tabela 9 que o uso da visibilidade permite que uma solução que não é a melhor, mas por estar em uma área promissora, seja

mais desejável pelo algoritmo e assim explorar uma área com mais chances de se encontrar melhores soluções. A visibilidade na proposta 1, em conjunto com um valor de q suficientemente grande para permitir que as demais soluções, além da melhor, sejam escolhidas, faz com que o algoritmo seja capaz de identificar uma área promissora e escapar de um máximo local. Vale lembrar que, mesmo que uma solução que já atingiu o máximo local da região em que se encontra seja escolhida para ser média em uma nova amostragem, devido ao desvio padrão existe a possibilidade de escapar para uma área mais promissora. O que pretende-se com o comportamento descrito para as propostas de visibilidade, é aumentar a velocidade de convergência do algoritmo em melhores soluções.

A segunda proposta de visibilidade, por ter a solução encontrada na busca local adicionada diretamente ao vetor população (e não como informação extra da solução original[1]), não corre o risco de uma boa área deixar de ser explorada, pois se a busca local encontrar uma solução melhor, ao ser adicionada ao vetor população, ela será a melhor do arquivo e consequentemente a escolhida na próxima iteração.

4.2 PROBLEMAS USADOS NAS SIMULAÇÕES

Os problemas usados para comparar o $ACO_{\mathbb{R}}$ original com as duas propostas de visibilidade deste trabalho são apresentadas na Tabela 10. Os problemas são os mesmos usados por Socha e Dorigo (2006) para comparar o $ACO_{\mathbb{R}}$ com outras técnicas de otimização em domínios contínuos. Socha usou estas funções porque os outros trabalhos da literatura com os quais comparou sua proposta, utilizavam estas funções como estudo de caso, assim ele não precisaria replicar os experimentos para aquelas técnicas. Para cada problema existem diferentes números de variáveis (dimensão) e diferentes intervalos.

A Tabela 11 mostra as fórmulas de cada problema apresentado na Tabela 10. Como introduzido no Capítulo 1, o objetivo é encontrar a melhor combinação de valores para as variáveis do problema, assim busca-se aqueles cuja saída da função apresente o menor valor possível, pois os problemas testados são todos problemas de minimização. O fato dos problemas testados serem de minimização não implica que o algoritmo não possa ser aplicado ou não funcione em problemas de maximização e isso fica claro ao lembrar que uma mudança de sinal na saída da função pode converter um problema de maximização em problema de minimização e vice-versa. Cada solução do

[1] Somente na primeira proposta de visibilidade a solução encontrada na busca local é colocada como informação extra na estrutura da própria solução original.

Tabela 10: Funções usadas com as implementações do ACO para domínio contínuo.

Função	Dimensão	Domínio
B_2	2	$[-100,100]$
Branin RCOS	2	$[-5,15]$
Cigar	10	$[-3,7]$
De Jong	3	$[-5{,}12,5{,}12]$
Easom	2	$[-100,100]$
Ellipsoid	10	$[-3,7]$
Goldstein and Price	2	$[-2,2]$
Martin and Gaddy	2	$[-20,20]$
Rosenbrock	2	$[-5,5]$
Sphere	10	$[-3,7]$
Tablet	10	$[-3,7]$
Zakharov	5	$[-5,10]$

arquivo população será composta por um conjunto de n valores referentes as n variáveis do respectivo problema.

Todas as funções apresentadas nas Tabelas 10 e 11 são problemas conhecidos na literatura, pois podem ser encontrados em diversos trabalhos, como por exemplo: Socha (2004), Socha e Dorigo (2006) e Kern et al. (2004). São problemas propostos com a finalidade de serem usados para testar técnicas de otimização. Por serem problemas famosos, seus valores ótimos são conhecidos e são usados neste trabalho para o critério de parada do algoritmo juntamente com um erro permitido (precisão).

4.3 CRITÉRIO DE PARADA DO ALGORITMO

Na Subseção 2.4.2 onde é mostrada a importância da visibilidade em domínios discretos, o critério de parada usado para o algoritmo foi um número máximo de iterações definido. Assim, ao atingir essa quantidade de iterações, haviam diferentes valores de resultado final (tamanho do caminho percorrido pelo caixeiro) para o algoritmo em cada um dos casos testados. Para a comparação no domínio contínuo, foi utilizado o critério de parada definido pela Equação 4.1. Quando a diferença entre a saída da melhor solução encontrada pelo algoritmo f e o valor ótimo para o problema f^* for menor que ε, a

Tabela 11: Funções usadas com as implementações do ACO para domínio contínuo.

Função	**Fórmula**
B_2	$f_{B_2}(X) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{2}{5}\cos(4\pi x_2) + \frac{7}{10}$
Branin RCOS	$f_{RC}(X) = \left(x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
Cigar	$f_{CG}(X) = x_1^2 + 10^4 \sum_{i=2}^{n} x_i^2$
De Jong	$f_{DJ}(X) = x_1^2 + x_2^2 + x_3^2$
Easom	$f_{ES}(X) = -\cos(x_1)\cos(x_2)\, e^{-((x_1-\pi)^2+(x_2-\pi)^2)}$
Ellipsoid	$f_{EL}(X) = \sum_{i=1}^{n}\left(100^{\frac{i-1}{n-1}} x_i\right)^2$
Goldstein and Price	$f_{GP}(X) = (1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \cdot (30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2))$
Martin and Gaddy	$f_{MG}(X) = (x_1 - x_2)^2 + \left(\frac{x_1 + x_2 - 10}{3}\right)^2$
Rosenbrock	$f_{Rn}(X) = \sum_{i=1}^{n-1} 100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2$
Sphere	$f_{SP}(X) = \sum_{i=1}^{n} x_i^2$
Tablet	$f_{TB}(X) = 10^4 x_1^2 + \sum_{i=2}^{n} x_i^2$
Zakharov	$f_{Z_n}(X) = \left(\sum_{i=1}^{n} x_i^2\right) + \left(\sum_{i=1}^{n} \frac{ix_i}{2}\right)^2 + \left(\sum_{i=1}^{n} \frac{ix_i}{2}\right)^4$

simulação encerra[2]. Assim, tem-se praticamente o mesmo resultado do algoritmo e a principal análise que é feita é sobre a velocidade de convergência, ou seja, quantas iterações são necessárias até o algoritmo encontrar o valor ótimo e atingir a precisão especificada.

$$|f - f^*| < \varepsilon \tag{4.1}$$

Este critério de parada foi adotado por ser o mesmo usado por Socha e Dorigo (2006). Os autores, por usa vez, adotaram esse critério para facilitar a comparação com outras técnicas de otimização em domínios contínuos encontradas na literatura que usavam esse mesmo critério. Assim, a comparação poderia proceder sem a necessidade de que os autores reimplementassem outros trabalhos da mesma classe de problemas.

[2]O valor de ε reflete a precisão requerida, o erro aceitável e foi definido juntamente com os demais parâmetros do algoritmo na Tabela 5 da Seção 4.1.

