UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TIAGO BOECHEL

ALGORITMO DE OTIMIZAÇÃO: UMA ABORDAGEM HÍBRIDA UTILIZANDO O ALGORITMO DAS FORMIGAS E GENÉTICO.

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. José Mazzucco Junior, Dr.
Orientador
mazza@inf.ufsc.br

Florianópolis, Agosto de 2003

RESUMO

Este trabalho trata do desenvolvimento de um método alternativo para a resolução de problemas de otimização. A busca por soluções para este tipo de problema requer a descoberta de novos algoritmos eficientes, capazes de encontrar soluções aceitáveis, o que não garante que a mesma seja a melhor.

Esta abordagem tem como base dois importantes modelos computacionais utilizados na otimização de problemas: o algoritmo das formigas e o algoritmo genético. O primeiro é baseado na estratégia utilizada pelas formigas na busca de alimento, e o segundo na evolução natural das espécies.

A estratégia investiga a potencialidade de um método híbrido baseado na combinação do algoritmo das formigas e do algoritmo genético. A avaliação do desempenho do método, foi realizada utilizando o problema do caixeiro viajante, e os resultados obtidos são demonstrados neste trabalho.

ABSTRACT

This work treats of the development of an alternative method to the resolution of the optimization problems. The search for solutions to this problem type, requests the discovery of new efficient algorithms, capable to find acceptable solutions, what doesn't guarantee that the same is the best.

This approach has as base two importants models to computers used in the optimization of problems: the ants' algorithm and the genetic algorithm. The first is based on the strategy used by the ants in the food search, and the second on the natural evolution of the species.

The strategy investigates the potentiality of a hybrid method based on the combination of the ants' algorithm and the genetic algorithm. The evaluation of the acting of the method was accomplished using the traveling salesman problem, and the obtained results are demonstrated in this work.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE GRÁFICOS

Capítulo 1

Neste capítulo é apresentada uma introdução sobre o trabalho desenvolvido, a motivação para a escolha do tema abordado, os objetivos e também a estrutura de organização do texto.

1.1 Introdução

A pesquisa sobre modelos computacionais inteligentes tem, nos últimos anos, se caracterizado pela tendência em buscar a inspiração na natureza, onde existem inúmeros exemplos vivos de processos considerados “inteligentes”. Para cientistas de computação, matemáticos e engenheiros, muitas das soluções que a mãe natureza encontrou, para problemas complexos de adaptação, fornecem modelos práticos interessantíssimos. Embora não se possa afirmar que tais soluções sejam todas ótimas, não há a menor dúvida que os processos naturais, em particular os relacionados diretamente com os seres vivos, sejam soberbamente bem concebidos e adequados ao nosso mundo (TANOMARU, 1995).

Físicos, biólogos e outros cientistas tentam desvendar os princípios que regem os fenômenos da natureza, enquanto que os matemáticos, cientistas de computação e engenheiros buscam idéias que possam ser copiadas ou pelo menos imitadas, e então aplicadas a problemas que a ciência atual ainda não consegue resolver satisfatoriamente (TANOMARU, 1995).

Inspirado na natureza, a área de otimização teve um considerável aumento em estudos baseados em tais modelos. Um dos principais motivos que levam ao estudo destes modelos é o fato de que a maioria dos problemas de otimização de larga escala não

possuem uma solução ótima em um tempo viável, podendo apenas ser resolvido de forma aproximada. No contexto de otimização, grande parte dos problemas são ditos NP-Completos (um problema não polinomial, indicando que o espaço de busca por soluções cresce exponencialmente com as dimensões do problema) (LAARHOVEN & AARTS, 1987).

A estratégia de colônia inspirada no comportamento de colônias de formigas no processo de busca de alimento, ou seja, no comportamento coletivo de colônias de insetos operam com a idéia da comunicação indireta explorada pelas sociedades de insetos e formam algoritmos distribuídos de multi-agentes. Esta estratégia está sendo aplicada a vários problemas de otimização de cadeias de telecomunicações, distribuição de tarefas e principalmente em problemas de otimização combinatória.

1.2 Motivação

A motivação para trabalhar com a otimização de estruturas utilizando algoritmos baseados em formigas e algoritmos genéticos, originou-se do conhecimento de fatos curiosos e interessantes que ocorrem com perfeição na natureza e que levaram, e ainda hoje o fazem por muitas vezes, os seres humanos neles se inspirarem em busca de soluções para problemas do seu meio. Podemos citar vários exemplos destas inspirações decorrentes da natureza como: pássaros levaram aos aviões, morcegos aos sonares, peixes aos submarinos, etc.

O fato curioso na natureza foi a descoberta realizada por pesquisadores que ao estudarem o comportamento de colônias de formigas observaram que mesmo sendo elas seres tão simples e irracionais possuem mecanismos naturais de otimização. Isto é, são capazes de encontrar um caminho mais curto entre o ninho e uma fonte de comida sem usar

sugestões visuais, mesmo que ocorram mudanças no ambiente original como a introdução de um obstáculo.

A explicação para a descoberta do caminho ótimo a ser percorrido pelas formigas é que elas depositam durante suas caminhadas, uma certa quantidade de feromônio, aproximadamente com taxa fixa através da urina, e preferem seguir uma trajetória probabilisticamente mais rica em tal substância.

Quando o caminho original é interrompido pela introdução de um obstáculo, as mesmas se dividem pelas alternativas possíveis e, com o decorrer do tempo, como pelo menor caminho terão passado mais formigas, ficando o mesmo com uma maior concentração de feromônio, este novo trajeto será então adotado.

A idéia e a necessidade de otimização, exemplificada por simples formigas, em conjunto com a seleção natural dos mais aptos no processo de evolução torna-se um campo interessante pelos inúmeros resultados de sucesso apresentados nas mais diferentes áreas de aplicação, uma alternativa de grande força por suas características: robustez, flexibilidade, eficácia e, principalmente, simplicidade.

1.3 Objetivo do Trabalho

O principal objetivo deste trabalho é a elaboração de um método de otimização híbrido baseado no algoritmo *Ant Colony*, inspirado no comportamento coletivo de colônias de formigas na busca de alimento, e *Algoritmo Genético*, fundamentado na teoria de seleção natural e mecanismos da genética.

1.4 Organização do Texto

O trabalho foi dividido em 7 capítulos, organizado da seguinte forma:

O capítulo 2 introduz o assunto de otimização, apresentando a formulação e os conceitos básicos do tema, e algumas alternativas disponíveis para o seu tratamento.

No capítulo 3, é abordada a estratégia empregada por sociedades de formigas na captura de alimentos, como se comportam e suas particularidades. Neste capitulo o algoritmo *Ant System* e apresentado e detalhado.

O capítulo 4 descreve os fundamentos dos algoritmos genéticos, sua fonte de inspiração, fundamentos e particularidades. São detalhados os procedimentos do algoritmo, os operadores e as configurações necessárias para o emprego deles.

O capítulo 5 apresenta o modelo proposto, uma abordagem que combina o algoritmo das formigas e algoritmo genético para a otimização de problemas combinatoriais. Inicialmente são apresentadas as definições do problema, do método existente e, por fim, o modelo proposto é detalhado.

No capítulo 6 é descrita a metodologia de avaliação do modelo proposto, as formas de se obter os resultados e, os resultados computacionais obtidos com a aplicação do modelo proposto são comparados com resultados obtidos pelo algoritmo original.

O trabalho termina no capítulo 7, onde são apresentadas as conclusões finais e propostas para trabalhos futuros.

CAPÍTULO 2

Otimização

Este capítulo apresenta uma visão geral sobre otimização. São vistos alguns conceitos básicos, critérios, métodos de otimização e também alguns exemplos de problemas de otimização mais comuns.

2.1 Conceitos Básicos

Uma grande parte dos problemas científicos pose ser formulada como problema de busca e otimização: basicamente, existe uma série de fatores influenciando o desempenho de um dado sistema, e tais fatores podem assumir um número ilimitado de valores, e podem estar sujeitos a certas restrições. O objetivo é encontrar a melhor combinação dos fatores, ou seja a combinação de fatores que proporcione o melhor desempenho possível para o sistema em questão. Em termos técnicos, o conjunto de todas as combinações possíveis para os fatores constitui o espaço de busca. Não é difícil perceber que existe uma dualidade entre os conceitos de busca e otimização, de tal modo que todo problema pode ser considerado um problema de otimização e vice-versa (TANOMARU, 1995).

Um problema de otimização pode ser tanto um problema de minimização ou maximização, dependendo do problema, e é especificado por um conjunto de instâncias do problema (AARTS & KORST, 1989).

Sem perda de generalidade, podemos considerar somente problemas de maximização, pois eles podem ser facilmente convertidos em problemas de minimização e vice-versa, através de artifícios bem simples como, por exemplo, multiplicar a função por (-1) (TANOMARU, 1995).

Entende-se por otimização, não um processo de busca do melhor absoluto, mas a procura sistemática do melhor prático. Algumas vezes, conhecer o mecanismo de um certo problema, determinando as relações entre suas variáveis, já constitui um avanço considerável no caminho de sua solução (NOVAES, 1978).

O processo de otimização pode ser descrito da seguinte forma: a determinação de uma ação que proporciona um máximo de benefício, medido por um critério de desempenho pré-estabelecido (YONEYAMA & NASCIMENTO, 2000).

O conceito de otimização de qualquer tarefa ou processo está diretamente ligado à sua realização, do modo mais eficiente possível. Esta eficiência pode ser avaliada de inúmeras maneiras, conforme o tipo de tarefa a ser realizada. Podendo ser, por exemplo:

- A minimização do tempo gasto na limpeza de um jardim;
- A maximização da velocidade de processamento de um chip eletrônico;
- A minimização do custo de execução de um projeto de barragem, dentre muitos outros.

Estes problemas estão naturalmente divididos em duas categorias: os problemas com variáveis contínuas e os problemas com variáveis discretas, chamados combinatórios. Nos problemas contínuos o que se procura é um conjunto de números reais ou mesmo uma função, nos problemas combinatórios o que se procura é um objeto dentre um finito ou contabilmente infinito conjunto de prováveis soluções (PAPADIMITRIOU, 1982).

Uma importante descoberta no campo de otimização, obtida no final da década de 1960, é a suposição, ainda não verificada, de que existe uma classe de problemas de otimização de tal complexidade que qualquer algoritmo, resolvendo cada instância do problema para um resultado ótimo, requer um esforço combinatório que cresce super-polinomialmente com o tamanho do problema (AARTS & KORST, 1989).

Entretanto estes problemas ainda precisam ser resolvidos e na construção de algoritmos para solução dos mesmos pode-se optar por dois caminhos. Um caminho que

ainda busca por soluções ótimas sobre o risco de tempos de execução muito longos ou possivelmente impraticáveis. Outro caminho que busca por soluções rápidas sobre o risco de soluções sub-ótimas (soluções próximas às ótimas) em um tempo de processamento aceitável (TANOMARU, 1995).

Adicionalmente os dois casos ainda podem ser distinguidos entre algoritmos gerais e algoritmos costurados (*tailored*). Os algoritmos gerais podem ser aplicados a uma grande variedade de problemas podendo ser chamados de independentes do problema, já os algoritmos costurados usam informações específicas acerca do problema tornando sua aplicação limitada a um número restrito de problemas (TANOMARU, 1995).

2.2 Critérios de Otimização

As técnicas de otimização são baseadas em duas premissas implícitas de conseqüências importantes na aplicação prática. Em primeiro lugar admite-se que possa ser definida uma "função objetivo", (função que exprime através de uma escala única a medida de mérito do sistema analisado, ou seja, o quanto à solução analisada é boa dentro do problema em questão). A segunda, refere-se ao caráter determinístico da avaliação: admite-se que as relações entre as variáveis dependentes e os parâmetros independentes ocorram deterministicamente, em outras palavras, um determinado conjunto de valores das variáveis independentes deve produzir apenas um resultado na função objetivo (NOVAES, 1978).

Na prática estas premissas freqüentemente não são satisfeitas. Em muitos casos observa-se uma pluralidade de objetivo a satisfazer. Este problema pode ser resolvido buscando uma medida de custo aproximada (mas única), efetuando-se uma análise de sensibilidade dos resultados para se levar em conta as eventuais características aleatórias das variáveis envolvidas (NOVAES, 1978).

2.3 Métodos de Otimização

Segundo TANOMARU (1995), há essencialmente três correntes de métodos gerais de otimização: métodos probabilísticos, numéricos e enumerativos. Existe, também, um grande número de métodos híbridos.

Métodos probabilísticos: são métodos que empregam a idéia de busca probabilística, o que não quer dizer que são métodos totalmente baseados em sorte, como é o caso dos chamados métodos aleatórios.

Métodos numéricos: podem ser divididos em analíticos ou baseados em cálculos numéricos. Os métodos analíticos de otimização são aplicáveis basicamente quando a função de custo *f* é explicitamente conhecida e derivável, ou pode ser aproximada por alguma função derivável até o grau desejado de precisão. Nestes casos basta resolver as equações que resultam de igualar as derivadas da função a zero dentro do espaço de busca. Nos métodos baseados em cálculo numérico, quando o espaço de busca é linear e, portanto, convexo, técnicas de pesquisa operacional, como o método *simplex*, são suficientes. Já em ambientes não-lineares, técnicas de gradientes ou de estatísticas de alta ordem são geralmente empregadas.

Métodos enumerativos: Estes métodos examinam cada ponto do espaço de busca, um por um, em busca de pontos ótimos. A idéia pode ser intuitiva, mas é obviamente impraticável quando há um número infinito ou extremamente grande de pontos a examinar.

2.4 Problemas Conhecidos

Muitos problemas de otimização conhecidos são NP, ou seja, não polinomiais, indicando que o espaço de busca cresce exponencialmente com as dimensões do problema.

Embora se acredite que os problemas considerados NP são e serão sempre intratáveis, não importa quanto a ciência dos computadores evolua, há um resultado interessante que diz que, se um algoritmo eficiente para um determinado problema existir, então o algoritmo poderá ser modificado para resolver todos os outros problemas NP também (PAPADIMITRIOU, 1985).

Por conseguinte, é sempre interessante aplicar técnicas de otimização a problemas NP clássicos, de modo que se possa ter uma idéia do potencial de cada método para resolver problemas combinatoriais que aparecem na prática .

Exemplos clássicos incluem o problema do caixeiro viajante, o problema da mochila, o problema do depósito, o roteamento de veículos, entre outros.

2.4.1 O Problema do Caixeiro Viajante

O problema do caixeiro viajante é um dos mais conhecidos na área de otimização. Trata-se de um vendedor ambulante que deve visitar um conjunto de cidades, partindo de uma cidade inicial, passando por todas as demais, uma única vez, e retornar a origem cobrindo a menor distância possível. Cada cidade deve ser visitada somente uma única vez.

Na prática, um problema com *n* cidades tem seu espaço de busca definido pela permutação:

$$S = (n - 1) ! / 2 \qquad (1)$$

Onde *S* é o número de possíveis rotas que podem ser percorridas, o que dá um número formidável mesmo para poucas cidades. Para $n = 30$, por exemplo, há um total de $4{,}42 \times 10^{30}$ rotas distintas. Com um computador capaz de analisar um milhão de rotas por segundo, a busca exaustiva levaria o equivalente a 10^7 vezes a idade do universo (TANOMARU, 1995).

