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

José Marcio Benite Ramos

Implementação e Análise do Problema do Caixeiro Viajante usando uma nova abordagem através dos Algoritmos Genético e *Simulated Annealing*

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

Prof. Dr. José Mazzucco Júnior
Orientador

Florianópolis, fevereiro de 2001

Implementação e Análise do problema Caixeiro Viajante usando uma nova abordagem através dos Algoritmos Genético e *Simulated Annealing*

José Marcio Benite Ramos

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Conhecimento e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Fernando Álvaro Ostuni Gauthier, Dr. (Coordenador)

_____

José Mazzucco Júnior, Dr. (Orientador)

Banca Examinadora

_____

João B. M. Alves, Dr.

_____

Luiz F. J. Maia, Dr.

_____

Fábio Paraguaçu Duarte da Costa, Dr.

Quando a abstinência do seu néctar o fizer
pensar que tudo está no fim lembre-se,
sempre haverá no fundo um azul que
poderá te salvar.

"NJEHAPPEPDB"

A Deus e a minha família

Aos meus pais, José Ramos Gimenez e Carmem Benite Ramos, pelo amor e paciência que demonstraram ao longo de toda a minha vida.

Agradeço também a todos que, direta e indiretamente, participaram na realização deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Atualmente observa-se uma forte tendência em se utilizar métodos aproximados na resolução de problemas de otimização combinatorial. Esses métodos, que muitas vezes vêm em substituição a métodos exatos, nem sempre garantem uma solução ótima para um problema, porém, normalmente são capazes de oferecer solução aproximada de boa qualidade, em um tempo de processamento aceitável.

Neste trabalho é apresentada e investigada uma nova proposta de um método de aproximação baseado na combinação dos algoritmos Genético (AG) e S*imulated Annealing* (SA). Na observação do seu comportamento foi utilizado o notório problema de otimização combinatorial, de complexidade NP-completo, conhecido como o Problema do Caixeiro Viajante (PCV).

Palavras-chave: Algoritmo Genético. S*imulated Annealing*. Roteamento. Caixeiro Viajante.

Abstract

Currently one observes one strong trend in if using methods approached in the resolution of problems of combinatorial optimization. These methods, that many times come in substitution the accurate methods, nor always guarantee an excellent solution for a problem, however, normally are capable to offer approach solution of good quality, in a time of acceptable processing.

In this work, it was proposal and investigated the potentiality of plus a method, based on the combination of Genetics Algorithms (GA) and Simulated Annealing (SA). In the comment of its behavior the well-known problem of combinatorial optimization was used, of NP-full complexity, known as the Travelling Salesman Problem (PCV).

Key-words: Genetic Algorithm. Simulated Annealing. Routing. Travelling Salesman

Capítulo 1

1.1 - Introdução

Os Problemas de Roteamento e *Scheduling* formam atualmente, um dos segmentos de maior sucesso nas áreas de Otimização Combinatória, Pesquisa Operacional e Computação Aplicada. Este sucesso em parte se deve a eficiência das técnicas existentes na literatura afim, pois quando são implementadas em situações reais, têm fornecido excelentes soluções do ponto de vista operacional. Mas devido a sua elevada complexidade em alguns modelos de Roteamento, o problema de gerar apenas uma solução viável, já é classificado como NP-Completo, e o uso exclusivo de técnicas exatas têm se restringido apenas a pequenas instâncias do problema.

O problema do caixeiro viajante é um problema de otimização de percursos de distribuição que se encaixa nas classes de problemas acima, o qual tem sua importância observada em muitas aplicações nas áreas como manufatura flexível, sistemas de transportes, problemas de roteamento e comunicação de dados, além de ser amplamente utilizado como *benchmark* em abordagens dessa natureza.

Atualmente, observa-se um aumento considerado no interesse por abordagens baseadas em técnicas de otimização de busca local para o tratamento de problemas de roteamento. São métodos aproximados que normalmente utilizam como base os algoritmos *tabu search*, redes neurais artificiais, genético e s*imulated annealing*.

Esses algoritmos têm sido intensivamente pesquisados em soluções de problemas gerais de otimização e, especialmente, nos combinatoriais. O extraordinário sucesso atualmente alcançado por esses algoritmos, principalmente com relação aos dois últimos, é conseqüência de diversos fatores, dentre eles: a utilização de mecanismos de otimização modelados diretamente da natureza, aplicabilidade geral,

flexibilidade nas adaptações às restrições específicas em casos reais, a excelente relação entre qualidade e facilidade de implementação e tempo de processamento.

Seguindo essa linha de pesquisa, baseada no desenvolvimento de mecanismos de otimização modelados diretamente da natureza, diferentes escolas de algoritmos evolucionários surgiram nos últimos anos como: *algoritmos genéticos*, *estratégias evolucionárias e programação evolucionária*. Cada um destes constitui uma aproximação diferente, entretanto, eles são inspirados no mesmo princípio de evolução natural. A idéia é simular aos métodos de melhoramento genético ou de seleção natural. Muito esquematicamente, uma manipulação genética consiste em produzir cromossomos filhos a partir de dois cromossomos pais em substituição dos genes do primeiro cromossomo pai com as partes do segundo. Os algoritmos genéticos foram introduzidos em meados de 1976 por John Holland e seus colaboradores da Universidade de Michigan; contudo o seu pleno desenvolvimento só ocorreu a partir da última década.

Classificado na literatura como uma Metaheurística, os Algoritmos Genéticos ao lado das Redes Neurais Artificiais (RNs) e *Simulated Annealing* são procedimentos que tem solucionado deficiências históricas dos algoritmos convencionais de busca heurística, como por exemplo, o problema de paradas prematuras em ótimos locais distantes da melhor solução em problemas de otimização.

Utilizar-se-á neste trabalho os métodos Algoritmo Genético e *Simulated Anealling* para estudar o comportamento da solução do problema do Caixeiro Viajante.

Resultados computacionais parciais comprovam a validade do uso destas técnicas híbridas onde são aproveitadas as boas características e as filosofias de cada procedimento dentro de um mesmo algoritmo.

1.2 - Objetivo do trabalho

Propõe-se neste trabalho, o desenvolvimento de um novo procedimento Metaheurístico híbrido utilizando Algoritmo Genético e *Simulated Annealing* com implementação seqüencial, que possa ser utilizado como instrumento para a resolução do problema do caixeiro viajante.

O problema do caixeiro viajante é um representante de uma classe de problemas de Otimização Combinatória, sendo de grande importância e que vem atraindo a atenção de muitos cientistas pela sua simplicidade na formulação e dificuldade de resolução.

1.3 - Organização

O trabalho em questão é composto e organizado da seguinte forma:

- Capítulo 2 – caracteriza o contexto geral no qual o trabalho desenvolvido está inserido, apresentando uma abordagem teórica e conceitual do problema do caixeiro viajante, bem como a descrição do problema e o modo com o qual aboradar-se-á a resolução do mesmo, onde definirá as premissas que fundamentam este trabalho.
- Capítulo 3 – introduz uma fundamentação básica conceitual do algoritmo genético, onde os elementos básicos desse método são apresentados e revistos.
- Capítulo 4 – apresenta o método *simulated annealing*, utilizado como melhoramento do AG, seus elementos e funcionamento.
- Capítulo 5 – apresenta a definição formal do problema e a modelagem utilizada no seu tratamento através de representação matemática, onde a abordagem através da combinação dos dois métodos do modelo proposto é definida e comparada com abordagens existentes
- Capítulo 6 – são apresentadas as considerações finais através dos resultados obtidos pelo modelo proposto.

Capítulo 2

2.1 - O Problema Caixeiro Viajante

Neste capítulo apresentar-se-á uma visão geral do contexto no qual o presente trabalho se encontra inserido. Inicialmente serão descritos alguns conceitos fundamentais e uma visão geral do problema do Caixeiro Viajante. Em seguida é apontada, a descrição da solução do problema de uma forma genérica para um número qualquer de cidades a serem visitadas.

2.2 - Conceituação Básica

Um dos problemas de matemática cuja solução tem resistido à passagem do tempo é o Problema do Caixeiro Viajante. A sua descrição é muito simples.

> *Dada uma lista de cidades, um vendedor partindo de uma cidade inicial pretende visitá-las todas uma única vez, e regressar à cidade de partida por forma a que a distância percorrida seja mínima.*

O nome "Problema do Caixeiro Viajante" surgiu pela primeira vez, no livro "*The Traveling Salesman, how he should be and what he should do to get Commissions and to be Successful in his Busines. By a veteran Traveling Salesman*", publicado na Alemanha em 1832. Este livro apresenta a essência do problema em seu último capítulo, onde se afirma: "*The most important aspect is to cover as many locations as possible, without visiting a location twice* [...]". Não se sabe ao certo quando este termo entrou nos círculos matemáticos, mas aponta-se como data provável 1931-32 pela mão de

Merril Flood. Posteriormente, o trabalho Dantzig, Fulkerson e Johnson [DFJ54] trouxe definitivamente o problema do caixeiro viajante para as páginas das revistas científicas, transformando-o num marco histórico da Investigação Operacional.

O problema do caixeiro viajante é um problema de otimização de percursos de distribuição. O objetivo é definir uma rota para o caixeiro de forma que cada cliente seja visitado uma única vez e a distância total percorrida seja mínima. Sua importância é notória no estudo de problemas de otimização combinatoriais pois, além de representar uma classe muito vasta de outros problemas dessa natureza, apresenta duas características fundamentais para o tratamento do referido assunto: simplicidade na formulação e complexidade na resolução.

2.3 – Formulação do problema do Caixeiro Viajante

Um caixeiro viajante deve visitar **n** cidades diferentes iniciando e terminando o seu percurso em uma mesma cidade, não importando a ordem na qual as cidades são visitadas. Considerando que todas as **n** cidades são interligadas, o problema do caixeiro viajante consiste em se determinar um caminho que torna mínima a viagem total.

Considerando que o caixeiro parte de uma cidade determinada, a próxima escolhida deve ser retirada do conjunto das (**n – 1**) cidades restantes. A seguinte, obviamente, do conjunto das (**n – 2**) cidades restantes. Dessa forma, através de um raciocínio combinatorial simples e clássico, concluí-se que o conjunto de rotas possíveis, do qual o caixeiro deve escolher a menor, possui cardinalidade dada por:

$$(n - 1) * (n - 2) * (n - 3) * ...* 2 * 1$$

ou seja, o caixeiro deve escolher sua menor rota de um conjunto de (n – 1)! Possibilidades. Por exemplo, para um pequeno problema com apenas 20 cidades, o número de rotas possíveis seria 19!, ou seja, 121645100408832000 caminhos possíveis.

Se for considerado que a cidade inicial (e final) também deva ser aleatório, o número de rotas possíveis, para **n** cidades, passa a ser dado por **n!**.

2.4 - O método de resolução utilizado

A premissa básica do PCV é a definição uma rota, com menor custo, para contemplar uma visita a todas as cidades e retorno a cidade de partida. De acordo com a descrição acima, uma estratégia seria a geração de cada uma das n! rotas possíveis e, calcular o comprimento total das viagens entre cidades sucessivas de cada rota e verificar qual rota possui o menor comprimento. Por se tratar de um problema NP-Completo, o número de rotas cresce exponencialmente à medida que aumenta o número de cidades visitadas, o que tornaria inviável encontrar uma solução ótima em um espaço de tempo adequado.

Uma abordagem do problema é tentar desenvolver um processo heurístico de solução, ou seja, um processo que usualmente funciona e que se caracteriza por sair buscando soluções cada vez melhores até, se houver sucesso, achar a melhor de todas. Embora não seja garantido encontrar a melhor de todas as soluções, é possível apresentar uma solução aceitável em função do tempo disponível.

No capítulo 4 é detalhada a maneira como foram combinados os algoritmos genético e *simulated annealing,* constituindo uma nova abordagem, na resolução de problemas do caixeiro viajante, tendo como base a abordagem teórica aqui aludida.

Capítulo 3

3.1 - O Algoritmo Genético

Este capítulo aborda o procedimento da evolução, através reprodução genética, que estabeleceu a inspiração para criação do algoritmo genético. Introduzindo o algoritmo através da sua criação apresentando sua estrutura básica e a descrição de seus principais elementos.

3.2 - Introdução

Os Algoritmos Genéticos foram primeiramente desenvolvidos pelo professor John Holland nas décadas de 60 e 70 e formalmente inseridos no seu livro *Adaptation in Natural and Artificial Systems* [HOL93]. Durante suas pesquisas nos processos adaptativos de sistemas naturais e suas aplicações nos projetos de softwares de sistemas artificiais, ele conseguiu incorporar importantes características da evolução dos sistemas naturais aos algoritmos. Para se entender este algoritmo vale citar os principais acontecimentos do processo histórico do desenvolvimento do conhecimento da evolução natural.

Durante muito tempo a teoria do criacionismo (cada espécie havia sido criada individualmente por um ser supremo) e a de geração espontânea, foram as únicas linhas de pesquisas dos naturalistas. No ano de 1775, Carolus Linnaeus, naturalista sueco, apresenta seu trabalho sobre a classificação biológica de organismos, baseado no seu interesse pela similaridade entre certas espécies, levando a acreditar na existência de uma certa relação entre elas, criando o Sistema de Classificação Binária, uma das bases da biologia moderna. Muitos outros cientistas influenciaram os naturalistas em direção à

teoria da seleção natural, tais como os de Jean Baptiste Lamark, que sugeriu uma teoria evolucionária no "uso e desuso" de órgãos; e de Thomas Robert Malthus, que propôs que fatores ambientais tais como doenças e carência de alimentos, limitavam o crescimento de uma população.

No princípio do século XIX, o biólogo francês Jean Baptiste Lamarck aventou a idéia de que as plantas e os animais se alteravam ou se adaptavam em função das mudanças ou de novas ameaças ocorridas no meio em que viviam. Lamarck construiu a teoria de que as características que um organismo alterava ou adquiria durante a sua vida eram transmitidas aos descendentes.

Mas só mais tarde, dos estudos que conduziu em ervilhas-de-cheiro, Gregor Mendel, monge e professor austríaco, formulou um relato preciso da forma como se transmitem as características genéticas de uma geração para a seguinte. O seu trabalho orientou o novo campo da genética e ainda hoje proporciona novos pontos de vista relativamente ao importante papel que os genes desempenham na nossa vida.

Em 1958 Charles Darwin [DAR53] apresentou a teoria de evolução através de seleção natural, depois de anos de pesquisas e observações das espécies, durante suas viagens pelo mundo. Contemporaneamente, o naturalista inglês Alfred Russel Wallace, apresenta um trabalho similar. Em 1959, Darwin publica o trabalho "*On the Origin of Species by Means of Natural Selection*" com a sua teoria completa, sustentada por muitas evidências observadas durante suas viagens.

Segundo Darwin [DAR53], "Cada membro de uma certa espécie é diferente"; atualmente define-se que cada membro tem uma característica genética específica. Todos os seres vivos produzem mais descendentes do que o habitat pode suportar. A competição entre eles determina que apenas uma pequena fração desses seres vivos cheguem à idade de reprodução. No ajuste da procriação ilimitada com a restrição dos recursos, Darwin encontrou o mecanismo que atua constantemente em extinguir muitas mutações, preservando exclusivamente aqueles que conseguem sobreviver e capazes de reproduzir.

Darwin [DAR53] chamou de "seleção natural", a possibilidade de transmissão das características genéticas para futuras gerações obtidas através da diferença entre os indivíduos, juntamente com pressão ambiental. Deste modo, apenas os mais aptos sobrevivem. Darwin concluiu que "a seleção natural introduzia novas espécies no

habitat, o que favoreceria a variedade de indivíduos. Enquanto os descendentes sofrem mutações e as espécies lutam constantemente para proliferarem, quanto mais diferenciados os descendentes se tornam, maior serão suas chances de sucesso para sobreviverem".

Seu trabalho influenciou e revolucionou toda a Biologia, e serviu de base para uma nova linha de pensamento o evolucionismo, a qual também influenciou o pensamento religioso, filosófico, político e econômico. Na mesma época, Charles Babbage, amigo de Darwin, desenvolveu a máquina analítica, considerada a precursora da computação moderna.

A teoria evolucionista, atualmente, combina os pensamentos de Lamark, Darwin e Wallace sobre a seleção natural, estabelecendo o princípio da Genética de Populações: a variabilidade entre indivíduos em uma população de organismos que se reproduzem é regida pelas forças evolutivas de mutação e recombinação genética. Este princípio foi desenvolvido nas décadas de 30 e 40, por biólogos e matemáticos. Nas décadas de 50 e 60, muitos biólogos começaram a desenvolver simulações computacionais de sistemas genéticos [LEC04].

