Josué Pereira de Castro

Um algoritmo Evolucionário Para Geração de Planos de Rotas

Dissertação submetida ao curso de Pós Graduação em Engenharia de Produção E Sistemas no Centro Tecnológico da Universidade Federal de Santa Catarina, como requisito parcial para obtenção do grau de Mestre em Engenharia de Produção e Sistemas.
Área de Concentração: Inteligência Artificial
Orientador: Prof. Dr. Roberto Carlos S. Pacheco

Florianópolis, agosto de 1999

Josué Pereira de Castro

Um Algoritmo Evolucionário para Geração de Planos de Rotas

Florianópolis
1999

UM ALGORITMO EVOLUCIONÁRIO PARA GERAÇÃO DE PLANOS DE ROTAS

JOSUÉ PEREIRA DE CASTRO

DISSERTAÇÃO ARPOVADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA DE PRODUÇÃO, NO CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO E SISTEMAS DA UNIVERSIDADE FEDERAL DE SANTA CATARINA

Ricardo Miranda Barcia, Ph.D
Coordenador do Programa de Pós-Graduação em Engenharia de Produção e Sistemas

BANCA EXAMINADORA

Prof. Dr. Roberto Carlos dos Santos Pacheco
Prof. Orientador
Departamento de Engenharia de Produção e Sistemas - UFSC

Prof. Dr. Alejandro Martins Rodrigues
Departamento de Engenharia de Produção e Sistemas - UFSC

Profa. Dra. Lia Caetano Bastos
Departamento de Engenharia de Produção e Sistemas - UFSC

FLORIANÓPOLIS, AGOSTO DE 1999

Para alguém Especial...

Agradecimentos

A minha família pelo apoio eterno.

Ao meu orientador, pelo apoio e confiança.

Aos professores Alejandro Martins e Rosina Weber pelo apoio.

A Capes, Pelo apoio financeiro.

Aos meus amigos por me aturarem.

Sumário

Lista De Ilustrações

Resumo

Este trabalho descreve a implementação de um algoritmo genético usado para resolver o problema de planejamento de rotas, como parte integrante de um sistema híbrido genético-RBC (Raciocínio Basedo em Casos). O algoritmo é utilizado para fazer a geração automática das rotas com base nos dados de uma malha viária, armazenando as rotas geradas em uma base de casos temporária, para posterior avaliação pelo módulo RBC, o qual é responsável pela avaliação e posterior armazenamento das mesmas em sua base de casos permanente, com o objetivo de, quando forem solicitadas rotas com as mesmas características, estas não sejam geradas novamente.

Palavras Chaves: Algoritmos Genéticos, Computação Evolucionária, Raciocínio Baseado em Casos, Sistemas Híbridos

Abstract

This work describes the implementation of a genetic algorithm used to solve the route planning problem, integrating a more complex genetic-CBR (Case-Based Reasoning) hybrid system. The algorithm described here, is used to generate routes automaticaly, based on a traffic network's data, storing the generated routes into a temporary case base, for evaluation *a posteriori* by the CBR Module, which is responsible for this task and by the storaging of the routes generated in a permanent case base, for future recovery in similar situations, avoiding re-generation of the same routes.

Keywords: Genetic Algorithms, Evolutionary Computation, Case-Based Reasoning, Hybrid Systems.

1. Introdução

Este trabalho descreve a implementação de um sistema híbrido genético – RBC (raciocínio baseado em casos) aplicado à área de transportes, especificamente para o problema do planejamento de rotas. De forma mais específica a implementação do módulo genético responsável pela geração das rotas, as quais serão posteriormente analisadas e armazenadas (ou não) em uma base de casos, pelo módulo RBC que compõe o sistema. Este trabalho também apresenta um referencial teórico que serve como base para compreensão do trabalho desenvolvido e discute como os dois módulos (genético e RBC) foram integrados na construção do sistema.

1.1 Motivação da Dissertação

Um dos maiores desafios ao planejamento de rotas consiste na ponderação de todos os fatores que fazem com que uma rota seja considerada aceitável ou não. Entram neste tópico muitos fatores, como as preferências individuais, e muitas vezes subjetivas dos executores da rota, além de características restritivas sobre as condições de trafegabilidade das vias em questão (por exemplo, em algumas ruas pode não ser permitido o tráfego de caminhões pesados). A maioria dos algoritmos clássicos da teoria dos grafos trata estas preferências e restrições como um peso que é atribuído ao arco que une dois nós. A dificuldade é, então, ponderar de que forma estes pesos serão atribuídos, uma vez que agora eles não representam mais apenas a distância, mas o custo associado ao percorrimento de uma aresta de ligação entre dois nós dados. [Aho e Ullman, 1995].

Estas dificuldades na etapa de representação das restrições nos levou a pensar em uma nova forma de encarar o problema, usando uma abordagem híbrida. Algoritmos genéticos têm a capacidade de gerar rapidamente rotas alternativas, porém a maior dificuldade com esta abordagem está na definição de sua função objetivo, que deve incluir várias restrições, por vezes qualitativas, que são difíceis de quantificar. Sistemas baseados em casos, por sua vez, são excelentes ferramentas para aprendizagem e reuso do conhecimento adquirido, e podem tratar com elementos qualitativos, em vez de quantitativos [Kolodner, 1993]. O problema com esta abordagem está no fato de ela ser incapaz de gerar algum

conhecimento novo a partir do zero. A possibilidade de juntar estas duas técnicas, explorando as vantagens de cada uma, de maneira simples, nos levou a idealização deste sistema.

1.2 Objetivos da Dissertação

Como objetivos gerais deste trabalho, pretende-se:

1. Desenvolver um sistema híbrido inteligente para resolução do problema de planejamento de rotas, considerando as preferências individuais de seus usuários.
2. descrever o algoritmo genético para geração de rotas (módulo Genético), propondo uma abordagem simples para este tipo de problema, onde as dificuldades na definição da função objetivo e as suas restrições não são avaliadas durante a geração das rotas, mas são postergadas até o momento de serem avaliadas.
3. Apresentar o protótipo desenvolvido mostrando alguns exemplos de utilização do mesmo.

Os objetivos específicos deste trabalho são:

1. fazer um breve estudo das áreas de planejamento de rotas, raciocínio baseado em casos (RBC), computação evolutiva e sobre sistemas híbridos, que servirá como referencial teórico para os objetivos específicos;

2. propor uma solução híbrida para o problema do planejamento de rotas. Esta solução consiste de um algoritmo genético, o qual é responsável pela geração da rota, e um módulo de raciocínio baseado em casos, o qual é o responsável pela avaliação das rotas de acordo com critérios definidos pelo usuário do sistema. A solução apresentada aqui separa em duas fases a criação do plano de rotas: geração, e avaliação. Na geração, busca-se por caminhos alternativos viáveis, e na avaliação faz-se uma ponderação de todos os fatores a serem considerados na escolha de uma rota, visando escolher a mais adequada a um determinado perfil de usuário, ou situação em particular [Peres et al, 1997].

1.3 Justificativas

Espera-se com este trabalho contribuir com a pesquisa científica e aplicada nas áreas de transportes e de algoritmos genéticos, mostrando a potencialidade que os algoritmos genéticos têm para resolver problemas naquela área. Espera-se que este trabalho possa servir como referencial para outros trabalhos nesta área, bem como em outras áreas cujos problemas possam ser modelados de forma semelhante a modelagem mostrada aqui.

1.4 Organização da Dissertação

Este trabalho está organizado em cinco capítulos, incluindo esta introdução, da seguinte maneira:

O capítulo 2 apresenta o problema de planejamento de rotas que pretende ser resolvido com este trabalho, os problemas mais comuns de roteamento, mostrando as abordagens utilizadas nas suas soluções e a fundamentação teórica necessária.

O capítulo 3 apresenta os algoritmos evolucionários, sua fundamentação teórica e são mostradas algumas aplicações na área de roteamento.

O capítulo 4 descreve, de modo geral, o sistema híbrido proposto para a solução do problema do planejamento de rotas, e apresenta também a fundamentação teórica sobre sistemas híbridos.

O capítulo 5 apresenta as conclusões desta dissertação e as perspectivas para trabalhos futuros a serem desenvolvidos.

2. Planejamento de Rotas

2.1 Introdução

O objetivo deste capítulo é fazer uma breve revisão dos problemas clássicos da área de Planejamento de rotas, bem como das técnicas necessárias para resolver tais problemas. São apresentadas formas de representação de malhas viárias através de grafos, uma visão geral sobre roteamento de veículos, alfuns domínios comuns para aplicações de roteamento, e por fim, são mostradas técnicas para se encontrar caminhos alternativos ao caminho mínimo. Alguns sistemas desenvolvidos para resolver este tipo de problema também são analisados.

2.2 Planejamento de Rotas

O planejamento de rotas é certamente um dos problemas mais comuns em nossa vida cotidiana. A escolha de rotas é necessária em decisões como quais ruas tomar para chegar ao trabalho, a um cinema, a um restaurante, etc., passando por escolhas de roteiros turísticos, de trajetos ótimos em transporte de cargas e encomendas, até a escolha do caminho mais rápido que possibilite uma rápida ação policial. Todas estas situações caracterizam um problema comum, o qual pode ser enunciado da seguinte maneira: "dados um ponto de origem e um ponto de destino, e um grafo que represente a malha viária, deve-se encontrar o caminho de menor distância entre a origem e o destino". Este problema é bastante conhecido na teoria dos grafos, pelo nome de "problema do caminho mínimo" e existem algoritmos clássicos de buscas em grafos que resolvem este tipo de problema [Aho e Ullman, 1995].

É importante não confundir este tipo de problema com problemas de transporte, que consistem em obter um plano ótimo de distribuição em uma rede de centros produtores e consumidores [Hillier e Gerald, 1995], nem com problemas do tipo caixeiro viajante [Aho e Ullman, 1995], [Cormen, Leiserson e Rivest, 1990], [Christofides, 1985] no qual, dado um grafo completo, deve-se encontrar o menor caminho que passe por todos os nós

do grafo, formando um ciclo. Este tipo de problema surge também, geralmente, na forma de um problema de distribuição, por exemplo, de um ônibus escolar, que deve recolher as crianças em casa e levá-las de volta, ou ainda como exemplo, um caminhão distribuidor de leite, que tem que encontrar uma rota mínima que passe por todos os pontos no qual o leite deve ser distribuído. Em nosso problema, ao contrário, esta passagem por todos os nós não é obrigatória, nem desejada, o que se deseja é encontrar o menor caminho possível entre dois pontos dados.

O Planejamento de rotas se configura como uma classe de problemas, cada um com características específicas, mas também com muitos aspectos em comum. Podemos classificar os problemas de planejamento de rotas em duas categorias, com relação à rota a ser traçada:

- problemas de ciclo de comprimento mínimo e
- problemas de caminho mínimo.

Nos problemas de ciclo de comprimento mínimo enquadram-se os problemas como: o caixeiro viajante, (ciclo hamiltoniano de menor custo) o problema do carteiro chinês (ciclo Euleriano de menor custo) e problemas clássicos de distribuição, com e sem janelas de tempo(Ex.: distribuição de jornais, ônibus escolar, etc...). Neste tipo de problema, estamos interessados em achar uma rota (ciclo) que passe por todos os vértices do grafo (Ex.: caixeiro viajante), ou que passe por todas as arestas (Ex.: Carteiro chinês).

Nos problemas de caminho mínimo estamos interessados em encontrar o camínho de menor custo (distância, ou qualquer outra medida imposta ao grafo) entre dois vértices de um dado grafo.

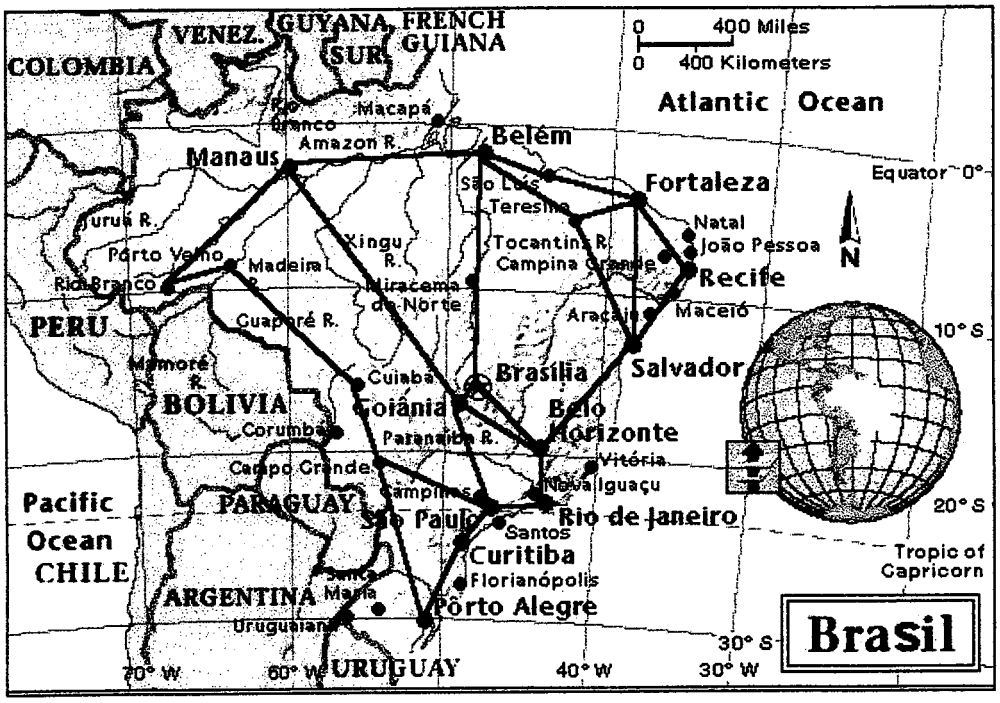


Figura 1: Grafo representando uma malha viária.
Fonte: http://janus.inf.ufsc.br:5312/temas/custo-minimo/minimo.html

A primeira abordagem utilizada para resolver este tipo de problema é o conhecido Algoritmo de Djkistra (Quadro 1: Algoritmo Djikstra.) [Cormen, Leiserson e Rivest, 1990], [Aho e Ullman, 1995], o qual, dado um grafo conexo, permite encontrar a menor distância de um ponto de origem dado a todos os outros pontos no grafo. Nesta abordagem, as arestas do grafo representam os custos associados à transição entre os dois nós pertencentes à aresta. A Figura 1 apresenta uma forma de como um grafo pode ser usado para representar uma malha viária.

Algoritmo Djikstra:

Seja r a raiz da busca

1. Inicie dr,t (a melhor estimativa para o custo mínimo de cada nodo t) e pt (o predecessor de cada vértice t no caminho de custo mínimo)
2. Enquanto houver vértice aberto:
   i. selecione o nodo x dentre os nodos ainda abertos tal que dr,x seja mínimo
   ii. feche o nodo x
   iii. atualize os custos de todos os nodos sucessores de x
   iv. atualize o predecessor x

Quadro 1: Algoritmo Djikstra.
Fonte: http://janus.inf.ufsc.br:5312/temas/custo-minimo/dijkstra.html

O Algoritmo de Dijkstra é um dos algoritmos que encontra o custo mínimo entre um vértice do grafo (a raiz da busca) para um destino. Ao calcular este custo, em realidade ele calcula o custo mínimo da raiz para todos os demais vértices do grafo. Apesar de ser bastante simples e com um bom nível de performance, este algoritmo não garante a exatidão da solução caso haja a presença de arcos com valor negativo.

A maioria das abordagens convencionais para este tipo de problema é baseada em algoritmos de busca em grafos, onde as arestas são ponderadas, onde os pesos representam o custo de percorrimento da aresta. A dificuldade, então, consiste em encontrar-se uma fórmula que seja capaz de representar todos os fatores que podem influenciar este custo. Há, também, o fato de que uma representação numérica como esta não reflete os valores qualitativos e subjetivos que são muitas vezes empregados na escolha de uma certa solução.

Outro fator a ser levado em consideração é o custo computacional. Um dos maiores problemas encontrados com os algoritmos tradicionais de buscas em grafos é o tempo de execução. Devido à sua complexidade muito elevada[1] [Hopcroft e Ullman, 1979], estes algoritmos servem apenas para grafos de pequeno porte, não servindo, portanto, para problemas maiores, como a representação da malha viária de uma cidade de grande porte. Um fator positivo a ser considerado, é que as soluções encontradas desta forma são realmente as melhores soluções, desde que se consiga representar corretamente o problema.

2.3 Uma Visão Geral Sobre Roteamento de Veículos:

O Roteamento de veículos tem sido um campo de grande sucesso dentro da pesquisa operacional nas últimas décadas. Um dos pontos chave deste sucesso tem sido a grande interação entre a pesquisa teórica e a aplicação prática: Se, de um lado, os pesquisadores mais voltados para a área acadêmica têm conseguido desenvolver novos algoritmos, ou mesmo aperfeiçoar os já existentes, por outro, vários avanços tecnológicos tanto da ciência da computação, em termos de software, quanto da engenharia elétrica, em termos de hardware, fizeram com que esta área tivesse um grande desenvolvimento nos setores comercial e industrial.

[1] Esta classe de problemas é NP-completos

Do ponto de vista da metodologia matemática subjacente, pode-se argumentar que os atuais algoritmos existentes não são tão mais sofisticados que as soluções criadas para resolver o problema clássico do caixeiro viajante. Apesar disso, o maior avanço na área do roteamento de veículos está na tentativa de capturar características do mundo real no ambiente de execução do plano de rota, tentando obter respostas úteis, mesmo sem deixar de levar em conta a sua complexidade computacional. Na maioria das aplicações de sucesso, os bons resultados alcançados devem-se a uma modelagem cuidadosa juntamente com um conjunto de heurísticas inteligentes e uma interface de usuário amigável e interativa [Assad, 1988].

Os Sistemas de rastreamento comerciais muitas vezes oferecem um amplo leque de características a serem consideradas, geralmente interligadas, e que em conjunto compõem um bom sistema de distribuição de produtos. Há no mercado atualmente vários pacotes que resolvem algumas das versões padrão do problema de roteamento de veículos:

- Problemas somente de entregas
- Problemas somente de coletas

Estes softwares são capazes de manipular uma grande variedade de restrições. Também existem pacotes que são capazes de resolver problemas combinados de coleta e entrega, inclusive com janelas de tempo. Tais Pacotes comerciais são naturalmente projetados para problemas padrão de roteamento de veículos enfrentados pela maioria das empresas que trabalham com distribuição/coleta de mercadorias.

2.3.1 Bases de Dados Geográficas e Localização de Veículos Em Roteamento

O uso de informações geográficas precisas na localização de clientes em malhas viárias tem concentrado muitas atenções dos pesquisadores nos últimos anos. A evolução atual dos meios de comunicação e a redução dos custos dos equipamentos de informática tornaram possível a implantação de equipamentos de rastreamento e de auxílio à navegação a bordo de veículos de várias categorias, desde carros de passeio até caminhões de cargas pesadas [Assad, 1988], [Bernstein e Kornhouser, 1998]. Um exemplo interessante de projeto de bancos de dados geográficos de malhas viárias é o sistema vendido pela ETAK [Assad, 1988], uma firma da Califórnia que desenvolve dispositivos de rastreamento para veículos. Este sistema fornece ao motorista um monitor onde é mostrado um mapa da estrada, no qual o veículo sempre aparece no centro deste mapa, e o seu display é constantemente

atualizado para refletir as mudanças que ocorrem de acordo com a movimentação do veículo. Este tipo de sistema de monitoração também é conhecido pelo nome de *MapMatching* [Bernstein e Kornhouser, 1998]. Este tipo de sistema caracteriza-se por fornecer, em um mapa, a posição corrente do usuário.

No caso do sistema ETAK, um pequeno monitor de 4,5 a 7 polegadas é instalado no veículo, e através dele o motorista pode visualizar uma área de ¼ de milha quadrada em um certo nível de detalhe. O motorista pode ainda dispor das funções de zoom para dentro e zoom para fora, o que lhe permite ajustar o nível de detalhe das informações visualizadas. Esta estrutura hierárquica é uma forma de selecionar o nível de detalhe a que se tem acesso, com o objetivo de não sobrecarregar o monitor com informações que não são necessárias num dado momento.

O sistema ETAK funciona de forma estática, ou seja, não está conectado a nenhum sistema de telecomunicações por rádio ou satélite que faça uma verificação em tempo real da posição do veículo. A posição do veículo é estimada usando a técnica Dead Reckoning (computação morta), que será brevemente abordada a seguir.

O sistema ETAK, em suas primeiras versões, utilizava-se de fitas cassete que continham a base de dados geográfica para uma determinada área do mapa geral, sendo esta uma grande desvantagem, pois o motorista tinha que trocar as fitas cassete sempre que passava de uma área para outra. Versões posteriores deste sistema passaram a utilizar bases de dados em CD-ROM.

2.3.2 Localização e Rastreamento de Veículos

Existem três técnicas que são consideradas as mais importantes para a localização e rastreamento de veículos, as quais serão brevemente discutidas a seguir:

2.3.2.1 Dead Reckoning (Computação Morta):

Dada uma base de dados considerada bastante precisa, é possível rastrear a posição do veículo na malha viária baseado apenas na informação detalhada sobre a sua movimentação. Sensores de navegação podem combinar informações sobre a velocidade do veículo e mudanças na direção do veículo com os dados disponíveis na malha viária. Usando

um modem entre o navegador e o rádio móvel no veículo, a posição do veículo pode ser transmitida para uma base de controle. Como mencionado anteriormente, o sistema ETAK utiliza esta abordagem.

2.3.2.2 Loran C--:

Neste sistema, pulsos de rádio de baixa freqüência são enviados de pares de estações de rádio para os veículos, que os enviam de volta à base. Isto permite ao computador localizar o veículo sobre o caminho hiperbólico baseado nos tempos entre os dois sinais recebidos pelo veículo. Sinais fornecidos por outro par de estações fornecem outro caminho e a do veículo é fixada na interseção dos dois caminhos calculados.

2.3.2.3 Sistema de determinação por Rádio e Pares de Satélites:

Nesta abordagem um sinal da base sinaliza ao veículo para enviar um sinal a um par de satélites, o qual então é enviado de volta para a base. Dado que as posições dos satélites são bem conhecidas, a posição do veículos pode ser determinada por triangulação.

2.4 Cálculo das Distâncias

Para sistemas de Planejamento de Rotas é essencial a capacidade de calcular distâncias de forma mais ou menos precisa. Uma das opções é usar as coordenadas (X, Y) em um plano como base para o cálculo da distância. Outra forma é usar um banco de dados que possua as distâncias reais medidas com um alto grau de precisão. Estas bases de dados detalhadas geralmente além de não estarem disponíveis tão facilmente, também aumentam bastante o custo computacional. Uma solução de custo mais baixo, que quase sempre acaba sendo adotada é o cálculo da distância euclidiana.

Dados dois pontos i e j e suas coordenadas (x, y), a equação abaixo fornece a distância euclidiana entre os pontos i e j:

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

É claro que a fórmula anterior produz uma certa margem de erro, pois ela aplica-se a uma superfície plana, não levando em consideração a forma esférica da terra. Ainda assim, ela representa uma boa aproximação com os mapas planares que são geralmente utilizados. Alguns mapas possuem um efeito de distorção muito acentuado, mas no caso de um sistema de roteamento urbano, este fator de distorção pode perfeitamente ser desprezado sem nenhum prejuízo à confiabilidade do sistema.

2.4.1 Conversão de Distâncias Para Tempos de Viagens:

As distâncias rodoviárias podem ser facilmente convertidas em tempos de viagens usando-se uma função de velocidade média. A determinação da velocidade média apropriada deve levar em conta os seguintes fatores:

A não-linearidade da velocidade média como uma função da distância (a velocidade é menor em seguimentos curvos, por exemplo)

Dependências de fatores externos, como horários e atrasos provocados por congestionamentos.

A velocidade máxima permitida na estrada ou rodovia e o tipo e a velocidade máxima do veículo em questão.

2.5 Alguns Domínios Comuns Para Aplicações de Roteamento

2.5.1 Coleta de lixo:

O problema da coleta de lixo está muito próximo de nossa realidade e de nosso cotidiano. Este tipo de problema também constitui uma das primeiras áreas onde foram aplicados procedimentos computadorizados para a sua solução. Existe um alto grau de flexibilidade relacionado aos dias nos quais a coleta pode ser feita. O principal fator que determina a freqüência das coletas é assegurar que a quantidade de lixo produzida pela população não se torne muito grande. Por uma série de razões institucionais e de gerenciamento, uma solução de coleta periódica é geralmente adotada, isto é, certos dias da

semana são designados para coletas em certas localidades, e outros dias são reservados para coletas em outras localidades. A maior vantagem deste tipo de solução é que o cliente sabe de antemão qual o dia da coleta e pode preparar-se para ela.

2.5.2 Distribuição de óleo combustível e gás industrial:

Estes sistemas de distribuição diferem dos sistemas mencionados acima pelo fato de serem direcionados pelo requisito de que um certo estoque de algum destes produtos seja mantido em algum tipo de armazenagem junto ao cliente. O distribuidor mantém uma frota de veículos que são despachados de um depósito central diariamente. Geralmente existem poucas restrições colocadas sobre o dia preciso no qual as entregas são feitas. A principal atenção do fornecedor é evitar que o estoque do consumidor seja totalmente gasto ou atinja níveis muito baixos. A falta de produtos em estoque diminui a satisfação do cliente, e, a longo prazo, reduzirá o número de clientes.

Nos dois exemplos acima o distribuidor não precisa conhecer o nível dos produtos do cliente e, geralmente, o cliente não precisa fazer pedidos de entrega. Cabe ao distribuidor a responsabilidade de estimar o nível de produtos do cliente e pelo planejamento das entregas.

2.5.3 Distribuição de bebidas:

Nos sistemas de distribuição de bebidas os pedidos de entrega não são conhecidos à priori; geralmente um funcionário (ou uma equipe). Nos sistemas de distribuição de bebidas os pedidos de entrega não são conhecidos a priori; geralmente um motorista visita um conjunto de clientes e lhes vende tantos produtos quanto estes necessitam. Todos os clientes visitados são clientes regulares que normalmente adquirem alguma quantidade de produtos. A questão é quanto cada cliente irá comprar. Para efeito de planejamento de rotas, pode-se fazer uma estimativa da quantidade usualmente adquirida por cada cliente, com alguma folga, e só então fazer o planejamento com base nestes dados.

2.6 Problemas de Roteamento Dinâmico de Veículos

Por roteamento dinâmico de veículos entende-se o despacho de veículos para atender a múltiplas demandas por serviços que evoluem de uma forma dinâmica (em tempo real). Estes veículos podem ser táxis, caminhões, navios, aeronaves, etc. O serviço fornecido consiste em levar um passageiro até um aeroporto, pegar ou entregar pequenas encomendas, entregar gases para clientes industriais, transportes de tropas e materiais bélicos em casos de mobilização, ou de maneira geral, satisfazer uma grande variedade de outras necessidades de transporte que tenham em si uma natureza dinâmica [Psarafits, 1988].