A Tabela 12 mostra qual é o valor ótimo de cada uma das funções usadas nas simulações. Esses valores foram usados na Equação 4.1 no parâmetro f^* para encontrar o limiar que o algoritmo deveria atingir de acordo com a precisão requerida. O valor referente a variável ε do algoritmo foi definido no capítulo anterior, na Seção 4.1.

Tabela 12: Valor ótimo de cada problema usado na simulação.

Função	**Valor ótimo**
B_2	$f^* = 0$
Branin RCOS	$f^* = 0.397887$
Cigar	$f^* = 0$
De Jong	$f^* = 0$
Easom	$f^* = -1$
Ellipsoid	$f^* = 0$
Goldstein and Price	$f^* = 3$
Martin and Gaddy	$f^* = 0$
Rosenbrock	$f^* = 0$
Sphere	$f^* = 0$
Tablet	$f^* = 0$
Zakharov	$f^* = 0$

4.4 RESULTADOS DAS SIMULAÇÕES

4.4.1 Experimento 1

Utilizando as duas implementações propostas para a visibilidade e o $ACO_{\mathbb{R}}$ original, juntamente com os valores definidos para os parâmetros do algoritmo para o primeiro conjunto de testes (Tabela 5 da Seção 4.1), são obtidos os resultados das Tabelas 13 e 14 referentes ao número de iterações necessárias para convergência em todos os problemas apresentados na Tabela 11[3]. Todos os valores são quantidades de iterações necessárias referentes as 50 simulações independentes realizadas até que o algoritmo atinja o objetivo com a precisão requerida. $ACO_{\mathbb{R}}$, $ACO_{\mathbb{R}}$v1 e $ACO_{\mathbb{R}}$v2 são, respectivamente, a técnica $ACO_{\mathbb{R}}$ original, a extensão com a primeira proposta de visibilidade e a extensão com a segunda proposta de visibilidade. Mínimo e máximo estão

[3]O motivo da separação dos resultados em duas tabelas é devido ao tamanho da tabela exceder o tamanho da página.

em número de iterações e correspondem, respectivamente, a simulação que utilizou a menor quantidade de iterações e a que utilizou o maior número de iterações, ou seja, as simulações com menor e maior velocidade de convergência para cada caso. A Figura 11 permite a visualização dos dados apresentados nas Tabelas 13 e 14 através de um gráfico de barras.

Tabela 13: Número de iterações do ACO para domínio contínuo no primeiro conjunto de testes.

Problema	Técnica	Média	Desvio Padrão	Mínimo	Máximo
B2	$ACO_{\mathbb{R}}$	371,32	24,268	335	488
	$ACO_{\mathbb{R}}$v1	278,46	37,121	212	363
	$ACO_{\mathbb{R}}$v2	133,58	9,181	113	152
Branin	$ACO_{\mathbb{R}}$	180,6	40,221	119	310
RCOS*	$ACO_{\mathbb{R}}$v1	169,38	113,851	54	740
	$ACO_{\mathbb{R}}$v2	38,46	15,235	6	76
Cigar	$ACO_{\mathbb{R}}$	1279,08	103,744	1141	1620
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	409,64	57,052	318	606
De Jong	$ACO_{\mathbb{R}}$	333,26	20,994	287	383
	$ACO_{\mathbb{R}}$v1	255,2	66,607	175	522
	$ACO_{\mathbb{R}}$v2	97,7	7,965	84	117
Easom	$ACO_{\mathbb{R}}$	428,6	29,279	365	491
	$ACO_{\mathbb{R}}$v1	359,98	144,685	240	1288
	$ACO_{\mathbb{R}}$v2	137,94	16,963	76	181
Ellipsoid	$ACO_{\mathbb{R}}$	1004,7	72,279	815	1179
	$ACO_{\mathbb{R}}$v1	1030,26	320,287	563	1822
	$ACO_{\mathbb{R}}$v2	218,4	22,576	180	281

* O valor de precisão $\varepsilon = 10^{-4}$ foi usado no lugar de $\varepsilon = 10^{-10}$ no critério de parada (Equação 4.1).
** O número máximo de iterações foi excedido sem alcançar a precisão.

O problema *Branin RCOS* é a única exceção em relação aos valores de precisão definidos na Tabela 5. Para este problema, foi usado o valor $\varepsilon = 10^{-4}$ ao invés de $\varepsilon = 10^{-10}$ em ambos os experimentos, para todas as 3 técnicas testadas. A justificativa para a mudança no valor de precisão é o fato de que nenhuma das técnicas alcançou um valor próximo 10^{-10} do valor ótimo dentro de um limite de 10000 iterações. Essa mesma decisão foi tomada por Socha e Dorigo (2006) durante suas simulações.

Analisando os resultados das Tabelas 13 e 14 é possível notar que a proposta 2 é a mais eficiente. A interpretação da visibilidade colocada na

Tabela 14: Número de iterações do ACO para domínio contínuo no primeiro conjunto de testes.

Problema	Técnica	Média	Desv. Padrão	Mínimo	Máximo
Goldstein and Price	$ACO_{\mathbb{R}}$	316,08	18,333	277	364
	$ACO_{\mathbb{R}}$v1	277,44	159,18	192	1295
	$ACO_{\mathbb{R}}$v2	51,6	42,212	1	111
Martin and Gaddy	$ACO_{\mathbb{R}}$	341,02	26,2	257	388
	$ACO_{\mathbb{R}}$v1	234,36	19,342	196	285
	$ACO_{\mathbb{R}}$v2	126,14	13,403	82	153
Rosenbrock	$ACO_{\mathbb{R}}$	2164,24	306,411	1376	2895
	$ACO_{\mathbb{R}}$v1	3366,46	931,383	1380	4704
	$ACO_{\mathbb{R}}$v2	370,22	82,683	200	582
Sphere	$ACO_{\mathbb{R}}$	924,48	50,04	787	1043
	$ACO_{\mathbb{R}}$v1	775,76	458,346	421	3057
	$ACO_{\mathbb{R}}$v2	197,06	26,686	159	303
Tablet	$ACO_{\mathbb{R}}$	1009,56	65,849	832	1203
	$ACO_{\mathbb{R}}$v1	979,36	439,543	481	2510
	$ACO_{\mathbb{R}}$v2	217,76	22,669	172	272
Zakharov	$ACO_{\mathbb{R}}$	274,12	19,067	219	314
	$ACO_{\mathbb{R}}$v1	203,46	27,441	158	290
	$ACO_{\mathbb{R}}$v2	91,58	9,427	70	112

proposta 2, embora seja mais simples, obteve resultados mais significantes para todos os problemas testados, gastando menos que 50% do número de iterações requeridas pelo $ACO_{\mathbb{R}}$ original. Por outro lado, a proposta 1 obteve bons resultados em apenas 9 dos 12 problemas ao qual foi apresentada, e o número de iterações necessárias para a convergência nestas 9 funções, embora melhor que o $ACO_{\mathbb{R}}$ original, não alcançou a mesma eficiência que a proposta 2 teve. A melhora na velocidade de convergência no ACO contínuo com uso da visibilidade da proposta 2 é similar a melhora que ocorre no ACO discreto com o uso da visibilidade apresentada na Subseção 2.4.2.

