UMA PROPOSTA DE ROTEAMENTO EM REDES
UTILIZANDO ALGORITMOS GENÉTICOS

Maria Elizabeth Vizhñay Zambrano

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA EM ENGENHARIA ELÉTRICA.

Aprovada por:

_____

Prof. Eugenius Kaszkurewicz, D.Sc.

_____

Prof. Amit Bhaya, Ph.D.

_____

Prof. Benjamin Barán Cegla, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2007

ZAMBRANO, MARIA ELIZABETH VIZHÑAY

Uma proposta de Roteamaento em Redes utilizando Algoritmos Genéticos. [Rio de Janeiro] 2007.

XIII, 98 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Caminho mais curto.
2. Algoritmo Genético.

I. COPPE/UFRJ II. Título (série)

***Aos meus queridos pais,***

José Vizhnay e Emilia Zambrano,

**As minhas irmãs**, Jacqueline, Rosa, Gisella e Cecilia.

**Aos meus sobrinhos**, Andrea e Sebastian.

**Aos meus cunhados**, Jose e Gustavo

Em gratidão por tamanho e continuo apoio no dia a dia do meu viver e no conquistar de novos horizontes.

A Ítalo Zambrano pelo apoio, estimulo, confiança e incentivos nos momentos mais difíceis, no transcurso da maestria.

# Agradecimentos

Nenhum tipo de agradecimento e suficientemente expressivo para demonstrar meu real reconhecimento. A todas as pessoas que me ajudaram, fica aqui o meu muito obrigada.

Ao meu orientador, Professor Ph. D. Eugenius Kaszkurewicz pelo apoio, estímulo, confiança, ensinamentos e orientações convincentes.

Ao Professor Amit pela guia oferecida, os oportunos esclarecimentos de dúvidas, e pelo freqüente interesse no desenvolvimento do presente trabalho.

À CAPES, CNPQ pelo apoio financeiro com o qual se tornou possível a finalização desta pesquisa.

Professores, funcionários e amigos da COPPE/UFRJ pelo apoio oferecido e convívio gratificante.

A todos os meus familiares, e amigos pessoais .

Companheiros de turma de mestrado, doutorado e amigos do Laspot pela sincera e duradoura amizade estabelecida.

A Enrique Chaparro pela sua amizade, apoio e incentivos que foram fundamentais nos momentos difíceis do desenvolvimento deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# UMA PROPOSTA DE ROTEAMENTO EM REDES UTILIZANDO ALGORITMOS GENÉTICOS

Maria Elizabeth Vizhñay Zambrano

Abril/2007

Orientador: Eugenius Kaszkurewicz

Programa: Engenharia Elétrica

No presente trabalho, implementou-se um algoritmo não adaptativo que utiliza como técnica heurística os algoritmos genéticos, procurando por meio de estes otimizar o processo de roteamento, que possa ser considerado com parâmetro intrínseco à confiabilidade de cada uma das conexões.

Para o roteamento, utilizou-se um algoritmo genético convencional com codificação de números inteiros, adicionando-se uma função reparação para tornar aptos certos indivíduos inviáveis, e para melhorar a velocidade de convergência até um ótimo global. A implementação do algoritmo foi feita nas versões seqüencial e paralela, conseguindo-se melhorar a eficiência computacional através da versão paralela do algoritmo em relação à sua contrapartida versão seqüencial.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.).

# A PROPOSAL FOR ROUTING IN NETWORK USING GENETIC ALGORITHMS

Maria Elizabeth Vizhñay Zambrano

April/2007

Advisor: Eugenius Kaszkurewicz

Department: Electrical Engineering

The dissertation implements a genetic algorithm for the optimization of a routing problem in which both length and reliability of the route is taken into account. The routing is achieved using a conventional genetic algorithm with integer coding, with the inclusion of a so-called repairing function that permits infeasible solutions to contribute to the richness of the population used in the genetic algorithm, as well as to the speed of convergence. The algorithm is implemented in both sequential and parallel versions, and latter, as expected,reduces computational time with respect to the former.

# ÍNDICE

# Lista de Figuras

# Lista de Tabelas

# Lista de Símbolos

| | |
|---|---|
| *AG* | Algoritmo Genético. |
| *C* | Matriz de conexões. |
| *F* | Matriz de confiabilidade. |
| $c_{ij}$ | Componente (*i*,*j*) da matriz de conexão. |
| $f_{ij}$ | Componente (*i*,*j*) da matriz confiabilidade. |
| *D* | Matriz de Distância. |
| $d_{ij}$ | Componente (*i*,*j*) da matriz distância. |
| *E* | Espaço de busca. |
| *G* | Rede |
| i, j, k | Denominações da numeração dada aos nós da rede. |
| *L* | Conjunto de conexões existentes. |
| *M* | Matriz de conexões possíveis a percorrer. |
| $m_{ij}$ | Componente (*i*,*j*) da matriz de conexões possíveis a percorrer. |
| $N_{part}$ | Número do nó de partida ou inicio. |
| $N_{cheg}$ | Número do nó de chegada ou destino ou final. |
| $nC_{max}$ | Número máximo de conexões possíveis em uma rede. |
| *nV* | Número total de nós existentes. |
| T | Trajetória. |
| *V* | Conjunto de nós ou vértices existentes. |
| *v* | Nó. |
| *x* | Variável de decisão |
| *X* | Conjunto de variáveis de decisão que formam uma população |
| *z* | Número de conexões existentes. |
| $\varepsilon_k$ | Nó $v_{k-1}$ que antecede $v_k$ no caminho mais curto de i para k |
| $\pi_k$ | A soma total das distancias correspondentes as conexões percorrida |
| $[\varepsilon_k, \pi_k]$ | Permanente→ caminho mais curto, ou,<br>Temporário → o melhor caminho. |
| [-,∞] | indica que ainda não foi encontrado qualquer caminho de *i* para *k* |
| (i,j) | Coordenada. |
| {} | Conjunto |

# Capítulo 1

# I. Introdução

## 1.1 Aspectos Gerais

O rápido crescimento na infra-estrutura das redes de comunicação, a criação de novos softwares e os serviços de internet levaram a um aumento crescente na demanda da comunicação de dados nos últimos tempos [27]. Sendo este o motivo pelo qual vem aumentando o interesse no problema de desenho na interconexão entre os nós de uma rede relacionando-o assim diretamente ao roteamento da informação.

Como o roteamento é o mecanismo através do quais dois nós que formam parte de uma rede se comunicam, no sentido de escolher a rota a ser seguida se criaram algoritmos responsáveis por este processo, que ao passar do tempo foi necessário os otimizar, introduzindo-se outros parâmetros neste processo como são largura de banda, tráfego médio, custo de comunicação, entre outros fatores. Este processo de roteamento tem uma grande importância tanto na fase de ampliação e projeto de novas redes como para os usuários.

Existem duas classes de algoritmos de roteamento: adaptativos e não adaptativos. Os algoritmos não-adaptativos [3] decidem o roteamento em medida da topologia atual ou estimativa do tráfego calculado off-line, este processo também é chamado de roteamento estático. E quando o algoritmo de roteamento vai mudando suas decisões, refletindo as mudanças normalmente no tráfego da rede se denominam algoritmos adaptativos.

Neste processo de roteamento existem problemas que surgem com o crescimento das redes de comunicação como o aumento na complexidade do algoritmo de busca do caminho mais curto, surgindo assim a necessidade de implementar sempre novos algoritmos de roteamento que sejam capazes de substituir os métodos tradicionais, os quais geralmente fornecem soluções não suficientemente ótimas (ótimos locais) quando aplicados num sistema de grande porte. Neste caso, a implementação do algoritmo exige grande poder computacional.

Novas técnicas de roteamento, que otimizam de forma rápida e eficiente o fluxo da transmissão de dados das redes de comunicação podem também contribuir para a solução de problemas análogos: otimização das redes de transporte numa determinada cidade, organização do fluxo de vôos nos aeroportos, melhoras nas rotas de navegação marítima, planejamento das estações de bombeamento através de tubulações, minimização das rotas entre cidades, entre outros. Isto é devido ao fato que cada um dos problemas descritos anteriormente pode ser formulado através da representação de redes de comunicação.

## 1.2 Objetivos

Considerando todas as idéias descritas acima, sobre o roteamento através de um algoritmo simples, rápido e eficaz, e motivada pelo nível de importância das redes de comunicação na atualidade, o presente trabalho tem os principais objetivos:

1. Desenvolver um algoritmo que seja eficiente e simples para o cálculo da rota de transmissão mais curta possível e mais confiável entre um nó origem e um nó destino utilizando uma técnica heurística de otimização global como são os Algoritmos Genéticos.

2. Utilizar os Algoritmos Genéticos (AG) para Problemas de Otimização Multi-Objetivos, com a finalidade de minimizar o caminho de transmissão e escolher a rota mais confiável, considerando, por exemplo, o tipo de tecnologia de comunicação utilizada (ponto a ponto, satélite, etc.), freqüências de falhas, riscos de queda de determinados centros de comunicação, entre outros.

3. Utilizar a computação paralela ou distribuída, para minimizar o tempo de processamento, quando o algoritmo de roteamento desenvolvido é aplicado a um sistema de grande porte, e obter melhores resultados devido às propriedades dos AG Paralelos [Apêndice B].

No presente trabalho, para o desenvolvimento da técnica de roteamento visando a confiabilidade de uma rede de comunicação, é utilizado o *AG*, pois está técnica evolutiva de otimização tem sido aplicada com sucesso em problemas tecnológicos complexos [2], devido à relativa facilidade de implementação.

Embora os métodos evolutivos não garantam a obtenção do ótimo teórico para todos os casos, eles conseguem, na maioria das vezes obter soluções ótimas para problemas em que os métodos numéricos tradicionais não são eficazes [10].

## 1.3 Estrutura da Dissertação

A estrutura do presente trabalho é descrita a seguir:

CAPÍTULO 2: DEFINIÇÃO DO PROBLEMA. Neste capítulo será definido o problema do caminho mais curto e confiável que será utilizado como base na implementação do algoritmo de roteamento ótimo. Os algoritmos mais conhecidos e utilizados na sua solução do caminho mais curto, bem como as vantagens e desvantagens destes. Também serão apresentados e comentados os diversos trabalhos utilizados como referencia, através de uma revisão bibliográfica.

CAPÍTULO 3: ALGORITMO GENÉTICO PROPOSTO PARA A RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO COM CONFIABILIDADE. Será descrito e representado matematicamente o problema de otimização do roteamento pelo caminho mais curto, utilizando um algoritmo simples e eficaz na busca da menor rota de transmissão, além de procurar pelo trajeto que apresente menor risco de falha, ou seja, a que apresente a mais alta confiabilidade.

CAPÍTULO 4: SIMULAÇÃO E RESULTADOS. Serão apresentados os diversos sistemas de comunicação utilizados como teste, os resultados de trajeto ótimo obtidos pelo algoritmo de roteamento proposto. Neste capítulo, são comparados entre si os resultados obtidos pela versão seqüencial e pela versão paralela do algoritmo de roteamento desenvolvido.

CAPÍTULO 5: CONCLUSÕES E TRABALHOS FUTUROS. Neste capítulo serão apresentadas as conclusões baseadas na implementação do algoritmo de roteamento proposto e os resultados obtidos na aplicação desta metodologia sobre os sistemas utilizados como teste. Além disso, são apresentadas sugestões para trabalhos futuros.

# Capítulo 2

# II. Definição do Problema

## 2.1 Introdução

Uma vez que as redes surgem em numerosos contextos e nas mais variadas formas, muitas ciências têm contribuído com idéias importantes, para a evolução do estudo de problemas de fluxo de redes, já que a forma mais simples de modelar matematicamente muitos problemas relacionados é utilizando grafos de redes. Apesar desta diversidade, a literatura da teoria de grafos e de redes não possui uniformidade, fazendo com que vários autores adaptem uma grande variedade de convenções e notações. Por este motivo a seguir será apresentada a terminologia utilizada na escrita desta tese.

## 2.2 Conceitos fundamentais

Um *grafo* [3] é um conjunto de vértices ($V$) chamados nós ou pontos conectados por linhas chamados de ligações ($L$) ou conexões, o qual pode ser representado matematicamente por $\boldsymbol{G}\{V,L\}$.

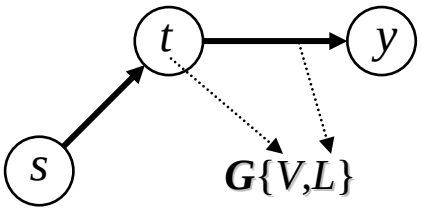



**Figura 2.1:** Exemplo de um grafo.

Na **Figura 2.1** se exemplifica um grafo de três nós com duas conexões. Cada *conexão* é representada através de um par ordenado de vértices (s,t) e (t,y), sendo que os vértices pertencem a $V$.

Existem três tipos de grafos:

- *Direcionado* quando todas as conexões são dirigidas.
- *Não direcionado* quando todas as conexões são não dirigidas.
- *Misto* quando algumas conexões são dirigidas e outras são não dirigidas.

Uma *conexão não dirigida* ou *não orientada* se for representada por um par não ordenado de nós qualquer, isto quer dizer que uma conexão que liga os nós 1 e 2 pode ser representada por (1,2) ou por (2,1)

Por exemplo, uma *conexão não dirigida* pode ser vista como uma rua de dois sentidos, que permite fluxo em ambas as direções (de 1 para 2 e de 2 para 1).

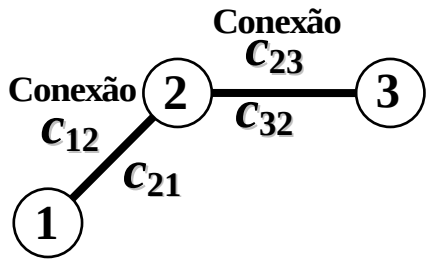



**Figura 2.2:** Exemplo das conexões não dirigidas.

Uma *conexão* diz-se *dirigida* ou *orientada* se for representada por um par ordenado de nós (1,2), sendo que o primeiro componente do par ordenado 1 corresponde ao vértice de partida e 2 corresponde ao vértice de chegada.

Uma *conexão dirigida* (1,2) pode ser vista como uma rua de um só sentido, que permite fluxo apenas de 1 para 2.

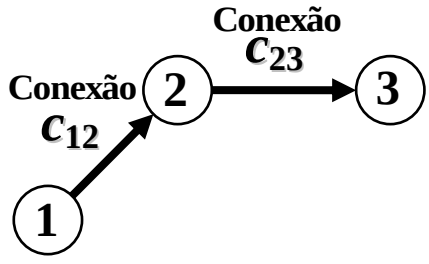



**Figura 2.3:** Exemplo das conexões dirigidas.

Na **Figura 2.3**, a *conexão* $c_{12}$ é o caminho físico direto entre um nó 1 e o nó 2. Assim neste gráfico temos duas conexões $c_{12}$ e $c_{23}$.

Um *trajeto* ou *caminho* (*T*) desde um nó s para um nó y é uma sucessão de vértices e conexões sem repetir os nós. Para exemplificar consideremos a **Figura 2.1**:

$$T \rightarrow c_{st}, c_{ty} \tag{2.1}$$

Um *ciclo* é um trajeto de s para s. Neste caso, consideremos que existe também caminho físico entre o nó y e o nó s.

$$T \rightarrow c_{st}, c_{ty}, c_{ys} \tag{2.2}$$

Uma *rede* pode ser representada através de um grafo cujas conexões têm associado valores numéricos (distâncias, confiabilidade, custos, etc.). A terminologia normalmente utilizada em redes é nós e conexões, em vez de vértices e ligações, respectivamente.

A *confiabilidade* [6] é um índice de desempenho que permite avaliar quantitativamente os tempos de falha que os produtos ou sistemas apresentam durante sua vida útil. Também se pode definir como uma métrica que descreve a facilidade que tem os sistemas em completar sua missão satisfatoriamente.

A *distância* [23] percorrida por um objeto é o comprimento de sua trajetória e se trata de uma magnitude escalar.

O *nó de partida* se define como o nó de onde vai sair a informação e *nó de chegada* o *final* se define como o nó onde vai chegar a informação sendo este o último nó da sucessão.

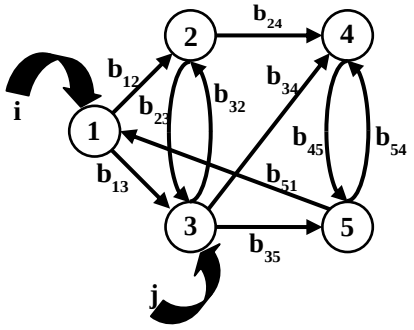
## 2.3 Representação de uma rede

A eficiência computacional de um algoritmo para resolver problemas de otimização em redes não depende apenas das suas características intrínsecas, mas também, e muito, das estruturas de dados utilizadas para representar a rede no computador (formas de armazenar manipular os dados associados à rede) bem como para armazenar os resultados intermediários necessários ao algoritmo [32]. Geralmente, para representar uma rede no computador são necessários dois tipos de informação:

- A topologia da rede (estrutura dos nós e conexões),
- Os dados (distâncias, confiabilidade, custos, etc.) associados aos nós e às conexões.

Normalmente, o esquema utilizado para armazenar a topologia da rede irá sugerir, naturalmente, uma forma de armazenar a informação associada às conexões e aos nós. Dentre as representações mais comuns de uma rede são as seguintes: matriz de adjacência, matriz de incidência, lista de adjacência [31].

Neste trabalho utilizou-se a matriz de estrutura de adjacência para a representação da rede e seus atributos. A matriz de estrutura de adjacência nó-nó, armazena uma rede *G* numa matriz quadrada de ordem $B_{nV \times nV}$ sendo que cada linha e cada coluna corresponde a um nó.




$$B = \begin{bmatrix} 0 & b_{12} & b_{13} & 0 & 0 \\ 0 & 0 & b_{23} & b_{24} & 0 \\ 0 & b_{32} & 0 & b_{34} & b_{35} \\ 0 & 0 & 0 & 0 & b_{45} \\ b_{51} & 0 & 0 & b_{54} & 0 \end{bmatrix}_{(nV \times nV)}$$

**Figura 2.4:** Representação matemática da rede através das conexões.

Se escolhermos $b_{ij} = c_{ij}$, sendo que cada elemento $c_{ij}$ da matriz assume um dos seguintes valores: 1 (hum), se existe conexão entre (i,j) ou 0 (zero), se não existe conexão entre (i,j), então obtemos a denominada matriz de adjacência usual $C = \{c_{ij}\}$.

$$C = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}_{(5 \times 5)}$$

**Figura 2.5:** Representação da existência das conexões.

Da mesma forma, para cada parâmetro associado às conexões da rede (distâncias, confiabilidade, custos, etc.) podemos definir uma matriz a partir da escolha adequada do elemento $b_{ij}$. Observa-se também que nas redes não direcionadas, a existência de cada arco (i,j) corresponde a duas conexões dirigidas: (i,j) e (j,i). Neste caso a matriz de adjacência resultante é simétrica.

Quando se escolhe $b_{ij} = d_{ij}$, sendo $d_{ij}$ um valor numérico não negativo que representa a distância entre nó *i* e nó *j*, obtemos a matriz de distância $D = (d_{ij})$.

Quando se escolhe $b_{ij} = f_{ij}$, sendo $f_{ij}$ um número real no intervalo [0,1] que representa confiabilidade da conexão entre nó *i* e nó *j*, sendo zero se não há conexão ($c_{ij}$=0), obtemos a matriz de confiabilidade $F = (f_{ij})$.

Os valores de $f_{ij}$ existentes podem ser interpretados como probabilidades de funcionamento sem falhas da conexão entre nó i e nó j, denominada confiabilidade, portanto a soma das probabilidades de transmissão de todas as rota que saem de cada um dos nós da rede deve ser igual a um, construindo-se finalmente uma matriz de confiabilidade, onde a soma de cada linha bem como a soma de cada coluna é igual a um [30].

Nos problemas em que a densidade da respectiva rede é muito baixa (matriz com grande quantidade de elementos nulos), este tipo de estrutura é bastante ineficiente, tanto relativamente ao tempo de pesquisa como ao espaço de memória necessário ao armazenamento de dados. Isto é porque a matriz de adjacência permite-nos armazenar $nV^2$ conexões e a rede apenas tem *z*, sendo que $z << nV^2$. Sabendo-se que nenhum nó pode ter conexão consigo mesmo, em uma rede de $nV$ nós se tem como número máximo de possíveis conexões existentes $nC_{max}$, a que se calcula através da seguinte equação:

$$\boldsymbol{nC_{max}} = nV \times \ nV - 1 \qquad \textbf{(2.3)}$$

Apesar disso, a simplicidade desta representação torna atrativa sua utilização na implementação de muitos algoritmos de redes. Isto porque se pode:

- Determinar os parâmetros associados a qualquer arco (i,j), tomando simplesmente o elemento (i,j) das respectivas matrizes.
- Obter facilmente os arcos que saem do nó i, examinando a linha i: se o j-esimo elemento dessa linha tem o valor de um (1) então (i,j) é um conexão da rede;
- Obter os arcos que chegam ao nó j, examinando a coluna j: se o i-esimo elemento dessa coluna tem valor um (1) então (i,j) é um arco da rede.

Uma vez esclarecidos alguns conceitos e definida a forma como a rede é modelada computacionalmente para a implementação da metodologia do roteamento, poderemos continuar definindo a base do problema.

## 2.4 O problema do caminho mais curto

O problema do caminho mais curto é fundamental e freqüente quando se estudam problemas em redes. Por exemplo, quando se pretende determinar o caminho de custo mínimo entre um ou vários pares de nós de uma rede, determinar o caminho de menor comprimento ou que resulte num tempo mínimo de comunicação, que tenha um máximo de segurança, entre os mais conhecidos. O problema do caminho mais curto surgiu devido a uma série de problemas práticos nas indústrias de telecomunicações e de transportes, tais como o encaminhamento de mensagens em sistemas de comunicação, ou quando se pretende enviar um veículo entre dois locais geograficamente distantes da forma mais rápida e o mais barato possível.

O planejamento do tráfego urbano constitui um exemplo importante, no qual os modelos utilizados para o cálculo do fluxo de tráfego padrão são problemas de otimização não linear, ou modelos de equilíbrio complexos. Contudo, a maioria das abordagens algorítmicas para determinar tráfegos urbanos padrão consistem, sob hipóteses simplificadoras, na resolução de um grande número de problemas de caminho mais curto, um para cada par de nós origem (partida) destino (chegada) da rede.

Existem três tipos de problemas de caminho mais curto:

- De um nó para outro,
- De um nó para todos os outros (árvore dos caminhos mais curtos),

- Entre todos os pares de nós.

Os dois primeiros tipos são identificados como *problema do caminho mais curto de origem única* ou apenas *caminho mais curto*, e o terceiro como *problema de caminho mais curto entre todos os pares de nós*.

Vamos iniciar nosso estudo baseado nas redes. Assim, a idéia é criar um gráfico onde cada nó i do gráfico represente um nó da rede *M* sendo que $i \in \{1, 2, ..., nV\}$ e cada arco que indica uma linha de comunicação se denomina conexão $c_{ij}$. Para escolher a rota entre um determinado par de nós i e j, o algoritmo simplesmente encontra o caminho mais curto entre eles.

Entretanto muitas outras unidades métricas também são possíveis de adotar além da distância. Por exemplo, cada conexão pode ser caracterizada através de: o custo da comunicação, largura de banda entre outros fatores. Nesse caso a rota a ser seguida deverá considerar os três fatores como um único valor, calculado através da soma ponderada dos fatores, assim, alterando a função de atribuição de pesos, o algoritmo selecionaria o caminho a ser seguido de acordo com determinados critérios que podem ser ou não combinados.

Para determinados criterios é possível utilizar o problema do caminho mais curto como base para a otimização do roteamento. A abordagem mais conhecida para este tipo de problemas é através dos algoritmos determinísticos, com é: O algoritmo de Dikjstra.

Uma forma de comparar a eficiência entre os diferentes algoritmos é através da complexidade de cada um deles. A complexidade de tempo de um algoritmo é o tempo necessário para computar o resultado para uma instância do problema de tamanho *n*. Para tanto, costuma-se obter duas medidas: o do pior caso e o do caso médio. Dessa forma, o tempo de computação está diretamente associado à complexidade do algoritmo [4][3][23].

## 2.5 Algoritmos Padrão no *Problema do Caminho Mais Curto*

Existem vários algoritmos eficientes para resolver o problema do caminho mais curto, como os algoritmos de *Dijkstra*, de *Ford* e de *Floyd*.

Os dois primeiros são aplicados a problemas de caminho mais curto de um nó inicial ou nó partida, *i*, para um nó final ou nó chegada, *j*, ou para todos os outros nós, e baseiam-se num processo de rotulação dos nós; o último é aplicado em problemas no qual são determinados os caminhos mais curto entre todos os pares de nós.