Para melhor compreendermos as diferenças entre o roteamento de veículos estático e dinâmico, podemos definir roteamento estático como aquele em que as entradas assumidas para o problema não mudam durante a execução do algoritmo que o resolve, nem durante a eventual execução da rota [Psarafits, 1988]. Por contraste, podemos entender por roteamento dinâmico aquele em que as entradas podem (e geralmente acontece) mudar ou serem atualizadas durante a execução do algoritmo e a eventual execução da rota. Neste tipo de algoritmo, a execução para fazer o roteamento e a execução da rota são processos concorrentes, em contraste com o modelo estático em que o processo de formação da rota precede claramente o processo de execução.

Segundo Psarafits [Psarafits, 1988], as principais diferenças entre os problemas de roteamento estático e dinâmico são, com relação ao planejamento dinâmico:

1. A dimensão do tempo é essencial.
2. A informação futura pode ser imprecisa, incompleta ou desconhecida.
3. Eventos próximos (no tempo) são mais importantes
4. Mecanismos de atualização das informações são essenciais
5. re-sequenciamento e re-atribuição de decisões devem ser garantidos
6. Tempos de computação rápidos são necessários.
7. Mecanismos de adiamento indefinido são essenciais
8. A função objetivo pode ser diferente
9. Restrições de tempo podem ser diferentes
10. Flexibilidade para variar o tamanho da rota de veículos é pequena
11. Considerações sobre enfileiramento podem se tornar importantes

Psarafits [Psarafits, 1988] discute cada um destes pontos em detalhes, apresentando também uma revisão dos principais métodos de resolução para o roteamento dinâmico. Powell [Powell, 1988] faz uma revisão das quatro abordagens metodológicas diferentes para a solução do problema de alocação dinâmica de veículos: a) Redes Determinísticas de Transportes de Cargas b) redes estocásticas/não-lineares, processos de decisão de Markov e programação estocástica. Estes métodos são comparados em termos de sua formulação da função objetivo e variáveis de decisão, o seu grau de aplicação prática e os requisitos computacionais de cada uma. Jaillet [Jaillet, 1988] Faz uma análise das variantes probabilísticas de dois problemas clássicos: o problema do roteamento de veículos probabilístico e o problema do caixeiro viajante probabilistico. Estes problemas diferem dos problemas clássicos pelo fato de que apenas um subconjunto dos clientes potenciais precisam ser visitados em alguma dada instância do problema. Este subconjunto é determinado de acordo com alguma lei de probabilidade dada. Neste artigo também são apresentadas várias propriedades interessantes do problema do caixeiro viajante e roteamento probabilístico e uma breve discussão de algumas heurísticas utilizadas para resolver o problema do caixeiro viajante probabilístico.

2.7 Alternativas para o Melhor Caminho

O coração de qualquer sistema de roteamento, planejamento de tráfego e assistentes de viagem é o seu gerador de caminhos. A maioria dos sistemas existentes hoje oferece apenas uma única solução para o usuário: o **melhor** caminho. Contudo, em muitas situações é necessário fornecer ao usuário várias alternativas de caminhos possíveis e viáveis. [Scott et al, 1997], [Scott e Bernstein, 1997].

Os algoritmos clássicos foram desenvolvidos para encontrar este **melhor** caminho através de uma rede ou grafo, onde "melhor" pode ser definido em termos de tempo, custo, distância ou qualquer combinação dos três. Muitos destes algoritmos são utilizados em sistemas de planejamento de rotas de algum tipo, e, apesar da sua eficiência, em muitas situações eles não são suficientes; não por falta de eficiência ou eficácia, mas por causa da natureza do problema. Existem situações nas quais é necessário gerar várias rotas alternativas ao melhor caminho.

Um bom exemplo deste tipo de situação é o planejamento de viagens de turismo, onde os participantes de uma excursão podem optar por vários roteiros, levando em consideração custo, tempo de viagem, atrações turísticas, etc. Outro exemplo são sistemas de navegação colocados em automóveis, os quais podem oferecer vários caminhos ao motorista, que pode escolher o caminho desejado usando seu próprio conhecimento sobre as condições de fluxo em determinado período do dia, facilitando assim ao usuário evitar congestionamentos descartando vias de fluxo muito intenso.

Um exemplo interessante deste tipo de sistema foi desenvolvido pelo Departamento de Pesquisa Operacional da Universidade de Princeton, o Projeto PULSAR – Princeton University's Large Scale Automobile Routing (Roteamento de Automóveis em Larga Escala da Universidade de Princeton), no qual várias técnicas para o planejamento de rotas foram estudadas e desenvolvidas. [Scott et al, 1997], [Bernstein e Kornhouser, 1998], [Scott, e Bernstein, 1997], [Padmos e Bernstein, 1997] discutem o problema restrito de planejamento de rotas entre uma única origem e um único destino.

2.7.1 Abordagens Existentes

Existem basicamente duas abordagens para a geração de caminhos alternativos [Scott et al., 1997]:

- Eliminação de arcos
- Achar os r-melhores caminhos

Na primeira abordagem o sistema primeiro fornece ao usuário o melhor caminho, então, caso haja necessidade de se encontrar uma nova alternativa, o usuário é questionado sobre quais os arcos desejaria que fossem excluídos da rota alternativa. A maior vantagem desta abordagem é a sua eficiência – as alternativas podem ser geradas tão facilmente quanto a solução inicial. A maior desvantagem é que o usuário pode não estar descontente com apenas um determinado arco ou arcos em particular, mas com o caminho como um todo.

A segunda abordagem encontra todos os r-melhores caminhos e os apresenta ao usuário (todos ou apenas alguns). As desvantagens deste tipo de sistema são: algoritmos para encontrar estes r-melhores caminhos geralmente são mais lentos do que aqueles que

encontram apenas o melhor caminho, e os caminhos encontrados por este método geralmente tendem a ser muito similares. O Projeto PULSAR utiliza uma técnica mista, baseada na idéia de que quando o usuário procura por caminhos alternativos, ele geralmente procura por caminhos que não tenham muitos arcos em comum, mas sem ter que especificar explicitamente quais os arcos que têm que ser diferentes. A estratégia usada pelo sistema PULSAR é a seguinte [Scott et al., 1997]:

Um caminho p é considerado *k-similar* a um caminho s se p e s têm pelo menos k arcos em comum. Se z denota o melhor caminho, então o problema consiste em encontrar o melhor caminho *k-similar* a z.

A diferença desta abordagem para as tradicionais está na inclusão das restrições $x \in \{0,1\}^n$ de forma explícita, já que ela não possui mais a propriedade da integralidade. A introdução da restrição de sobreposição de arcos dificulta bastante encontrar o menor caminho, ou seja, como problemas de caminho mínimo com apenas uma restrição tornam-se NP-Hard, o sistema PULSAR resolve este problema usando o método de relaxamento de Lagrange [Scott et al., 1997]

O Sistema PULSAR representa a sua malha viária, na forma de um grafo G, constituído de um conjunto finito de nós $N = \{1,\ldots,m\}$ e um conjunto finito de arestas (ou arcos) direcionados, $L = \{1,\ldots,n\}$. A representação computacional adotada é uma matriz de incidência nó-arco (denotada por A), cujos componentes são definidos da seguinte maneira: $a_{ij} = 1$ se arco j está direcionado para fora do nó i (saindo de i), $a_{ij} = -1$ se o arco j está direcionado para dentro do nó i (entrando em i) e $a_{ij} = 0$ caso contrário. O sistema também assume a existência de uma única origem O e um único destino D. O sistema se utiliza de um vetor b para indicar a origem e o destino, da seguinte forma:

$$b_O = 1, b_D = -1 \text{ e } b_i = 0, \quad i \in N - \{O,D\}$$

Assim, qualquer x que satisfaça

$$Ax = b$$
$$x \in \{0,1\}^n$$

corresponde a um caminho de O a D. Os custos associados ao percorrimento dos nós é dado pelo vetor c, onde $c=(c_j : j=1,\ldots n)$, sendo que o custo associado ao caminho x é dado por: $C(x)=C^T X$. Então o problema de encontrar o caminho mínimo pode ser formulado como:

$$\begin{aligned} &\min_x C^T x \\ &\textit{sujeito a} \quad Ax=b \\ &\textit{onde} \qquad\quad x \geq 0 \end{aligned}$$

A restrição $x \geq 0$ substitui a restrição $x \in \{0,1\}^n$ por que este problema tem a propriedade da integralidade. Supondo que z é uma solução para o sistema acima, dado outro caminho x, podemos observar que

$$z^T x = \sum_{i=1}^{n} z_i x_i$$

é o número de arcos que z e x tem em comum.

Desta forma, o problema de encontrar um caminho de custo mínimo que seja k-similar a z é dado por:

$$\begin{aligned} &\min_x C^T x \\ &\textit{Sujeito a} \quad Ax=b \\ &\qquad\qquad\quad z^T x < k \\ &\qquad\qquad\quad x \in \{0,1\}^n \end{aligned}$$

que é um caminho de custo mínimo que tem pelo menos k arcos em comum com o caminho z. Uma descrição mais detalhada do algoritmo usado pelo sistema PULSAR, é apresentada por Scott e Bernstein [Scott e Bernstein, 1997]. Outra característica interessante deste sistema são os estudos feitos para torná-lo disponível na internet [Padmos e Bernstein]. Há um protótipo do sistema, com características limitadas disponível na Internet no endereço: http://www.sor.princeton.edu/~dhb/PULSAR/ALTERNATIVE_PATHS/index.html (Figura 2). O sistema PULSAR é o sistema que mais se assemelha, em características, ao sistema proposto em nosso trabalho, características essas que são: a) a apresentação de caminhos alternativos, juntamente com o melhor caminho, e b) apresentação de uma solução em tempo

relativamente curto. Porém este sistema não permite a avaliação das rotas com respeito as preferências do usuário, levando em consideração apenas o fator distância.

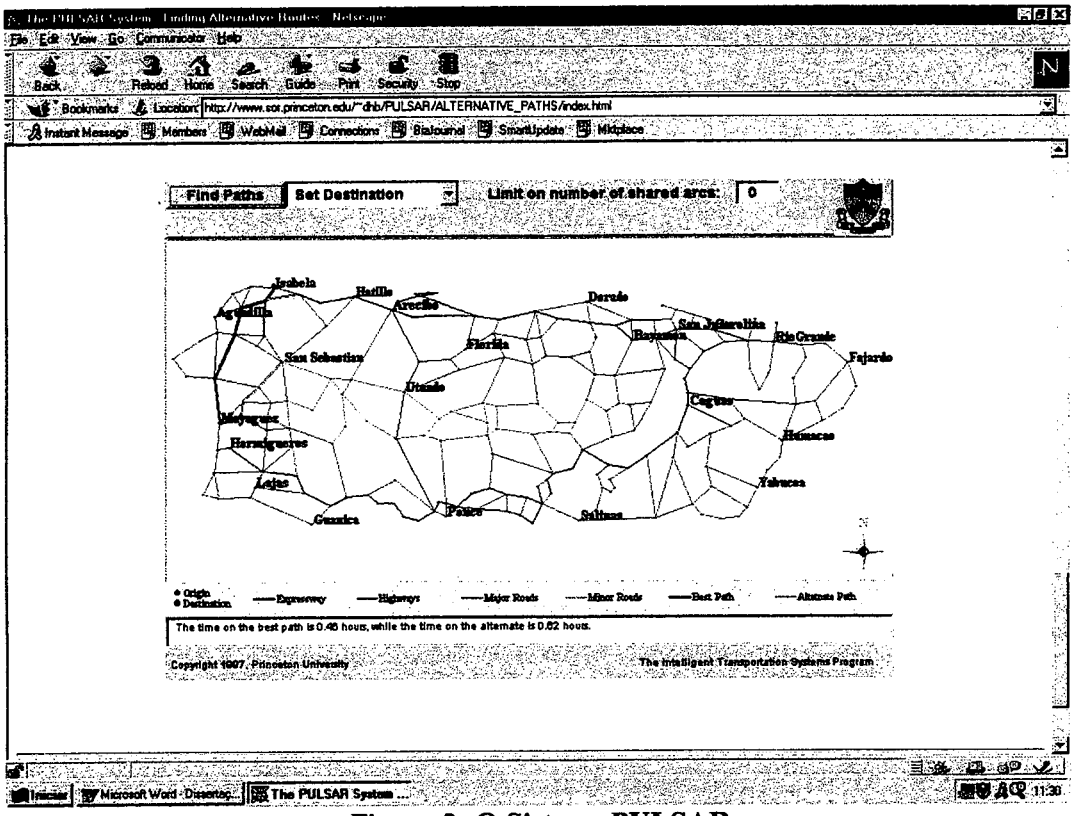


Figura 2: O Sistema PULSAR

http://www.sor.princeton.edu/~dhb/PULSAR/ALTERNATIVE_PATHS/index.html

2.8 Resumo

Este capítulo apresentou os conceitos básicos e problemas clássicos da área de planejamento de rotas, bem como as principais técnicas utilizadas para resolvê-lo. O objetivo deste capítulo é apresentar o problema, o qual estamos tentando resolver, utilizando as técnicas apresentadas no próximo capítulo, o qual apresenta os algoritmos evolucionários, raciocínio baseado em casos e sistemas híbridos.

3. Algoritmos Genéticos, Raciocínio Baseado em Casos e Sistemas Híbridos

3.1 Introdução

Este capítulo objetiva dar uma visão geral das técnicas usadas na implementação do protótipo, apresentar algumas das aplicações da computação evolucionária na área de transportes, e, finalmente, dar uma visão geral de sistemas híbridos, uma nova técnica de resolução de problemas que permite reunir as melhores qualidades de vários paradigmas diferentes de computação inteligente, na tentativa de construir um sistema mais robusto.

A seção 3.2 descreve brevemente a área da computação evolucionária, enfocando principalmente a sua sub-área de algoritmos genéticos, apresentando alguns conceitos e definições básicos desta subárea; a seção 3.3 analisa algumas das principais áreas de aplicação da computação evolucionária; a seção 3.4 descreve as principais aplicações da computação evolucionária na área de transportes; a seção 3.5 descreve brevemente e técnica de raciocínio baseado em casos; a seção 3.6 descreve a técnica de sistemas híbridos, e por fim, a seção 3.8 faz algumas considerações a respeito da integração de algoritmos genéticos e RBC.

3.2 Computação Evolucionária

O termo "*computação evolucionária*" ou "*algoritmos evolucionários*" é um termo abrangente usado para descrever uma série de técnicas computacionais que se baseiam no mecanismo de evolução[2] como o seu elemento chave. Uma grande variedade deste tipo de algoritmos foi proposta, entre eles: algoritmos genéticos, programação evolutiva, estratégia evolutiva, programação genética. Todas estas técnicas compartilham

[2] Nome dado ao processo de mudanças que ocorre em uma população com capacidade reprodutiva, na qual as capacidades de um indivíduo que o tornam mais apto a sobreviver em um determinado ambiente tem maior probabilidade de serem selecionados para reprodução e assim serem passadas aos seus descendentes (Biol.).

uma base conceitual comum, que é a simulação da evolução de estruturas individuais e/ou coletivas (espécies) através de um processo que envolve *seleção*, *mutação* e *cruzamento (reprodução)* [Spears et al., 1993]. Apesar de haver algumas diferenças entre elas, podemos dizer que todas estas técnicas são apenas variações de uma mesma tecnologia.

Algoritmos evolucionários mantém uma população de estruturas que evoluem de acordo com certas regras de seleção, e outros operadores (chamados de operadores de busca ou operadores genéticos – seleção, mutação, cruzamento, recombinação, etc.). Cada indivíduo gerado recebe uma medida da sua adequação ao meio ambiente (também chamada de função fitness) [Spears et al., 1993], [Holland, 1975], [GOLDBERG, 1989], [DAVIS, 1996].

A computação evolucionária adota uma filosofia um pouco diferente das técnicas tradicionais que se utilizam de cálculos para encontrar as melhores soluções. A computação evolucionária baseia-se na teoria da evolução das espécies, cujo princípio básico pode ser enunciado da seguinte maneira: partindo-se de um conjunto de soluções iniciais geradas aleatoriamente, realizam-se operações sobre estas soluções iniciais (operações estas que são baseadas nos fenômenos que ocorrem naturalmente na evolução das espécies, como cruzamento, mutação, etc.) busca-se fazer o aperfeiçoamento contínuo destas soluções, com o intuito de melhorá-las a cada nova iteração do algoritmo (chamada de geração). A escolha da solução ótima pode-se dar, então, de duas formas: através de uma convergência dos elementos do conjunto de soluções, ou então pela escolha da melhor solução após um número determinado de gerações.

Um fator negativo a ser levado em consideração é que as soluções oferecidas pela computação evolucionária nem sempre são as soluções ótimas, dado que todo o processo de melhoramento das soluções é aleatório, porém, orientado em direção à melhor solução. Podem surgir problemas, como a convergência prematura para um resultado não-ótimo (um valor de mínimo local, por exemplo), ou o conjunto de soluções iniciais pode estar de tal forma distribuído que a sua evolução em direção à melhor solução pode levar um tempo considerável.

3.2.1 Definições e Conceitos Básicos

3.2.1.1 Algoritmos Genéticos

Algoritmos genéticos são um tipo de algoritmo de busca que se utiliza do paradigma genético/evolucionário [Holland, 1975]. Algoritmos Genéticos foram criados com o intuito de imitar alguns dos processos observados na evolução natural das espécies. Os mecanismos que realizam esta evolução ainda não estão completamente bem compreendidos, mas algumas de suas características já são bastante bem compreendidas e aceitas. A evolução acontece nos cromossomos, que são os elementos orgânicos responsáveis pela codificação genética dos seres vivos [Davis, 1996]. As características e fenômenos específicos desta codificação ainda são objetos de muitas pesquisas nos dias de hoje. Segundo [Davis, 1996] as principais características gerais da teoria evolucionária que já são amplamente aceitas são:

a) a seleção natural é um processo que atua sobre os cromossomos, e, portanto, sobre os seres vivos que eles codificam;

b) a seleção natural é o elo entre cromossomos e a performance das suas estruturas decodificadas. O processo de seleção natural faz com que os cromossomos que codificam estruturas bem sucedidas se reproduzam mais vezes e com maior probabilidade que as estruturas mal sucedidas;

c) o processo de reprodução é o ponto onde a evolução acontece. Mutações podem provocar mudanças nos cromossomos dos filhos, fazendo com que eles sejam diferentes dos padrões genéticos dos seus pais, e processos de recombinação podem criar diferentes cromossomos para os filhos, pela combinação dos cromossomos dos pais;

d) a evolução biológica não tem memória. Tudo o que se sabe sobre como produzir indivíduos bem adaptados ao seu meio ambiente está contido no seu genoma – o conjunto de cromossomos carregados pelos indivíduos da população atual – e na estrutura dos cromossomos decodificados.

No começo dos anos 70, John Holland, quando pesquisava as características da evolução natural, acreditava que, se estas características fossem adequadamente incorporadas a algoritmos computacionais, poderia produzir uma técnica para solucionar

problemas difíceis da mesma forma que a natureza fazia para resolver os seus problemas, ou seja, usando a evolução. Acreditando nisto ele deu início a uma pesquisa sobre algoritmos que manipulavam strings de 0's e 1's, a qual ele deu o nome de **cromossomos**. Os algoritmos de Holland realizavam a evolução simulada de populações destes cromossomos. Desta forma, imitando a natureza, seus algoritmos resolviam muito bem o problema de encontrar bons cromossomos através da manipulação do material contido nos cromossomos.

Outro ponto interessante nas técnicas desenvolvidas por Holland é que, assim como a natureza, estes cromossomos não têm conhecimento nenhum sobre o tipo de problema que eles estão resolvendo. A única informação que eles dispunham era uma avaliação de cada cromossomo produzido. O objetivo desta avaliação era verificar quais os cromossomos que estavam mais adaptados e, com base nisto, aumentar as suas chances de serem selecionados para a reprodução.

Quando Holland começou os seus estudos sobre estes algoritmos, eles ainda não tinham um nome. Foi apenas quando esta técnica começou a demonstrar o seu potencial que houve a necessidade de se dar um nome adequado e significativo à ela. Como uma referência às suas origens na biologia, Holland os batizou de **Algoritmos Genéticos**. De maneira geral, um algoritmo genético pode ser brevemente descrito da forma mostrada no Quadro 2 [Davis, 1996]:

Inicializar uma população de cromossomos.
Avaliar cada cromossomo na população.
Criar novos cromossomos pela combinação dos cromossomos atuais; aplicar mutação e recombinação sobre os cromossomos dos pais durante a combinação.
Deletar membros da população antiga para dar espaço à nova população de cromossomos.
Avaliar os novos cromossomos e inseri-los na população.

Quadro 2: Um Algoritmo Genético Simples [Davis, 1996]

A técnica usada para codificar as soluções varia de problema para problema, e de algoritmo genético para algoritmo genético. A técnica usada no trabalho de Holland, e até hoje a mais usada, consistia em usar strings de bits, mas com o passar do tempo outros pesquisadores apresentaram outras formas de codificação. Pode-se afirmar com certeza que nenhuma forma de codificação (representação) funcionaria igualmente bem em todas as situações. Para cada caso deve-se fazer uma escolha cuidadosa do tipo de codificação a ser utilizado, pois uma codificação ruim pode não levar ao resultado esperado.

O elemento de ligação entre o algoritmo genético e o problema a ser resolvido é a *função de avaliação* (ou função fitness). A função de avaliação toma como entrada um cromossomo, e retorna um número (ou lista de números) que representam a medida de performance do cromossomo com relação ao problema a ser resolvido. Esta função desempenha, no algoritmo genético, o mesmo papel desempenhado pelo meio ambiente na teoria da evolução natural das espécies. O Quadro 2 descreve de forma geral o funcionamento de um algoritmo genético; esta, contudo, não é a única forma de implementação existente e possível, vários pesquisadores têm apresentado implementações diferentes, adaptadas a problemas diferentes e com formas diferentes de codificação do cromossomo.

3.2.1.2 Seleção, Mutação e Cruzamento

Estes três elementos estão intimamente relacionados no modelo básico de um algoritmo genético, e os três em conjunto fazem a evolução da população acontecer. A finalidade da seleção em um algoritmo genético é escolher os elementos da população que devem se reproduzir. Esta escolha deve ser feita de tal forma que dê maior chance de reprodução aos membros da população que sejam mais adaptados ao meio ambiente (aqueles que apresentam um valor da função fitness mais elevado). Existem várias formas de se fazer a seleção, entre elas, a mais conhecida e utilizadas é a roleta (ou algoritmo de Monte Carlo) [Goldberg, 1989], [Davis, 1996]. Outros esquemas têm sido propostos por vários pesquisadores. Entre eles podemos citar [Davis, 1996]: reprodução proporcional, seleção por ranking, e seleção por torneio. Entre estes mecanismos, detalharemos aqui um pouco mais o método da roleta:

- **Seleção Por Roleta:**

O Quadro 3 [Davis, 1996] mostra o algoritmo básico para o esquema de seleção por roleta. O algoritmo mostrado no Quadro 3 tem o efeito de retornar sempre um elemento escolhido aleatoriamente para reproduzir, porém é difícil garantir que a melhor escolha será sempre feita, ou seja, não há garantias de que apenas os melhores elementos da população serão selecionados. De fato, é possível que os piores elementos de toda a população sejam selecionados a cada execução deste algoritmo. Este fato não chega a ser ruim, pois o fato de que nem sempre os melhores elementos serão selecionados garante a variedade genética da população, o que pode ajudar a impedir fenômenos como a convergência prematura [Goldberg, 1989].

Este algoritmo tem o nome de roleta porque pode ser encarado como uma roleta dividida em fatias, sendo cada fatia atribuída proporcionalmente, de acordo com o valor da função fitness de cada cromossomo, a cada elemento da população. Uma das vantagens desta técnica é que ela dá chance a todos os elementos de serem selecionados para a reprodução. É evidente que, como já foi dito, de acordo com o valor da sua função fitness, um elemento pode ter maior ou menor chance de ser selecionado. [Goldberg e Deb, 1991].

Algoritmo de seleção por roleta
Some os valores das funções fitness de todos os membros da população; chame este resultado de fitness total.
Gere um número randômico n entre 0 e o fitness total.
Retorne o primeiro membro da população precedente que seja maior ou igual a n

Quadro 3: Algoritmo de seleção por Roleta

- **Cruzamento e Mutação:**

Estas duas características, por estarem intimamente relacionadas, serão expostas brevemente aqui, de maneira conjunta. O objetivo final de ambas é fazer com que os cromossomos criados durante o processo de reprodução sejam diferentes dos cromossomos dos pais. O operador de cruzamento é responsável por combinar os cromossomos dos pais na criação dos cromossomos filhos, e o operador de mutação é

responsável pela introdução de pequenas mudanças aleatórias nos cromossomos dos filhos. Vários tipos de operadores de cruzamento e mutação foram desenvolvidos por vários pesquisadores, alguns adequados a um tipo específico de codificação dos cromossomos, outros com inteção de serem mais genéricos. Mencionaremos aqui apenas os mais comumente utilizados.

- **Mutação de bit:**

A mutação de bit é aplicável em todas as formas binárias de representação de cromossomos. O processo da mutação de bit é bem simples, e normalmente é realizado da seguinte maneira: Dada uma certa probabilidade de mutação (normalmente muito baixa e determinada de forma empírica) para cada bit na string do cromossomo é avaliado para saber se este bit deverá sofrer mutação; caso este bit deva sofrer mutação, o seu valor é simplesmente trocado por um valor determinado aleatoriamente entre os valores que podem ser assumidos pelo cromossomo.

Cromossomo Anterior	Nºs Aleatórios				Novo bit	Cromossomo novo
1010	0,801	0,102	0,266	0,373	-	1010
1100	0,120	0,096	**0,005**	0,840	0	1100
0010	0,760	0,473	0,894	**0,001**	1	**0011**

Tabela 1: Exemplos de mutação de bit

A Tabela 1 mostra 3 cromossomos de comprimento 4 e os números aleatórios gerados para cada um dos bits no cromossomo, os novos bits que demonstram as possibilidades de mutação e o resultado final após a mutação.

- **Cruzamento em um Ponto**

É a técnica de cruzamento mais simples e a mais utilizada. Consiste em dividir os cromossomos selecionados em ponto de sua cadeia, ponto este escolhido aleatoriamente. Após isso, copia-se para os novos cromossomos uma parte de cada um dos cromossomos selecionados (cromossomos pais), formando assim os novos cromossomos (cromossomos filhos). Nas implementações mais tradicionais, é comum um par de cromossomos selecionados dar origem a dois filhos, mas este não é um fator restritivo. A

princípio, pode-se criar qualquer quantidade de filhos, desde que, é claro, o número de alelos permita o número desejado de combinações diferentes.

- **Cruzamento de múltiplos pontos**

Um pouco menos utilizado que o cruzamento de um ponto, esta técnica divide o cromossomo em vários pontos, e recombina-os para formar os filhos. Esta técnica se assemelha mais ao processo como ocorrido na vida real e tem a vantagem de assegurar uma variedade genética maior.

O cruzamento é uma etapa extremamante importante dos algoritmos genéticos. Segundo [Davis 1996], "muitos autores acreditam que se retirarmos o operador de cruzamento de um algoritmo genético, o resultado não é mais um algoritmo genético". [Mühlenbein, 1991] faz uma distinção entre algoritmos evolucionários e algoritmos genéticos. Em suas palavras: "em termos biológicos, algoritmos evolucionários modelam a evolução natural pela reprodução assexuada, com mutação e seleção. Algoritmos de busca que modelam a reprodução sexuada são chamados algoritmos genéticos." De fato, vários pesquisadores acreditam que o uso do operador de cruzamento distingue os algoritmos genéticos de todos os outros algoritmos de otimização.