Várias situações na vida real confirmam que a seleção natural é um acontecimento indiscutível, podendo não ser o único mecanismo da evolução, no entanto sem dúvida um dos fatores fundamentais do processo. Exemplo: *A asa da mosca é importante para a sua sobrevivência?* Em quase todo o planeta, as moscas, assim como a maioria dos insetos, tem nas asas um eficiente meio de sobrevivência — com elas, podem procurar alimento e fugir dos predadores. Assim, a asa é extremamente importante, e, sem ela, o indivíduo teria poucas chances de competir e sobreviver nesse meio. Entretanto, nas praias de uma ilha do Pacífico, onde sopra um vento forte e constante, vive, entre pedras empilhadas sem ordem, onde está protegida do vento, uma espécie de mosca sem asas. Ocasionalmente, ocorre uma mutação que gera o aparecimento de asas rudimentares. Basta, então, que essas asas, agora ou nas gerações futuras, sejam suficientes para um "vôo experimental" e o indivíduo é levado para o mar através do vento. É um exemplo marcante e real de como a seleção natural atua. As asas, que são vantajosas em quase todo o planeta, passam a ser nocivas nesse ambiente. O que, normalmente, seria um direcionamento evolutivo natural, será sempre "deletado" nesse ambiente, em especial [BIO00].

A evolução biológica, mesmo nos dias de hoje, ainda não é uma teoria aceita entre todos, apesar da maioria da comunidade científica apóia-la e milhares de trabalhos terem sido publicados, muita coisa ainda não está totalmente compreendida. Contudo, alguns pontos são conhecidos e aceitos [MOL92]:

- A evolução é um mecanismo interno de melhoria dos seres vivos que se transmite às suas descendências. Caracteriza-se como um método de verificação exógeno ao sistema evolutivo propriamente dito ou sistema de criação das modificações genéticas.
- A ação da seleção natural consiste em selecionar genótipos (toda a informação hereditária de um organismo que está codificada em seus cromossomos - longa sequência de DNA) mais bem adaptados a uma determinada condição ecológica, eliminando aqueles desvantajosos para essa mesma condição.
- A pura combinação de um conjunto de elementos irá nos dar sempre um subconjunto do mesmo, ou seja, nunca aparecerão na combinação elementos diferentes dos iniciais. Assim, ao combinarmos uma sequência de três números s = { 1, 2, 3 }, sempre teremos os mesmos indivíduos, isto é, o número 4 (quatro) não irá aparecer. Da mesma forma pode-se pensar no passado, ou seja, para ver a origem da vida.
- A operação de mutação é responsável pela variabilidade dos cromossomos, pois aumenta o número de alelos (cada uma das várias formas alternativas do mesmo gene de um cromossomo), condição que incrementa o conjunto gênico da população. Possibilitando que indivíduos descendentes possuam.
- Através da recombinação genética os cromossomos são reorganizados. A recombinação é uma operação realizada na reprodução sexuada. Essa recombinação é responsável pela singularidade de cada indivíduo de uma mesma espécie.

Durante seus estudos, Holland [HOL93], verificou que o processo de adaptação, inspirado no processo de evolução natural dos seres vivos, baseado nos estudos de Darwin e Mendell, poderia ser incorporado em um algoritmo de busca. Ele observou que os sistemas biológicos desenvolvem, durante seus ciclos de vida, estratégias de adaptação, que possibilitaram a sobrevivência e a perpetuação de suas espécies. Com

base nesta analogia, foi construído um algoritmo matemático para otimização em sistemas complexos, sendo denominado de Algoritmo Genético. O algoritmo apresentado tinha a intenção simular matematicamente o processo da evolução biológica, agregando todas as suas características e vantagens.

Fazendo uma analogia aos sistemas biológicos, os cromossomos foram representados através de cadeias binárias, onde os algoritmos proporcionavam evoluções simuladas em populações de tais cromossomos. A técnica consistia em criar uma população inicial de cromossomos, representando as possíveis soluções. Através de uma função de avaliação, eram verificados quais os melhores indivíduos dessa população. A reprodução era então aplicada nesses indivíduos fazendo combinações entre eles e gerando assim uma nova geração de descendentes. Assim, soluções iniciais, que não eram adequadas ou pouco viáveis para o problema, evoluiriam para novas soluções mais aceitáveis.

Os AG's conservam a informação sobre o ambiente (fornecida através da avaliação de cada cromossomo que os mesmos produzem), embora não conheçam nada a respeito do problema tratado, acumulando-a durante o período de adaptação. Empregavam essa informação para reduzir o domínio de busca, no problema, e originar novas gerações de soluções mais aceitáveis.

Embora simples, os resultados conseguidos com a aplicação desta técnica, segundo Goldberg [GOL89], permitem concluir que os AG são um método de busca robusto, eficiente e eficaz em uma grande variedade de problemas.

Pode-se observar em Mazzucco [JMJ99] uma comparação com o método tradicional de busca baseado em cálculos.

> Este método, que tem sido intensamente utilizado e pesquisado, não é considerado um método robusto, uma vez que depende fortemente da existência de derivadas (valores de inclinação bem definidos). Mesmo que se permitam aproximações numéricas para o cálculo dos valores das derivadas, ainda acaba sendo uma severa restrição. Por outro lado, esse método realiza a busca sempre na vizinhança do ponto corrente, constituindo, portanto, um método de busca local. No método baseado em cálculo,

> assim como em outros existentes, a busca pela melhor solução se processa sempre de um único ponto para outro, no espaço de decisão, através da aplicação de alguma regra de transição. Essa característica dos métodos ponto a ponto constitui um fator de risco, uma vez que, em um espaço com vários picos, é grande a probabilidade de um falso pico ser retornado como solução. Em contraste, o algoritmo genético trabalha simultaneamente sobre um rico banco de pontos (uma população de cromossomos), subindo vários picos em paralelo e reduzindo, dessa forma, a probabilidade de ser encontrado um falso pico.

3.2.1 - Definição de Algoritmos Genéticos

Algoritmos Genéticos são algoritmos de procura, baseados nos mecanismos de seleção natural. Ele combina a sobrevivência de estrutura de caracteres (indivíduos) através de testes de aptidão com uma outra estrutura. Em cada geração, um novo conjunto de indivíduos artificiais, *caracteres*, é criado usando bits e parte do teste de ajuste de uma geração velha; uma parte nova ocasionalmente é experimentada. Algoritmos Genéticos não é nenhum passeio simples, ele explora eficientemente informações históricas para especular novos pontos de procura com uma performance melhorada, Goldenberg [GOL89].

Para Tanomaru [TAN95], 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.

Segundo Ochi et al. [OCH96], Algoritmos Genéticos consistem de programas de evolução baseados nos princípios da seleção natural e transferências de fatores genéticos. Segundo esses princípios, uma dada população de indivíduos, aqueles com

"boas" características genéticas têm uma oportunidade melhor de sobreviver e produzir filhos, enquanto aqueles com características "ruins" tendem a desaparecer. No Algoritmo Genético, normalmente, cada indivíduo, cromossomo, na população corresponde a uma solução do problema. Um mecanismo de reprodução, baseado nos processos evolucionários, é aplicado à população corrente, com o objetivo de explorar o espaço de pesquisa e obter soluções melhores.

Algoritmos Genéticos podem tratar muitos problemas complexos. Eles estão mais próximos da classe de algoritmos probabilísticos, sendo muito diferente de algoritmos randômicos, pois combinam elementos de pesquisas estocásticas e direcionadas. Por isso, AG's são mais robustos que os métodos de pesquisas direcionados existentes. Uma outra importante propriedade é que eles mantêm uma população de soluções potenciais – todos os outros métodos processam um único ponto de pesquisa no espaço Michalewicz [MIC92].

3.2.2 - Codificação e Otimização

A utilização do AG na resolução de um problema depende fortemente de dois importantes passos iniciais, Sheble [S&W99]:

- **Codificação**: encontrar uma forma adequada de se representar soluções possíveis do problema em forma de cromossomo e;
- **Otimização**: determinar uma função de avaliação que forneça uma medida do valor (da importância) de cada cromossomo gerado, no contexto do problema.

Ao implementar um AG é necessária a criação de uma representação eficiente para as soluções apresentadas. Na sua forma convencional, proposta por Holland [HOL93], um AG trabalha adequadamente com uma representação binária (bits 0 e 1) para associar uma solução ou partes de uma solução do problema proposto. Embora a representação binária tenha se mostrado eficiente para vários problemas, verifica-se que muitas as aplicações de AGs não são possíveis de serem representadas através de bits. Assim, surgiram novas formas de representação, no caso do PCV, a forma mais

utilizada e que melhor se adapta é a representação por um vetor de números inteiros, onde cada número representa uma cidade e o vetor representa a rota a ser tomada.

Escolher a representação dos indivíduos é um ponto decisivo para o desenvolvimento de AGs, sendo um dos principais fatores de sua funcionalidade e desempenho. Ao optar por uma representação é necessário avaliar, características como: Koza [KOZ92]

- **Completude**: determina se todas as soluções podem ser representadas;
- **Coerência**: indica se a partir do esquema de representação é possível gerar um genótipo que codifique um fenótipo não pertencente ao espectro de soluções do problema. No caso PCV seria a geração de uma rota inexistente;
- **Simplicidade**: representa o grau de complexidade dos atos de codificação e decodificação das soluções;
- **Localidade**: pequenas alterações no genótipo acarretam pequenas alterações em seu fenótipo correspondente.

A Função de Avaliação é considerada como parte da descrição do problema, a união entre o algoritmo e o problema analisado. Ela exprime o grau de adaptação que uma solução possui, isto é, a quão esta solução é melhor ou pior que outra, baseada no resultado obtido através da avaliação aplicada.

Esta função desempenha um papel crucial no processo de evolução, definindo a aptidão de cada indivíduo, medida essa que o AG utiliza na realização do processo de reprodução. Quanto melhor adaptado, maior a probabilidade desse indivíduo ser selecionado no processo de reprodução, aumentando suas chances de disseminar suas características nas gerações futuras.

A função de avaliação deve ser relativamente simples e rápida, pois como os AG's trabalham com uma razoável quantidade de soluções potenciais (população), avaliar cada um dos indivíduos incide um preço.

"Basicamente, os algoritmos genéticos empregam uma solução candidata (um ponto no espaço de busca) propriamente codificada, denominada cromossomo. Estes cromossomos são agrupados em conjuntos denominados populações. Uma população definida num dado intervalo de tempo é denominada uma geração. Neste contexto, os algoritmos genéticos funcionam através da combinação de pedaços de soluções de uma população, cujas aptidões são determinadas por uma apropriada função de avaliação, de

modo que soluções melhores sejam obtidas a cada geração. A manipulação de soluções é feita por elementos, denominados operadores genéticos.” Oliveira et al. [OOJ97].

3.2.3 - Descrição do Algoritmo Genético

Os Algoritmos Genéticos fazem parte de uma das classes dos chamados algoritmos evolucionários, empregam uma analogia direta ao processo da evolução natural, onde os indivíduos representam as prováveis soluções para um determinado problema. A cada um destes indivíduos é aplicada uma pontuação estabelecida de acordo com sua aptidão, avaliada com base na resposta produzida ao problema. Os mais aptos têm uma maior oportunidade de reproduzirem-se através de cruzamentos entre os mesmos, produzindo descendentes com características de cada elemento. Um Algoritmo Genético desenvolvido de forma correta, produzirá uma população (grupo de possíveis respostas) que convergirá a uma solução adequada ao problema proposto.

Para Michalewicz [MIC92] um Algoritmo Genético (como um programa evolucionário) para um problema particular deve ter os seguintes cinco componentes:

1. Uma representação genética para soluções possíveis para o problema;
2. Um modo de criar uma população inicial com as soluções possíveis;
3. Uma função para avaliação que define regras do ambiente evolucionário, analisando soluções de acordo com o “*fitness*” de cada uma;
4. Operadores genéticos que alteram a estrutura dos descendentes;
5. Valores para vários parâmetros usados nos algoritmos genéticos (tamanho da população; probabilidades aplicadas aos operadores genéticos; etc.).

Cada passo, dos algoritmos genéticos, durante sua evolução no tempo, são chamados de gerações. Partindo de gerações iniciais (população inicial), normalmente criadas aleatoriamente, o algoritmo executa continuamente um laço principal envolvendo três etapas, Davis [DAV91], Goldberg [GOL89] e Tanese [TAN89]:

1. **Avaliação**: pondera cada indivíduo da população calculando sua aptidão, através da função de avaliação;

2. **Seleção**: determina os indivíduos que irão gerar os descendentes para uma nova geração, com base nas aptidões apuradas na etapa de avaliação;
3. **Reprodução**: produz descendentes através da utilização das operações de cruzamento, *crossover*, mutação e inversão sobre os cromossomos escolhidos na etapa de seleção, produzindo a próxima geração e; se o critério de parada for satisfeito, termina e retorna o melhor indivíduo até então gerado, senão volta à etapa de Avaliação.

As etapas apresentadas serão explanadas nos itens **3.2.5**, **3.2.6** e **3.2.7** respectivamente. Elas têm seus comportamentos influenciados por um conjunto de parâmetros necessários a utilização do algoritmo genético:

- Tamanho da população;
- Taxa de crossover e taxa de mutação.
- Critério de parada.

A Figura a seguir mostra o algoritmo genético em pseudocódigo, adaptado de Michalewicz [MIC92]:

Figura 3.1 Algoritmo Genético em pseudo código

```
Programa AG
        {t = 0
        inicia (P, t)
        avalia (P, t);
        repita
                { t = t +1;
                seleção (P,t);
                crossover (P, t);
                mutação (P, t);
                avaliação (P, t);
                sobrevivem (P, t)
        até (t = T)}
}
```

onde:
t - tempo atual;
T - tempo determinado para finalizar o algoritmo;
P – população

3.2.4 - Parâmetros Genéticos

Alguns parâmetros influenciam diretamente o comportamento dos algoritmos genéticos. Neste capítulo analisar-se-á de que maneira estes parâmetros agem, para que se possa determiná-los conforme as definições do problema analisado e dos recursos disponíveis.

- **Tamanho da População**. Define a quantidade de indivíduos (soluções) manipulados em uma determinada geração. Este parâmetro tem influencia direta no desempenho e eficiência dos AG's. Populações reduzidas podem diminuir a performance, pois fornecem uma pequena cobertura (poucas soluções) do domínio do problema. Já populações maiores, normalmente, oferecem maiores coberturas no domínio de soluções do problema, e também previnem convergências prematuras para mínimos locais. Entretanto, para se manipular um maior número de indivíduos, são necessários maiores recursos computacionais ou uma maior quantidade de tempo de processamento.

  Também, uma questão importante é a utilização ou não de tamanho variado para população, ou definição de sub-população. Na prática, o bom senso e conhecimento prévio do problema podem auxiliar na determinação deste parâmetro.

- **Taxa de *Crossover***. O *Crossover* é o principal processo na reprodução dos cromossomos, onde ocorre a troca de estruturas entre os pais. Através do *crossover* novas estruturas podem ser criadas, assim quanto mais elevada for esta taxa, com maior velocidade essas estruturas poderão ser inseridas na população. Entretanto se esta taxa for muito elevada, estruturas com bons desempenho poderão ser removidas com maior velocidade.

- **Taxa de Mutação**. Esta taxa previne que uma determinada estrutura fique inalterada, além de permitir que se atinja qualquer ponto do domínio de soluções do problema. Um valor relativamente baixo reduz a possibilidade mudanças, já valores relativamente elevados, tornam a procura essencialmente aleatória. Aqui uma questão que pode ser tratada, e avaliada, é o uso de taxas mais reduzidas para

populações iniciais e, em gerações futuras, quando as populações são mais homogêneas, usar taxas mais elevadas.

- **Critério de parada**. Como se está trabalhando com problemas de otimização, o ideal seria que o algoritmo parasse no momento que a solução ótima fosse encontrada. O critério de parada mais utilizado, para um algoritmo evolucionário, é definir o número máximo de gerações que serão produzidas, ou um tempo limite de processamento. Também, pode-se utilizar o critério de homogeneidade da população. Pois em gerações mais recentes os indivíduos de uma população apresentam um grau de adaptação mais igualitário. Ou seja, quando o desvio-padrão, *coeficiente de variação*, de uma população é menor que um dado valor, então o processo de geração de novas populações é terminado e a melhor solução é aquela dentre os indivíduos que mais se adaptam à função de avaliação.

3.2.5 - Avaliação

Esta etapa consiste na aplicação de uma função de avaliação em cada um dos indivíduos, cromossomos, da população atual.