A partir dos resultados e analisando o caso da proposta 1, acredita-se que o uso de um processo de amostragem unidirecional pode ser eficiente para alguns problemas, mas existem outros onde a superfície de erro é mais complexa, com muitos pontos de máximos locais, e a amostragem pode gerar um valor que facilmente passe além de um pico de máximo global[4]. Ou-

[4] Acredita-se que esse é o caso do problema *Cigar*, onde nenhuma das 50 simulações alcançou a precisão requerida dentro de um máximo de 10000 iterações.

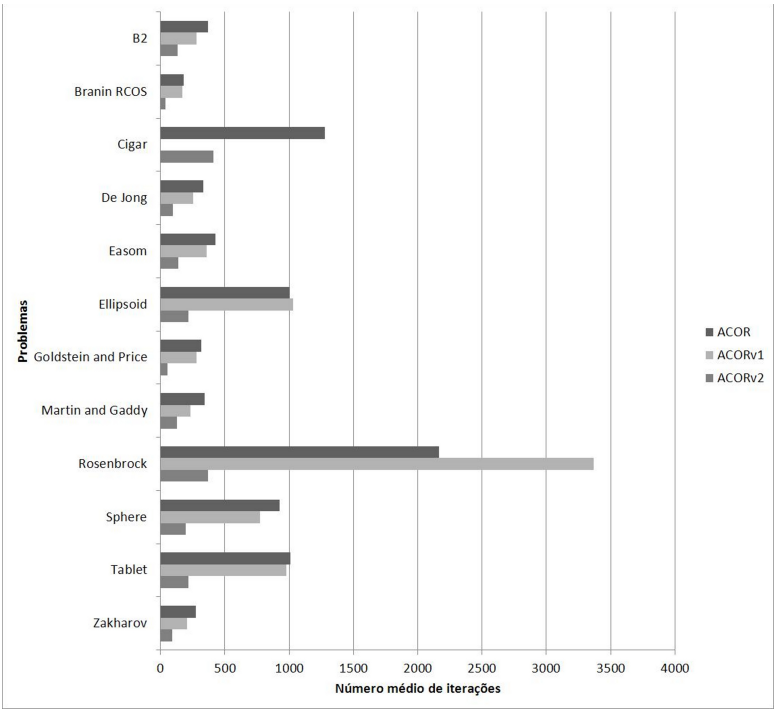

Figura 11: Gráfico para comparação do número médio de iterações, necessárias por cada técnica, para os problemas apresentados. Valores referentes ao uso dos parâmetros propostos para o experimento 1.

tra possível análise referente a proposta 1, é que por causa da amostragem unilateral, não é possível escapar de uma região de máximo local, pelo fato do processo permitir somente amostragens em uma direção que talvez seja contrária a ideal. Apesar disso, simulações realizadas previamente usando uma amostragem normal bidirecional na proposta 1, embora convirja para todos os problemas alcançando a precisão sem estourar o limite de iterações, demanda um número maior de iterações até atingir o objetivo, se comparado a outras técnicas. Existe um balanço crítico nesse assunto e fica a sugestão para pesquisas futuras.

4.4.2 Experimento 2

Para os parâmetros do segundo conjunto de testes, também comparando as duas propostas de visibilidade com o $ACO_{\mathbb{R}}$ original, chegamos aos resultados das Tabelas 15 e 16 que, assim como no experimento 1, os resultados foram divididos em duas tabelas pelo motivo do tamanho da tabela exceder os limites da página. Também como no experimento 1, os resultados apresentam significativa redução da quantidade média de iterações necessárias entre as diferentes técnicas. Os valores também são referentes a 50 simulações independentes. A Figura 12 permite a visualização dos dados apresentados nas Tabelas 15 e 16 através de um gráfico de barras.

Tabela 15: Número de iterações do ACO para domínio contínuo no segundo conjunto de testes.

Problema	**Técnica**	**Média**	**Desv. Padrão**	**Mínimo**	**Máximo**
B2	$ACO_{\mathbb{R}}$	275,72	13,949	237	316
	$ACO_{\mathbb{R}}$v1	223,22	29,775	160	296
	$ACO_{\mathbb{R}}$v2	105,52	7,919	77	119
Branin RCOS*	$ACO_{\mathbb{R}}$	123,92	29,9	45	217
	$ACO_{\mathbb{R}}$v1	103,22	46,22	58	331
	$ACO_{\mathbb{R}}$v2	30,72	9,269	13	56
Cigar	$ACO_{\mathbb{R}}$	982,4	141,512	739	1516
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	602,86	184,164	291	1082
De Jong	$ACO_{\mathbb{R}}$	239,18	14,61	206	282
	$ACO_{\mathbb{R}}$v1	195,82	55,731	146	415
	$ACO_{\mathbb{R}}$v2	79,16	6,563	60	91
Easom	$ACO_{\mathbb{R}}$	318,68	21,424	263	415
	$ACO_{\mathbb{R}}$v1	270,78	72,787	204	706
	$ACO_{\mathbb{R}}$v2	109,26	13,019	79	144
Ellipsoid	$ACO_{\mathbb{R}}$	771,88	87,674	605	1025
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	204,1	125,868	129	998

* O valor de precisão $\varepsilon = 10^{-4}$ foi usado no lugar de $\varepsilon = 10^{-10}$ no critério de parada (Equação 4.1).
** O número máximo de iterações foi excedido sem alcançar a precisão.

No segundo conjunto de testes (Tabelas 15 e 16), é possível observar o mesmo comportamento emergido no primeiro conjunto (experimento

Tabela 16: Número de iterações do ACO para domínio contínuo no segundo conjunto de testes.

Problema	Técnica	Média	Desv. Padrão	Mínimo	Máximo
Goldstein	$ACO_{\mathbb{R}}$	232,16	13,525	201	266
and	$ACO_{\mathbb{R}}$v1	195,76	38,965	149	357
Price	$ACO_{\mathbb{R}}$v2	63,34	28,937	1	93
Martin	$ACO_{\mathbb{R}}$	238,7	15,521	204	278
and	$ACO_{\mathbb{R}}$v1	179,64	13,612	146	201
Gaddy	$ACO_{\mathbb{R}}$v2	94,42	10,432	67	121
Rosenbrock	$ACO_{\mathbb{R}}$	1308,14	274,163	938	2319
	$ACO_{\mathbb{R}}$v1	2184,56	682,753	825	4154
	$ACO_{\mathbb{R}}$v2	274,78	56,753	149	410
Sphere	$ACO_{\mathbb{R}}$	684,9	76,995	505	903
	$ACO_{\mathbb{R}}$v1	440,38	151,481	255	920
	$ACO_{\mathbb{R}}$v2	167,62	56,297	115	488
Tablet	$ACO_{\mathbb{R}}$	768,02	104,322	625	1073
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	191,56	39,747	126	317
Zakharov	$ACO_{\mathbb{R}}$	207,24	13,381	174	237
	$ACO_{\mathbb{R}}$v1	162,76	31,108	118	296
	$ACO_{\mathbb{R}}$v2	76,42	7,434	59	91