Certos algoritmos de otimização também trabalham com populações de indivíduos, e aplicam operações semelhantes à mutação, preservando apenas os melhores (programação dinâmica). Alguns autores também acreditam que a performance de técnicas como estas são tão boas quanto a dos algoritmos genéticos, pois os algoritmos genéticos usam o operador de cruzamento e outras técnicas não. Contudo, tais afirmações precisam ainda ser suportadas por testes empíricos. Ainda segundo [Davis, 1996] muitos pesquisadores afirmam que a performance de um algoritmo genético é degradada na aplicação a vários problemas práticos. Ele também afirma que é necessário mostrar que quando o cruzamento é adicionado a outros métodos a performance destes sofra algum ganho considerável.

Outro fato que depõe a favor do operador de cruzamento é que a noção de cruzamento está intimamente relacionada ao esquema usado para a reprodução pela maioria das espécies existentes, ou seja, a reprodução sexuada. Algumas espécies, porém, utilizam-se da reprodução assexuada, onde, portanto, só há a necessidade de seleção de um

único pai. Nestes casos, o filho gerado pode ser uma clone, uma cópia genética expressa do pai, salvo nos casos em que ocorram mutações.

Se ainda compararmos os dois esquemas de reprodução, veremos que no esquema de reprodução sexuada é necessário haver mais de um tipo de indivíduo, estes indivíduos devem ter diferenças significativas em alguns aspectos, e devem desprender uma boa parcela de seu tempo e energia para encontrar um parceiro certo para a reprodução. Isto representa um custo a mais para o indivíduo/algoritmo. Porém, como o esquema de reprodução sexuada parece ter vencido esta guerra, pode-se concluir que este talvez seja um preço pequeno a pagar, comparado aos benefícios que ele traz consigo.

Um desses benefícios é que a reprodução sexuada permite a combinação rápida de características benéficas, o que não é possível no caso da reprodução assexuada. Mas nem tudo depõe contra a reprodução assexuada. Uma das formas de vida que mais demonstra possuir uma alta capacidade de adaptação reproduz-se por esta forma, os vírus. O alto poder de adaptação dos vírus vem do fato de que eles são altamente mutáveis, o que pode nos levar a concluir, que a capacidade de sofrer mutações também é uma determinante nos organismos naturais. Ainda que não tenhamos cruzamento, se tivermos uma taxa de mutação bastante elevada, nossa população poderá ser capaz de comportar-se como os vírus, mudando sempre para se adaptar ao seu meio ambiente, e reproduzindo-se de forma assexuada.

- **O Operador de Inversão**

Holland [Holland, 1975] define três técnicas para criar filhos diferentes dos pais. As duas primeiras (cruzamento e mutação) já foram explicados nos itens anteriores. Veremos agora o último deles, o operador de **Inversão**.

O operador de inversão, assim como a mutação, atua sobre um único nic
cromossomo. Ele inverte a ordem dos elementos entre dois pontos escolhidos aleatoriamente no cromossomo. Apesar deste operador ter sido inspirado por processos naturais, ele apresenta um overhead muito grande, o que degrada a performance do algoritmo. Na prática, este algoritmo não é muito utilizado [Davis, 1996].

3.3 Algumas Áreas de Aplicação da Computação Evolucionária

Segundo [Koza, 1996] há treze áreas relacionadas em que a computação evolucionária pode ser aplicada com grande sucesso. São elas:

3.3.1 Controle ótimo:

Consiste em achar uma estratégia de controle que use o estado corrente das variáveis de um sistema para escolher um valor adequado para uma variável de controle que provocará a mudança de estado do sistema, em direção à meta desejada, tentando ao mesmo tempo minimizar alguma medida de custo ou maximizar alguma medida de eficiência.

3.3.2 Planejamento:

Consiste em achar um plano que recebe informações de detectores ou sensores colocado em um ambiente sobre o estado de objetos colocados neste ambiente e utiliza esta informação para selecionar ações efetivas que modifiquem o estado de um sistema. Segundo [Koza, 1996] esta definição aplica-se ao planejamento na Inteligência Artificial e Robótica.

3.3.3 Indução de Seqüências:

Indução de seqüências consiste em achar uma expressão matemática que possa regenerar uma seqüência de elementos S_j para qualquer posição dada do índice j em uma seqüência $S = S_0, S_1, S_2, ..., S_j$ após a observação de algumas amostras dos valores destas seqüências. A indução de seqüências é um caso especial da regressão simbólica, brevemente exposta abaixo.

3.3.4 Regressão Simbólica:

Consiste em achar uma expressão matemática, na forma simbólica, que seja capaz de fornecer um ajuste perfeito, ou pelo menos aproximado, de uma série de valores amostrais de variáveis independentes e os valores associados das suas variáveis dependentes, ou seja, consiste em encontrar um modelo que se ajuste da melhor forma possível a um conjunto de dados amostrais. Nos casos em que há muito ruido nos dados amostrais coletados do mundo real, este problema é chamado de descoberta empírica. Se a faixa das variáveis independentes for a dos inteiros não-negativos, esta regressão é muitas vezes chamada de *indução de sequência.* Se existem múltiplas variáveis dependentes, este processo é chamado de *Regressão simbólica múltipla.*

3.3.5 Programação Automática:

Uma fórmula matemática para resolver um problema particular começa com certos valores dados (as entradas) e produz certos valores desejados (saídas). Em outras palavras, uma fórmula matemática pode ser vista como um programa de computador que toma vários valores de entrada e produz o resultado esperado como a sua saída. Como exemplo, considere o par de equações lineares

$$a_{11}x_1 + a_{12}x_2 = b_1$$

e

$$a_{21}x_1 + a_{22}x_2 = b_2$$

sobre duas incógnitas, x_1 e x_2. As duas fórmulas matemáticas bem conhecidas para resolver um par de equações lineares com seis valores dados (os quatro coeficientes e as duas variáveis) e os dois termos constantes. As duas fórmulas produzem como resultado os valores das duas incógnitas que satisfazem as equações. Os seis valores dados correspondem as entradas de um programa de computador, e os resultados calculados pelas fórmulas correspondem as saídas. A programação automática consiste então em descobrir automaticamente estas fórmulas que resolvam um determinado problema de acordo com certos parâmetros.

3.3.6 Descoberta de estratégias de jogos:

Consiste em encontrar uma estratégia que especifique qual o próximo movimento que um jogador deve fazer em determinado ponto do jogo.

3.3.7 Descoberta Empírica e Previsão:

Consiste em encontrar um modelo que relacione um conjunto de valores amostrais de variáveis independentes e os valores associados de suas variáveis dependentes, em algum sistema observado no mundo real. (Veja os itens 3.3.3 e 3.3.4).

3.3.8 Integração e diferenciação simbólica:

Consiste em encontrar a expressão matemática que é a integral ou a derivada de uma curva dada na sua forma simbólica. A curva dada pode ser apresentada como uma expressão matemática na forma simbólica, ou na forma de dados amostrais discretos.

3.3.9 Problemas Inversos:

Consiste em encontrar uma expressão matemática, na sua forma simbólica, que seja a inversa de uma curva dada. Este procedimento é idêntico ao da regressão simbólica e a busca por uma expressão matemática (programa de computador) que se ajuste aos dados amostrais finitos em questão. Esta função inversa pode ser vista como um programa de computador que toma os valores das variáveis dependentes de uma dada função matemática como entrada e produz os valores das variáveis independentes. A esta função damos o nome de função inversa.

3.4 Aplicações da Computação Evolucionária em Transportes

Daremos agora uma visão geral de como os algoritmos evolucionários têm sido utilizados aplicados a problemas reais na área de transportes. A dificuldade em

encontrar bibliografia especializada nesta área foi um fator limitante em nossas pesquisas, dado que a maior parte das publicações sobre algoritmos evolucionários enfocam muito mais questões teóricas sobre a técnica do que sobre aplicações práticas. Na maioria das publicações encontradas, o tipo de problema resolvido era o problema do Caixeiro Viajante, e geralmente, estas publicações resumiam-se a usar o problema do caixeiro mais como um elemento para mostrar algum avanço ou aperfeiçoamento nos algoritmos evolutivos tradicionais. Começamos analisando a aplicação de algoritmos a problemas clássicos de transportes, de alguma forma relacionados ao problema que é o objeto de estudo desta dissertação.

3.4.1 O Problema do Caixeiro Viajante

Este é um dos problemas clássicos de transportes, e é sempre um dos exemplos mais usados quando se quer testar algum algoritmo novo para roteamento. O problema pode ser definido de maneira simples, da seguinte forma [Hoffman e Wolfe, 1985]: um caixeiro viajante, tem de percorrer um certo número de cidades, começando por sua cidade natal, e terminando novamente em sua cidade, de tal forma que o seu percurso passe exatamente uma única vez em cada cidade, e a distância percorrida seja mínima.

Segundo [Hoffman e Wolfe, 1985] a importância do problema do caixeiro viajante não está em um grande leque de aplicações derivadas ou semelhantes a este problema, pois não existem muitas, mas sim porque este é um problema típico do seu gênero, a **otimização combinatorial**. O problema do caixeiro viajante é um problema de otimização, pois o objetivo é minimizar a distância total percorrida, porém, neste caso, de nada adianta se fazer uso das técnicas convencionais baseadas em cálculo diferencial, pois esta é uma situação de otimização combinatorial.

Um método diferente de resolução de problemas de otimização surgiu com o aparecimento **da programação linear**. Este ramo da matemática trata do problema de encontrar o mínimo de uma função linear sobre um poliedro descrito por um sistema de equações lineares e desigualdades em variáveis não-negativas. Como o número de pontos no qual um mínimo deve estar localizado deve incluir um vértice do poliedro, e o número de vértices é finito, a programação linear pode ser usada como uma ferramenta de otimização combinatorial. Muitos dos pesquisadores da área de Pesquisa Operacional

tentaram resolver o problema do caixeiro viajante com esta técnica, porém ele ainda permanece sem uma solução definitiva.

O motivo porque este problema ainda é considerado insolúvel é porque nenhuma das abordagens apresentadas até o momento consegue apresentar uma solução efetiva de maneira eficiente para um grande número de cidades. O problema do caixeiro viajante pode ser encarado como uma variante de um outro problema, também muito conhecido da teoria dos grafos, o problema do **ciclo hamiltoniano**. Este problema, apresentado no século XIX pelo matemático irlandês Sir William Rowan Hamilton, consiste em encontrar, em um grafo conexo e cíclico, um ciclo tal que contenha todos os vértices, passando apenas uma vez em cada vértice. De fato, segundo [Hoffman e Wolfe, 1985] o problema de decidir se um grafo tem um ciclo hamiltoniano é um caso especial do problema do caixeiro viajante.

Se atribuirmos a todas as arestas do grafo o comprimento 0, e para cada aresta ausente criarmos uma aresta de comprimento 1, obtemos um novo grafo que não contém ciclos Hamiltonianos. Resolvendo o problema do caixeiro viajante para este novo grafo, produziremos um ciclo hamiltoniano cujo comprimento total é zero, ou então terá um comprimento positivo, mostrando que não existe tal ciclo.

Apesar do esforço de muitos pesquisadores, o problema do caixeiro viajante ainda continua resistindo a todos os esforços para se encontrar uma solução definitiva e satisfatória para um grande número de cidades. A seguir, veremos uma pequena amostra dos tipos de problemas práticos que podem ser encarados como problemas do tipo do caixeiro viajante:

- **Roteamento de veículos:**

Por roteamento de veículos entende-se o problema de se determinar, para uma frota de veículos quais os clientes que devem ser servidos por quais veículos, e em que ordem cada veículo deve visitar seus clientes. Restrições a este tipo de problema geralmente incluem as capacidades dos veículos (para transportar algum tipo de carga) bem como as janelas de tempo para cada um dos clientes. Alguns algoritmos para este problema usam o modelo do problema do caixeiro viajante para o problema de ordenar cada cliente dos veículos.

- **Sequenciamento de tarefas:**

Considere o problema de sequenciar um conjunto de n tarefas em uma única máquina. As tarefas podem ser feitas em qualquer ordem, e o objetivo é completar todas estas tarefas no menor espaço de tempo possível.

3.4.2 Algoritmos Evolucionários e o Problema do Caixeiro Viajante

Com relação a área de transportes, talvez este seja o problema que mais tem sido foco dos pesquisadores de algoritmos evolutivos. Dentre as várias estratégias apresentadas na literatura, podemos destacar [Mühlenbein, 1991], que descreve a aplicação de um algoritmo genético paralelo, baseado na simulação de um mundo 2-D no qual os indivíduos de sua população vivem. Uma característica importante deste algoritmo é a capacidade que cada indivíduo tem de poder melhorar o seu fitness durante o seu tempo de vida. Este algoritmo é assíncrono e foi desenvolvido para executar em computadores paralelos. Neste artigo o autor examina a representação genérica para o problema do caixeiro viajante, na qual o gene no local i do cromossomo codifica as arestas do grafo (ou elos) que partem da cidade i. Cada elo deve aparecer no cromossomo apenas uma vez, para assegurar que o indivíduo codificado por aquele cromossomo produza uma solução válida. A grande falha deste algoritmo está no fato de que o operador de cruzamento quando aplicado ao cromossomo pode produzir indivíduos inválidos. O autor também apresenta uma outra forma de realizar o cruzamento, criando um novo operador de cruzamento chamado de MPX (Maximal Preservative Crossover Operator – Operador de cruzamento preservativo maximal) [Mühlenbein, 1991].

3.5 Raciocínio Baseado em Casos (RBC)

Raciocínio baseado em casos pode ser definido como uma técnica que tenta resolver problemas através da análise de casos anteriores que apresentem características semelhantes ao problema a ser resolvido. Se o problema a ser resolvido é muito similar a um ou mais casos anteriores, então a mesma solução utilizada anteriormente é aplicada ao problema atual. Caso contrário, uma nova solução poderá ser adaptada com base nas soluções adotadas para os casos mais similares [Kolodner, 1993].

Esta técnica representa uma forma de aprendizado bastante similar à forma com que seres humanos acumulam experiência durante a vida: Depois de enfrentar situações parecidas por diversas vezes, a solução para novas situações similares é facilmente obtida através da "lembrança" de situações passadas. Um dos exemplos comuns do uso deste princípio em nosso cotidiano é apresentado em [Kolodner, 1993], onde se descreve a atuação de um advogado, que, frente a um novo caso, recorre a jurisprudência para basear os seus argumentos. Por causa de sua característica de acúmulo e uso de experiências anteriores na solução de novos problemas, diz-se que o raciocínio baseado em casos é um modelo que incorpora solução de problemas, compreensão e aprendizado junto com processamento de memória [Kolodner, 1993].

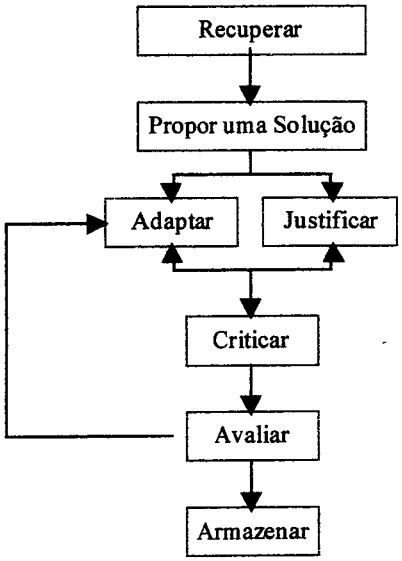


Figura 3: Processo de Raciocínio baseado em casos [Kolodner, 1993]

A técnica de raciocínio baseado em casos é bastante vantajosa em situações recorrentes, ou seja, situações que ocorram seguidamente, com pequenas alterações em cada uma de suas ocorrências. A Figura 3 mostra o processo de raciocínio baseado em casos, destacando seus componentes maiores. A seguir veremos os conceitos básicos desta técnica.

3.5.1 Conceitos Básicos:

3.5.1.1 Caso:

Um caso representa um conhecimento específico referente à um certo contexto. Podem possuir formatos e tamanhos diferentes, e associam soluções com problemas, resultados com situações, ou ambos. Um caso registra experiências que são diferentes do esperado. Segundo [Kolodner]: "*Um caso é um pedaço contextualizado de conhecimento que representa uma experiência que ensina uma lição fundamental para alcançar as metas do raciocinador.*"

3.5.1.2 Índice:

Um índice pode ser visto como uma estrutura auxiliar que auxilia no processo de recuperação dos casos. Ele diz sob que circunstâncias é apropriado recuperar o caso [Kolodner].

3.5.1.3 Recuperação:

Chama-se recuperação ao processo de se obter, da base de casos, um caso, ou um conjunto deles, que sejam suficientemente similares ao caso em questão sendo analisado, e que, em virtude desta similaridade sejam capazes de apontar uma solução para o problema. O processo de recuperação envolve dois passos:

Recuperação de casos anteriores: A meta desta passo é recuperar "bons" casos que possam servir de base para o raciocínio que será desenvolvido nos passos posteriores. Os casos são considerados "bons", se tiverem potencial para fazer predições relevantes sobre o novo caso. Neste passo, a recuperação é feita utilizando-se como índice as características do novo caso para efetuar a busca na base de casos.

Seleção do melhor subconjunto: Neste passo selecionam-se os casos mais promissores a partir daqueles que foram gerados no passo 1, com o objetivo de separar apenas os casos mais relevantes, obtendo com isso apenas uns poucos candidatos, que realmente mereçam ser considerados. Para algumas classes de problemas precisamos apenas do melhor caso; em outras, precisamos de um conjunto de casos.

3.5.1.4 Proposta de uma solução:

Neste passo, as partes relevantes dos casos selecionados durante a recuperação são extraídas para formar uma solução para o novo caso. Várias questões surgem na construção de uma solução:

Qual a porção apropriada do caso antigo que deve ser selecionada? Um caso antigo pode ser muito grande e é importante deixar de lado as partes deste caso que não tenham relevância para o problema que estamos tentando resolver;

A estrutura interna do caso antigo, especialmente as dependências entre partes diferentes do caso dizer ao raciocinador como expandir o seu foco de maneira relevante. Isto faz com que, quando o raciocinador enfoca uma solução em um caso antigo, aquelas características do caso que levam à seleção das soluções relevantes também estão em foco.

Estas são questões de relevância com respeito a primeira proposta de uma solução com base apenas nos casos recuperados. Após esta fase, é necessário adaptar-se a solução encontrada, considerando as diferenças existentes entre os casos recuperados e o novo caso que representa o problema a ser resolvido. Isto é feito na etapa seguinte: adaptação.

3.5.1.5 Adaptação:

Na adaptação, tentamos encontrar uma solução para os novos casos com base nas soluções encontradas para os casos anteriores. Apenas recuperar as soluções para os casos anteriores e apresenta-las como solução para o novo caso pode não ser uma boa estratégia, pois as novas situações raramente casam totalmente com as situações passadas. No processo de adaptação, podemos identificar dois passos, que são:

e) Calcular o que precisa ser adaptado e

f) fazer a adaptação.

Várias questões também surgem neste processo. Considerando o próprio processo de adaptação, percebemos que, para um dado domínio ou tarefa, precisamos de um conjunto de estratégias ou heurísticas, que nos auxilie no processo de adaptar as

soluções para os casos antigos em soluções para os casos novos. Estas estratégias ou heurísticas podem ser implementadas de forma *ad hoc*. Outra abordagem seria tentar encontrar um conjunto de estratégias ou heurísticas gerais de adaptação, as quais poderiam servir para qualquer domínio, servindo de orientação para a definição de estratégias de adaptação mais especializadas.

Outro elemento importante na adaptação são as metodologias para a avaliação de quais as partes de uma solução antiga necessitam de adaptação para se constituírem uma nova solução.

3.5.1.6 Raciocínio Avaliativo: Justificativa e Crítica

Nestes dois passos, a solução é justificada, muitas vezes até sem Ter sido testada no mundo real. Esta etapa pode ser pensada como uma etapa de validação, se dispomos de todo o conhecimento necessário para isto. Na maioria das vezes, contudo, este não é o caso. Quando não dispomos de todo o conhecimento necessário para justificar a solução encontrada, ainda podemos fazer uma crítica desta solução. Esta crítica é feita comparando-se e contrastando-se a solução proposta com outras soluções similares. Este passo requer chamadas recursivas ao processamento da memória para recuperar casos com soluçoes similares. Outra forma de se efetuar críticas é propondo situações hipotéticas que testem a robustez da solução encontrada, através de simulações.

3.5.1.7 Avaliação

Neste passo, o resultado do raciocínio é experimentado no mundo real, permitindo assim que haja um feedback entre a solução oferecida e o resultado obtido com a aplicação desta solução. Se o resultado obtido com a aplicação da solução for o esperado, então não são necessárias mais análises a respeito do problema. Caso contrário, torna-se necessário tentar fornecer uma explicação para o resultado anormal que foi obtido. Esta explicação consiste em considerar o que causou a anomalia, e o que poderia Ter sido feito para preveni-la. Em alguns casos isto pode ser feito utilizando-se o raciocínio baseado em casos, reaplicando uma explicação anterior. Outros casos, no entanto, requerem o uso de outras técnicas inteligentes para tentar obter esta explicação, como sistemas especialistas.

3.5.1.8 Armazenamento (ou atualização da memória)

Este passo trata do armazenamento do novo caso na base de casos do sistema, o que permitira sua utilização em situações futuras. Compõem este novo caso: o problema e sua solução, acrescidos de quaisquer fatos subjacentes dos quais o sistema sabe como fazer uso. Neste passo, um dos índices mais importantes diz respeito à indexação. É de extrema importância escolher a forma adequada para indexar o novo caso. Esta indexação deve ser feita de tal forma que permita que este novo caso seja recuperado sempre que for útil na solução de novos problemas. Os índices escolhidos para indexar este novo caso não podem ser tão poucos que permitam que o caso seja deixado de lado em situações em que ele seria útil, e também não podem ser tantos que provoquem a sua recuperação inutilmente. A escolha destes índices deve ser bastante cuidadosa.

3.5.2 Principais Vantagens e Desvantagens da Técnica RBC

As principais vantagens dos sistemas RBC são as seguintes [Kolodner, 1993]:

12. Permitir a proposição de uma solução rapidamente, evitando gastar tempo desnecessário desenvolvendo uma solução a partir do zero.
13. Permite que uma solução seja proposta, ainda que o domínio não seja completamente compreendido pelo raciocinador.
14. Fornece um mecanismo para que o raciocinador avalie as soluções, mesmo em casos em que não há um método algorítmico disponível para fazê-la.
15. Casos são úteis na interpretação de conceitos abertos e mal definidos.
16. A recordação de experiências passadas é bastante útil para alertar sobre a possível ocorrência de problemas potenciais, prevenindo o raciocinador para que este tome as providências necessárias, evitando assim a reincidência de algum eventual problema.
17. Os casos ajudam o raciocinador a enfocar o seu potencial de raciocínio nas partes importantes do problema, apontando quais as características do problema que são relevantes.

Em contrapartida, as principais desvantagens dos sistemas RBC são as seguintes:

1. raciocinador pode ser tentado a reutilizar uma solução antiga de forma "cega", ou seja, sem tentar validá-la de acordo com o novo contexto em que esta situação ocorre.
2. raciocinador pode permitir que alguns casos influenciem-no demais na solução de um novo problema.
3. A maioria das pessoas, especialmente os novatos, não são capazes de recordar um conjunto apropriado de casos quando raciocinam.

Segundo [Kolodner, 1993], o raciocínio baseado em casos é uma forma natural de raciocinar, e o esforço em explicar os processos envolvidos no raciocínio baseado em casos pode nos ajudar a aprender como ensinar pessoas a raciocinar melhor utilizando a sua experiência passada. Cabe somente lembrar que apenas reaplicar soluções anteriores sem fazer uma validação pode resultar na obtenção de um resultado ineficiente ou mesmo incorreto. É importante, portanto fazer-se uma avaliação adequada antes de propor uma solução.

3.5.3 Classificação dos Sistemas RBC

Kolodner [Kolodner, 1993] classifica os sistemas RBC em duas grandes categorias:

a) Problem Solving (Resolução de Problemas)

b) Sistemas interpretativos

3.5.3.1 RBC para Resolução de Problemas (Problem Solving)

Os sistemas nesta categoria englobam uma série de tarefas, que incluem: planejamento, diagnóstico e design. Em cada um destes casos, o acúmulo de experiências na forma de casos permite um aumento na eficiência do raciocinador, tornando-o capaz de resolver rapidamente problemas que levariam mais tempo para serem resolvidos se a resolução partisse do zero.

3.5.3.2 RBC Interpretativo

O raciocínio baseado em casos interpretativo caracteriza-se por ser um processo de avaliação de situações ou soluções no contexto de sua experiência anterior.

Este tipo de raciocínio pode ser melhor compreendido, se tomarmos como exemplo as cortes de justiça, que tomam suas decisões com base na interpretação de alguma lei em contraste com algum caso, fazendo uso da jurisprudência relativa ao caso. Advogados fazem uso de raciocínio interpretativo para fundamentar os seus argumentos.

O raciocínio baseado em casos interpretativo é bastante útil quando não existem métodos computacionais disponíveis para se avaliar uma certa solução ou posição a ser assumida. O raciocínio interpretativo pode atuar em uma série de tarefas, incluindo: classificação, ajuste de situação, Localização de defeitos, etc.

Após esta revisão das duas técnicas utilizadas na contrução do protótipo, passaremos, na próxima seção, a discutir sistemas híbridos, onde trataremos da união destas e outras técnicas inteligentes, na construção de um sistema inteligente mais robusto.

3.6 Sistemas Híbridos

Atualmente, em virtude do desenvolvimento e do melhor estudo das várias técnicas de inteligência artificial que foram desenvolvidas, tem-se conseguido chegar a uma visão mais clara destas técnicas, tanto com respeito as suas capacidades quanto o que respeita as suas limitações. Hoje não se procura mais uma técnica universal, que resolva a todos os problemas com a mesma eficiência. Busca-se hoje a integração das técnicas existentes, com o intuito fazer com que uma determinada técnica seja capaz de suprir a deficiência de outras, ou mesmo servir como complemento a outras técnicas, objetivando ampliar a performance global dos sistemas. Desta idéia de junção, de complementação, nasceram os sistemas híbridos.

A integração das várias tecnologias inteligentes abrange um campo bastante vasto, indo desde questões fundamentais sobre a natureza da cognição até os problemas da teoria da computação, na tentativa de se encontrar as melhores maneiras de se implementar sistemas híbridos [Medsker, 1995]. Outro aspecto que torna bastante interessante a pesquisa na área de sistemas híbridos é o fato de que os seres humanos também são "máquinas" híbridas de processamento de informações [Medsker, 1995]. Muitas das nossas ações são determinadas, em parte por fatores genéticos, e em parte pelo nosso próprio poder de decisão com relação as situações adversas. Esta nossa capacidade de decisão

também consiste de uma série de processos complexos através dos quais nós aprendemos e tomamos decisões. Assim, desenvolver sistemas híbridos representa um desafio, não apenas no sentido tecnológico, mas também como ferramenta para tentar entender melhor o mecanismo de raciocínio humano.