Na figura abaixo temos uma instância de um problema do caixeiro viajante com oito cidades e uma possível solução definida *I = (1, 2, 3, 4, 5, 6, 7, 8)*. Aqui o espaço de

busca *S* é definido, pela permutação das 8 cidades somando um total de 2.520 rotas distintas. A função *f* a ser minimizada busca encontrar o menor custo para percorrer todas as cidades entre as possíveis rotas.

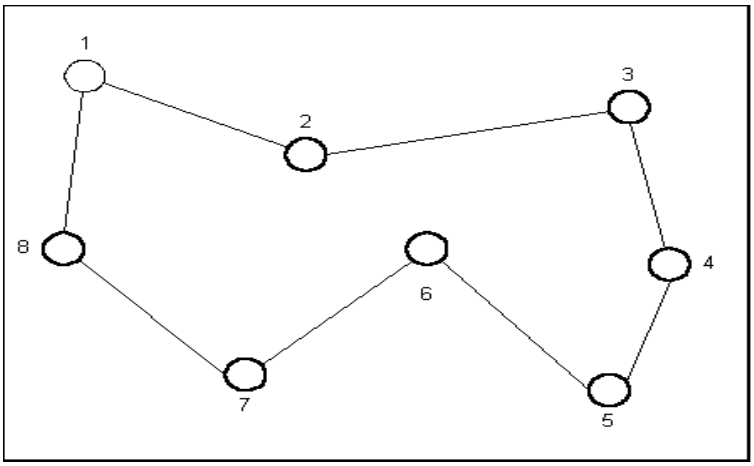

Figura 1 - Instância de um problema do caixeiro viajante.

Segundo BRASSARD & BRATLEY (1996), a solução do problema do caixeiro viajante, utilizando-se qualquer algoritmo conhecido, se torna impraticável para largas instâncias do problema.

A popularidade do problema do caixeiro viajante provavelmente se dá ao fato de que é um problema muito fácil de ser entendido e visualizado, mas porém muito difícil de ser resolvido. Muitos outros problemas da classe NP não são somente de difícil resolução, mas como também de difícil compreensão.

Em instâncias de problemas com um número *n* de cidades, existem *(n-1)!* possíveis soluções. Para exemplos pequenos não se torna nenhum problema gerar todas as possíveis soluções e escolher o caminho mais curto. Mas quando o número de cidades aumenta, as possíveis soluções “explodem”. É ai então que precisamos de métodos mais inteligentes e viáveis computacionalmente para resolver, ou pelo menos achar soluções aceitáveis.

A procura por soluções razoáveis é um outro modo de ver os problemas de otimização como o do caixeiro viajante. Em vez de tentar achar a melhor solução realizando infinitos cálculos em uma quantidade de tempo que talvez nem se possa estimar, achar uma solução aceitável em um espaço de tempo mais curto do que seria necessário para resolver de fato o problema.

A estratégia de resolução usada nestes casos é uma estimativa dos custos envolvidos achando entre as possíveis soluções um substituto ótimo como resultado.

O problema do caixeiro viajante é um dos mais importantes e estudados problemas de otimização. A importância de encontrar um algoritmo aproximado para este problema reside no fato de muitos problemas que ocorrem na ciência da computação, e em muitas outras áreas, poderem ser modelados a partir do mesmo. Outra importante característica é a grande dificuldade de se achar uma solução exata para instâncias maiores, visto que é um dos problemas de otimização mais difíceis, dentre os conhecidos. Nas últimas décadas, vem crescendo o desenvolvimento de algoritmos aproximados, geralmente utilizando métodos heurísticos que forneçam soluções aceitáveis em tempo de processamento compatível com as necessidades (LIN, 1965 e GOLDBARG, 2000).

Capítulo 3

Colônias de Formigas

No capítulo 3 é abordada a estratégia empregada por sociedades de formigas na captura de alimentos, como se comportam e suas particularidades. Neste capítulo o algoritmo das formigas é apresentado e detalhado.

3.1 Introdução

Insetos que vivem em sociedade, tais como formigas e abelhas, sobrevivem em praticamente toda a Terra, sendo que acompanharam a evolução da Terra. Sem dúvida, sua organização social, em particular o desenvolvimento genético do comprometimento de cada indivíduo para a sobrevivência da colônia, é um fator chave que sustenta seu sucesso (DORIGO et al, 2002).

Além disso, estas sociedades de insetos exibem a fascinante propriedade que as atividades dos indivíduos, assim como as atividades da sociedade como um todo, não são reguladas por nenhuma forma de controle centralizado. Forças evolucionárias geraram indivíduos que combinaram um total comprometimento com a sociedade junto com comunicação específica e habilidades de ação que estimulam o surgimento de padrões complexos e comportamentos em relação ao nível global (DORIGO et al, 2002).

Dentre os insetos que vivem em sociedade, as formigas podem ser consideradas a família que teve mais sucesso. Existem cerca de nove mil espécies diferentes, cada uma com diferentes conjuntos de características especializadas que lhe permitem viver em grande número e, virtualmente, em qualquer lugar.

A observação e estudo de formigas e sociedades de formigas há muito tempo vem atraindo a atenção de entomologistas e leigos, mas nos últimos anos, o modelo de organização e interação das formigas tem chamado o interesse dos engenheiros e cientistas da computação (DORIGO et al, 2002).

A presença simultânea e única das características da sociedade das formigas, tais como: autonomia dos indivíduos, controle completamente distribuído, direção e comunicação mediada pelo meio, surgimento de comportamentos complexos com relação ao repertório de uma única formiga, estratégias coletivas e cooperativas, e organização própria, tem feito dessa sociedade um atraente e inspirador modelo para construir novos algoritmos e novos sistemas multi-agentes.

Nos últimos anos, sociedades de formigas têm fornecido o impulso para um crescente corpo de trabalho científico, nos campos de robótica, pesquisa operacional, e telecomunicações. Pesquisadores e cientistas de todo o mundo tem feito progressos significantes, tanto na parte teórica quanto em implementações, possibilitando assim dar a este campo de pesquisa uma base sólida. Além disso, tem-se mostrado que o “caminho das formigas”, quando cuidadosamente construído e estudado pode resultar em aplicações de sucesso para muitos problemas do mundo real (DORIGO et al, 2002).

Assim como no campo dos algoritmos genéticos e redes neurais, para citar dois exemplos, a natureza parece oferecer uma fonte de idéias para o projeto de novos sistemas e algoritmos.

A estratégia empregada por sociedades de formigas na captura de alimentos tem inspirado um novo paradigma computacional chamado de *Ant Colony*. O modelo inicialmente proposto por DENEUBOURG et al, 1983, sugere que há um determinado grau de aleatoriedade na comunicação das formigas que minimiza o tempo de captura do alimento distribuído entre várias fontes.

Comportamentos globais muito complicados podem resultar da interação de sistemas dinâmicos individuais muito simples. Esses comportamentos podem ser observados em vários sistemas biológicos, como, por exemplo, em colônias de

formigas. A visão tradicional sobre o comportamento desses insetos enxerga-os como pequenos seres autômatos que obedecem a um programa genético estritamente estabelecido. Assim, supõe-se que eles possuem uma quantidade de "inteligência" individual suficiente para cooperarem de modo organizado. Essa visão se origina da idéia de que atividades complicadas estão necessariamente associadas a indivíduos complicados. Atualmente, uma nova interpretação vem ganhando força, segundo a qual os elementos aleatórios do meio ambiente e a maleabilidade do comportamento individual desempenham papéis fundamentais na organização do funcionamento (DENEUBOURG et al, 1983; DENEUBOURG et al, 1986) e da estruturação da sociedade (FRANKS & FRANKS, 1993; BONEBEAU et al, 1996).

O processo de procura e coleta de alimento realizado por colônias de formigas pode ser entendido como uma dessas atividades. O ambiente onde colônias de formigas vivem é mais ou menos previsível no tempo e no espaço. Pode-se comparar duas situações extremas em que fontes de alimento se apresentam: uma colônia de pulgões, como fonte durável (da ordem de meses), e um pássaro morto, que é uma fonte ocasional. A fim de explorar os pulgões, as formigas desenvolvem rotas estáveis entre o formigueiro e o ninho dos pulgões. Nessa situação, é benéfico que as formigas mantenham estruturas (rotas) permanentes com baixo nível de erro ou de flutuação, isto é, que poucas formigas "errem" o caminho (DENEUBOURG et al, 1983; DENEUBOURG et al, 1986).

Porém, depender de uma única fonte de alimento é muito arriscado para a sobrevivência da colônia. Assim, também é interessante explorar o pássaro morto, como fonte extra de alimento. Para isso, algumas formigas têm que "errar" o caminho original, isto é, deixar as rotas estáveis para que possam descobrir, ao acaso, a ave morta (DENEUBOURG et al, 1983; DENEUBOURG et al, 1986).

As formigas, como outros seres vivos, desenvolveram interações e mecanismos de comunicação que podem apresentar graus diversos de aleatoriedade que variam com a espécie e com as condições do meio. Um desses mecanismos é baseado na deposição de feromônio no percurso realizado, geralmente entre o formigueiro e a fonte de alimento.

Feromônio é o hormônio produzido pelas formigas cuja função é estabelecer um padrão de identificação e de comunicação entre elas. Uma quantidade inicial de feromônio depositado na trilha fará com que essa trilha seja detectada por outras formigas e, à medida que caminham pela trilha em direção à fonte de alimento, as formigas recém-atraídas também depositam feromônio. Tem-se, a partir daí, um processo de realimentação positiva no qual, quanto mais feromônio estiver presente na trilha, mais formigas perdidas serão atraídas e maior será a quantidade de feromônio depositado na trilha, deixando o sistema mais determinístico (PASTEELS et al, 1987).

Nos últimos anos tem se notado o surgimento de um significativo conjunto de técnicas e algoritmos computacionalmente muito eficientes, denominados heurísticos ou aproximativos baseados em técnicas de otimização de busca local para o tratamento de problemas NP-Completos. Embora eficientes, esses algoritmos não garantem uma solução ótima do problema devido a sua complexidade, isso demonstra cada vez mais a importância de se encontrar algoritmos aproximados para resolver esses tipos de problemas (GOLDBARG, 2000).

O comportamento natural das formigas inspirou a construção de um algoritmo no qual um conjunto de formigas artificiais cooperam para a solução de um problema através da troca de informação via o feromônio depositado sobre as trilhas. Esse algoritmo tem sido aplicado em problemas de otimização combinatória, como o problema do caixeiro viajante (DORIGO & GAMBARELLA, 1997). A analogia adotada advém do fato das formigas conseguirem restabelecer rotas interrompidas por obstáculos, de maneira que a nova rota é a mais curta possível.

Nesse algoritmo, formigas “virtuais” são enviadas para várias cidades e, após cada iteração, a quantidade de feromônio em cada trilha é atualizada proporcionalmente à distância percorrida pelas formigas, selecionando o percurso ótimo entre as cidades (DORIGO & GAMBARELLA, 1996).

Uma recente extensão dessa idéia é a elaboração de um método de busca em espaços contínuos (BILCHEV & PARMEE, 1997). Modelando-se a vizinhança do formigueiro (ponto de partida) por uma estrutura discreta, chega-se a um sistema computacional dinâmico – conhecido como algoritmo *Ant System* – que é empregado para solucionar problemas de otimização em espaços contínuos.

Ao estudar insetos sociais como vespas, abelhas e em particular formigas, podemos ver que as colônias são muito organizadas, entretanto parece que os insetos individualmente pouco sabem o que há ao redor dele, caminham uns atrás dos outros, fazendo todos as mesmas coisas. De alguma maneira na iteração entre essas entidades muito simples, padrões emergem. O termo usado para este fenômeno é auto-organização (KAUFMAN, 1995).

Auto-organização é basicamente um conjunto de mecanismos de estruturas dinâmicas e aparecem em um nível mais alto, porém só aparecem devido as iterações entre componentes em um outro nível mais baixo. Nestes meios não há nenhum supervisor para guiar os níveis mais baixos. Como resultado disto, as decisões feitas por estes componentes estão somente baseadas em informação local. As formigas não tem acesso ao conhecimento global contido no sistema como um todo.

Sistemas assim são achados em vários lugares e em diferentes níveis na natureza. Isto é uma indicação de que algoritmos baseados nestes sistemas podem ser muito poderosos. Podemos ver isso pela seleção natural, todos os sistemas são eficientes e robustos e apesar das dificuldades e mudanças no ambiente em que vivem, continuam vivendo e sempre se adaptando. Se eles não forem bastante eficientes e robustos, podem ser extintos e substituídos por outras espécies.

Como as formigas tiveram bastante êxito na evolução da terra, é visto que esta auto-organização acontece em vários níveis. Por exemplo, dentro da colônia, umas estão trabalhando enquanto outras estão fora procurando comida, são formadas redes de rastros entre o ninho e várias fontes de comida. Barreiras naturais, normalmente, não representam nenhum problema porque as formigas acham o modo como desviar ao redor deles. Estas redes de rastros normalmente são muito eficientes. Isso é uma das razões principais que inspira o uso de colônias de formigas como um exemplo para técnicas de otimização.

De acordo com BONABEAU et al (1999), há quatro aspectos importantes, mostrados com exemplos específicos em colônias de formigas:

- Avaliação positiva - Reforçar as soluções melhores. É necessário o uso de algum modo de avaliar as experiências positivas. Para formigas que procuram comida, os rastros deixados pelas outras é uma forma de avaliação positiva. Os

rastros de feromônio deixados são maiores para fontes de comida, e assim mais formigas irão lá o que sucessivamente aumentará a quantia de feromônio daquela trilha.

- Avaliação negativa - É preciso contrabalançar a avaliação positiva e estabilizar o padrão coletivo. Um exemplo é o esgotamento de fontes de comida. Essa avaliação negativa é embutida no algoritmo pela evaporação de feromônio, ou seja, os rastros diminuem com o passar do tempo.
- Amplificação de flutuações - Algum fator aleatório é freqüentemente crucial para achar soluções melhores. Precisam ser descobertas novas fontes de comida e isso ocorre quando uma formiga às vezes tenta um caminho novo, ou se perde, em vez de seguir o caminho velho acaba descobrindo um nova fonte de comida.
- Interações múltiplas - A auto-organização só funciona para o bem do conjunto se os indivíduos puderem interagir entre si. As formigas usam um rastro de feromônio como mecanismo para interagir umas com as outras. Os rastros são um exemplo de memória coletiva.

3.2 A Escolha do Caminho

Formigas são capazes de encontrar o caminho mais curto de uma fonte de comida até o ninho sem utilizar sinais de visão. Elas também são capazes de se adaptar a mudanças no meio em que estão, como, por exemplo, encontrando um novo caminho mais curto uma vez que o antigo não está mais viável por causa de um obstáculo.

Como ilustrado na Fig. 2, as formigas estão se movendo por uma linha reta que liga a fonte de comida ao ninho.

Figura 2 - Caminho entre o ninho e a fonte de comida.

O meio primário das formigas para formar e manter a linha é deixar um rastro através do depósito de uma substância denominada feromônio. Segundo ZONTA (2001), “feromônio é um sistema de comunicação química entre animais de várias espécies. Este sistema transmite informações essenciais de sobrevivência e reprodução, ou informações como idade, sexo e parentesco do animal emissor. O sistema é sentido pelo olfato do animal receptor, interpretado por ele para decidir qual o seu comportamento posterior”.