** O número máximo de iterações foi excedido sem alcançar a precisão.

1). Ambas as propostas de visibilidade guiam para uma redução significante para a maioria dos problemas e, comparando as duas extensões (proposta 1 com a proposta 2), a proposta 2 é mais eficiente que a proposta 1. Novamente a proposta 2 precisou de menos que 50% das iterações necessárias pelo $ACO_{\mathbb{R}}$ original para resolver todos os problemas, exceto para o problema *Cigar*, onde a redução chegou a 38,6%. Neste experimento, três funções não foram resolvidas pela proposta 1 dentro do limite máximo de 10000 iterações (*Cigar*, *Ellipsoid* e *Tablet*). Além dos motivos mencionados no experimento 1 (pois isso também ocorreu no problema *Cigar* do experimento 1), este fato também pode ser atribuído aos baixos valores de ξ, que é o parâmetro multiplicado na fórmula do desvio padrão (Equação 2.10) para a amostragem de novos valores, e consequentemente a baixa probabilidade de serem gerados valores longe da média pode evitar que a técnica escape de um máximo local.

Todas as técnicas atingiram melhores resultados no experimento 2 para todos os problemas se comparados aos valores encontrados no experimento 1, o que implica que os parâmetros referentes ao segundo experimento

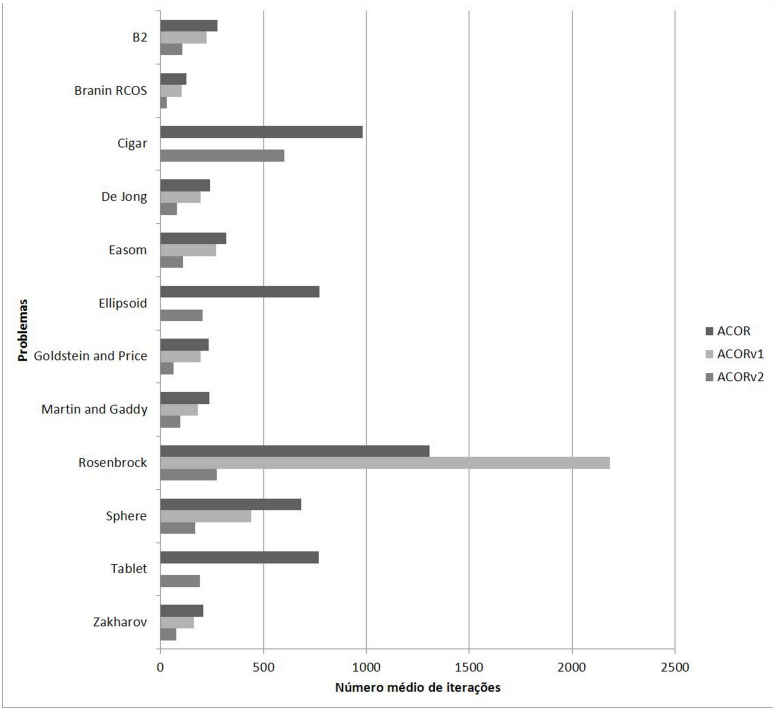

Figura 12: Gráfico para comparação do número médio de iterações, necessárias por cada técnica, para os problemas apresentados. Valores referentes ao uso dos parâmetros propostos para o experimento 2.

na Tabela 5 resultam em melhor desempenho para o $ACO_{\mathbb{R}}$ e as extensões propostas. As exceções são os problemas *Cigar* e *Goldstein and Price*, que na segunda proposta de visibilidade, pioraram com o uso do valor $q = 0{,}65$ pois o número de iterações e o tempo necessário[5] para alcançar a precisão aumentaram. Isso possivelmente tem relação com as superfícies das funções, pois em uma superfície com diversos pontos de máximos locais com bases grande[6], ao diminuirmos o valor do fator que controla o tamanho do desvio padrão na amostragem, diminuímos a chance do algoritmo escapar desses máximos locais. Além disso, a proposta 1 não pode resolver mais 2 outros problemas, *El-*

[5] A análise dos tempos de execução são apresentadas na Seção 4.5.

[6] Quanto maior a base do pico, maior deve ser o valor do desvio padrão para aumentar as chances de escapar dele.

lipsoid e *Tablet*. A proposta 1 não é tão robusta quanto a proposta 2 e o $ACO_{\mathbb{R}}$ original. A proposta 2 mostrou-se ser uma boa interpretação para o termo de visibilidade em algoritmos baseados em ACO para domínios contínuos, pois teve um comportamento similar ao visto para ACO em domínios discretos.

Todos os valores apresentados nas Tabelas 13, 14, 15 e 16 são referentes as 50 simulações independentes para cada caso. Como esperado, a visibilidade mostrou, em problemas com domínios contínuos, o mesmo comportamento apresentado em domínios discretos. Ou seja, em todos os problemas testados, o uso da heurística de visibilidade guiou o algoritmo a encontrar boas soluções em uma quantidade significantemente menor de iterações.

Comparando as extensões sugeridas neste trabalho, a proposta 2 apresentou melhor desempenho que a proposta 1 em ambos os experimentos. Também é importante lembrar que os parâmetros utilizados no segundo conjunto de testes (experimento 2), retornaram valores significativamente melhores que os do primeiro conjunto (experimento 1) para a maioria dos problemas. Isto significa que aqueles parâmetros têm grande importância no algoritmo e eles também influenciam na velocidade de convergência do ACO.

4.5 TEMPO DE EXECUÇÃO DO ALGORITMO

Na Seção 3.5 chama-se a atenção para o fato de ambas as extensões propostas de visibilidade para o $ACO_{\mathbb{R}}$ realizarem mais instruções que o original. Entretanto, pelo fato de que o algoritmo encontra a precisão requerida como critério de parada em uma quantidade significativamente menor de iterações, o tempo de execução é menor para as extensões do que para a versão original.

As Tabelas 17 e 18 apresentam o tempo gasto por cada uma das 3 implementações ($ACO_{\mathbb{R}}$, $ACO_{\mathbb{R}}$v1 e $ACO_{\mathbb{R}}$v2) para gerar os resultados de cada um dos problemas da Tabela 11 utilizando os parâmetros definidos para o primeiro conjunto de testes. A divisão em duas tabelas se deve ao fato do excesso de tamanho ao mostrar os dados em apenas uma tabela. Os tempos informados são referentes aos necessários para gerar cada replicação, enquanto que as análises dos dados (média, desvio padrão, mínimo e máximo) foram realizadas sobre o conjunto das 50 replicações do mesmo problema e estão em milissegundos. Mínimo e máximo correspondem respectivamente a simulação mais rápida e a mais lenta.

Os resultados também podem ser vistos na Figura 13 onde são mostrados em um gráfico de barras.

Todas as simulações, ou seja, todas as execuções das implementações das 3 técnicas ($ACO_{\mathbb{R}}$ original, visibilidade 1 e visibilidade 2) para todos os

Tabela 17: Tempo de execução (em milissegundos) de cada técnica, referente ao primeiro conjunto de testes.