### 2.5.1 Algoritmo de Dijkstra

O algoritmo de *Dijkstra* [2] só pode ser aplicado a redes onde a distância $d_{ij}$, que é um parâmetro característico da conexão $c_{ij}$, esta associada apenas a um valor não negativo. Neste algoritmo uma conexão $c_{ij}$ é o caminho entre o nó *i* e qualquer outro nó *j*.

O algoritmo baseia-se num processo de rotulação dos nós da rede e classificação dos respectivos rótulos. A cada nó *k* (último nó percorrido) é atribuído um rótulo $[\xi_k, \pi_k]$ que pode ser permanente ou temporário. Isto é,

$[\xi_k, \pi_k]$ permanente (o caminho mais curto de *i* para *k*).

$\xi_k \longrightarrow$ nó *k*-1 que antecede *k* no caminho mais curto de *i* para *k*.

$\pi_k \longrightarrow$ a soma total das distâncias correspondentes a cada conexão percorrida até chegar a *k*.

$[\xi_k, \pi_k]$ temporário (o melhor caminho, até o momento, de *i* para *k*).

$\xi_k \longrightarrow$ nó *k*-1 que antecede *k* no melhor caminho, até o momento, de *i* para *k*.

$\pi_k \longrightarrow$ a soma total das distâncias correspondentes a cada conexão percorrida até chegar a *k*.

$[\xi_k, \pi_k] = [-, \infty]$ indica que ainda não foi encontrado qualquer caminho de *i* para *k*.

O algoritmo consiste em um processo de fixação dos rótulos dos nós da rede, começando pelo nó *i*, de forma ordenada segundo as distâncias a cada nó. Em cada iteração é escolhido o nó *k* com rótulo temporário com menor valor de $\pi$, que se torna permanente, depois de varrer todos os nós adjacentes a este.

O algoritmo termina quando não existirem nós com rótulos temporários ou quando o rótulo do nó j passe a ser permanente. O algoritmo de Dijkstra está descrito a seguir em pseudocódigo:

$[\xi_s,\pi_s] \leftarrow [s,0]$ (caminho mais curto de **s** para **s** tem comprimento 0 e **s** é o nó que antecede **s**).

$[\xi_i, \pi_i] \leftarrow [-, \infty]$ , $\forall_i \in N - \{ s \}$ *(o caminho mais curto de **s** para **i** é desconhecido)*

$R \leftarrow N$ *(R = conjunto de nós com rótulo temporário)*

$\overline{R} \leftarrow \varnothing$ *($\overline{R}$ = conjunto de nós com rótulo permanente)*

**Enquanto** $R \neq \varnothing$ (ou $R \neq N$) **Fazer**

$k \leftarrow$ nó de R tal que $\pi_k$ é mínimo *($k : \pi_k = min \{ \pi_x \}, x \in R$)*

$R \leftarrow R - \{ k \}$ *(k deixou de ter rótulo temporário)*

$\overline{R} \leftarrow \overline{R} \cup \{ k \}$ *(k passou a ter rótulo permanente)*

**Para** todo $j \in N$ tal que $(k, j) \in C$ **Fazer**

**Se** $\pi_k + c_{kj} < \pi j$ **Então**

$\pi_j \leftarrow \pi_k + c_{kj}$

$\xi_j \leftarrow k$

**Fim do Enquanto**

**Figura 2.6:** Pseudocódigo do Algoritmo de Dijkstra.

A **Figura 2.7** apresenta um grafo orientado com cinco vértices, no qual se aplicará o algoritmo de *Dijkstra* na forma de ilustrar o processo de rotulação do algoritmo.

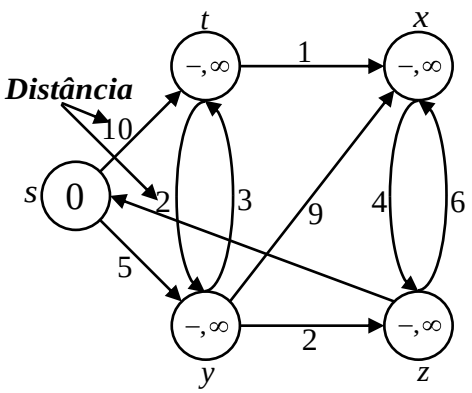



**Figura 2.7:** Grafo orientado**.**

Considerando como ponto de partida o vértice *s*, é observado que existem duas conexões $c_{st}$ e $c_{sy}$ rotulando-se ao nó *t* e ao nó *y*, temporários e outro como permanente. Como se mostra na **Figura 2.8**:

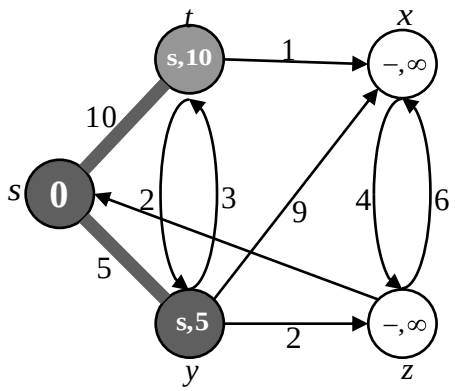



**Figura 2.8:** Os caminhos considerados pelo algoritmo de *Dijkstra*.

Na **Figura 2.8**, o caminho mais curto corresponde à conexão $c_{sy}$ que liga os nós *s* e *y*. O nó associado ao caminho mais curto é rotulado com uma pontuação menor (neste caso 5). Porém, a outra conexão $c_{st}$ que liga o nó *s* e *t* é atribuída uma pontuação igual a dez (pontuação maior). Desta maneira, o nó com pontuação (rotulação) menor adquire o *status* de nó permanente.

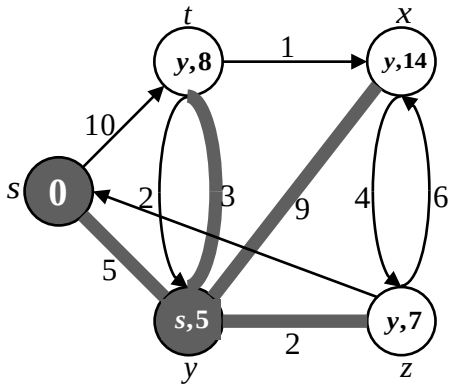



**Figura 2.9:** Os caminhos considerados pelo algoritmo de *Dijkstra*.

Repete-se a analise anteriormente descrita, se escolhe o caminho que vai acumulando a menor distância (neste caso o nó *z*) que tem uma pontuação de valor igual a sete, conforme mostrado na **Figura 2.9**. Dessa maneira, a rotulação do nó *z* adquire *status* permanente. Seguidamente, a partir do nó *z* é selecionada a conexão $c_{zx}$, pois a distância acumulada correspondente é a menor entre todas as conexões adjacentes.

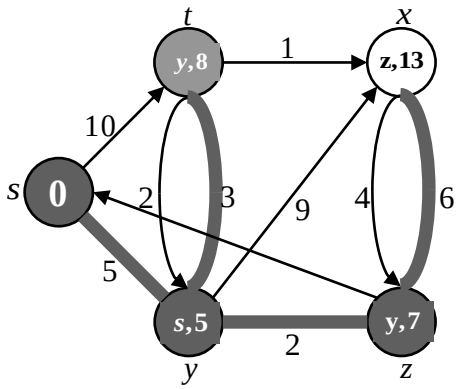



**Figura 2.10:** As conexões consideradas pelo algoritmo de *Dijkstra*.

Como o vértice *x* não tem conexões orientadas até outros nós o algoritmo procura outro ramo do grafo, portanto colocando-se nó vértice *t* é rotulado o nó adjacente sendo assim escolhido por ter a menor distância.

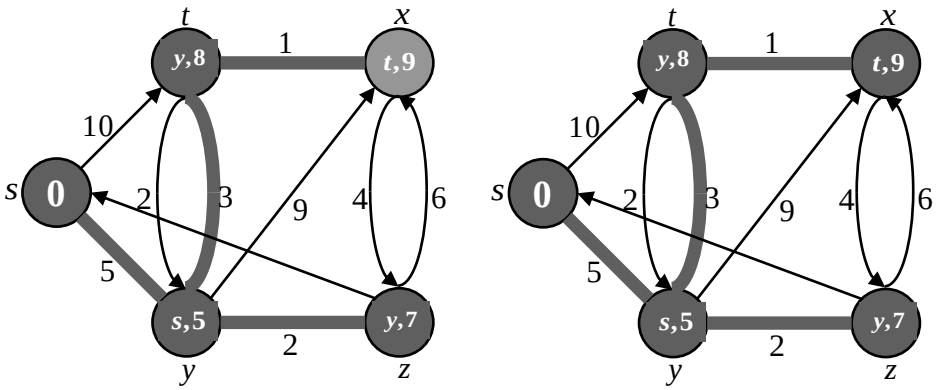



**Figura 2.11:** As conexões consideradas pelo algoritmo de Dijkstra**.**

A **Figura 2.11** mostra como rotula o último vértice temporariamente até assegurar-se que o trajeto escolhido pertence à menor distância, ficando assim a menor distância até cada um dos nós da rede.

O algoritmo de *Dijkstra* procura o caminho mais curto de um nó origem até todos os nós restantes da rede. Porém, quando o algoritmo procura o caminho mais curto entre um nó origem ou partida e um único nó destino ou chegada a sua complexidade é de ordem $N^2$ ($N^2$ operações).

Portanto, quando o algoritmo de *Dijkstra* busca o caminho mais curto entre o nó origem e todos os nós restantes da rede, sua complexidade aumenta para $N^3$.

Dessa maneira, pode se perceber que este não é um algoritmo completamente eficiente para redes de grande porte porque o tempo de computação aumenta em forma cúbica conforme cresce a dimensão do problema (medida através do número de nós e interconexões existentes entre eles).

### 2.5.2 Algoritmo de *Ford*

Ao contrario do algoritmo de *Dijkstra*, o algoritmo de *Ford* [2] pode ser aplicado a redes onde a distância $d_{ij}$ que é uma característica das conexões $c_{ij}$ tem associado valores negativos, não podendo contudo existir trajetos de valor negativo. Neste algoritmo supõe-se que existe pelo menos um caminho entre *i* e qualquer outro nó *j*.

Este algoritmo consiste em rotular os nós da rede, corrigindo todas as vezes que sejam necessárias, até que todos os rótulos satisfaçam a seguinte condição (de otimalidade):

$$\pi_f \leq \pi_k + d_{kf}, \qquad \text{para todo nó } k \text{ e } f \in C \qquad \textbf{(2.6)}$$

onde $\pi_k$ corresponde à soma das distâncias correspondentes as conexões percorridas do nó *i* para o nó *k*, e $d_{kf}$ consiste no valor de distância da conexão $c_{kf}$.

Neste algoritmo, os rótulos dos nós têm o mesmo significado dos rótulos utilizados no algoritmo de *Dijkstra*, só que agora os rótulos só se tornam definitivos após terminar a execução do algoritmo; quer dizer, nos estados intermediários do algoritmo apenas existem rótulos temporários isto significa que o melhor caminho encontrado só é definido no final da execução do algoritmo. O algoritmo de Ford está descrito a seguir em pseudocódigo:

```
[ξs,πs] ← [s,0]    (caminho mais curto de s para s tem comprimento 0 e s é o nó que antecede s).
[ξi, πi] ← [−, ∞] , ∀i ∈ N − { s }      (o caminho mais curto de s para i é desconhecido)
R ← { s }
Enquanto R ≠ ∅  Fazer
      R ← R − { i },  para i qualquer
      Para  todo arco (i, j)  Fazer
            Se  πj > πi + cij  Então
                  πj ← πi + cij
                  ξj ← i
                  Se  j ∉ R
                        Então  R ← R ∪ { j }
Fim do Enquanto
```

**Figura 2.12:** Pseudocódigo do Algoritmo de Ford.

A **Figura 2.12** apresenta um grafo orientado com 5 vértices que tem como principal característica o uso de pesos negativos, no qual é aplicado o algoritmo de Ford para ilustrar seu uso.

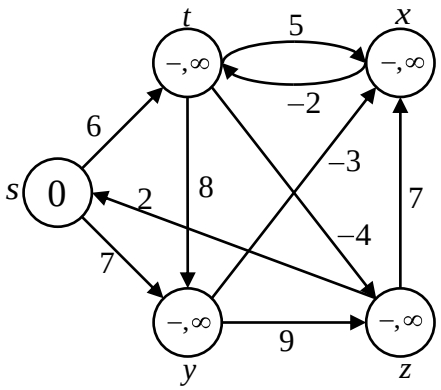



**Figura 2.13:** Grafo orientado

Considerando como ponto de partida o vértice *s*, observa-se que existem duas conexões a seguir-se $c_{st}$ e $c_{sy}$, rotulando seus respectivos vértices com rótulos temporários. Como se mostra na **Figura 2.14**:

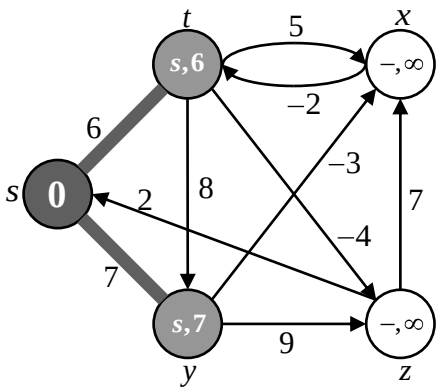



**Figura 2.14:** As trajetórias consideradas pelo algoritmo de Ford**.**

Agora, partindo de cada nó ou vértice (*t* e *y*) rotulado temporariamente são consideradas as novas possíveis conexões a escolher-se rotulando assim os de menor distância acumulada, como se apresenta na **Figura 2.15**.

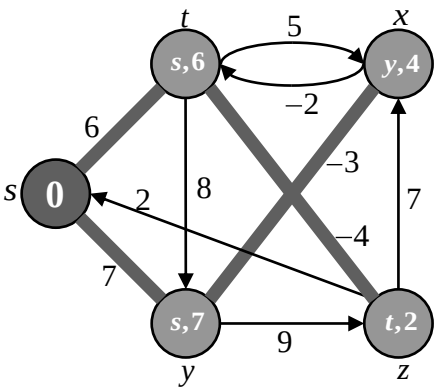



**Figura 2.15:** As trajetórias consideradas pelo algoritmo de Ford.

A partir dos últimos nós rotulados considera-se novamente as possíveis conexões a seguir-se, no caso de precisar atualizar, a rotulação é escolhida como sempre o rótulo de menor valor.

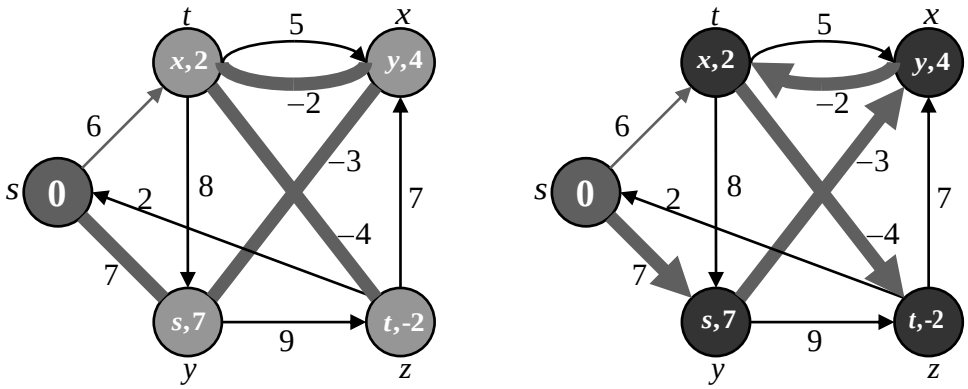



**Figura 2.16:** A trajetória considerada pelo algoritmo de Ford.

A **Figura 2.16** mostra como se rotulam os vértices temporariamente até assegurar-se que o caminho escolhido pertence á menor distancia.

Este algoritmo consiste fundamentalmente em calcular a distância de um nó inicial, *i*, até um único nó destino, *j*, sem passar por nenhum nó duas vezes, considerando cada nó intermediário, *k*, até que sejam consideradas todas as alternativas possíveis. Portanto, a complexidade do algoritmo correspondente é da ordem de $N^3$.

### 2.5.3 Algoritmo de *Floyd*

Ao pretender determinar o caminho mais curto entre todos os pares de nós, pode-se aplicar o algoritmo de *Dijkstra* ou *Ford n* vezes, utilizando cada nó sucessivamente como nó origem.

No entanto existem outros algoritmos para resolver este problema, como é o caso do algoritmo de *Floyd* [2], desde que não tenha trajetórias negativas; mas, a distância das conexões podem ter associados valores positivos ou negativos.

O algoritmo de *Floyd* utiliza uma matriz de ordem n*V*xn*V* (n*V* número total de nós da rede), das distâncias entre os nós, onde os nós que não tem uma conexão física que os liga têm associado uma distância direta infinita e as conexões com eles mesmos, têm associado uma distância de comprimento 0 (zero). O algoritmo de Floyd é descrito a seguir em pseudocódigo:

```
B ← matriz das distâncias diretas
P_ij ← j , ∀ i, j ∈ N
Para k = 1, ..., nV Fazer
        Para i, j = 1, ..., nV tal que i, j ≠ k Fazer
                Se B_ij > B_ik + B_kj Então
                        B_ij ← B_ik + B_kj
                        P_ij ← P_ik
Fim
```

**Figura 2.17:** Pseudocódigo do Algoritmo de Floyd.

## 2.6 O problema de determinar os *k* caminhos mais curtos

Este problema ocorre quando é necessário conhecer, além do caminho mais curto entre dois nós de uma rede, o 2º, o 3º, ... , até o *k*-ésimo caminho mais curto, tal que o valor do *k*-ésimo caminho é menor ou igual ao valor do *j*-ésimo caminho (considerando que $j > k$). Pretende-se então determinar uma ordenação crescente dos caminhos mais curtos, que constituem caminhos mais curtos alternativos ao melhor caminho.

O problema de determinar os *k* caminhos mais curtos pode ser aplicado a casos que se pretende determinar o caminho mais curto, mas seguindo certas restrições, em que a melhor solução pode não ser necessariamente o caminho mais curto. Portanto ao ordenarem-se os caminhos em ordem crescente a seus comprimentos, é possível chegar ao caminho mais curto que cumpra na sua vez as restrições impostas.

Nos problemas de circulação de veículos interessa, muitas vezes, conhecer caminhos alternativos ao mais curto, em virtude de existir engarrafamento ou corte em qualquer tramo de aquele caminho.

Nos problemas de encaminhamento de chamadas ou envio de mensagens, pode interessar conhecer caminhos alternativos ao mais curto, no caso de que alguma ligação deste caminho esteja saturada.

Além de caracterizar este tipo de algoritmo para a determinação dos *k* caminhos mais curtos, se pode combinar este problema do caminho mais curto com outros objetivos como foi mencionado anteriormente.

## 2.7 Um problema de caminho mais curto com múltiplos objetivos

Muitos problemas reais podem ser modelados baseando-se no problema de caminho mais curto e associar-se outros objetivos onde esses dados do modelo são organizados de acordo à estrutura de rede a ser analisada.

No caso de circulação veicular, determinar o melhor caminho entre dois nós de uma rede de estradas considerando dois objetivos, como são a distância percorrida e o tempo gasto. Considerando que o tempo gasto num percurso pode ser influenciado por atrasos, uma vez que o tempo de circulação entre dos nós da rede pode variar muito devido a problemas de engarrafamento, acidentes, obras de manutenção, filas de espera para pagamento de taxas, etc.

Geralmente, deseja-se determinar uma solução cujo valor econômico associado ao objetivo distância esteja o mais próximo possível do respectivo mínimo na forma de rentabilizar o lucro da empresa. Neste caso se têm um problema de caminho mais curto bi-objetivo.

Ao acrescentar-se mais um objetivo como a minimização da quantidade de nós visitados, o problema transforma-se num problema de caminho mais curto tri-objetivo. E se acrescentar o nível de periculosidade em termos de acidentes rodoviários, o problema transforma-se num de caminho mais curto tetra-objetivo ou denominado Multi-Objetivos, pois se deseja uma solução cujo valor associado a este quarto objetivo seja também o menor possível.

A solução destas situações reais cada vez mais complexas e dinâmicas se vê complicada com o aumento do tamanho da rede, isto se deve a rigidez dos modelos matemáticos ou modelos determinísticos utilizados, principal desvantagem dos métodos tradicionais de otimização exata.

O problema desta falta de flexibilidade foi um pouco reduzido a partir do momento em que se passou a associar técnicas de otimização com ferramentas de Inteligência Artificial, mais especificamente, com as ferramentas de busca heurística. De fato as heurísticas, se caracterizam pela sua flexibilidade e tem como objetivo encontrar soluções de boa qualidade num tempo computacional aceitável.

As heurísticas isoladamente também possuem suas limitações e a principal delas é a deficiência histórica de que em muitos casos, não conseguirem superar as armadilhas dos ótimos locais em problemas de otimização. Além disso, a falta de uma base teórica dos métodos heurísticos produz algoritmos muito especializados, ou seja, a pesar de sua flexibilidade em incorporar novas situações, o seu desempenho pode oscilar muito com modificações pequenas, no problema analisado.

Contorneando as desvantagens bem como aproveitando as principais vantagens nessa linha, a metodologia deste trabalho utiliza una técnica heurística para propor assim uma solução alternativa ao problema de roteamento através da otimização de seus objetivos. Considerando-se unicamente dois objetivos neste processo de otimização, onde se minimizará o primeiro objetivo (devido a *Somatória dos Trajetos* desde o nó partida até o nó chegada), e simultaneamente se maximizará o segundo objetivo (associado à *Multiplicação dos Índices de Confiabilidade* do trajeto selecionado).

Esses dois objetivos poderão ser conflitantes entre si, porque a mínima distancia não necessariamente indica a máxima confiabilidade já que a ligação entre esses nós pode ser feita, por exemplo, através de linha dedicada (tecnologia de comunicação mais lenta e pouco robusta); enquanto, que uma outra ligação pode ter comprimento maior, mas a confiabilidade correspondente pode ser superior devido a que os nós estão conectados através de fibras óticas (que permite maior rapidez de transmissão e os pacotes da mensagem podem conter mais informação).

## 2.8 Formulação Matemática

Nesta seção será descrita a formulação matemática proposta e desenvolvida no presente trabalho, para isso deve-se conhecer previamente a localização geográfica dos nós da rede e definir as conexões entre eles formando uma matriz de conexões onde cada componente de existir terá o valor de 1 e de não existir o valor de 0, assim onde exista valor de conexão 1 correspondera um valor de distância e confiabilidade respectivamente, para a busca da trajetória ótima de transmissão, quer dizer, a trajetória que esta associada à menor distância de transmissão e à de maior confiabilidade possível,

Basicamente, o *Problema de Roteamento* é um *Problema de Otimização Multi-Objetivo* que inclui um conjunto de *nV* parâmetros, um conjunto de *k*=2 funções objetivo, e um conjunto de *m* restrições. As características próprias de cada uma das redes são as que orientam a construção de cada uma das variáveis de decisão, *x*, e esta variável na sua vez serve como artifício para originar uma matriz de dimensão $nV_x nV$ e que correspondera as conexões escolhidas para formar a trajetória desejada.

Tomando primeiro objetivo "$f_1$" como o inverso da distância e o segundo objetivo "$f_2$" como a confiabilidade, formula-se o seguinte problema.

$$\begin{aligned} &\textit{maximizar} && w_1 f_1(x) + w_2 f_2(x) \\ &\text{Sujeito a} && r(x) = (r_1(x), r_2(x), \ldots, r_m(x)) \geq 0 \\ &\text{Onde} && x = (x_1, x_2, \ldots, x_{nV-1}) \end{aligned} \qquad \textbf{(2.4)}$$

sendo *x* a variável de decisão, *r* são as restrições e $w_k$ é o peso usado para ponderar a *k*-esima função objetivo.

Se cada uma das componentes da função objetivo está expressa na mesma grandeza e unidade de medida, então os pesos na equação **(2.4)** podem ser iguais a um ($w_1 = w_2 = 1.0$), significando com isso que cada função tem o mesmo nível de importância.

Entretanto, na maioria dos casos, cada componente da função objetivo está dada por grandezas diferentes. Uma heurística para escolha de cada peso $w_k$ está dada pela seguinte expressão:

$$w_k = \frac{1}{f_{k\max}} \qquad \forall\, k \in \{1, 2\} \qquad \textbf{(2.5)}$$

onde $f_{k\,max}$ corresponde limitante a um superior da função $f_k$.

A principal vantagem da abordagem por Soma Ponderada é sua implementação direta, devido a que se manipula uma única função objetivo. E sua principal desvantagem é que nem todas as soluções ótimas de um problema multiobjetivo correspondente podem ser pesquisadas quando a frente do espaço solução, que comumente se denomina frente de Pareto, não é convexo [8].

Na literatura se mostra vários teoremas que demonstram o relacionamento entre a solução ótima através da soma ponderada com as soluções ótimas de Pareto. A seguir enunciamos dois teoremas importantes [26].

**Teorema 2.1.** A solução de um problema representado por a equação **(2.4)** é Pareto ótima se os pesos são positivos para todos os objetivos.