Durante o processo de reprodução, cadeias individuais são copiadas de acordo com os valores gerados através de suas funções objetivo f (biólogos chamam essa função de função de avaliação). Intuitivamente, pode-se pensar na função f como uma medida de lucro, utilidade, ou bondade que se queira maximizar. Reproduzir cadeias, de acordo com seus valores de aptidão, significa que, cadeias mais aptas têm maiores chances de contribuir com um maior número de descendentes na geração seguinte. Esse operador é uma versão artificial da seleção natural, isto é, a função de avaliação é o árbitro final que definirá se uma cadeia irá sobreviver ou morrer.

A função de avaliação deve definir se um indivíduo possui uma qualidade mais aprimorada dentre outros indivíduos da população, no domínio do problema considerado. Essa função, é extremamente importante, pois estabelece a ligação entre o AG e o problema a ser resolvido, depende profundamente da maneira com que as soluções foram representadas. Essa função dá, para cada indivíduo, uma medida de

quão bem adaptado ao ambiente ele está, ou seja, quanto maior os valores obtidos pela função objetivo, maior são as chances do indivíduo sobreviver no ambiente e reproduzir-se, passando parte de seu material genético às gerações posteriores, [OOB00].

Em muitos casos, o desenvolvimento de uma função de avaliação pode estar baseado no rendimento e representar somente uma avaliação parcial do problema. Adicionalmente deve ser rápida, já que vai ser aplicada para cada indivíduo de cada população e das sucessivas gerações; devido a este fato, grande parte do tempo gasto por um algoritmo genético se aplica a função de avaliação, [GSI99].

"A grande maioria das aplicações de algoritmos genéticos utilizam representações indiretas das soluções, ou seja, o algoritmo trabalha sobre uma população de soluções codificadas. Desta forma, antes da avaliação de um cromossomo, antes da aplicação da função de avaliação, uma transição da codificação da solução para a solução real, necessita ser feita. Por outro lado, uma aplicação que empregue representação direta de solução, o valor que é passado para a função de avaliação é o próprio cromossomo. Neste caso, toda informação relevante ao particular problema que está sendo tratado, deve estar incluída na representação da solução", Mazzucco [JMJ99].

3.2.6 - Seleção

A base fundamental do funcionamento dos Algoritmos Genéticos é que um critério seletivo vai fazer com que, após várias gerações, o conjunto inicial de indivíduos gere descendentes mais adaptados. O processo de seleção em algoritmos genéticos simula os processos de reprodução sexuada e seleção natural. De maneira geral, uma população temporária de N indivíduos é gerada inicialmente, onde estes são extraídos com probabilidade proporcional à adequabilidade relativa de cada um na população.

Em sua maioria, as técnicas de seleção são projetadas para selecionar, prioritariamente, indivíduos com aptidões mais elevadas, embora não unicamente, com

a finalidade de conservar a heterogeneidade da população. Os critérios de seleção podem ser divididos em:

- **Aleatório**: onde cada indivíduo possui a mesma chance de ser selecionado, independentemente da sua aptidão;
- **Adaptação**: neste processo, indivíduos com baixa adequabilidade terão alta probabilidade de desaparecerem da população, ou seja, serem extintos, ao passo que indivíduos adequados terão grandes chances de sobreviverem, Oliveira [OOB00]. Serão considerados selecionáveis apenas os indivíduos que estiverem acima de uma faixa limite que pode ser representada pela média da população mais uma variação aceitável.

  Em Holland [HOL93], a expectativa individual é calculada através da divisão da aptidão individual pela média das aptidões de toda a população da geração corrente.

$$e_i = apt_i / mapt_t,$$

  Onde: e_i ➔ expectativa do indivíduo *i*;

  apt_i ➔ aptidão do indivíduo *i*;

  $mapt_t$ ➔ aptidão média da população no tempo *t*.

Segundo esta fórmula, verifica-se que 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*; e que a soma dos valores de todas as expectativas individuais será igual ao tamanho da população, Mazzucco [JMJ99].

Este método pode apresentar, pelo menos, dois problemas durante o cálculo das expectativas dos indivíduos:

1. Manipulação de aptidões negativas, as quais poderiam ser obtidas a partir de avaliações negativas. Estes valores não possuem sentido no processo quando se deseja a maximização dos resultados. Uma forma de garantir que funções de avaliação não retornem valores negativos, seria simplesmente, tomar o módulo desta função:

$$f(x) = |f(x)| \qquad 3.1$$

   Esta solução não poderá ser adotada caso a função seja desconhecida, pois se pode obter um máximo falso, através de um valor mínimo (negativo) acentuado.

Outra forma, mais segura, seria utilizar um valor *V* constante mínimo adicionado ao resultado de avaliação. Este valor poderia ser reajustado no decorrer do processo, evitando assim, distorções indesejadas.

$$\begin{cases} f(x) = f(x) + V & \text{se } f(x) + V > 0 \\ f(x) = 0 & \text{se } f(x) + V \leq 0 \end{cases} \qquad 3.2$$

2. Convergência, indesejada, de indivíduos com valores elevados em relação à população. Essa situação poderia ocorrer em duas ocasiões, segundo Mazzucco [JMJ99], no início, quando criada a população inicial, indivíduos poderiam receber uma aptidão acima da média e gerarem uma quantidade desordenada de descendentes. E também, durante o processo, pois indivíduos com maiores aptidões tendem a gerar um maior número de descendentes, o que pode ser acumulativo.

Em Goldberg [GOL89] a expectativa individual é calculada pelo método do truncamento sigma, baseado na aptidão média de toda a população e no desvio padrão das aptidões sobre a população. Onde 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. Dessa forma, a expectativa *e* de um indivíduo *i* é calculada por:

$$e_i = (f_i - f_{med}) / 2dp_t + 1 \text{ se } dp_t \neq 0$$

ou

$$e_i = 1 \text{ se } dp_t = 0$$

Onde: e_i ➔ expectativa do indivíduo *i*;

f_i ➔ aptidão do indivíduo *i*;

f_{med} ➔ aptidão média da população no tempo *t*.

dp_t ➔ desvio padrão das aptidões no tempo (ou geração) *t*.

Este método também pode apresentar alguns problemas como pode ser verificado em Mazzucco [JMJ99]:

1. Caso o *dpi* = 0, então qualquer indivíduo na população terá o mesmo valor de aptidão. Assim, será atribuido, automaticamente, um valor de expectativa igual a *1* para todos os indivíduos da população;
2. Se o *dpi* tornar-se negativo, ele será alterado para um valor pequeno, como por exemplo, *0.1*, de forma que aqueles indivíduos com aptidões muito baixas terão pequenas probabilidades de se reproduzirem. Onde pode-se notar que, a soma dos números esperados de descendentes para todos os indivíduos na população não necessariamente será igual ao tamanho da população, como ocorreria no primeiro.

Um conjunto de operações é necessário para que, dada uma população, se consiga gerar populações sucessivas que (espera-se) melhorem sua aptidão com o tempo. Eles são utilizados para assegurar que a nova geração seja totalmente nova, mas possuí, de alguma forma, características de seus pais, ou seja, a população se diversifica e mantém características de adaptação adquiridas pelas gerações anteriores. Para prevenir que os melhores indivíduos não desapareçam da população pela manipulação dos operadores genéticos, eles podem ser automaticamente colocados na próxima geração, através da reprodução elitista, [GSI99].

3.2.6.1 - Roleta Russa

O número de descendentes de um indivíduo não pode ser o seu valor de expectativa, uma vez que o mesmo é um número real e, por exemplo, para um indivíduo que recebesse o valor de expectativa igual a *1,2* não faria sentido afirmar que o mesmo geraria um descendente e um quinto Mazzucco [JMJ99].

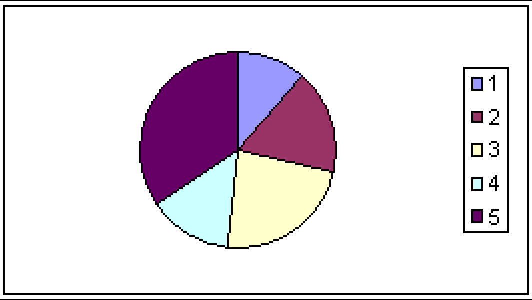
Durante o processo de reprodução, cada indivíduo, da população corrente, irá gerar uma quantidade de descendentes (valor de descendente individual) que pode ser calculada por meio de várias técnicas. Entre elas, pode-se destacar a técnica conhecida como “roleta russa”, utilizada em diversos métodos para simular a seleção natural ocorrida nos indivíduos da população corrente.

Nesta técnica, cada indivíduo da população é representado em uma roleta proporcionalmente ao seu índice de aptidão Ochi [OCH97]. Deste modo, aos indivíduos

com aptidão mais elevada é destinada uma fatia maior da roleta, enquanto aos de menores aptidão é destinada uma fatia relativamente menor da roleta. Em seguida, a roleta é girada uma quantidade definida de vezes, dependendo do número de elementos da população, e então são selecionados, como indivíduos que participarão do processo de reprodução, aqueles sorteados na roleta.

Figura 3.2 - Indivíduos de uma população e a sua correspondente roleta de seleção.

Indivíduo	Aptidão	Med Apd	Porção
I1	30	3,33	11
I2	20	5,00	17
I3	15	6,67	23
I4	25	4,00	14
I5	10	10,00	34
Total	100	29,0	100



Para fins de implementação, a roleta russa pode ser representada através de um vetor *v* de N elementos [1..N], onde N é a somatória das expectativas de cada indivíduo, e seja um índice aleatório *i*, *i* =1,..,N. Então, *v*(i) corresponde ao indivíduo i selecionado no processo. No exemplo acima temos: N = 100, onde, $v=\{I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_1,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_2,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_3,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_4,I_5,I_5,I_5,I_5,I_5,I_5,I_5,I_5,I_5,I_5\}$. Se *i* = 20, então o indivíduo selecionado é o de valor I_1.

3.2.7 - Reprodução

A reprodução é um processo de criação de novos cromossomos, e consiste basicamente em aproveitar características de ambos os progenitores (cromossomos pais) e gerar novos cromossomos filhos baseados nessas características.

Neste processo a seleção é empregada para determinar quais elementos da população atual serão escolhidos para gerarem novos indivíduos para a próxima

geração. Durante esta etapa, os indivíduos selecionados da população serão recombinados para gerar descendentes que constituirão a próxima geração.

O processo de reprodução dos indivíduos da população pode ser dividido em dois passos:

1. Inicialmente é realizada a escolha dos indivíduos que participarão como progenitores da próxima população. Pode-se aplicar a roleta russa, apresentada anteriormente, para definir quantas vezes cada um participará da reprodução;
2. Em seguida, esses indivíduos serão agrupados, dois a dois, aleatoriamente, formando os pares de progenitores.

Os pares de progenitores, por meio da utilização dos operadores genéticos, poderão gerar dois descendentes para a próxima geração. Os principais operadores genéticos e suas descrições serão apresentados a seguir.

3.2.8 - Operadores Genéticos

Estes operadores criam novos nós de busca no domínio de soluções baseado nos elementos da população atual. Segundo Holland [HOL85], os operadores genéticos básicos são: *Crossover*, Mutação e Inversão.

3.2.8.1 - *Crossover*

O processo de *crossover* ou recombinação é um processo sexuado, isto é, envolve pares de indivíduos. Onde são efetuados cortes nos cromossomos pais trocando-se os fragmentos resultantes entre os pares de cromossomos segundo uma regra pré-definida, gerando os cromossomos descendentes. Os cortes podem ser fixos ou aleatórios.

É um dos principais elementos do algoritmo genético, e seu mecanismo é avaliado como componente fundamental do algoritmo. Os cromossomos homólogos

unem-se, duplicam-se e trocam materiais genéticos entre si. Neste processo há uma troca recíproca de segmentos de DNA entre casais de cromossomos.

Crossover: atua sobre um par de progenitores (presumidamente indivíduos aptos), permutando alguns segmentos de suas estruturas. A partir dos progenitores, cortam-se seus cromossomos em uma partição selecionada, para produzir segmentos e, realizar uma troca para obtenção de dois novos cromossomos.

O processo de *crossover* é dividido em três classes de acordo com o corte aplicado nos cromossomos pais:

a) **Um ponto:** Um único corte é feito nos cromossomos pais, trocando-se a segunda parte para obtenção dos cromossomos filhos, utilizado originalmente por Holland [HOL93].

Figura 3.3. Operador Crossover com 1 corte

Cromossomos Pais	**Cromossomos Filhos**
10110010|**0011**	10110010|**1111**
11110011|**1111**	11110011|**0011**
Corte	Corte

b) **Dois pontos:** Dois cortes são feitos no cromossomo, onde são trocadas as partes extremas para obtenção dos cromossomos filhos.

Figura 3.4 Operador *Crossover* com 2 cortes

Cromossomos Pais	**Cromossomos Filhos**
1011|0010|**0011**	**1111**|0010|**1111**
1111|0011|**1111**	**1011**|0011|**0011**
Corte Corte	Corte Corte

c) **Uniforme:** Cada gene do cromossomo pai é escolhido aleatoriamente para geração dos correspondentes genes nos cromossomos filhos.

Figura 3.5. Operador *Crossover* Uniforme

Cromossomos Pais	**Cromossomos Filhos**
1**0**1**10**0**10**0011 **1**11**0**0**0**111**1**1	**1011001111111**

Note que o processo de *crossover* é responsável por uma maior diversidade dos cromossomos gerados durante o processo de reprodução.

Tradicionalmente, o número de operações de *crossover* sofridas por um par de geradores, é determinado por uma variável aleatória, cuja distribuição é a de Poisson, com média igual ao parâmetro fornecido (probabilidade de ocorrência de *crossover* por par de indivíduos). Portanto, se a taxa de *crossover* fornecida for igual a *1*, o número médio de operações *crossover* por par de geradores será *1*. Entretanto, é possível que alguns pares de geradores sofram a operação de *crossover* mais de uma vez, Mazzucco [JMJ99].

3.2.8.2 - Mutação

A operação de mutação é considerada de retaguarda, representando um papel secundário. O processo de mutação consiste na mudança de um gene (característica), escolhido de forma aleatória, em um cromossomo descendente, transformado este em outro alelo (valor de uma característica) possível. O processo é geralmente controlado por uma taxa determinada por um parâmetro fornecido, a probabilidade de ocorrência de mutação dos genes.

Apesar de usualmente concebida como um operador cujo papel é secundário, a mutação é responsável pela introdução de novas soluções no domínio do problema. A operação de mutação alterar uma parte, selecionada aleatoriamente, da estrutura do cromossomo (indivíduo) recém-criado.

No caso numérico a mutação consiste em substituir com determinada probabilidade (taxa de mutação) os valores dos genes de um cromossomo, isto é, o gene

de uma determinada posição no cromossomo é trocado com o gene de uma outra posição.

Com o decorrer das gerações, os indivíduos em uma população tendem a convergirem para valores semelhantes em determinadas posições e a operação de *crossover* não é capaz de introduzir valores diferentes nestas posições. Em determinadas situações, apenas a mutação é capaz de inserir novos valores.

Num Algoritmo Genético básico, o usuário deve definir o tamanho da população além das probabilidades de recombinação, mutação e outros operadores. Quanto ao parâmetro da mutação, a intuição indica que "quanto mais, melhor", entretanto isso pode tornar o processo essencialmente aleatório, uma vez que, as constantes trocas podem afetar toda a estrutura da população, fugindo do foco principal da "evolução natural". Portanto, o bom senso e o conhecimento do problema podem definir taxas adequadas de utilização da mutação.

Em um cromossomo com representação numérica por caminho ou inteiro, a mutação consiste na simples troca entre os genes, semelhante ao processo de inversão mostrado em 3.2.8.3.

Figura 3.6. Processo de Mutação na Representação Numérica

Gene (Mutação) Escolhidos	**Cromossomo resultante Filhos Após a Mutação**
13456**2**	**6**3451**2**

3.2.8.3 - Inversão

O operador inversão consiste na produção de um "*crossover*" em uma estrutura simples de um cromossomo. O processo baseia-se em três passos principais:

1. Seleciona-se um cromossomo (*A*) de uma população;
2. Define-se, na cadeia de *A*, uma posição inicial (x1) e uma posição final (x2), as quais limitaram uma subseqüência de alelos deste cromossomo;

3. Forma-se uma nova estrutura a partir de *A*, invertendo-se a seqüência definida por x1 e x2.

Figura 3.7. Processo de Inversão.

Cromossomo Escolhido	**Cromossomo resultante Após a Inversão**
123456789 x1 x2	127654389

Um quarto operador genético, Clonagem, pode ser considerado, como em Júlio Francisco [JFB99]. O processo de clonagem consiste em manter alguns indivíduos da população atual como integrantes das populações descendentes. Estes indivíduos, considerados mais aptos, permaneceriam constantes durante o processo de reprodução por algumas gerações, o que permitiria transmitir suas qualidades para as gerações descendentes, até que uma geração de indivíduos mais adaptados fosse formada, passando estes a serem considerados no processo de clonagem. É aplicado com uma probabilidade igual a 1-p_c.