As formigas depositam uma certa quantidade de feromônio enquanto caminham, e cada formiga probabilisticamente prefere seguir uma direção rica em feromônio. Este comportamento elementar de formigas pode ser usado para explicar como elas podem encontrar o caminho mais curto que reconecta uma linha depois do aparecimento repentino de um obstáculo que interrompe o caminho original, conforme ilustrado na Fig. 3.

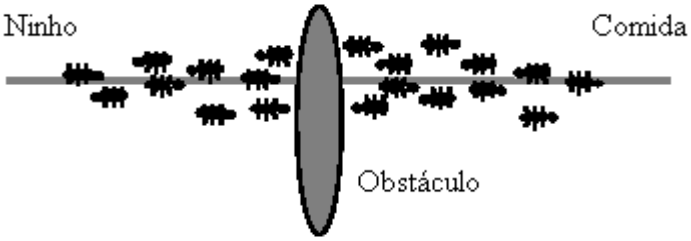

Figura 3 - Caminho entre o ninho e a fonte comida com obstáculo.

De fato, uma vez que o obstáculo aparece, aquelas formigas que estão em frente ao obstáculo não podem continuar a seguir a trilha de feromônio e portanto devem escolher entre virar à esquerda ou à direita. Nesta situação, pode-se esperar que metade das formigas escolham virar à esquerda e a outra metade à direita. Uma situação similar pode ser observada no outro lado do obstáculo, conforme ilustrado da Fig. 4.

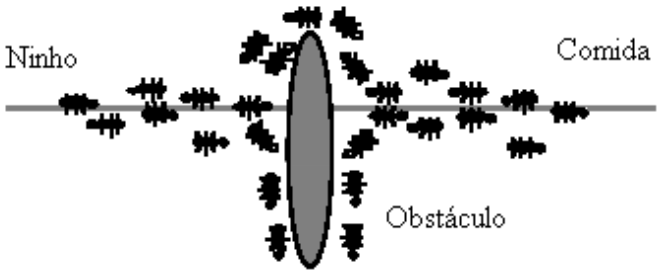

Figura 4 - Caminho percorrido pelas formigas com obstáculo.

É interessante perceber que aquelas formigas que escolheram, por acaso, o caminho mais curto em torno do obstáculo vão reconstituir mais rapidamente a trilha de feromônio e comparado às formigas que escolheram o caminho mais longo. Assim, o caminho mais curto vai receber uma quantidade maior de feromônio por unidade de tempo e, sendo assim, um número maior de formigas vai escolher o caminho mais curto.

Devido a este processo de *feedback* positivo (*autocatalytic*), todas as formigas irão rapidamente escolher o caminho mais curto, conforme ilustra a Fig. 5.

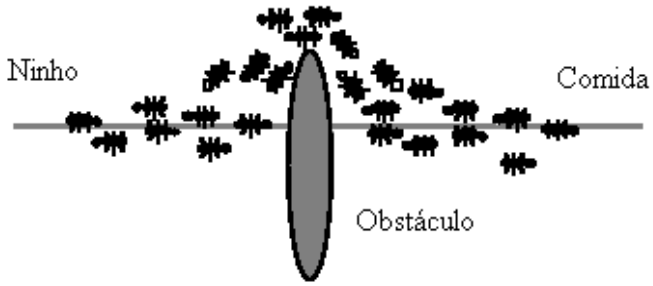

Figura 5 - Caminho escolhido pelas formigas.

O aspecto mais interessante neste processo é que encontrar o caminho mais curto em torno do obstáculo parece ser uma propriedade da interação entre a forma do objeto e o comportamento distribuído das formigas. Embora todas as formigas se movam, aproximadamente, a mesma velocidade e depositem a trilha de feromônio, aproximadamente, a mesma taxa, é fato que se leva mais tempo para contornar obstáculos pelo seu lado maior do que pelo seu lado menor, o que faz com que a trilha de feromônio acumule-se mais rápido no lado mais curto. É a preferência das formigas pelas trilhas com altas taxas de feromônio que faz com que a acumulação continue maior no lado mais curto do obstáculo.

Esta nova heurística pode ser usada com o propósito de encontrar soluções para diferentes problemas de otimização combinatorial devido a suas características. Neste trabalho é aplicada ao conhecido problema do caixeiro viajante.

3.3 O Algoritmo Ant System

O algoritmo *Ant System* foi o primeiro algoritmo baseado em colônias de formigas, e quando foi introduzido pela primeira vez, ele foi aplicado ao problema do caixeiro viajante (*TSP – Traveling salesman problem*). Este foi protótipo de outros algoritmos, os quais encontraram muitas aplicações interessantes e de sucesso. O problema do caixeiro viajante consiste basicamente em encontrar a menor rota de forma que esta passe em todas as cidades de um dado conjunto. No algoritmo *Ant System* formigas artificiais constroem soluções (rotas) para o caixeiro viajante movendo-se no grafo do problema de uma cidade para outra.

Dado um conjunto de *n* cidades, o caixeiro viajante pode ser definido como um problema que precisa encontrar uma maneira de visitar as *n* cidades uma única vez percorrendo a menor distância possível.

Para simular formigas artificiais algumas simplificações em relação à biologia são feitas:

- Para cada intervalo novas formigas são criadas e morrem logo após completar o caminho;
- As formigas se movem em uma velocidade constante (1 unidade de cada vez por exemplo);
- A cada intervalo de tempo um novo conjunto de formigas percorre o caminho;
- O feromônio deixado pelas formigas evapora completamente entre um intervalo e outro.

Uma instância do problema do caixeiro viajante é definida por um gráfico *(N, E)*, onde *N* é o conjunto de todas as cidades e *E* o conjunto de caminhos que podem ser percorridos (todas as ligações entre todas as cidades).

O comprimento do caminho entre duas cidades *i* e *j*, é representado por *Dij*, onde *Dij* é também chamado de distância euclidiana entre *i* e *j*.

$$D_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2} \qquad (2)$$

Sendo b*i(t)(i1, i2,..., in)* o número de formigas na cidade i no tempo t e sendo o número total de formigas. Cada formiga é um simples agente e age segundo as características a seguir:

- A formiga escolhe a cidade para onde vai de acordo com a probabilidade, sendo que esta é uma função de distância da cidade e da quantidade de feromônio presente na trilha;
- Para forçar as formigas a percorrerem caminhos legais, são proibidas de percorrer caminhos para cidades já visitadas, até que todas as formigas completem o caminho (isto é controlado por uma lista chamada tabulist).
- Quando uma formiga completa o caminho *(i j)*, deixa uma substância chamada feromônio no caminho percorrido.

Sendo *Tij* a intensidade do feromônio deixado no caminho (*i, j*) no tempo *t.* É gerada uma matriz de feromônio como mostra a tabela abaixo:

-	A	B	C	D	E
A	-	τ_{AB}	τ_{AC}	τ_{AD}	τ_{AE}
B	τ_{BA}	-	τ_{BC}	τ_{BD}	τ_{BE}
C	τ_{CA}	τ_{CB}	-	τ_{CD}	τ_{CE}
D	τ_{DA}	τ_{DB}	τ_{DC}	-	τ_{DE}
E	τ_{EA}	τ_{EB}	τ_{EC}	τ_{ED}	-

Tabela 1 - Matriz de feromônio.

A matriz de feromônio é atualizada somente depois que a formiga completa todo o seu caminho, de acordo com as fórmulas seguintes:

$$Tij = (1 - p) * Tij + 1 / L \quad (3)$$

$$Tij = (1 - p) * Tij \quad (4)$$

Onde $p \in [0, 1]$ é um coeficiente que representa a taxa de evaporação do feromônio entre os tempo *t* e *t + n* e *L* representa a melhor solução encontrada até então.

A atualização da matriz de feromônio é feita pelas fórmulas 2 e 3 sendo que a rota percorrida pela formiga no final do caminho é atualizada pela fórmula 2 intensificando o feromônio da trilha e os demais pontos da matriz são atualizados pela fórmula 3, que corresponde a evaporação do feromônio.

Cada novo passo dado por uma formiga depende do valor do feromônio deixado no caminho e da heurística utilizada, neste caso segundo a fórmula:

$$P_{ij} = [(T_{ij})^{\alpha} * (N_{ij})^{\beta}] / (\Sigma_{ij} (T_{ij})^{\alpha} * (N_{ij})^{\beta)} \quad (5)$$

Onde *Nij* é chamado de visibilidade e corresponde a quantidade *1 / dij*. Esta quantidade não é modificada durante a execução do algoritmo, ao contrario do feromônio que é alterado pelas formulas 2 e 3.

α e β são coeficientes utilizados como parâmetros que controlam a importância relativa entre a distância e a visibilidade. No entanto a probabilidade de transação é *trade-off* entre a visibilidade (que diz qual a cidade que deve ser escolhida com alta probabilidade) e a intensidade do feromônio em um tempo t (que diz que em um caminho (i,j) houve bastante tráfego, então é fortemente desejável, implementando assim o processo *autocatalytic*).

Para satisfazer a exigência que uma formiga precisa visitar todas as cidades e apenas uma vez, foram associadas aos caminhos, uma estrutura de dados chamada de tabulist, que evita que as cidades já visitadas pela formiga seja novamente visitada em

um mesmo tour. Após o término de um tour, ou seja, quando uma rota é completada, a lista usada para computar a solução atual da formiga, isto é a distância do caminho percorrido pela formiga. A lista é inicializada passando a fazer referência aos dados de um novo tour, a formiga esta livre para escolher novamente o caminho.

3.4 O Fluxo do Algoritmo

Tendo visto as definições do algoritmo, tratamos agora do fluxo do algoritmo e do seu comportamento durante a execução.

No tempo=0 acontece uma fase de inicialização, durante o qual são posicionadas formigas em cidades diferentes e um valor inicial para a intensidade de feromônio é fixado para todos caminhos, inicializando assim a matriz de feromônio.

Cada formiga possui uma estrutura associada e ela, chamada *tabulist*, que serve como um controle, para informar se a formiga visitou ou não aquela cidade. O primeiro elemento dessa lista é definido com o mesmo valor correspondente ao da cidade inicial. Cada passo dado pela formiga inserido nesta lista. A cada nova iteração a *tabulist* da formiga é inicializada novamente, recebendo valor zero o que inicia uma nova fase.

Depois da etapa de inicialização são definidos os primeiros movimentos da formiga. Cada movimento dado pela formiga é determinado através da função 4, que considera a intensidade do feromônio contido nas trilhas e a distância do próximo ponto. Como no tempo *t=0* o valor do feromônio contido na trilha ainda é igual a zero, o que influência o movimento inicial da formiga é a distância do ponto. A função 4 utiliza a informação deixada na trilha pela formiga, definida como *Tij (t)*, que corresponde ao feromônio depositado, e também considera a visibilidade *Nij*, complementando a tomada de decisão de qual caminho seguir. Como mostrado anteriormente as formigas agem muito mais pelo faro do que pelo que estão vendo, ou seja, são quase cegas mas não totalmente.

Depois que todas as formigas completaram o tour, a *tabulist* das formigas estará cheia, neste momento temos para cada formiga *k* o valor de *L (k)* utilizado na fórmula 2, ou seja a distância total dos caminhos percorridos pela formiga *k* sendo possível encontrar entre todas as *n* formigas o caminho mais curto percorrido naquela iteração.

For fim, encontrado um caminho adotado como melhor possível até então, é atualizada a matriz de feromônio, aplicando a fórmula 2 nos pontos pertencentes ao melhor caminho, intensificando a quantidade de feromônio na trilha do caminho percorrido, e aplicando a fórmula 3 no restante dos pontos, ou seja, evaporando parte do feromônio depositado nos pontos não pertencentes ao melhor tour.

Este processo continua até o contador de iterações alcançar o número máximo de ciclos NC_{Max} (valor pré-definido), ou todas as formigas estarem percorrendo a mesma rota. Chama-se este último caso de estado estagnado(*stagnation behavior)*, porque ele denota uma situação na qual o algoritmo para de procurar soluções alternativas, sendo que as mesmas já não são mais possíveis.

Fluxo do Algoritmo Ant System

```
Procedure  Ant System
Inicio
     t = 0
     Para todos os caminhos(i, j)
           ΔT_ij = 0
     Posicione as m formigas nas n cidades
     s = 1
     Para k := 1 to m faca
           tabu_k[ s ] = posicaoinicial[ k ]
     Repita ate que tabulist esteja completa
     Inicio
           s = s + 1
           Para k := 1 to m do
           Inicio
                 P_ij = [( T_ij )^α * ( N_ij )^β ] / ( Σ_ij ( T_ij )^α * ( N_ij )^β
                 move a formiga k para a cidade com a
probabilidade Pij
                 Insere a cidade j em tabu_k[ s ]
           Fim
           Para k:= 1 to m do
           Inicio
                 Move a formiga k de tabu_k[ n ] para tabu_k[ 1 ]
                 Compute o tamanho L_k descoberto pela formiga k
                 Atualiza o valor do melhor tour encontrado
           Fim
           Para cada caminho(i, j) faca
                 Se caminho(i, j) ∈ tabu
                    ΔTij = ( 1 – p )  *  Tij  + 1 / L
                 Senao
                    ΔTij = ( 1 – p )  *  Tij
           t = t + 1
           Para cada caminho(i, j) ΔT_ij = 0
           Inicializa Tabu
     Fim
```

Quadro I - Fluxo do Algoritmo Ant System.

Capítulo 4

Algoritmos Genéticos

Neste capítulo são descritos os fundamentos dos algoritmos genéticos, sua fonte de inspiração e particularidades. São detalhados os procedimentos do algoritmo, os principais operadores e as configurações necessárias para o emprego deles.

4.1 Introdução

Vista de forma global, a evolução natural implementa mecanismos adaptativos de otimização que, embora estejam longe de serem uma forma de busca aleatória, com certeza envolvem aleatoriedade. É este tipo de busca inteligente, mas não determinística que os algoritmos genéticos tentam imitar (TANOMARU, 1995).

Segundo MAZZUCCO (1999), muitas experiências comprovam que a seleção natural é um fato incontestável, podendo não ser o único mecanismo evolutivo, porém sem dúvida um dos fatores principais do processo. Uma das principais constatações dessa evolução foi alcançada através de uma experiência realizada com bactérias. As bactérias foram expostas a doses de penicilina cada vez maiores e somente as bactérias mais fortes sobreviviam e se reproduziam. Este processo se repetiu por cinco gerações e no final apresentou-se uma linhagem de bactérias resistente a uma dose de penicilina duas mil e quinhentas vezes maior do que a inicial. O importante aqui é o fato de que a resistência à penicilina não foi uma característica desenvolvida individualmente e sim herdada das gerações anteriores.

Outra constatação interessante foi observada em uma população de pombos. Os pombos viviam perto de uma mina de carvão e tinham como cor predominante, a cor preta (isto porque, com a mina de carvão, o ambiente onde os pombos viviam era sujo e com a cor preta os pombos conseguiam mais facilmente se camuflar e escapar de seus predadores naturais). Quando a mina de carvão fechou, com o passar dos tempos observou-se uma gradativa alteração na cor predominante daquela população. Os pombos passaram a ter cores mais claras. Como a mina havia fechado, o ambiente tornara-se mais limpo e com isso os pombos mais claros se escondiam mais facilmente de seus predadores.