3.6.1 Classificação de Sistemas Híbridos

Goonatilake e Khebal [in Medsker, 1995] apresentam um esquema de classificação que considera fatores como funcionalidade, arquitetura de procesamento e requisitos de comunicação. O objetivo do esquema de classificação apresentado por Goonatilake e Khebal não visa apenas classificar os sistemas híbridos criados pela junção de técnicas de IA, mas também outros sistemas híbridos que sejam criados usando outras técnicas, como clusterização estatística, técnicas de regressão, programação linear, etc., o que faz com que esta classificação seja de âmbito bastante genérico.

O sistema de classificação de Goonatilake e Khebal divide os sistemas híbridos em três categorias:

a) Substituição de função

b) Intercomunicantes

c) Polimórficos

3.6.1.1 Sistemas Híbridos de Substituição de Função

Neste tipo de sistema, é realizada uma decomposição funcional de uma única técnica de IA, onde a função principal de uma dada técnica é substituída por outra técnica de processamento inteligente. Nesta categoria podemos citar sistemas híbridos de redes neurais e algoritmos genéticos, onde o mecanismo de mudança de pesos por backpropagation pode ser substituido pelos operadores genéticos de um algoritmo evolucionário.

3.6.1.2 Sistemas híbridos intercomunicantes

Os sistemas híbridos construídos desta forma são compostos geralmente por sistemas independentes e auto-contidos, que realizam funções separadas e bem definidas, trocando informações entre si para produzir o comportamento desejado. Este tipo de sistema é normalmente desenvolvido quando o problema a ser resolvido pode ser dividido em tarefas de processamento bastante distintas, permitindo que módulos de processamento inteligente que usam técnicas diferentes possam ser usados para se resolver cada uma das partes destes problemas. Normalmente, neste esquema, os módulos que implementam técnicas diferentes são coordenados por um módulo de controle.

3.6.1.3 Híbridos Polimórficos

Sistemas híbridos polimórficos são aqueles que tentam alcançar a funcionalidade de diferentes técnicas de processamento inteligente através de uma arquitetura de processamento único. Nesta categoria encontram-se sistemas como redes neurais que tentam realizar tarefas simbólicas, ou ainda, que simulam o comportamento de uma busca genética.

3.6.2 Modelos de Sistemas Híbridos de Medsker

Medsker [Medsker, 1995] propõe uma outra forma de classificação, baseada mais na forma de acoplamento dos módulos de processamento inteligente do que na sua funcionalidade ou arquitetura. Segundo a sua classificação, os sistemas híbridos estão divididos em cinco classes:

a) Modelos Stand-alone

b) Transformações

c) Acoplamento fraco

d) Acoplamento forte

e) Integração plena

3.6.2.1 Modelos Stand-Alone

Modelos Stand-alone consistem de módulos independentes que não interagem entre si de nenhuma forma. Muito embora pareça estranho, a princípio, o desenvolvimento de sistemas com estas características, existem algumas situações em que este tipo de arquitetura é bastante útil: em primeiro lugar, podem servir como uma forma de comparação direta de duas técnicas diferentes na resolução de dado problema; usados em paralelo, fornecem redundância no processamento, o que pode aumentar a confiabilidade destes sistemas; e, por último, este modelo pode ser usado para promover o desenvolvimento de uma outra técnica que pode Ter o seu desempenho diretamente comparado com a técnica já implementada.

Exemplos de sistemas desta categoria podem ser, por exemplo, sistemas de diagnósticos que são compostos de dois módulos distintos com a mesma finalidade, ou seja, fornecer o diagnóstico. Os resultados dos dois módulos são comparados; e caso haja discordância quanto ao diagnóstico, fica a cargo do usuário decidir quais as medidas a serem tomadas.

3.6.2.2 Modelos Transformacionais

Estes modelos são bastante similares aos modelos stand-alone, no sentido de que, o resultado final ainda é um sistema em que os módulos não interagem entre si. Um exemplo característico citado em [Medsker, 1995] é o de um sistema para pesquisa de marketing, onde uma rede neural é utilizada para obter o conhecimento inicial que servirá de base para o desenvolvimento de um sistema especialista, o qual será posto em operação, em substituição a rede neural original. Este tipo de sistema normalmente é desenvolvido desta forma quando a aquisição de conhecimentos não pode ser feita diretamente por um engenheiro de conhecimento. Medsker também cita que, apesar de menos comum, sistemas especialistas também podem ser transformados em redes neurais, quando estes mostram-se incapazes de resolver o problema adequadamente, seja por questões de velocidade ou de captura de generalizações.

3.6.2.3 Modelos de Acoplamento Fraco

Os sistemas que se enquadram nesta categoria representam a primeira forma de um sistema híbrido realmente integrado, se bem que de forma ainda bastante restrita. Segundo este modelo, os sistemas são desenvolvidosem módulos separados que implementam técnicas inteligentes distintas e realizam tarefas distintas e bem definidas, e trocam informações entre si através de arquivos de dados.

3.6.2.4 Modelos de Acoplamento Forte

Aqui existe uma certa sobreposição desta categoria com a descrita anteriormente, pois [Medsker, 1995] descreve esta categoria como englobando os sistemas híbridos construídos por módulos independentes, mas que se comunicam através estruturas de dados residentes em memória. Atualmente, o conceito de memória tem se expandido para englobar também a memória secundária, na forma de memória virtual. Considerando isto, podemos perceber que parece não haver tanta diferença entre um modelo e outro.

3.6.2.5 Modelos de Integração completa

Os sistemas dentro desta categoria compartilham tanto estruturas de dados quanto representações de conhecimento, e a sua comunicação ocorre de maneira altamente integrada entre os módulos. Exemplificando, num sistema composto por uma rede neural e um sistema especialista, esta pode ser usada para receber e generalizar os dados para um sistema especialista, o qual, de posse destes dados pode tomar alguma decisão, que será então apresentada ao usuário. Diferentemente dos esquemas anteriores, neste não há um pré-armazenamento por um dos módulos, o qual poderá ser utilizado por outro, mas a saída de um módulo de sistema é justamente a entrada de outro módulo. Isto torna a integração muito mais eficiente, porém torna mais difícil construir o sistema, visto que as informações tem que ser traduzidas de um paradigma para outro, por exemplo, a informação neural processada pela rede neural tem que ser convertida em símbolos para possibilitar o seu processamento simbólico pelo sistema especialista.

3.7 Considerações Sobre a Integração de Algoritmos Genéticos e Raciocínio Baseado em Casos

A literatura sobre sistemas híbridos envolvendo algoritmos genéticos e raciocínio baseado em casos ainda é bastante pequena, isto porque é mais comum encontrarmos trabalhos relativos à integração das outras técnicas mais tradicionais da IA, como sistemas especialistas, redes neurais e sistemas difusos. A técnica de raciocínio baseado em casos ainda é relativamente recente, e somente agora surgem as primeiras tentativas de integração com outras tecnologias.

Em [Medsker, 1995] encontramos a seguinte relação de trabalhos relacionados à integração de genéticos e raciocínio baseado em casos: [Louis, McGraw, Wyckof, 1993] discute o uso de raciocínio baseado em casos como um mecanismo de justificativa para os resultados obtidos por um algoritmo genético; [Ramsey e Grefenstette, 1993] propõe um método baseado em casos para inicializar algoritmos genéticos e uma abordagem geral para aprendizado contínuo em um ambiente mutável; [Oppacher e Deugo, 1991] dissertam sobre a integração de raciocínio baseado em casos e algoritmos genéticos, enfocando problemas como a adição de casos redundantes, sobreposições e conflitos com os casos antigos, e a dificuldade que, mesmo os métodos fortemente adaptativos tem de produzir soluções repetitivas.

A área de sistemas híbridos, como podemos perceber, ainda é um terreno bastante fértil, principalmente com relação a integração de algoritmos genéticos e raciocínio baseado em casos, duas técnicas relativamente recentes e bastante promissoras.

3.8 Resumo

Neste capítulo abordamos as duas técnicas de Inteligência Artificial que foram utilizadas na construção do protótipo desenvolvido neste trabalho. Abordamos primeiramente a computação evolucionária, apresentando os seus conceitos básicos, e a seguir descrevemos a técnica de raciocínio baseado em casos, enfocando suas características principais. Após estas duas técnicas terem sido abordadas, foi abordada a técnica de sistemas híbridos, técnica utilizada para reunir duas ou mais técnicas diferentes

de IA num mesmo sistema, como forma de tentar suprir as falhas de cada uma das técnicas. O próximo capítulo descreve a implementação do sistema PROTEUS, mostrando como estas técnicas foram utilizadas na construção deste sistema híbrido.

4. O Sistema PROTEUS

4.1 Introdução

Este capítulo descreve a implementação e mostra exemplos de utilização do sistema PROTEUS (Planejador de ROtas para Transportes em Espaços Urbanos e Similares). Serão apresentados aqui detalhes da sua estrutura híbrida, a forma de representação da malha viária, descrevendo brevemente o funcionamento do módulo RBC e a seguir, descrevendo o módulo genético com um enfoque um pouco mais detalhado sobre sua estrutura, dado que este representa o ponto principal desta dissertação.

Com a implementação deste sistema híbrido, espera-se obter um sistema eficiente, que possua um bom desempenho; espera-se também demonstrar como duas técnicas diferentes da IA podem ser combinadas para a resolução de problemas reais, mostrando como isto podem servir para compensar as falhas de ambas [Peres et al. 1997]. Como exemplos de sistemas similares, citamos nos capítulos anteriores os sistemas ETAK [Assad, 1988] e PULSAR [Scott et al, 1997], [Bernstein e Kornhouser, 1998], [Scott, e Bernstein, 1997], [Padmos e Bernstein, 1997], ambos enfocam o mesmo problema, porém com abordagens diferentes.

4.2 Arquitetura híbrida

O Sistema PROTEUS foi projetado como um sistema de arquitetura híbrida, com o objetivo de explorar o que há de melhor nas duas técnicas nas quais ele está baseado: Raciocínio Baseado em Casos (RBC) e Algoritmos Genéticos (AG). O sistema pode ser classificado, segundo Medsker [Medsker, 1995] como de acoplamento fraco, dado que a comunicação entre o módulo RBC e o módulo AG se dá através de arquivos. Segundo a taxonomia de [Goonatilake e Khebal, 1995], este sistema poderia ser classificado como um sistema de módulos intercomunicantes, dado que ambos os módulos realizam tarefas bem definidas e complementares, trocando dados entre sí através de arquivos.

O Sistema PROTEUS (Planejador de ROtas para Transportes em Espaços Urbanos e Similares) foi inicialmente planejado para ser utilizado em transportes rodoviários em ambientes urbanos, mas o seu uso pode ser estendido para outras situações de transportes, como transportes ferroviários, marítmos e aéreos, já que a sua estrutura representacional flexível permite que tal adaptação seja feita sem grandes esforços. A arquitetura do sistema pode ser classificada, segundo Goonatilake e Khebal, como sendo uma arquitetura de substituição de função, dado que a função de geração de novos casos dá-se pelo módulo genético, que é responsável por abastecer a base de casos para que o módulo RBC possa fazer a inferência e selecionar os melhores planos de rotas que serão submetidos ao usuário. A Figura 4 mostra a arquitetura do sistema de maneira geral.

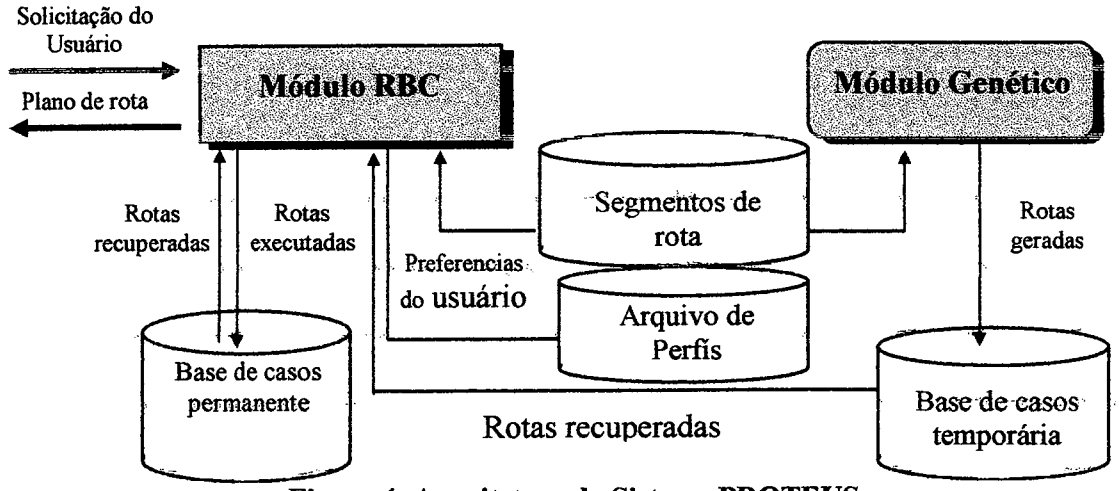


Figura 4: Arquitetura do Sistema PROTEUS

O Sistema PROTEUS compõe-se de dois módulos principais: o Módulo Genético, responsável pela geração das rotas e o seu armazenamento em uma base de rotas temporária, para posterior avaliação pelo módulo RBC e o Módulo RBC, responsável pela recuperação dos casos a partir de uma base de rotas permanentes, as quais já foram previamente executadas pelo usuário, e por serem consideradas satisfatórias foram armazenadas nesta base, responsável também pela avaliação das rotas geradas pelo módulo genético, as quais estão armazenadas em uma base de casos temporária. A próxima seção descreve resumidamente o funcionamento do módulo RBC. [Peres et al., 1997]

4.3 Módulo RBC

O módulo RBC tem duas grandes responsabilidades neste sistema:

f) É o módulo dedicado à interação com o usuário, e

g) É o responsável pela avaliação das rotas, considerando o perfil do usuário (armazenado em uma base de perfis). A Figura 5 mostra um fluxograma simplificado do modo de execução deste módulo.

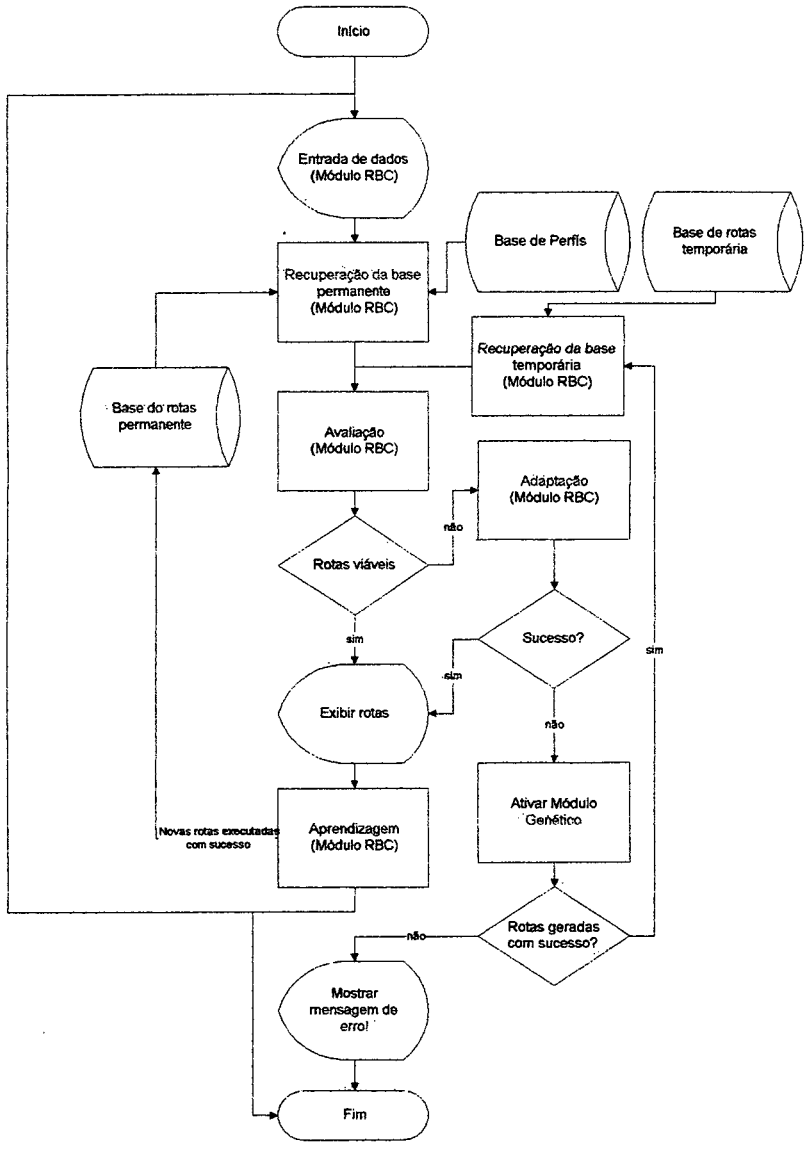


Figura 5: Fluxograma do Módulo RBC

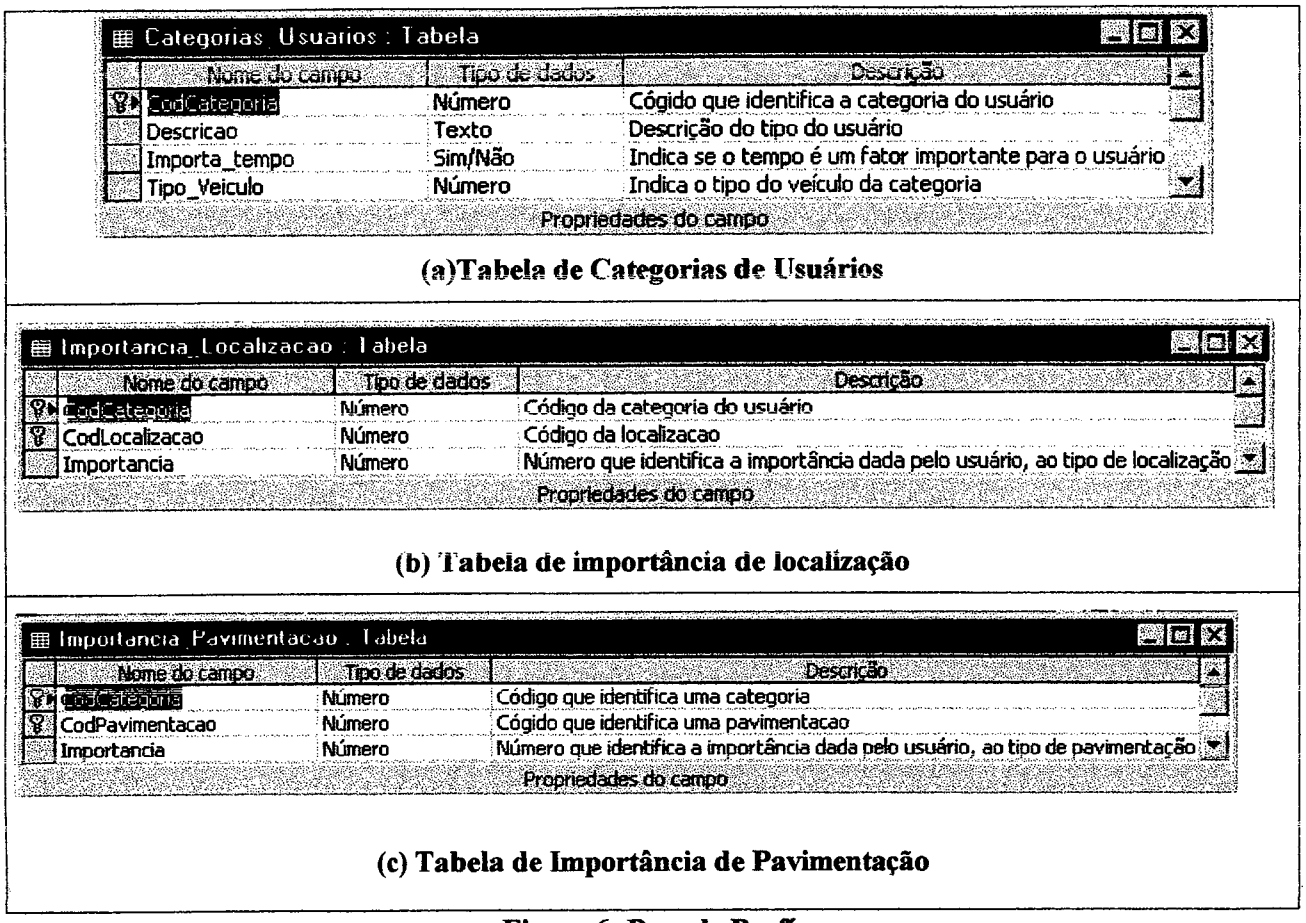
O Módulo RBC recebe a solicitação do usuário para obtenção de um plano de rotas, informando a origem e o destino desejados. A partir destes dados de entrada, o módulo

RBC consulta a base de perfis e, com base nas preferências pessoais do usuário[3](Figura 6), verifica em sua base de casos permanente se há alguma rota que já tenha sido executada anteriormente com sucesso, para esta mesma origem e este mesmo destino; se existir(em) tal (is) rota (s), ela(s) é (são) apresentada(s) ao usuário. Caso esta(s) rota(s) não exista(m), o módulo RBC tentará adaptar alguma(s) rota(s) existente(s), de modo a tentar fornecer uma resposta ao usuário. Caso esta adaptação falhe, o módulo RBC acionará o módulo genético, solicitando a este que gere rotas alternativas entre a origem e o destino dados. Após a execução do algoritmo genético, as rotas são armazenadas em uma base temporária, para que o módulo RBC possa avaliá-las contra as preferências do usuário armazenadas no arquivo de perfis [Peres, 1999], [Peres et al, 1997].

O primeiro requisito proposto em nosso modelo é que a rota seja adequada ao arquivo de perfis. Seres humanos têm uma tendência natural de manter ou adaptar soluções de sucesso na resolução de problemas anteriores, quando encaram novos problemas com características similares. Esta foi a principal motivação para a construção de um módulo de raciocínio baseado em casos para realizar a avaliação dos planos de rotas, aplicando raciocínio analógico sobre situações anteriores armazenadas em sua base de rotas [Peres, 1999], [Peres et al, 1997] .

Inicialmente, o módulo vasculha a base, tentando encontrar as rotas mais similares ao perfil do usuário. Eventualmente, caso nenhuma rota suficientemente adequada seja encontrada, ou caso ocorram fatos que tornem uma solução temporariamente inaplicável (Ex: Acidentes de trânsito), torna-se necessário realizar uma adaptação, a qual é efetuada realizando-se a troca dos segmentos interditados por outro conjunto de segmentos que também sejam satisfatórios, dentro de uma margem de tolerância. A Figura 7 mostra um exemplo de adaptação, onde o intervalo ABC pode ser substituído pelo intervalo ADEC, o qual foi observado em um caso diferente.

[3] Na implementação deste protótipo não se implementou uma interface para obter os dados do perfil do usuário. A base de perfis foi preenchida de maneira estática, e, neste protótipo o usuário apenas seleciona um dos perfis já existentes. A implementação desta característica, no entanto, é bastante trivial.



Categorias_Usuarios : Tabela

Nome do campo	Tipo de dados	Descrição
CodCategoria	Número	Cógido que identifica a categoria do usuário
Descricao	Texto	Descrição do tipo do usuário
Importa_tempo	Sim/Não	Indica se o tempo é um fator importante para o usuário
Tipo_Veiculo	Número	Indica o tipo do veículo da categoria

Propriedades do campo

(a)Tabela de Categorias de Usuários

Importancia_Localizacao : Tabela

Nome do campo	Tipo de dados	Descrição
CodCategoria	Número	Código da categoria do usuário
CodLocalizacao	Número	Código da localizacao
Importancia	Número	Número que identifica a importância dada pelo usuário, ao tipo de localização

Propriedades do campo

(b) Tabela de importância de localização

Importancia_Pavimentacao : Tabela

Nome do campo	Tipo de dados	Descrição
CodCategoria	Número	Código que identifica uma categoria
CodPavimentacao	Número	Cógido que identifica uma pavimentacao
Importancia	Número	Número que identifica a importância dada pelo usuário, ao tipo de pavimentação

Propriedades do campo

(c) Tabela de Importância de Pavimentação

Figura 6: Base de Perfis

Se nenhuma solução satisfatória é encontrada, o módulo genético é ativado, na tentativa de criar uma nova base de casos temporária, o que permite que o módulo RBC reinicie o processo de recuperação e/ou adaptação sobre esta base temporária. Após este processo, o usuário deve escolher e executar uma das rotas encontradas e decidir quais delas devem ser armazenadas na base de rotas permanentes.

Perfis de usuários			
Código	Descrição	Tempo-crucial	Tipo-Veículo
1	Policial	Sim	1
2	Policial	Sim	2
3	Ambulância	Sim	3
4	Taxista	Não	2

Tabela 2: Exemplo de perfis de usuários

Um dos aspectos mais interessantes desta modelagem é que ela permite uma rápida adaptação, de acordo com as características dinâmicas da própria malha viária, o que é tornado possível pela característica de aprendizado contínuo do módulo de RBC, através do crescimento contínuo da base de rotas. No início, a base de rotas permanente encontra-se vazia, e a medida que o sistema vai sendo executado, ela vai sendo preenchida com as rotas geradas ou adaptadas e executadas com sucesso, o que caracteriza o seu aprendizado contínuo.

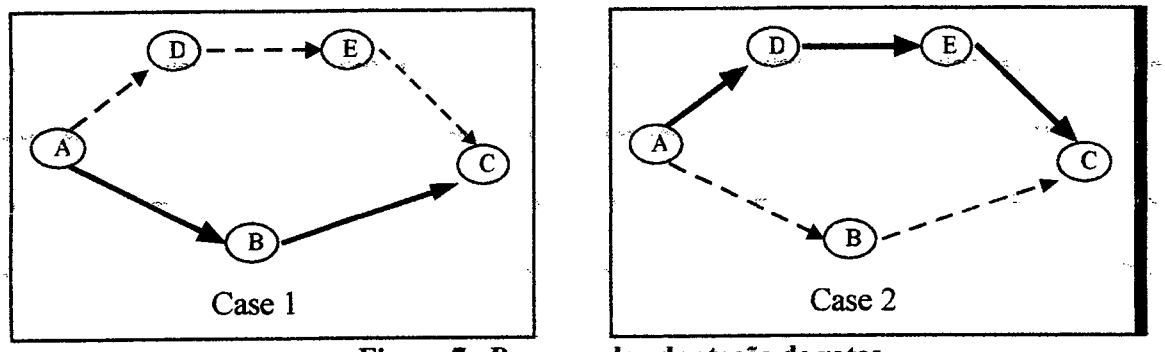


Figura 7 : Processo de adaptação de rotas

A indexação dos casos pelo módulo RBC [Peres, 1999] é feita pelos atributos "origem" e "destino" da rota. No sistema PROTEUS cada rota é considerada como sendo composta por sub-rotas, cada uma delas podendo ser utilizada como solução para alguma requisição feita ao sistema. Observe a Figura 7. Nela podemos observar que a rota apresentada no caso 2 (A-D-E-C) possui a sub-rota (D-E-C), o que faz com que este caso seja retornado quando uma solicitação por uma rota com "origem" = D e "destino" = C for apresentada ao sistema.