3.2.9 - Fundamentação Matemática do AG

Os AG's são relativamente simples de descrever e programar, mas seu funcionamento não é tão simples de ser explicado, existem muitas questões abertas a respeito de como e porque os algoritmos genéticos funcionam.

O teorema fundamental do Algoritmo Genético define o embasamento matemático para o Algoritmo Genético. Será mostrado formalmente, através de uma análise do teorema apresentado em Goldberg [GOL89], considerando os efeitos da reprodução, *crossover* e mutação, durante o processamento de um AG. Esta análise quantifica as taxas de crescimento e redução mais precisamente.

Basicamente, AG opera encontrando, destacando e combinando bons blocos de construção nas soluções. Esses blocos de construção, ou simplesmente esquemas Holland [HOL93], produzem boas soluções. Para analisar o crescimento e a redução

desses esquemas contidos em uma população, serão consideradas apenas, as operações de reprodução, *crossover* e mutação em um único esquema contido em uma população.

O efeito da reprodução em um número esperado de esquemas de uma população é particularmente fácil de determinar. Suponha que em um dado espaço de tempo *t* existam *m* exemplos de um esquema particular *H* contidos em uma população A(t). Considere m(H, t) número de instâncias de *H* no instante *t* (há a possibilidade de diferentes quantidades de diferentes esquemas *H* em tempos *t* diferentes). Durante a reprodução, um indivíduo é selecionado em função de sua aptidão, ou mais precisamente um indivíduo A_i é selecionado com probabilidade $\rho_i = f_i/\Sigma f_j$. Após a obtenção de uma população, não sobreposta, de tamanho *n* com base na população A(t), serão esperadas m(H, t+1) instâncias do esquema *H* na população no tempo *t* + 1, obtido através da equação:

$$m(H, t+1) = m(H, t)*n*f(H)/\Sigma fj \qquad 3.3$$

onde, *f*(H) é a função de avaliação dos indivíduos representando o esquema *H* no tempo *t*. Se for reconhecido que a função de avaliação de uma dada população pode ser escrita como *f*' = Σ*f*j/n então será possível reescrever o esquema de crescimento reprodutivo como:

$$m(H, t + 1) = m(H, t)*f(H)/f' \qquad 3.4$$

Descrevendo, um esquema particular cresce num raio entre a função de avaliação do esquema e a função de avaliação da população. Em outras palavras, esquemas com valores de aptidão acima da média da população irão receber um número crescente de amostras na próxima geração, enquanto esquemas com aptidão abaixo da média terão uma redução no número de amostra. É interessante observar que este comportamento esperado é efetivado com cada esquema H contido em uma determinada população A paralelamente. Isto é, todo esquema em uma população cresce ou diminui de acordo com suas aptidões na operação de reprodução.

O resultado da reprodução em um número de esquemas é qualitativamente claro; ao aumentar a aptidão os esquemas crescem, abaixando a aptidão os esquemas diminuem.

Assumindo que um esquema particular H permanece acima da média em um montante c*f*' com c constante, tem-se *f*(H) = *f* + c*f*'. Com isso, pode-se reescrever a equação 3.4:

$$m(H, t + 1) = m(H, t)*(f + cf')/f' = (1 + c)*m(H, t) \qquad 3.5$$

Iniciando t = 0 e assumindo um valor estacionário para c, pode-se obter a equação:

$$m(H,t) = m(H,0)*(1+c)^t \qquad 3.6$$

O efeito da reprodução torna-se agora visível; reprodução aloca um aumento (ou decremento) exponencial do número de experimentos para acima (ou baixo) da média dos esquemas.

Em seguida, será investigado como *crossover* e mutação afetam essa alocação de experimentos.

A reprodução por si só não faz nada para promover a exploração de novas regiões do espaço de pesquisa, uma vez que não são pesquisados novos pontos; se apenas forem copiadas estruturas antigas sem mudanças, então nunca será experimentado algo novo. É aqui onde entra a operação de *crossover*. *Crossover* é uma estrutura de mudança aleatória de informações entre indivíduos. *Crossover* cria novas estruturas, com um mínimo de perturbações às estratégias de atribuições ditadas pela reprodução. Isso resulta em um incremento exponencial (ou decremento) de proporções dos esquemas de uma população.

Crossover pode destruir ou criar instâncias de *H*. Será considerado, no momento, apenas o efeito destrutivo, isto é, que diminui a quantidade de instâncias de *H*. Considerando $\delta(H)$ o comprimento de *H* e *l* o tamanho de uma cadeia particular, e $\rho_s = 1 - \delta(H)/(l - 1)$ a probabilidade de um simples *crossover* ocorrer sobre uma cadeia, e supondo que uma instância do esquema *H* seja particionada para ser um pai. O esquema *H* é dito *sobrevivente* a uma recombinação (de um ponto) se um dos seus filhos também é uma instância do esquema *H*. Se o *crossover* é realizado através de escolha aleatória, com probabilidade ρ_c em um determinado acasalamento, a probabilidade de sobrevivência pode ser dada pela expressão:

$$\rho_s \geq 1 - \rho_c * \delta(H)/(l - 1) \qquad 3.7$$

O efeito da combinação da reprodução e *crossover* pode ser agora considerado. Quando se considera a reprodução isoladamente, tem-se o cálculo do número de um determinado esquema *H* esperado na próxima geração. Assumindo independência das operações de reprodução e *crossover*, pode-se obter a seguinte estimativa:

$$m(H, t+1) \geq m(H, t)* f(H)/f'[1 - \rho_c * \delta(H)/(l - 1)] \qquad 3.8$$

Comparando com a expressão anterior, para reprodução isoladamente, o efeito da combinação de *crossover* e reprodução, é obtido multiplicando-se o número esperado de esquema para a reprodução isoladamente pela probabilidade de sobrevivência sobre o *crossover* ρ_c. O esquema *H* cresce ou diminui, dependendo de um fator de multiplicação. Em ambos, cruzamento e reprodução, o fator depende de duas coisas: se o esquema está acima ou abaixo da média da população e se a definição do seu tamanho for relativamente curta ou longa.

O último operador a ser considerado é a mutação (alteração aleatória de uma simples posição com probabilidade p_m). Para um esquema *H* sobreviver, todas as posições especificadas devem sobreviver. Portanto, uma vez que um único alelo sobreviva com probabilidade $(1 - p_m)$, e uma vez que cada uma das mutações é estatisticamente independente, um esquema específico sobrevive quando cada uma das o(H) posições fixas, dentro do esquema, sobrevivem.

Multiplicando a probabilidade de sobrevivência $(1 - p_m)$ por o(H) vezes, obtém-se a probabilidade de sobrevivência da mutação, $(1 - p_m)^{o(H)}$. Para valores pequenos de $p_m (p_m << 1)$, a probabilidade de sobrevivência do esquema pode ser aproximada pela expressão $1 - o(H) * p_m$.

Pode-se, portanto, concluir que um determinado esquema *H* recebe um número esperado de cópias na próxima geração em reprodução, *crossover* e mutação, dado pela seguinte equação:

$$m(H, t+1) \geq m(H, t) * f(H)/f' [1 - \rho_c * \delta(H)/(l - 1) - o(H) * p_m] \qquad 3.9$$

A equação 3.9 é conhecida como Teorema Fundamental do Algoritmo Genético ou Teorema do Esquema Holland [HOL93] e Goldberg [GOL89]. Este teorema implica em afirmar que esquemas curtos e de baixa ordem, cujas adaptações permanecem acima da média, recebem aumento exponencial do número de instâncias nas subseqüentes gerações.

Capítulo 4

4.1 - *Simulated Annealing*

Neste capítulo é abordado o método conhecido por *simulated annealing*, utilizado para otimização no processo do AG, apresentando inicialmente o modelo original da física no processo de têmpera do aço, do qual foi inspirado, em seguida são apresentados a sua estrutura e seus elementos básicos.

4.2 - Introdução

O algoritmo de *Simulated Annealing* é inspirado no processo de têmpera do aço, o qual é submetido a altas temperaturas, que são posteriormente, gradativamente abaixadas, até que a estrutura molecular se torne suficientemente uniforme. Se o processo de resfriamento for muito lento, perde-se tempo, mas se o processo for muito rápido, corre-se o risco de estabilizar certas partes do material antes de outras, criando pontos de estabilização locais que depois poderão prejudicar a estabilização do todo. No *Simulated Annealing*, estes pontos correspondem aos máximos locais. O *Simulated Annealing* é uma formulação estocástica oriunda do processo de resfriamento de materiais sólidos, proposto por Metropolis [MET53].

Em Mazzucco [JMJ99], *Simulated Annealing* é considerado um tipo de algoritmo conhecido como de busca local. Ele se constitui em um método de obtenção de boas soluções para problemas de otimização de difíceis resoluções. Desde a sua introdução como um método de otimização combinatorial, esse método vem sendo vastamente utilizado em diversas áreas, tais como projeto de circuitos integrados auxiliados por computador, processamento de imagem, redes neurais, etc. Em alguns

casos mostrou-se superior a heurísticas mais tradicionais enquanto noutros não foi competitivo

Simulated Annealing foi introduzido independentemente nos trabalhos de Kirkpatrick, Gelatt, e Vechi [KIV83], e Cerny [CER85], onde é mostrado como um modelo de simulação de recozimento de sólidos, como proposto em Metropolis [MET53]. O método tem sido aplicado em um grande número de problemas de otimização nas áreas de ciência e engenharia. É comumente empregado em problemas de otimização, onde a função objetivo a ser minimizada corresponde à energia (temperatura) de materiais. É um método computacional que imita o processo natural de achar a estabilização de um sistema com o mínimo de energia.

O algoritmo de SA foi desenvolvido com base em um procedimento utilizado para conduzir um material a seu estado de equilíbrio, ou seja, a um estado de energia mínima. O processo se baseia no aquecimento de materiais e em seguida no resfriamento controlado, o qual é denominado recozimento (*annealing*). Este processo pode ser descrito da seguinte maneira:

1. Aquece-se um determinado material, inicialmente, a uma temperatura elevada, de forma que ele se expanda, ou derreta, e seus átomos passem a se mover com maior liberdade. Com a expansão, aumenta o espaço ocupado pelos átomos.
2. Em seguida, inicia-se o processo de redução da temperatura deste material lentamente, de forma que, a cada redução de temperatura, os átomos possam se mover o suficiente para adotarem um arranjo mais estável.
3. Se o material derretido for resfriado adequadamente, seus átomos serão capazes de atingir um estado de equilíbrio máximo (energia mínima), aumentando sua resistência. Caso contrário ele se tornará uma substância quebradiça.

4.3 - Princípio da Termodinâmica

Em Cerny [CER85], na termodinâmica, grandes sistemas a uma determinada temperatura se aproximam espontaneamente de um estado de equilíbrio, caracterizado

por um valor médio de energia em função da temperatura. Através da simulação da transição para estado de equilíbrio e a arrefecimento da temperatura do sistema, é possível a obtenção de valores cada vez mais baixos para a energia média do sistema.

Assumindo *s* uma configuração do sistema, *E*(*s*) a energia de *s*; os átomos do sistema se encontram em equilíbrio, em uma temperatura *T*. Vale lembrar que um sistema em equilíbrio não se encontra em uma configuração estática. O sistema altera aleatoriamente seu estado, passando de uma configuração a outra de forma que a probabilidade de se encontrar o sistema em uma configuração particular *E(s')* é dada pela constante de Boltzmann, [C&Z00]:

$$P(s) = e^{-E(s')/k*T} \qquad (4.1)$$

De acordo com a expressão (4.1) pode-se constatar que, a probabilidade *P(s)* da energia de um sistema em equilíbrio passar de *E(s)* para *E(s')* é dada por $e^{-E(s')/k*T}$. Como *k* em $e^{-E(s')/k*T}$, é uma constante, observa-se que a medida que *T* diminui, a probabilidade da energia do sistema se alterar torna-se cada vez menor. Pode-se verificar, também, que, nessas condições, quanto menor o valor de *E(s')*, maior a probabilidade da mudança ocorrer.

4.3 - O Algoritmo SA

É através da função *annealing* que o algoritmo simula o processo térmico de resfriamento dos materiais, a uma temperatura elevada. Por meio de uma branda e gradativa diminuição da temperatura, novas configurações do sistema são alcançadas e em cada configuração atingida, o sistema apresenta uma nova quantidade de energia. Este processo termina quando o sistema encontra-se resfriado, atingindo o ponto de solidificação, apresentando uma configuração de energia mínima.

De acordo com Ribeiro e Pires [R&P99], para se empregar o algoritmo SA em problemas de otimização combinatorial, devem ser observados quatro requisitos básicos:

1. Descrição concisa do problema;
2. Geração aleatória das alterações de uma configuração para outra;
3. Uma função objetivo correspondente a "energia do estado" do sistema;

4. Uma definição do estado inicial, do número de iterações a serem executadas para cada temperatura e o seu esquema de *annealing*.

O algoritmo, pertencente à classe de busca local, inicia a partir de uma solução inicial *s* qualquer, obtida de acordo com o **passo 2** acima, com uma temperatura inicial *T* suficientemente elevada. Na sequência, deve ser gerado, aleatoriamente, um vizinho *s'* da solução corrente *s*. A cada geração de um novo vizinho, deve-se testar a variação do valor da função objetivo. Para o teste desta variação é feito o seguinte cálculo:

$$\begin{cases} \Delta E = f(s') - f(s) \\ s = s' \text{ se } \Delta E \leq 0 \\ \text{senão } s = s' \text{ se } e^{-\Delta E/T} > rand[0,1] \end{cases} \qquad 4.2$$

A cada iteração do algoritmo, é dado um deslocamento aleatório a um átomo do sistema, o que implica em uma nova energia do sistema, $\Delta E = f(s') - f(s)$. Se ΔE é menor ou igual a zero,o deslocamento é aceito, isto é, *s* recebe o novo estado *s'*, senão, ainda é possível aceitar o deslocamento se a condição $e^{-\Delta E/T} > rand[0,1]$ definida em 4.1 for verdadeira. Isto é, um número gerado aleatoriamente, entre 0 e 1, for menor que constante de Boltzmann. Isso permite que o processo evite a convergência para um mínimo local, aprovando uma solução s' mesmo que esta represente uma solução com mais energia.

A temperatura *T* é retirada de uma faixa definida pelos limites superior (valor inicial da temperatura) e inferior. Uma dada temperatura é mantida constante por um número predeterminado de repetições e, então, multiplicada por um valor real denominado fator de redução. O processo se repete até *T* atingir o limite inferior da faixa, ou seja, o sistema está estável, Mazzucco [JMJ99].

- A medida que a temperatura *T* vai diminuindo, a possibilidade de aceitações de soluções piores vai se reduzindo, pois o sistema vai se aproximando do ótimo global.

A Figura a seguir mostra o algoritmo em pseudocódigo Eglese [RWE90]:

Figura 4.1 *Simulated Anneling* algoritmo em pseudo código.

```
Selecione um estado inicial i ∈ S;
Selecione uma temperatura inicial T > 0;
Atribua o contador de mudança de temperatura t = 0;
Repita
Atribua contador de repetição n = 0;
      Repita
      Gere estado j, um vizinhode i;
      Calcule δ = f(j) – f(i);
      Se δ < 0 então i := j
            Senão se random(0, 1) < exp(-δ/T) então i := j;
      n := n + 1;
      Até que n = N(t);
t := t + 1;
T := T(t);
Até que o critério de parada seja verdadeiro.
```

Assim, o que o algoritmo *Simulated Annealing* faz é atribuir uma certa "energia" inicial ao processo de busca, permitindo que, além de atingir um mínimo local, o algoritmo seja capaz de aceitar uma solução em que o valor de *f* aumente se a energia for grande. Esta energia é reduzida ao longo do tempo, o que faz com que o processo vá se estabilizando em algum máximo que terá a maior chance de ser uma solução para o problema.

O fator de energia é incorporado no algoritmo *simulated annealing* através de um teste ao estado sucessor. Se o estado sucessor for melhor que o estado atual, a transição ocorre normalmente. Caso contrário, a transição só ocorre se a temperatura atual divida pela diferença de energia for menor do que um número aleatório gerado no intervalo [0,1]. Ou seja, se a temperatura for alta a chance de ocorrer transição é maior do que se a temperatura for baixa. Além disso, se a diferença de energia for pequena, a chance de haver a transição é maior do que se a diferença de energia for grande.