O cientista inglês Charles Darwin estudando as espécies e suas evoluções, coletou durante anos um volumoso material que demonstrou principalmente, a existência de inúmeras variações em cada espécie (DARWIN, 1953). Seus estudos aliados as pesquisas de outros cientistas no assunto tornaram evidente que as espécies realmente se modificam.

Segundo DARWIN (1953), novas variedades são produzidas por meio da seleção natural. As variações mais favoráveis de cada espécie conseguem sobreviver com mais facilidade, uma vez que tem mais chance de se reproduzirem.

Embora DARWIN (1953), tenha formulado a Teoria da Evolução há bastante tempo, só recentemente se tentou idealizar um modelo matemático do processo evolutivo. Nos anos 60, John Holland da Universidade de Michigam, em suas explorações dos processos adaptativos de sistema naturais e suas possíveis aplicabilidades em projetos de softwares de sistemas artificiais no final da década de 1970, conseguiu incorporar importantíssimas características da evolução natural a um algoritmo (HOLLAND, 1993).

Segundo MAZZUCCO (1999), embora os mecanismos que conduzem a evolução natural não estejam ainda plenamente compreendidos, algumas de suas importantes características são conhecidas. A evolução dos seres vivos se processa nos cromossomos (dispositivos orgânicos onde as estruturas desses seres são codificadas). Parte da criação de um ser vivo é realizada através de um processo de codificação e decodificação de cromossomos. Muito embora, os processos específicos de codificação e decodificação de cromossomos ainda não estejam totalmente esclarecidos, existem

algumas características gerais na teoria desse assunto que estão plenamente consolidadas:

- A evolução é um processo que se realiza nos cromossomos e não nos seres que os mesmos codificam;
- A seleção natural é a ligação entre os cromossomos e o desempenho de suas estruturas decodificadas. Os processos da seleção natural determinam que aqueles cromossomos bem sucedidos devem se reproduzir mais freqüentemente do que os mal sucedidos;
- É no processo de reprodução que a evolução se realiza. Através de mutação (alteração aleatória nos genes) os cromossomos de um ser descendente podem ser diferentes dos cromossomos de seu gerador. Através do processo de recombinação (crossover) dos cromossomos de dois seres geradores, é também possível que os cromossomos do ser descendente se tornem muito diferentes daqueles de seus geradores;
- A evolução biológica não possui memória. A produção de um novo indivíduo depende apenas de uma combinação de genes da geração que o produz.

HOLLAND (1993), convenceu-se de que estas características da evolução biológica poderiam ser incorporadas a um algoritmo para constituir um método extremamente simples de resolução de problemas complexos, imitando o processo natural, ou seja, através da evolução.

Os algoritmos genéticos pertencem à classe dos métodos probabilísticos de busca e otimização, embora não sejam aleatórios. Usa-se o conceito de probabilidade, mas algoritmos genéticos estão longe de serem buscas aleatórias. Pelo contrário, algoritmos genéticos tentam dirigir a busca para regiões do espaço onde é provável que os pontos ótimos estejam (TANOMARU, 1995).

4.2 Descrição do Algoritmo Genético

Algoritmos genéticos são métodos computacionais de busca baseados nos mecanismos de evolução natural e na genética. Em algoritmos genéticos, uma população de possíveis soluções para o problema em questão evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua (TANOMARU, 1995).

O primeiro passo para aplicação do algoritmo genético a um problema qualquer é representar cada possível solução *x* no espaço de busca como uma seqüência de símbolos *s* gerados a partir de um alfabeto finito *A*. Nos casos mais simples usa-se o alfabeto binário A = {0,1}, mas no caso geral, tanto o método de representação quanto o alfabeto genético dependem de cada problema. Usando algumas das metáforas extremamente simplistas, mas empregadas pelos teóricos e praticantes de algoritmos genéticos com freqüência, cada seqüência *s* corresponde a um cromossomo, e cada elemento de *s* é equivalente a um gene. Como cada gene pode assumir qualquer valor do alfabeto A, cada elemento de A é equivalente a um alelo, ou seja, um valor possível para um dado gene. A posição de um gene num cromossomo, ou seja, o índice dentro da seqüência, corresponde a um *lócus gênico* (TANOMARU, 1995).

Na maior parte dos algoritmos genéticos assume-se que cada indivíduo seja constituído de um único cromossomo, razão pela qual é comum usarem-se os termos indivíduo e cromossomo indistintamente. A grande maioria dos algoritmos genéticos propostos na literatura usam uma população de numero fixo de indivíduos, com cromossomos também de tamanho constante (TANOMARU, 1995).

Tendo definido a representação cromossômica para o problema, gera-se um conjunto de possíveis soluções, chamadas de soluções candidatas. Um conjunto de soluções codificadas de acordo com a representação selecionada corresponde a uma população de indivíduos *P(0)*. Algoritmos genéticos são iterativos, e a cada iteração a população é modificada. Cada iteração de um algoritmo genético é denominada uma

geração, embora nem todos os indivíduos de uma população sejam necessariamente filhos de indivíduos da população anterior. Designando cada geração por um índice *t*, o fluxo geral de um algoritmo genético é demonstrado na tabela abaixo:

```
Procedure algoritmo genético
Inicio
        t = 0
        Inicializar P(t)
        Avaliar P(t)
        Enquanto (condicao) faca
        Inicio
                t = t + 1
                Selecionar P(t) a partir de P(t-1)
                Cruzamento P(t)
                Mutação P(t)
                Avaliar P(t)
        Fim
Fim
```

Quadro II - Fluxo básico de um algoritmo genético.

O algoritmo genético pertence a uma das classes dos chamados algoritmos evolucionários. Ele se constitui em um método de busca de propósito geral, baseado em um modelo de evolução biológica natural. O seu objetivo é explorar o espaço de busca na determinação de melhores soluções, permitindo os indivíduos da população evoluírem com o tempo. Cada passo do algoritmo no tempo, na escala evolutiva, é chamado de geração. Partindo de uma geração inicial, normalmente criada aleatoriamente, o algoritmo executa continuamente o seguinte laço principal (MAZZUCO, 1999):

1) Avalia cada cromossomo da população através do cálculo de sua aptidão, utilizando a função de avaliação;

2) Seleciona os indivíduos que produzirão descendentes para a próxima geração, em função dos seus desempenhos apurados no passo anterior;

3) Gera descendentes através da aplicação das operações de cruzamento e mutação sobre os cromossomos selecionados no passo anterior, criando a próxima geração;

4) Se o critério de parada for satisfeito, termina e retorna o melhor cromossomo até então gerado, senão volta ao passo 1.

4.2.1 Processo de Inicialização

A população inicial de indivíduos ou cromossomos é na maioria das vezes realizada de forma aleatória, embora existam ocasiões onde é mais apropriada uma seleção heurística da mesma, introduzindo logo de início, um ou mais indivíduos "interessantes", como por exemplo, soluções aproximadas conhecidas contendo algum tipo de informação prévia. Diversos trabalhos realizados comprovam que a inicialização, em geral, não é crítica, desde que a população inicial contenha cromossomos suficientemente variados (GOLDBERG, 1989).

Como no caso biológico, não há evolução sem variedade. Ou seja, a teoria da evolução natural ou "lei do mais forte" necessita que os indivíduos tenham diferentes graus de adaptação ao ambiente em que vivem. De acordo, é muito importante que a população inicial cubra a maior área possível do espaço de busca (TANOMARU, 1995).

4.2.2 Avaliação e Adequabilidade

Algoritmos genéticos necessitam da informação do valor de um função objetivo para cada membro da população. Nos casos mais simples, usa-se simplesmente o valor da função que se quer maximizar. A função de avaliação dá para cada indivíduo uma medida de quão adaptado ao ambiente ele está, ou seja, quanto maior o resultado da avaliação maior a chance do indivíduo sobreviver ao ambiente e reproduzir-se, passando parte de seu material genético para as futuras gerações. A avaliação de cada indivíduo resulta em um determinado valor de aptidão (que em inglês é denominado de "*fitness*") (TANOMARU, 1995).

O processo de avaliação consiste na aplicação de uma função de avaliação em cada um dos indivíduos da população corrente. Esta função deve expressar a qualidade de cada indivíduo da população no contexto do problema considerado e é extremamente importante, uma vez que constitui o único elo entre o algoritmo genético e o problema em questão. A função de avaliação depende fortemente da forma como as soluções foram representadas (MAZZUCCO, 1999).

A maior parte das aplicações, baseadas em algoritmos genéticos, utilizam representações indiretas das soluções, o que significa que o algoritmo trabalha sobre uma população de soluções codificadas. Desta forma, antes que seja realizada a avaliação sobre os cromossomos uma tradução da solução codificada para a solução real deve ser realizada. Por outro lado em uma aplicação que empregue representação direta da solução, o valor que é passado para a função de avaliação é o próprio cromossomo. Neste caso toda a informação relevante ao problema deve ser incluída na representação da solução (MAZZUCCO, 1999).

4.2.3 Seleção

O mecanismo de seleção em algoritmos genéticos emula os processos de reprodução assexuada e seleção natural. Em geral gera-se um população temporária de *N* indivíduos extraídos com probabilidade proporcional à adequabilidade relativa de cada indivíduo na população.

A seleção efetua com base nas aptidões individuais (calculadas através da função de avaliação), a seleção dos indivíduos que procriarão para formarem a próxima geração. Esta seleção é probabilística, assim, um indivíduo com aptidão alta tem maior probabilidade de se tornar um gerador do que um indivíduo com aptidão baixa. No caso de um algoritmo genético com a população fixa, são escolhidos tantos indivíduos quanto for o tamanho da população (MAZZUCCO, 1999).

O processo inicia-se com a conversão de cada aptidão individual em uma expectativa que é o número esperado de descendentes que cada indivíduo poderá gerar. No algoritmo original proposto por HOLLAND (1993), essa expectativa individual era calculada através da divisão da aptidão individual pela média das aptidões de toda a população da geração corrente. Desta forma, a expectativa *e* de um indivíduo *i*, pode ser calculada por:

$$e_i = apt_i / mapt_t \qquad (6)$$

Onde apt_i é a aptidão do indivíduo *i* e $mapt_t$ é a aptidão média da população na geração *t*.

As expectativas assim produzidas têm as seguintes características: um indivíduo com aptidão acima da média terá expectativa maior do que *1*, enquanto que um indivíduo com aptidão abaixo da média terá uma expectativa menor do que *1*; a soma dos valores de todas as expectativas individuais será igual ao tamanho da população.

No entanto, pelo menos dois problemas podem ser apontados na utilização desse método de cálculo das expectativas. Em primeiro lugar, o método não deve manipular valores de aptidão negativos, uma vez que poderiam ser convertidos em valores de expectativa negativos. Em segundo lugar, esse método pode apresentar efeitos

indesejáveis logo no início, quando a população é aleatoriamente calculada e a média das aptidões é baixa, onde indivíduos aptos podem receber um número desordenado de descendentes, conduzindo a uma convergência prematura. Também no decorrer do processamento, onde esse método tem uma forte tendência em alocar a todos os indivíduos um descendente, mesmo para aqueles indivíduos com expectativa muito baixa, resultando em uma exploração ineficiente do espaço de busca e, conseqüentemente, em um retardo acentuado na convergência do método (MAZZUCCO, 1999).

Segundo MAZZUCCO (1999), um segundo método de conversão de valores de aptidão em números esperados de descendentes, referenciado como o "método do truncamento sigma" considera a aptidão média da população e o desvio padrão das aptidões sobre a população. Sua idéia básica é muito simples: um indivíduo, cuja aptidão seja igual à média das aptidões da população terá uma expectativa de produzir um descendente; um indivíduo, cuja aptidão seja um desvio padrão acima da média, terá uma expectativa de produzir um descendente e meio; um indivíduo, cuja aptidão seja dois desvios padrões acima da média, terá uma expectativa de produzir dois descendentes, e assim por diante.

Baseado no método do truncamento sigma, a expectativa *e* de um indivíduo *i* pode ser calculada por:

$$e_i = (apt_i - mapt_t) / 2dp_t + 1 \qquad (7)$$

No caso onde $dp_t \neq 0$ ou:

$$e_i = 1 \qquad (8)$$

No caso onde $dp_t = 0$, onde apt_i é a aptidão do indivíduo *i* e $mapt_t$, e dp_t são, respectivamente, a média e o desvio padrão das aptidões na geração *t*.

Caso o desvio padrão seja zero, qualquer indivíduo na população terá o mesmo valor de aptidão. Neste caso, automaticamente esse método irá atribuir um valor de expectativa igual a *1* para todos os indivíduos da população. Se o valor da expectativa tornar-se negativo, ele será alterado para um valor pequeno como *0.1*, por exemplo, de forma que aqueles indivíduos com aptidões muito baixas terão pequenas probabilidades

de se reproduzirem. Nota-se, que com esse segundo método, a soma dos números esperados de descendentes para todos os indivíduos na população não necessariamente será igual ao tamanho da população.

Depois de calculada a expectativa de cada indivíduo, resta ainda um último cálculo, onde o valor da expectativa é transformado em número definitivo de descendentes que cada indivíduo irá gerar. Esta última operação se faz necessária, uma vez que o valor da expectativa é um número real e não pode ser utilizado como valor real de descendentes, pois suponha que um indivíduo receba a expectativa *1,5*, um indivíduo não pode gerar um descendente e meio.

Segundo GOLDBERG (1989) e MIRANDA (2002), o número inteiro e definitivo de descendentes que cada indivíduo receberá, chamado de "valor de descendente individual", pode ser obtido através de uma técnica denominada de "roleta". Para visualizar este método considere um círculo dividido em *n* regiões (com *n* sendo o tamanho da população). A área de cada região é proporcional à expectativa de cada indivíduo. Coloca-se sobre este círculo uma "roleta" com *n* cursores, igualmente espaçados. Após um giro da roleta, a posição dos cursores indica os indivíduos selecionados. Evidentemente, os indivíduos cujas regiões possuem maior área terão

maior probabilidade de serem selecionados mais vezes. Como conseqüência, a seleção de indivíduos pode conter várias cópias de um mesmo indivíduo enquanto outros podem desaparecer. Aqui os indivíduos mais bem adaptados (com maiores aptidões) têm mais chances de serem selecionados do que os indivíduos mais fracos (com aptidões menores).