**Teorema 2.2.** Se *x* é uma solução Pareto ótima para um problema multi-objetivo convexo, então existe um vetor de pesos positivos *w* não zeros, tal que *x* é uma solução para um problema representado por a equação **(2.4)**.

Em [8] ilustra-se como o calculo através da suma ponderada pode encontrar uma solução ótima de Pareto considerando um problema de dois objetivos, como é mostrado na **Figura 2.18**.

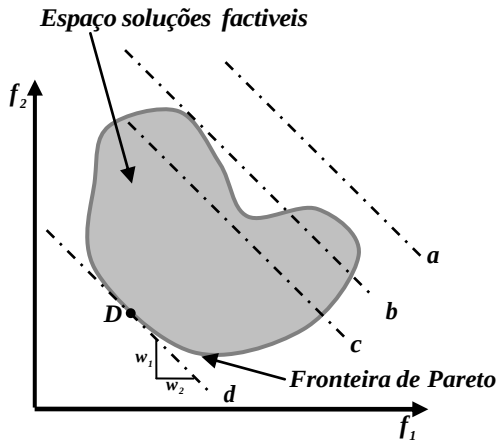



**Figura 2.18:** Ilustração da aproximação da soma ponderada sobre o frente convexo de Pareto ótimo.

O espaço de soluções factíveis e seu correspondente conjunto de soluções ótimas de Pareto são mostrados. Tendo-se dois objetivos, existem dos pesos $w_1$ e $w_2$, mas só um é independente, já que conhecendo qualquer um, o outro pode ser calculado por simples subtração.

Na **Figura 2.18** a superfície de contorno que se forma pela combinação dos objetivos é a que se denomina o espaço de soluções factíveis. Mas conhecendo-se os pesos é possível calcular a função $f(x)$ como é mostrado pelas linhas 'a', 'b', 'c' e 'd' da **Figura 2.18**. Sendo que $f(x)$ é uma combinação linear de ambos objetivos $f_1$ e $f_2$, se espera como linha de contorno de $f(x)$ uma linha reta no espaço solução. Isto é porque qualquer solução sobre a linha de contorno terá o mesmo valor de $f(x)$. Observe-se que está linha de contorno não é uma linha arbitraria já que sua pendente esta relacionada à escolha do vetor pesos. Note-se que a pendente na **Figura 2.18** é $-w_1/w_2$. A ubiquação da linha depende do valor de $f(x)$ sobre qualquer ponto na linha. O efeito de abaixar a linha de contorno de 'a' a 'b' é de fato um salto das soluções de uns valores mais elevados de $f(x)$ a um valor mais baixo.

Quando o problema representado por a equação **(2.4)** requer ser minimizado, a tarefa é encontrar a linha de contorno de valor mínimo de $f(x)$. Isto acontece com a linha de contorno que é tangencial ao espaço solução e cai no canto inferior esquerdo deste espaço. Na **Figura 2.18**, esta linha é denominada com a letra 'd'. O ponto tangente 'D' é a solução mínima de $f(x)$, e consequentemente é a solução ótima de Pareto correspondente a esse vetor peso.

Se o vetor pesos é modificado, a pendente da linha de contorno será diferente. Mas o procedimento descrito, em geral deverá resultar em uma solução ótima diferente. Utilizando o teorema 2.2, podemos resolver o problema da equação **(2.4)** com múltiples vetores de pesos positivos, um à vez, encontrando uma solução ótima de Pareto diferente por vez, consiguindo-se assim encontrar as múltiplas soluções ótimas de Pareto.

A variável de decisão $x$ da **Figura 2.19** foi suposta na forma de um vetor para ajudar a interpretar cada valor de $x$ na matriz de conexões. Assim é assumindo um vetor $x$ qualquer, baseado na rede exemplo da **Figura 2.4** e que corresponde a um possível conjunto de conexões que segue uma trajetória continua desde o nó 1 ao nó 5. O vetor $x$ é um subconjunto do conjunto completo de conexões $c_{ij}$, portanto se utiliza este vetor para formar uma matriz $M$.

Na mesma forma que $C$ a matriz $M$ é uma matriz quadrada de dimensão igual a $nV$x$nV$.

$$x = \begin{bmatrix} c_{13} \\ c_{24} \\ c_{32} \\ c_{45} \\ 0 \end{bmatrix} \qquad M = \begin{bmatrix} 0 & 0 & m_{13} & 0 & 0 \\ 0 & 0 & 0 & m_{24} & 0 \\ 0 & m_{32} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_{45} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Figura 2.19:** Construção da matriz *M* a partir do vetor *x*.

Na **Figura 2.19** se apresenta o vetor *x* que da origem à matriz *M* correspondente as conexões possíveis a percorrer-se. Está matriz pode ser definida matematicamente como:

$$Se\ i,j \in \{1,2,...,nV\},\ \forall m_{ij} \neq 0 \Rightarrow \exists\, c_{ij} \neq 0 \qquad \textbf{(2.6)}$$

A matriz através da qual se representa a trajetória é a denominada matriz de conexões possíveis a percorrer-se $M = \{m_{ij}\}$. Cada elemento $m_{ij}$ da matriz assume um dos seguintes valores: 1 (hum), se (i,j) $\in M$ ou 0 (zero), se (i,j) $\notin M$.

$$M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Figura 2.20:** Representação matemática de uma trajetória qualquer.

Essas componentes $\{m_{11}, m_{12}, ....., m_{nVnV}\}$ da matriz de conexões possíveis a percorrer-se para chegar de um nó origem ou partida a um nó destino ou chegada, servem para maximizar cada um dos objetivos ao multiplicar-se elemento por elemento da matriz *M* com cada uma das matrizes objetivos por vez, *D* ou *F*.

$$\begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1nV} \\ m_{21} & m_{22} & & m_{2nV} \\ \vdots & & & \\ m_{nV1} & m_{nV2} & & m_{nVnV} \end{bmatrix} \cdot \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1nV} \\ d_{21} & d_{22} & & d_{2nV} \\ \vdots & & & \\ d_{nV1} & d_{nV2} & & d_{nVnV} \end{bmatrix}$$

$$= \begin{bmatrix} m_{11}d_{11} & m_{12}d_{12} & \cdots & m_{1nV}d_{1nV} \\ m_{21}d_{21} & m_{22}d_{22} & & m_{2nV}d_{2nV} \\ \vdots & & & \\ m_{nV1}d_{nV1} & m_{nV2}d_{nV2} & & m_{nVnV}d_{nVnV} \end{bmatrix} = \boldsymbol{M \cdot D}$$

**Figura 2.21:** Multiplicação elemento por elemento da matriz *M* com a matriz *D*.

- *Objetivo 1:* Como o objetivo é otimizar o roteamento de uma rede se faz necessário maximizar o parâmetro distância correspondente às conexões que se devem percorrer para chegar de um ponto a outro. Dessa maneira, a função objetivo $f_1$ pode ser expressa da seguinte forma:

$$f_1 = \frac{1}{m_{11}d_{11} + m_{12}d_{12} + \cdots + m_{21}d_{21} + \cdots + m_{nVnV}d_{nVnV}}$$

$$\textit{Maximizar} \qquad f_1 = \frac{1}{\sum_{\substack{i=1 \\ j=1}}^{V} m_{ij} \cdot d_{ij}} \qquad \textbf{(2.7)}$$

- *Objetivo 2:* No sentido de rotear a rede de comunicação é necessário maximizar o parâmetro confiabilidade das conexões utilizadas para chegar de um ponto a outro da rede. O valor numérico dessa função objetivo $f_2$ é calculado a partir da seguinte expressão matemática:

$$f_2 = m_{11}cf_{11} \times m_{12}cf_{12} \times \cdots \times m_{21}cf_{21} \times \cdots \times m_{nVnV}cf_{nVnV}$$

$$\textit{Maximizar} \qquad f_2 = \prod_{\substack{i=1 \\ j=1}}^{nV} m_{ij} \,.\, cf_{ij} \tag{2.8}$$

Como introdução na revisão bibliográfica definirmos basicamente a técnica meta-heurística mais utilizada no processo de roteamento de uma rede de comunicação.

## 2.9 Algoritmos Genéticos.

Em 1975, Holland desenvolveu uma metodologia baseada na seleção e genética natural para otimização de sistemas artificiais. Posteriormente, esta metodologia passou a ser conhecida como Algoritmos Genéticos (AG), os quais têm sido aplicados na resolução de problemas em diversas áreas da ciência: otimização de dispositivos eletromagnéticos [1]; sistemas elétricos de potência e outros.

Muitos projetos incluem problemas que são de difícil solução na prática, tais como: problemas do tipo NP-completo sendo destaque o caixeiro viajante, funções multimodais, etc. Nestes casos, métodos heurísticos são aplicados com o intuito de reduzir o espaço de busca e, como conseqüência, gerar o mais rapidamente possível conjuntos aproximados de soluções.

Os AG processam algumas possíveis soluções candidatas de uma determinada função avaliação de uma forma iterativa até atingir um determinado critério de parada. Cada solução candidata é comumente representada através de um vetor das variáveis independentes que modelam o problema sob estudo. Cada variável independente pode ser codificada utilizando o Sistema Binário, ou bem, representando-os através do ponto flutuante. Na literatura dos AG, esse vetor de variáveis independentes é conhecido como cromossomo. Uma descrição mais detalhada sobre os AG será dada no Apêndice A.

A metodologia de otimização dos AG está formado por quatro estágios ou procedimentos. O primeiro estagio é executado uma única vez e os restantes são repetidos ciclicamente até satisfazer algum critério de parada. Em cada um desses estágios o tamanho global da população se mantém constante.

A seguir, são descritos cada um desses estágios:

1. Geração aleatória de uma população de indivíduos[1] codificados para a aplicação desejada. Na realidade são inicializados aleatoriamente cada um dos valores dos parâmetros (variáveis independentes) que conformam um determinado cromossomo.

2. Avaliação dos indivíduos. Neste estágio é calculado o valor numérico da função objetivo associado a cada indivíduo.

3. Seleção dos indivíduos. São selecionados probabilisticamente os melhores indivíduos tendo em conta o valor da sua função de aptidão (função objetivo).

4. Geração da nova população por meio dos operadores genéticos de cruzamento e mutação.

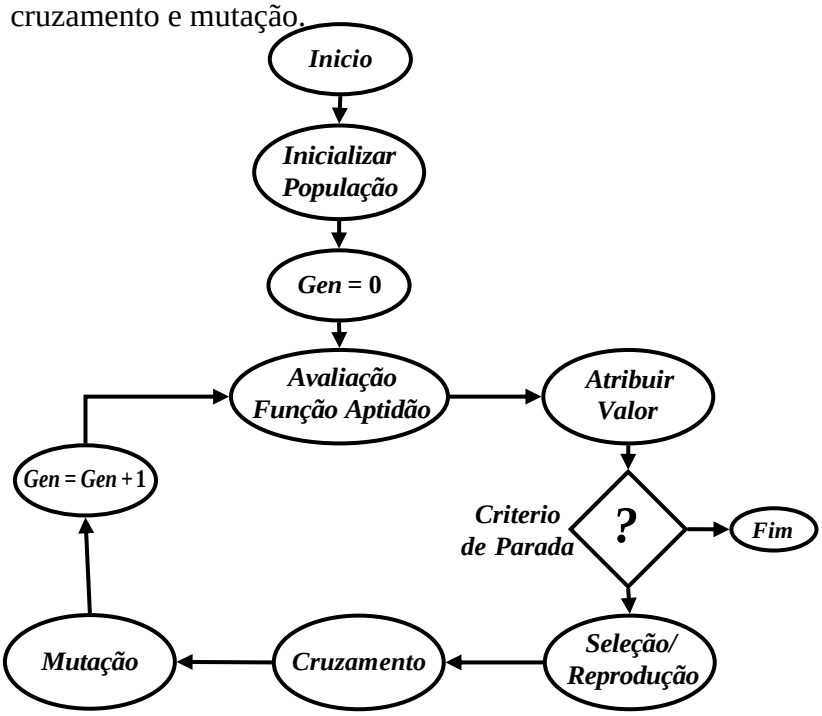



**Figura 2.22:** Diagrama dos Algoritmos Genéticos.

Na **Figura 2.24** será apresentado um diagrama de fluxo dos *AG* como a ferramenta que ajudará a visualizar melhor as quatro etapas em que se divide o processo de otimização dentro destes.

[1] Nos AG, cada indivíduo corresponde a uma estrutura de dados formada principalmente pelo correspondente cromossomo (vetor de variáveis independentes), a dimensão do cromossomo, e o valor numérico da função objetivo.

A seguir se apresenta uma revisão dos trabalhos desenvolvidos nesta linha de pesquisa o que ajudará a ter uma idéia mais clara dos métodos utilizados assim como dos mecanismos de aplicação das técnicas de alto desempenho proposto por alguns pesquisadores tendo como objetivo melhorar o desempenho (tempo) e a qualidade (solução global) dos diferentes algoritmos.

## 2.10 Revisão Bibliográfica.

Obter o caminho mais curto e confiável é um problema clássico de otimização combinatória, que continua sendo o centro de atenção dos pesquisadores, por se tratar de um problema para o qual é quase improvável a existência de um algoritmo que obtenha uma solução ótima em tempo polinomial. Portanto, a busca de um algoritmo de otimização tem sido o enfoque principal dos trabalhos nesta área de pesquisa.

Dengiz (1997) apresenta uma metodologia baseada nos *AG* [10] para minimizar o trajeto de transmissão e maximizar a confiabilidade de uma rede de comunicação. Nesse trabalho são apresentadas novas técnicas de codificação das variáveis (cromossomo), inicialização da população, bem como operadores de busca local especializados.

O objetivo de todas essas novedosas implementações consiste em gerar soluções factíveis já no inicio da criação da população, através da utilização do *spanning tree* (árvores abrangentes) [16]. Dessa maneira, cada solução está formada por um vetor cujas componentes representam as conexões entre dois nós consecutivos. A construção de cada um desses cromossomos começa a partir de qualquer nó da rede, e com uma simples heurística, é decidido que trajeto seguir a partir desse nó seguindo assim até completar até formar uma árvore de ramais desconexas e radiais. Assim, o cromossomo está formado por cada uma das possíveis conexões que formam um trajeto completo, ligando entre si todos os nós da rede.

Mas mesmo utilizando a técnica do *spanning tree*, apresenta aleatoriedade na geração de cada conexão, por causa de que em cada componente trabalha-se com limites expandidos fixos; o qual, ainda pode gerar indivíduos não viáveis. Nesse caso, utiliza-se um método de correção de cromossomos para a sua correção de tal forma que as N soluções sejam todas factíveis.

Dessa maneira, o autor apresenta os resultados da aplicação do algoritmo proposto em diferentes redes de comunicação não direcionadas, sendo a maior delas uma rede com 20 nós.

Deeter et al. (1998), utilizam os algoritmos genéticos [9] como metodologia para identificar o melhor caminho a ser percorrido considerando conjuntamente a confiabilidade e os custos por unidade de distância. Deeter et al. assumem que os nós da rede são perfeitamente confiáveis e não falham, portanto as conexões têm como características dois possíveis estados: bom ou falido, simplificando assim o problema. O método apresentado é testado com redes de diferentes tamanhos, sendo a rede de maior tamanho igual a 18 nós.

Inagaki et al. (1999), propõem como alternativa um método de roteamento utilizando *AG* [21]. Este método não seleciona necessariamente a rota mais curta, senão são determinadas as *k* trajetórias descendentes através da memorização das possíveis soluções produzidas no processo de busca.

Inagaki et al. representam os cromossomos através de uma seqüência de números inteiros e cada gene, ou componente, do cromossomo representa um determinado nó da rede. Todos os cromossomos da população têm igual comprimento. Então, o cromossomo representa o trajeto de comunicação sorteado pelo AG.

A **Figura 2.26**, mostra a um exemplo de trajetória indicada através do cromossomo.

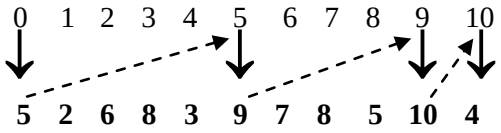



**Figura 2.23:** Representação do cromossomo.

A **Figura 2.26** mostra que as posições no cromossomo se enumeram começando desde o zero. Então seguindo a ordem: a posição zero contém o número cinco (nó 5), o qual conduz à posição 5 no cromossomo. Nessa posição está indicada o número nove (nó 9) o qual conduz à posição nove no cromossomo, nessa posição se encontra o número dez. Então, o número dez na posição nove do cromossomo conduz à posição dez. Nessa posição se encontra o número quatro. O número quatro aponta à posição quatro no cromossomo. Nessa posição se tem o número três, o qual aponta à posição três no cromossomo. Nessa posição se tem o número oito, o qual aponta à posição oito. Esse processo continua até atingir a trajetória: 5 – 9 – 10 – 4 – 3 – 8.

O algoritmo genético utilizado memoriza as possíveis soluções para assim escolher a melhor, portanto o autor destaca a possibilidade de conseguir escolher outra rota alternativa sem a necessidade de ser a mais curta.

Laufer, (2000) usa a combinação de algoritmos diferentes atuando conjuntamente na solução de um mesmo problema global, comumente conhecido como Team Algorithm [24]. A solução de problemas utilizando Team Algorithm pode ser implementada em paralelo, designando cada um dos diferentes processos aos processadores disponíveis de um sistema distribuído assíncrono, como é uma rede de computadores. Esta estrutura denomina-se A-Teams (Asynchronous Team).

Portanto neste trabalho é proposto um A-Team, baseado na combinação de algoritmos genéticos com operadores de busca local, para o projeto econômico de redes confiáveis de computadores. Apresentando também as vantagens obtidas utilizando as técnicas de processamento paralelo assim como a comparação com outros trabalhos, limitando-se à utilização de redes de até 18 nós.

Duarte [11], pretende encontrar as melhores topologias para a rede, a partir da informação sobre a localização geográfica dos centros de comunicação, os custos, e a confiabilidade dos diferentes tipos de tecnologias de conexão disponíveis. A técnica utilizada para a implementação deste trabalho são os algoritmos evolutivos multi-objetivos. Especificamente é utilizada uma versão do Strength Pareto Evolucionary Algorithm (*SPEA*) que serve para encontrar um conjunto de soluções ótimas, e não uma única solução. O autor apresenta resultados com redes de até 19 nós.

Chang et al. (2002), apresentam uma metodologia de busca do caminho mais curto, baseado nos *AG*, para ser aplicado a um problema de roteamento, típico em redes computacionais como as que existem na *Internet*, *Redes Móveis Ad hoc*, entre outros [5]. Foram utilizados cromossomos de comprimento variável. Cada cromossomo representa uma possível rota de comunicação ligando entre nós (ou *routers*). Cada uma dessas rotas de comunicação corresponde à componente do cromossomo onde são representados.

Naquele trabalho foi implementado um operador de cruzamento que faz intercambio parcial de *genes* entre um determinado par selecionado de *cromossomos*. Porém, o autor não restringe a variação de cada gene, gerando-se soluções inviáveis, ou seja, caminhos descontínuos. Dessa maneira, foi desenvolvido também um algoritmo de correção que repara indivíduos após serem aplicados sobre eles os operadores genéticos.

Todas as implementações foram feitas utilizando *MatLab*®. Assim, o algoritmo de roteamento proposto, foi aplicado em diferentes redes de comunicação existentes na literatura, sendo a maior delas uma rede com 50 nós.

Wei et al. (2004), utiliza também a técnica dos *Algoritmos Genéticos* para resolver o problema do caminho mais curto [33]. Na metodologia, cada um dos números inteiros existentes em cada posição do cromossomo representa um determinado nó da rede. Nesse trabalho, os cromossomos têm comprimento variável, desconsiderando-se nós que não forma uma trajetória de comunicação. No trabalho de Wei o primeiro nó representava o ponto de partida e o ultimo nó do cromossomo representava o ponto final, assim na forma de simplificar as operações, o autor omitia os dois nós trabalhando unicamente com os nós intermediário.

Como se apresentou em outros trabalhos o Algoritmo Genético precisava de uma função de correção que neste caso reparava os indivíduos que tinham genes denominados redundantes. A metodologia foi aplicada numa rede não direcionada de 100 nós.

Nesmachnow et al. (2005), utilizam algumas características do problema de Steiner para o desenho de redes confiáveis, o mesmo que é modelado através de técnicas meta-heurísticas [27].

O autor junta as características mais importantes de algumas técnicas meta-heurísticas como são os algoritmos genéticos, simulated annealing, pesquisa em vizinhança variável entre outros na implementação da metodologia que determine a confiabilidade da rede baseado na distância.

Compara os resultados obtidos utilizando separadamente com cada técnica como trabalhando em conjunto com alguma outra na forma de mostrar a eficiência obtida neste trabalho. Apresentando os resultados comparativos com redes de até 13 nós.

Os principais objetivos no projeto de redes tem sido tradicionalmente o custo e a confiabilidade. Ambas as funções tem sido estudadas em quase todos os artigos encontrados referentes a problemas de otimização do roteamento das redes. No caso especifico do roteamento das redes se há apresentado em dois variantes. Na primeira se expressa o problema como minimizar o custo mantendo uma restrição sobre a confiabilidade, enquanto na segunda se apresenta o problema como: maximizar a confiabilidade mantendo umas restrições sobre o custo. Uma terceira variante tem sido sugerida como é maximizar todos os objetivos e aplicar a soma ponderada.

# Capítulo 3

# III. Algoritmo Genético proposto para a resolução do problema de roteamento com confiabilidade.

## 3.1 Conceitos Gerais

Este capítulo destina-se ao detalhamento de uma proposto de um algoritmo genético para a resolução do problema de roteamento com confiabilidade formulado no capítulo anterior.

Como já foi descrito existem duas formas para resolver este tipo de problemas:

- Implementar uma metodologia baseada em métodos determinísticos dentre os mais tradicionais estão os algoritmos de Dijkstra, Ford e Floyd, os que carregam altos custos computacionais, ou,
- Utilizar uma metodologia baseada em métodos probabilísticos ou meta-heurística como os Algoritmos Genéticos, *AG*, que se caracteriza por ser eficientes na hora de solucionar alguns problemas complexos de otimização.

Depois de pesquisar as alternativas observou-se que geralmente os métodos determinísticos trabalham como otimizador local sendo necessário o auxilio de outros métodos, motivando a utilização dos métodos meta-heurísticos na forma que estes consigam otimizar o processo de roteamento.

## 3.3 Metodologia utilizada no processo de roteamento.

Como vai se guardar a informação da rede dentro dos AG é a primeira pergunta que surge na hora de implementar a metodologia, ou seja, precisa-se definir qual será o critério para a codificação desta nos AG. Neste trabalho o critério de codificação vem dado pelo vetor *x* de decisão apresentado na **Figura 3.1** e que corresponde as possíveis alternativas de conexão para formar uma trajetória, já que na própria definição do vetor diz que este guarda as informações necessárias para o processo de roteamento a implementar-se.

Como se conhece o algoritmo genético, na sua forma original, trabalha normalmente com uma representação binária (0 e 1) a mesma que é associada a uma solução ou componentes de uma solução do problema. Embora esta representação tenha se mostrado eficiente para vários problemas, observou-se na medida em que foram crescendo as aplicações de *AG´s*, que esta não é a única alternativa, surgindo outras como é a representação por inteiros, onde o cromossomo é descrito por um vetor de números inteiros.

$$x \Rightarrow \begin{bmatrix} 0 \; ou \; c_{12} \; ou \; c_{13} \\ 0 \; ou \; c_{23} \; ou \; c_{24} \\ 0 \; ou \; c_{32} \; ou \; c_{34} \; ou \; c_{35} \\ 0 \; ou \; c_{45} \\ 0 \; ou \; c_{51} \; ou \; c_{54} \end{bmatrix}$$

$$x \Rightarrow \left\{ \begin{bmatrix} c_{12} \\ c_{24} \\ c_{32} \\ 0 \\ c_{51} \end{bmatrix}, \begin{bmatrix} c_{13} \\ 0 \\ c_{35} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ c_{23} \\ c_{34} \\ c_{45} \\ 0 \end{bmatrix}, \begin{bmatrix} c_{12} \\ c_{24} \\ 0 \\ c_{45} \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} c_{13} \\ c_{24} \\ c_{32} \\ c_{45} \\ 0 \end{bmatrix} \right\}$$

**Figura 3.1:** As possíveis conexões a ser escolhidas e o conjunto de vetores *x*.

- Como o vetor de decisão *x* não pode ser implementado na forma que foi apresentado na **Figura 3.1** já que está construído por variáveis numéricas e alfanuméricas na sua vez, utiliza-se neste caso como um artifício o número da posição correspondente à conexão $c_{ij}$ escolhida. Assim $c_{13}$ corresponde a 2 ou 2 corresponde à conexão $c_{13}$.