A similaridade em nosso sistema é calculada como a somatória dos pesos atribuídos a cada uma das características presentes nas rotas geradas pelo algoritmo genético, ou recuperadas da base permanente de rotas. Esta medida é aplicada a cada rota encontrada, avaliando-se cada um dos segmentos que a compõe em contraste com o perfil de usuário selecionado. O calculo da similaridade foi equacionado da seguinte maneira [Peres, 1999]:

$$sim = \left(1 - \left(\frac{1}{1 + \left(\frac{\partial}{\ell}\right)}\right)\right) \times 100$$

onde:

sim = valor da similaridade em porcentagem do caso recuperado em relação ao caso analisado (problema de entrada)

∂ = valor inicial da similaridade entre a rota avaliada e a rota desejada;

ℓ = comprimento total da rota;

e temos que

$$\partial = \sum_{i=1}^{n} \frac{\sum_{j=1}^{m} \text{peso de i em j}}{\text{m}}$$

onde :

i = característica a ser analisada;

j = segmento a ser analisado;

m = número de segmentos da rota avaliada;

n = número de características consideradas;

A Próxima seção detalha o módulo genético, Objeto central desta dissertação.

4.4 Módulo Genético

A tarefa do módulo genético consiste em encontrar rotas válidas entre os pontos de origem e destino dados, levando em consideração apenas a distância mínima entre os pontos. O Algoritmo genético apenas tenta encontrar o maior número possível de caminhos viáveis, classificando-os em função de seu comprimento (Menor distância). A Figura 8 mostra o fluxograma simplificado da execução deste módulo.

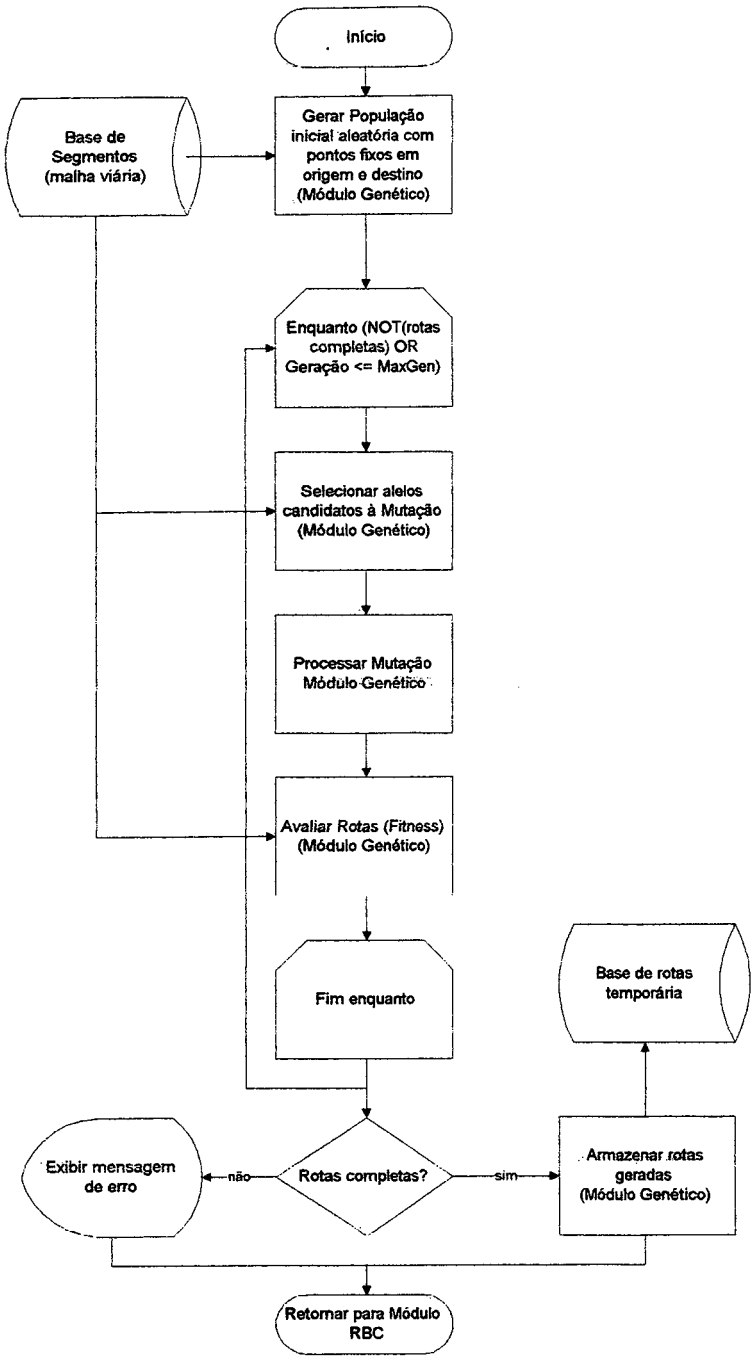
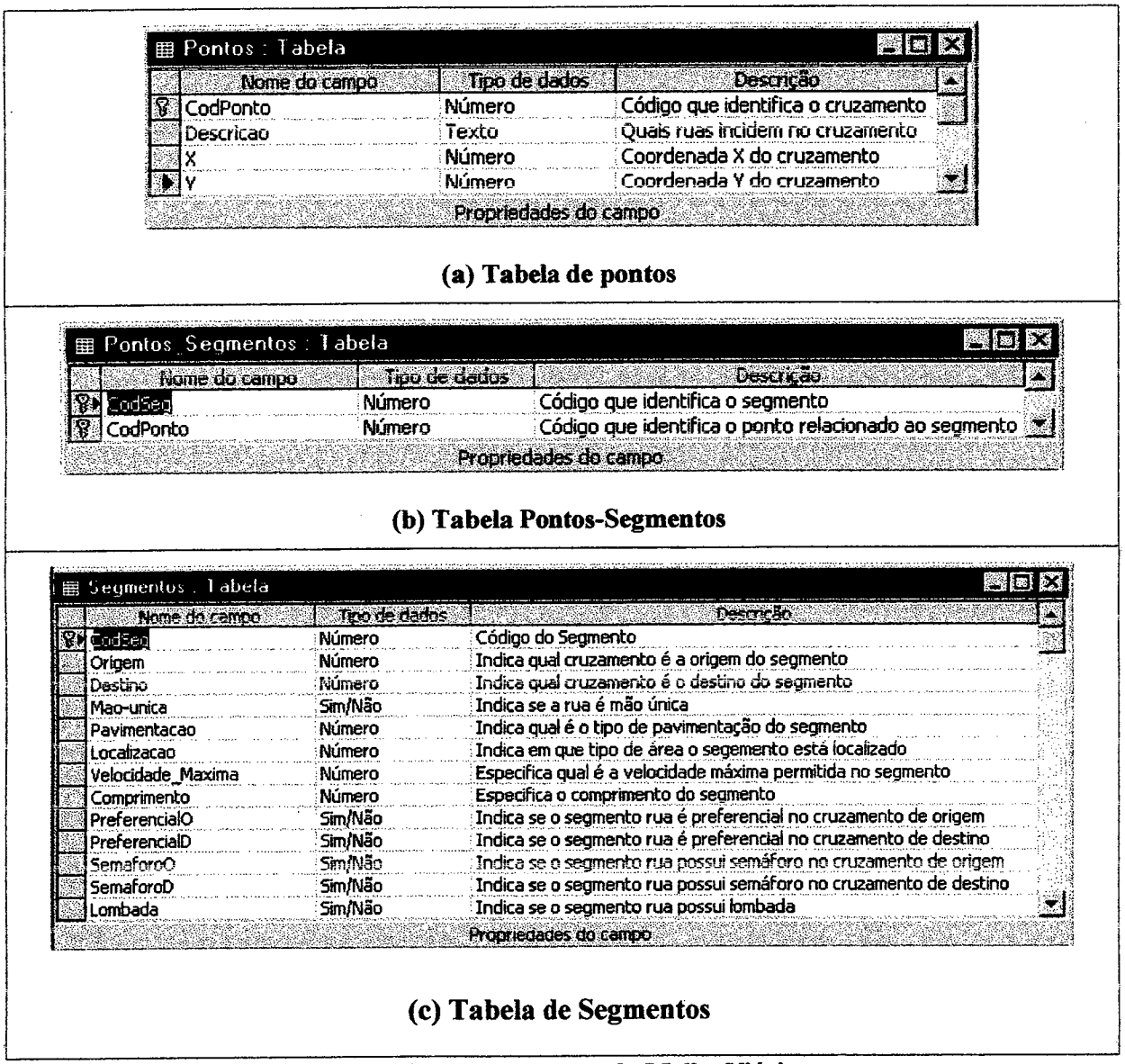


Figura 8: Fluxograma do Módulo Genético

4.4.1 A Representação dos Dados Geográficos – A malha viária

O sistema PROTEUS representa os dados geográficos na forma de um grafo direcionado, onde as ruas são representadas pelas arestas do grafo, com os pesos das arestas representando as distâncias entre os vértices da aresta, e os vértices representando cruzamentos. Este grafo está representado na base de dados do sistema por um conjunto de tabelas, que representam os pontos (cruzamentos - Figura 9(a)), associados com a sua posição

no mapa, segmentos de ruas (arestas - Figura 9(b)) e as características subjacentes a cada segmento (tipo de pavimentação, velocidade máxima permitida no trecho, existência de lombada, preferenciais e semáforos - Figura 9(c)). Estes dados serão utilizados para a avaliação posterior pelo módulo RBC. O comprimento de cada segmento de rua não está explicitamente armazenado, porém é calculado com base no posicionamento dos vértices que compõem a aresta.



Pontos : Tabela

Nome do campo	Tipo de dados	Descrição
CodPonto	Número	Código que identifica o cruzamento
Descricao	Texto	Quais ruas incidem no cruzamento
X	Número	Coordenada X do cruzamento
Y	Número	Coordenada Y do cruzamento

Propriedades do campo

(a) Tabela de pontos

Pontos_Segmentos : Tabela

Nome do campo	Tipo de dados	Descrição
CodSeg	Número	Código que identifica o segmento
CodPonto	Número	Código que identifica o ponto relacionado ao segmento

Propriedades do campo

(b) Tabela Pontos-Segmentos

Segmentos : Tabela

Nome do campo	Tipo de dados	Descrição
CodSeg	Número	Código do Segmento
Origem	Número	Indica qual cruzamento é a origem do segmento
Destino	Número	Indica qual cruzamento é o destino do segmento
Mao-unica	Sim/Não	Indica se a rua é mão única
Pavimentacao	Número	Indica qual é o tipo de pavimentação do segmento
Localizacao	Número	Indica em que tipo de área o segemento está localizado
Velocidade_Maxima	Número	Especifica qual é a velocidade máxima permitida no segmento
Comprimento	Número	Especifica o comprimento do segmento
PreferencialO	Sim/Não	Indica se o segmento rua é preferencial no cruzamento de origem
PreferencialD	Sim/Não	Indica se o segmento rua é preferencial no cruzamento de destino
SemaforoO	Sim/Não	Indica se o segmento rua possui semáforo no cruzamento de origem
SemaforoD	Sim/Não	Indica se o segmento rua possui semáforo no cruzamento de destino
Lombada	Sim/Não	Indica se o segmento rua possui lombada

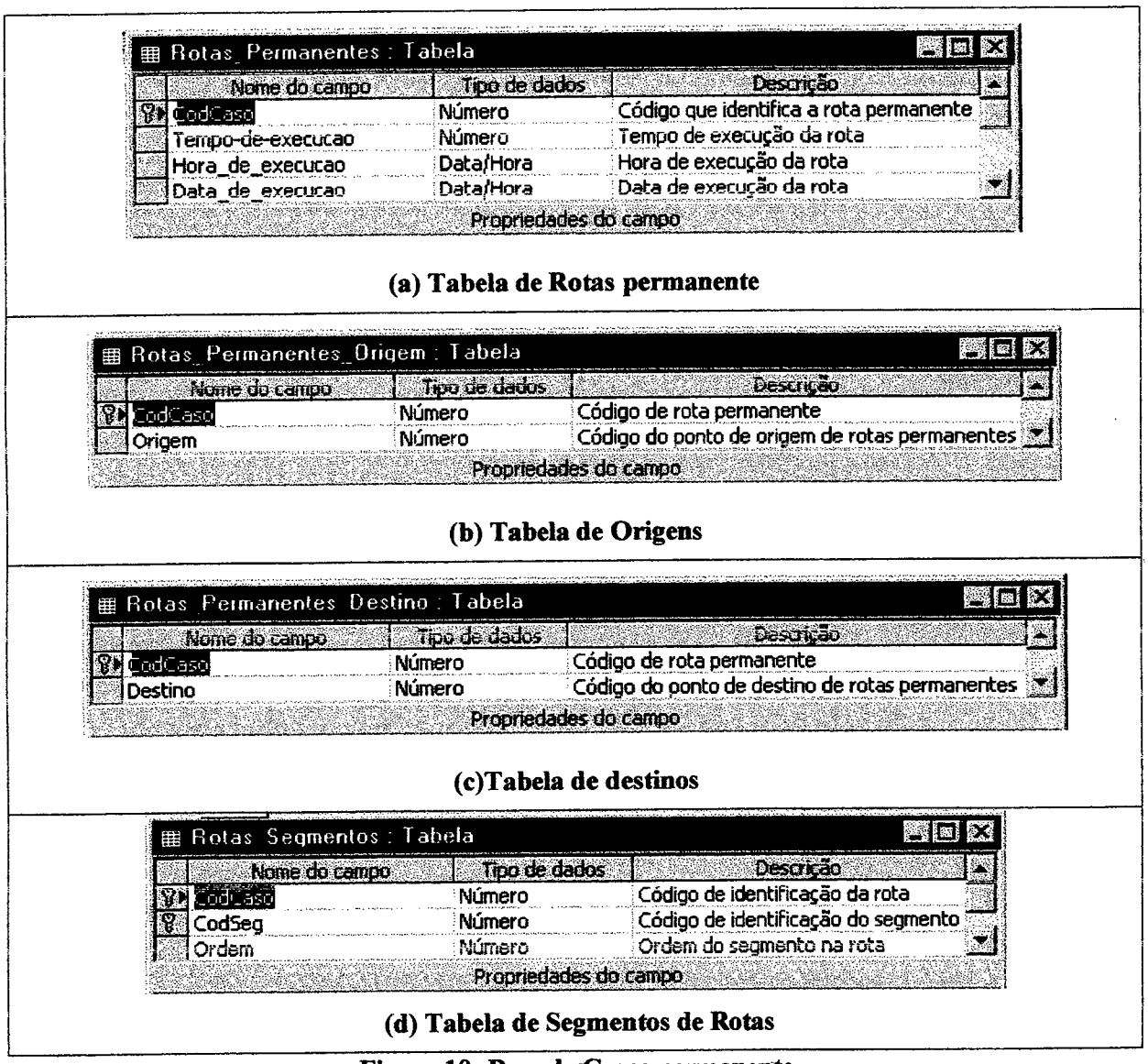
Propriedades do campo

(c) Tabela de Segmentos

Figura 9: Representação da Malha Viária

Outras bases de dados que compõe o sistema são as bases de casos permanentes (Figura 10), que contém as rotas geradas e já avaliadas e armazenadas para

futuras recuperações e adaptações, a base de casos temporária, que contém as rotas geradas pelo algoritmo genético, porém ainda não avaliadas pelo módulo RBC, e a base de perfis, que contém as características e preferências pessoais do usuário, esta base permite que vários perfis sejam armazenados, tornando o sistema bastante flexível.



(a) Tabela de Rotas permanente

(b) Tabela de Origens

(c)Tabela de destinos

(d) Tabela de Segmentos de Rotas

Figura 10: Base de Casos permanente

No Módulo genético a rota é representada na forma de um cromossomo de comprimento fixo, inicializado de acordo com o número de segmentos existente na base de segmentos [Peres et al, 1997], [Peres, 1999]. Esta forma de representação tem a desvantagem do consumo de memória muito elevado, pois ainda que um segmento não faça parte da rota

ele precisa estar representado no cromossomo. Em contrapartida, esta representação fornece um acesso bastante rápido aos dados, sendo que estes precisam ser carregados apenas uma só vez na memória, para formarem o cromossomo (Figura 11).

4.4.2 A Estrutura do Algoritmo Genético

O papel principal do algoritmo genético é encontrar os menores caminhos possíveis entre os pontos de origem e destino selecionados pelos usuários. Esta busca é realizada utilizando-se de uma estratégia evolutiva implementada na forma de um algoritmo genético.

Na literatura encontram-se algumas formas de representação de grafos que são utilizadas por algoritmos evolucionários, as quais representam as rotas como códigos binários [Withley, Starkweather and Shanner, 1991], abordando esta questão permitindo todas as combinações possíveis entre quaisquer dois pontos. Esta abordagem mostrou-se eficiente, mas apenas nos casos em que o grafo é totalmente conexo. Malhas viárias reais geralmente não podem ser representadas na forma de um grafo totalmente conexo, de tal modo que isto torna esta abordagem inviável em virtude de sua elevada complexidade computacional. [Hopcroft e Ullman, 1979] [Cormen, 1990].

Outro ponto que faz com que esta abordagem se torne difícil de ser aplicada para problemas reais se deve à quebra dos esquematas, que neste ponto significam os segmentos que fazem parte da rota. A aplicação do operador de cruzamento causa a destruição de trechos de rotas que não poderiam ser destruídos por já representarem parte do caminho que deveria ser seguido. Isto causa uma demora muito grande na geração das rotas, e mesmo faz com que nenhuma rota seja gerada dentro de um limite de gerações razoável. Assim sendo, optamos por um esquema evolutivo de reprodução assexuada, no qual cada geração que substitui a anterior sofre apenas mutação, o que preserva a maior parte da rota que já foi gerada na geração anterior. Neste trabalho desenvolvemos um esquema de representação independente de ordem, que pode ser aplicado a qualquer tipo de grafo conexo. Este esquema garante a modelagem tanto de grafos orientados, como de não orientados.

O algoritmo genético começa gerando a população inicial de rotas de forma aleatória. Cada rota é um conjunto de segmentos orientados, que representam caminhos de crescimento gradual. A adição de segmentos à cada uma das rotas representadas por cada

indivíduo é guiada pela distância euclidiana entre os nós candidatos a entrar na rota e o ponto de destino da rota. Esta é montada em um processo contínuo, partindo do ponto inicial, e para quando se atinge o ponto de destino ou se esgota o número máximo de gerações fixado pelo algoritmo.

Neste módulo, cada rota é representada conforme mostrado na Figura 11, onde cada segmento do grafo que representa a malha viária é representado por um alelo, cada alelo pode receber um dentre três valores possíveis: {0, 1, *}, onde '0' significa que o segmento representado pelo alelo não pertence a rota, '1' indica que o segmento é um candidato a pertencer a rota, e '*' indica que o segmento já faz parte da rota.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
1	0	0	0	*	1	0	1	1	*	*	1		1	1	*	...

Figura 11: Rota representada na forma de um cromossomo

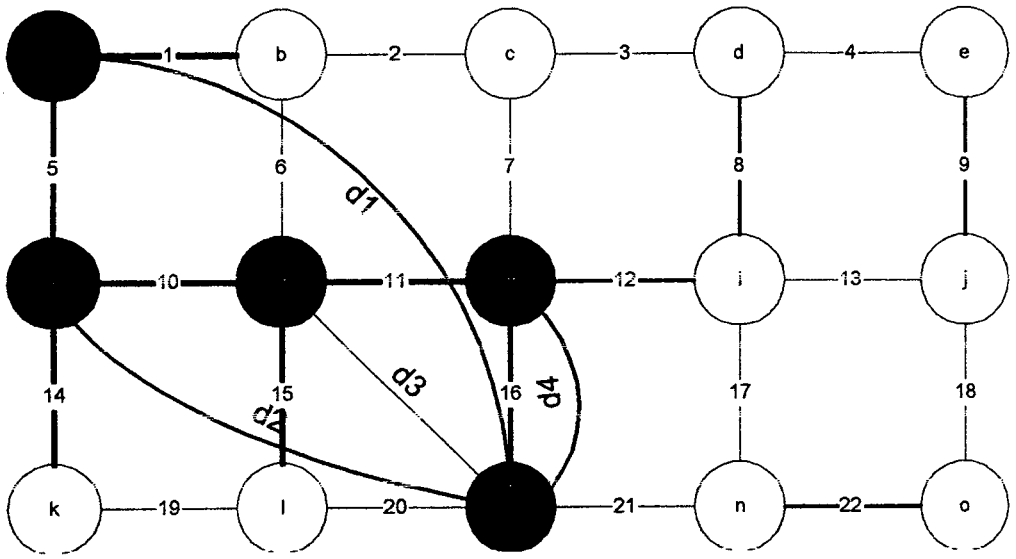


Figura 12: A rota do exemplo da Figura 11 representada na forma de um grafo. Os nós vermelhos representam os pontos de origem (nó "a") e destino ("m"); os nós em verde são os pontos que fazem parte da rota (f, g, h); Os segmentos em verde representam os segmentos que fazem parte da rota; Os segmentos em azul representam os nós candidatos (segmentos c/ valor 1 que estejam ligados ao último ponto da rota atual) e os nós vermelhos e pretos representam os nós não candidadatos. Os vermelhos representam segmentos com valor 1, porém não conectados ao último segmento da rota atual, e os pretos representam os nós valorados com 0.

Uma função fitness é aplicada a todos os alelos valorados com 1, para verificar se os mesmos têm condições de serem adicionados à rota. O teste consiste em verificar se os

alelos valorados com um, quando adicionados à rota, fazem com que a distância euclidiana entre o seu último ponto e o ponto de destino diminua, como mostra a Figura 12. Nela podemos verificar que a medida que cada segmento é incluído na rota, a distância entre o último ponto do segmento inserido e o ponto de destino torna-se menor. Esta distância inicial entre os pontos de origem e destino é dada por d1, e após a inserção do segmento 5 na rota, o ponto final desta nova rota está mais perto do destino que o ponto anterior (ponto de origem, antes da inserção do segmento5), o que significa que estamos nos aproximando do ponto de destino desejado. A escolha entre os segmentos candidatos para saber quem integrará a rota é feita selecionando-se sempre o candidato que fará esta distância diminuir. Assim teremos sempre d1 > d2 > d3 > d4, e a rota caminhará para o seu destino.

A menor das distâncias euclidianas entre os candidatos faz com que este seja convertido para '*', de acordo com o seguinte critério: Entre todos os candidatos a fazer parte da rota tome aquele que pode ser conectado à rota corrente, e cuja distância euclidiana até o destino é a menor, e converta-o para '*'. Após um determinado número de gerações, verifica-se quantos dos cromossomos alcançaram os pontos de destino, avalia-se a sua similaridade de acordo com o perfil do usuário, e apresenta-se as três menores rotas mais similares.

4.5 Exemplos

Serão apresentados nesta seção exemplos da utilização do sistema, os exemplos mostrados aqui enfocam apenas a geração de rotas. Exemplos sobre a recuperação de rotas podem ser encontrados em [Peres et al., 1997].

4.5.1 Geração de Novas Rotas

O protótipo inicia a sua execução pela sua janela principal (Figura 13), onde estão presentes três caixas de texto, que servem como entradas para o sistema. Como entradas o sistema espera por um identificador do ponto de origem, um identificador do ponto de destino e a categoria do usuário que está utilizando o sistema naquele momento. A categoria do usuário representa o seu perfil, o qual está armazenado na base de perfis. A Figura 13 mostra a tela do protótipo do sistema.

Neste exemplo escolhemos como ponto inicial da rota a ser gerada o ponto 20, e como ponto final da rota o ponto 90; como categoria de usuário escolheremos a categoria 1, que representa um policial militar motociclista (veja Tabela 2). Após selecionarmos a opção de gerar rotas, o sistema realiza uma busca na sua base de dados permanente e na base de casos temporária, para verificar se já existe uma rota armazenada com estas mesmas características. Em caso positivo, esta rota é apresentada ao usuário. Como em nosso exemplo esta rota ainda não existe em nenhuma das bases de casos, a resposta que obtemos é "nenhum caso recuperado" o que indica que o sistema acionará o módulo genético para que este tente gerar novas rotas .

A medida que as rotas são geradas e analisadas, o sistema fornece também um ranking baseado na sua similaridade

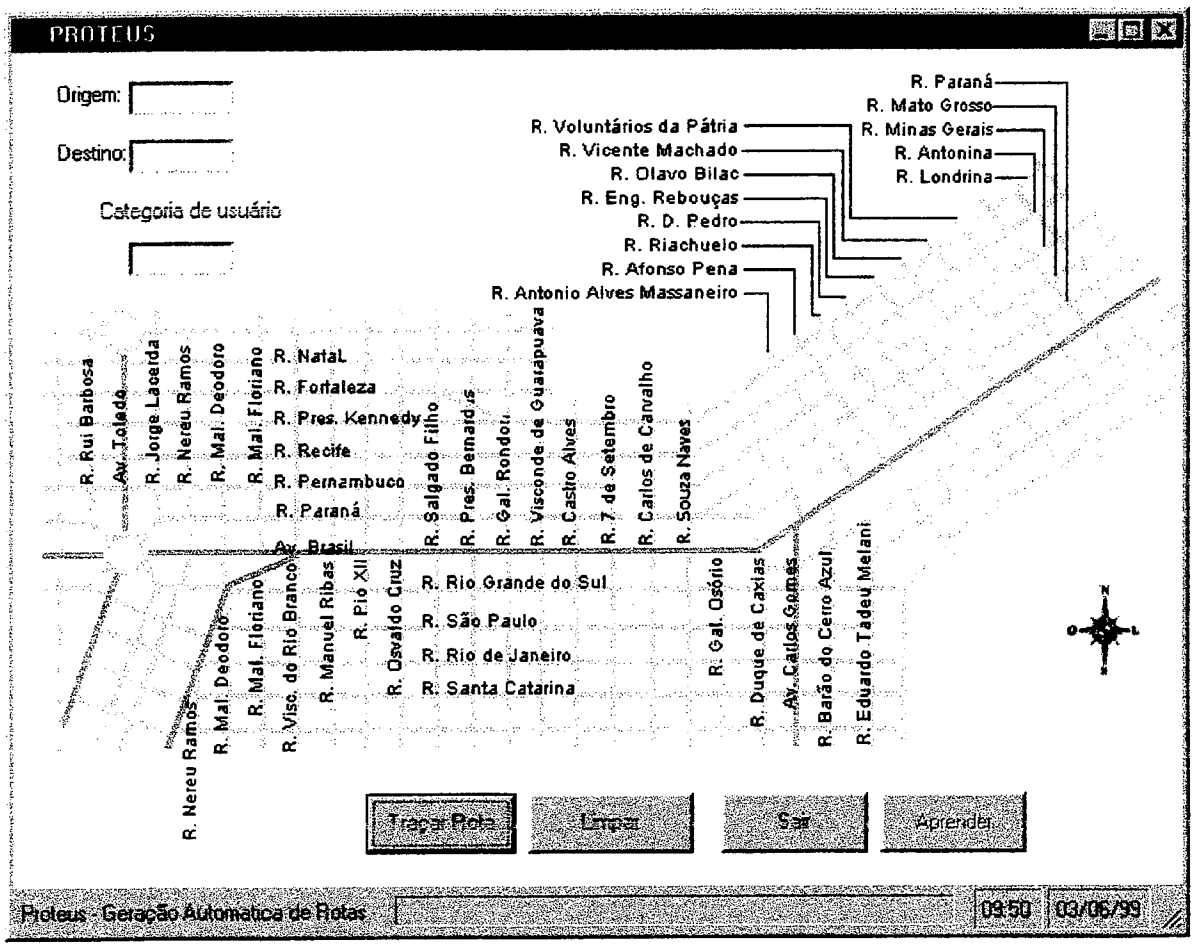


Figura 13: Janela do protótipo PROTEUS

A Figura 14 demonstra a apresentação de uma rota determinada pelo sistema, com a origem e o destino indicados. A caixa de mensagem indica que esta rota foi gerada pelo primeiro indivíduo da população de cromossomos do algoritmo genético. A Figura 15 mostra o número de gerações necessário para se completar todas as rotas, e a Figura 16 mostra o tempo total decorrido para a geração da rota. Neste exemplo em particular, apenas uma rota foi gerada, ao fim das trinta gerações para as quais o programa foi configurado para efeito de teste. É importante ressaltar aqui que o número de gerações pode ser configurado, e foi deixado como trinta apenas para efeito de testes no sistema. Este limite de trinta gerações foi estabelecido de forma empírica, e, em nossos testes, demonstrou ser suficiente para gerar qualquer rota (a exceção de poucos casos) dentro do limite restrito deste protótipo. Um aumento neste número de gerações poderia fazer com que o algoritmo gerasse mais rotas alternativas, em vez de apenas uma, já que a composição dos trechos de rota é aleatória.