Pode-se comparar o processo de *simulated annealing* a um balão de gás solto em um ambiente cujo teto é irregular com pontos mais baixos como uma caverna, por

exemplo. Se não houver vento no ambiente, o balão subirá e se alojará no primeiro ponto de máximo que encontrar, mas se houver vento, o balão poderá ser deslocado deste ponto (descendo *f*) se alojando em outros pontos de máximos. Quando a quantidade de vento é muito grande, o balão move-se aleatoriamente entre os diversos pontos do teto, mas à medida que este vento vai diminuindo, fica cada vez mais difícil para o balão sir de um buraco.

4.3.1 - Parâmetros do SA

A implementação do algoritmo SA deve ter como ponto inicial a definição dos seguintes parâmetros:

- **Temperatura inicial (T)**: gerada aleatoriamente, mas deve possuir um valor inicial suficientemente alto para que soluções ruins sejam aceitas com maior freqüência no início do processo, permitindo assim, que o sistema descarte mínimos locais.
- **Temperatura final**: próxima de 0 (zero), ou ainda limitada por tempo ou iterações. Finalização depois de um certo tempo ou /iterações sem que haja atualização da solução incumbente.
- **Quantidade de iterações em uma temperatura (n):** define a quantidade de iterações para se alcançar o equilíbrio em uma determinada temperatura. Pode ser representado por um valor fixo de iterações ou um determinado tempo sem alteração no valor da solução atual.
- **Redução da temperatura T(t)**: o processo de redução da temperatura depende de quanto se quer explorar determinadas regiões, quanto mais rápida for a redução, menor será o número de soluções exploradas na região.

4.4 - Considerações Teóricas

Segundo Mazzucco [JMJ99], o algoritmo *Simulated Annealing* pode ser modelado através da teoria de cadeias de Markov. Ele pode ser descrito como uma

seqüência de cadeias de Markov homogêneas de comprimento finito, usando valores decrescentes do parâmetro *T* da temperatura.

Para Davis [DAV93], as cadeias de Markov são modelos com propriedades estocásticas e suficientemente genéricas para resolverem uma grande variedade de problemas categorizados como casos especiais. As Cadeias de Markov consistem de uma mistura de movimentos determinísticos e saltos aleatórios gerando estados.

As Cadeias de Markov são classificadores, Silva & Muntz [S&M92], assim, necessitam menos tempo para sua implementação, são menos susceptíveis a erros, mas modelos realísticos freqüentemente possuem um espaço de estados cuja cardinalidade torna proibitiva a solução.

Em Mazzucco [JMJ99], vários resultados importantes, tratando de condições suficientes para a convergência, têm surgido na literatura. A grande maioria desses trabalhos, entretanto, não leva em consideração o número de iterações necessárias para se atingir essa convergência. Uma vez que o tamanho do espaço de solução *E* cresce exponencialmente com o tamanho do problema, o tempo de execução de um algoritmo desse tipo pode alcançar níveis inviáveis.

Um resultado muito importante é fornecido no trabalho publicado por Sasaki e Hajek [SAH88], no qual, não só as condições necessárias e suficientes para a convergência assintótica do algoritmo para um conjunto de soluções ótimas globais são fornecidas, mas também, o número de iterações necessárias para que essa convergência possa ocorrer.

4.4.1 Cadeias de Markov

O processo estocástico é uma abstração matemática de um processo cujo desenvolvimento é governado por leis de probabilidade. Do ponto de vista matemático, um processo estocástico é definido por uma família de variáveis aleatórias, $\{X(t), t \in T\}$, definidas no conjunto T. O conjunto T é por vezes definido como um espaço de tempo, e X(t) define o estado do sistema no instante t, Costa et. al.[COS00].

Uma cadeia de Markov pode ser descrita como uma sequência de variáveis aleatórias X_k que assume valores em $T \rightarrow R$, tal que:

$$P\{X_k = i_k | X_{k-1} = i_{k-1}, \ldots, X_0 = i_0\} = P\{X_k = i_k | X_{k-1} = i_{k-1}\} \quad 4.3,$$

a cadeia é dita finita se T é finito.

De acordo com 4.3, em uma cadeia de Markov, a probabilidade de mudança de um estado i para um estado j não depende dos estados anteriores.

Capítulo 5

5.1 - O Método Desenvolvido

Neste capítulo, é descrita a forma proposta da combinação dos Algoritmos Genético (AG) e S*imulated Annealing* (SA) no procedimento de resolução do Problema do Caixeiro Viajante (PCV). Inicialmente, o problema é formalmente definido com a apresentação do modelo matemático utilizado. Em seguida, são apresentadas as principais abordagens existentes no tratamento desse tipo de problema. O método desenvolvido é, então, descrito na sua versão seqüencial. Ao final são apresentados os testes computacionais realizados e a análise dos resultados obtidos.

5.2 - Definição do problema

De acordo com Ochi [OCH91] o Problema do Caixeiro Viajante pode ser descrito da seguinte forma: "Dado um conjunto de *n* cidades *N = {1, 2, 3, ..., n}*, e uma matriz de distâncias ligando cada par de cidades de *N*, o objetivo do PCV é partir de uma cidade origem, visitar uma única vez cada uma das *n - 1* cidades restantes e retornar à cidade origem minimizando a distância total percorrida".

Se tivermos n cidades, as rotas (caminhos entre as cidades) podem ser representadas como uma permutação de 1 a n. Isto corresponde fisicamente a um circuito ligando as cidades pela ordem indicada pela permutação e considerando que existe uma ligação entre a última cidade e a primeira. Por exemplo, a permutação [2, 4, 3, 1, 5] corresponde a um circuito que liga as cidades 2 ➔ 4 ➔ 3 ➔ 1 ➔ 5 ➔ 2.

O conjunto das permutações constitui o espaço das potenciais soluções, cuja cardinalidade é **n!**. Para o cálculo do número de rotas para as n cidades, pode-se colocar

qualquer uma das n cidades na primeira posição. Na segunda posição poderá ir qualquer uma das n-1 cidades ainda não escolhidas. Por razão análoga, na terceira posição pode-se colocar qualquer uma das n-2 cidades ainda não escolhidas; na quarta posição pode-se colocar qualquer uma das n-3 cidades ainda não escolhidas e assim por diante. Assim verifica-se que o número total de escolhas que se pode fazer é (n - 1) * (n - 2) * ... * 2 * 1 ou **(n - 1)!**. Além disso, deve ser observado que se a distância entre duas cidades for independente do sentido (distância simétrica), o número de circuitos diferentes é então de (n - 1)!/2.

5.3 - Modelagem do problema

Na aplicação, tanto de SA como AG, na resolução de qualquer problema de otimização combinatorial, é fundamental que se defina primeiramente a forma de como serão representadas as configurações ou soluções do problema.

5.3.1 - Representação por caminho

Representa uma solução de um problema da forma mais natural para descrever uma rota para PCV. O cromossomo é representado por um vetor idêntico à rota correspondente, ou seja, dado um cromossomo do tipo n = (1 2 3 4 5 6 7 8 9), este cromossomo representará uma rota **t** de um PCV com nove cidades, isto é, **t** = {1 2 3 4 5 6 7 8 9}.

Uma representação natural do espaço de soluções para o problema em questão é, portanto, aquela que contém todos os conjuntos de permutação. Define-se, assim, uma solução i para o problema, como sendo o conjunto de permutações das rotas:

$$N_i = \{n_{i1}, n_{i2}, ..., n_{im}\}$$

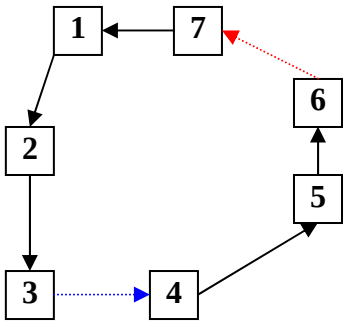
onde cada elemento desse conjunto, n_{ik}, deve ser interpretado como a ordem na qual as rotas entre as cidades n serão processadas, naquela solução N_i. Conseqüentemente, o número de soluções possíveis, ou seja, o tamanho do espaço, para um problema com n cidades, como vimos anteriormente para um número de rotas diferentes é de (n – 1)!/2.

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

Adicionalmente à forma de representação, deve-se definir uma estrutura de vizinhança através da qual, especialmente no SA, é possível se chegar de uma solução a outra por meio de pequenas mudanças. Existem vários métodos para se fazer essa perturbação, entretanto, o mais empregado está baseado na proposta de LIN [LIN93] que é o conceito de *λ-opt*. Neste contexto uma estrutura de vizinhança pode ser definida pelo conjunto de todas as rotas que se obtém removendo k ligações e substituindo-as por k outras ligações, de tal modo que se mantenha uma rota válida. Para k > 3 existem diferentes maneiras de depois de removidos as k ligações, se voltar a ligar os circuitos. Para k = 2, existe apenas uma maneira de ligar: se se retirarem as ligações (i, i + 1) e (j, j + 1) então a única maneira admissível de criar um outro circuito é ligando (i, j) e (i + 1, j + 1).

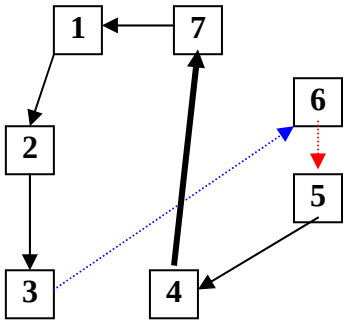
Considere o exemplo de um circuito mostrado na figura abaixo, correspondendo à representação de uma rota para o caso de n = 7 cidades como sendo {1, 2, 3, 4, 7, 6, 5, 1}. Se as ligações (3, 4) ou seja i = 3 e (6, 7) ou seja j = 6, forem removidas, então as únicas ligações possíveis, de modo a manter-se um circuito, são (3, 6) e (4, 7), e invertendo a direção das ligações entre 5 e 6 e entre 4 e 5.

Figura 5.1 – Representação de um circuito ligando as n cidades.



O novo circuito assim obtido, mostrado na figura abaixo, é representado pela seqüência {1, 2, 3, 6, 5, 4, 7, 1}.

Figura 5.2 – Representação de um circuito ligando as n cidades após a troca de duas ligações.



Deste modo cada par de valores i, j, (com i < j) representa um vizinho. A dimensão da vizinhança é o número de combinações de n dois a dois ➔ n(n-1)/2.

Assim, dada uma seqüência, um vizinho é obtido por troca de lugar de dois números nessa seqüência. Além disso, qualquer circuito pode ser obtido pela aplicação de uma ou mais vezes (número finito) desta operação a partir de outro circuito qualquer. Por outras palavras, qualquer ponto do espaço é atingível a partir de qualquer outro ponto do espaço pela aplicação sucessiva do operador de vizinhança.

Adicionalmente esta estrutura de vizinhança é fácil de ser implementada, pois consiste apenas na troca de valores entre duas posições, posições essa que podem ser escolhidas aleatoriamente extraindo sem reposição dois números *i* e *j* entre 1 e *n*.

Retornando à figura 4.2, a solução para o problema ilustrado é:

{1, 2, 3, 6, 5, 4, 7, 1}

esta expressão deve ser interpretada como: o caixeiro parte da cidade 1 viaja até a cidade 2 segue até a cidade 3, e assim por diante. Tendo em vista que a abordagem proposta no presente trabalho envolve a combinação dos algoritmos AG e SA, essa definição de solução é importante, pois pode ser perfeitamente utilizada por ambos.

5.4 - Abordagens Existentes

Como já citado anteriormente, o Problema do Caixeiro Viajante é NP-Completo, o que justifica, portanto, as muitas pesquisas voltadas ao desenvolvimento de heurísticas. Dentre muitos disponíveis na literatura, pode-se destacar os de

Procedimento de Construção de Rotas (Vizinho mais próximo, Método de inserção e Vizinho mais distante), e os de Procedimentos de Melhorias de Rotas.

Devido ao crescimento exponencial do tempo computacional em função do tamanho do problema, a geração de soluções ótimas globais, ainda para problemas reais de tamanho modesto, permanece em princípio, computacionalmente impraticável através de métodos de enumeração completa.

Neste sentido, procedimentos baseados em heurísticas têm sido extensivamente utilizados como abordagens de solução ao problema em questão. Estes tipos de abordagem são hoje, as formas mais eficientes de obter uma solução boa e satisfatória para o problema, em tempo computacional razoável.

Recentemente observa-se um aumento considerado no interesse por abordagens baseadas em técnicas de otimização de busca local para o tratamento do PCV. Basicamente, essas técnicas consistem no refinamento de uma dada solução que, através de um processo iterativo, é submetida a uma série de alterações simples. Cada alteração realizada resulta em uma nova solução. Esses métodos podem ser vistos como ferramentas de busca que atuam em um espaço de soluções viáveis, procurando por melhores soluções, dentro de limitações de tempos razoáveis. Como exposto em Ochi [OCH91], é fundamental ressaltar que, nesse tipo de abordagem, o importante é desenvolver técnicas para a localização rápida de soluções de alta qualidade que se encontram em espaços de soluções imensos e complexos, porém, sem oferecer qualquer garantia de otimização.

Em situações reais, devido as suas características genéricas, os métodos de otimização de busca local podem se apresentar como uma ótima opção, além de que esse tipo de abordagem geralmente é de fácil implementação. Figuram nesta categoria, dentre outros, os métodos baseados nos Algoritmos Genéticos, *Simulated Annealing* e *Tabu Search*, Mazzucco [JMJ99].

Algoritmos Genéticos como propostos em sua forma convencional, não apresentam bons desempenhos quando comparados a algoritmos específicos para problemas conhecidos. É o que acontece com o PCV, o qual possui uma série de abordagens específicas referentes à sua resolução. Com isso, nos últimos anos, os pesquisadores têm proposto alterações nos AGs convencionais, acrescentando técnicas de busca local, e ferramentas de outras meta heurísticas tais como: *Simulated Annealing*,

Tabu Search entre outros, ou desenvolverem versões paralelas para melhorar o desempenho dos AGs.

Essas versões híbridas dos AGs tem como finalidade tornar os AGs convencionais mais especializados, e assim mais eficazes, como se pode observar em Brown [BRS95], Goonatilake [GOK95] e Glover [GLO95]. Dentre as técnicas híbridas envolvendo Algoritmos Genéticos destaca-se a união de AGs com *Tabu Search* Glover [GLO96] e Rochat [ROT95]. Onde, tem se mostrado como ótimas opções para a solução aproximada de problemas de elevado nível de complexidade, principalmente na área de Otimização.

5.5 – Abordagem proposta

De acordo com o que foi exposto, neste capítulo será mostrado o método para combinar o algoritmo genético e *simulated annealing* e como aplicá-lo na resolução do PCV. Em seguida serão realizadas análises dos resultados computacionais obtidos, e uma comparação com algumas soluções propostas.

5.5.1 - Descrição do método

Projetos que envolvem a resolução de problemas através do modelo do Algoritmo Genético, possuem uma estrutura básica, que consiste de algumas etapas definidas durante este trabalho. A descrição do método proposto será detalhada a seguir, na especificação de cada uma das etapas que constituem o Algoritmo Genético proposto.

A utilização do *Simulated Annealing* ocorrerá durante a etapa de *crossover*, onde este fará um papel de "adaptador" dos esquemas dos cromossomos pais que formarão os indivíduos filhos. A descrição sobre o processo *crossover* aplicado no método em questão, assim como sua hibridização com o SA, serão apresentadas em **5.4.5.1** e **5.4.7** respectivamente.

5.5.2 - Representação de uma solução na estrutura de cromossomo para o PCV

Para a codificação de uma solução do PCV será utilizada a representação através de valores inteiros mais especificamente a Representação por Caminhos ou Numérica. Este tipo de representação é mais natural para descrever uma rota para PCV. O vetor cromossomo será idêntico à rota correspondente, ou seja, dado uma rota *t* de um PCV com sete cidades, isto é, t = {1, 2, 3, 4, 5, 6, 7, 1}, esta será representada por um cromossomo do tipo p = (1, 2, 3, 4, 5, 6, 7).

5.5.3 - Construção de uma população inicial de cromossomos

O processo de geração de uma população inicial de cromossomos num AG para o PCV não é tão complicado como possa parecer, pelo contrário, é bastante simples. Será utilizado para este fim um procedimento aleatório. Por exemplo, para o PCV com *n* cidades; gerar *i* soluções viáveis na estrutura de *i* cromossomos, usando uma representação como em **4.4.2**, basta gerar um conjunto de vetores p_i, onde cada p_i é composto por uma seqüência de *n* diferentes números inteiros *c* tal que $c \in \{1, 2, 3, \ldots, n\}$. Por exemplo: p_1 = (1, 2, 3, 4, 5,..., n-1, n); p_2 = (2, 1, 3, 4, 5, …, n-1, n); p_3 = (1, 2, 3, …, n, n-1) …

A população inicial será gerada da seguinte forma: inicialmente seleciona-se aleatoriamente uma cidade. Selecionado esta cidade, escolhe-se uma das alternativas das rotas até a próxima cidade; representando este arco com um número inteiro, o qual representa a cidade escolhida, e assim sucessivamente, selecionando as rotas remanescentes até completar o caminho percorrendo as n cidades e retornando a cidade inicial. Este processo se repete para cada indivíduo da população.