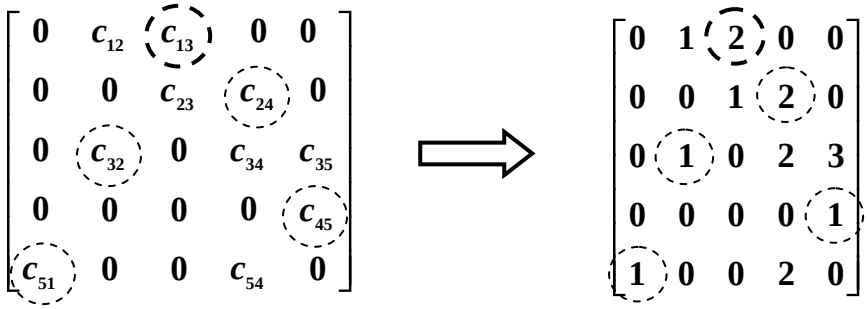



**Figura 3.2:** Representação do número da posição correspondente à conexão.

Suponhamos um vetor *x* que contenha uma trajetória qualquer.

$$X = \begin{bmatrix} 0 \text{ ou } c_{12} \text{ ou } c_{13} \\ 0 \text{ ou } c_{23} \text{ ou } c_{24} \\ 0 \text{ ou } c_{32} \text{ ou } c_{34} \text{ ou } c_{35} \\ 0 \text{ ou } c_{45} \\ 0 \text{ ou } c_{51} \text{ ou } c_{54} \end{bmatrix} \Rightarrow X = \begin{bmatrix} c_{13} \\ c_{24} \\ c_{32} \\ c_{45} \\ c_{51} \end{bmatrix}$$

**Figura 3.3:** Representação do vetor de decisão.

Sendo que cada linha do vetor *x* mostrado ao lado esquerdo da **Figura 3.3** corresponde às conexões existentes por linha e que poderiam fazer parte de uma trajetória, a escolha aleatória das conexões que formam o vetor *x* da direita de dita figura não necessariamente corresponde a uma trajetória correta, portanto se denomina inicialmente como uma possível trajetória.

A cada uma das variáveis do vetor *x* na **Figura 3.4** se atribui um número em ordem ascendente seguindo a mesma ordem das conexões existentes em cada linha, ou também se poderiam enumerar em ordem de direita a esquerda cada conexão existente por linha na matriz de conexões *C* da **Figura 3.5**:

$$X = \begin{bmatrix} 0 \text{ ou } 1 \text{ ou } 2 \\ 0 \text{ ou } 1 \text{ ou } 2 \\ 0 \text{ ou } 1 \text{ ou } 2 \text{ ou } 3 \\ 0 \text{ ou } 1 \\ 0 \text{ ou } 1 \text{ ou } 2 \end{bmatrix} \Rightarrow X = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

**Figura 3.4:** Representação numérica de uma possível trajetória gerada aleatoriamente.

Observando simultaneamente a **Figura 3.3** e **3.4** pode-se dizer que a conexão $c_{13}$ corresponde a 2 ou 2 corresponde à conexão $c_{13}$ e que a conexão $c_{35}$ corresponde a 3. Finalmente na forma de exemplificar o vetor *x* da direita da **Figura 3.4** corresponde à forma como estará codificada cada uma das conexões da rede dentro do algoritmo genético.

Está representação numérica do vetor de decisão apresentada na **Figura 3.4** ajudará a entender melhor como se definem os limites máximos e mínimos. Escolhendo-se zero (0) como o limite mínimo ou inferior e o número de conexões presentes $c_{ij}$ (diferentes de zero) em cada uma das filas da matriz de conexões *C* como limite máximo ou superior.

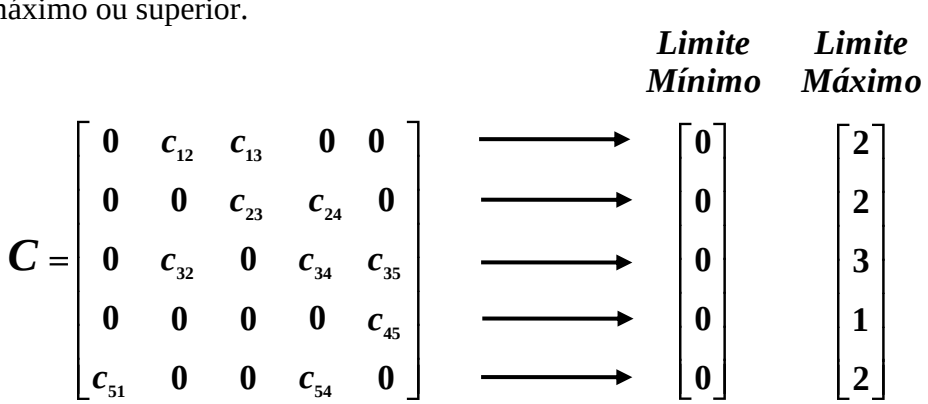



**Figura 3.5:** Representação dos limites.

Estes limites determinar um intervalo de possíveis valores para cada linha da matriz de conexões de tal forma que ao longo da execução do *AG* os operadores genéticos (*Cruzamento* e *Mutação*) gerem novos indivíduos respeitando-os, além que seja impossível gerar conexões que não existam na rede.

Portanto existem outras considerações que devem ser tomadas ao impor os valores nos limites, já que estes também podem produzir problemas nas trajetórias geradas, como se observa ao graficar o vetor *x* suposto na **Figura 3.3**.

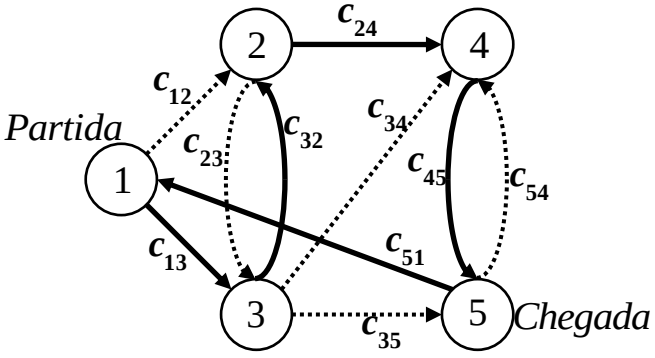



**Figura 3.6:** Representação gráfica do vetor de decisão da Figura 3.5.

Através da **Figura 3.6** pode se observar que existe um ciclo fechado de conexões. Para analisar o problema recordamos como dados do problema o ponto de partida e o ponto de chegada, neste caso o ponto de partida é o nó 1 e como ponto de chegada o nó 5.

*Hipóteses* 1: Começaremos revisando quando na trajetória da **Figura 3.7,** o ponto de partida se escolhe o valor do limite mínimo, ou seja, zero como valor presente no vetor de decisão *x*, na vez de $c_{13}$.

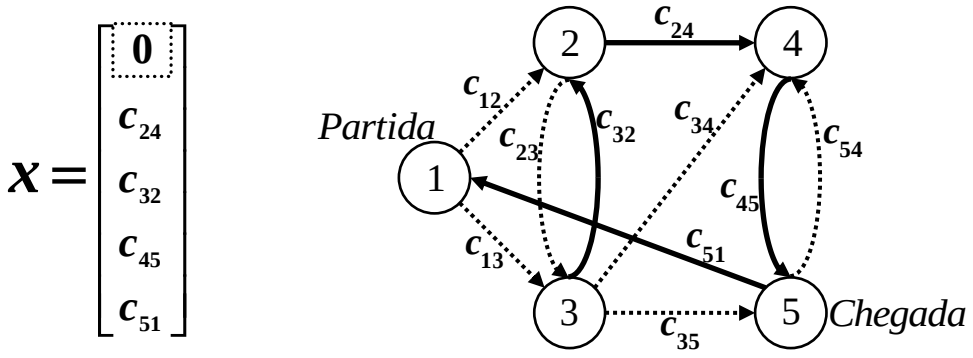



**Figura 3.7:** Representação do vetor de decisão modificado no ponto de partida.

Como se observa não existe nenhuma conexão saindo do nó 1 que corresponde ao nó de partida, portanto o trajeto descrito por esse vetor nunca poderia ser um caminho válido. Concluindo-se finalmente que o nó de partida não pode ter como limite mínimo o zero (0) este deve ser considerado sempre com um valor de um (1).

*Hipóteses* 2:Analisaremos agora a condição que se deve ter presente para que o ponto de chegada não enlace outro nó na trajetória ocasionando uma continuação na trajetória ou um ciclamento da mesma. Exemplificaremos através do vetor de decisão que gerou a **Figura 3.7.**

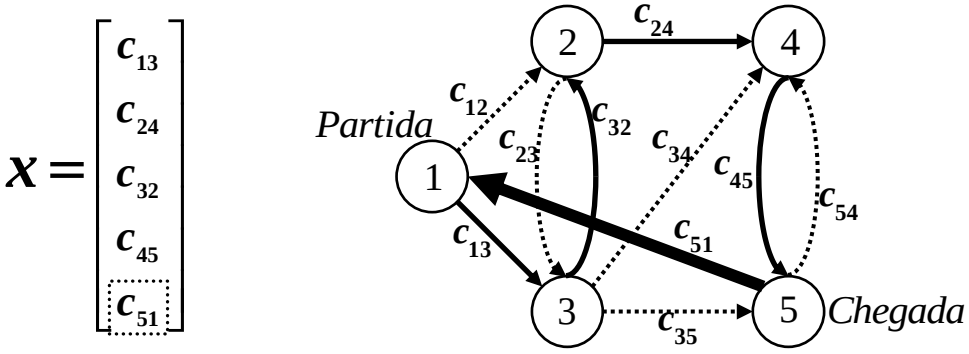



**Figura 3.8:** Representação de um ciclo gerado através do vetor de decisão.

O vetor de decisão da **Figura 3.8** mostra que a conexão $c_{51}$ não deveria existir porque está indica que existe continuação da trajetória além de agregar valor em termos de distância, confiabilidade, etc. Para evitar isto se deve ter sempre presente que não se deve escolher uma conexão da linha correspondente ao nó de chegada que neste caso é 5, isto quer dizer que sempre deve assumir-se o valor de zero (0) no limite superior ($x_5^{(s)} = 0$) ou máximo da linha correspondente ao nó de chegada.

O limite do vetor de decisão fica modificado ao considerarem-se as características apresentadas como conclusão na hipótese 1 e hipótese 2.

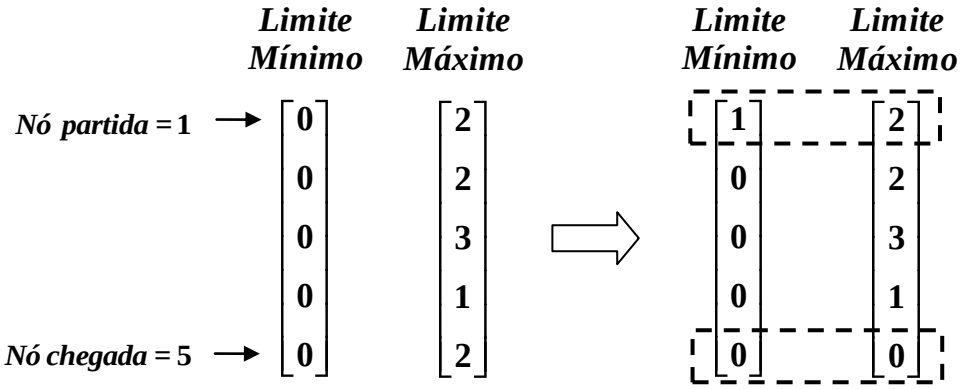



**Figura 3.9:** Representação dos limites uma vez adicionada as características próprias do problema.

Ao modificarem-se os limites que ajudaram na criação do vetor de decisão *x*, pode-se observar claramente na **Figura 3.9** que a fila correspondente ao ponto de chegada não apresenta uma faixa de valores entre o limite mínimo e máximo, portanto não justifica considerar este parâmetro como parte do *AG*. Ficando reduzida a dimensão do vetor de decisão em uma unidade *nV*-1.

A partir de agora o denominado vetor de decisão *x* que sempre foi um vetor coluna será representado como um vetor fila e cada um destes vetores filas serão um cromossomo ou indivíduo do *AG* na vez que cada componente do cromossomo se denomina gene, portanto um cromossomo tem *nV*-1 genes.

Assim definidos os limites do vetor *x* é gerada a população inicial de indivíduos ou cromossomos a qual se cria através de um gerador randômico de números inteiros já que a rede se pode ser modelada através de estes como se observou anteriormente. Além disso, a codificação do *AG* em números inteiros evita perda de tempo computacional ao contrastar-se com a conversão a binária.

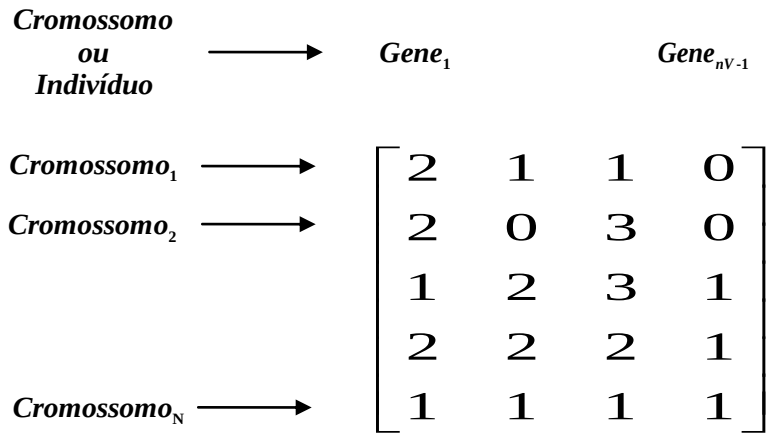



**Figura 3.10:** População gerada aleatoriamente.

Na **Figura 3.10** apresentam-se a primeira população formada por *N* indivíduos, onde cada gene de um cromossomo ou indivíduo representa de maneira numérica uma conexão $c_{ij}$ que pertence à matriz de conexões C.

A representação do vetor de decisão é apresentada na **Figura 3.11** em forma de um vetor fila formando finalmente uma matriz de *N* vetores de decisão, a que se denomina como matriz de decisão.

$$X = \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 2 & 0 & 3 & 0 \\ 1 & 2 & 3 & 1 \\ 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

**Figura 3.11:** Matriz de decisão X.

Lembrando como detalhes importantes desta matriz de decisão *X* o seguinte:

1. O número correspondente á coluna em analise corresponde ao número da linha na matriz de conexões *C*, já que o vetor *x* era inicialmente um vetor coluna e passo a ser um vetor linha. Deve ser observando o espaço deixado pelo nó correspondente ao nó de chegada que neste caso é o cinco, portanto não se tem um pulo na numeração das colunas.

2. O valor de cada gene corresponde a uma das conexões $c_{ij}$ existentes, ou seja, diferentes de zero, as que são enumeradas por linha em ordem ascendente de direita a esquerda sempre começando de hum (1).

Continuando na descrição da metodologia passo a passo analisaremos cada um dos vetores de decisão criados aleatoriamente na **Figura 3.12**. Assim começaremos:

**Primeiro Individuo:**

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ x = 2 & 1 & 1 & 0 \end{array}$$

**Figura 3.12:** Primeiro vetor *x* gerado aleatoriamente.

Agora se cria uma matriz *M* nula, ou seja, todos seus componentes são iguais a zero, e de dimensão igual à matriz de conexões onde a dimensão destas matrizes esta associada à quantidade de nós existentes na rede.

Esta matriz *M* que inicialmente é nula começará a variar seus componentes conforme está descrito no vetor de decisão apresentado na **Figura 3.13**. No momento que se identifica na matriz *C* a conexão localizada na posição sinalada por cada um dos genes pertencente a *x*, se procede a cambiar o valor de zero (0) presente na matriz *M* por o valor de um (1) em essa posição.

$$C = \begin{bmatrix} 0 & c_{12} & c_{13} & 0 & 0 \\ 0 & 0 & c_{23} & c_{24} & 0 \\ 0 & c_{32} & 0 & c_{34} & c_{35} \\ 0 & 0 & 0 & 0 & c_{45} \\ c_{51} & 0 & 0 & c_{54} & 0 \end{bmatrix} \qquad M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Figura 3.13:** Matriz de conexões possíveis a percorrer *M*.

A matriz *M* gerada através do vetor de decisão *x*, nem sempre tem conexões que se ligam entre si para formar um caminho contínuo. Sendo assim, sempre deve verifica-se essa condição.

A necessidade de cuidar a existência de um caminho contínuo faz obrigatório percorrer o caminho descrito por *M* ao tempo que exige criar um vetor que funcione como etiqueta (*Flag*), portanto deve ser um vetor tipo coluna. Para começar a percorrer a trajetória nos colocamos na linha correspondente ao ponto de partida e vamos a onde exista como valor um 1, as coordenadas (i,j) em que se encontra o 1 nos direcionaram até a próxima linha repetindo-se este processo até chegar ao ponto final. Sim esquecer

incrementar o valor da etiqueta na fila correspondente no momento que se posiciona em uma linha determinada da matriz.

No caso da matriz *M* apresentada na **Figura 3.14** nos colocamos na linha número um e procuramos a coordenadas (i,j) onde se tem um número 1, sendo (1,3) estas nos levam à fila número 3. O número um desta fila esta localizado nas coordenadas (3,2) levando-nos à linha 2, a mesma que nos leva novamente a 3.

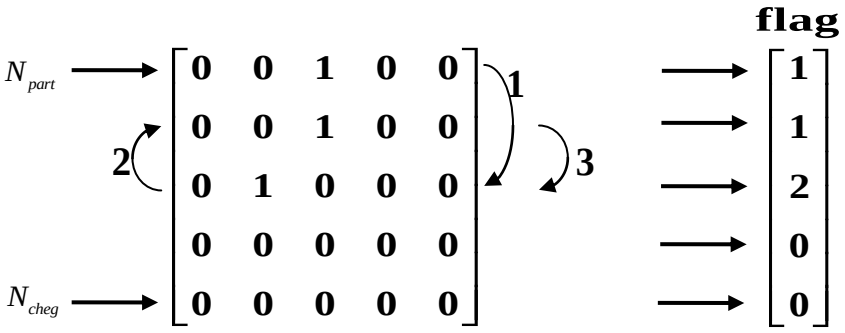



**Figura 3.14:** Representação do percurso marcado pelo primeiro indivíduo.

Na hora que alguma das componentes do vetor *Flag* passa a ser maior do que hum (1) o processo de seguimento do percurso finaliza. Penalizando-se a este individuo com um valor numérico de *fitness* muito pequeno com o objetivo que seja considerado um individuo não apto no processo de otimização do *AG*, sim ser zero por que existe o caso que o objetivo $f_1$ corresponde ao inverso da distância.

$$f_1 = 1.10^{-6} \qquad \textbf{(3.1)}$$

Este valor foi considerado depois de analisar-se o valor das grandezas que intervêm neste processo de roteamento.

**Segundo Individuo:**

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| x = | 2 | 0 | 3 | 0 |

**Figura 3.15:** Segundo vetor *x* gerado aleatoriamente.

A partir do vetor de decisão se gera a matriz *M* com ajuda da matriz de conexões, na forma que se ilustra na **Figura 3.16**.

$$C = \begin{bmatrix} 0 & c_{12} & c_{13} & 0 & 0 \\ 0 & 0 & c_{23} & c_{24} & 0 \\ 0 & c_{32} & 0 & c_{34} & c_{35} \\ 0 & 0 & 0 & 0 & c_{45} \\ c_{51} & 0 & 0 & c_{54} & 0 \end{bmatrix} \qquad M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Figura 3.16:** Matriz *M* gerada a partir do segundo indivíduo.

Procede-se agora a verificar a continuidade do caminho descrito pela matriz *M* da mesma forma como foi feito para o primeiro individuo.

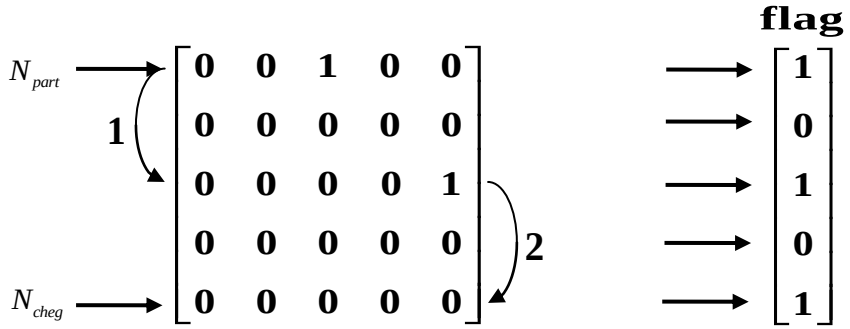



**Figura 3.17:** Representação do percurso marcado pelo segundo indivíduo.

Neste individuo o processo se deitem na linha cinco a que não tem nenhuma coordenada que contenha um 1 procedendo-se primeiramente a perguntar se o 5 corresponde ao nó de chegada. Se não for, se penaliza a função aptidão, neste caso chegou ao nó final depois se procede a verificar si todas as linhas do vetor etiqueta foram modificadas, se não for este o caso se verifica que não exista valores de hum (1) nessas linhas.

Com a certeza que a matriz *M* corresponde a um caminho contínuo desde o nó de partida ao nó de chegada. Calcula-se:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} 0 & d_{12} & d_{13} & 0 & 0 \\ 0 & 0 & d_{23} & d_{24} & 0 \\ 0 & d_{32} & 0 & d_{34} & d_{35} \\ 0 & 0 & 0 & 0 & d_{45} \\ d_{51} & 0 & 0 & d_{54} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & d_{13} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_{35} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Figura 3.18:** Cálculo do objetivo distância.

Está multiplicação não corresponde a uma operação de multiplicação de matrizes senão a uma multiplicação de coordenada da uma matriz *M* com a mesma coordenada da matriz *D* correspondente ao objetivo que esta sendo calculado e se denomina multiplicação elemento por elemento, como se apresenta na matriz da direita da **Figura 3.18.** A matriz objetivo pode ser a matriz distância, a matriz confiabilidade ou qualquer outro tipo de objetivo.

A matriz *M* da **Figura 3.18** corresponde a uma solução factível já que descreve um caminho contínuo, dita matriz serve para calcular a distância total do caminho como foi descrito através da equação **(2.4)** e que para o caso de maximizar um objetivo corresponde ao inverso do mesmo, ficando:

$$\frac{1}{d_{13}+d_{35}} = f_1 \qquad \textbf{(3.2)}$$

O valor numérico da função objetivo $f_1$ que foi calculado através da equação **(3.2)** corresponde à função aptidão parcial ou total do indivíduo 1 no caso ser de um só objetivo.

Na mesma forma como foi determinado o objetivo $f_1$ se procede a multiplicar as coordenadas da matriz *M* com as mesmas coordenadas da matriz *F* correspondente ao objetivo $f_2$ de confiabilidade de maneira semelhante à representada na **Figura 3.18**. Resultando como se apresenta na equação **(3.3)**:

$$f_{13} \cdot f_{35} = f_2 \qquad \textbf{(3.3)}$$

A soma ponderada dos objetivos $f_1$ e $f_2$, apresentados nas equações **(3.2)** e **(3.3)** respectivamente, corresponde ao total da função aptidão ou comumente denominado fitness.

$$fitness = \frac{1}{f_{1max}\ d_{13}+d_{35}} + \frac{f_{13} \cdot f_{35}}{f_{2max}} \qquad \textbf{(3.4)}$$

**Terceiro Indivíduo:**

$$\begin{matrix} & 1 & 2 & 3 & 4 \end{matrix}$$

$$x = \begin{matrix} 1 & 2 & 3 & 1 \end{matrix}$$

**Figura 3.19:** Terceiro vetor *x* gerado aleatoriamente.

A partir do vetor de decisão *x* cria-se a matriz *M* a que servira para verificar-se a trajetória corresponde a um caminho contínuo ou não.

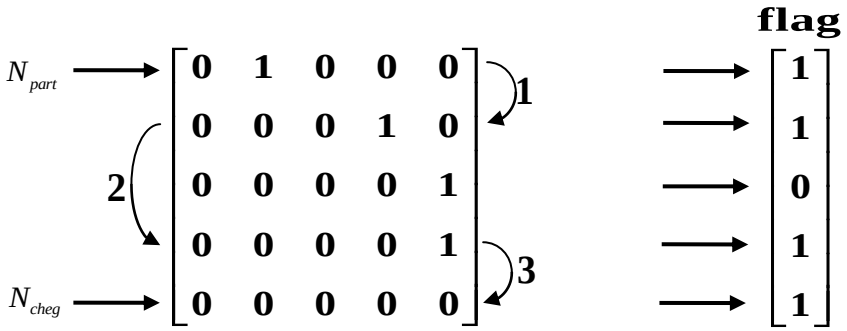



**Figura 3.20:** Representação do percurso marcado pelo terceiro indivíduo.

Primeiramente verifica-se que a fila que tem zeros seja a correspondente ao nó de chegada. Neste caso deve passar a observar se as componentes do vetor indicador (*flags*), se existem zeros, se observa a fila correspondente na matriz *M*, caso esta não ter nenhum hum se continua na revisão e caso exista algum valor de hum este dever ser trocado por zero. Já que esta conexão não esta sendo necessária percorrer para chegar ao ponto final, como se observa na **Figura 3.21**.