Problema	Técnica	Média	Desv. Padrão	Mínimo	Máximo
B2	$ACO_{\mathbb{R}}$	11,26	11,492	5	61
	$ACO_{\mathbb{R}}$v1	23,28	17,964	11	110
	$ACO_{\mathbb{R}}$v2	11,28	12,228	2	57
Branin	$ACO_{\mathbb{R}}$	9,3	11,648	2	60
RCOS	$ACO_{\mathbb{R}}$v1	13,9	14,09	3	72
	$ACO_{\mathbb{R}}$v2	3,62	3,585	0	21
Cigar	$ACO_{\mathbb{R}}$	63,92	21,428	45	177
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	71,58	15,802	52	139
De Jong	$ACO_{\mathbb{R}}$	10,32	10,991	5	60
	$ACO_{\mathbb{R}}$v1	21,46	18,705	8	93
	$ACO_{\mathbb{R}}$v2	7,02	6,123	2	38
Easom	$ACO_{\mathbb{R}}$	10,94	11,612	5	62
	$ACO_{\mathbb{R}}$v1	25,8	16,793	12	114
	$ACO_{\mathbb{R}}$v2	8,24	8,04	2	43
Ellipsoid	$ACO_{\mathbb{R}}$	44,48	17,868	28	129
	$ACO_{\mathbb{R}}$v1	294,18	115,914	121	580
	$ACO_{\mathbb{R}}$v2	35,04	14,055	24	107

** O número máximo de iterações foi excedido sem alcançar a precisão.

problemas e as replicações foram realizados em uma máquina *Sun Ultra 27 Workstation* com 3.2 GHz de frequência do processador, 8 MB em memória cache e 12 GB de memória RAM em DDR3. A máquina executava o sistema operacional *UBUNTU LINUX 10.04 64 bit* e a codificação das 3 técnicas foi realizada utilizando a linguagem JAVA. Mais detalhes sobre a máquina podem ser encontrados na página http://www.sun.com/desktop/workstation/ultra27.

Analisando os tempos de execução das Tabelas 17 e 18, podemos confirmar um comportamento desejável, pois a proposta 2 apresenta tempos de execução menores ou equivalentes ao $ACO_{\mathbb{R}}$ original para todos os problemas além de encontrar a resposta com menos iterações como foi visto na seção anterior. Outro fator importante é que os valores de desvio padrão da visibilidade 2 também são menores ou equivalentes ao $ACO_{\mathbb{R}}$ o que indica um intervalo de confiança menor, consequentemente maior precisão sobre qual a faixa de tempo a técnica demanda para encontrar soluções para os problemas. A proposta 1, ao ser comparada com o $ACO_{\mathbb{R}}$ original, apresentou resultados equivalentes para alguns problemas (aqueles que tiveram significa-

Tabela 18: Tempo de execução (em milissegundos) de cada técnica, referente ao primeiro conjunto de testes.

Problema	Técnica	Média	Desv. Padrão	Mínimo	Máximo
Goldstein and Price	$ACO_{\mathbb{R}}$	8,88	10,317	4	59
	$ACO_{\mathbb{R}}$v1	27,46	24,52	10	134
	$ACO_{\mathbb{R}}$v2	5,16	6,774	0	33
Martin and Gaddy	$ACO_{\mathbb{R}}$	11	13,596	4	69
	$ACO_{\mathbb{R}}$v1	13,92	11,498	8	70
	$ACO_{\mathbb{R}}$v2	6,48	7,754	2	43
Rosenbrock	$ACO_{\mathbb{R}}$	78,66	29,837	32	180
	$ACO_{\mathbb{R}}$v1	375,86	184,009	88	805
	$ACO_{\mathbb{R}}$v2	11,5	11,802	4	62
Sphere	$ACO_{\mathbb{R}}$	47,42	29,006	31	201
	$ACO_{\mathbb{R}}$v1	205,36	185,507	60	1290
	$ACO_{\mathbb{R}}$v2	24,06	12,165	16	90
Tablet	$ACO_{\mathbb{R}}$	45,68	17,011	32	121
	$ACO_{\mathbb{R}}$v1	311,26	171,346	105	1033
	$ACO_{\mathbb{R}}$v2	38,04	12,279	27	98
Zakharov	$ACO_{\mathbb{R}}$	8,24	9,516	3	55
	$ACO_{\mathbb{R}}$v1	17,28	12,274	9	65
	$ACO_{\mathbb{R}}$v2	7,18	6,727	3	38

tiva redução no número médio de iterações) e piores para os outros (maioria), o desempenho foi pior naqueles problemas onde o algoritmo não reduziu significativamente o número de iterações ou acabou consumindo mais iterações. Mesmo em funções onde a proposta 1 foi melhor que o $ACO_{\mathbb{R}}$ original, pelo fato da quantidade de iterações necessárias não ter sido significativamente reduzida ou suficientemente menor, o tempo foi de execução foi maior devido as instruções extras que são realizadas para a extensão.

O $ACO_{\mathbb{R}}$v2 converge para o valor ótimo dos problemas demandando menos iterações e, na pior das hipóteses, consumindo tempo equivalente ao $ACO_{\mathbb{R}}$, caracterizando assim um melhor desempenho. Por outro lado, o $ACO_{\mathbb{R}}$v1 não foi tão eficiente quanto a técnica original neste quesito de comparação, que é o tempo de execução.

Nas Tabelas 19 e 20 referentes aos resultados do segundo experimento, assim como aconteceu para o primeiro experimento, são apresentados os tempos gastos por cada uma das três implementações para gerar os resultados de cada um dos problemas apresentados neste trabalho, utilizando os parâmetros definidos para o segundo conjunto de testes (experimento 2, Tabela 5). Os

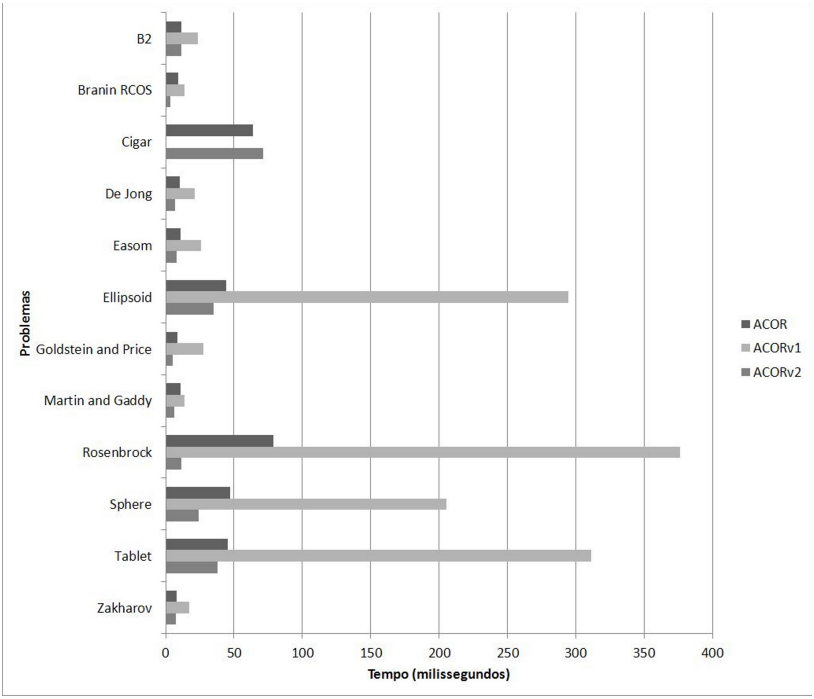

Figura 13: Gráfico de barras referente ao tempo de execução (em milissegundos) para o experimento 1.

valores estão em milissegundos e são referentes a 50 replicações analisando os tempos necessários para gerar a resposta de cada um dos problemas.