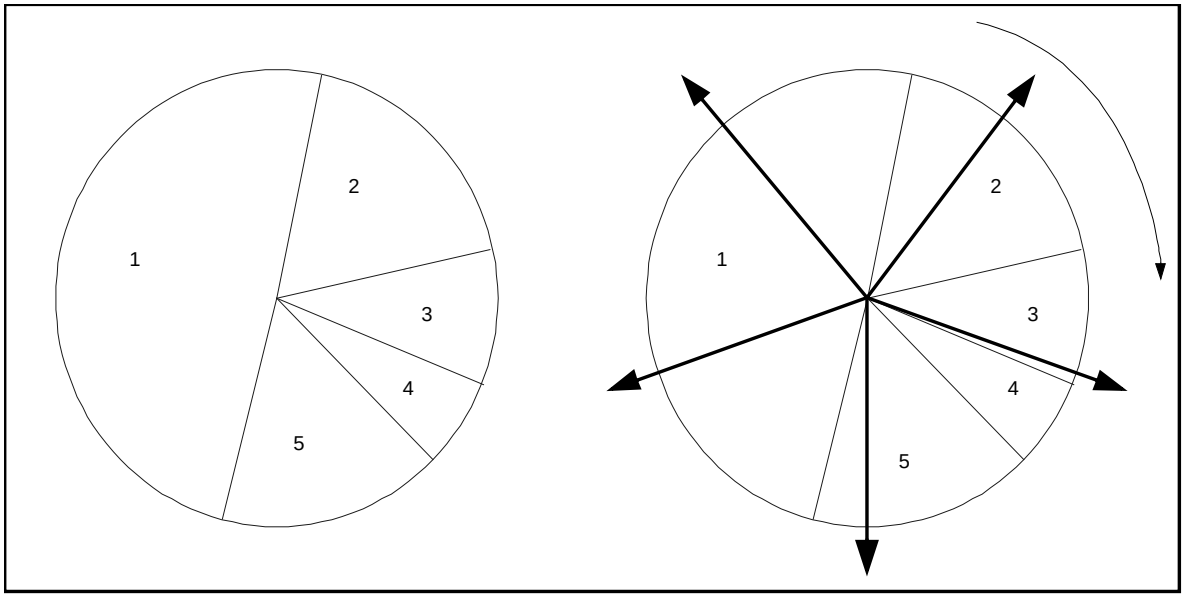

Figura 6 - Método de seleção denominado de “roleta”.

A Fig. 6 demonstra o método da roleta aplicado a uma população com 5 cromossomos. Nesta figura, o cromossomo *1* tem sua aptidão muito maior que os demais cromossomos e, após o giro da roleta, o mesmo recebe dois descendentes. Já o cromossomo *4* tem sua aptidão muito inferior aos demais e após o giro da roleta ele não teve nenhum cursor apontando para sua área. Ele não irá gerar nenhum descendente e, conseqüentemente, será extinto.

Encerrando o segundo passo do algoritmo genético, o qual descreve o processo de seleção dos cromossomos que produzirão a próxima geração, é importante salientar que apesar de sua aparência confusa, este processo, de fácil implementação, vem tornar o algoritmo genético ainda mais robusto, uma vez que generaliza a função de avaliação, permitindo que a mesma retorne valores bem definidos (MAZZUCCO, 1999).

KOZA & RICE, 1994, ressaltam que o processo de seleção utilizado pelo algoritmo genético é probabilístico e é um dos pontos essenciais do algoritmo. Como já aludido, o processo aloca a cada indivíduo uma chance de ser selecionado (não importando o quanto a aptidão desse indivíduo seja pobre) e participar no processo dos operadores genéticos.

4.2.4 Operadores Genéticos

O princípio básico dos operadores genéticos é transformar a população através de sucessivas gerações, para obter um resultado satisfatório no final do processo. Deste modo, eles são extremamente necessários para que a população se diversifique e mantenha as características de adaptação adquiridas pelas gerações anteriores.

Existem inúmeros operadores para a manipulação dos indivíduos, sendo os mais comuns, considerados clássicos, o cruzamento (*crossover*) e a mutação (*mutation*).

4.2.4.1 Cruzamento (*Crossover*)

O processo de cruzamento é geralmente controlado por um parâmetro fixo que indica a probabilidade de um gene sofrer cruzamento (normalmente se utiliza uma taxa de cruzamento moderada). Este operador possui o efeito de explorar as informações contidas em seus geradores, ou seja, tendem a gerar melhores indivíduos com o passar das gerações baseados nas gerações anteriores (TANOMARU, 1995).

Ao cruzar pelo menos dois pais, uma ou mais novas soluções são criadas intercambiando-se a informação genética dos genitores em um ou mais pontos que também são selecionados aleatoriamente.

O operador de cruzamento pode variar conforme a representação utilizada. Segundo GOLDBERG (1989), quando a representação é binária, os dois tipos mais utilizados são: o cruzamento de um ponto e o cruzamento multiponto.

Métodos de Cruzamento

Cruzamento de um ponto: É realizado através da escolha aleatória de um só ponto de corte. Cada par de cromossomos escolhidos como pais gera dois descendentes através da permuta de suas partes finais, depois do ponto de corte.

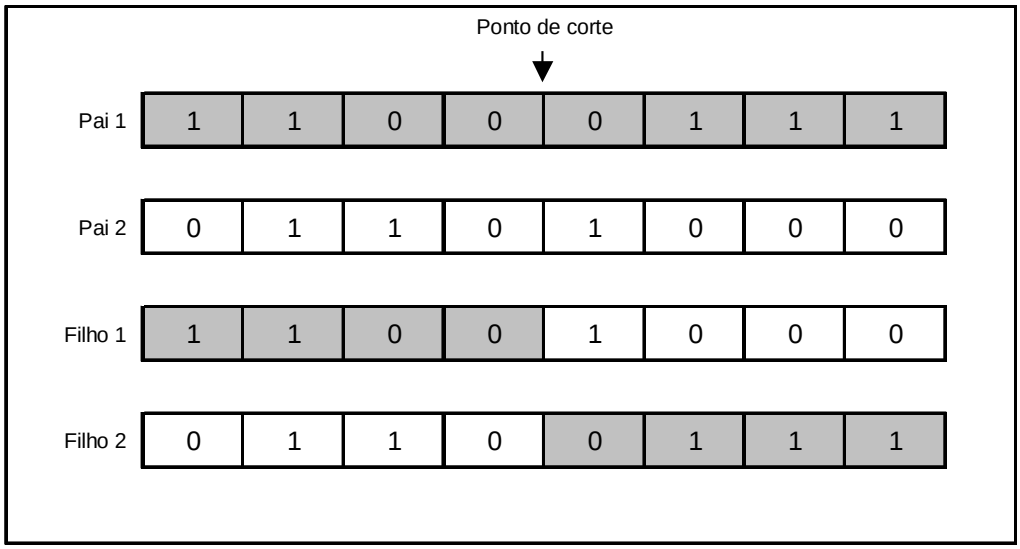

Figura 7 - Exemplo de cruzamento de um ponto.

Um dos problemas deste tipo de cruzamento é que os descendentes sempre irão conter as partes finais dos seus pais, não podendo fazer outra combinação possível, caracterizando uma tendência de sobrevivência das partes finais (SPILLMAN, 1993).

Cruzamento multiponto: muito parecido com o operador de um ponto, o cruzamento multiponto busca melhorar o cruzamento de um ponto, sendo escolhidos mais pontos de corte para troca de material genético entre o par de geradores. A figura abaixo mostra um exemplo de cruzamento multiponto com 2 pontos de corte.

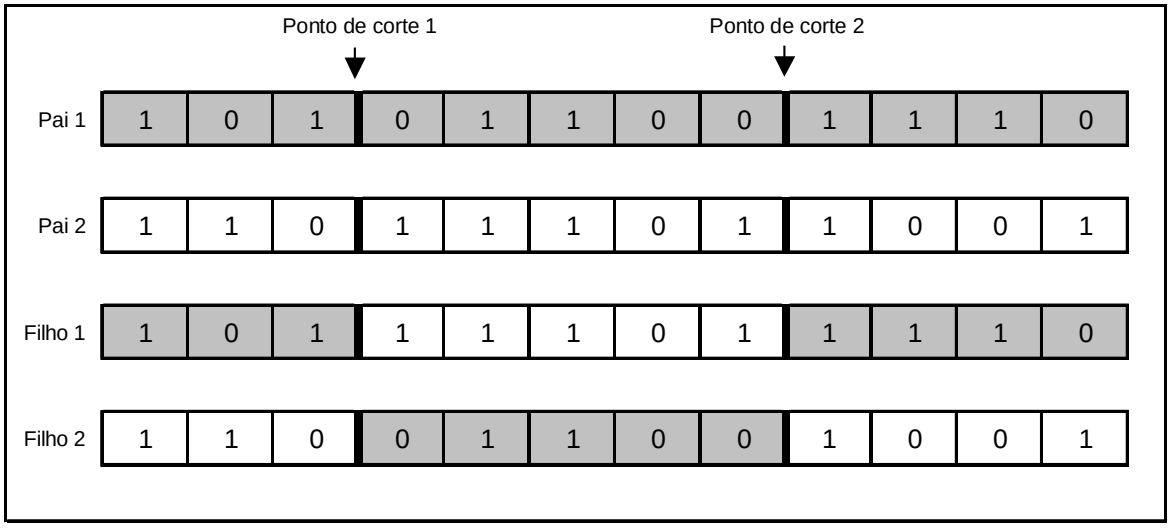

Figura 8 - Exemplo de cruzamento multiponto.

Existem representações onde os operadores de cruzamentos mencionados não podem ser utilizados, como no caso da representação em forma de permutação. Uma das maiores dificuldades destas representações é que o operador de cruzamento pode produzir ciclos inviáveis, como no exemplo da Fig. 9. Esta figura traz um exemplo de cruzamento aplicado a cromossomos com representação através de números inteiros, onde, após aplicarmos os cruzamentos de um ou mais pontos de corte o operador gera dois cromossomos inválidos, pois os mesmos possuem alelos repetidos, o que não é permitido (no caso do problema do caixeiro viajante seria o mesmo que dizer que o viajante não percorreu todas as cidades uma, e somente uma, vez como deveria) (RAULINO, 2001).

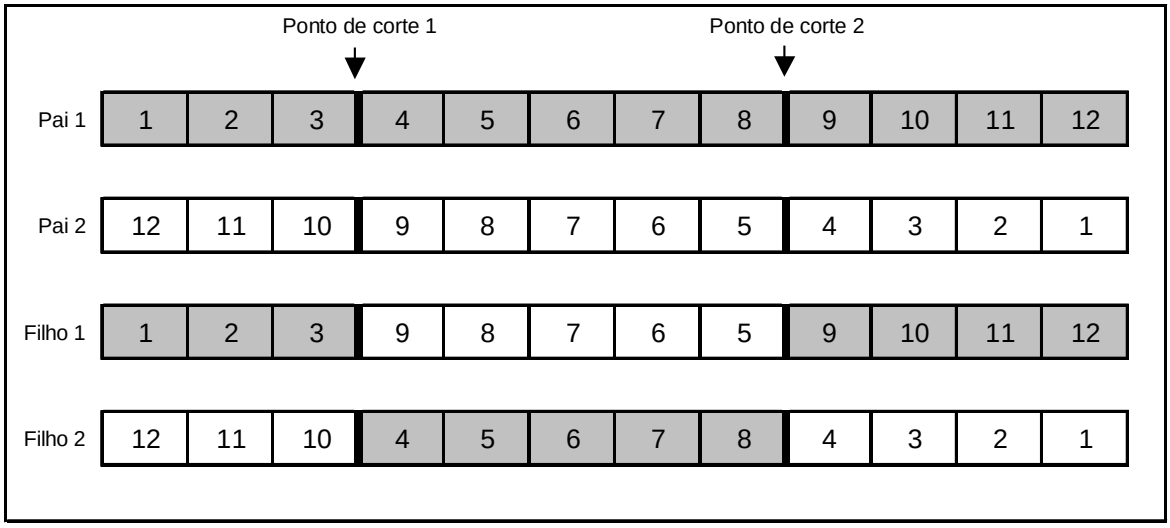

Figura 9 - Exemplo de cruzamento inválido.

Para solucionar estes problemas, foram desenvolvidos operadores especiais que evitam a produção de filhos inviáveis (geneticamente abortivos), dado que os pais sejam representações de soluções viáveis. Segundo GOLDBERG, 1989, dentre estes operadores, os que mais se destacam, devido à quantidade de aplicações em que são empregados são: o operador OX, o operador PMX e o operador CX.

Operador OX (*Order Crossover*): este operador começa pela escolha aleatória de dois pontos de corte em cada um dos elementos selecionados. A seção definida entre estes dois pontos é copiada integralmente no descendente. Os lugares restantes são preenchidos usando as informações não repetidas na seção de cruzamento, começando do segundo ponto de corte. A Fig. 10 traz um exemplo de cruzamento OX, onde a seqüência j, k, c (volta ao inicio) d, e, f, g, h, i, é o gene contido no segundo pai, começando no segundo ponto de corte. De maneira similar, obtém-se a seqüência j, k, c, e, f, d, b, i, a, do segundo filho.

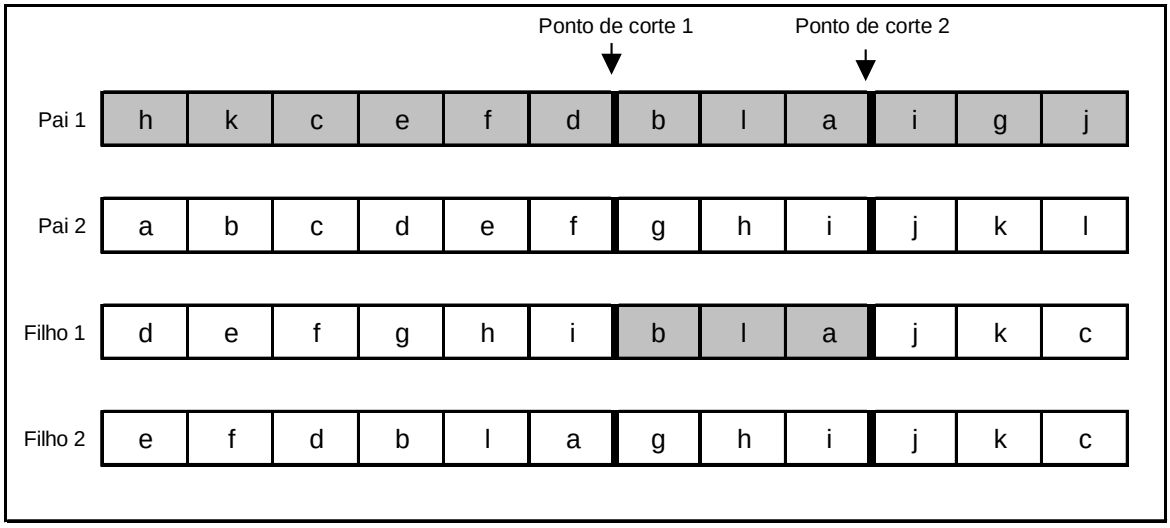

Figura 10 - Exemplo de cruzamento com o operador OX.

Operador PMX (*Partially Mapped Crossover*): este operador também é executado escolhendo-se aleatoriamente dois pontos de corte. Este processo é ilustrado na Fig. 11, onde as informações contidas entre os dois pontos de corte, (b, i, a) e (g, h, i) são intercambiadas, obtendo-se as representações intermediárias. Porém, estas representações não são válidas, pois possuem informações repetidas e algumas faltando. Assim, o passo final é substituir estes genes repetidos mapeando (g, h, i) para (b, i, a) e vice-versa, obtendo assim as estruturas finais.