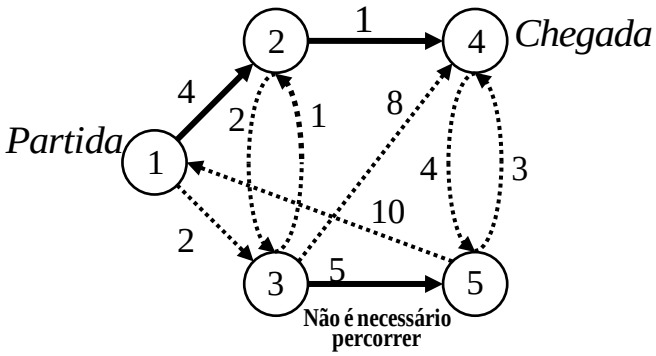



**Figura 3.21:** Representação gráfica do percurso marcado pelo terceiro vetor de decisão.

Na **Figura 3.24** se apresenta graficamente este processo de modificação da componente que não esta sendo utilizada na trajetória para chegar ao ponto final.

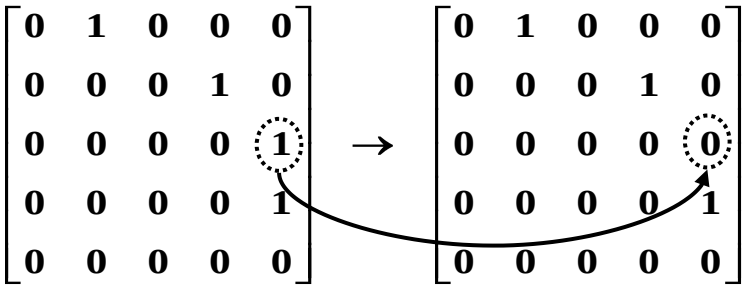



**Figura 3.22:** Representação gráfica de como se modifica a matriz *M*.

Uma vez corrigida a matriz *M*, se procede a calcular a função aptidão conforme foi descrito anteriormente para depois modificar o indivíduo quase da mesma forma como foi modificada a matriz de conexões possíveis a percorrer.

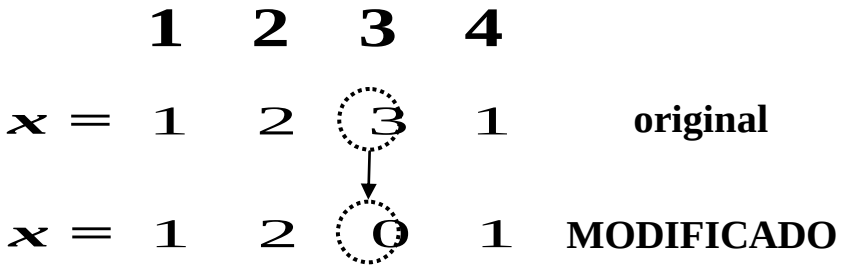



**Figura 3.23:** Representação gráfica de como se modifica o vetor *x*.

Como observamos no terceiro vetor *x*, podem criar-se na primeira população como nos processos de mutação ou cruzamento indivíduos que descrevam trajetórias com algumas conexões isoladas, o que levaria a ter um maior número de indivíduos não aptos, dificultando dessa maneira a convergência do algoritmo.

## 3.4 Função Reparação

Na procura de eliminar este problema se implementou uma função denominada *reparação*, há que é implementada por alguns pesquisadores para diversos fins. Existem trabalhos feitos onde se implementou esta função de reparo na busca do caminho mais curto como são: a referencia [5] e [33] onde os autores procedem a encontrar e eliminar os laços ou ciclos presentes em uma trajetória, modificando o cromossomo através do uso da função reparação. Esta função foi implementada para atuar depois do processo de cruzamento, podendo ser considerada como outro operador do algoritmo genético.

A função *reparo* implementada neste trabalho e apresentada através de um exemplo do processo proposto de roteamento, como se mostra na **Figura 3.22** e **3.23**, não atua como um operador do *AG*, neste caso a função *reparação* opera nas trajetórias que chegam ao ponto final e tem adicionadas outras conexões então repara os cromossomos limpando-os das trajetórias isoladas. Está *Reparação* do cromossomo está inserida na função aptidão, como se apresenta na **Figura 3.24**, onde se verifica primeiramente a continuidade da trajetória.

Sendo importante destacar, baseando-se na experiência ao inicio do trabalho de desenvolvimento desta tese, que a *não existência* da função de reparação no processo de roteamento produz lentidão no processo de convergência e em muitos casos nem consegue determinar o ótimo global ficando preso nos ótimos locais.

Cada um dos individuo é associado a um valor numérico de desempenho, obtido através da função avaliação como foi descrito anteriormente. Após a avaliação dos indivíduos, a operação de *seleção* pode ser executada. Essa operação é feita probabilisticamente, onde os indivíduos com maiores probabilidades de ser selecionados são aqueles com desempenho superior à média.

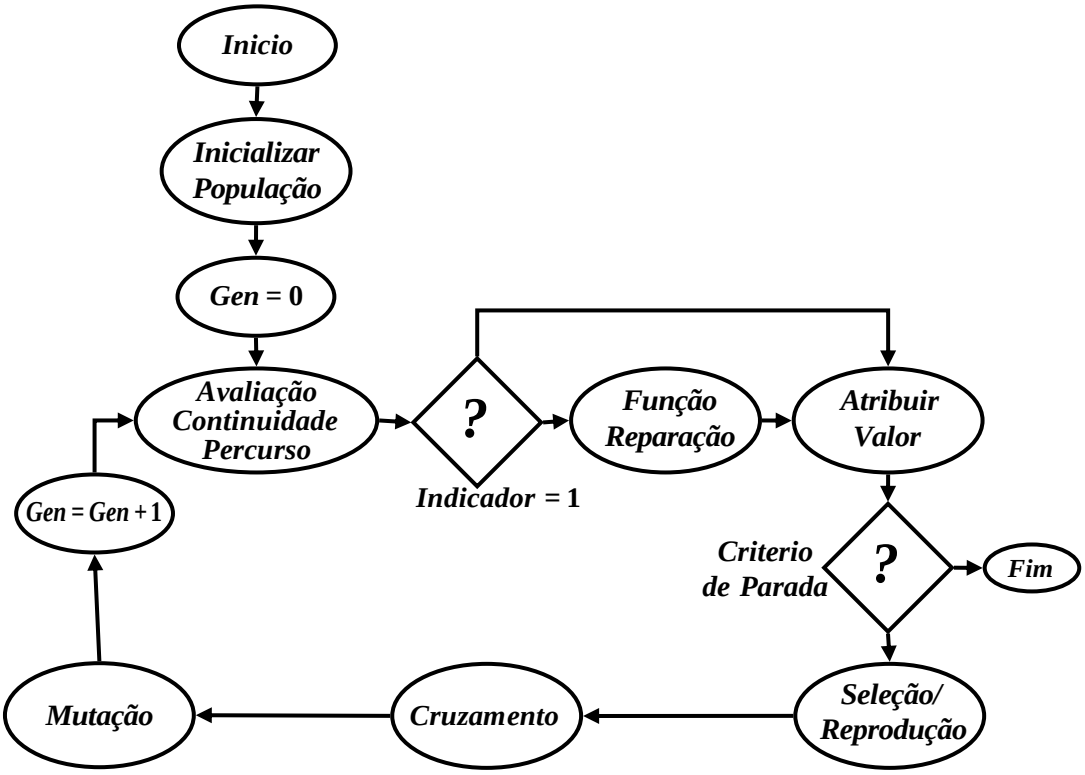



**Figura 3.24:** Diagrama da modificação dos Algoritmos Genéticos.

Após a seleção, pares de indivíduos são escolhidos aleatoriamente para se cruzarem. Nessa operação, dois novos indivíduos são originados, os quais possuem características genéticas de seus pais. O *cruzamento* constitui-se em um importante mecanismo de recombinação de soluções, ou seja, permite a permuta de material genético entre os indivíduos participantes, originando dois novos pontos no espaço de otimização.

A *mutação* é realizada em seguida. Essa operação é realizada com uma pequena probabilidade, a qual consiste na alteração randômica no valor de um gene da estrutura cromossômica. Ela exerce dois importantes papéis no processo iterativo, isto é, permite a restauração do material genético perdido nas etapas seleção, cruzamento, mutação e também na hora que são reparados dos cromossomos, e insere novas características na população.

Destaca-se que os operadores genéticos são fundamentais para a obtenção de novos pontos soluções no espaço de busca. Os novos indivíduos obtidos substituem os anteriores. O processo de obtenção da solução global para uma dada função objetivo é iterativo e se finda após o alcance de um predeterminado número de gerações. Ressalta-se que o desempenho médio dos novos indivíduos surgidos após a ação dos mecanismos genéticos é superior, como foi observado ao longo das iterações do algoritmo.

## 3.5 Espaço de Busca

Ao conjunto de todas as possíveis soluções a um problema concreto se denomina como espaço de busca [19] e se resolve o problema quando se determina a melhor solução dentre esse conjunto de soluções possíveis. No caso especifico de uma rede o conjunto solução está situado no espaço de todas as permutações do conjunto de nós da mesma.

Onde o tamanho do espaço de busca é igual a:

$$\boldsymbol{E} = \frac{nV - 1\ !}{2} \qquad \textbf{(3.5)}$$

Definir o espaço de busca não é simples já que este dimensiona através de um valor a dimensão do conjunto solução. Esta métrica que resulta ser muito importante desde o ponto de vista computacional serve para avaliar a utilização das diversas técnicas.

A solução ao problema do caminho mais curto através das técnicas determinísticas, como é o algoritmo de *Dijkstra* entre outros, consiste em encontrar a melhor solução de todo o conjunto de soluções possíveis. Isto resulta ser uma desvantagem desde o ponto de vista de processamento, porque é necessário caminhar em todas as direções conseguindo todas as combinações possíveis, ou seja, procurar em todo o espaço de busca para assim determinar a melhor solução [25]. Além disso, o aumento do número de objetivos *n* a ser otimizados produz um aumento em *n*-vezes do tempo computacional a utilizar-se.

No entanto, a técnica heurística dos algoritmos genéticos utiliza a dimensão da população e o número de gerações como os parâmetros que definem diretamente o espaço de busca a ser coberto. Devido a está característica os *AG* se tornam vantajosos, já que estes podem ser ainda mais eficientes em sistemas de grande porte. Outra característica positiva da implementação é o aproveitamento da propriedade de paralelizar intrinsecamente o calculo dos objetivos reduzindo enormemente o tempo computacional a ser gasto.

# Capítulo 4

# IV. Simulação e Resultados

## 4.1 Introdução

No presente capítulo serão apresentados e analisados os resultados experimentais obtidos da aplicação sobre vários sistemas de comunicação considerados para teste do algoritmo de *Roteamento de Redes*, proposto neste trabalho. Como já foi descrito anteriormente o algoritmo de *Roteamento* baseado nos Algoritmos Genéticos tem como objetivo a obtenção do mínimo trajeto simultaneamente procura pela que seja mais confiável.

Na seguinte seção será descrito, primeiramente, o sistema computacional utilizado para o desenvolvimento do algoritmo de *Roteamento de Redes* proposto. Será introduzido com as características do computador pessoal utilizado para o desenvolvimento e teste do aplicativo. Depois, será descrito o *cluster* de microcomputadores utilizados para obter os resultados experimentais utilizando a versão paralela do algoritmo de *Roteamento* inicialmente desenvolvido.

Imediatamente depois, os parâmetros globais do AG serão apresentados, cujos valores não mudam quando se utiliza a versão seqüencial ou paralela do algoritmo proposto.

Além será descrita a metodologia seguida para obter os resultados experimentais, com a finalidade de apresentar de forma clara a versatilidade, a simplicidade e a confiabilidade do algoritmo proposto; quer dizer, será apresentada através dos resultados experimentais a fácil adaptação para redes de grande porte, a simplicidade de implementação, pois o AG não foi variado, e também a capacidade de obter boas soluções em tempos razoáveis de processamento.

As qualidades, brevemente descritas sobre o algoritmo de *Roteamento* serão finalmente comparadas com outro algoritmo de comprovada eficácia: o algoritmo *Disjktra*, considerado como uns dos algoritmos para *Roteamento* mais rápidos existentes.

## 4.2 Descrição dos Sistemas Computacionais Utilizados

Inicialmente o algoritmo de *Roteamento*, baseado no AG, foi desenvolvido usando o *ToolBox* de Algoritmos Genéticos do *MatLab*®. A razão disso foi com a finalidade de comparar o algoritmo de *Roteamento* desenvolvido com outro normalmente utilizado e confiável, o algoritmo de *Dijkstra*. O algoritmo de *Dijkstra* já se tenha disponibilizado em formato *MatLab*®.

Como já foi descrito anteriormente, o AG simples foi adaptado para otimização Multi-Objetivo. A técnica de otimização Multi-Objetivo utilizada corresponde à soma ponderada dos objetivos. No toolbox de AG, os cromossomos são de tipo binário. Porem, para o tipo de problema de *Roteamento* o AG foi primeiramente adaptado para trabalhar com cromossomos do tipo inteiro.

Porém, visando a implementação paralela da metodologia proposta, foi re-escrito o mesmo algoritmo utilizando a Linguagem C *ANSI* (C Padrão).

As primeiras execuções da versão seqüencial, como para a comparação dos resultados obtidos pelo algoritmo de roteamento proposto e o algoritmo de *dijkstra*, foram feitas num computador com microprocessador *INTEL PENTIUM* 3, sistema operacional *Windows XP*, 256 *Megabyte* de *RAM* e 1 *GHz* de velocidade de processamento.

No caso da versão paralela, para a metodologia de busca da rota confiável e de mínima distância, aplicado sobre diversos sistemas de transporte de diferentes dimensões, foi utilizado o *cluster* de microprocessadores do Núcleo de Computação de Alto Desempenho (*NACAD*) da COPPE/UFRJ. O *cluster* está formado por 14 microprocessadores de 64 bits, sistema operativo *UNIX*, e 28 Gigabytes *RAM* (compartilhada *NUMA*)**Error! Reference source not found.**.

## 4.3 Parâmetros Globais do *Algoritmo Genético*.

Nesta seção serão descritos os parâmetros globais do AG, aqueles cujos valores numéricos não variam quando utilizados em diversos sistemas de transporte (diferentes dimensões).

A seguir, na **Tabela 4.1** são apresentados os parâmetros globais utilizados para executar o *AG*:

**Tabela 4.1:** Parâmetros do *Algoritmo Genético*

| Parâmetro do *Algoritmo Genético* | Descrição / Grandezas |
|---|---|
| Codificação do *Cromossomo* | Inteiro |
| Probabilidade de Cruzamento | 0.9 |
| Probabilidade de Mutação | 0.05 |
| Tipo de Operador de Seleção | Roleta |
| Tipo de Operador de Cruzamento | Cruzamento de Dois Pontos de Corte |
| Tipo de Operador de Mutação | Mutação Constante |

Durante a execução do algoritmo de *Roteamento* proposto foi variado o tamanho da população. Dessa forma, se observa a influencia do numero considerado de soluções factíveis com o qual o AG trabalha na procura de soluções de trajeto mínimo e confiável.

Também, no procedimento de obtenção de resultados experimentais é variado o numero de gerações (iterações) como critério de parada. Dessa forma é observada a influencia da variação do critério de parada na qualidade da solução obtida (mínimo valor da trajetória completa e maior índice de confiabilidade determinado).

Cada componente de um indivíduo (ou seja, do *cromossomo* associado) corresponde à posição de uma conexão existente na matriz de conexões. Portanto, o conjunto de variáveis que formam um indivíduo tem uma dimensão igual ao número de nós que compõem a rede menos o nó de chegada por ser o ponto final da trajetória. Como mencionado com anterioridade, o nó de chegada não entra na representação do *cromossomo* de cada indivíduo.

## 4.4 Sistemas de Transporte Utilizados para Teste

Nesta seção serão apresentados os sistemas utilizados para teste. Todos os sistemas aqui descritos provem da seguinte referencia [18][17]. Foram utilizados quatro sistemas de transporte para validar a metodologia proposta para *Roteamento* mínimo e confiável.

A dimensão de cada um desses sistemas está determinada pela quantidade de nós de comunicação (trajeto) existente no sistema. Na **Tabela 4.2** se apresenta a nomenclatura utilizada para descrever cada um dos sistemas de teste:

**Tabela 4.2:**.Nomenclatura utilizada para cada sistema.

| ***Nome*** | ***No. Nós do sistema*** |
|---|---|
| Sistema 1 | 5 |
| Sistema 2 | 19 |
| Sistema 3 | 150 |
| Sistema 4 | 323 |

O sentido do trajeto entre um nó e o seguinte é definido através da matriz de conexões. A matriz de conexões, associada a cada sistema de teste, é apresenta no Apêndice C.

Na **Tabela 4.2** os sistemas 1 até 3 estão associados a dados das matrizes de conexões pertencentes ao Problema do Vendedor Viajante (*TSP* – Travelling Salesman Problem) simétrico. Porém, o Sistema 4 corresponde ao *TSP* do tipo assimétrico. Esses sistemas de transporte servem com muita aproximação para validar o algoritmo de *Roteamento* proposto para minimização da trajetória de comunicação e confiabilidade na transmissão de dados em redes de comunicação.

A dimensão de cada sistema não só está estabelecido pelo número de nós de comunicação existentes, mas também pelo número de conexões possíveis (estabelecidos pela matriz de conexões) entre um nó e os demais pertencentes a rede correspondente sob análise. O número de ligações possíveis determina a complexidade do algoritmo, o qual representa o espaço de busca do *AG* (conforme descrito no Capítulo 2).

Além disso, foi utilizado o Sistema Argentino de Transporte para validar o desempenho do algoritmo de *Roteamento* desenvolvido com o a convergência do algoritmo de *Disjktra*. Os dados correspondentes ao Sistema Argentino de Transporte são dados no Apêndice C.

## 4.5 Análise dos Resultados Experimentais

Para análise dos resultados experimentais e validação do algoritmo de *Roteamento* proposto primeiramente foram considerados os quatro sistemas de transporte descritos anteriormente. Dessa forma, é analisado o tempo de computação, bem como a trajetória mínima e confiável encontrada, considerando os sistemas de teste.

Seguidamente, para validação do algoritmo proposto foi considerado o sistema de transporte argentino (sistema não simétrico). Sobre esse sistema foi aplicado o algoritmo de *Dijkstra* e também o algoritmo de *Roteamento* desenvolvido com confiabilidade igual a um para todas as arestas. Os resultados da trajetória ótima e o tempo de execução de ambos os algoritmos são comparados entre si. Dessa maneira, é possível validar o algoritmo proposto.

A versão seqüencial implementada em *MatLab*® inicialmente e aplicada sobre os quatro sistemas de transporte utilizado para teste bem como sobre o sistema de transporte argentino. Porém, para implementação paralela do algoritmo de Roteamento proposto, a versão inicial feita em *MatLab*® foi re-escrita completamente utilizando a Linguagem *C* (*C ANSI*).

O algoritmo de *Roteamento* proposto, baseado em AG, foi adaptado para computação paralela através da técnica denominada *Mestre* / *Escravo*; quer dizer, existe um processador denominado *Mestre* que se encarrega de administrar a execução do AG propriamente ditos, e vários outros processadores denominados *Escravos*. Cada *Escravo* tem como tarefa a avaliação (cálculo do valor numérico da função objetivo) de um determinado conjunto de indivíduos (soluções candidatas), e devolver ao *Mestre* o resultado da avaliação.

A técnica de paralelização brevemente descrita anteriormente é aplicada exclusivamente para minimizar tempos de execução e para economizar recursos computacionais (capacidade de armazenamento e operação na memória de trabalho – *RAM*).

### 4.5.1 Resultados Experimentais (versão Seqüencial)

A versão seqüencial do algoritmo de *Roteamento* proposto foi desenvolvido primeiramente em *MatLab*® e foi re-escrita completamente utilizando a Linguagem *C* (*C ANSI*). Fazendo uso deste programa de roteamento se utilizou no primeiro estagio os quatro sistemas de transporte, para a realização disto se assumiu uma matriz de confiabilidade duplamente estocástica para cada um destes sistemas.

Na **Tabela 4.3** apresenta os resultados deste primeiro estagio onde pode ser observado como cresce o tempo de execução do algoritmo quando aumenta o tamanho da população e o máximo número de gerações (critério de parada). A tabela de resultados correspondentes foi montada considerando a melhor solução de entre duas execuções realizadas para cada um dos sistemas de transporte utilizados como teste.

Para cada um dos sistemas considerados para o teste, foi utilizado um único ponto de partida e de chegada (um único caso) para obter os resultados apresentados na **Tabela 4.3**, com a finalidade de facilitar as comparações entre si.

**Tabela 4.3:** Resultados variando número de gerações e tamanho da população.

| Sistema | Geração | População | Tempo [seg.] |
|---|---|---|---|
| **1** | 5 | 40 | 0.086 |
| | 10 | 40 | 0.18 |
| | 15 | 40 | 0.26 |
| | 20 | 40 | 0.33 |
| | 5 | 60 | 0.15 |
| | 10 | 60 | 0.29 |
| | 15 | 60 | 0.38 |
| | 20 | 60 | 0.52 |
| **2** | 5 | 40 | 0.48 |
| | 10 | 40 | 0.81 |
| | 15 | 40 | 0.99 |
| | 20 | 40 | 1.23 |
| | 5 | 60 | 0.64 |
| | 10 | 60 | 1.56 |
| | 15 | 60 | 1.74 |
| | 20 | 60 | 2.65 |
| **3** | 5 | 40 | 50.97 |
| | 10 | 40 | 81.00 |
| | 15 | 40 | 106.66 |
| | 20 | 40 | 189.78 |
| | 5 | 60 | 73.80 |
| | 10 | 60 | 87.67 |
| | 15 | 60 | 125.00 |
| | 20 | 60 | 246.77 |
| **4** | 5 | 40 | 371,72 |
| | 10 | 40 | 1004,15 |
| | 15 | 40 | 1291.30 |
| | 20 | 40 | 1960.00 |
| | 5 | 60 | 639.50 |
| | 10 | 60 | 1447.50 |
| | 15 | 60 | 1500.00 |
| | 20 | 60 | 3333.33 |

Na **Tabela 4.4**, é apresentado o ponto de partida e o nó de chegada, a distância e confiabilidade correspondente à melhor trajetória encontrada pelo algoritmo de *Roteamento* proposto, bem como a iteração media no qual já foi encontrada a melhor solução.

**Tabela 4.4:** Parâmetros utilizados em cada sistema para executar o *Roteamento* proposto.

| Sistema | Distância (m) | Confiabilidade | Trajetória |
|---|---|---|---|
| **1** | 250,00 | 0,33 | 2, 3 |
| **2** | 1823,00 | 0,12 | 2, 8, 3 |
| **3** | 1118,23 | 0,013 | 2, 3 |
| **4** | 20,00 | 0,0008 | 2, 3 |

Como pode ser observado na **Tabela 4.4**, o ponto de partida corresponde ao nó 2 e o ponto de chegada ao nó 3 os que estão apresentados como nó inicial e nó final da trajetória respectivamente, e cuja seleção é mantida em cada um dos sistemas de teste utilizados para validar o algoritmo de *Roteamento* proposto.

Como pode ser observado através da **Tabela 4.4**, a metodologia de *Roteamento* proposta converge uma solução aparentemente aceitável. Sugerindo assim uma trajetórias a ser seguida em cada um dos sistemas.

No caso dos quatro sistemas de transporte utilizados para teste pode-se notar que o tempo de computação aumenta conforme cresce a dimensão do sistema de comunicação sob estudo, como mostra o seguinte gráfico:

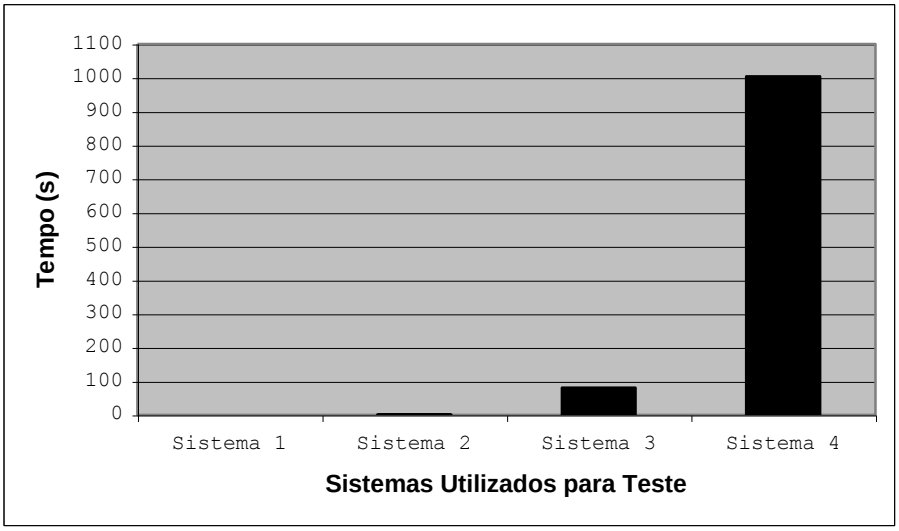



**Figura 4.1:** Tempo versus Sistemas de Comunicação de Teste.

Na **Figura 4.1** é observado que o tempo de computação cresce exponencialmente conforme aumenta o tamanho do sistema de comunicação sob estudo. Dessa maneira, os recursos computacionais requeridos (tempo de computação e tamanho da memória de trabalho) pelo algoritmo de *Roteamento* proposto para encontrar o trajeto mais confiável e de menor comprimento pode aumentar proibitivamente conforme cresce a dimensão do número de nós existentes num sistema de comunicação.