A Figura 14 apresenta os mesmos valores das Tabelas 19 e 20 através de um gráfico de barras, assim como foi feito para os tempos do experimento 1, mas agora para o experimento 2.

Analisando as Tabelas 19 e 20, é possível concluir que esse conjunto de valores para os parâmetros (experimento 2), além de diminuir o número médio de iterações como foi visto na seção anterior, também diminui o tempo de execução da maioria das implementações. Novamente a proposta 2 apresentou tempos equivalentes ou melhores que a técnica original, por outro lado, a proposta 1, assim como dito sobre os tempos de execução do primeiro experimento, por não ter apresentado uma diminuição significante no número de iterações para os problemas, somado ao fato da extensão realizar algumas instruções a mais, demandou mais tempo que o ACO$_{\mathbb{R}}$ original.

Tabela 19: Tempo de execução (em milissegundos) de cada técnica, referente ao segundo conjunto de testes.

Problema	Técnica	Média	Desv. Padrão	Mínimo	Máximo
B2	$ACO_{\mathbb{R}}$	8,72	12,411	3	55
	$ACO_{\mathbb{R}}$v1	15,76	6,297	9	47
	$ACO_{\mathbb{R}}$v2	9,04	8,715	2	37
Branin RCOS	$ACO_{\mathbb{R}}$	5,98	7,378	1	36
	$ACO_{\mathbb{R}}$v1	9,7	9,23	3	57
	$ACO_{\mathbb{R}}$v2	3,48	3,919	0	21
Cigar	$ACO_{\mathbb{R}}$	44,98	17,686	27	118
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	159,94	59,552	61	310
De Jong	$ACO_{\mathbb{R}}$	8,3	9,79	3	55
	$ACO_{\mathbb{R}}$v1	13,4	11,371	7	62
	$ACO_{\mathbb{R}}$v2	7,74	9,03	2	47
Easom	$ACO_{\mathbb{R}}$	9,84	9,797	4	50
	$ACO_{\mathbb{R}}$v1	17,3	16,251	10	114
	$ACO_{\mathbb{R}}$v2	6,84	7,066	3	42
Ellipsoid	$ACO_{\mathbb{R}}$	34,12	18,804	23	119
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	31,72	21,076	17	142

** O número máximo de iterações foi excedido sem alcançar a precisão.

De uma forma geral, todas as técnicas ficaram mais rápidas com os parâmetros do experimento 2 (Tabela 5), com exceção de alguns casos, como o problema *Cigar* que, para o $ACO_{\mathbb{R}}$v2, demandou mais tempo para encontrar o valor ótimo no experimento 2 do que no experimento 1. Se medirmos o tamanho da redução no experimento 2 em comparação com o experimento 1, o $ACO_{\mathbb{R}}$ e o $ACO_{\mathbb{R}}$v1 tiveram maiores reduções, diminuindo a diferença para a melhor técnica, que é o $ACO_{\mathbb{R}}$v2. Esse fato pode ter ocorrido pelo $ACO_{\mathbb{R}}$v2 já estar muito próximo do melhor tempo possível, assim, colocando um conjunto melhor de parâmetros, apesar da técnica melhorar seus resultados, não houve um diferença tão significativa, já que os valores anteriores já eram muito bons, talvez próximo de um limite para a técnica em específico.

Realizando uma análise geral dos resultados dos experimentos, dada a grande quantidade de replicações feitas, pode-se considerar as médias como sendo valores realmente indicativos ao valor real referente a cada técnica. Além disso, os resultados foram todos considerados significantes a favor do $ACO_{\mathbb{R}}$v2 dada a grande diferença na quantidade média de iterações e a equi-

Tabela 20: Tempo de execução (em milissegundos) de cada técnica, referente ao segundo conjunto de testes.

Problema	Técnica	Média	Desv. Padrão	Mínimo	Máximo
Goldstein and Price	$ACO_{\mathbb{R}}$	7,62	8,884	3	49
	$ACO_{\mathbb{R}}$v1	12,88	7,79	7	41
	$ACO_{\mathbb{R}}$v2	5,02	5,601	0	32
Martin and Gaddy	$ACO_{\mathbb{R}}$	7,5	9,279	3	55
	$ACO_{\mathbb{R}}$v1	10,2	8,68	6	59
	$ACO_{\mathbb{R}}$v2	5,94	6,4	2	33
Rosenbrock	$ACO_{\mathbb{R}}$	35,04	27,057	17	188
	$ACO_{\mathbb{R}}$v1	227,02	116,072	61	596
	$ACO_{\mathbb{R}}$v2	9,8	10,853	4	63
Sphere	$ACO_{\mathbb{R}}$	32,36	18,561	18	131
	$ACO_{\mathbb{R}}$v1	108,42	73,267	39	396
	$ACO_{\mathbb{R}}$v2	21,2	13,26	11	83
Tablet	$ACO_{\mathbb{R}}$	36,98	17,303	24	112
	$ACO_{\mathbb{R}}$v1	**	**	**	**
	$ACO_{\mathbb{R}}$v2	34,44	14,157	21	110
Zakharov	$ACO_{\mathbb{R}}$	7,12	8,578	3	49
	$ACO_{\mathbb{R}}$v1	13,9	9,228	8	57
	$ACO_{\mathbb{R}}$v2	6,62	5,624	2	32

** O número máximo de iterações foi excedido sem alcançar a precisão.

valência ou melhora no que diz respeito ao tempo de execução na comparação entre as técnicas.

A proposta 2 para a visibilidade mostrou ser uma extensão robusta e eficiente. Apresentou o comportamento esperado ao ser submetida a todos os problemas propostos, resolvendo todas estas funções. Foi capaz de acelerar a velocidade de convergência diminuindo não só o número de iterações até atingir o objetivo, mas também diminuindo o tempo de execução do algoritmo até que a solução desejada fosse encontrada na maioria dos problemas aos quais foi submetida.