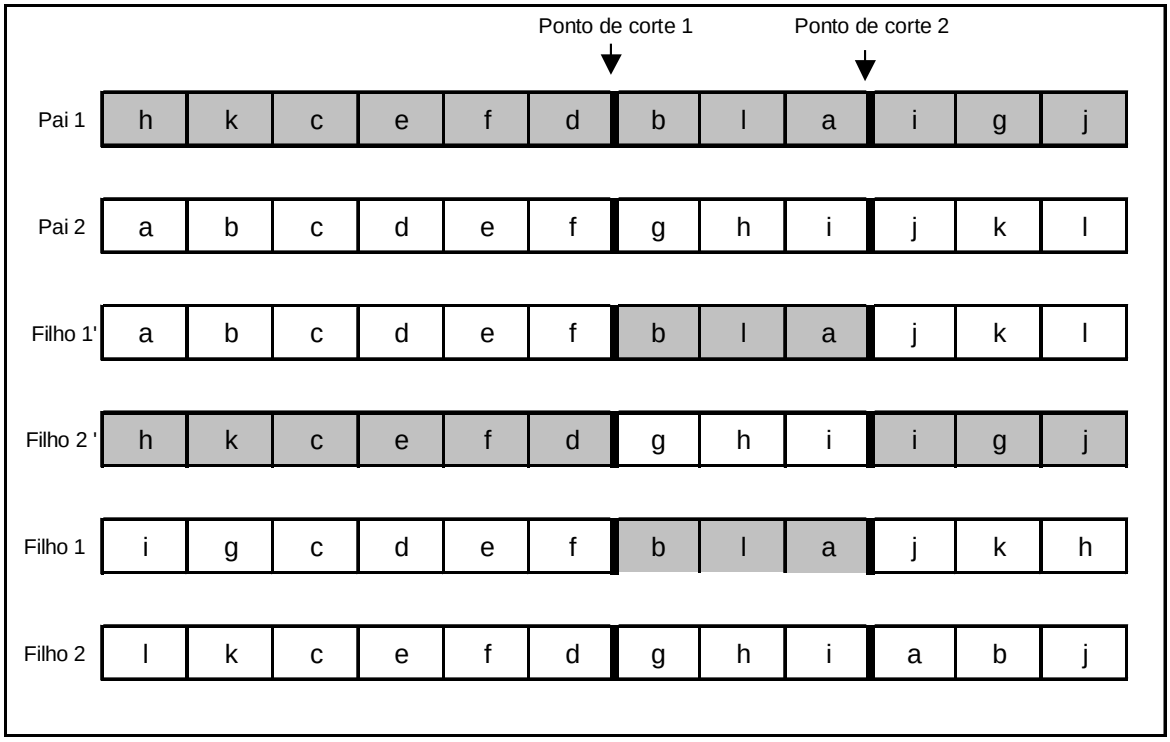

Figura 11 - Exemplo de cruzamento com o operador PMX.

Operador CX (*Cycle Crossover*) o esquema do operador CX é diferente dos demais mostrados anteriormente. O operador CX executa recombinações de forma que cada um dos genes de seus descendentes venham da posição correspondente de qualquer um dos pais (neste tipo de cruzamento não há necessidade de escolher pontos de cruzamento, como os usados no operador PMX ou no operador de ordem OX).

Originalmente, este procedimento é iniciado da primeira posição mais à esquerda. Na Fig. 12 temos um exemplo de cruzamento CX onde o procedimento é iniciado a partir da primeira posição mais à esquerda, escolhendo um gene do primeiro pai. Já que por este operador cada gene vem da mesma localização de um dos pais, a escolha da atividade 2 do pai 1 significa que a atividade 5 da segunda posição deve ser escolhida, porque 2 ocupa a segunda posição do segundo pai. Esta seleção, por sua vez, implica que a atividade 4 seja escolhida do pai 1, que por sua vez leva a escolher a atividade 3 do pai 1, porque 3 está na sétima posição do pai 2. Neste ponto, se continuar com o processo, a escolha da atividade 3 significa ter que selecionar a atividade 2 do pai um. Entretanto, isto não é possível porque 2 já foi selecionado como o segundo gene da solução. A posição destes genes é dita de formar um ciclo que, eventualmente, retorna

ao primeiro gene selecionado. Então o procedimento para e os espaços vazios são preenchidos com os genes do segundo pai. O segundo filho é obtido realizando o cruzamento complementar.

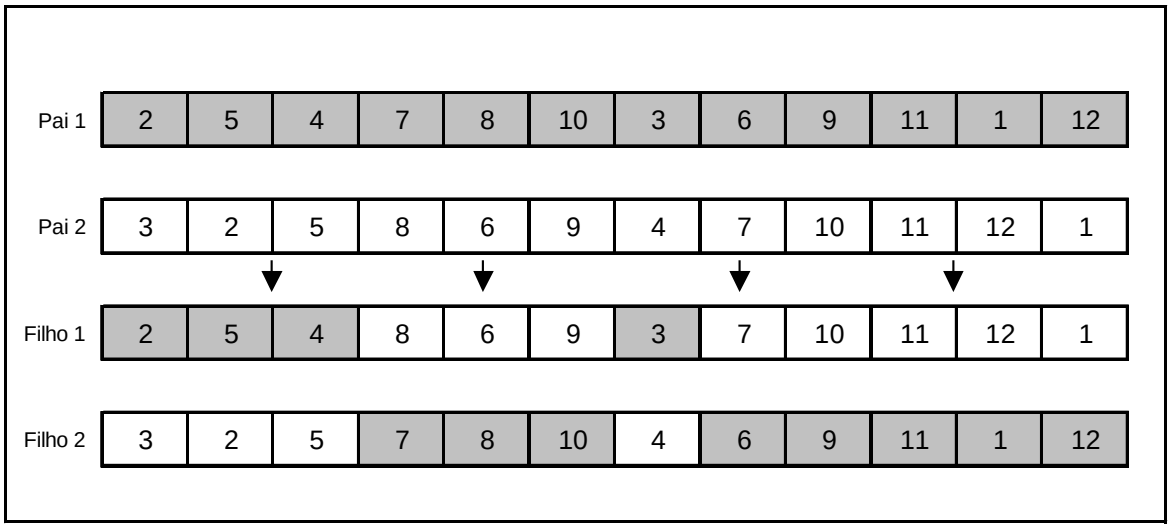

Figura 12 - Exemplo de cruzamento com operador CX.

4.2.4.2 Mutação

O operador de mutação é bastante simples, basicamente, seleciona-se uma posição num cromossomo e muda-se o valor do gene correspondente aleatoriamente para outro alelo possível. A mutação é aplicada à população de cromossomos descendentes gerados após o cruzamento.

O operador de mutação é necessário para introdução e manutenção da diversidade genética na população, alterando arbitrariamente um ou mais componentes de uma estrutura (cromossomo) escolhida, fornecendo assim meios para a introdução de novos elementos na população. Desta forma, a mutação assegura que a probabilidade de alcançar qualquer ponto no espaço de busca nunca será completamente eliminada, além de contornar o problema de se cair em mínimos locais, pois a direção de busca do algoritmo genético é alterada sutilmente com a ocorrência da mutação (CHAMBERS, 1995).

O processo é geralmente controlado por um parâmetro fixo que indica a probabilidade de um gene sofrer mutação (normalmente se utiliza uma taxa de mutação pequena). Este operador possui o efeito de aumentar a diversidade da população evitando que as estruturas tornem-se muito homogêneas. O aumento na diversidade das estruturas permite reduzir a possibilidade de convergência prematura, isto é, a obtenção de soluções sub-ótimas (TANOMARU, 1995).

Apesar das vantagens da utilização do operador de mutação, ele deve ser aplicado em pequena escala no processo do AG, semelhante à forma como a mutação ocorre na genética natural, pois a ocorrência excessiva de mutações pode gerar pontos muito dispersos na população, ao invés de convergir para pontos ótimos (KOZA, 1992)

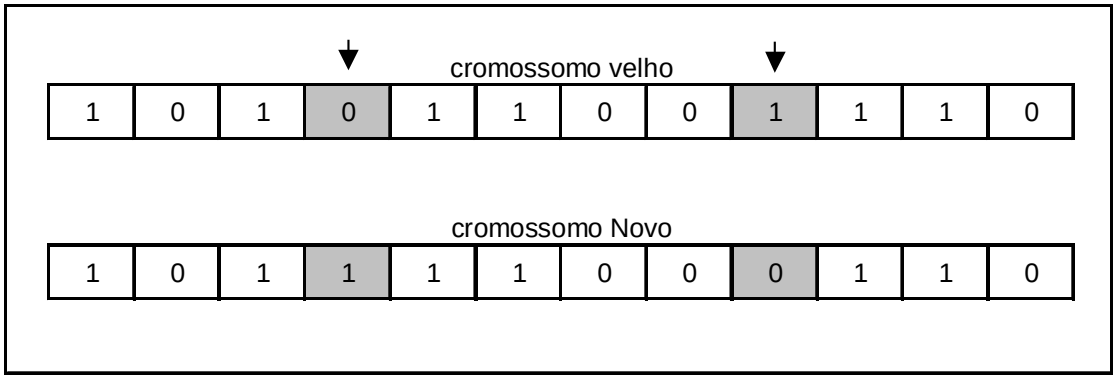

Figura 13 - Exemplo de mutação.

Os algoritmos genéticos, geração após geração, tentam encontrar indivíduos com melhor desempenho que os anteriores. Mas, problemas intrínsecos do algoritmo podem dificultar a procura ou mesmo conduzir a pesquisa a uma direção errada. Nesses casos provavelmente, ocorre uma falha quantitativa ou qualitativa na ação dos operadores. Falhas quantitativas podem ser definidas como o tamanho da população, o comprimento do cromossomo e as probabilidades de cruzamento e mutação. E falhas qualitativas são os métodos de cruzamento, as estratégias de seleção, entre outros.

4.2.5 Condições de Término

O ideal para todo algoritmo de otimização seria que o processo terminasse assim que o ponto ótimo fosse descoberto. Na prática, entretanto, não se pode afirmar com certeza que o ponto ótimo encontrado pelo algoritmo seja o ponto “ótimo-global”. Como conseqüência disso, geralmente o critério para o término utilizado é um número máximo de gerações ou um tempo limite de processamento, “o que ocorrer primeiro” (KOZA & RICE, 1994 e TANOMARU, 1995).

Outro critério que também é empregado é a idéia de “estagnação”, onde o algoritmo genético termina quando nenhuma melhoria for encontrada na população depois de várias gerações consecutivas (TANOMARU, 1995 e MICHALEWICS, 1999).

Capítulo 5

Modelo Proposto

Neste capítulo é descrito o modelo proposto, uma abordagem que combina o algoritmo das formigas e algoritmo genético para a otimização de problemas combinatoriais. Inicialmente são apresentadas as definições do problema, do método existente e, por fim o modelo proposto.

5.1 Introdução

A proposta inicial deste trabalho é a criação de uma nova abordagem com o intuito de buscar soluções de boa qualidade para problemas de otimização. O trabalho é baseado no algoritmo das formigas explicado anteriormente.

Na tentativa de melhorar o desempenho do algoritmo original, desenvolveu-se um novo modelo híbrido, onde foram combinados, o algoritmo das formigas e o algoritmo genético.

Uma operação chamada *crossover* (cruzamento) passou a ser realizada sobre algumas soluções selecionadas a partir do algoritmo das formigas, resultando de novas soluções, as quais foram incorporadas ao conjunto. O processo de avaliação implementado pelo algoritmo das formigas, a partir daí, compara essas soluções, que podem, ou não, serem aceitas em função da qualidade da solução encontrada.

O resultado final constituiu um método híbrido tendo como base o algoritmo das formigas e como complementação o algoritmo genético. A forma como estes algoritmos foram combinados é descrita a seguir.

5.2 Descrição Formal do Problema

O problema do caixeiro viajante pode ser definido da seguinte forma: um viajante precisa visitar um conjunto de cidades, somente uma de cada vez. O viajante pode iniciar em qualquer cidade, retornando a cidade inicial, percorrendo a menor distância possível.

A modelagem do problema do caixeiro viajante pode ser feita pela análise combinatória ou pela teoria de grafos, na qual a solução consiste em encontrar um circuito hamiltoniano, ou seja, um circuito que visite todas as cidades, somente uma vez, obtendo o menor custo. Podemos definir custo (*i*,*j*) como o valor da aresta entre os vértices *i* e *j*.

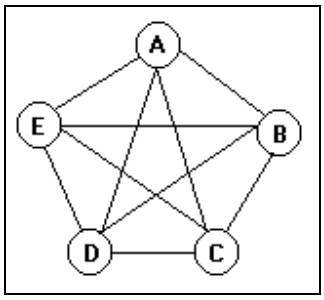

Figura 14 - Circuito Hamiltoniano.

Desta forma o problema do caixeiro viajante pode ser representado por um grafo *G* onde os vértices são as n cidades a visitar e uma matriz *D = [dij]* de distâncias entre cada cidade *(i, j)*.

Matematicamente, o problema do caixeiro viajante pode ser formulado como uma matriz de custo *C = [dij]*, onde *dij* representa o custo de ir da cidade *i* para a cidade *j* e

encontrar uma permutação $\pi(i1, i2, i3,...,in)$ de inteiros, de 1 ate n que minimize a quantidade obtida pela soma. Onde o custo C = $(Ci1,i2+Ci2,i3+...+Cin,i1)$.

5.3 Modelagem

Inicialmente é definida a forma de representação das soluções do problema. Uma representação natural do espaço de soluções para o problema em questão é, portanto, aquela que contém o conjunto de todas as permutações possíveis. Define-se, assim, uma solução I para o problema, como sendo um elemento do conjunto de todas as permutações, que é uma possível rota.

$$I = (i_1, i_2, \dots , i_n) \qquad (9)$$

Onde cada elemento i do conjunto deve ser interpretado como as cidades naquela solução I e a ordem na qual as n cidades devem ser visitadas.

Dessa forma, utilizando esse modelo de representação, o problema passa a ter seu espaço de busca como sendo o conjunto de todas as permutações possíveis, isto é, determina-se, para cada permutação, o custo da rota correspondente. A melhor solução será aquela que apresentar a rota de menor custo.

O processo de avaliação da função de custo para uma rota do problema pode considerar inúmeras variáveis, como, por exemplo, a distância, o tempo, o custo, entre outros. Neste trabalho, a função de custo considera apenas o tamanho do percurso e pode ser definida da seguinte forma.

$$f = c_{i1,i2} + c_{i2,i3} + \dots + c_{in,i1} \qquad (10)$$

Onde c significa o custo para ir de uma cidade a outra e $i_1, i_2,...,i_n$ são as cidades na ordem que as mesmas devem ser visitadas.

5.4 O Modelo Existente

O comportamento natural das formigas inspirou a construção de um algoritmo no qual um conjunto de formigas artificiais cooperam para a solução de um problema através da troca de informação via o feromônio depositado sobre as trilhas. Esse algoritmo tem sido aplicado em problemas de otimização combinatória, como o problema do caixeiro viajante (DORIGO & GAMBARELLA, 1997).

A analogia adotada advém do fato das formigas conseguirem restabelecer rotas interrompidas por obstáculos, de maneira que a nova rota é a mais curta possível. Nesse algoritmo, formigas "virtuais" são enviadas para várias cidades e, após cada interação, a quantidade de feromônio em cada trilha é atualizada proporcionalmente à distância percorrida pelas formigas, selecionando o percurso ótimo entre as cidades (DORIGO & GAMBARELLA, 1996).