Dessa forma, a dimensão de um sistema de transporte para a versão seqüencial do algoritmo proposto está limitada principalmente pelo *hardware* do sistema computacional utilizado (basicamente pela capacidade de armazenamento da memória *RAM* ou memória temporal de trabalho). Essa dimensão também está limitada pela sofisticação utilizada no algoritmo desenvolvido (técnicas de esparsidade, operações a nível de bits – caso de utilizar a Linguagem C/C++ –, etc.).

Uma maneira de resolver tal inconveniente consiste na paralelização da metodologia de *Roteamento* proposto. Tanto a metodologia utilizada para paralelizar quanto os resultados experimentais obtidos serão descritos na seguinte seção.

### 4.5.2 Resultados Experimentais (versão Paralela)

Para implementação paralela do algoritmo de *Roteamento* proposto, primeiramente a versão seqüencial foi re-escrita completamente utilizado a Linguagem *C ANSI*. Como foi mencionado anteriormente, o tipo de implementação paralela adaptada para o algoritmo de *Roteamento* desenvolvido corresponde à estrutura *Mestre*/*Escravo*.

Para desenvolvimento da versão paralela do algoritmo de *Roteamento* utilizou-se o *MPI* (*Message Passing Interface*) [29], o qual constitui um pacote de múltiplas funções que administram a transmissão de dados entre um e vários processadores numa rede homogênea ou heterogênea de computadores.

Assim, para implementação e execução da versão paralela do algoritmo de Roteamento considerou-se o cluster de 16 microprocessadores duais da ITAUTEC do Núcleo de Computação de Alto Desempenho da COPPE/UFRJ. Cada nó (microprocessador) possui uma memória de 512 MB *RAM*, e com memória global distribuída de 8.0 Gigabytes.

Como observado na **Figura 4.1**, o algoritmo de *Roteamento* proposto demora mais em encontrar a trajetória ótima quando aplicado sobre os sistemas de transporte de 150 nós (Sistema 3) e 323 nós (Sistema 4). Então, somente sobre esses sistemas foi aplicada a versão paralela do algoritmo de *Roteamento* desenvolvida. Assim, considerou-se uma população de tamanho igual a 20 *indivíduos*, e máximo número de gerações igual a 10, como critério de parada.

No algoritmo de *Roteamento* paralelo tipo *Mestre*/*Escravo*, a transmissão de um determinado conjunto de *indivíduos* desde o *Mestre* [Anexo B] para os outros processadores (*Escravos*) é feito em cada geração. São enviados para os *Escravos* os *cromossomos* associados aos *indivíduos* correspondentes. Então, o *Escravo* avalia (cálculo da função de aptidão) cada *cromossomo* do conjunto recebido. Depois disso, são devolvidos pelos *Escravos* ao *Mestre* as avaliações associadas a cada *individuo*.

**Tabela 4.5:** Resultados para a versão paralela do algoritmo de *Roteamento*.

| Sistema | nprocs | Tempo (s) |
|---|---|---|
| 3 | 1 | 44,47339 |
| | 2 | 21,07770 |
| | 4 | 11,59390 |
| 4 | 1 | 90,73775 |
| | 2 | 75,97575 |
| | 4 | 34,60799 |

Na **Tabela 4.5**, a coluna `nprocs` refére-se ao número de processadores utilizados do *cluster*. Dessa maneira, é comparado o tempo de computação da versão seqüencial do algoritmo de *Roteamento* (re-escrito utilizando a Linguagem *C*) com as implementações paralelas do algoritmo. A última coluna corresponde ao máximo tempo de computação requerido pelo algoritmo para atingir o critério de parada e fornecer a solução ótima. A **Tabela 4.5** montou-se considerando o melhor resultado de *roteamento* fornecido pelo algoritmo depois de três execuções para cada caso.

Com a finalidade de determinar a eficiência do algoritmo de *Roteamento* proposto foi estabelecida a **Tabela 4.6**, na qual são apresentados os valores numéricos que representam o *speedup*, aceleração da convergência quando são utilizados vários processadores numa rede de computadores.

**Tabela 4.6:** *Speedup* do algoritmo de *Roteamento* proposto.

| **Sistema** | ***Speedup* (linear)** | ***Speedup*** |
|---|---|---|
| **3** | 2 | 2,110 |
| | 4 | 3,836 |
| **4** | 2 | 1,194 |
| | 4 | 2,622 |

Na **Tabela 4.6**, a segunda coluna corresponde ao *speedup* linear (teórico) comparado com o *speedup* calculado para o algoritmo de *Roteamento* paralelo desenvolvido (terceira coluna). O speedup aqui apresentado e calculado através da seguinte expressão matemática [12]:

$$speedup = \frac{tempo\ sequencial}{tempo\ paralelo} \qquad \textbf{(4.1)}$$

Na expressão matemática **(4.1)** para o calculo de *speedup* de um algoritmo paralelo, o numerador corresponde ao tempo de execução do algoritmo seqüencial que no presente trabalho refere-se à versão seqüencial do algoritmo de *Roteamento* proposto. O denominador corresponde ao tempo de computação da versão paralela do algoritmo de *Roteamento* desenvolvido.

Para o Sistema 3 pode ser observado que o *speedup* é sublinear, quer dizer que a curva de *speedup*, ou aceleração de convergência, decresce conforme aumenta o numero de processadores utilizados do *cluster* afastando-se da curva linear de *speedup* (curva teórica esperada). O *speedup* da versão paralela do algoritmo de *Roteamento* desenvolvido quando aplicado sobre o Sistema 4 apresenta também um comportamento sublinear como no caso anterior.

Porem, a versão paralela do algoritmo de *Roteamento* proposto se afasta mais da resposta linear teórica quando aplicado sobre o Sistema 3. Mas, em media o algoritmo terá o mesmo desempenho quando aplicado a sistemas de transporte de dimensão maior. Para isso deveram ser realizadas mais simulações (superiores às três execuções) sobre os Sistemas de transporte 3 e 4 utilizadas para teste.

Existem diversas topologias paralelas e metodologias de transmissão de dados entre processadores numa rede de computadores ou *cluster* que permitem aproximar o *speedup* do algoritmo para a resposta linear teórica, ou seja, fazer que o algoritmo paralelo tenha um desempenho mais linear (tempo de computação decresce de maneira inversamente proporcional ao numero de processadores da rede). Varias dessas metodologias de paralelização são apresentados no Apêndice C.

### 4.5.3 Resultados Experimentais ( Utilizando uma única função)

Com a finalidade de validar a metodologia proposta através da comparação de seu desempenho com outro algoritmo comprovadamente eficiente e amplamente utilizado para os problemas de *Roteamento*, foi considerado o algoritmo de *Dijkstra* para busca de trajetória ótima. Porém, o algoritmo de *Dijkstra* utilizado nesta pesquisa como ferramenta de validação, trabalha na procura de um mínimo trajeto entre dois pontos, não está incluída nessa procura a rota mais confiável.

Dessa maneira, utilizou-se a metodologia proposta de *Roteamento* somente para da trajetória ótima de comunicação entre dois nós. Ou seja, otimizou-se uma única função objetivo (rota mínima).

O sistema de teste utilizado para validação e comparação de desempenho entre a metodologia de *Roteamento* proposto e o algoritmo *Dijkstra* corresponde ao Sistema de Transporte Argentino, cujos dados são apresentados no Apêndice A do presente trabalho. Nesse sistema, as diferentes trajetórias otimizadas e comparadas são apresentadas e descritas na **Tabela 4.7**.

Na **Tabela 4.7** a primeira coluna, **Nó Inicial** corresponde ao ponto de partida; a segunda coluna, **Nó final**, corresponde ao ponto de chegada. Na coluna nomeada de **Dikjstra** são indicados os tempos de computação do algoritmo de *Dikjstra* para encontrar a rota mínima para cada um dessas trajetórias.

Porém, na quarta coluna da **Tabela 4.7**, indicada através do símbolo **MEVZ_1**, é descrita o tempo de computação que o algoritmo de *Roteamento* proposto leva até encontrar a mesma rota mínima determinada pelo *Dikjstra*.

**Tabela 4.7:** Resultados de Mínima Trajetória utilizando *Dikjstra* e o algoritmo de *Roteamento* proposto.

| Nó Inicial | Nó Final | Trajetória | Dikjstra | MEVZ1 |
|---|---|---|---|---|
| BUENOS AIRES | SAN. S. DE JUJUY | BUENOS AIRES - SAN. S. DE JUJUY | 0,200 | 0,2295 |
| BUENOS AIRES | SALTA | BUENOS AIRES – SALTA | 0,201 | 0,2346 |
| BUENOS AIRES | S. M. DE TUCUMAN | BUENOS AIRES-S. M. DE TUCUMAN | 0,200 | 0,2290 |
| BUENOS AIRES | SGO. DEL ESTERO | BUENOS AIRES-SGO. DEL ESTERO | 0,200 | 0,2256 |
| BUENOS AIRES | FORMOSA | BUENOS AIRES-CORRIENTES-FORMOSA | 0,190 | 0,2338 |
| BUENOS AIRES | RESISTENCIA | BUENOS AIRES-CORRIENTES-RESISTENCIA | 0,190 | 0,2743 |
| BUENOS AIRES | SANTA FE | BUENOS AIRES - SANTA FE | 0,220 | 0,2403 |
| BUENOS AIRES | CORRIENTES | BUENOS AIRES – CORRIENTES | 0,210 | 0,2949 |
| BUENOS AIRES | POSADAS | BUENOS AIRES – POSADAS | 0,210 | 0,2830 |
| BUENOS AIRES | PARANA | BUENOS AIRES – PARANA | 0,201 | 0,2134 |
| BUENOS AIRES | CORDOBA | BUENOS AIRES – CORDOBA | 0,200 | 0,2298 |
| BUENOS AIRES | LA RIOJA | BUENOS AIRES - LA RIOJA | 0,201 | 0,2406 |
| BUENOS AIRES | SAN JUAN | BUENOS AIRES - SAN JUAN | 0,200 | 0,2593 |
| BUENOS AIRES | SAN LUIS | BUENOS AIRES - SAN LUIS | 0,210 | 0,2525 |
| BUENOS AIRES | CATAMARCA | BUENOS AIRES – CATAMARCA | 0,210 | 0,2281 |
| BUENOS AIRES | MENDOZA | BUENOS AIRES – MENDOZA | 0,211 | 0,2722 |
| BUENOS AIRES | SANTA ROSA | BUENOS AIRES - SANTA ROSA | 0,200 | 0,2066 |
| BUENOS AIRES | NEUQUEN | BUENOS AIRES – SANTA ROSA – NEUQUEN | 0,201 | 0,1931 |
| BUENOS AIRES | VIEDMA | BUENOS AIRES – VIEDMA | 0,201 | 0,2665 |
| BUENOS AIRES | RAWSON | BUENOS AIRES – RAWSON | 0,201 | 0,2972 |
| BUENOS AIRES | RIO GALLEGOS | BUENOS AIRES - RIO GALLEGOS | 0,201 | 0,2528 |
| BUENOS AIRES | USHUAIA | BUENOS AIRES – USHUAIA | 0,200 | 0,2324 |

# Capítulo 5

# V. Conclusões e Trabalhos Futuros

## 5.1 Conclusões

As metodologias de roteamento ótimo baseadas em técnicas determinísticas (métodos numéricos) requerem de altos recursos computacionais (memória de trabalho – RAM, elevados tempos computacionais, entre outros) para atingir boas soluções. Num meio corporativo, onde a velocidade de transmissão de dados é relevante, a escolha de um trajeto confiável, onde diversas tecnologias de comunicação estejam sendo utilizadas, minimização de custos na transmissão de dados, entre outros objetivos conflitantes entre si pode tornar o problema de roteamento ótimo não apto para utilização de métodos determinísticos.

No presente trabalho, foi desenvolvida uma metodologia de *Roteamento* ótimo baseado nos AG com a finalidade de minimizar a distância de comunicação entre dois nós de uma rede de transmissão de dados e ao mesmo tempo encontrar a rota mais confiável entre esses dois nós. Ou seja, a finalidade no desenvolvimento de um determinado algoritmo de *Roteamento* e para otimizar simultaneamente dois objetivos conflitivos entre si.

A metodologia de *Roteamento* proposta encontra soluções de trajeto ótimo utilizando um reduzido número de gerações (critério de parada) para convergir. Além disso, não precisa de muitos indivíduos na sua população para encontrar as soluções ótimas. Tudo isso contribui a uma rápida convergência para boas soluções.

Os resultados de trajeto ótimo fornecido pelo algoritmo de *Roteamento* proposto aplicado a um sistema de transporte argentino (caracterizado por uma matriz de distancias não simétrica) são comparados com os resultados obtidos através da utilização do algoritmo de *Dikjstra*. O algoritmo de *Roteamento* proposto encontra o mesmo trajeto que o *Dikjstra* determina.

Porém, o algoritmo de *Dikjstra* tem uma convergência mais rápida do que o algoritmo de *Roteamento* desenvolvido. A diferença de tempo de execução entre o algoritmo de *Roteamento* proposto e o algoritmo de *Dikjstra* não é muito grande. Os tempos de computação estão na mesma ordem. Mas, e observado que o tempo de computação do algoritmo de *Roteamento* implementado depende diretamente do tamanho selecionado para a população. Para uma população de tamanho menor o tempo de computação decresce de tal forma que o algoritmo de *Roteamento* proposto resulta mais rápido do que o algoritmo de *Dikjstra*, entretando com resultados menos precisos.

Portanto a metodologia de *Roteamento* proposto poderia resultar numa ferramenta útil ser desenvolvida em problemas de *Roteamento* e projeto de Redes de comunicação, nas quais o tempo de resposta para a obtenção de uma solução subótima é mas importante do que a obtenção do ótimo.

Foi observado também que o tempo de computação do algoritmo cresce exponencialmente conforme aumenta a dimensão do sistema de transmissão (numero de nós existentes na rede de comunicação). Como alternativa de lidar com este inconveniente desenvolviu-se a versão paralela do algoritmo de *Roteamento* proposto. A versão paralela do algoritmo é de tipo *Mestre* / *Escravo*. Foi implementado este tipo de algoritmo paralelo por a sua simplicidade. Porém, foi observado que esta implementação paralela é sublinear afastando-se do comportamento linear esperado (curva teórica). Portanto, para aumentar o *speedup* do algoritmo é necessário desenvolver outras técnicas de paralelização mais eficientes.

Embora o algoritmo de roteamento ótimo proposto seja comparativamente rápido na sua busca da rota mínima e mais confiável, conforme é demonstrado através das simulações e gráficos apresentados anteriormente, ele está sujeito às limitações próprias do *hardware*.

Essas limitações estão dadas principalmente pela capacidade de armazenamento de dados na memória de trabalho (*RAM*); de tal forma que para sistemas de até 1000 nós (cujo espaço de busca pode atingir até milhares de possibilidades de possíveis rotas a serem analisadas) está restrito à quantidade dos *cromossomos* que podem ser representados na população, para execução do *AG*.

## 5.2 Trabalhos Futuros

Apesar dos bons resultados obtidos no desenvolvimento deste trabalho, existem alguns aspectos já mencionados que podem ser desenvolvidos no futuro. Esses aspectos a serem implementados futuramente são descritos a seguir:

a) Para melhorar ainda mais o desempenho do algoritmo (velocidade de convergência) pretende-se utilizar técnicas de esparsidade para representar as matrizes de distância e aos dados de confiabilidade;

b) Com a finalidade de aproximar a versão paralela do algoritmo de Roteamento proposto para uma curva de speedup linear é necessário considerar outros tipos de técnicas de paralelização e outras topologias de comunicação de dados entre os processadores disponíveis num determinado *cluster* de computadores;

c) Desenvolver também técnicas de paralelização de granularidade fina, ou seja, manter em cada processador um grupo de indivíduos (inclusive um único individuo), com a finalidade de otimizar os recursos de armazenamento da memória temporal de trabalho (memória *RAM*), e ao mesmo tempo acelerar grandemente a convergência à solução ótima.

d) Utilizar outras formas de paralelização do AG, de tal forma a procurar não só reduzir o tempo de computação senão também melhorar os resultados finais; ou seja, encontrar trajetos de menor distância e mais confiáveis àqueles obtidos nas versões seqüenciais.

e) Aplicar a metodologia de roteamento ótimo proposto em sistemas de maior porte, mais complexos e mais realistas.

# Referências Bibliográficas

**[1]** A.L.Bergamo "*Ajuste Coordenado de Estabilizadores de Sistemas de Potencia usando Algoritmos Genéticos*". Tese de Doutorado,COPPE/UFRJ, Rio de Janeiro-Brasil,Março 2000

**[2]** Ahuja R. K., Magnanti T. L. e Orlin J. B., *Network Flows – Theory Algoritms and Applications*, Prentice – Hall, 1993.

**[3]** Bastos, Eduardo, "*Introdução a Grafos*". Universidade Católica de Pelotas, 2003.

**[4]** Bruno da Rocha Braga, "*A evolução dos Algoritmos de Primalidade e seu impacto na Criptografia*", Módulo Security - Portal de Segurança da Informação. Brasil, 2003.

**[5]** Chang Wook Ahn, Ramakrishna R. S., *A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations*, IEEE Transactions Evolutionary Computation, Vol. 6, No. 6, 2002.

**[6]** Claudia Dias, *Segurança e Auditoria da Tecnologia da informação*, Axcel Books do Brasil Editora.

**[7]** D.E.Goldberg "*Genetic Algorithms in Search, Optimization and Machine learning"*.Addison-Wesley,MA,1989

**[8]** Deb Kalyanmoy, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Ltd, 2001.

**[9]** Deeter D., Smith A., *Economic Design of Reliable Networks*, IIE Transactions, Special Issue on Economics of Reliability Engineering, 1998.

**[10]** Dengiz B., Altiparmak F., Smith A., *Local Search Genetic Algorithm for Optimal Design of Reliable Networks*, IEEE Transactions on Evolutionary Computation, vol. 46, 1997.

**[11]** Duarte S., Barán B, *Algoritmos Evolutivos para el Diseño de Redes Económicas y Confiables*, Conferencia de Informática y Tecnología Aplicada (JIT-CITA), 2001

**[12]** E.Cantu-Paz and D.E.Goldberg, "*P*redicting Speedups of Idealized Bounding cases of Parallel Genetic Algorithms", Proceedings of the Seventh International Conference on Genetic Algorithms. San Francisco-United Status,1997

**[13]** E.Cantú-Paz, "*A summary of Research on Parallel Genetic Algorithms*". Illinois Genetics Algorithms Laboratory-IlliGal Report No. 95007, July 1995

**[14]** E.Cantu-Paz, "*Topologies, Migration Rates, and Multi-population Parallel Genetic Algorithms*" Illinois Genetic Algorithms Laboratory-IlliGAL Report No.99007 University of Illinois Urbana-Champaign, United Status. January, 1999.

**[15]** Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison – Wesley, Reading, MA, 1989.

**[16]** Hopcroft, J. E., and Ullman, J. D., "*Set merging algorithms*", SIAM Journal Computation, Vol. 2, pp. 296 – 303, 1973.

**[17]** http://elib.zib.de/pub/mp-testdata/tsp/tslib/atsp/index.html Universidad de Heidelberg.

**[18]** http://iwr.uni-heidelberg.de/gropups/comopt/software/TSPLIB95/ Universidad de Heidelberg.

**[19]** http://lear.inforg.uniovi.es/ia/Genetico_TSP/Espacio_busqueda.htm. Universidad de Oviedo, Asturias.

**[20]** http://www.vialidad.gov.ar/mapas_provinciales/pagina_mapas_provinciales.htm Dirección Nacional de Vialidad.

**[21]** Inagaki Jun, Haseyama Miki, Kitajima Hideo, *A Genetic for Determining Multiple Routes and its Applications*, Circuits and Systems IEEE, 1999.

**[22]** J.Tanomaru, "*Motivação, Fundamentos e Aplicações de Algoritmos Genéticos*". II Congresso Brasileiro de Redes Neurais, III Escola de Redes Neurais. Curitiba-Brasil, 1995.

**[23]** Kruse, Robert L, *Data Structures and program Design*, Second Edition, Prentice Hall.

**[24]** Laufer Fabian, Baran Benjamin, *Diseño Económico de Redes Confiables empleando A-Teams*, XXVI Conferencia Latinoamericana de CLEI´00,2000.

**[25]** Magnago, Hetor E., *Algoritmos Evolutivos Aplicados a Problemas de Diseño de Redes Confiables*, M.Sc., Universidad Nacional de la Plata, 2006.

**[26]** Miettinen, K., *Nonlinear Multiobjetive Optimization*. Boston: Kluwer, 1999.

**[27]** Nesmachnow Sergio, *Evaluating Metaheristic for the Generalized Steiner Problem*, Revista JCS&T, Vol. 4, No. 4, December 2005.

**[28]** P. Souza, J. R. Favilla, "*Asynchronous Teams for Steel Industry: Mini Mills*" International Conference on Information System Analysis and Synthesis (ISAS'96).Orlando-Estados Unidos, 1996.

**[29]** Peter S. Pacheco, *Parallel Programming with MPI,* Second Edition, Morgan Kaufman Publishers Inc.

**[30]** Paul L. Meyer, Probabilidade – Aplicações à Estatística, Segunda Edição, Editora LTC, Rio de Janeiro, 1983.

**[31]** Stanley I. Grossman, *Álgebra Lineal*, Grupo Editorial Iberoamerica.

**[32]** Tanenbaum S. Andrew, *Redes de Computadores,* Terceira edição, Editora Campus Ltda., 1997.

**[33]** Wu Wei, Ruan Qhuqi, *A Gene Constrained Genetic Algorithm for Solving Shortest Path Problem,* ICSP`04 Proceedings, 2004.

# Anexo A

# Algoritmos Genéticos

## A.1 Introdução

Nas últimas décadas surgiu uma nova ciência computacional inspirada na natureza formando uma nova área de pesquisa que se denominou *Computação Natural*. Isto é, na natureza existem dois possíveis candidatos, altamente robustos na solução de uma ampla gama de problemas tecnológicos, os quais são considerados como "muito potentes solucionadores de problemas". Esses algoritmos podem imitar os processos de aprendizado e memória do cérebro humano (*Computação Neural*); e, têm outros que imitam a capacidade de otimização dos processos evolutivos das espécies (*Computação Evolutiva*). Dentre esses algoritmos que imitam os processos naturais, os tipos de algoritmos a serem utilizados no presente trabalho referem-se aos algoritmos evolutivos de otimização; e, entre os algoritmos evolutivos os mais estudados e utilizados com sucesso são os AG.

As técnicas de busca e otimização, empregando *AG*, está baseada na idéia de *Darwin* da *Seleção Natural* das espécies, onde os indivíduos mais aptos possuem maior probabilidade de ter os seus genes propagados ao longo de sucessivas gerações, através da combinação entre os genes dos indivíduos (processo de *cruzamento*) que perduram na espécie. Os *AG* foram desenvolvidos por Holland [15] na década dos 60, e foram criadas para resolver problemas difíceis de otimização.

As primeiras pesquisas tiveram dois objetivos: (1) resumir e explicar rigorosamente os processos adaptativos dos sistemas naturais, e (2) desenvolver programas computacionais que imitem os mecanismos evolutivos dos sistemas naturais. Na atualidade, os *AG* são aplicados em problemas tecnológicos de diferentes áreas na ciência e são utilizados para a busca de ótimos globais de funções matemáticas que modelam sistemas físicos complexos.

Nos *AG*, cada um dos indivíduos da população representa uma possível solução para um problema dado, sendo que através de uma função de aptidão (*Fitness*), cada um dos indivíduos da população é avaliado, e uma pontuação (valor numérico) lhe é atribuída. Aos mais adaptados são dadas as maiores oportunidades de se reproduzir na seguinte geração, graças a uma função que simula a seleção natural (Operador de *Seleção*), e os indivíduos reproduzidos são cruzados entre eles mesmos mediante cruzamentos (Operador de *Cruzamento*), produzindo descendentes com características de ambas as partes.