5.5.4 - Avaliação das soluções - Função objetivo

Tal função tem o papel de avaliar o nível de aptidão (adaptação) de cada cromossomo gerado pelos algoritmos evolucionários, quanto maior a adaptação, maior é a chance do indivíduo sobreviver no ambiente e reproduzir-se, passando parte de seu material genético às gerações posteriores.

A função de avaliação a ser otimizada é a soma das distâncias, isto é, minimizar os custos do percurso ou encontrar a menor rota possível. No caso do PCV a função de avaliação *f* é dada por:

$$f = \sum_{i=2}^{n} d_{ci\text{-}1\ c1} + d_{cn\ c1}$$

onde c_i é a cidade visitada na ordem i, i ∈ {1, 2, 3, ..., n} e ***d*** é a distância mínima entre as cidades e dc_nc_1 é a distância da cidade final do percurso até a cidade inicial, a qual completa a rota.

5.5.5 - Reprodução

O processo de reprodução consiste na seleção de indivíduos da população (pais) com maior aptidão, para a formação de novos indivíduos (filhos). A geração dos novos indivíduos está associada à estrutura dos operadores genéticos.

O operador genético representa o núcleo de um AG. Seu objetivo básico é promover a geração de novos cromossomos que possuam propriedades genéticas herdadas dos pais. Os operadores genéticos utilizados no modelo são o *crossover* e mutação, os quais serão vistos a seguir em **5.4.5.1** e **5.4.5.2**.

O processo Roleta Russa será aplicado para a escolha dos indivíduos pais envolvidos na operação de *crossover*. Este processo é baseado na probabilidade associada à aptidão de cada indivíduo, e sua implementação consiste em: a) criar um vetor de tamanho proporcional ao número de indivíduos da população; b) popular o vetor com os indivíduos, atentando para a quantidade de cada indivíduo inserido, a qual deve ser proporcional a sua aptidão em relação à média da população. Seu funcionamento consiste em gerar um número aleatório entre 1 e o tamanho do vetor, o

indivíduo que estiver na posição do vetor, correspondente a este número, será um dos escolhidos para a operação de *crossover*.

5.5.5.1 - Operador *crossover*

O operador *crossover* utilizado, será o operador *crossover* OX (***O****rder Crossover*), criado por Davis [DAV85]. Este operador se adapta perfeitamente ao PCV representado por caminhos, impedindo que seqüências de uma rota (esquema) inválidas sejam criadas no processo de geração dos filhos. Além de manter uma subestrutura, subseqüência de uma rota, associada a um cromossomo pai para os descendentes. O operador OX prioriza a ordem das cidades e não suas posições na rota.

Davis propôs um operador que constrói um descendente (*offspring*) escolhendo uma subseqüência de uma rota associada a um cromossomo pai p_1 e preservando a ordem relativa das cidades do outro cromossomo pai p_2. Este operador é uma extensão do operador *crossover* um ponto, visto em **3.1.7.1**. Uma posição de corte é escolhida aleatoriamente no primeiro cromossomo pai, então um descendente é criado herdando a primeira parte do corte e em seguida preenchendo a segunda parte a partir do gene inicial do segundo pai, eliminando-se as repetições.

Por exemplo, considerando os cromossomos pais p_1 = (1, 2, 4, 5, 3, 6, 7, 9, 8) e p_2 = (2, 3, 1, 5, 6, 9, 8, 4, 7), dois esquemas utilizando-se informações dos cromossomos pais são criados através de um corte aleatório, os quais servirão de base para a geração dos descendentes, como se pode ver na Figura 4.3. Em seguida, um segundo esquema é montado a partir do primeiro gene do segundo pai, eliminado os genes já existentes no primeiro esquema, e mantendo a ordem em que são encontrados. Desta maneira, a junção dos dois esquemas resulta em um cromossomo filho com uma rota válida.

Figura 5.3 – Representação do operador *crossover* OX.

Cromossomos Pais	**Esquemas gerados pelo corte a partir do primeiro pai**	**Esquemas resultantes, gerados a partir do segundo pai**
1 2 4 5 3 | 6 7 9 8 corte	**1 2 4 5 3**	**6 9 8 7**
2 3 1 | 5 6 9 8 4 7 corte	**2 3 1**	**4 5 6 7 9 8**

A Figura 5.3 mostra o processo de criação de dois cromossomos filhos através do operador OX. Como se pode observar, este processo resulta em dois cromossomos filhos o_1 = (1, 2, 4, 5, 3, 6, 9, 8, 7) e o_2 = (2, 3, 1, 4, 5, 6, 7, 9, 8,).

5.5.5.2 - Operador mutação

O operador mutação, como visto em **3.1.7.2** e **3.1.3** altera um ou mais genes de um cromossomo de acordo com uma função de probabilidade. Seu processo consiste na mudança de um gene, escolhido de forma aleatória, em um cromossomo descendente, transformado este em outro alelo possível. Para a representação por inteiro ou posição, o processo de mutação consiste em trocar de posição dois alelos do cromossomo.

Por exemplo, considere um cromossomo p = (1, 3, 2, 4, 5, 6), aplica-se uma função probabilidade, caso esta função retorne um valor dentro dos parâmetros pré-definidos, seleciona-se aleatoriamente dois genes. Suponha que os genes c_2 e c_4 foram os escolhidos para a mutação; troca-se as posições de c_2 e c_4, obtendo um cromossomo filho (*offspring*) p = (1, 4, 2, 3, 5, 6).

A função probabilidade é bem simples, podendo ser implementada apenas gerando um número aleatório entre 0 e 100. O parâmetro de mutação (porcentagem em que a mutação ocorrerá) fornecido é comparado com o número gerado pela função, se este for menor então o processo de mutação é executado.

5.5.6 - Definição dos parâmetros

Os parâmetros de um AG podem ser definidos no momento da execução, e permitem um controle mais restrito do tempo de processamento e da quantidade de soluções estudadas. Conforme abordado em **3.1.3**, os parâmetros utilizados na implementação serão:

- tamanho da população;
- taxa de *crossover* e taxa de mutação.
- critério de parada.

Tamanho da População

Permite a definição da quantidade de soluções estudadas em cada geração. O tamanho da população tem grande influência no número de *crossover* realizados durante cada geração e na quantidade de memória utilizada. Define o tamanho do vetor que contém cada um dos indivíduos (cromossomos) de cada geração.

Taxa de *Crossover*

Define a porcentagem de *crossover* realizada durante o processo de reprodução. Conforme abordado em **3.1.7.1**, para cada par de cromossomos pais, serão gerados dois cromossomos filhos. Se a taxa de *crossover* for de 100% (cem por cento), todos os indivíduos da população atual serão substituídos pelos descendentes. Portanto esta taxa regula a quantidade e freqüência com que os indivíduos participarão das novas gerações sem ter seu material genético alterado.

Taxa de Mutação

A mutação faz com que a estrutura de um cromossomo se altere, permitindo a geração de uma nova estrutura. Ela torna possível a introdução de uma nova solução em uma população, ampliando assim a área de procura. Durante o processamento, conforme abordado em **3.1.3**, esquemas com maior aptidão tendem a se propagarem pela população, fazendo com que as soluções tornem-se homogêneas, a mutação pode inserir novas soluções alterando a seqüência dos alelos de um cromossomo.

Este parâmetro define a proporção com que a mutação será realizada em cada filho de uma nova geração. Para cada novo filho criado em um *crossover*, um número aleatório entre 0 e 100 é gerado, caso este número seja menor que o parâmetro estipulado, a função definida pelo operador mutação, em **5.4.4**, é executada.

Critério de Parada

Este parâmetro tem como principal papel limitar o tempo de processamento das soluções estudadas. Como a maioria dos problemas tratados, incluindo o PCV, por Algoritmos Genéticos não têm soluções pré-definidas ou estas podem durar tempos impraticáveis de processamento, é necessário definir um limite para o processamento.

O critério de parada utilizado, é definido pelo número máximo de gerações produzidas. Assim, o processo de geração de novas populações é terminado quando o número de gerações pré-definido é atingido, e a melhor solução é aquela dentre os indivíduos que mais se adaptam à função de avaliação.

5.5.7 - A Hibridização com *Simulated Annealing*

Nessa seção se concentra a parte mais importante deste trabalho. Nela é descrito o processo proposto e implementado da hibridização dos algoritmos Genéticos e Simulated Annealing.

O processo de geração dos dependentes se completa com a utilização do SA. Este atuará como um adaptador dos esquemas gerados, no segundo passo, durante o processo de *crossover* em **5.4.5.1**. Estes esquemas, apesar de preservarem a ordem das cidades do segundo pai, não mantêm as adjacências das mesmas, como ilustrado na figura abaixo, um fator de grande importância para o PCV como mostrado por Grefenstette [GRE85]. Seguindo esse raciocínio, a aplicação do SA nestes esquemas tem como objetivo tratar esta alteração sofrida nas adjacências, de maneira que a rota representada por esta subseqüência de cidades tenha seu percurso minimizado.

Figura 5.4 – Representação dos esquemas onde será aplicado o SA.

Cromossomos Pais	Esquema gerado pelo corte a partir do primeiro pai	Esquema resultante do segundo pai, onde será aplicado o SA
1 2 4 5 3 6 | 7 9 8 corte	1 2 4 5 3 6	9 8 7
2 3 1 | 5 6 9 8 4 7 corte	2 3 1	4 5 6 7 9 8

A utilização do SA se dá durante o processo do *crossover*, sendo o mesmo aplicado antes do término do processo de *crossover*.

O processo completo de hibridização do operador OX com o SA é mostrado a seguir:

1º Passo ➔ Corte nos cromossomos pais selecionados. Os cortes são executados em posições aleatórias nos cromossomos pais:

pai1 = 124536|789;

pai2 = 231|569847.

2º Passo ➔ Mantém-se o esquema da esquerda gerado pelo corte para cada pai. O corte, nos cromossomos pais, gera dois esquemas como pode ser visto acima, um com os alelos (124536) da esquerda e (789) da direita para o primeiro pai. Os esquemas da esquerda são mantidos inalterados para a geração dos cromossomos filhos.

Esquemas mantidos inalterados para a geração dos filhos:

filho1 = 124536XXX;

filho2 = 231YYYYYY.

3º Passo ➔ O processo do operador OX terminaria com o preenchimento dos alelos restantes, para cada filho, oriundo do pai oposto. Isto é, para o filho1 são escolhidos, na ordem em que aparecem no pai2, os alelos que faltam para completar a sua rota. Assim tem-se,

XXX = 789 e

YYYYYY = 456798.

4º Passo ➔ O algoritmo SA será aplicado em XXX (789) e YYYYYY (456789), antes de concatená-los aos filho1 e filho2 respectivamente. Isto é, antes de se completar o processo de *crossover*.

Os esquemas XXX e YYYYYY serão tomados como soluções iniciais no processo do SA e tratados como rotas completas, embora sejam apenas sub rotas de um cromossomo. A aplicação do SA nestes esquemas tem como objetivo a adaptação dos mesmos, antes de juntarem-se aos esquemas gerados no 2º passo.

5º Passo ➔ Após ser aplicado o SA em XXX e YYYYYY, estes poderão ter suas estruturas (seqüência dos alelos) alteradas, representados agora por PPP e ZZZZZZ. O processo é completado concatenando-se os esquemas PPP a 124536 (herdado do pai1 no 2º passo) e ZZZZZZ a 231 (herdado do pai2 no 2º passo), gerando-se os cromossomos filhos:

filho1 = 124536PPP;

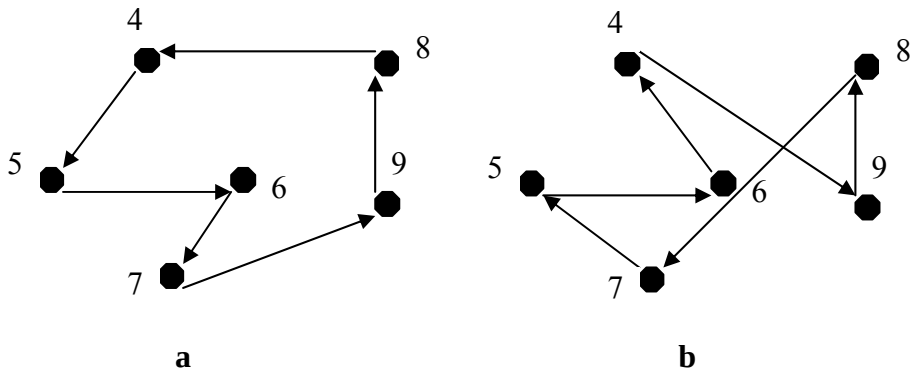
filho2 = 231ZZZZZZ.

5.5.7.1 - Estrutura de Vizinhança

A estrutura de vizinhança aplicada no SA, durante o processo de adaptação dos esquemas vistos no item anterior, é de implementação simples e segue o mesmo raciocínio apresentado no item **5.3**. Dada uma solução corrente T_i = {4, 5, 6, 7, 9, 8}, obtida a partir da figura 4.4, uma transição é efetuada, ou seja, uma nova solução nas vizinhanças de T_i é gerada, primeiramente, escolhendo as posições c_i das cidades correspondentes a rota, por exemplo c_1 e c_4.

Após a escolha das posições, uma nova solução é obtida pela troca das cidades que se encontram nestas posições. Dessa forma, em T_i, uma transição como esta resulta na reversão da cidade 4 e 7, obtendo-se como solução T_{i+1} = {7, 5, 6, 4, 9, 8}. A figura abaixo ilustra um processo de reversão, considerando (a) como a situação original e (b) como a situação final.

Figura 5.5 Ilustração de um processo de reversão das posições de cidades.



5.5.7.2 - O Controle da Temperatura no SA

Para o controle da temperatura, inicialmente é atribuído um valor para a temperatura máxima, e o rebaixamento é expresso pela seguinte expressão:

$$temp = temp * \propto$$

onde, $\propto$ representa a constante de rebaixamento da temperatura.

Alguns parâmetros devem ser observados para durante o processo de controle da temperatura:

- Temperatura Inicial deve ter um valor suficientemente alto para que soluções ruins sejam aceitas com bastante freqüência no início da heurística. Visando assim, uma busca mais ampla durante os primeiro passos;
- Processo de redução (alteração) da temperatura depende de quanto se quer explorar determinadas regiões. Para uma exploração maior, a redução deve ser menos acentuada, ou seja, o valor de $\propto$ deve ser próximo a 1;
- Temperatura final não é definida, o término do processo, no modelo proposto, é limitado por iterações.

5.5.8 - Escolha da nova geração

Depois de completado os processos de reprodução obtêm-se duas populações, a dos geradores e a prole. A população que irá prosseguir com um novo ciclo de reprodução, será formada pelos indivíduos mais adaptados de cada uma das populações.

O processo de geração desta nova população consiste na seleção do indivíduo mais adaptado entre os integrantes das duas populações atuais, em seguida o próximo é escolhido dentre os restantes, e assim sucessivamente até que uma nova população, com o mesmo número de indivíduos da geração anterior, seja criada.

5.6 - Análise dos resultados computacionais

Os testes realizados utilizaram os problemas propostos pela internet na *home page* da TSPLIB [TSPLI]. Para isso, foi desenvolvido um sistema na linguagem Pascal, utilizando-se o ambiente do Delphi 5, e o computador utilizado AMD K6-2 500 MHz com 64 Mb de RAM.

Para comparação com o método proposto utilizou-se o Algoritmo Genético com operadores *crossover* clássicos (OX e ERX), e o *Simulated Annealing* proposto por Cerny [CER85]. Os operadores *crossover* utilizados foram escolhidos por apresentarem os melhores resultados em relação ao Problema do Caixeiro Viajante. Em seus estudos Oliver et al. [OLI97], constatou que o operador OX tinha um desempenho melhor que os operadores PMX e CX. Criado por Grefenstette et al. [GRE85] o operador ERX mostrou-se bem adaptado ao PCV, pois prioriza as adjacências das cidades, fator importante ao PCV. O operador ZEX foi criado durante o desenvolvimento do método proposto. O método desenvolvido neste trabalho será referenciado pelo operador AGSA (Algoritmo Genético *Simulated Annealing*).