Já em relação aos parâmetros usados, é importante dizer que os valores utilizados em ambos os experimentos não são os melhores para todos os problemas. A ideia foi apresentar valores com bons comportamentos e compará-los de forma a evidenciar a diferença e a importância dessa escolha, entretanto cada problema possui seus próprios valores ótimos para os parâmetros e os aqui apresentados são apenas boas sugestões. Maiores estudos devem ser feitos em relação aos parâmetros e dependendo da necessidade,

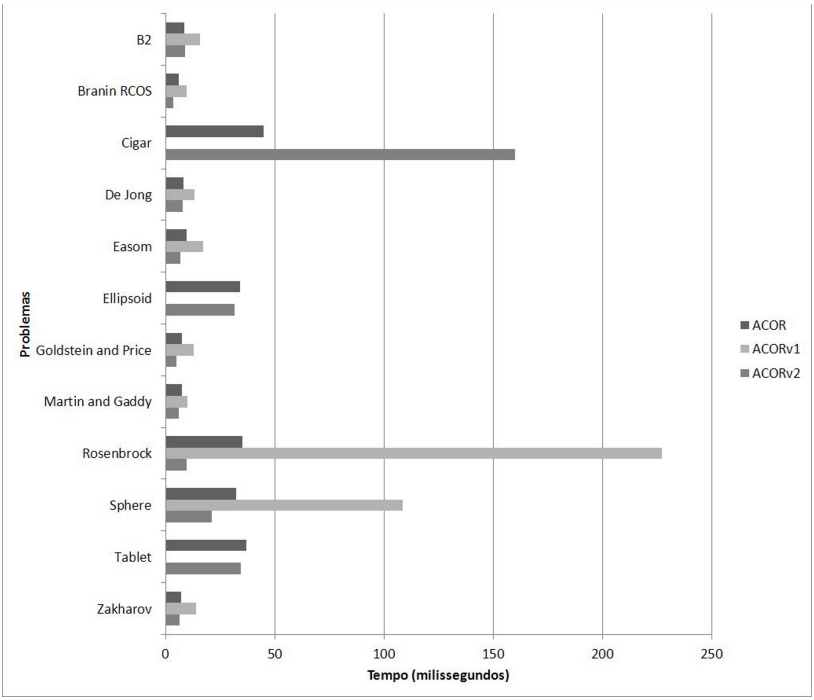

Figura 14: Gráfico de barras referente ao tempo de execução (em milissegundos) para o experimento 2.

devem ser analisados caso a caso (para cada problema).

5 CONCLUSÃO

Este trabalho apresentou a importância do termo heurístico, conhecido como *visibilidade*. A visibilidade faz parte da proposta original de ACO feita para domínios discretos, entretanto, as técnicas para domínios contínuos que mais se assemelham a ideia original, não fazem uso desse recurso. Aplicando a visibilidade a algoritmos baseados em ACO, tanto para resolver problemas de domínios discretos quanto para problemas de domínios contínuos, o termo influencia na velocidade de convergência dos algoritmos e na qualidade dos resultados encontrados.

Entre as propostas de implementação do ACO em domínios contínuos, $ACO_{\mathbb{R}}$ foi escolhido por ter maior similaridade com os aspectos conceituais do ACO original. Apesar disso, esta técnica não explora o termo de visibilidade no algoritmo. Este trabalho propôs duas extensões diferentes para o $ACO_{\mathbb{R}}$, com interpretações de como a visibilidade poderia ser implementada em domínios contínuos. Estas extensões foram idealizadas buscando um comportamento semelhante ao seu uso em domínios discretos.

As extensões propostas foram implementadas e diversos testes foram realizados usando problemas de otimização que são referências na área de problemas contínuos. Estes problemas são os mesmos usados por Socha e Dorigo (2006), que haviam aplicado-os no $ACO_{\mathbb{R}}$ original. Os resultados mostram melhora na velocidade de convergência, refletida em um número menor de iterações necessárias até atingir bons resultados, em ambas as extensões propostas. A visibilidade proposta para o $ACO_{\mathbb{R}}$ apresentou melhora de desempenho do algoritmo, similar ao do ACO clássico.

Apesar das extensões realizarem mais instruções por iteração que o $ACO_{\mathbb{R}}$, a significante redução no número de iterações refletiu em um tempo de execução inferior do algoritmo proposto. Também foi mostrado que, para os parâmetros do algoritmo, existem diferentes valores do que aqueles propostos no trabalho do $ACO_{\mathbb{R}}$ original (SOCHA; DORIGO, 2006) com maior eficiência na velocidade de convergência para o mesmo conjunto de problemas. São parâmetros que, assim como a heurística de visibilidade, influenciam de maneira significativa o desempenho do algoritmo. Além disso, diferentes interpretações da visibilidade retornaram diferente efetividade nos resultados. A proposta 2 mostrou ser mais robusta, apresentando influência similar em todos os problemas testados.

5.1 PERSPECTIVAS PARA TRABALHOS FUTUROS

No algoritmo AS baseado no ACO clássico e apresentado por Dorigo, Maniezzo e Colorni (1996), a matriz de feromônio do ambiente é inicializado com a mesma quantidade da substância em todas as arestas do grafo. Assim, nas primeira iterações, a visibilidade é a principal influência a guiar o algoritmo para determinadas áreas do domínio. No $ACO_{\mathbb{R}}$, o arquivo população é inicializado com soluções geradas aleatoriamente, o que pode acelerar a convergência (se as soluções aleatórias estiverem em boas regiões do domínio) ou retardar a convergência do algoritmo (se as soluções geradas aleatoriamente estiverem em regiões com baixos valores para a função objetivo). Portanto, uma heurística para a inicialização do arquivo população, ao invés de valores aleatórios, é um tema a ser explorado.

O tempo de execução do algoritmo com a segunda extensão proposta para a visibilidade, apesar da implementação realizar algumas instruções a mais, foi menor devido ao significativo decremento no número de iterações. Por outro lado, a primeira proposta de visibilidade, teve tempo de execução superior ao $ACO_{\mathbb{R}}$ original para alguns dos problemas testados, mesmo diminuindo o número de iterações até atingir o objetivo. Isso ocorreu porque a proposta 1 não obteve uma redução de iterações tão grande como a proposta 2. Diferentes implementações da busca local, otimizando a quantidade de instruções extras que são realizadas, é outro assunto que pode resultar em novos trabalhos.

A proposta 2, apesar de ser idealizada sob os mesmos conceitos que a proposta 1, apresentou maior robustez, sendo significativamente eficiente para todos os problemas testados. Diferentes interpretações da visibilidade em domínios contínuos podem ser novos tópicos de pesquisas futuras.

REFERÊNCIAS BIBLIOGRÁFICAS

AHUJA, R. K. et al. Faster algorithms for the shortest path problem. *Journal of the ACM*, v. 37, n. 2, p. 213–223, abr. 1990. ISSN 00045411.

BILCHEV, G.; PARMEE, I. C. The ant colony metaphor for searching continuous design spaces. *AISB Workshop on Evolutionary Computation*, v. 993, p. 25–39, 1995.

BOX, G. E. P.; MULLER, M. E. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, v. 29, n. 2, p. 610–611, 1958.

CAPRARA, A.; FISCHETTI, M.; TOTH, P. A Heuristic Method for the Set Covering Problem. *Operations Research*, v. 47, n. 5, p. 730–743, set. 1999. ISSN 0030-364X.

CERNý, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, v. 45, n. 1, p. 41–51, jan. 1985. ISSN 0022-3239. Disponível em: <http://www.springerlink.com/index/10.1007/BF00940812>.