O modelo existente, utiliza como base o primeiro modelo proposto por COLORNI et al (1991) e DORIGO (1992), inspirado nos estudos do comportamento das formigas em DENEUBOURG et al *(1983)*. O algoritmo utilizado como base para este trabalho foi implementado por EYCKELHOF (2001).

O fluxo do algoritmo é muito simples como explicado no capítulo 3 e pode ser visto no quadro abaixo:

```
Procedure  Ant System
Inicio
    t = 0
    Para todos os caminhos(i, j)
          ΔT_ij = 0
    Posicione as m formigas nas n cidades
    s = 1
    Para k := 1 to m faca
          tabu_k[ s ] = posiçãoinicial[ k ]
    Repita ate que tabulist esteja completa
    Inicio
```

```
            s = s + 1
            Para k := 1 to m do
            Inicio
                  P_ij = [( T_ij )^α * ( N_ij )^β ] / ( Σ_ij ( T_ij )^α * ( N_ij )^β )
                  move a formiga k com a probabilidde Pij
                  Insere a cidade j em tabu_k[ s ]
            Fim
            Para k:= 1 to m do
            Inicio
                  Move a formiga k de tabu_k[ n ] para tabu_k[ 1 ]
                  Compute o tamanho L_k descoberto pela formiga k
                  Atualiza o valor do melhor tour encontrado
            Fim
            Para cada caminho(i, j) faca
                  Se caminho(i, j) ∈ tabu
                     ΔTij = ( 1 – p )  *  Tij  + 1 / L

                  Senao
                     ΔTij = ( 1 – p )  *  Tij
            t = t + 1
            Para cada caminho(i, j) ΔT_ij = 0
            Inicializa Tabu
      Fim
Fim
```

Quadro III - Fluxo básico do algoritmo Ant System.

5.5 O Modelo Proposto

O modelo proposto se caracteriza pela introdução de um novo operador no processo do algoritmo das formigas. Este operador é baseado no algoritmo genético e é aplicado nas soluções geradas pelo algoritmo das formigas. O fluxo do novo algoritmo é mostrado no quadro abaixo:

```
Procedure  ASGA
Inicio
     t = 0
     Para todos os caminhos(i, j)
           ΔTij = 0
     Posicione as m formigas nas n cidades
     s = 1
     Para k := 1 to m faca
           tabuk[ s ] = posiçãoinicial[ k ]
     Repita ate que tabulist esteja completa
     Inicio
           s = s + 1
           Para k := 1 to m do
           Inicio
                 Pij = [( Tij )a * ( Nij )b ] / ( Sij ( Tij )a * ( Nij )b )
                 move a formiga k com a probabilidde Pij
                 Insere a cidade j em tabuk[ s ]
           Fim
           Para k:= 1 to m do
           Inicio
                 Move a formiga k de tabuk[ n ] para tabuk[ 1 ]
                 Compute o tamanho Lk descoberto pela formiga k
```

```
                Atualiza o valor do melhor tour encontrado
          Fim
          Para n:= 1 to qdeCruzamentos faca
                Crossover(p1, p2)
          Para k:= 1 to m do
          Inicio
                Compute o tamanho Lk descoberto pela formiga k
                Atualiza o valor do melhor tour encontrado
          Fim
          Para cada caminho(i, j) faca
                Se caminho(i, j) ∈ tabu
                   ΔTij = ( 1 – p )  *  Tij  + 1 / L
                Senao
                   ΔTij = ( 1 – p )  *  Tij
          t = t + 1
          Para cada caminho(i, j) ΔTij = 0
          Inicializa Tabu
    Fim
Fim
```

Quadro IV - Fluxo do Algoritmo Híbrido.

O fluxo do algoritmo é semelhante ao do algoritmo original sendo introduzido um operador de cruzamento chamado *crossover*. Essa operação é realizada sobre algumas soluções selecionadas a partir do algoritmo das formigas resultando de novas soluções as quais são incorporadas ao conjunto. A avaliação da qualidade e viabilidade da nova solução permanece sendo realizada pelo algoritmo das formigas, a partir daí, compara essas soluções, que podem, ou não, serem aceitas em função da qualidade da solução encontrada.

5.5.1 O Operador Genético

A implementação da operação de cruzamento e baseada no operador OX (*Order Crossover*). Este operador começa pela escolha aleatória de dois pontos de corte em cada um dos elementos selecionados. A seção definida entre estes dois pontos é copiada integralmente no descendente. Os lugares restantes são preenchidos usando as informações não repetidas na seção de cruzamento, começando do segundo ponto de corte.

O quaodro abaixo mostra o procedimento utilizado para realizar esta tarefa sobre as soluções selecionadas:

```
Procedure Crossover(var p1,p2: TTour);
var
   Posicao,Indice,Indice1, Indice2, Aux, aux1 : Integer;
   filho: Ttour;
   Achou: Boolean;
   filho1, filho2: array[1..qdeMaxCidades] of Ttour;
begin
    buscaindices;
    Aux := 0; aux1:=0;
    For Aux :=  Indice1 to Indice2 do
    begin
         Inc(Aux1);
         Filho.pos[Aux1] := P1.pos[aux];

    end;
    Indice  := Aux1 + 1;
    Posicao := 1;
    Repeat
          Filho.pos[Indice] := P2.pos[Posicao];
          Achou := False;
          For Aux := Indice1 to Indice2 do
          begin
```

```
                if P1.pos[Aux] =  Filho.pos[Indice] then
                begin
                     Achou := True;
                     Break;
                end;
          end;
          Inc(Posicao);
          if Posicao > QdeCidades then
             Posicao := 1;
          if Not(Achou) then
             Inc(Indice);
    until Indice > QdeCidades;
    for aux:=1 to qdeformigas do
       P1.pos[aux]:=filho.pos[aux];
end;
```

Quadro V - Procedimento Crossover.

5.6 Implementação

A implementação do algoritmo foi feita utilizando a linguagem de programação Delphi versão 7.0, em um micro computador PC compatível com processador AMD Athlon, com *clock* de 2.0 Ghz, 512 Mb de memória RAM e sistema operacional Windows XP.

Capítulo 6

Análise dos Resultados Obtidos

Neste capítulo é descrita a metodologia de avaliação do modelo proposto, as formas de se obter os resultados e, os resultados computacionais obtidos com a aplicação do método proposto são comparados com resultados obtidos pelo algoritmo original.

6.1 Metodologia Utilizada

Neste trabalho, o método proposto é avaliado através de um protótipo baseado no algoritmo das formigas original, onde foram combinados o algoritmo das formigas e um novo operador de cruzamento. A avaliação é realizada comparando o desempenho do modelo híbrido em relação ao algoritmo original, aplicando os dois algoritmos às mesmas instâncias de problemas.

As instâncias utilizadas para testes estão divididas em duas categorias. Na primeira categoria são utilizadas instâncias em grade que tem como principal característica a possibilidade de se calcular o custo ótimo, independente da quantidade de cidades envolvidas. E na segunda categoria são utilizados *benchmarks* com dados obtidos de situações reais.

Os dados, para análise dos resultados, foram obtidos através da execução de ambos os algoritmos, selecionando os resultados variando o número de execuções e respeitando os critérios com relação às condições de término do algoritmo.

6.2 Origem das Instâncias

As diversas instâncias do problema do caixeiro viajante que são encontradas têm suas origens baseadas nas mais diversas situações, que tanto podem representar situações do mundo real como imaginárias. Dentre estes problemas, os que mais têm causado interesse, principalmente em função de sua aplicabilidade, em situações práticas, é o que trata de instâncias euclidianas, ou seja, as cidades correspondem a pontos no plano e as distâncias são calculadas na métrica euclidiana (ARAUJO, 2001).

As instâncias utilizadas para testes neste trabalho, foram retiradas de duas fontes. A primeira categoria pertence às instâncias em grade e a segunda pode ser encontrada em REINELT (2002).

A primeira fonte pertence às instâncias em grade. Este tipo de instância tem uma importante característica que motiva sua utilização, elas possibilitam a obtenção do custo ótimo. Nas instâncias em grades, as cidades são colocadas nos vértices de uma grade regular quadrada no espaço euclidiano (HANSEN, 1995).

A segunda fonte contém uma biblioteca de amostras de instâncias para o problema do caixeiro viajante, e problemas relacionados, onde podem ser encontrados os mais recentes resultados obtidos em pesquisas sobre o assunto, contendo várias instâncias com os mais variados números de cidades.

A figura abaixo mostra uma grade com 16 cidades. A distância entre duas cidades é determinada pelo comprimento do lado do quadrado ou pela distância da diagonal.

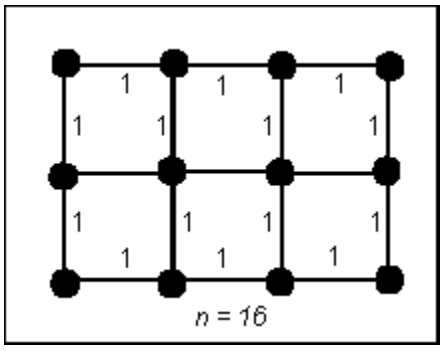

Figura 15 - Exemplo de instância em grade 4X4.

Segundo ARAUJO (2001), com a utilização de instâncias em grade, a obtenção do custo ótimo se torna bastante simples, bastando calcular a soma das distâncias. O custo ótimo é obtido da seguinte forma:

Quando o número de cidades for par:

$$f_{opt} = gn \qquad (11)$$

Quando o número de cidades for ímpar:

$$f_{opt} = g(n-1+\sqrt{2}) \qquad (12)$$

Para avaliar os resultados é escolhida a melhor execução, dentre todas as execuções, para uma determinada instância. A qualidade da solução obtida é medida em termos de porcentagem em relação à melhor solução conhecida. ARAUJO (2001), sugere o cálculo da qualidade baseado na fórmula:

$$Q = ((f(.) - f_{opt}) \ / \ f_{opt}) \ . \ 100 \qquad (13)$$

Onde $f(.)$ é o custo encontrado e f_{opt} o custo da melhor solução conhecida para aquela instância.

6.3 Resultados Obtidos

O modelo implementado não busca atingir valores recordes em quantidade de cidades, nem mesmo comparar o tempo de processamento com outros métodos existentes, pois como visto, este modelo tem como principal objetivo a investigação das potencialidades de um modelo híbrido combinando dois métodos já comprovadamente viáveis. A avaliação portanto, foi realizada sobre a qualidade e o desempenho do modelo proposto em relação ao método original.

Os resultados são demonstrados através de gráficos, que comparam os resultados obtidos no decorrer do tempo (gerações).

A tabela abaixo mostra o problema com o número de cidades e o custo ótimo conhecido para o problema. As colunas 3, 4, 5 e 6 mostram a média de gerações necessárias para se encontrar o custo ótimo e o tempo de processamento gasto para a obtenção dos resultados das instâncias testadas.

Problema	Custo Òtimo	Algorítmo Original		Algorítmo Híbrido	
		Média Gerações	Tempo (mm:ss)	Média Gerações	Tempo (mm:ss)
(4x4) 16 Cidades	16	12	0:01:10	9	0:01:03
(5x5) 25 Cidades	25,4	18	0:02:05	15,5	0:01:56
(6x6) 36 Cidades	36	30	0:03:57	27	0:03:11
(8x8) 64 Cidades	64	88	0:08:32	72	0:07:56
(10x10) 100 Cidades	100	134	0:14:49	123	0:13:38

Tabela 2 - Comparação do desempenho dos algoritmos original e híbrido.

Os gráficos abaixo mostram o desempenho do algoritmo original e do algoritmo híbrido, utilizando instâncias em grade com 16, 25, 36, 64 e 100 cidades. Os testes foram realizados variando o número de gerações necessárias conforme o problema, buscando encontrar para cada problema o valor do custo ótimo conhecido. As linhas em azul representam os resultados obtidos em uma execução do algoritmo original e a linha vermelha representa os resultados obtidos em uma execução do algoritmo híbrido, ambos utilizando as mesmas instâncias do problema.

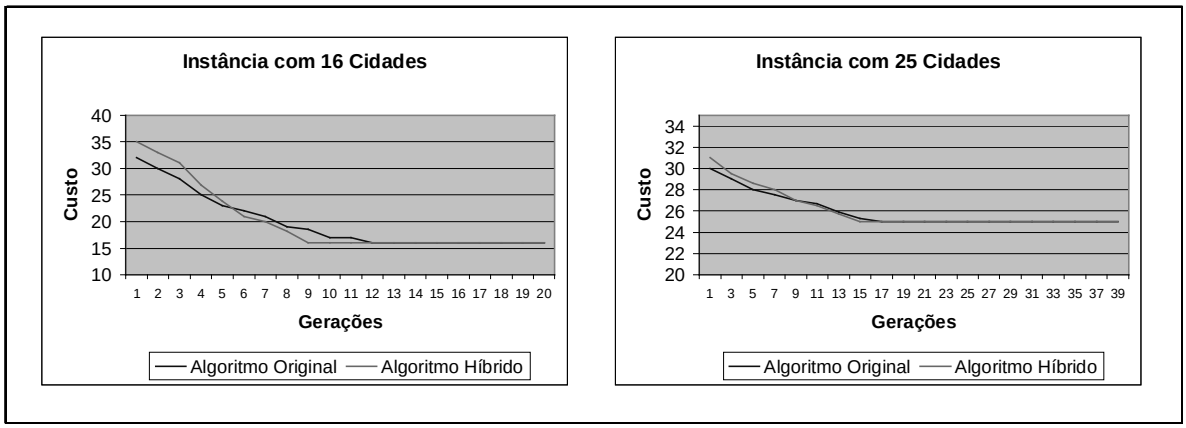

Gráfico 1 - Resultados obtidos com instâncias em grade de 16 e 25 cidades.

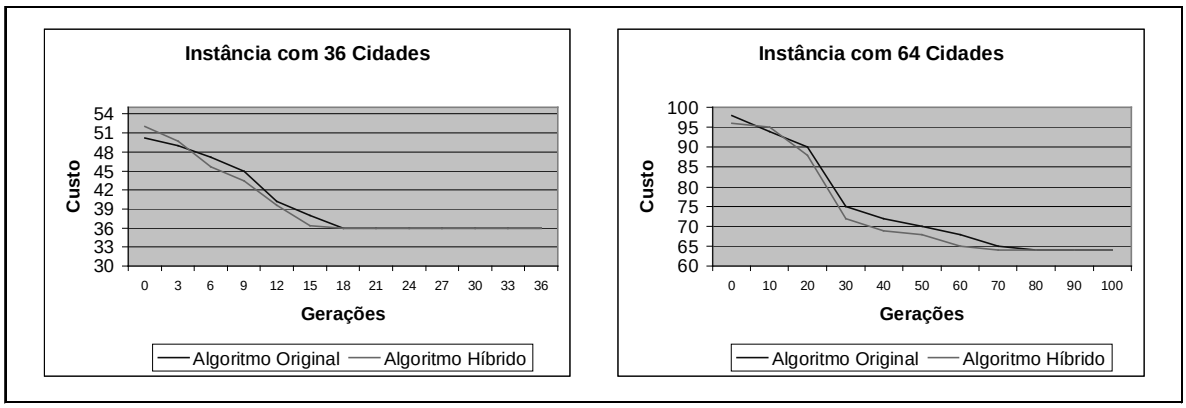

Gráfico 2 - Resultados obtidos com instâncias em grade de 49 e 64 cidades.