Na Tabela 5.1 estão relacionados os resultados referentes ao problema proposto por Christofides/Eilon (EIL51), com 51 cidades representadas por pontos no espaço. Nas Tabelas 5.2 e 5.3 estão relacionados os resultados referentes aos problemas propostos por Fricker (FRI26) e Groetschel (GR24) respectivamente, com 26 e 24 cidades e a relação das distâncias entre todas. Para cada operador genético comparado

são tabelados: a média dos valores encontrados em cinco rodadas (MMS), a média dos tempos de processamento em segundos (TG) e o melhor resultado alcançado (MS).

Tabela 5.1 Resultados do PCV utilizando EIL51.

Operador	NC[1]	NG[2]	TP[3]	TG	MS	MMS
OX	51	100	100	00:00,8	651,972887	721,330017
ERX	51	100	100	00:01,9	681,100512	693,179946
ZEX	51	100	100	00:05,5	838,69559	912,012934
AGSA	51	100	100	02:31,1	486,552208	502,997675
Simulated Annealing	51	2		00:14,9	899,820578	947,007091

[1] NC = Número de Cidades.

[2] NG = Número de Gerações.

[3] TP = Tamanho da População.

Tabela 5.2 Resultados do PCV utilizando FRI26.

Operador	NC	NG	TP	TG	MS	MMS
OX	26	100	100	00:00,6	1144	1166
ERX	26	100	100	00:00,9	1055	1119
ZEX	26	100	100	00:02,5	1306	1396,4
AGSA	26	100	100	00:39,9	937*	967,4
Simulated Annealing	26	2		00:05,7	1482	1600,6

Tabela 5.3 Resultados do PCV utilizando GR24.

Operador	NC	NG	TP	TG	MS	MMS
OX	24	100	100	00:00,6	1572	1674,8
ERX	24	100	100	00:00,8	1496	1602,6
ZEX	24	100	100	00:02,3	1764	1875,4
AGSA	24	100	100	00:34,7	1272*	1312,8
Simulated Annealing	24	2		00:05,2	1929	2043,2

Os números tabelados em 5.2 e 5.3, seguidos de um asterisco, correspondem aos valores ótimos dos testes correspondentes. Na Tabela 5.1 nenhum dos métodos apresentado obteve o resultado ótimo para o problema proposto (MS = 437). Como se pode observar, somente o operador AGSA retornou uma rota ótima para FRI26 e GR24, e apesar de não encontrar em EIL51, foi o que mais se aproximou do menor valor.

A grande desvantagem do operador AGSA, em relação aos outros operadores propostos, está relacionada com o tempo de processamento. Como se pode observar nas tabelas acima, a média para o operador AGSA é muito maior que a dos outros.

A Tabela 5.4 mostra os resultados obtidos pelos operadores OX, ERX e ZEX com os parâmetros NG = 900 e TP = 900 para EIL51.

Tabela 5.4 Resultados do PCV utilizando EIL51, NG = 900 e TP = 900.

Operador	NC	NG	TP	TG	MS	MMS
OX	51	900	900	02:50,2	601,808742	619,731549
ERX	51	900	900	03:41,2	533,282934	550,768178
ZEX	51	900	900	04:28,0	865,501517	869,01114

Nota-se que os operadores OX, ERX e ZEX tiveram uma melhora significativa nos resultados, contudo os tempos de processamento obtidos foram maiores que os do operador AGSA, analisados na Tabela **5.1**.

Capítulo 6

6.1 - Conclusões

Neste trabalho, foi proposta e investigada a potencialidade de mais um método, baseado na combinação dos algoritmos genético (AG) e *simulated annealing* (SA). Embora se tenha dado ênfase ao PCV, há uma independência do método com relação ao problema tratado, pois não é necessária a incorporação de conhecimentos particulares do problema ao método de resolução. Apenas a conversão em uma estrutura de cromossomo e uma função de análise são atreladas ao problema tratado.

Através dos resultados obtidos em **5.6**, observa-se que numa comparação interna entre as versões dos operadores *crossover* utilizados no AG, a versão SA foi a melhor. Estes resultados parciais produzem uma projeção muito otimista em relação ao potencial do método proposto, já que estes são os primeiros testes realizados e ainda sem nenhum estudo de controle de parâmetros, estudos estes, que podem produzir uma melhora no desempenho do AG.

A qualidade da abordagem proposta neste trabalho, fruto das inovações realizadas, foi enfatizada através dos testes realizados com problemas dispostos via internet, e muito utilizados por pesquisadores em testes de métodos para resoluções do PCV. Entretanto, deve-se ressaltar que o algoritmo aqui proposto pode se adaptar a qualquer problema que possa ser resolvido por otimização combinatorial.

6.2 - Propostas de Novas Pesquisas

Estudos complementares poderiam ser realizados, sobre os efeitos na alteração de parâmetros do AG, com o objetivo de se buscar os "intervalos ótimos e as relações de dependências" de cada um deles no AG.

Resultados computacionais atestaram que o método apresentado consegue encontrar soluções melhores se comparado a outros métodos de algoritmos genéticos propostos para o mesmo fim. Entretanto, o tempo de processamento necessário para rodar este algoritmo em problema desta natureza, é considerado muito alto. Uma alternativa para se tentar reduzir esse tempo de execução é implementá-lo como um algoritmo distribuído, onde várias tarefas possam ser realizadas em paralelo.

Uma outra possibilidade de otimização nos tempos de processamento seria o desenvolvimento, como projeto futuro, de uma versão paralela do AG para sistemas do tipo PCV.

Como exposto em **6.1**, o método proposto é independe do problema tratado, permitindo assim que estudos futuros possam ser realizados com outras classes de problemas.

Anexo 1 - Layout das telas do sistema desenvolvido.

Figura A.1.1 Tela principal do programa Algoritmo Genético, desenvolvido em Delphi 5.

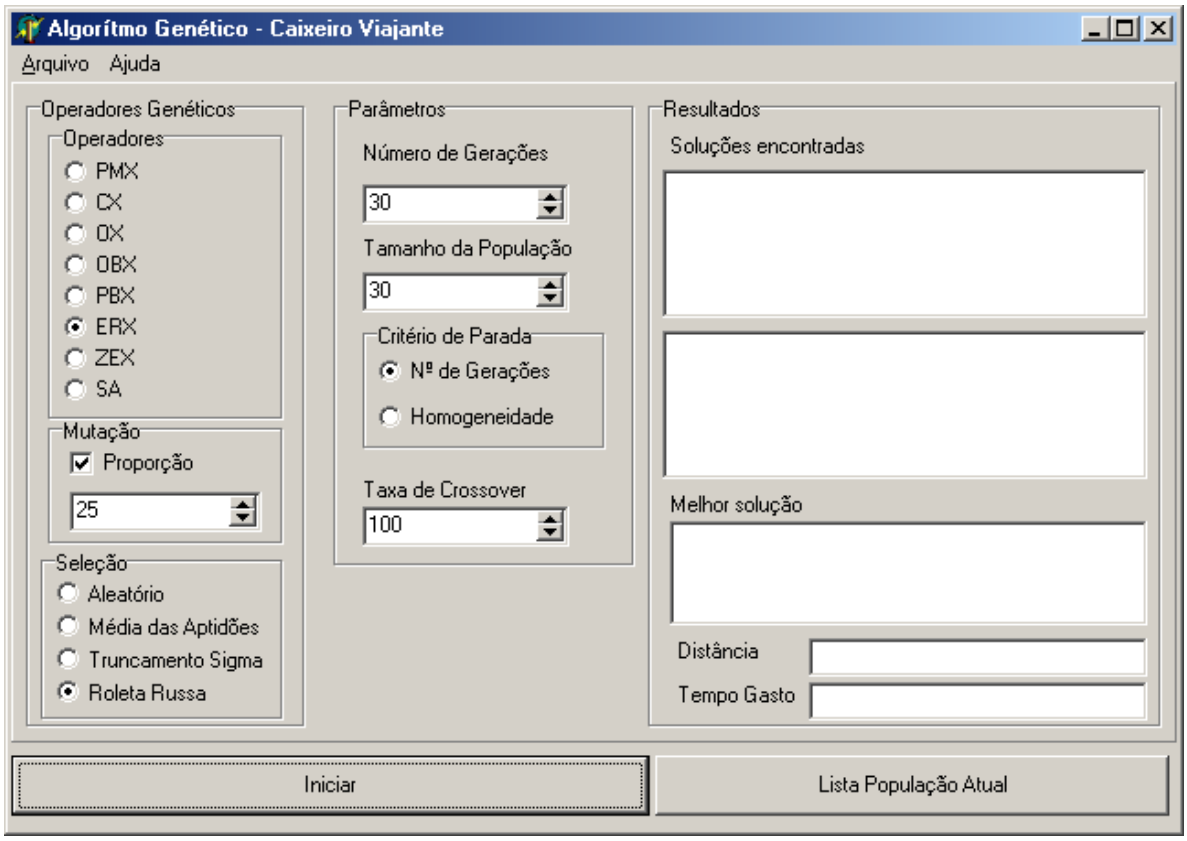


Figura A.1.2 Tela principal do programa *Simulated Annealing*, desenvolvido em Delphi 5.

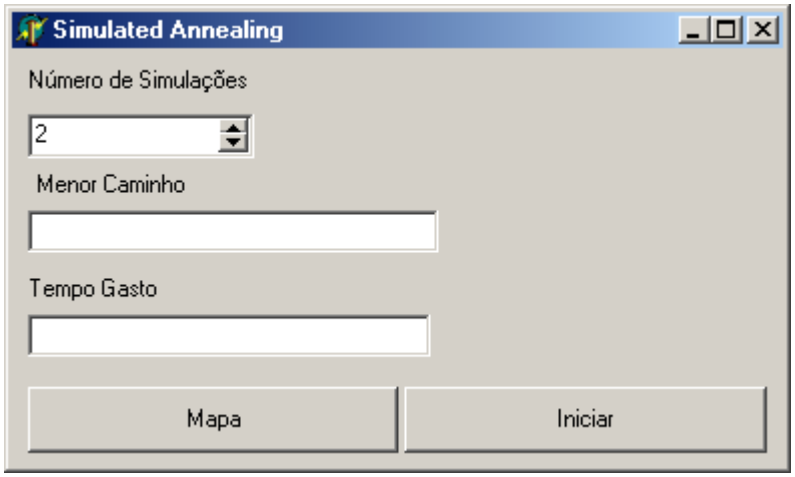
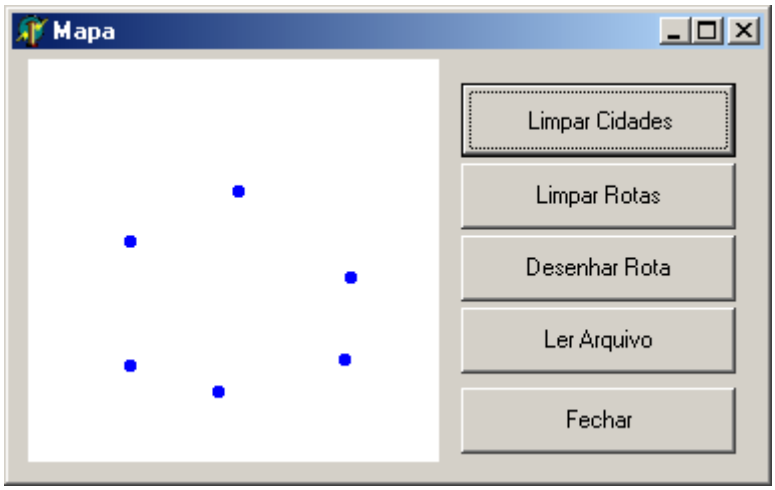


Figura A.1.3 Tela do mapa das cidades.



Anexo 2 – Resultados Obtidos

A seguir são apresentados os resultados obtidos pelos operadores descritos em **4.5** para um conjunto de cidades representadas na tela através de pontos, no sistema desenvolvido.

Tabela A2.1 Resultado do PCV utilizando o mapa com 20 cidades.

Operador	NC	NG	TP	TG	MS	MMS
OX	20	30	30	00:00,1	1123,78015	1235,27844
ERX	20	30	30	00:00,1	1053,5087	1147,82769
ZEX	20	30	30	00:00,3	1175,79041	1435,13037
S.A	20	30	30	00:03,8	821,173671	831,471468
Simulated Annealing	20	2		00:03,9	1209,39031	1287,47221

Figura A2.1 Mapa com 20 cidades.

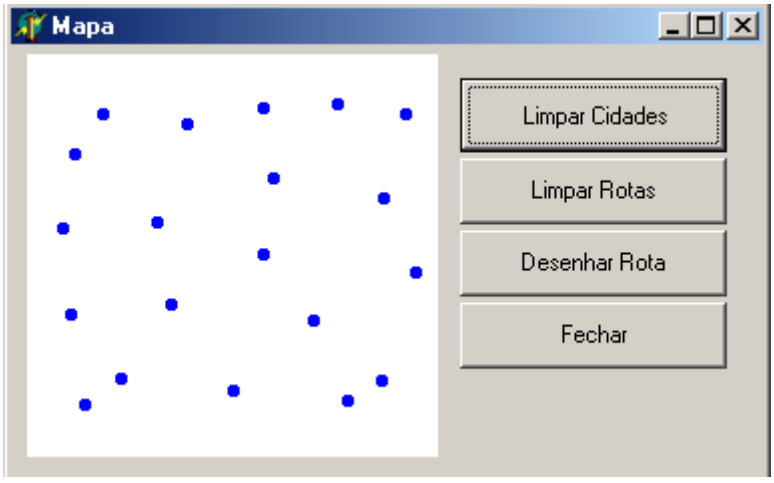


Figura A2.2 Melhor solução para o problema com 20 cidades.

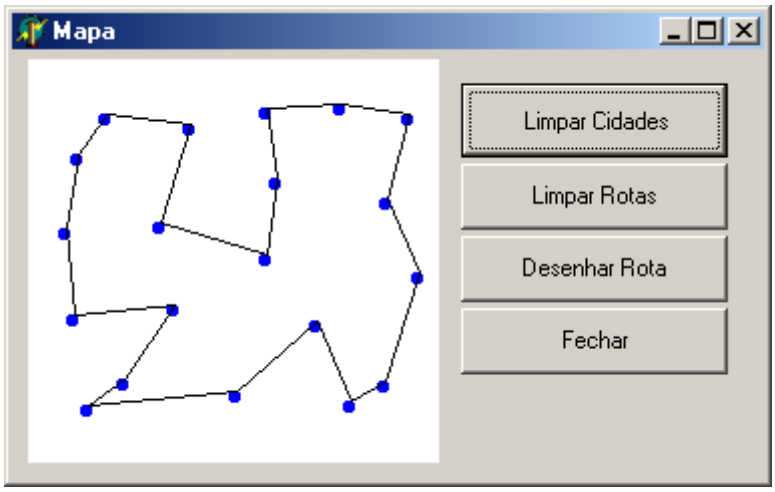


Tabela A2.2 Resultado do PCV utilizando o mapa com 30 cidades.

Operador	NC	NG	TP	TG	MS	MMS
OX	30	30	30	00:00,1	1614,20013	1824,28148
ERX	30	30	30	00:00,1	1509,40469	1726,48101
ZEX	30	30	30	00:00,3	1556,43001	1997,13611
S.A	30	30	30	00:07,4	1017,60464	1061,30195
Simulated Annealing	30	2		00:05,9	1717,81265	1842,37052

Figura A2.3 Mapa com 30 cidades.

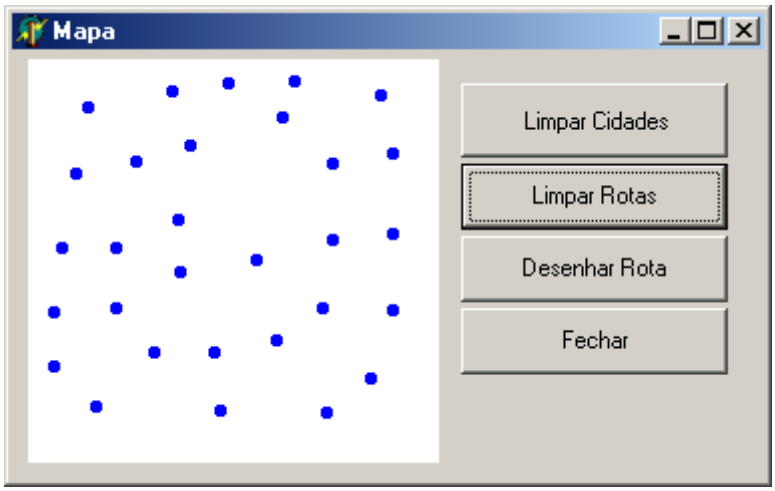


Figura A2.4 Melhor solução para o problema com 30 cidades.