CORMEN, T. H. et al. Greedy algorithms. In: ____. *Introduction to Algorithms*. 3rd. ed. [S.l.]: MIT Press, 2009. cap. 16.

DORIGO, M. *Optimization, Learning and Natural Algorithms*. Tese (Doutorado) — Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992.

DORIGO, M.; BONABEAU, E.; THERAULAZ, G. Ant algorithms and stigmergy. *Future Generation Computer Systems*, v. 16, n. 8, p. 851–871, jun 2000.

DORIGO, M.; CARO, G. D.; GAMBARDELLA, L. M. Ant algorithms for discrete optimization. *Artificial Life*, v. 5, n. 2, p. 137–172, jan 1999.

DORIGO, M.; GAMBARDELLA, L. M. Ant colonies for the travelling salesman problem. *BioSystems*, v. 43, n. 2, p. 73–81, jul 1997a.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 1, n. 1, p. 53–66, 1997b. ISSN 1089-778X.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. *Positive feedback as a search strategy*. [S.l.], 1991.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, v. 26, n. 1, p. 29–41, jan 1996.

DORIGO, M.; STüTZLE, T. *Ant Colony Optimization: Overview and Recent Advances*. [S.l.], maio 2009.

DRéO, J.; SIARRY, P. A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. In: DORIGO, M.; CARO, G. D.; SAMPELS, M. (Ed.). *Ant Algorithms*. [S.l.]: Springer Berlin / Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2463). p. 216–221.

GLOVER, F. Tabu search - part i. *INFORMS JOURNAL ON COMPUTING*, v. 1, n. 3, p. 190–206, 1989. Disponível em: <http://joc.journal.informs.org/cgi/content/abstract/1/3/190>.

GLOVER, F. Tabu search - part ii. *INFORMS Journal on Computing*, v. 2, n. 1, p. 4–32, jan. 1990.

GLOVER, F. W.; LAGUNA, M. *Tabu Search*. [S.l.]: Springer, 1997. 408 p.

HOLLAND, J. *Adaptation in natural and artificial systems*. Univ. of Michigan Press, Ann Arbor, 1975.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. [S.l.]: University of Michigan Press, 1992a. 228 p.

HOLLAND, J. H. Genetic algorithms. *Scientific American*, v. 267, n. 1, p. 66–72, 1992b.

HUANG, H.; HAO, Z. Aco for continuous optimization based on discrete encoding. *Ant Colony Optimization and Swarm Intelligence*, p. 504–505, 2006.

IBARRA, O. H.; KIM, C. E. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *Journal of the ACM*, v. 22, n. 4, p. 463–468, out. 1975. ISSN 00045411.

JALALI, M. R.; AFSHAR, A.; MARIñO, M. A. Multi-Colony Ant Algorithm for Continuous Multi-Reservoir Operation Optimization Problem. *Water*

Resources Management, v. 21, n. 9, p. 1429–1447, nov. 2006. ISSN 0920-4741. Disponível em: <http://www.springerlink.com/index/10.1007/s11269-006-9092-5>.

KERN, S. et al. Learning probability distributions in continuous evolutionary algorithms - a comparative review. *Natural Computing*, v. 3, n. 1, p. 77–112, 2004. ISSN 1567-7818.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science (New York, N.Y.)*, v. 220, n. 4598, p. 671–80, maio 1983. ISSN 0036-8075. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/17813860>.

KONG, M.; TIAN, P. A binary ant colony optimization for the unconstrained function optimization problem. In: HAO, Y. et al. (Ed.). *Computational Intelligence and Security*. [S.l.]: Springer Berlin / Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3801). p. 682–687.

KONG, M.; TIAN, P. Application of aco in continuous domain. *Advances in Natural Computation*, p. 126–135, 2006a. Disponível em: <http://www.springerlink.com/index/q524765u115wq162.pdf>.

KONG, M.; TIAN, P. A direct application of ant colony optimization to function optimization problem in continuous domain. *Ant Colony Optimization and Swarm Intelligence*, p. 324–331, 2006b.

LAPORTE, G. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, v. 59, n. 2, p. 231–247, jun. 1992. ISSN 03772217.

MADADGAR, S.; AFSHAR, A. An Improved Continuous Ant Algorithm for Optimization of Water Resources Problems. *Water Resources Management*, v. 23, n. 10, p. 2119–2139, nov. 2008. ISSN 0920-4741.

MATHUR, M. et al. Ant colony approach to continuous function optimization. *Industrial & Engineering Chemistry Research*, v. 39, n. 10, p. 3814–3822, 2000.

MEYER, B. On the convergence behaviour of ant colony search. *Asia-Pacific Conference on Complex Systems, Cairns*, v. 12, p. 1–15, 2004. Disponível em: <http://www.complexity.org.au/ci/vol12/msid05/file.pdf>.

MONMARCHè, N.; VENTURINI, G.; SLIMANE, M. On how pachycondyla apicalis ants suggest a new search algorithm. *Future Generation Computer Systems*, v. 16, n. 9, p. 937–946, 2000.

OPTIMIZATION. fev. 2011. Encyclopædia Britannica Online. Acessado 03-Fevereiro-2011. Disponível em: <www.britannica.com/EBchecked/topic/430575/optimization>.

PERRETTO, M.; LOPES, H. S. Reconstruction of phylogenetic trees using the ant colony optimization paradigm. *Genetics and Molecular Research*, v. 4, n. 3, p. 581–589, 2005.

SERGIENKO, V.; SHYLO, V. P. Problems of discrete optimization: challenges and main approaches. *Cybernetics and System Analysis*, v. 42, n. 4, p. 465–482, 2006.

SOCHA, K. ACO for continuous and mixed-variable optimization. *Ant Colony Optimization and Swarm Intelligence*, p. 25–36, 2004.

SOCHA, K.; BLUM, C. *Ant Colony Optimization*. [S.l.], 2006.

SOCHA, K.; DORIGO, M. Ant colony optimization for continuous domains. *European Journal of Operational Research*, v. 185, p. 1155–1173, 2006.

SOCHA, K.; DORIGO, M. *Ant Colony Optimization for Mixed-Variable Optimization Problems*. [S.l.], out. 2007.

STüTZLE, T.; HOOS, H. H. MAX-MIN ant system. *Future Generation Computer Systems*, v. 16, p. 889–914, 2000.

STüTZLE, T.; HOOS, H. H. MAX-MIN ant system and local search for the traveling salesman problem. *Evolutionary Computation, 1997., IEEE*, 2002.

TSALLIS, C.; STARIOLO, D. Generalized simulated annealing. *Physica A: Statistical and Theoretical Physics*, Elsevier, v. 233, n. 1-2, p. 395–406, ago. 1996. ISSN 0378-4371. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0378437196002713>.

WERBOS, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Tese (Doutorado) — Harvard University, Cambridge, MA, Aug 1974.

ZHAO, Y.; WANG, J.; XIE, X. Continuous Ant Colony Algorithm Based on Entity and Its Convergence. *2008 Second International Symposium on Intelligent Information Technology Application*, Ieee, p. 80–84, dez. 2008. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4739539>.