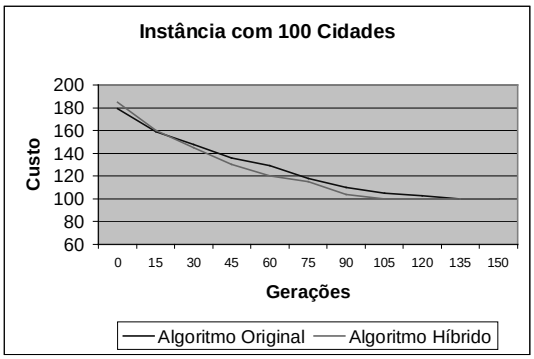

Gráfico 3 - Resultados obtidos com instâncias em grade de 100 cidades.

A tabela abaixo mostra o problema com o número de cidades e o custo ótimo conhecido para o problema. As colunas 3, 4, 5 e 6 mostram a média de gerações necessárias para se encontrar o custo ótimo e o tempo de processamento gasto para a obtenção dos resultados das instâncias testadas.

Problema	Custo Òtimo	Algorítmo Original		Algorítmo Híbrido	
		Média Gerações	Tempo (mm:ss)	Média Gerações	Tempo (mm:ss)
51 Cidades	426	65	0:06:34	58	0:05:52
76 Cidades	539	103	0:11:28	96	0:10:45
101 Cidades	629	165	0:19:12	158	0:18:07
264 Cidades	49135	210	2:10:08	193	2:04:49
783 Cidades	8806	300	8:43:15	265	7:51:34

Tabela 3 - Comparação do desempenho dos algoritmos original e híbrido.

Os gráficos abaixo, mostram o desempenho do algoritmo híbrido e do algoritmo original, utilizando instâncias encontradas na internet com 51, 76, 101, 264 e 783 cidades. Os testes foram realizados variando o número de gerações necessárias conforme o problema.

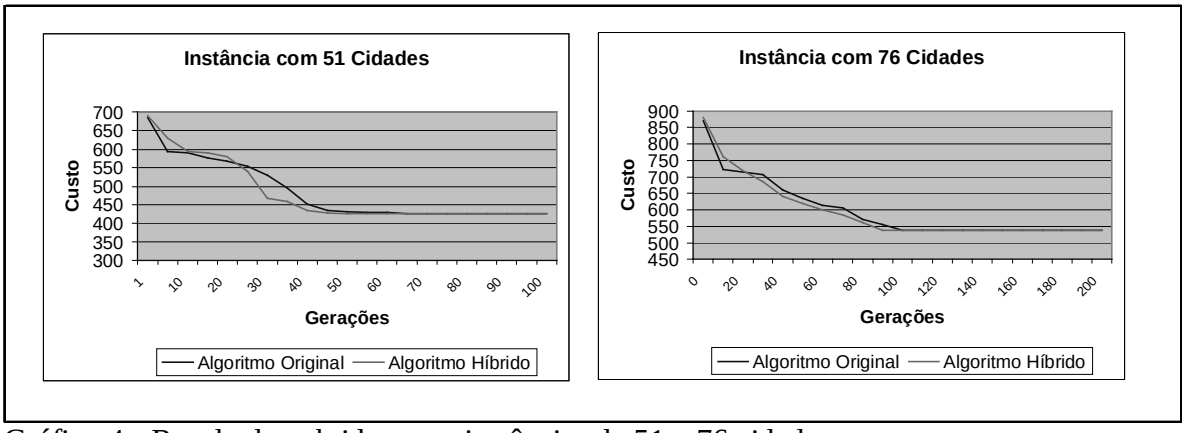

Gráfico 4 - Resultados obtidos com instâncias de 51 e 76 cidades.

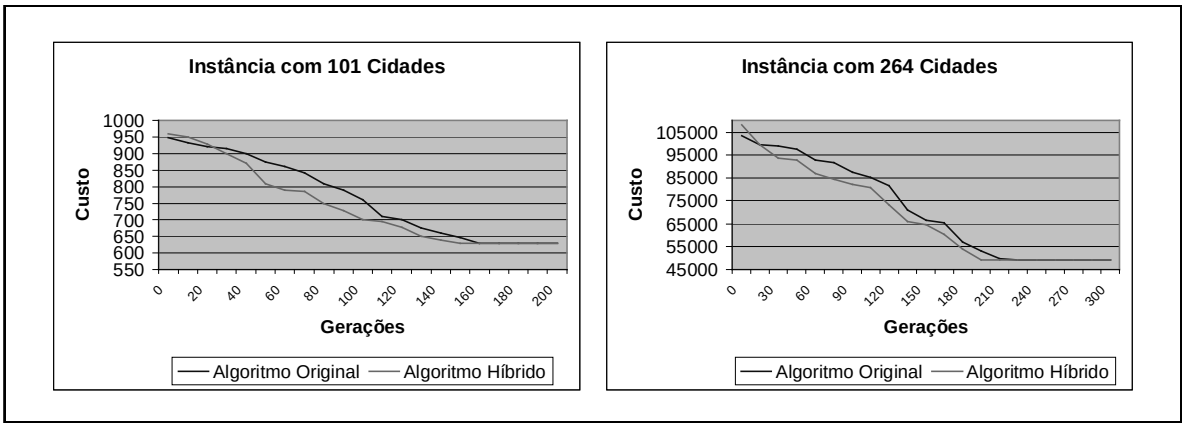

Gráfico 5 - Resultados obtidos com instâncias de 101 e 264 cidades.

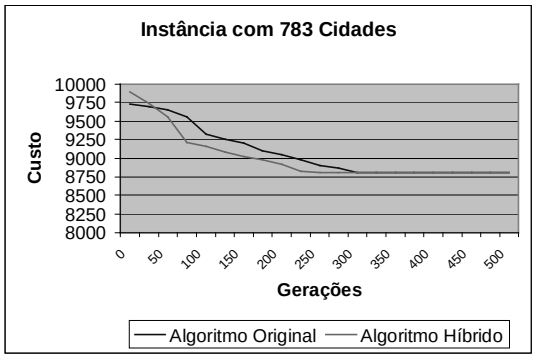

Gráfico 6 - Resultados obtidos com instâncias de 783 cidades.

Os resultados obtidos pelo método proposto são considerados superiores se comparados aos resultados obtidos com o algoritmo original. Os gráficos mostram que o método torna a busca mais agressiva obtendo soluções melhores para as instâncias testadas.

O método proposto se comportou conforme o esperado e como podemos observar, que o método obtem bons resultados chegando a um custo ótimo com um número menor de iterações em relação ao algoritmo original.

Capítulo 7

Considerações Finais

7.1 Conclusões

O crescente número de pesquisas sobre modelos computacionais que buscam inspiração na natureza tem provado que esta é uma fonte inesgotável de exemplos vivos que podem ser modelados e utilizados com êxito. Este trabalho investiga um método novo, baseado em uma abordagem híbrida, utilizando duas técnicas conhecidas e comprovadamente viáveis. A união destes dois métodos pode ser uma alternativa para a resolução de problemas de otimização, neste trabalho aplicado ao problema do caixeiro viajante, porém facilmente adaptado para outros problemas da mesma classe.

Para demonstrar a viabilidade do novo método, foram realizados testes utilizando instâncias do problema do caixeiro viajante. Os testes realizados mostraram que o método proposto pode ser considerado como uma boa alternativa, superando a implementação original do algoritmo das formigas.

As simulações realizadas neste trabalho não priorizam o tempo de processamento para a obtenção do resultado, apesar de ser uma importante característica do tipo de problema. Porém essa característica enfrenta uma grande dificuldade de ser analisada que é a escassez de *benchmarks* que possibilitem, e forneçam embasamento para uma análise mais precisa em relação ao tempo de processamento. Na maior parte dos resultados encontrados, apenas os valores ótimos são expostos, sendo desconsiderados, os métodos utilizados, e o tempo de processamento para obtenção dos resultados.

Neste trabalho, são demonstrados apenas os resultados da implementação do operador genético OX (*Order Crossover*), além deste operador, outros dois foram implementados, o CX (*Cycle Crossover*) e o PMX (*Partially Mapped Crossover)*. Todos os três operados foram testados com instâncias do problema do caixeiro viajante, porém os resultados demonstrados neste trabalho foram obtidos com o operador OX, que obteve soluções melhores se comparados aos outros dois operadores. Os demais operadores encontraram resultados muito inferiores aos obtidos com o OX.

Alguns testes foram realizados com o intuito de testar a influência do número de formigas em relação ao número de cidades. Inicialmente o número de formigas foi igualado ao número de cidades, dessa maneira, em cada cidade uma formiga parte para percorrer o caminho. Esta estratégia é válida para problemas relativamente pequenos, porém para grandes instâncias com números maiores de cidades se torna impraticável. Outros testes foram feitos diminuindo o número de formigas, e percebe-se que com um número maior de iterações encontram-se os mesmos resultados em menos tempo, ou seja, são necessárias mais iterações para se obter bons resultados, mas o tempo de cada iteração é menor. A primeira estratégia é sem dúvida uma alternativa viável, pois explora um espaço maior a cada iteração, porém no aspecto da implementação, ocorre um problema, que é o tempo de processamento de cada iteração. Este problema pode ser contornado adotando uma estratégia diferente, diminuindo o número de formigas, assim são necessárias mais iterações, porém cada iteração leva menos tempo para percorrer o caminho. Outro ponto importante nesta estratégia é o aumento do número de ocorrências do operador crossover que é aplicado ao final de cada iteração, aumentando a chance de se encontrar soluções melhores.

Os resultados obtidos, embora ainda não conclusivos, são otimistas em relação ao potencial do método, podendo ser melhorado em vários aspectos que são citados como proposta para trabalhos futuros.

7.2. Proposta para Novas Pesquisas

Existem diversas pesquisas e diferentes implementações da estratégia de colônia de formigas em ambientes distribuídos, utilizando threads, o que comprovadamente diminui o tempo de processamento. Uma proposta interessante seria a implementação do modelo proposto em um ambiente distribuído, permitindo assim a avaliação de instâncias envolvendo um número maior de cidades. Também poderiam ser aplicados outros tipos de operadores, e avaliar com mais precisão os valores utilizados como parâmetros. Tudo isso em fim, para que se possa obter uma melhora e permitir a comparação com outros métodos levando em conta também o tempo de processamento e a qualidade das soluções.

Referências Bibliográficas

AARTS, Emile; KORST, Jan. **Simulated annealing and boltzmann machines, a stochastic approach to combinatorial optimization and neural computing**. Grã-Bretanha, Courier, 1989.

ARAUJO, Haroldo Alexandre. **Algoritmo simulated annealing: uma nova abordagem**.Universidade Federal de Santa Catarina, Florianópolis, 2001.

BONABEAU, E. DORIGO M. e THERAULAZ G., ***Swarm Intelligence: From Natural to Artificial System*s**. Oxford University Press, New York, 1999.

BRASSARD, Gilles; BRATLEY, Paul. **Fundamentals of algorithmics**. New Jersey : Prentice Hall, 1996.

CHAMBERS, Lance. **Practical handbook of genetic algorithms aplications**. CRC Press LLC, Rio de Janeiro, 1995.

DARWIN, Charles. **The origin of species by means of natural selection**. Chicago: Encyclopedia Britannica, 1953.

DENEUBOURG J.L., PASTEELS J.M. e VERHAEGHE J.C. **Probabilistic behaviour in ants: a strategy of errors**. *J. Theor. Biol.* 105, 259-271, 1983.

DENEUBOURG J.L., ARON S., GOSS S., PASTEELs J.M. e DUERINCK G. **Random behaviour, amplification processes and number of participants: how they contribute to the foraging properties of ants**. *Physica* 22D, 176-186, 1986.

DENEUBOURG J.L., GOSS S., BECKERS R. e SANDINI G. **Collective self-solving problems. Self-Organization, Emerging Properties, and Learning**. (Babloyantz A., ed.) 267-278. New York: Plenum Press, 1991.

DORIGO M., DI CARO G, SAMPELS M. **Ants 2002 – From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms**. Brussels, Belgium, 2002.

DORIGO M. e GAMBARELLA L.M. **Ant Colony System: A cooperative learning approach to the traveling salesman problem**. *IEEE Trans. Evol. Comp.* 1, 53-66, 1997.

DORIGO M., MANIEZZO V., COLORNI A. **Ant System: optimization by a colony of cooperating agents**. *IEEE Trans. Syst., Man, Cybern.* B 26, 29-41, 1996.

EYCKELHOF C. J. **Ant Systems for Dynamic Problems, 2001**. Disponivel em www.eyckelhof.nl. Acesso: Dezembro de 2002.

GOLDBARG, Marcos Cesar; LUNA, Henrique Pacca L. **Otimização combinatória e programação linear:** modelos e algoritmos. Rio de Janeiro: Editora Campus, 2000.

GOLDBERG, David E. **Genetic algorithms in search, optimization, and machine learning**. Massachusets: Addilson Wesley Publisher Company, 1989.

HANSEN, P.B.. **Studies in Computational Science:** Parallel Programming Paradigms. Prentice Hall, 1995.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Cambridge: MIT Press, 1993.

KAUFMAN S. **At Home In The Universe: The Search For Laws Of Self-Organization And Complexity.** Oxford University Press, 1995.

KOZA, John R.; RICE, James P. **Genetic programming II:** automatic discovery of reusable programs. Cambridge: MIT Press, 1994.

LAARHOVEN, P. J. M.; AARTS, Emile. **Simulated annealing:** theory and applications. Holanda: Kluwer Academic Publischers, 1987.

LIN, S. **Computer solutions of the traveling salesman problem**. Bell System Technical Journal, n. 44, p. 2245-2269, 1965.

MAZZUCCO, José Júnior. **Uma abordagem híbrida do problema da programação da produção através dos algoritmos genéticos e Simulated Annealing**. Universidade Federal de Santa Catarina, Florianópolis,1999.

MICHALEWICZ, Zbigniew. **Genetic algorithms + data structures = evolution programs**. Berlin: Springer, 1999.

NOVAES, Antonio Galvão. **Métodos de otimização aplicações aos transportes**. CRC Press LLC, São Paulo, 1978.

PAPADIMITRIOU, Christos H. Combinatorial optimization : algorithms and complexity. Prentice-Hall, Massachusets, 1982.

RAULINO, Rangel G. **Uma abordagem híbrida para solucionar problemas de otimização através dos algoritmos: genético e simulated annealing**. Universidade Federal de Santa Catarina, Florianópolis, 2001.

REINELT, Gerhard. **TPSLIB.** Disponível em: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. Acesso: Dezembro/2002.

TANOMARU, Julio. **Motivação, fundamentos e aplicações de algoritmos genéticos** II Congresso Brasileiro de Redes Neurais, Curitiba,1995.

YONEYAMA, Takashi; NASCIMENTO, Cairo L. Júnior. **Inteligência artificial em controle e automação**. São Paulo, Edgard Blucher LTDA, 2000.

ZONTA, Tiago. **Inteligência Computacional aplicada à resolução do problema de corte unidimensional.** Universidade do Oeste de Santa Catarina, Chapecó, 2001.