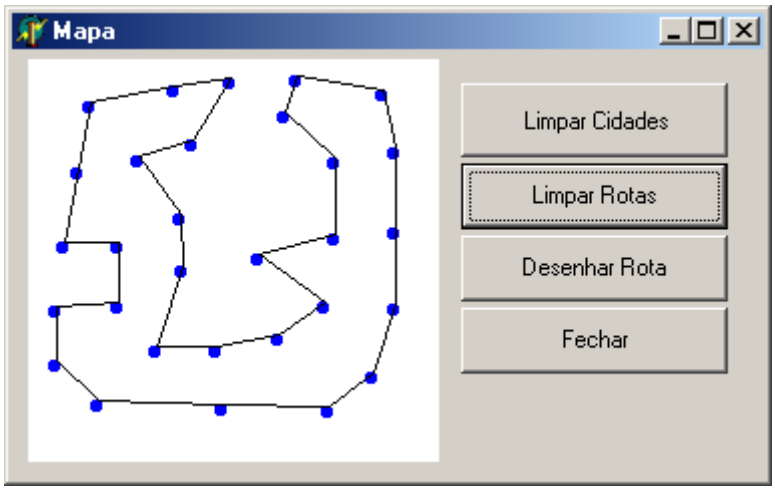


Tabela A2.3 Resultado do PCV utilizando o mapa com 40 cidades.

Operador	NC	NG	TP	TG	MS	MMS
OX	40	30	30	00:00,1	2116,64407	2600,05737
ERX	40	30	30	00:00,2	2342,05274	2453,7847
ZEX	40	30	30	00:00,4	2511,22286	2775,48817
S.A	40	30	30	00:14,7	1142,63679	1202,80787
Simulated Annealing	40	2		00:08,9	2380,93505	2454,93288

Figura A2.5 Mapa com 40 cidades.

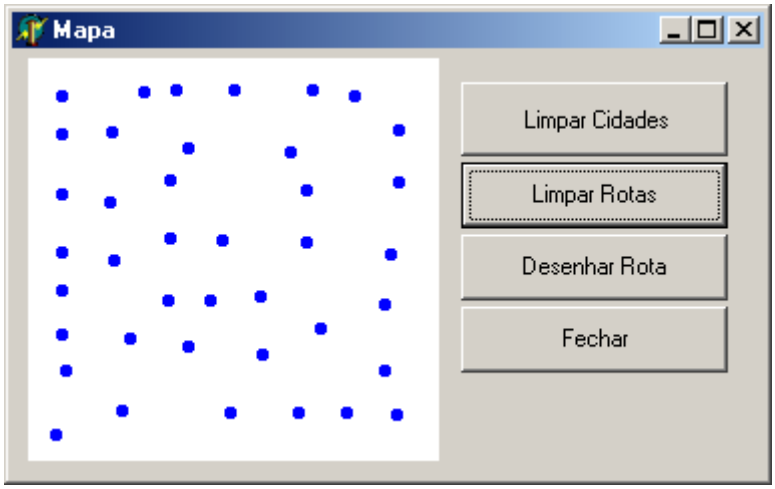
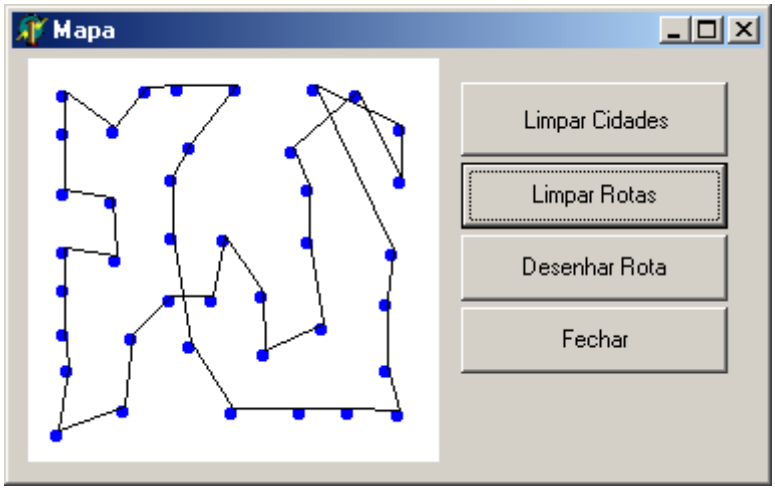


Figura A2.6 Melhor solução para o problema com 40 cidades.



Anexo 3 – Código

Procedimentos utilizados na implementação do método proposto.

Obs: Linguagem utilizada: Pascal em ambiente Delphi.

A Unit Umapa1 contém o procedimento "Distancia (cid1, cid2)" utilizado para calcular a distância entre duas cidades.

```
unit UnitAGSA;

interface

uses
  UMapa1, SysUtils, UnitDeclara;

  function tamanho(t: cromossomo): double;
  procedure gera(n: integer; var r1, r2: integer);
  procedure seleciona(n: integer; var t: cromossomo; var si, sj: integer; var dE: single);
  procedure troca(var tt: cromossomo; i, j: integer);
  function aceita(dE, temp: single): boolean;
  procedure busca(n: integer; var t: cromossomo; temp: double; tentativas, trocas:
integer);
  procedure anneal(n: integer; var t: cromossomo; Tmax, alfa: double;
                passos, tentativas, changes: integer);
  procedure crossoverAGSA(numCid: integer; pai1, pai2: cromossomo; var filho1,
filho2: cromossomo);

implementation

function tamanho(t: cromossomo): double;
var i: integer;
    soma: double;
begin
  soma := formMapa.distancia(t[high(t)], t[0]);
  for i := 0 to high(t) - 1 do soma := soma + formMapa.distancia(t[i], t[i + 1]);
  tamanho := soma;
end;{tamanho}

procedure gera(n: integer; var r1, r2: integer);
begin
```

```
  r1 := random(n);
  r2 := random(n);
end;

procedure seleciona(n: integer; var t: cromossomo; var si, sj: integer; var dE: single);
var ts: cromossomo;
begin
  gera(high(t), si, sj);
  ts := Copy(t, 0, High(t) + 1);
  troca(ts, si, sj);
  dE := tamanho(ts) - tamanho(t)
end; { seleciona }

procedure troca(var tt: cromossomo; i, j: integer);
var aux: alelo;
begin
  aux := tt[i];
  tt[i] := tt[j];
  tt[j] := aux;
end; { troca }

function aceita(dE, temp: single): boolean;
begin
  if (dE > 0.0) then aceita := exp(-dE / temp) > random else aceita := true;
end; { aceita }

procedure busca(n: integer; var t: cromossomo; temp: double; tentativas, trocas:
integer);
var i, j, ntent, ntrocas: integer;
   dE: single;
begin
 ntent := 0; ntrocas := 0;
 while (ntent < tentativas) and (ntrocas < trocas) do
   begin
     seleciona(n, t, i, j, dE);
     if (aceita(dE, temp)) then begin
       troca(t, i, j);
       ntrocas := ntrocas + 1;
     end;
     ntent := ntent + 1;
 end;
end; { busca }

procedure anneal(n: integer; var t: cromossomo; Tmax, alfa: double; passos, tentativas,
changes: integer );
var temp: double;
   k:  integer;
begin
```

```
  temp := Tmax;
  for k := 1 to passos do begin
    busca(n, t, temp, tentativas, changes);
    temp := temp * alfa;
  end;
end; { anneal }

procedure inicializa(ti: cromossomo; ns, n: integer);
var s: integer;
begin
  for s := 1 to ns do begin
//1ª    anneal(n, t, sqrt(n), 0.95, trunc(20 * ln(n)), 100 * n, 10 * n);
//2ª    anneal(n, ti, sqrt(n), 0.95, trunc(2 * ln(n)), 5 * n, n div 2);
    anneal(n, ti, sqrt(n), 0.90, trunc(2 * ln(n)), 2 * n, n div 2);
  end;
end;

function existe(et: cromossomo; cidade: alelo; numcid: integer): boolean;
var i: integer;
    e: boolean;
begin
  e := false;
  for i := 0 to high(et) do if et[i] = cidade then e := true;
  existe := e;
end;

procedure crossoverAGSA(numCid: integer; pai1, pai2: cromossomo; var filho1, filho2:
cromossomo);
var t11, t12, t21, t22: cromossomo;
    i, pos: integer;
begin
  pos := random(high(pai1) - 2) + 1;
  t11 := Copy(pai1, 0, pos);
  t21 := Copy(pai2, pos, high(pai2));
  SetLength(t12, high(pai1) - high(t21));
  SetLength(t22, high(pai1) + 1 - pos);
  pos := 0;
  for i := 0 to numcid - 1 do begin
    if not existe(t11, pai1[i], numcid) then begin
      t22[pos] := pai1[i];
      pos := pos + 1;
    end;
  end;
  pos := 0;
  for i := 0 to numcid - 1 do begin
    if not existe(t21, pai2[i], numcid) then begin
      t12[pos] := pai2[i];
      pos := pos + 1;
```

```
    end;
  end;
  inicializa(t12, 1, numcid);
  inicializa(t22, 1, numcid);
  for i := 0 to high(t11) do filho1[i] := t11[i];
  for i := 0 to high(t22) do filho1[i + high(t11) + 1] := t22[i];
  for i := 0 to high(t21) do filho2[i] := t21[i];
  for i := 0 to high(t12) do filho2[i + high(t21) + 1] := t12[i];
end;
end.
```

Referências

[BRS95] Brown, D.E. and W.T.Scherer, Intelligence Scheduling Systems, Kluwer Academic Publishers. (1995).

[BIO00] Home Page Instituto Aqualung, Biodiversidade – http://www.institutoaqualung.com.br /info_biodiversidade25.html. (2000)

[CER85] Cerny, V., *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, Journal of Optimisation Theory and Applications, 45, p. 41-51. (1985)

[C&Z00] Castro Leandro, Zuben Fernando Von, Hill Climbing e Simulated Annealing, DCA/FEEC/Unicamp, Campinas, (2000).

[COS00] Costa, Luciano Cajado, Teoria das Filas, acessado em: http://www.decom.ufop.br/prof/rduarte/CIC271/TeoriaFilas_Cajado.pdf, (2000).

[DAR53] Darwin, C., The Origin of Species, New York, Mentor Books, (1953)

[DAV85] Davis, L., Applying adaptive algorithms to epistactic domains, in: Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'85), Los Angeles, CA, 162-164. (1985)

[DAV93] Davis, M. H. A., Markov Models and Optimization, Monoghaphs on Statistics and Applied Probabylity, Chapman & Hall, (1993)

[DFJ54] G Dantzig, R. Fulkerson, and S. Johnson, Solution of a large-scale traveling-salesman problem. *Operations Research*, 2, (1954)

[GLO95] Glover, F., Tabu Search Fundamentals and uses, University of Colorado at Boulder (1995)

[GLO96] Glover, F., Tabu Search and Adaptive memory programming: Advances, applications and challenges, Interfaces in Computer Science and Oper. Res., Kluwer Academic Publ., p. 1-75. (1996)

[GOK95] Goonatilake, S. and S. Khebbal, Intelligence Hybrid Systems, John Wiley & Sons (1995)

[GOL89] Goldberg, D.E., *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, (1989)

[GRE85] Grefenstette, J., Gopal, R., Rosmaita, B.J. and Gucht, D.V., *Genetic algorithms for the traveling salesman problem*, in: Proc. 1st Int. Conf. Genetic Algorithms (ICGA'85) Carnegie-Mellon University, Pittsburg, PA, 160-168. (1985).

[GSI99] GSI - Grupo de Sistemas Inteligentes. Algoritmos Genéticos (http://www.din.uem.br/ia), (1999).

[HOL93] Holland, J. H., *Adaptation in Natural and Artificial Systems*, Cambridge, MIT Press, p. 211. (1993)

[LIN93] LIN, T. T., KAO, C. Y., HSU, C. C., *Applying the Genetic Approach to Simulated Annealing in Solving some NP-Hard Problems*, IEEE Transactions on Systems, Man and Cybernetics, v. 23, n. 6, p. 1752-1767. (1993)

[JFB99] Júlio Francisco Barros Neto, Análise de Desempenho dos Operadores Genéticos Aplicados ao Problema do Caixeiro Viajante – http://po1.pep.ufrj.br/~jfbarros

[JMJ99] José Mazzucco Junior, Uma Abordagem Híbrida do Problema da Programação da Produção Através dos Algoritmos *Simulated Annealing* e Genético, (1999)

[KIV83] Kirkpatrick, S., Gelatt, Jr. C.D., and Vecchi, M.P., *Optimization by Simulated Annealing*, Science 220, p. 671-680. (1983)

[KOZ92] J. R. Koza, *Genetic Programming*. MIT Press, (1992).

[LEC04] Lecheta, Edson Martins, Algoritmos Genéticos para Planejamento em Inteligência Artificial, Curitiba, 2004.

[MET53] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., *Equation of State Calculations by Fast Computing Machines*, Journal of Chemical Physics 21, p. 1087-1092. (1953)

[MIC92] Zbigniew Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer (1992).

[MOL92] Molina, Maria José T. Teoria Geral Da Evolução Condicionada da Vida, livro on-line disponível em http://www.molwick.com. (1992)

[OCH91] Ochi, Luiz Satoru, Algoritmos Genéticos: Origem e Evolução – http://www.pgcc.uff.br/~satoru/artigos.html

[OCH96] Ochi, Luiz Satoru, Rabelo, Patricia, A Evolutionary Heuristic Based On Genetic Agorithm For The Clustered Traveling Salesman Problem (1996).

[OCH97] Ochi, Luis Satoru, L. M. A. Drummond and R. M. V. Figueiredo, Design and Implementation of a Parallel Genetic Algorithm for the Traveling Purchaser Problem, Proceedings of the 1997 ACM Symposium on Applied Computing, San Jose, California, USA. Editors: B.Bryant, J.Carroll,

D.Oppenheim, J.Hightower and K.M.George,The Assoc. for Comp. Machinery, Inc., pp. 257-263, (1997).

[OOB00] Oliveira, Osmar De B. J., Otimização De Horários Em Instituições De Ensino Superior Através De Algoritmos Genéticos, (2000).

[OOJ97] Osmar de Oliveira Braz Jr, Oscar Ciro Lopez, João Bosco da Mota Alves, Agentes para auxiliar nas condições de convergência e definição de intervalos ideais para parâmetros em Algoritmos Genéticos. (1997)

[PER99] Pereira, Edenir Rodrigues Filho, Sistemas Mecanizados Acoplados A Forno De Microondas Para A Mineralização Em Linha De Amostras De Interesse Alimentício: Determinação De Ferro E Cobalto, Campinas, (1999).

[ROT95] Rochat, Y. and E. Taillard, Probabilistic Diversification and Intensification in Local Search for Vehicle Routing, Centre de Recherche sur les Transports, Universite de Montreal (To appear in Journal of Heuristics) (1995).

[RWE90] R. W. Eglese, *Simulated Annealing: A tool for Operacional Reseach*, European Journal of Operacional Research 46, p. 271-281, (1990).

[R&O97] Rabelo, P.G.; Ochi, L.S.; Maculan, N. (1997), A New Genetic Algorithm for the Clustered Travelling Salesman Problem, in Proceedings of the II Meteheuristic International Conference (II MIC), Sophia - Antipolis, France, (1997).

[R&P99] Ribeiro R. A., Pires F. M., Fuzzy Linear Programming via Simulated Annealing, Kybernetica, Vol. 35, N.1, 57-67, (1999).

[SAH88] Sasaki, G. H., and Hajek, B., *The time complexity of maximum matching by simullated annealing*, Journal of the ACM 35, p. 387-403, (1988).

[S&M92] Silvia, E. S., Muntz, R. R., Métodos Computacionais de Solução de Cadeias de Markov:Aplicações a Sistemas de Computação e Comunicação, VIII Escola de Computação, Gramado, (1992).

[S&W99] Sheble, G. B; Walters, D. C. Genetic algorithm solution of economic dispatch with valve point loading. IEEE Trans. on PS., Vol. 8, No. 3, p. 1325-1332. (1993).

[SOA97] Soares, Gustavo Luís. Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações, jun (1997).

[TAN89] Tanese, R., Distributed Genetic Algorithms for Function Optimization, Ph. D. Dissertation, University of Michigan, (1989).

[TAN95] Tanomaru, J, Staff Scheduling by a Genetic Algorithms with Heuristic Operators. Chap 820. In: 1995 Ieee International Conference on systems, Man and cybernetics, Vol 1-5. (1995).

[TSPLI] Home Page TSPLIB. Library of sample instances for the Traveling Salesman Problem, http://softlib.rice.edu/softlib/tsplib/