Na natureza, além dos processos de cruzamento, existem também alterações genéticas que marcam uma diferença nas próximas gerações de uma mesma espécie. Essas mudanças na estrutura genética acontecem segundo uma probabilidade muito pequena (Operador de *Mutação*).

## A.2 Características dos Algoritmos Genéticos

Serão descritas a seguir as características que tornam os AG atrativos para a sua implementação em problemas tecnológicos de grande porte [15]:

1) Os AG trabalham com a codificação das variáveis independentes da função matemática que será otimizada. As variáveis independentes da função podem ser codificadas usando um sistema binário (usar um vetor de 1 e 0 para representar o valor numérico de uma variável da função), números reais e inclusive símbolos alfanuméricos. Na literatura, a representação codificada das variáveis independentes é denominada *cromossomo*.

2) Os AG começam a busca a partir de uma população de possíveis soluções (pontos do espaço de busca), e não a partir de um só ponto tendo um paralelismo implícito [7].

3) O *AG* precisa somente do valor numérico da função objetivo para guiar a busca, ao contrário dos métodos matemáticos tradicionais de otimização que precisam de informações adicionais para atingir o ponto ótimo.

4) Os *AG* utilizam regras de transição probabilísticas (operadores probabilísticos de busca), e não determinísticos.

As qualidades descritas acima tornam ao *AG* num método de busca robusto que tem bom desempenho para um número considerável de problemas.

## A.3 Implementação Básica do Algoritmo Genético

Um *AG* pode ser definido basicamente como a aplicação sucessiva de operações sobre um conjunto de indivíduos (população). Essas operações são descritas correspondem à *Seleção*, ao *Cruzamento* e à *Mutação*.

Na maioria das aplicações do *AG*, a população inicial é gerada aleatoriamente, aplicando-se os três operadores probabilísticos citados acima sobre a população de indivíduos em cada geração. Para atingir uma boa solução é preciso que a população tenha suficiente variedade genética para evitar a estagnação do *AG* em máximos locais. Esse critério está baseado na *Teoria da Seleção Natural* [15].

Os *AG* são algoritmos iterativos, e em cada iteração (geração) a população é modificada. Cada indivíduo da população atual será um "descendente" da população anterior. O procedimento de busca dos *AG* é descrito na **Figura A.1**, onde cada geração é associada à letra *t*.

```
t ← 0;
Iniciar População ( t );
Avaliar População ( t );
FAZER ENQUANTO um critério de parada não for alcançado
        t ← t + 1;
        Selecionar População ( t ) da População ( t – 1 );
        Cruzar e mutar População ( t );
        Avaliar População ( t );
FIM FAZER
```

**Figura A.1**: Pseudocódigo do Algoritmo Genético.

Descrições mais detalhada dos operadores probabilísticos serão apresentados nas subseções a seguir. O primeiro operador probabilístico a ser descrito corresponde ao Operador de *Seleção*, continuando a descrição com o Operador de *Cruzamento* e finalmente o Operador de *Mutação*. No *AG*, na obtenção de novos *indivíduos*, os operadores são aplicados na ordem apresentada anteriormente produzindo uma metodologia de busca robusta e guiada de ótimos globais.

### A.3.1 Operador de Seleção

Simula o processo de seleção da natureza, no qual o mais forte tem maior capacidade de sobrevivência. No *AG* a capacidade de sobrevivência de um *indivíduo* está associada ao valor numérico da função objetivo a maximizar. O operador de *Seleção* é aplicado sobre toda a população, em cada geração, *N* vezes com a finalidade de escolher *N indivíduos*, sendo *N* o tamanho da população.

O índice de aptidão de um *indivíduo* é medido através do seu *Fitness*, o qual está associado com o valor numérico da função matemática a ser otimizada. Geralmente, o valor numérico da função objetivo é igual ao *Fitness* do *indivíduo* correspondente. Nesse contexto, os *indivíduos* com altos *Fitness* têm mais probabilidade de ser escolhidos para a geração seguinte.

Três tipos de operadores de *Seleção* são os mais utilizados na literatura: *Seleção Estocástica*, *Seleção Determinística* e *Seleção Mista*. O operador de *Seleção* escolhe os *indivíduos* tendo em conta o *Fitness* de cada um deles; portanto, o desenvolvimento deste operador é independe do tipo de codificação das variáveis do problema.

#### A.3.1.1 Seleção de tipo estocástica.

Uma característica desse método é a determinação do número de cópias que cada *indivíduo* irá receber, baseado no valor do seu *Fitness*. Esse método criado por Holland [7] é denominado de *Seleção Proporcional* ou da *Roleta*. O método da seleção pode ser mais bem descrito através da **Figura A.2**.

Suma = 0

SumaParcial = RAND $\times \sum_{i=1}^{N} f_i$ ; /*RAND é multiplicado por Sumatoria de *Fitness* */

**FAZER** *i* = 1 **ATÉ** *i* = *N*

Suma = Suma + $f_i$ ;

**SE** Suma $\geq$ SumaParcial **ENTÃO**

Escolher *i* anterior como índice do *individuo* selecionado;

Sair de o ciclo FAZER

**FIM SE**

**FIM FAZER**

**Figura A.2**: Pseudocódigo do Operador de Seleção tipo estocástico ou da Roleta.

Na **Figura A.2**, a variável RAND tem um valor real entre 1.0 e 0.0, fornecido por algum gerador de números pseudoaleatórios. A expressão $f_i$ (onde $i$ = 1, 2, ..., $N$) representa o *i*-ésimo valor do *Fitness*, onde $N$ determina o número de indivíduos na população. Dessa maneira, a probabilidade de que um *indivíduo* seja escolhido muitas vezes na seguinte geração depende de seu valor de *Fitness*.

### A.3.1.2 Seleção de tipo determinístico.

A *Seleção Elitista* (ou *Elitismo*) faz parte desse tipo de operador de *Seleção* [22]. Este método seleciona o melhor *indivíduo* da população anterior e o preserva na população seguinte, substituindo o pior *indivíduo*. Essa estratégia evita a perda de soluções boas, propagando-os para a população seguinte.

### A.3.1.3 Seleção de tipo misto.

Este método contém, em forma simultânea, as características estocásticas e as determinísticas. Um exemplo é a *Seleção* tipo *Torneio Estocástico* [22], onde uma escolha pseudoaleatória de um conjunto de *indivíduos* é feita, e o melhor deles é selecionado. A quantidade de *indivíduos* nesse conjunto é denominada de "*tamanho do torneio*". O tamanho do torneio influencia na pressão da seleção. Quanto maior o número de *indivíduos* participantes do torneio maior será a quantidade de *indivíduos* com *Fitness* alto para a seguinte geração (pressão de seleção alta e quase determinística). Porém, isto levará à formação de uma população com *indivíduos* de *Fitness* similares e, portanto, haverá mais possibilidades de convergência para ótimos locais.

## A.3.2 Operadores Genéticos.

Após da aplicação do operador de *Seleção*, os *indivíduos* selecionados são reproduzidos para a próxima geração. Então, pares de progenitores são escolhidos de entre esses *indivíduos* selecionados, e novos descendentes são criados a partir do intercâmbio do material genético entre eles. Os descendentes podem ser diferentes de seus "pais", mas com caraterísticas genéticas de ambos os progenitores.

A estrutura dos operadores de *Cruzamento* e *Mutação*, que constituem os operadores genéticos do *AG* depende, principalmente, da representação escolhida, binária, inteira, ou real, das variáveis independentes. Neste trabalho, serão descritos os diferentes tipos de operadores de *Cruzamento* para codificação inteira devido a que o algoritmo de roteamento ótimo implementado está baseado nos *AG* que utilizam *cromossomos* inteiros.

Porém, a maioria dos algoritmos de cruzamento implementado em *AG* tem a mesma estrutura, a qual independe do tipo de codificação utilizada; só para problemas específicos é que eles diferem segundo o tipo de codificação utilizado para os parâmetros que definem o problema sob estudo.

### A.3.3 Operadores Genéticos que trabalham com *cromossomos* inteiros.

Quando o sistema físico em estudo é muito grande e complexo, geralmente a função matemática que o modela é multivariável e multimodal. E quando o sistema em estudo é de grande porte a quantidade de variáveis que definem o modelo matemático, que representa o problema sob análise, é proibitivamente elevada como para utilizar uma representação binária tradicional, por causa do grande recurso computacional que será exigido.

Portanto, surgem alternativas que minimizam o esforço computacional e de alguma maneira também, facilita a representação matemática dos objetivos a serem otimizados. Assim sendo, é mais conveniente à utilização de operadores genéticos projetados para trabalhar com *cromossomos* que não sejam os binários.

Como o algoritmo de roteamento ótimo, baseado em *AG*, implementado no presente trabalho utiliza *cromossomos* inteiros na subseção seguinte serão descritos os diferentes tipos de operadores de cruzamento utilizados na codificação inteira.

#### A.3.3.1 Operadores de Cruzamento.

O efeito do operador de *Cruzamento*, que trabalha sobre os valores inteiros dos parâmetros que formam o *cromossomo* se dá apenas no ponto de corte. Para que haja um efeito mais amplo e semelhante aos da representação binária, utilizam-se operadores de permutação.

**a. Operador de Cruzamento Tradicional.**

Os operadores de Cruzamento tradicionais são aqueles mesmos utilizados também no cruzamento de pares de *cromossomos* binários; sendo os mais conhecidos: o *Cruzamento de um Ponto de Corte*, o *Cruzamento de dois Pontos de Corte*, e o *Cruzamento Uniforme*.

Como acontece na codificação binária, dentre o conjunto de *N cromossomos* inteiros escolhidos através do operador de *Seleção* são formados aleatoriamente pares de cromossomos para serem cruzados.

Então, é determinado aleatoriamente um número entre 2 e $\ell$ (sendo $\ell$ a dimensão do cromossomo). Esse número será corresponde ao ponto de corte ($P_{\text{corte}}$). Dessa maneira, a partir do $P_{\text{corte}}$ se realiza um intercâmbio entre as componentes dos dois *cromossomos* progenitores, conforme é mostrado na **Figura A.3**:

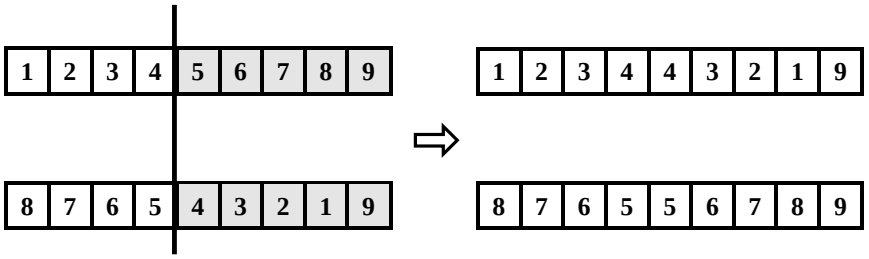



**Figura A.3:** Cruzamento de Um Ponto de Corte.

No caso do *Cruzamento de Dois Pontos de Corte*, são determinados dois pontos de corte de forma aleatória entre 2 e $\ell$ ($P_{\text{corte 1}}$ e $P_{\text{corte 2}}$). Assim, o cruzamento entre dois *cromossomos* é feito entre esses dois pontos de corte, como é descrito a seguir:

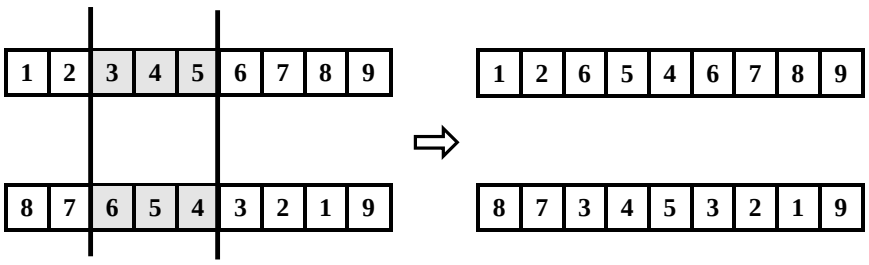



**Figura A.4:** Cruzamento de Dois Ponto de Corte.

O *Cruzamento Uniforme* monta, primeiramente, um arranjo (denominado de *máscara*) de 1s e 0s da mesma dimensão dos *cromossomos* selecionados.

Assim, como no caso da codificação binária, os genes de ambos os progenitores que estão associados aos 1s da *máscara* são cruzados, conforme mostrado a seguir:

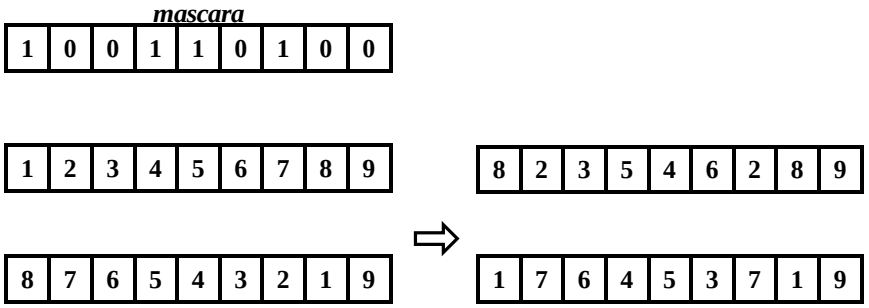



**Figura A.5:** Cruzamento Uniforme.

Os tipos de operadores de *Cruzamento* utilizados em *cromossomos* binários são utilizados da mesma forma para o caso inteiro, produzindo também as mesmas eficientes buscas locais (obtenção de novos pontos na vizinhança dos pontos da iteração anterior).

**b. Operador de Cruzamento baseado na Permutação.**

Neste caso, os *cromossomos* são construídos segundo um arranjo na qual um evento (um nó) acontece antes do seguinte (seguintes nós formando um trajeto). Isto é, neste tipo de representação ou codificação das variáveis do problema, existe uma ordenação entre os parâmetros que definem o problema.

Os problemas nos quais cada parâmetro segue uma ordenação pré-estabelecida chamam-se de *Problemas de Permutação* ou *Problemas Combinatórias*, como por exemplo, o Problema do Vendedor Viajante (*Traveling Salesman Problem*). O problema de confiabilidade de redes, que o presente trabalho está abordando, corresponde a um *Problema Combinatorial* que também pode ser posto na forma de um problema de otimização.

Nesse tipo de problemas de otimização, um determinado trajeto entre pontos (cidades ou nós, no caso de redes de comunicação) determina o modelo matemático que representa o objetivo a ser minimizado. Dessa forma, um trajeto específico pode ser o *cromossomo* que define a um determinado *indivíduo* no *AG*, sendo o número de cidades, ou nós, menos um a dimensão desse *cromossomo*.

Entre os operadores de cruzamento especializados nos *cromossomos* inteiros mais utilizados na literatura está o *Cruzamento de Primeira Ordem de Permutação*, ou simplesmente, *Operador de Permutação*.

A idéia no *Operador de Permutação* é preservar a relativa ordenação dos índices dos nós que formam um trajeto completo, num problema de roteamento ótimo, por exemplo. Então, um algoritmo básico que descreve como este operador funciona é apresentada a seguir:

1. Selecionar aleatoriamente um conjunto de nós no primeiro *cromossomo* progenitor. Esse conjunto é denominado como *conjunto de corte*. O número de nós desse conjunto será constante ao longo da execução do *AG*;
2. Copiar os nós pertencentes ao *conjunto de corte* do primeiro *cromossomo* progenitor aos mesmos lugares relativos no primeiro *cromossomo* descendente;
3. Copiar depois os nós que não estão no conjunto de corte do primeiro *cromossomo* progenitor, começando a preencher todo o primeiro *cromossomo* descendente. A cópia se faz a partir do lado direito do *conjunto de corte*, com os nós pertencentes ao segundo *cromossomo* progenitor;
4. Similar procedimento se segue para criar o segundo *cromossomo* descendente, só que agora os nós de intercâmbio provem do primeiro *cromossomo* progenitor.

O processo de cruzamento descrito acima pode ser compreendido melhor através da descrição da **Figura A.6**:

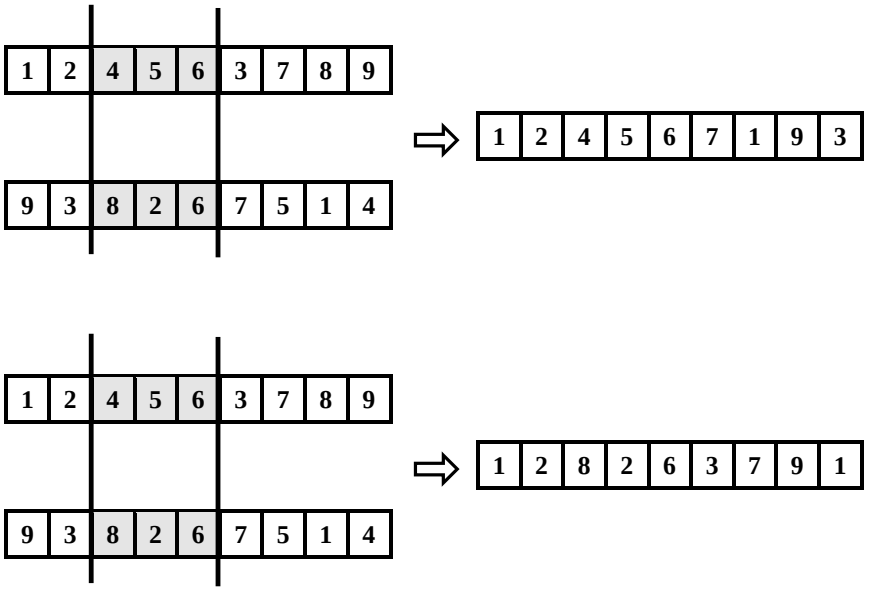


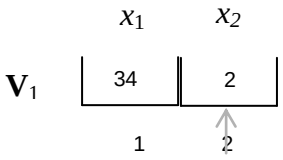

**Figura A.6:** Operador de Permutação.

No presente trabalho, porém, foram obtidos bons resultados utilizando os operadores de cruzamento tradicionais, adaptados para que trabalhem com codificação inteira.

A razão pela qual foi decidido usar os operadores tradicionais é que para a utilização do cruzamento baseado na permutação inteira é preciso de uma regra previa de construção de cada componente no *cromossomo* inteiro, de tal forma que cada nó esteja ordenado segundo um trajeto de comunicação. Mas, o algoritmo de manipulação desses dados está baseado na *Teoria de Grafos* postos na forma matricial facilitando o uso dos operadores tradicionais.

### A.3.3.2 Operadores de Mutação.

Na mutação a ser aplicada sobre um *cromossomo* inteiro, uma variável é escolhida para mutação. Uma das formas de fazer a mutação é escolher um valor aleatório entre os limites pré-estabelecidas da variável correspondente. Por exemplo, supondo um vetor de variáveis independentes como mostrado a seguir:



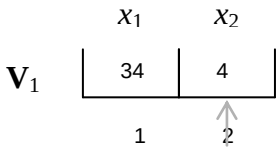

**(A.1)**

e os limites entres os quais a variável é definida sejam:

$$x_1 \in (20, 50) \;\; | \qquad x_1 \in \text{II} \qquad \textbf{(A.2)}$$

$$x_2 \in (-1, 10) \;\; | \qquad x_2 \in \text{II}$$

Considerando, ainda, que a posição no *cromossomo* da variável que foi selecionada para mutação corresponde à componente 2, então, um número inteiro que substituirá essa variável será obtido aleatoriamente entre os limites de $x_2$. Como por exemplo:




**(A.3)**

Esse tipo de mutação é denominado de *Mutação Uniforme*. Essa denominação é devido a que qualquer número pertencente ao intervalo de $x_2$ tem a mesma probabilidade de ser selecionado para substituir ao valor anterior. Mas, com este tipo de mutação a busca é muito mais aleatória e global, porque pode ser escolhido qualquer número do domínio de definição da função.

Uma outra forma de mutação especializada também pode ser utilizada nos *cromossomos* inteiros, construídos segundo uma ordenação ou permutação como já fora descrito para os *Problemas Combinatórias*. Esse tipo de mutação denomina-se *Mutação baseada na Permutação*, a qual é descrita através do seguinte algoritmo:

1. Selecionar aleatoriamente dois nós pertencentes a um *cromossomo*. O *cromossomo* também será escolhido aleatoriamente;
2. Mover o segundo nó selecionado ao lado do primeiro escolhido. Depois, o organizar o resto dos nós ao longo do *cromossomo*.


**Figura A.7:** Mutação baseado na Permutação.

Porém, também foi utilizado no algoritmo de roteamento ótimo proposto a *Mutação Uniforme* com uma probabilidade muito baixa.

## A.4 Problemas de Otimização Multi-Objetivo

Em problemas de otimização *Multi-Objetivo*, também conhecidos como *MOP* (*Multi-Objective Optimization Problems*), há mais do que um único objetivo a otimizar. Sendo assim, para cada um desses objetivos, haverá uma solução ótima diferente, podendo até mesmo ser conflitantes entre si. No entanto, em problemas de otimização mono-objetivo existe uma única solução ótima ($x^*$), conforme os problemas de otimização mostrados abaixo:

a) Problema de Otimização Mono-Objetivo:

$$Otimizar\ f(x) \tag{A.4}$$

Solução ótima: $x^*$ tal que $u = f(x^*)$
onde:

$x \in \Re^n$ é o vetor de variáveis do problema;
$f : \Re^n \Rightarrow \Re$ é a função objetivo;
$u \in \Re^n$ é o resultado obtido da função objetivo no ponto ótimo ($x^*$).

Porém, nos problemas de otimização multi-objetivo existem para uma solução factível, $x \in \Re^n$, vários valores numéricos dos objetivos a serem otimizados simultaneamente, conforme mostrado a seguir:

b) Problema de Otimização Multi-Objetivo:

$$\begin{aligned} &otimizar\ f_1(x) \\ &otimizar\ f_2(x) \\ &\quad\vdots \\ &otimizar\ f_p(x) \end{aligned} \tag{A.5}$$

Solução ótima: $x^*$ tal que $u = \{f_1(x^*), f_2(x^*), ..., f_p(x^*)\}$
onde:

$x \in \Re^n$ é o vetor de variáveis do problema;
$f_i : \Re^n \Rightarrow \Re$, $\forall\ i \in \{1, 2, ..., p\}$, é uma função objetivo;

$u \in \Re^p$ é o vetor com os resultados obtidos de cada uma das *p*-funções objetivo no ponto ótimo ($x^*$).

O problema de otimização multi-objetivo pode ser re-escrito da seguinte forma reduzida:

$$otimizar\ f(x) = \{ f_1(x), f_2(x), \ldots, f_p(x)\} \qquad \textbf{(A.6)}$$

onde:

$f : \Re^n \Rightarrow \Re^p$, é o vetor formado pelas *p*-funções objetivo;

Neste caso, para todas as funções objetivo o critério de otimização deverá ser o mesmo, ou seja, em todas as funções, a otimização deve ser somente de minimização, ou somente de maximização (max { $f_i$ (z)} = – min { – $f_i$ (z)}, i = 1, 2, ..., p).

Existem hoje, diversos métodos para se encontrar a solução ótima (*x**) de um *MOP*, e um dos mais usuais é o de transformar o problema de otimização multi-objetivo em um mono-objetivo através de métodos ou modelos matemáticos. Uma das transformações mais simples é através da soma ponderada dos valores numéricos das funções objetivos de um determinado problema para obter um único valor numérico, conforme mostrado abaixo:

$$f(x) = w_1 \times f_1(x) + w_2 \times f_2(x) + \ldots + w_p \times f_p(x) \qquad \textbf{(A.7)}$$

onde $w_i$, $\forall\ i \in \{1, 2, \ldots, p\}$, é um peso associado a cada uma das *p*–funções objetivo.

Porém, nem sempre esta prática é viável e satisfatória devido, por exemplo, à incompatibilidade dos tipos de resultados a serem matematicamente manipulados.

Como a solução ótima de um *MOP* é dada pelo conjunto formado pela solução de cada uma das funções objetivos, que melhore um objetivo sem piorar aos outros. Assim sendo, vários métodos baseados em computação evolucionária foram desenvolvidos para determinar soluções ótimas de um *MOP*. A ferramenta mais comum utilizada em vários destes métodos são aqueles baseados nas regras de *Dominância e Otimalidade de Pareto*.

## A.5 Regra de Dominância de Pareto

O conjunto de soluções de um problema Multi-Objetivo, usando as regras de *Dominância de Pareto*, é formado pelas soluções onde não existe um ordenamento (ou seja, não há forma de definir, a partir da avaliação dos valores numéricos das funções objetivos, qual solução é melhor do que a outra). Desta forma, as soluções ($x^*$) pertencentes a este conjunto serão sempre consideradas como sendo as melhores.