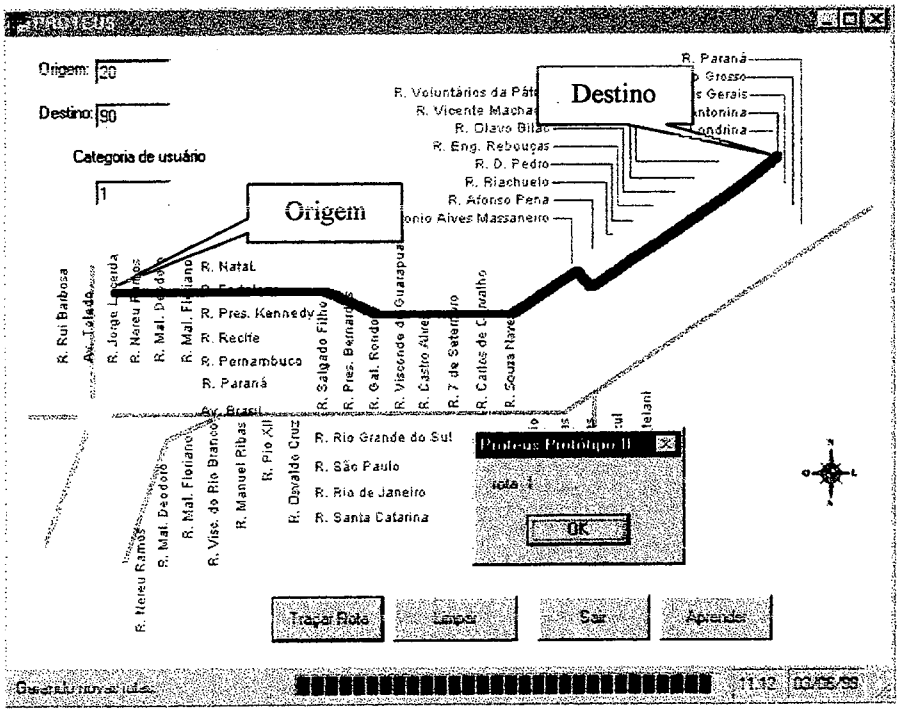


Figura 14: Uma exemplo de rota gerada pelo sistema

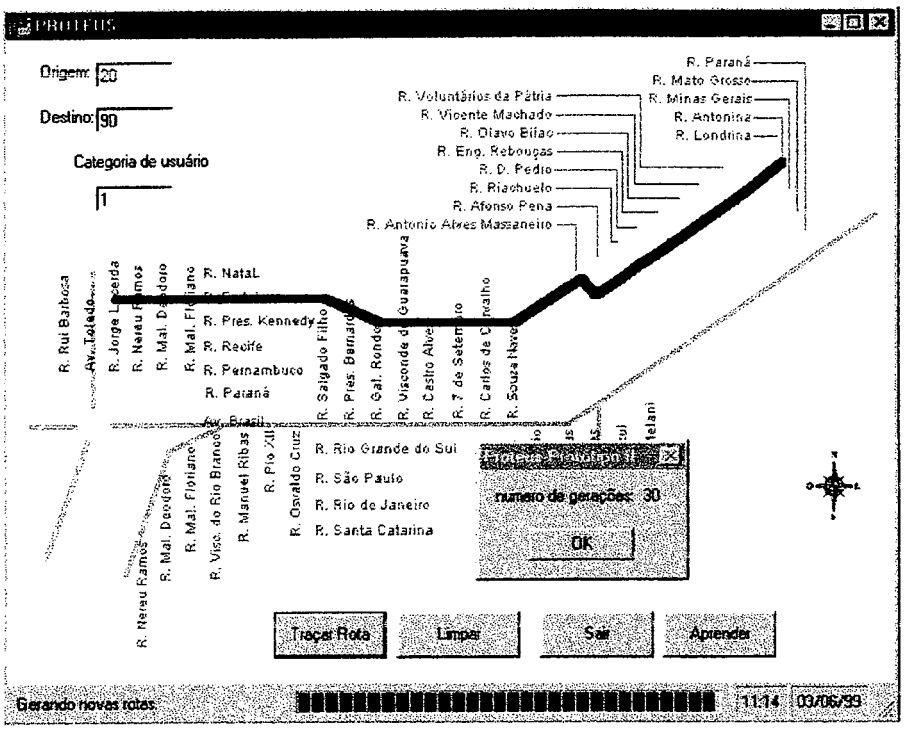


Figura 15: O número de gerações em que a rota foi gerada

A figura a seguir mostra o tempo total, em segundos, decorrido para a geração desta única rota. Como já mencionado anteriormente, o tempo total máximo é constante, e relacionado ao número máximo de gerações. O tempo máximo para obtenção de uma resposta, mesmo nos casos em que o sistema não conseguiu gerar uma rota, com 30 gerações foi de inferior a 800 segundos. Este tempo é bastante razoável, mesmo considerando variações entre as máquinas em que foi executado.

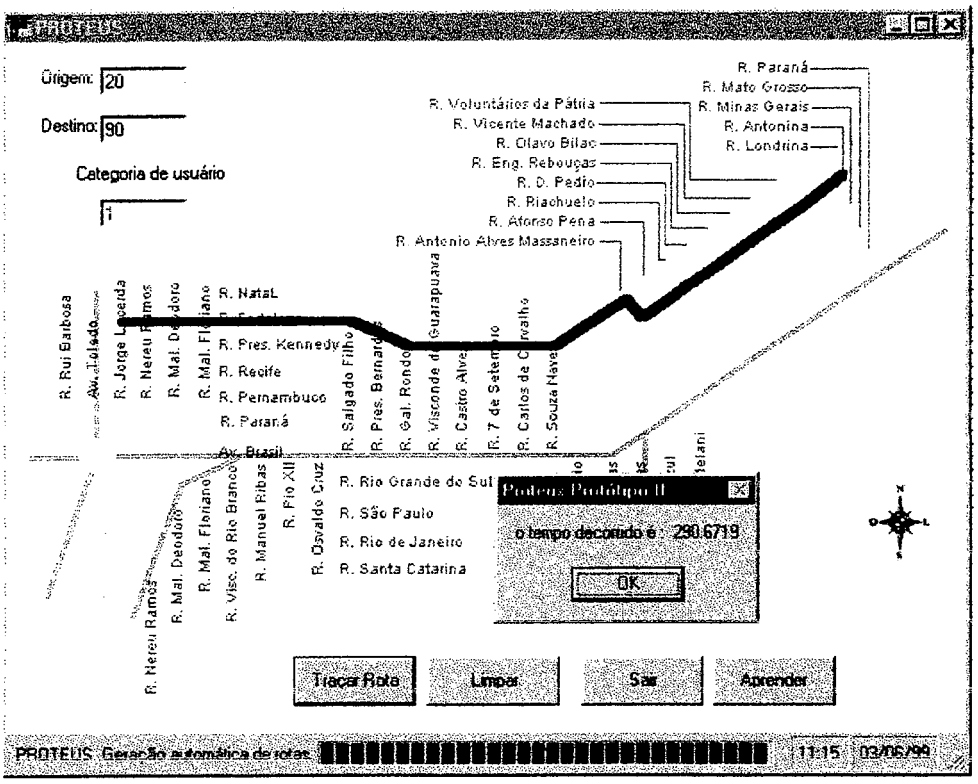


Figura 16: Tempo total decorrido para a geração de uma rota no número máximo de gerações.

Abaixo temos outro exemplo, desta vez uma rota menor, na qual o algoritmo pode encontrar várias rotas possíveis de serem executadas. Também desta vez o algoritmo genético foi chamado porque nenhum caso anterior foi encontrado na base permanente de rotas, nem na base temporária. As figuras 17 e 18 mostram o novo exemplo:

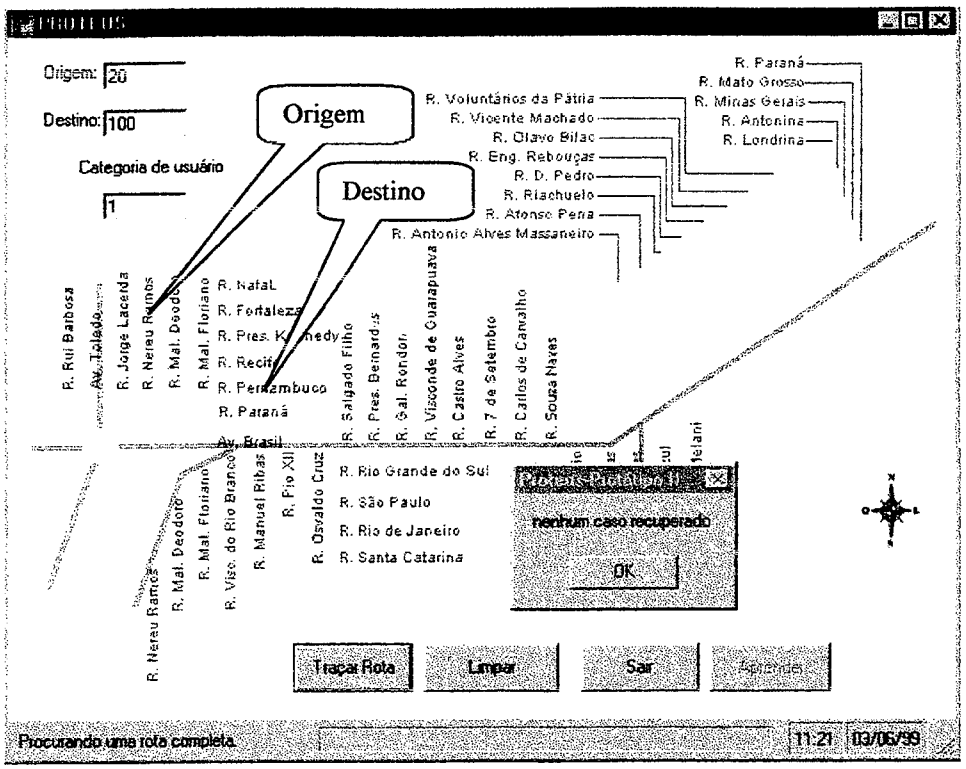


Figura 17: Novo exemplo de geração de rotas.

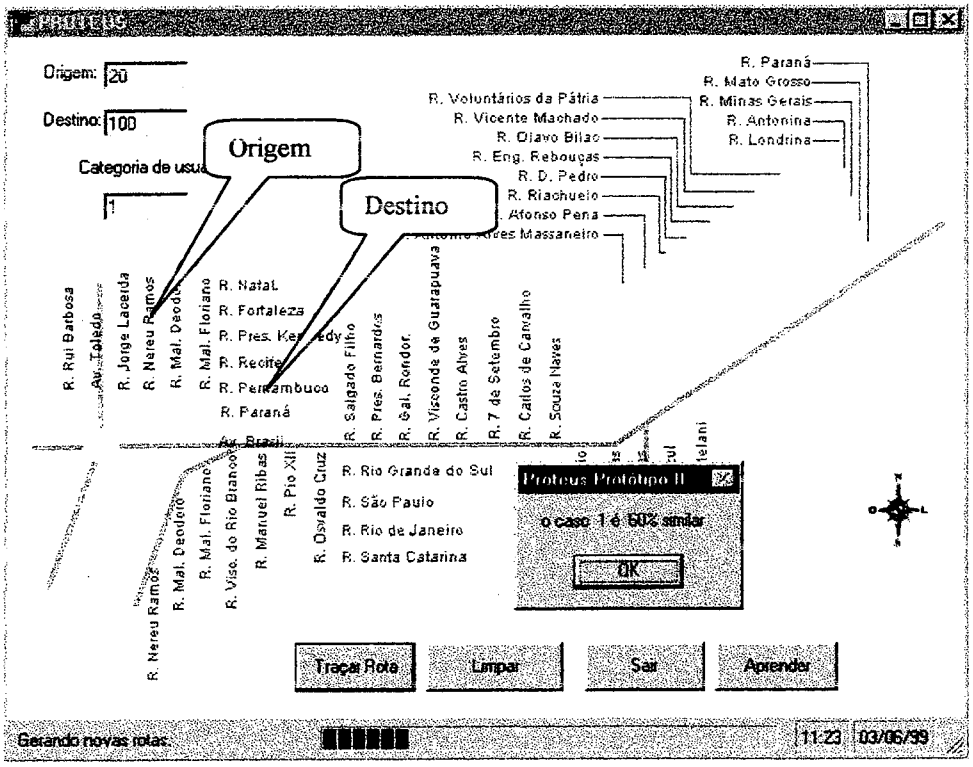


Figura 18: Similaridade da primeira rota gerada

As figuras 19 e 20 mostram a primeira e segunda rotas geradas pelo algoritmo genético.

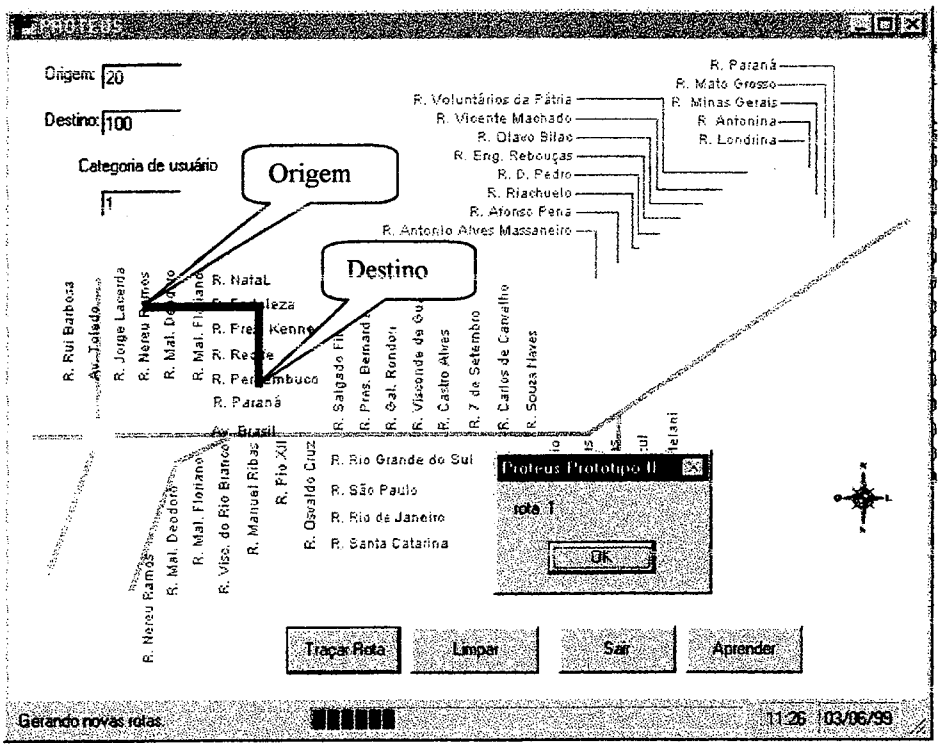


Figura 19: Primeira rota gerada pelo algoritmo genético

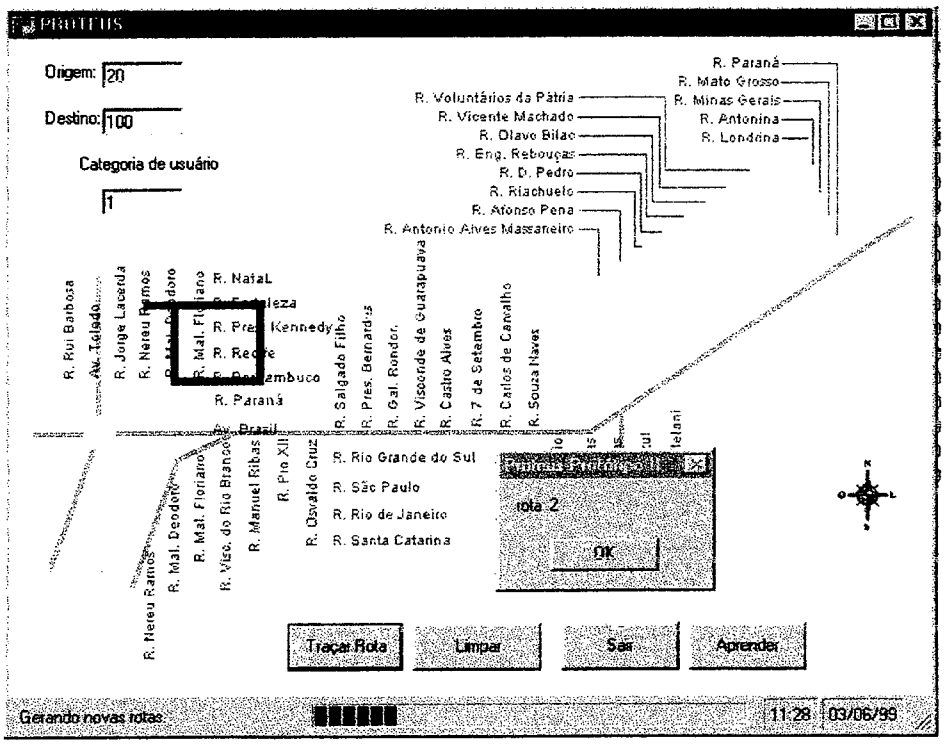


Figura 20: A segunda rota gerada pelo algoritmo genético

A figura 21 mostra a terceira rota mais similar gerada pelo algoritmo genético. Note que a mensagem a descreve como sendo a rota 4. Isto significa que esta rota foi gerada pelo quarto elemento da população de cromossomos.

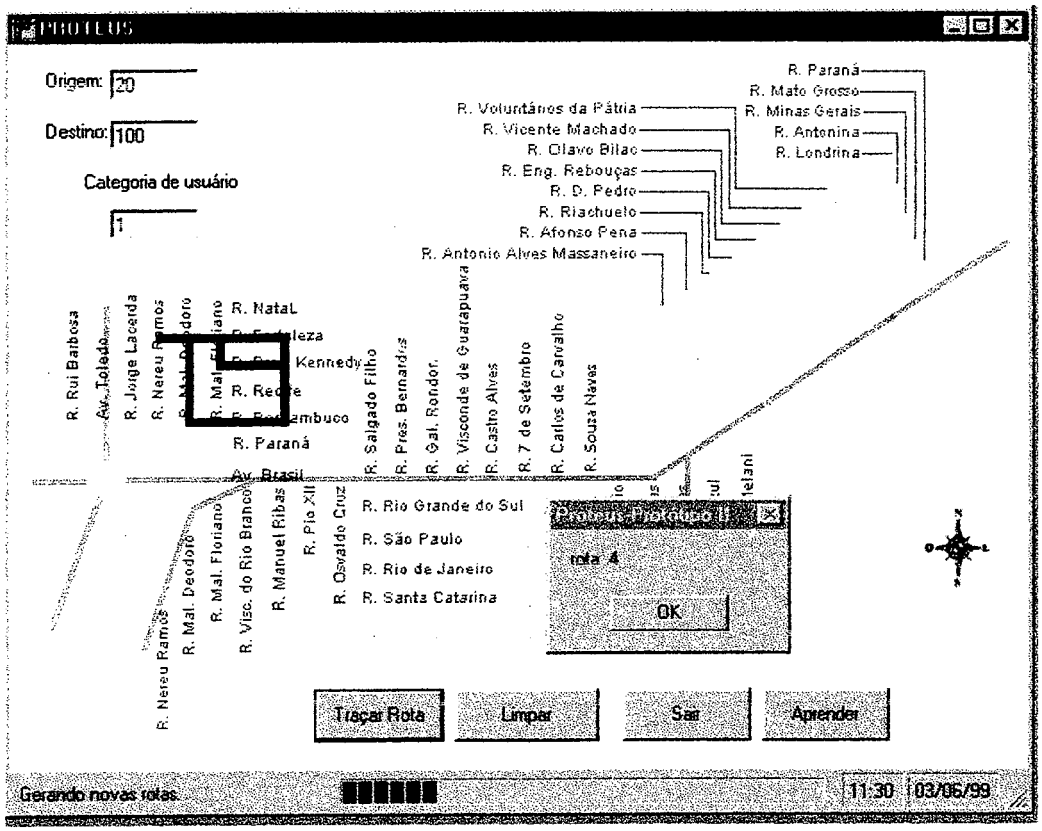


Figura 21: Terceira rota gerada pelo Algoritmo genético

Na verdade, o algoritmo genético é capaz de gerar um número de rotas igual ao de sua população de cromossomos, porém impusemos como restrição exibir apenas as três rotas mais similares ao perfil que tenham sido geradas. Em caso de empate, o primeiro elemento da população é apresentado ao usuário. Esta restrição foi imposta ainda devido ao fato de que apresentar muitas rotas ao usuário o deixaria indeciso. Assim optamos por exibir uma rota principal e no máximo duas rotas alternativas.

As figuras 22 e 23 mostram o número de gerações que foi necessário para se alcançar o resultado mostrado nas figuras anteriores, e o tempo total decorrido para gerar estas rotas.

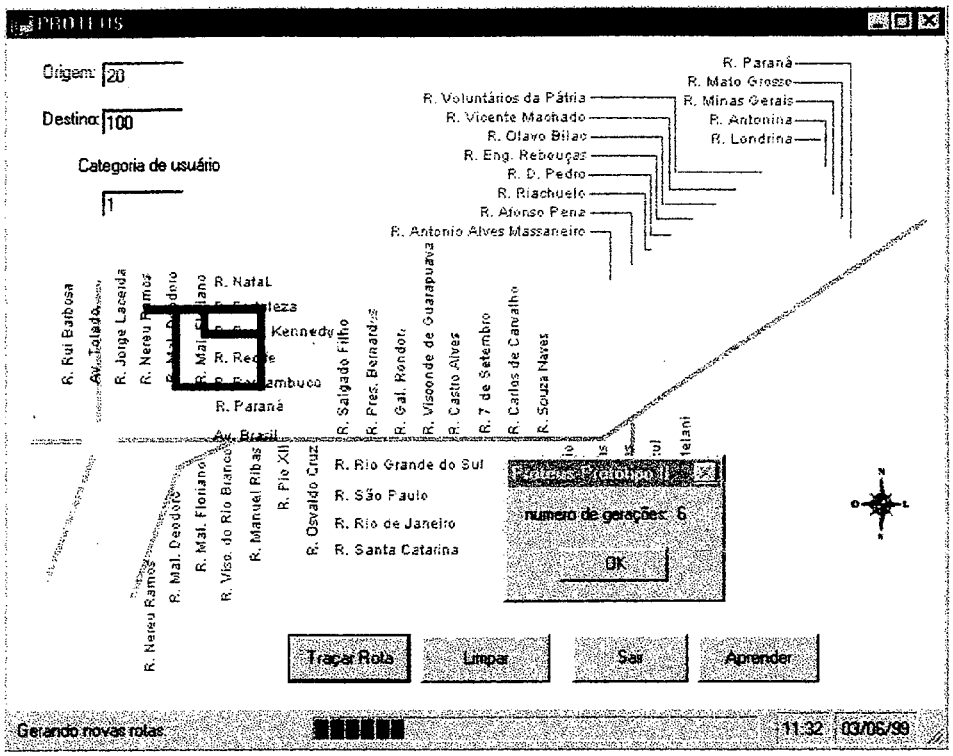


Figura 22: Número de gerações

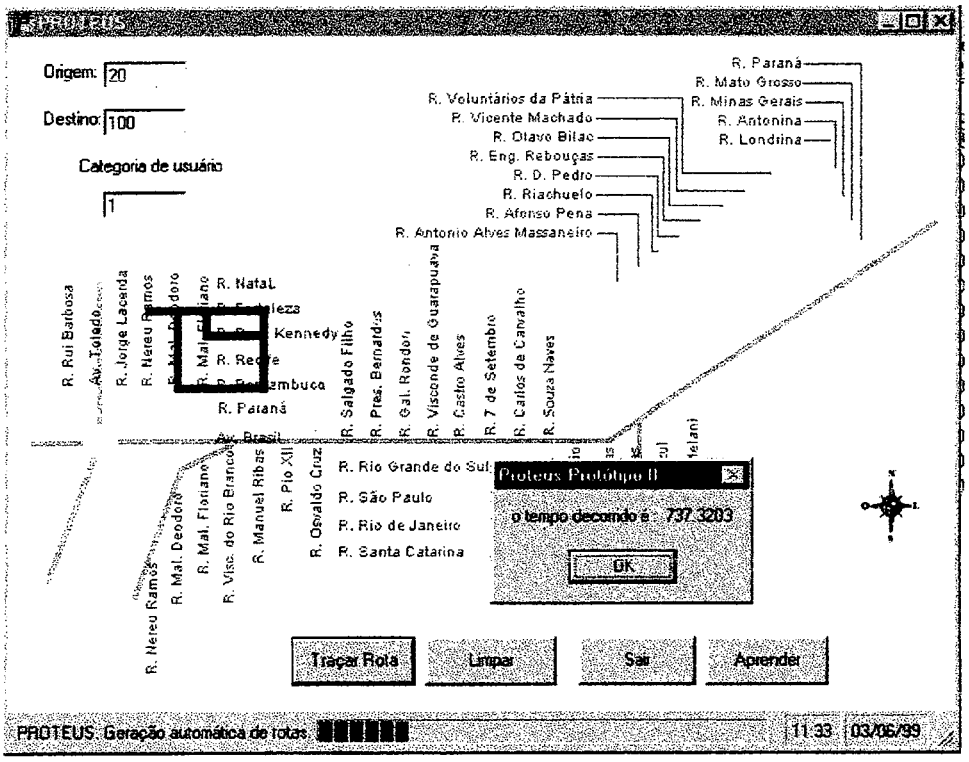


Figura 23: Tempo total decorrido

Para encerrar esta seção de exemplos, forneceremos apenas mais um, onde o tempo de geração das rotas chegou perto do limite máximo de 800 segundos, porém o sistema ainda foi capaz de gerar pelo menos uma solução viável (Figura 24)

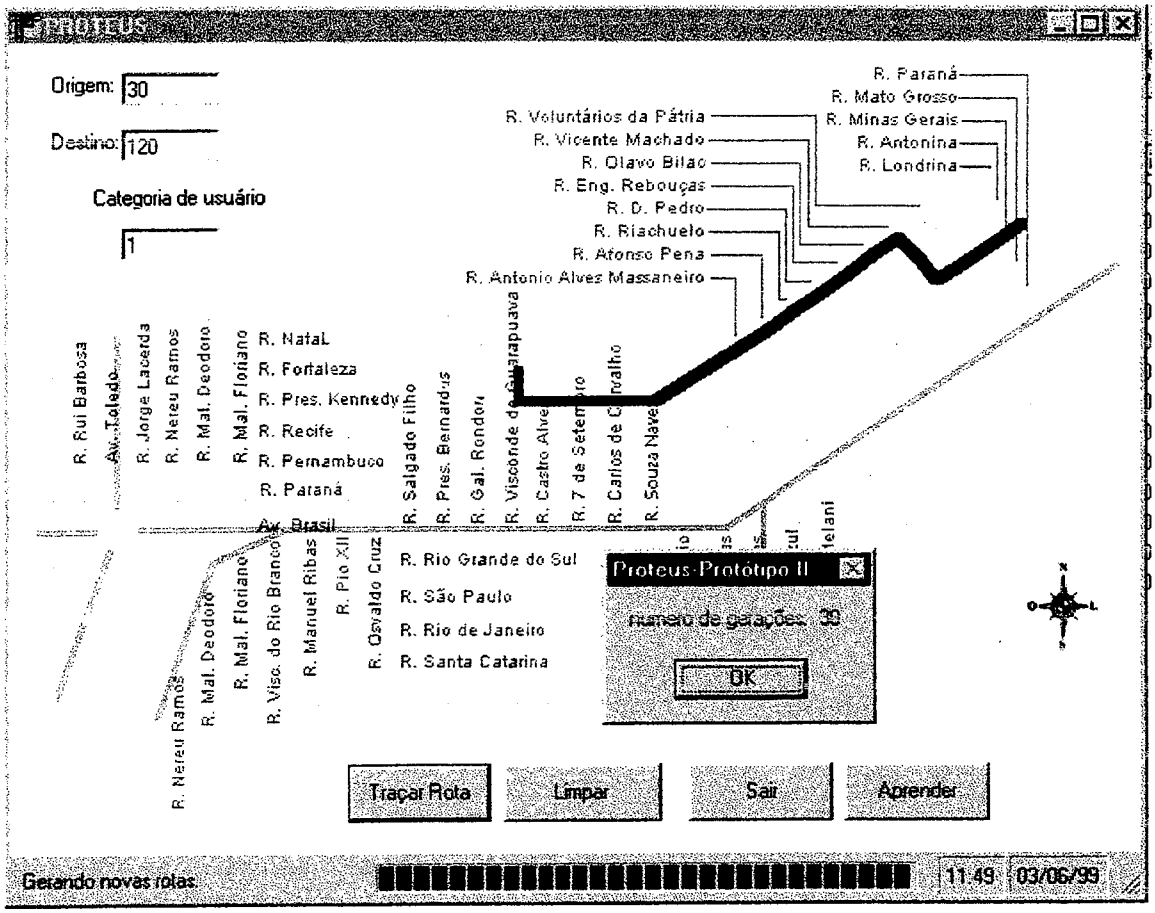


Figura 24: Exemplo de rota gerada no limite de gerações.

Como já citado anteriormentre, percebe-se que, se o número de gerações fosse aumentado, o programa poderia eventualmente oferecer aos seus usuários um leque bem maior de rotas. O mesmo aconteceria aumentando-se o tamanho da população. No protótipo por nós construido estabelecemos empiricamente o valor destes parâmetros, mas uma perfeita calibração destes dois parâmetros torna o programa mais eficiente.

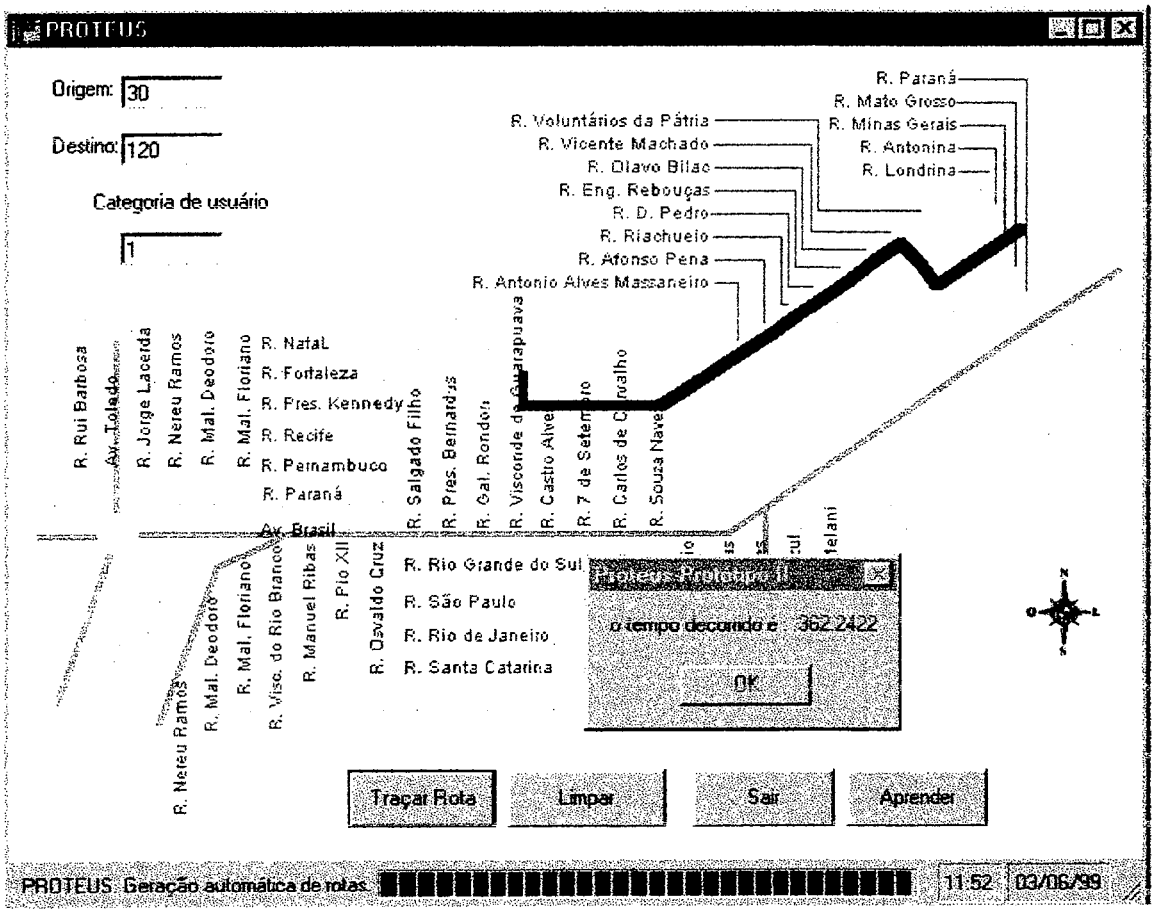


Figura 25: Tempo decorrido para a geração desta rota

4.6 Discussão

Encerramos aqui a apresentação dos exemplos de geração de rotas efetuados pelo sistema. Neste capítulo apresentamos a estrutura do sistema, e como ele integra as duas técnicas – Algoritmos genéticos e RBC – na solução do problema de planejamento de rotas. Os exemplos mostrados aqui enfocam apenas a geração de rotas, que é onde se situa o módulo genético, sendo por assim dizer o núcleo do sistema PROTEUS. Em [Peres, 99] encontram-se exemplos relativos à recuperação de casos previamente armazenados.

É importante ressaltar novamente que, por ser um algoritmo não determinístico, a natureza das soluções encontradas por este algoritmo é diferente das soluções apresentadas pelos algoritmos tradicionais, tornando assim impossível uma comparação entre ambas, em termos de complexidade computacional. Pode-se dizer, que aqui não se busca a "melhor" rota, mas um conjunto de rotas que sejam muito boas, e isto num tempo razoável, o que é a grande vantagem deste sistema com relação aos métodos

tradicionais. Sistemas com esta característica são mais indicados em situações onde o tempo de espera pela geração de uma nova rota é crucial. É importante observar que, na medida em que o sistema aprende novas rotas a cada interação com o usuário, menor será a necessidade de ativação do módulo genético. Isto significa um ganho de performance, pelo fato de que não ocorrerá o overhead de se ter de gerar novamente as mesmas rotas para uma dada situação. Em contrapartida, haverá o crescimento da base de casos, o que certamente degradará a performance da recuperação, em virtude do volume de casos na base. Como as técnicas de gerenciamento de Bancos de Dados atualmente são bastante eficientes, cremos que esta queda de performance não seja muito significativa.

4.7 Resumo

Neste capítulo foi descrito em detalhes a implementação e o funcionamento do módulo genético, o qual tem a responsabilidade da geração de rotas. Também foi apresentada a forma de integração deste módulo ao sistema PROTEUS, e como este se realciona com o módulo RBC. Foram também expostas algumas idéias para aperfeiçoamentos no sistema, com intenção de melhorar a sua performance. No próximo capítulo apresentamos as considerações finais e conclusões sobre a realização deste trabalho.

5. Considerações Finais

Faremos agora as considerações finais sobre o trabalho exposto nesta dissertação. O primeiro ponto a ser considerado nesta seção será um paralelo entre os processos clássicos de planejamento de rotas e a abordagem evolutiva, e a seguir, trataremos um pouco sobre as vantagens do modelo híbrido construído; Por fim apresentaremos algumas idéias que poderiam ser utilizadas na realização de trabalhos futuros.

5.1 Genéticos X Otimização Clássica: Vantagens e Limitações

Atualmente existem vários algoritmos tradicionais que resolvem problemas de planejamento de rotas, como por exemplo, o problema do caixeiro viajante, e muitas de suas variações e os problemas de caminho. Todos estes algoritmos têm em comum entre si um problema: O algoritmo tende a tornar-se ineficiente a medida que o número de nós aumenta. Este tipo de problema é conhecido como um problema NP-completo [Hopcroft e Ullman, 1979]. Os problemas do tipo do caixeiro viajante podem ser vistos como uma extensão, ou variação do problema de se encontrar um ciclo hamiltoniano em um grafo; o problema do caixeiro viajante pode-se resumir então a encontrar o menor dos ciclos hamiltonianos porventura existentes no grafo.

A importância da classe de problemas NP-completos reside no fato de que ela inclui muitos problemas que são naturais no mundo real, e que tem demandado sérios estudos na busca de soluções eficientes. Não é sabido se nenhum destes problemas tem uma solução em tempo polinomial. Uma outra classe de problemas de grande importância que também é NP-completo é a classe de problemas resolvidos pela programação linear inteira (Inclusive um dos problemas resolvidos por esta técnica é o problema do caixeiro viajante).

O ponto fundamental a ser considerado, é que os algoritmos clássicos funcionam bem para um número pequeno de vértices, mas quando este número aumenta, o seu tempo de execução sobe a níveis elevadíssimos, tornando a busca pela solução ótima inviável. Muitos algoritmos tentam amenizar este fato através do uso de heurísticas, que ajudam a descartar algumas das possibilidades que devem ser consideradas na solução do problema. Com a utilização do algoritmo desenvolvido aqui, conseguimos um bom

desempenho na busca de soluções eficientes e em um espaço de tempo razoável de espera para o usuário. O algoritmo não garante que a melhor solução é encontrada, mas apresenta boas soluções para o problema, o que muitas vezes é suficiente.

Uma das limitações enfrentadas pelo módulo genético diz respeito à representação das rotas na forma de cromossomos. A representação utilizada aqui apesar de eficiente e rápida, consome um espaço de memória bastante grande, o que poderia inviabilizar a execução deste algoritmo em um ambiente onde este recurso seja escasso. Uma possível solução para este problema poderia ser a resolução do problema em partes, dividindo-se a malha viária em regiões de tamanho razoável, ligadas por certos corredores. O algoritmo então resolveria o problema entre as regiões, procurando primeiro como chegar em uma determinada região, possivelmente passando por outras regiões (orientado pela distância euclidiana entre os centros das regiões) e então, uma vez dentro da região desejada, o algoritmo daí encontraria o ponto de destino.

Uma outra limitação do algoritmo genético é que, por ser de natureza não determinística, não podemos garantir que uma solução será encontrada dentro do número de gerações especificado. Há casos em que, devido à aleatoriedade do processo, após o número de gerações especificado, ainda não há nenhuma rota completada. E algumas vezes, duas ou três gerações depois a resposta é encontrada. O mesmo se dá com respeito ao número de indivíduos na população. Um número muito grande de indivíduos diminui a performance do algoritmo, mas garante uma diversidade maior de possíveis soluções, o que poderia mais tarde garantir que uma solução muito boa tenha sido encontrada. É necessário uma série de testes empíricos para determinar os valores destes parâmetros, levando em consideração não apenas a performance do algoritmo em termos de tempo, mas também em termos da adequação da própria rota encontrada. É necessário um certo equilíbrio entre velocidade e eficácia.

5.2 Vantagens do Modelo Híbrido

O sistema completo apresentado aqui compõe-se de dois módulos, um responsável pela geração das rotas e outro pelo seu armazenamento e recuperação. Pode-se pensar que, já que o módulo de geração de rotas apresentou bom desempenho, porque então a necessidade de se armazenar e recuperar rotas? Uma boa resposta a esta pergunta pode ser

respondida pelo fato de que, desta forma se pode evitar o re-trabalho, ou seja, se uma rota foi gerada uma vez, e a sua execução mostrou-se satisfatória, é provável que ela seja reutilizada várias vezes até que alguma situação nova faça com que uma outra alternativa de rota seja solicitada.

Outra resposta, é que as pessoas têm uma tendência natural à adotarem uma certa rotina, ou seja, a maioria das pessoas tem uma grande tendência a seguir sempre o mesmo caminho, de casa para o trabalho, do trabalho para casa, etc. Um novo caminho só é procurado quando surge alguma situação nova, e este novo caminho somente se tornará o novo caminho rotineiro caso ele demonstre ser mais vantajoso (em termos de tempo ou custo).

O sistema híbrido apresentado aqui comporta-se mais ou menos da mesma esr
forma que uma pessoa quando tem que tomar uma decisão a respeito de qual caminho seguir: Se é a primeira vez que este caminho tem que ser executado, a pessoa senta e planeja qual a melhor rota. Uma vez planejada esta rota, sempre que for necessário se repetir este caminho, a pessoa apenas o recupera em sua memória, ou no seu mapa. Se alguma situação impedir o tráfego em algum trecho, a pessoa tentará primeiro adaptar este trecho, fazendo um desvio ou tomando um atalho. Somente se a rota inteira se mostrar ineficiente, ou alguma nova situação tornar impraticável grande parte da rota é que esta pessoa sentará novamente à frente de seu mapa para planejar uma nova rota. Acreditamos ter conseguido modelar este tipo de comportamento neste sistema.

O comportamento acima demonstrado pelo sistema parece mais natural a um a u
ser humano do que se tivéssemos apenas utilizado o algoritmo genético para gerar sempre uma nova rota: pessoas tendem a não confiar em um sistema que pode lhe dar respostas diferentes diante da mesma situação; criamos assim um sistema flexível, que permite ao usuário escolher, dentre um conjunto de rotas muito boas, a que mais lhe agrada, como também decidir quando gerar uma nova rota. Acreditamos assim ter construído um sistema bastante confiável e robusto.

Outro aspecto a ser mencionado, é quanto à utilização do sistema para malhas diferentes. Quanto a isso, o sistema é bastante flexível, bastando capturar a malha desejada através de um scanner ou outro dipositivo, e alimentar o banco de dados que descreve as características de cada segmento de rota. Este processo pode ser bastante trabalhoso e consumir bastante tempo, pois estas informações são bastante numerosas, e difíceis de

encontrar caso a cidade ou região em questão não possuir um bom sistema de cadastro urbano. Outra fonte de problemas refere-se ao fato de que este tipo de informação sofre uma desatualização muito rápida, principalmente em cidades ou regiões onde o crescimento urbano é muito grande. Manter a base de informações sobre a malha viária atualizada demandaria muito trabalho e esforço.

5.3 Recomendações para Trabalhos Futuros

Apresentamos a seguir algumas sugestões e idéias que poderiam ser usadas no desenvolvimento de trabalhos futuros relacionados à esta área, que também poderiam servir como complemento ao trabalho realizado nesta dissertação.

Uma primeira idéia seria estudar outras formas de representação para a malha viária na forma de cromossomos, buscando uma representação que seja mais econômica em termos de espaço de armazenamento. Isto se refletiria diretamente na performance do algoritmo, permitindo a sua atuação em malhas maiores.

Poderia ser realizado um estudo sobre outras heurísticas para a orientação da busca: neste trabalho usou-se a distância euclidiana para orientar o direcionamento da rota em direção ao destino. Poderia-se experimentar o uso de outras heurísticas, ou mesmo um pequeno sistema especialista com um pequeno conjunto de regras (que poderia ser baseado na lógica difusa) para fazer esta orientação. Este módulo difuso poderia ser utilizado para orientar a busca em termos de variáveis linguísticas tais como "perto" e "longe", utilizando-se inclusive de modificadores como "muito" e "pouco".

Seria interessante também pensar em adaptar este algoritmo para trabalhar em malhas viárias de grande extensão, dividindo-se esta malha em regiões (como mencionado acima) e adaptar o módulo de raciocínio baseado em casos para realizar a recuperação da rota nesta nova forma de representação. Esta adaptação traz uma série de novos problemas, como por exemplo, estabelecer pontos de interconexão entre uma região e outra, de tal forma que a passagem por uma região não implique no aumento da distância, ou mesmo saber como fazer para saber sobre quais regiões a rota irá ou não passar. Isto oferece um grande campo ainda por ser explorado.

Ao fim deste trabalho, pudemos observar o poder que as técnicas evolutivas têm para a resolução de problemas complexos, mas percebemos que um dos problemas está ainda na forma da representação da estrutura de dados necessária para que o algoritmo funcione de forma eficiente, eficaz e efetiva. Esperamos com este trabalho ter contribuído para o desenvolvimento destas área tão interessantes que são a área de transportes e a área da computação evolutiva.

6. Referências Bibliográficas

Aho, A. V.; Ullman, J. D. "Foundations of Computer Science". C. Ed. New York; Computer Science Press, 1995.

Assad, A. A. "Modeling and Implementation Issues in Vehicle Routing" in Golden, B. L.; Assad, A. A. "Vehicle Routing: Methods and Studies". Amsterdam; Elsevier Science Publishers. (1988). pp. 7-45.

Bernstein, David and Kornhauser, Alain, "Map Matching for Personal Navigation Assistants". The Transportaion Research Board, 77th Annual Meeting, January 11-15, 1998, Washington, D.C.

Christofides, N. "Vehicle Routing" in Lawler, L. E. et al. "The traveling Salesman Problem: A Guided Tour of Combinatorial Optimization". Great Britain; John Wiley & Sons; 1985; pp. 431-448.

Cormen, T. H.; Leiserson, C. R.; Rivest, R. L. "Introduction to Algorithms". 1th. Ed. USA; MIT Press; 1990.

Goonatilake, S.; Khebbal, S.. "Intelligent Hybrid Systems"; 1th ed. England; John Wiley & Sons; 1995.

Hillier, S. H.; Gerald, J. L.; "Introduction to Operations Research"; 1th ed.;USA; McGraw-Hill, 1995.

Hoffman, A. J.; Wolfe, P.; "History" In Lawler, L. E. et al.; "The Thavelling Salesman Problem: A Guided Tour of Combinatorial Optimization"; Great Britain; John Wiley & sons, 1985; pp. 1-15.

Holland, J.H. (1975) "Adaptation in natural and artificial systems", Ann Arbor, MI: The University of Michigan Press. [HOLLAND75]: 2ª ed. (1992)

Hopcroft, J. E.; Ullman, J. D.; "Introduction to Automata Theory, Languages, and Computation"; Addison Wesley Series in Computer Science; Reading, Massachussets, 1th ed, 1979.

Jaillet, P.; "The Probabilistic Vehicle Routing Problem" in Golden, B.L.; Assad, A. A.; "Vehicle Routing: Methods and Studies"; Amsterdam; Elsevier Science Publishers B. V, 1988; pp. 293-318.

Kolodner, J.; "Case-Based Reasoning"; 1th ed.; Harcover; Morgan Kaufman Publisher; 1993.

Koza, J. R.; "Genetic Programming: On the Programming of Computers by Means of Natural Selection; 1th ed.; USA; MIT Press, 1992.

Medsker, L. R.; "Hibrid Intelligent Systems"; 1th ed.; USA; Kluwer Academic Publisher, 1995.

Mühlenbein, H.; "Evolution in Time and Space – The Parallel Genetic Algorithm"; in: "Foundations of Genetic Algorithms"; USA; Morgan Kaufman Publishers, 1991.

Padmos, Jeremy and Bernstein, David, "Personal Travel Assistants and The World Wide Web". The Transportation Research Board, 76th Annual Meeting, January 12-16, 1997, Washington, D.C.

Peres, S. M. et al.; "A Hybrid System For Route Planning"; in: AAAI Spring Symposium Series; Technical Report, Stanford, 1997; pp. 112-116.

Peres, S. M.; "Raciocínio Baseado em casos para avaliação de Planos de Rotas" Dissertação de Mestrado desenvolvida no Departamento de Engenharia de Produção e Sistemas da Universidade Federal de Santa Catarina; 1999.

Powell, W. B., "A comparative Review of Alternative Algorithms For The Dynamic Vehicle Allocation Problems". in: Golden, B., L.; Assad, A. A.; "Vehicle Routing: Methods and Studies"; Amsterdam; Elsevier Science Publishers, 1988; pp. 249-291

Psarafits, H. N.; "Dynamic Vehicle Routing" in Golden, B. L.; Assad, A. A.; "Vehicle Routing: Methods and Studies"; Amstredam; Elsevier Science Publishers, 1988; pp. 223-248.

Scott, Kelley and Bernstein, David, "Solving a Best Path Problem When the Value of Time Function is Nonlinear". The Transportation Research Board, July, 1997

Scott, Kelley et al., "Finding Alternatives to the Best Path". The Transportation Research Board, 76th Annual Meeting, January 12-16, 1997, Washington D.C.

Spears, W. M., et al, (1993) "An Overview of Evolutionary Computation". In Proceedings of European Conference on Machine Learning, pp. 442-459.

Withley, D.; StarkWeather, T.; Shaner, D.; "The Travelling Salesman and The Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination", in "HandBook of Genetic Algorithms"; 1th ed; New York; 1991; pp. 350-372.

7. Bibliografia

Castro, J.L., Delgado, M. and Herrera, F. "A Learning Method of Fuzzy Reasoning by Genetic Algorithms". Proceedings of First European Congress on Fuzzy and Intelligent Technologies, (EUFIF'93), vol. II, pp 804-809, Aachen, Germany, september, 7-10, 1993.

Cordón, O., Herrera, F., Lozano, M. "A Classified Review on The Combination Fuzzy Logic-Genetic Algorithm Bibliography" Technical Report #DECSAI 95129, December, 4, 1996

De Jong, K. A., and Spears, W. M., (1990) "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms". In International Workshop Parallel Problem Solving from Nature, University of Dortmund, pp. 38-47.

De Jong, K.A. and Spears, W. M., (1992) "A Formal Analysis of the Role of Multi-Point Crossover in Genetic Algorithms. In Annals of Mathematics and Artificial Intelligence Journal, vol. 5, n° 1, pp. 1-26.

De Jong, K. A., and Spears, W. M., (1993) "On the State of Evolutionary Computation". In Proceedings of the 1993 International Conference on Genetic Algorithms, Urbana-Champaign, IL, pp. 618-623.

Golden, B. L.; Assad, A. A. "Vehicle Routing: Methods and Studies". Amsterdam: Elsevier Science Publisher B. V., 1988.

Haigh, K. R.; Shewchuck, J. R.; Veloso, M. M.; "Exploiting Domain Geometry In Analogical Route Planning"; Journal of Experimental and Theoretical Artificial Intelligence, nº 9; pp. 509-541; 1997.

Haigh, K. R.; Veloso, M. M.; "Combining Search and Analogical Reasoning in Path Planning From Road Maps". In AAAI-94 Workshop: Case Based Reasoning: Working Notes; Washington: AAAI Press; jul. 1993; pp. 182-187.

Haigh, K. R.; Veloso, M. M.; "Route Planning by Analogy; In ICCBR-95; Proceedings Case-Based Reasoning Research and Development; Sisembra; Springer Verlag, Oct.

1995; pp. 169-180.

Herrera, F. and Lozano, M. (1996) "Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers, In Herrera, F. and Verdegay, J. (eds) Genetic Algorithms and Soft Computing, pp. 95-125, Physica Verlag.

Herrera, F. and Herrera-Viedma, E. "Aggregation Operators for Linguistic Weighted Information", Technical Report #DECSAI-95120, October, 1995

Johnson, D. S.; Papadimitriou, C. H.; "Computational Complexity". In: Lawler, L. E. et al.; "The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization"; Great Britain: John Wiley & Sons; 1985; pp. 37-85.

Johnson, D. S.; Papadimitrious, C. H.; "Performance Guarantees for Heuristics"; in: In: Lawler, L. E. et al.; "The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization"; Great Britain: John Wiley & Sons; 1985; pp. 145-179

Klir, G. J.; Yuan, B.; "Fuzzy Sets and Logic: Theory and Applications"; 1th ed.; New Jersey; Prentice Hall PTR, 1995.

Lawler, L.et al.; "The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization"; Great Britain; John Wiley &Sons, 1985.

Liu, B.; "Using Knowledge to Isolate Search in Route Finding. 1995.

Liu, B.; "Intelligent Route Finding: Combining Knowledge, Cases and An efficient Search Algorithm"; in: 12th European Conference on Artificial Intelligence; John Wiley & Sons; 1996.

Nakamiti, G.; Gomide, F.; "An Evolutive fuzzy mechanism based on past experiences"; in: Second European Congress on Intelligent Techniques in Soft Computing; Aachen, sep, 1994; pp. 1211-1217.

Rabuske, M. A.; "Introdução à Teioria dos Grafos"; 1ª ed.; Florianópolis; Editora da UFSC, 1992.

Rich, E.; Knight, K.; "Inteligência Artificial"; 2ª ed; Saõ Paulo; Makron Books, 1993.

Russel, S. J.; Norvig, P.; "Artificial Inteligente: A Modern Approach"; 1th ed; USA; Prentice Hall, 1995.

Schultz, A. C., and Grefenstette, J.J., (1990) "Improving tactical plans with genetic algorithms". In Proceedings of IEEE Conference on Tools for Artificial Intelligence, Herndon, VA, pages 328-334. IEEE Society Press.

Schultz, A. C., (1991) "Using a Genetic Algorithm to Learn Strategies for Collision Avoidance and Local Navigation". In Proceedings of the Seventh International Symposium on Unmanned Untethered Submersible Technology, pp. 213-225, University of New Hampshire, Marine Systems Engineering Laboratory.

Schultz, A.C. and Grefenstette, J. J., (1992) "Using a Genetic Algorithm to Learn Behaviors for Autonomous Vehicles". In Proceedings of the American Institute of Aeronautics and Astronautics Guidance, Navigation and Control Conference, pp. 739-749.

Schultz, A., C., Grefenstette, J.J. and De Jong, K.A. (1992) "Adaptive Testing of Controllers for Autonomous Vehicles". In Proceedings of the Symposium on Autonomous Underwater Vehicles Technology, Washington DC, pp. 158-164. IEEE Press.

Stathopoulos, A. Papoutsi, E.; "Understanding Drivers Routes Choice Behaviour: An Experimental Approach"; Seventh International Conference on Travel Behaviour (IATBR-1994); Santiago, jun, 1994; pp. 151-163.

Spears, W. M., De Jong, K. A., "An Analysis of Multi-Point Crossover". In Proceedings of the Foundations of Genetic Algorithms Workshop, Bloomington, IN.

Spears, W. M. and De Jong, K. A. (1990) "Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Problems". In Proceedings of the International Joint Conference on Neural Networks, Washington DC, pp. 118-121.

Spears, W. M., and Anand, V., (1991) "A Study of Crossover Operators in Genetic Programming". In Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems, Charlotte, NC, pp. 409-418.

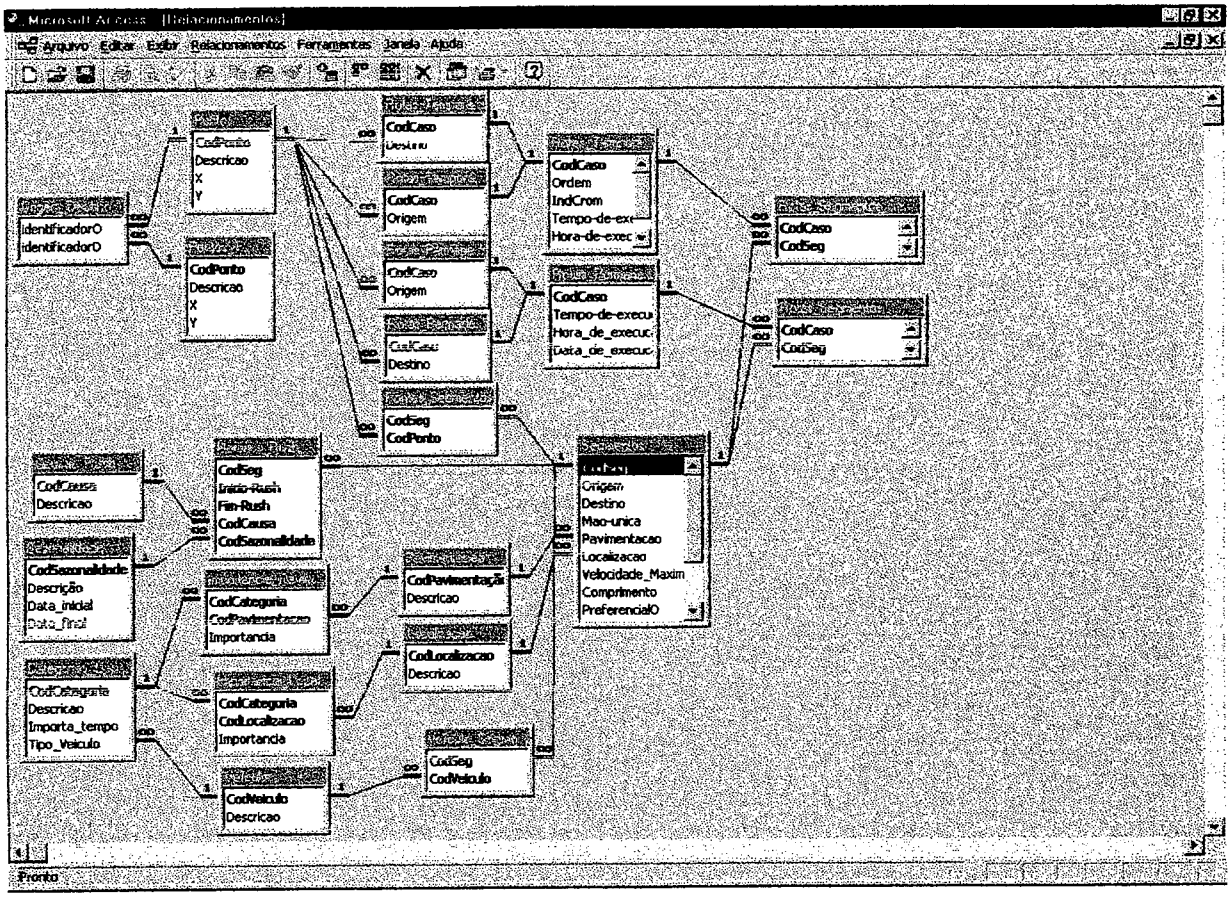
Spears, W. M., and De Jong, K. A., (1991) "On the Virtues of Parameterized Uniform Crossover". In Proceedings of the 4th International Conference on Genetic Algorithms; La Jolla, CA, pp. 230-236. Morgan Kaufman.

Spears, W. M., (1992) Crossover or Mutation? In D. Whitley, editor, Proceedings of the Foundations of Genetic Algorithms Workshop, Vail, CO, pp. 221-237. Morgan Kaufmann.

Spears, W.M. (1995) "Adapting Crossover in Evolutionary Algorithms". To appear in Proceedings of the Fourth Annual Conference on Evolutionary Programming, San Diego, CA.

Sun, R.; "Robust reasoning: Integrating rule-based and similarity-based reasoning"; Artificial Intelligence; Alabama; n. 75, pp. 241-295; 1995

Anexo 1: Diagrama de Entidade-Relacionamento do Protótipo Proteus


Arquivo Editar Exibir Relacionamentos Ferramentas Janela Ajuda
CodPonto
Descricao
X
Y
identificadorO
identificadorD
CodPonto
Descricao
X
Y
CodCaso
Destino
CodCaso
Origem
CodCaso
Origem
CodCaso
Destino
CodSeg
CodPonto
CodCaso
Ordem
IndCrom
Tempo-de-exe
Hora-de-exec
CodCaso
Tempo-de-execu
Hora_de_execuc
Data_de_execuc
CodCaso
CodSeg
CodCaso
CodSeg
CodCausa
Descricao
CodSeg
Inicio-Rush
Fim-Rush
CodCausa
CodSazonalidade
CodSazonalidade
Descrição
Data_inicial
Data_final
CodCategoria
CodPavimentacao
Importancia
CodPavimentação
Descricao
CodLocalizacao
Descricao
CodCategoria
Descricao
Importa_tempo
Tipo_Veiculo
CodCategoria
CodLocalizacao
Importancia
CodVeiculo
Descricao
CodSeg
CodVeiculo
CodSeg
Origem
Destino
Mao-unica
Pavimentacao
Localizacao
Velocidade_Maxim
Comprimento
PreferencialO
Pronto

Anexo 2: Artigo Publicado Referente ao trabalho

A Hybrid System For Route Planning

Sarajane Marques Peres, Josué Pereira de Castro*
Alejandro Martins, Roberto C. S. Pacheco
Aran Bey Tcholakian, Rosina Weber Lee
Lia Caetano Bastos, Fernando Gauthier

Graduate Program in Production Engineering - PPGEP
Federal University of Santa Catarina, Brazil, UFSC
martins@eps.ufsc.br

Abstract

This paper presents a hybrid system for route planning. The system is composed of two principal modules: a case based reasoning (CBR) and a genetic module. The CBR module is responsible for the user interface. It retrieves the solutions founded by the genetic module, and adapts these solutions to the user preferences. The genetic module is responsible for generating new routes, based on the shortest path between the origin and destination points.[1]

Introduction

Several research areas have addressed the route planning optimization problem, including geographic information systems (Abdel-Aty, Abdallah and As-Saidi, 1997), decision theory, genetic algorithms (GA), operational research (Golden and Assad 1988), and case-based reasoning (CBR). We propose the combination of GA and CBR to improve route planning efficiency. An effective solution toward this end should deal with constraints such as rush hour delays, traffic patterns, street conditions, and so on. By integrating different artificial intelligence models we have designed a system that targets such objective.

The system is composed of two modules: the case-based reasoning and the genetic module. The first is responsible for the user interface. It evaluates the routes stored in a case base and executes the best retrieved solution. If none adequate route is retrieved, the genetic algorithm module is activated to create new routes. When the GA does not meet any feasible route, the system proceeds by adapting one of the similar solutions found in the case base.

Route Planning

Route planning is an optimization task with a variety of applications. Several authors have dealt with this subject. Lee and Fishwick (1995), for instance, developed simulation techniques to automate the decision making process in uncertain and complex environments. Christofides (1985) also describes an algorithm and several heuristics to solve vehicle routing problems.

The proposed system aims to help drivers to decide the route to be taken between two given points in urban zones. The route chosen must be both efficient (regarding minimal feasible length) and satisfactory (in terms of driver's preferences). Meeting both objectives has been neglected in route planning literature, making optimization techniques insufficient to balance optimality and user's satisfaction. The combination of CBR and GA made possible to find routes where both aspects are considered.

CBR and GA in Related Applications

Case-based reasoning and genetic algorithms have already been applied in traffic control and route planning, either as combined or stand-alone approaches.

Nakamiti and Gomide (1994) have applied CBR and GA to manage traffic flow more efficiently. Given a traffic condition, the system controls the light times by adapting successful solutions used in similar previous conditions. The system uses GA to adapt the retrieved cases from the case base. The retrieval is based on attribute similarity between the input and the stored cases.

Haigh and Veloso (1995) developed a case-based reasoning system for route planning method. The routes used in the past are stored and may be retrieved and reused to generate new routes. In this system, the planner can use several retrieved cases to generate a new route by merging these cases. An *efficiency value* is associated to each retrieved case, indicating its "quality". Depending on these values, the system either reuses the known routes or searches for alternative routes.

Choi and Woo (1997) proposed an evolutionary route planning algorithm to reveal the optimal route between origin and destination in road traffic networks. Their work also presents a simulation of a network with several constraints.

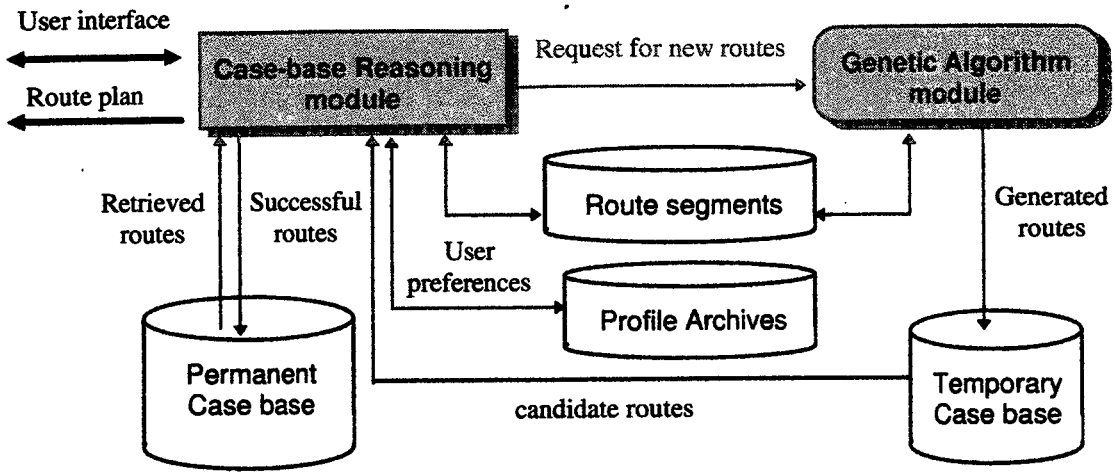


Figure 1: System Architecture.

The System Modeling

Balancing optimality and user satisfaction requires a model that keeps track of both targets while seeking for a solution. This can be understood as the need for *general knowledge*, that is, the model has to balance requirements coming from different frameworks. In CBR theory, a system that employs an external method of general knowledge is classified as a horizontal integrating system (Aamodt, 1993). In order to provide general knowledge we combine CBR with GA (improving the search for intended results). The system is then classified as a *hybrid system*, since it combines two AI techniques on a single horizontal integrating system (Medsker, 1995).

The route site (street traffic network) is represented by road segments and their intersections. Each route is composed by a subset of adjacent and ordered segments. The route begins at the initial point of the first segment (origin) and ends at the final point of the last segment (destination).

The system uses the following parameters: (a) *the user's profile*, consisting of an archive with restrictions (*e.g.*, site restrictions, street conditions preferred, etc.) for each user category (*i.e.*, ambulance service, taxi drivers, etc.); and (b) *the user's request* (origin, destination, and traffic time). The user's request is converted into a case, firing the CBR module. This module searches for similar cases (eventually adapting the most similar routes to the input case), presenting the solutions to the user. If no similar case is found, the genetic module is activated.

When requested, the genetic module creates new route plans. These routes are converted into cases and returned to the CBR module. The CBR, then, proceeds the adaptation to the input case (*i.e.*, the user's route request). Once again, the user checks the solution. If, after this process, a satisfactory route was not found, the system displays the restrictions prohibiting the route generation. The system architecture is briefly described in Figure 1.

The Case-based Reasoning Module

The first requirement in the proposed model is to meet the user profile. Humans tend to keep or adapt previous successful approaches when facing new similar situations. This was the main motivation for building a Case-based Reasoning module to pursue a satisfactory route plan, applying analogical reasoning. This module is responsible for presenting route suggestions to the user based on his or her request. Figure 2 depicts the route representation used in the CBR module. Some attributes are used as indexes and others as route descriptors. The case indexes are the following: identification, origin, destination, case suitability degree. The other attributes are descriptive and serve two purposes: first, as additional information to the user; and, second, as the basis for calculating the route suitability, that is, the strength to which the route fulfill the user requirements. This is part of the CBR adaptation process, where the suggested route is compared to the plan wished by the user.

ID	Orig	Dest	Suitability Degree	Descriptive Attributes

(a)

ID	Segments

(b)

Figure 2: (a) Route representation in the case base. (b) route segment file.

Initially, the module searches the case base for the most similar route to the one desired by the user. If no satisfactory solution is found, the genetic module is activated, and a new temporary case base is created, restarting the retrieving and adapting processes. When

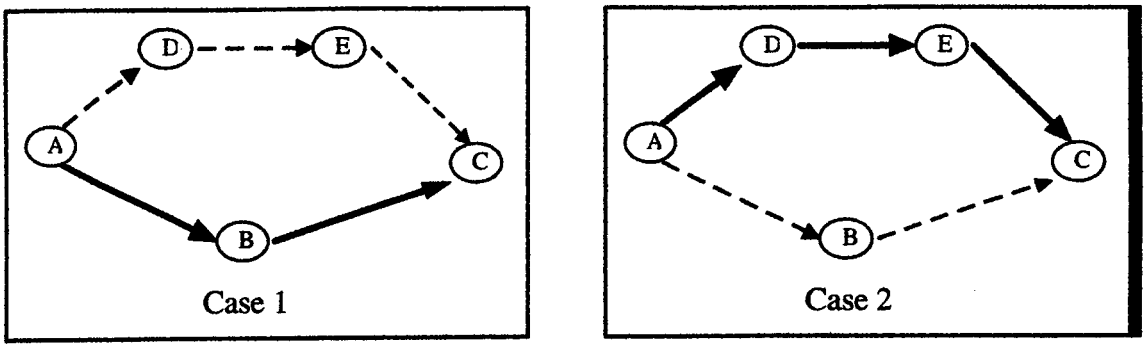


Figure 3: Route adapting process.

there is need for performing route adaptation (*e.g.*, traffic accident), the CBR try to replace part of the route by another set of *satisfactory* segments. The substitutive set of segments is obtained from a partial case retrieval. Figure 3 illustrates an example of the adaptation process. The interval ABC can be replaced by the interval ADEC, observed in a different case.

If the route selection was successful, the system runs the solution. Afterwards the user chooses whether or not to include the route in the case base.

One of the most significant aspects in route planning is the need for rapid adaptation due to traffic network updating. In this system, the CBR module makes adaptive learning possible. In the beginning, the case base is empty. With its utilization, the routes are created, run, and stored in the case base, characterizing a continuous learning process.

The Genetic Module

The GA role is to find feasible routes between the user's origin and destination points. The strategy consists in searching and evaluating alternatives, seeking for the shortest route path.

The first modeling issue is how to represent routes as binary codes (*i.e.*, GA chromosomes). Previous work (*e.g.*, Whitley, Starkweather and Shaner, 1991) have addressed this matter by allowing every two-point combinations. This is valid only for networks fully connected. Actual routes require every segment to be a feasible path. This makes the usual mutation and cross-over operations inadequate, since they can lead to non-existent segments. In this work, we have developed an order independent scheme that makes the GA operations applicable in non-fully connected networks. This scheme allows the modeling of oriented graph and guarantees route feasibility.

The GA module begins by establishing a random set of routes. Each route is a set of oriented segments, created by gradually adding paths. The addition of segments is guided by the shortest Euclidean distance between the candidate nodes and the destination point. The route assembling continues until the algorithm reaches either a final segment (reaching the destination) or the maximum number of generations. GA proceeds attempting to establish more routes until reaching a fixed number of runs (population size).

In this model, each route is represented as a chromosome (see Figure 4), where:

- an allele represents a street traffic network;
- each allele receives one of the values in {0,1,*,#}. Value '0' means that the segment, at the current iteration, does not belong to the route; '1' indicates that the segment is a path candidate; '*' indicates a route segment; and '#' marks the segment as unfeasible for the current route;

1	2	3	4	5	6	7	8	9	10	...	N
0	*	0	#	*	1	*	#	1	0		0

Figure 4: Route represented in a chromosome form.

In this model, *mutation* is a two-step process, composed by (a) *candidate mutation*; and (b) *mutation reset*, described below.

The *candidate mutation* is applied to every allele valued with '1'. It is defined by three operations: (a) calculus of the Euclidean distances between all candidates and the destination; (b) *feasibility test*, which verifies each segment feasibility; and (c) *evaluation*, which checks for either change the allele value (to '*' or '#') or keep it as a candidate, according to the following criteria:

- If the candidate cannot be connected to the current segment set (*i.e.*, it is an unfeasible segment), take its Euclidean distance to the destination (d_c). Take also the shortest Euclidean distance between the current node set and the destination (s_d) (*i.e.*, the shortest Euclidean distance among the '*' alleles). If $d_c > s_d$, mark the candidate allele with '#'. This means that this segment will definitely not belong to the current route.

- Among all feasible candidates, take the one with the minimum Euclidean distance to the destination and convert its allele value to '*'.

The *mutation reset* operator works on each allele valued with either '0' or '1', randomly deciding whether to change it or not to its opposite value. The alleles valued with '*' or '#' remain unchangeable, fixing the schemata in order to assure segment feasibility and the algorithm efficiency in further operations.

The algorithm in Table 1 describes the operations in the genetic module:

Table 1: Algorithm used in the GA module.

MaxPop = max number of desired individuals (constant)
MaxGen = max number of desired generations (constant)
Gen = 0 {generation counter}
CompleteRoute = logic {does the route reach destination?}
For I := 1 to MaxPop do
  Repeat Mutation
    Candidates {try to change '1's to '' or '#' alleles}*
    Reset {random change in the '0' and '1' alleles}
  Until (CompleteRoute) or (++Gen = MaxGen)
End
Fitness {order the population according to actual distance}

In this model, *fitness evaluation* occurs in the end. The population found is sorted according to its total actual distance (*i.e.*, sum of the segment length). The GA output is then an ordered set of optimized routes. The single criteria for discovering these routes was the shortest path. The other user's criteria are still missing. In order to balance length and satisfactory aspects, the GA responses have to be migrated to the CBR module. This is performed by transforming the GA outputs into a case base format (temporary case base). Depending upon the user decision, the temporary cases are incorporated into the permanent case base by the CBR module.

Example

As an example of the system application, we chose an ambulance service, whose principal restriction is the execution speed of the route. In time critical problems, it is important to avoid paths where traffic is influenced by daytime. For this reason, we chose the route execution hour and day as the attributes in the profile archive. Figure 5 depicts an actual street traffic network for the ambulance.

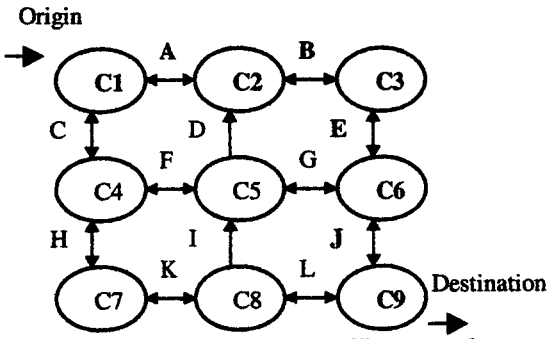


Figure 5: Example of street traffic network.

First, the user (ambulance driver) asks for a route plan. The request is evaluated by the CBR module, which seeks for similar cases. In this case, the suggested route was 'A-D-I-L'. The user was unsatisfied with this route plan. He knows that segment 'L' is currently blocked. The CBR module then calls the GA module, asking for alternative solutions, and marking the segment L as unfeasible in the street traffic network profile.

A	B	C	D	E	F	G	H	I	J	K	L
*	1	1	0	0	0	1	0	1	0	0	#

Figure 6: First individual of the population.

As shown in Figure 6, the GA module begins by establishing a first individual of the population (*i.e.*, a set with the first route segment and other candidates).

The next step is to evaluate this individual, searching for the segment feasibility (Figure 7). In this case, segment B is the only feasible path. The algorithm also analyzes whether the unfeasible segments can eventually belong to the current route. In this case, segment C will never belong to any route containing segment A. Thus, this segment is made permanently unfeasible (*i.e.*, it is marked with '#').

A	B	C	D	E	F	G	H	I	J	K	L
*	*	#	0	1	1	0	0	0	1	0	#

Figure 7: Individual after fitness followed by mutation.

The route design process remains until a final segment is found or until the algorithm hits the maximum number of generations. Figure 8 shows the result after a sequence of GA iterations over the same route plan.

A	B	C	D	E	F	G	H	I	J	K	L
*	*	#	0	*	0	1	1	1	*	0	#
1	2			3					4		

Figure 8: Complete and ordered route obtained after some fitness and mutation operations.

The next GA step is to convert the suggested route plan into a case and send it to the temporary case base. The result is shown in Figure 9.

ID	Source	Destination	Adeq. Degree	Hour	Day	Weather Condition s
1	C1	C9	90%	12:00	Mon	Good

(a)

ID	Segment
1	A
1	B
1	E
1	J

(b)

Figure 9: (a) Temporary case base (route adjusted in a form of a case); (b) route segment file.

Finally, the CBR module takes place and presents the new route to the user. If he decides to take the suggestion, the CBR integrates the new route into the permanent case base for future use in similar situations.

Conclusions and Future Work

In this work, we have presented a new model for route planning problems. In this approach, the suggested route plans are both efficient and satisfactory. They represent a balance between optimality (minimal feasible length) and user requirements. This was made possible by means of a hybrid intelligent system, including a module for each purpose. The user's requirements are pursued by the Case-base Reasoning module, while optimality is seek by the Genetic Algorithm module.

Analogical Reasoning and adaptive learning were the main motivation for building a CBR. This module not only takes similar past experiences to propose route plans, but also uses the new information to increase its memory for further iterations. When a solution is not available (insufficiency of knowledge or user dissatisfaction), CBR calls the Genetic Algorithm module.

The GA purpose is to establish route plans that meet the new conditions and keep optimality as the primer concern. In addition, the method had to keep feasibility constraints and boost optimality search. We achieved these objectives developing a new chromosome representation scheme, suitable for route plan problems in oriented graph networks. The found route plans are evaluated by the CBR and presented to the user. The new route plan is eventually incorporated in the permanent case base.

The interaction between the CBR and GA aims to balance satisfaction and optimality constraints. By keeping the objectives separate and integrated we avoid increasing complexity in each module (*e.g.*, by including user profile in the GA, one would increase iteration time, since the fitness would be a more complex operator). We have also made adaptive learning possible, since the case base is dynamic with the time.

As future work, the group is evaluating different alternatives. First, we intend to implement an interface with a Geographic Information System (GIS). This optimizes the addition of new street traffic network configurations. We are also studying the possibility of including a fuzzy expert system to deal with season variables (*e.g.*, tourist season, school concerning, etc.) and temporal data (*e.g.*, daytime), related to traffic flow conditions. An example of combining GIS and fuzzy modeling can be found in Lee and Lee's work (1996). Finally, we are also studying other approaches for the optimality search, such as the use of GA* (Logan and Poli, 1997).

References

Abdel-Aty, M.A., Abdallah, M.N.and As-Saidi, A. H. 1997. A Methodology for Route Selection and Guides Using GIS and Computer Network Models. In *Proceedings of 76th Annual Meeting of The Transportation Research Board*, 20-21. Washington D.C.: Transportation Research Board.

Aamodt, Agnar. Explanation-Driven Case-Based Reasoning. *Topics in Case-Based Reasoning, (First European Workshop, EWCBR-93).* Wess, Stefan, Althoff, Klaus-Dieter & Ricthter, Michael (editors) Springer-Velarg, 274-288, 1993.

Choi, G.S. and Woo, K.D. 1997. The development of Optimal Route Algorithm Based On Evolution Program. *The Transactions of The Korean Institute of Electrical Engineers* 46(2):294-297

Christofides, N. 1985. Vehicle Routing In The Traveling Salesman Problem - In *A Guided Tour of Combinatorial Optimization*, 431-448. Great Britain.: John Wiley & Sons Ltd.

Golden, B. L., and Assad, A., 1988. *Vehicle Routing: Methods and Studies.* Amsterdam, Netherlands: Elsevier Science Publishers B.V.

Haig, K. Z. & Veloso, M.,. 1995 Route planning by analogy. *Proceedings of Case Based Reasoning research and development, First International Conference: ICCBR-95*, Springer-Verlag, 171-180.

Lee, J. and Fishwick, P.A., 1995. Simulation Based Real Time Decision Making for Rout Planning. *1995 Winter Simulation Conference*, 1087-1095. Crystal City, VA.

Lee, H.C., and Lee, C.C. 1996. Inexact strategy and Planning in Taipei City. *In Proceedings of Asian Fuzzy Systems Symposium, 1996, Kenting, Taiwan, Soft Computing in Intelligent Systems and Information processing*, 308-313. Piscataway, NJ.: IEEE Service Center.

Logan, B and Poli, R. 1997. Route Planning with GA*, Technical Report, CSRP-97-17, School of Computer

Science, University of Birmingham, England

Medsker, L. R. , *Hybrid Intelligent Systems*, Kluwer Academic Publishers: Boston, 1995.

Nakamiti, G. & Gomide, F., An evolutive fuzzy mechanism based on past experiences,. *Second European Congress on Intelligent Techniques in Soft Computing*, Aachen, Germany, sept, 1994.

Whitley, D., Starkweather, T. and Shaner, D. 1991. The Traveling Salesman and the Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. In *Handbook of Genetic Algorithms*, 350-372. New York, NY.: Van Nostrand Reinhold.

Ackowledgements

* The author would like to acknowledge the financial support of Federal University of Pará and the Brazilian federal agency CAPES.