Dessa forma, sejam os vetores $u = \{u_1, u_2, ..., u_p\}$ e $v = \{ v_1, v_2, ..., v_p \}$ soluções de um problema de minimização com $p$-funções objetivo. O vetor $u$ domina ao vetor $v$ se e somente se $u$ é parcialmente menor do que $v$, portanto para $u$ dominar $v$ os seguintes critérios deverão ser satisfeitos:

i - $u_i \leq v_i$, $\forall\ i \in \{1, 2, ..., p\}$, ou seja, todos os elementos de $u$ deverão ser menores ou iguais aos do vetor $v$;

ii - $\exists\ i \in \{1, 2, ..., p\} \mid u_i < v_i$, ou seja, existe pelo menos um único elemento no vetor $u$ menor do que o seu correspondente elemento no vetor $v$.

Seja, por exemplo, o problema de otimização mostrado abaixo, onde $x \in \{x_a, x_b, x_c, x_d, x_e, x_f, x_g\}$:

$$minimizar\ f(x) = \{ f_1(x), f_2(x), f_3(x)\} \qquad \textbf{(A.8)}$$

Têm-se os seguintes resultados da função objetivo para os sete diferentes valores de $x$:

$$\begin{aligned} a &= f(x_a) = \{1.0, 5.0, 4.0\} \\ b &= f(x_b) = \{2.0, 3.0, 5.0\} \\ c &= f(x_c) = \{1.0, 3.0, 5.0\} \\ d &= f(x_d) = \{1.0, 3.0, 4.0\} \\ e &= f(x_e) = \{1.0, 3.0, 5.0\} \\ f &= f(x_f) = \{1.0, 2.0, 5.0\} \\ g &= f(x_g) = \{3.0, 1.0, 6.0\} \end{aligned} \qquad \textbf{(A.9)}$$

A **Tabela A 1** mostra quais são os vetores ($a$, $b$, $c$, $d$, $e$, $f$, ou $g$) que dominam, e quais deles são dominados no problema de minimização apresentado. Observa-se que, apesar do vetor $g$ possuir os piores elementos referentes às funções objetivo $f_1$ e $f_3$, o elemento referente à $f_2$ é o melhor (menor), e devido a estas características, o vetor $g$ não domina e nem é dominado por outros vetores.

**Tabela A.1** Pareto dominância.

| Vetor | Domina o(s) vetor(es) | Dominado por |
|---|---|---|
| *a* | – | *d* |
| *b* | – | *c, d, e, f* |
| *c* | *b* | *d, f* |
| *d* | *a, b, c, e* | – |
| *e* | *b* | *d, f* |
| *f* | *b, c, e* | – |
| *g* | – | – |

## A.6 Soluções Pareto Ótimo

Uma solução $x_u \in U$ é uma solução Pareto ótimo, se e somente se, não há nenhum $x_v \in U$ no qual $v = f(x_v) = \{v_1, v_2, ..., v_p\}$ domine $u = f(x_u) = \{u_1, u_2, ..., u_p\}$.

Portanto, uma solução $x^*$ é considerada como sendo Pareto ótimo se e somente se não existir outra solução $x$, tal que $x$ possua um resultado que melhore um dos critérios de otimização (função objetivo) sem piorar outro critério (Coelho, 1999). Desta forma, problemas multi-objetivo fornecerão inúmeras soluções que constituirão o conjunto de soluções Pareto ótimo.

Como mostrado na **Tabela A.1**, as soluções $x_d$, $x_f$ e $x_g$ são as únicas associadas ao conjunto de soluções Pareto ótimo, pois são as únicas que produzem funções objetivo não dominadas. Os valores numéricos destas funções objetivo são chamados de vetores não-dominados (Horn, 1997). Portanto, as soluções ótimas do exemplo apresentado na seção 2.2 são: $x_d$, $x_f$ e $x_g$.

O conjunto de soluções não-dominadas forma a chamada *Frente de Pareto* (Coelho, 1999). A solução ótima final correspondente a um MOP é obtida escolhendo-se uma das soluções contidas no conjunto Pareto ótimo, esta escolha é feita por um agente tomador de decisões (*decision maker*) (Veldhuizen e Lamont, 1998).

## A.7 Otimização Multi-Objetivo usando AG

Solucionar problemas de otimização com funções de diversas grandezas (sejam físicas, econômicas, adimensionais, etc.) tem sido desde os primórdios do desenvolvimento da teoria da otimização uma meta importante.

Assim sendo, cada método de otimização diferente que já foi implementado se baseia em diversas premissas a respeito da característica da função matemática a otimizar: linearidade, convexidade, diferenciabilidade, etc. A classe de métodos que atingiu a melhor aproximação para o problema da otimização de funções matemáticas de diversas grandezas pertence à família dos *Métodos Estocásticos de Otimização*, e o Algoritmo Genético pertence a essa família.

Devido às características, já descritas, dos *AG* e a capacidade de se adaptar a uma ampla gama de problemas tecnológicos e científicos, se justifica sua aplicação em problemas do tipo Multi-Objetivo. Além disso, a idéia de se utilizar *AG* para resolver problemas de otimização multi-objetivo foi primeiramente proposto na década dos 60 por Rosenberg (Rosenberg, 1967) em problemas de química.

# Anexo B

# Algoritmos Genéticos Paralelos

## B.1 Introdução

Os *AG* seqüenciais demonstraram ser uma boa ferramenta de otimização global para problemas tecnológicos complexos, como demonstram diversos trabalhos conhecidos na literatura [1], [7], [28], [13], devido à relativa simplicidade na sua implementação e à robustez do seu método de busca. Mas, na prática, os *AG* seqüenciais precisam de muito tempo de processamento para alcançar boas soluções. Uma alternativa que permita a redução dos recursos computacionais pode ser a paralelização dos *AG*.

O objetivo da paralelização consiste em dividir um determinado problema global em vários subproblemas, os quais são alocados em diversos computadores disponíveis numa rede de microcomputadores, ou em vários processadores de um computador paralelo. Assim, cada processador executa a computação do subproblema nele alocado obtendo resultados parciais que serão transmitidos aos outros processadores do sistema, ou da rede de computadores sendo utilizada, colaborando todos eles na solução do problema global.

Além da redução no tempo de processamento, os *AG* Paralelos (*AGP*) podem melhorar ainda os resultados numéricos quando comparados com as soluções fornecidas pelos *AG* seqüenciais, isso por causa do efeito sinergético da computação paralela nos algoritmos evolutivos.

A paralelização dos *AG* pode ser feita através de diversos métodos divulgados na literatura [13], [14], [12]. Alguns métodos usam uma única população, enquanto que outros dividem essa população em um conjunto de sub-populações que são executadas separadamente. Nestes últimos, as sub-populações comunicam um conjunto de *indivíduos* após um determinado número de iterações (gerações) do *AG*.

A variedade de modelos que vem sendo propostos é baseada nas características do *hardware* do sistema computacional utilizado. Essas características têm a ver com a tecnologia de comunicação utilizada (rapidez nos *buses* num computador paralelo, ou tipo de cabo de rede num *cluster* de computadores), número de processadores pertencentes ao computador paralelo ou número de microcomputadores numa rede, entre outros.

## B.2 Arquiteturas para Processamento Paralelo

Nesta seção serão descritos as diversas características do *hardware* dos computadores paralelos, bem como as das redes de computadores, que determinam as diversas metodologias utilizadas na literatura para abordar problemas de difícil solução, ou incluso aqueles que não tem solução através de técnicas seqüenciais tradicionais, através de algoritmos de processamento paralelo.

O fato dos processadores compartilharem, o não, a mesma memória de armazenamento de dados é uma das principais diferenças entre os modelos de arquiteturas paralelas. Portanto, existem duas grandes famílias de arquiteturas paralelas: com memória compartilhada e com memória distribuída.

Um multi-processador com memória compartilhada é um computador clássico no qual foram adicionados vários *chips* microprocessadores, e cada um deles tem acesso, para leitura e escritura de dados, num único espaço de memória. O acesso, a essa memória compartilhada, é feito através de um *bus* de dados de uso comum. Cada processador trabalha de forma assíncrona, e qualquer mecanismo de sincronização necessária deve ser feito de forma explicita.

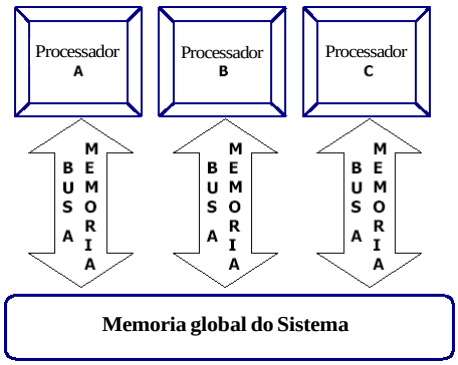



**Figura B.1:** Arquitetura paralela de memória compartilhada.

Nos sistemas multi-processadores com memória distribuída existe uma memória privada pertencente a cada *chip* microprocessador, que não é visível para os outros processadores. A comunicação, e sincronização, entre os processadores são feitas de forma explícita. Uma vantagem que apresenta este tipo de arquitetura é que não existe o problema de engarrafamento na comunicação dos dados, problema que existe no caso da memória compartilhada; porém, tem a desvantagem de um aumento no tempo de processamento quando executa uma tarefa semelhante comparada no caso da memória global.

Um tipo de multi-processadores de memória distribuída são os *clusters* de computadores, ou as redes de computadores, interconectados através de cabos de transmissão de dados, de baixa e mediana velocidade. Um recurso compartilhado, como os cabos de comunicação, introduz uma desvantagem em comparação com outros modelos de arquitetura paralela, do ponto de vista da eficácia. Esse tipo de arquitetura ficou popularizado nas últimas décadas como conseqüência de seu baixo custo operativo além de serem altamente escaláveis; isto é, permite o aumento do recurso computacional disponível só adicionando mais computadores à rede.

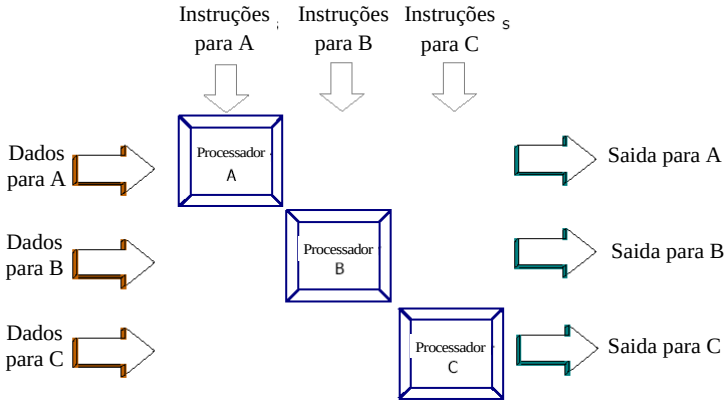



**Figura B.2:** Arquitetura paralela de memória distribuída.

## B.3 Ferramentas para a implementação paralela

As bibliotecas de funções, baseadas em envio de mensagens, oferecem um maior nível de abstração para a comunicação de soluções, ou resultados, parciais entre processadores, resolvendo assim, os problemas de manipulação de tipos de dados e representações, tolerância à falhas, mecanismos de segurança, entre outros.

Estas bibliotecas apresentam, numa forma integral, diversas rotinas para a transferência de dados e sincronização na comunicação entre processadores, evitando assim a necessidade do programador de usar linguagem de programação de baixo nível. Exemplos desse tipo de bibliotecas de funções podem ser o "*Parallel Virtual Machine" – PVM*, e o *Message Passing Interface – MPI*, entre os mais utilizados na literatura. Na subseção a seguir, é descrita a biblioteca MPI; a qual foi utilizada no presente trabalho para desenvolver uma versão paralela do algoritmo de *Roteamento Ótimo*.

### B.3.1 A biblioteca *MPI*

O *MPI Standard* foi definido em 1992 por um conjunto de programadores de *software* para aplicações cientificas, em conjunto com importantes empresas, como a *IBM*, *Intel*, *PVM* e *nCUBE*, para proporcionar uma biblioteca implementável sobre diversas arquiteturas multiprocessador com a finalidade de desenhar aplicações distribuídas portáveis e eficientes. As características principais desta biblioteca são apresentadas a seguir:

- Trabalhar sob plataformas de memória distribuída, embora inclua formas primitivas para ser utilizadas sob plataformas compartilhadas.
- Trabalhar na idéia de que o paralelismo é explicito, já que o programador tem total controle das aplicações.
- Propõe-se como mecanismo de implementação de algoritmos paralelos, o modelo *SPMD* (um único programa executado em forma assíncrona sob diferentes conjuntos de dados), podendo ser adaptado para o modelo *MPMD* (diversos programas executados concorrentemente sob diferentes conjuntos de dados).

O *MPI* suporta o desenvolvimento de algoritmos paralelos e distribuídos, onde os componentes se comunicam através de funções de recepção e envio de mensagens desde e até os outros processadores. O conjunto de processadores tem um número que é fixo, e é definido no momento da criação da rede, podendo se utilizar diferentes tipos de transmissão, como comunicação *Ponto a Ponto* ou coletivas, síncronas ou assíncronas, etc., sempre de acordo as necessidades da lógica de aplicação paralela.

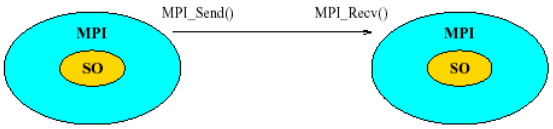



**Figura B.3:** Representação da utilização do *MPI*.

## B.4 Classificação dos *AGP*

Considerando os critérios descritos na seção anterior, os *AGP* podem ser classificados segundo a quantidade de processadores envolvidos na execução do algoritmo paralelo desenvolvido. Alguns são implementados com a finalidade de redução do tempo de processamento, enquanto outros, além de reduzir o tempo de cômputo, melhoram a qualidade da solução. Assim, de acordo com os diversos métodos de paralelização do *AG* é possível estabelecer quatro tipos de *AGP*, os quais são descritos a seguir:

Os *AGP* do tipo *Mestre/Escravo* usam uma única população, sendo que o processo que é executado em diversos processadores do sistema computacional corresponde à avaliação dos *indivíduos*,

1) Os *AGP* de *granularidade* fina usam uma única população distribuída; isto é, cada *indivíduo* está alocado num único processador do sistema computacional,
2) *AGP* de *granularidade* grossa e múltiplas populações.
3) *AGP* do tipo Hierárquico, nos quais são combinadas as metodologias de Mestre/Escravo com os *AGP granularidade* fina no caso de implementação num computador paralelo, ou com o método de *granularidade grossa* no caso de redes de computadores.

Nas subseções seguintes serão descritos cada uma das metodologias, listadas acima, para transformar os *AG* seqüenciais em processos paralelos.

### B.4.1 *AGP* do tipo *Mestre/Escravo*.

Neste tipo de implementação a avaliação dos *indivíduos* da população é explicitamente paralelizada, sendo o processo *Mestre* quem aplique em cada geração os operadores genéticos para obtenção de novos *indivíduos*. O *AGP* do tipo *Mestre/Escravo* pode ser muito eficiente quando a avaliação dos indivíduos investe considerável quantidade de cálculos e por tanto, precisa-se reduzir o tempo de processamento. Porém, a qualidade dos resultados obtidos não é semelhante aos resultados fornecidos por um determinado *AG* seqüencial.

Neste tipo de *AGP*, o processador *Mestre* distribui os *indivíduos* ao resto dos processadores (*escravos*), do computador paralelo ou da rede de microcomputadores, para serem avaliados. O processador *Mestre* também executa a avaliação e nele fica armazenada toda a população. Uma ilustração de como este método trabalha é mostrado na **Figura B.4**.

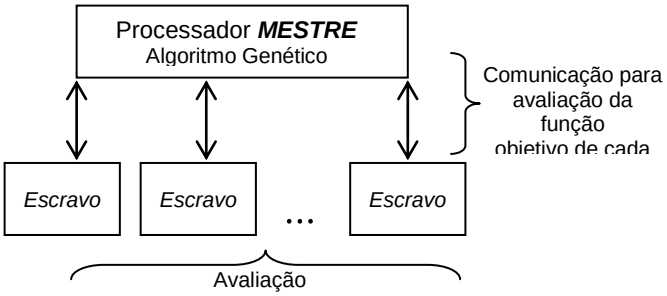



**Figura B.4:** Topologia de comunicação usada no método de paralelização global do AG.

Como pode ser observado na **Figura B.4**, não existe comunicação de dados entre os *escravos* durante a avaliação dos *indivíduos.* Uma vez que cada escravo termine de avaliar o conjunto de indivíduos nele associados, os *fitness* calculados são devolvidos ao Mestre.

O pseudocódigo da implementação do AGP desenvolvida no presente trabalho é mostrada a seguir:

```
Ler dados de entrada
Iniciar semente do gerador de números pseudoaleatórios;
t ← 0;
Criar População (t);de forma pseudoaleatoria;
Dividir População (t);
Enviar os indivíduos aos Escravos para avaliação;
Avaliação no Mestre de uma parte da População(t);
Recepção dos resultados da avaliação de todos os Escravos;
Estatística da População (t);
FAZER ENQUANTO um critério de parada não for alcançado
  Selecionar Indivíduos da População (t) para obter a população (t+1);
  Cruzar e mutar População (t )para obter a população (t+1);
  Dividir População;
  Enviar os indivíduos aos escravos para avaliação;
  Avaliação no mestre de uma parte da População (t+1);
  Recepção dos resultados da avaliação de todos os Escravos;
  Estatística da População (t+1)
      t ← t + 1;
FIM FAZER ENQUANTO
```

**Figura B.5:** Pseudocódigo do Mestre do AGP do tipo Mestre/Escravo.

No pseudocódigo do Mestre do AGP de tipo Mestre / Escravo o mestre divide a população para os escravos mais ele também faz o cálculo do valor da função objetivo para cada individuo do grupo correspondente, designado a ele. Depois de cada escravo finalizar a sua computação, envia ao Mestre os resultados da avaliação. Por tanto, cada Escravo tem como objetivo único fazer o computo dos valores da função objetivo do grupo designado e enviar de volta ao Mestre.

Depois da avaliação dos indivíduos obtêm-se a estatística da população com a finalidade de testar a evolução do algoritmo genético. Logo enquanto o critério de parada não for atingido, aplicam-se os operadores de seleção, da recombinação e da mutação sobre a população para obter indivíduos cada vez mais aptos para as seguintes gerações, avançando em direção a solução. O calculo do valor da função objetivo, durante as iterações ou gerações se faz parte de forma paralela.

No algoritmo síncrono, o *Mestre* aguarda todos os *escravos* executarem suas tarefas (avaliação da função objetivo de cada *indivíduo*) para dar continuidade ao processamento. Esta espera contribui para o aumento do tempo de processamento, se os processadores do computador paralelo, ou os diferentes computadores da rede, tiverem diferentes velocidades de cômputo. Este método de paralelização tem as mesmas características de um *AG* seqüencial.

Entretanto, no método assíncrono, esse tempo de espera não acontece, porém, ele não trabalha como um *AG* simples. Em geral, na literatura os *AGP* do tipo *Mestre/Escravo* são implementados considerando comunicação síncrona.

A implementação deste tipo de *AGP* pode ser feita em computadores de memória compartilhada, e também em computadores de memória distribuída, ou mesmo em uma rede de computadores porque não sofre influência da arquitetura computacional usada.

Os *AGP* do tipo *Mestre/Escravo*, implementados em computadores de memória compartilhada, a população é armazenada na memória de acesso comum e cada processador lê os *indivíduos* alocados nele; mas, a avaliação dos *Fitness* é distribuída, e feita em cada processador do sistema. Porém, no computador paralelo que tem múltiplos processadores que usam memória distribuída, a população é colocada em um único processador designado como o *Mestre*. O resto dos processadores do sistema constitui os *escravos*.

Uma rede de computadores é semelhante a um computador de memória distribuída. Assim, o *AGP* do tipo *Mestre/Escravo* é implementado da mesma forma descrita para os computadores paralelos de memória distribuída.

Pesquisas recentes mostram que uma maior redução no tempo de computação é obtida com o aumento do número de processadores disponíveis no sistema computacional. Porém, devido a um número maior de processadores utilizados espera-se um aumento no tempo de transmissão [12].

Existirá, então, uma solução de compromisso entre a redução do tempo e aumento no tempo de comunicação. Nessa solução de compromisso existe um número ótimo de *escravos* que minimiza o tempo total da execução.

### B.4.2 *AGP* de *granularidade* fina.

Antes da descrição deste tipo de implementação é conveniente explicar o significado de *granularidade* em paralelismo.

A *granularidade* em paralelismo é o resultado da divisão entre o tempo de computação e o tempo de comunicação, e sua expressão matemática é apresentada a seguir:

$$granularidade = \frac{\text{Tempo de computação}}{\text{Tempo de comunicação}} \qquad \textbf{(B.1)}$$

Para que implementações paralelas sejam consideradas de *granularidade* fina, a expressão matemática (B.1) deve fornecer um valor numérico levemente superior a unidade pouco maior do que a unidade (*granularidade* > 1), porque o valor do tempo de computação é pouco maior do que o tempo de comunicação. O *AGP* de *granularidade* fina é adequado para implementações em computadores massivamente paralelos, devido ao grande número de processadores que eles possuem. A topologia de comunicação entre os diversos processadores forma um quadriculado, como mostrada na **Figura B.5**. Cada ponto da quadrícula corresponde a um processador do sistema.

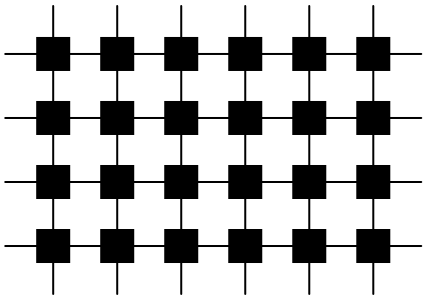


**Figura B.6:** Topologia de comunicação usada nos Algoritmos Genéticos Paralelos de *granularidade* fina.

No *AGP* de *granularidade* fina, cada processador contém unicamente um *indivíduo*. Para fazer a recombinação é selecionado um *indivíduo* da vizinhança, e logo é aplicado o operador de *Cruzamento* e o operador de *Mutação*.

No final da operação anterior de recombinação e mutação, é selecionado o melhor dos descendentes (indivíduo com *Fitness* maior) substitui ao *indivíduo* anterior do processador correspondente. O processo de maximização é feito de forma assíncrona e a população tende a manter um bom nível de diversidade.

### B.4.3 *AGP* de *granularidade* grossa.

Neste tipo de implementação paralela, a *granularidade* é muito maior do que a unidade (*granularidade* >> 1) devido a que o tempo de computação é maior do que o tempo de comunicação. Os *AGP* com *granularidade* grossa, também denominados de "*Multi-População*", são compostos por poucas sub-populações de tamanho relativamente grande, e são caracterizados por ter uma implementação bem mais trabalhosa do que os *AG* seqüenciais. Esta dificuldade está associada, principalmente, à introdução de um novo operador denominado de *Migração*. Além disso, o número de sub-populações é igual ao número de processadores disponíveis.

A *Migração* estabelece a forma como será organizada, e estruturada, a comunicação dos *indivíduos* entre os diversos processadores disponíveis. Os novos parâmetros introduzidos na *Migração* são descritos a seguir [13], [12]:

1) *Intervalo de Migração*. Estabelece quantas iterações (gerações) têm que transcorrer num determinado processador antes de fazer o intercâmbio de um determinado número de *indivíduos* com os outros processadores do sistema computacional utilizado. Uma questão importante que surgiu em diferentes estudos sobre a *Migração* está relacionada ao instante certo para executar a migração. Uma opção é a *Migração* ocorrer cedo. Porém, nas primeiras gerações ainda não existem bons *indivíduos* (com altos *Fitness*) cujas características genéticas (variáveis codificadas no *cromossomo*) possam influenciar na direção certa da busca, após a transmissão. Então, recursos de comunicação de alto custo computacional seriam desperdiçados, o que indica que existe um instante no qual é vantajoso executar a transmissão onde a quantidade de bons *indivíduos* é maior.

2) *Taxa de Migração*. Indica quantos *indivíduos* serão comunicados às outras sub-populações quando o *intervalo de Migração* ocorre.

3) *Critério de Seleção dos Indivíduos*. Define quais serão os *indivíduos* que migrarão. Devido à filosofia aleatória dos *AG* pode-se estabelecer que os *indivíduos* que serão transmitidos sejam escolhidos aleatoriamente. Porém, é possível auxiliar às outras sub-populações selecionando os *indivíduos* que possuem altos *Fitness*. Esta última opção parece mais natural, do ponto de vista biológico, pois na Natureza as migrações significam grandes esforços para os viajantes e somente os mais fortes sobrevivem à travessia.

4) *Topologia de Comunicação*. Estabelece como estarão organizadas as comunicações entre os diferentes processadores disponíveis para a transmissão dos *indivíduos*. A *Topologia da Comunicação* determina quão rápido ou quão lento uma boa solução é disseminada entre as sub-populações, e também se a topologia tem uma conectividade densa ou fraca [13], [14]. A conectividade, na *Topologia de Comunicação*, determina com quantas sub-populações está conectado um processador para a migração. Para uma conectividade densa, mostrado na **Figura B.7**, soluções boas se disseminarão mais rapidamente para todas as sub-populações. Por outro lado, se a topologia for esparsamente conectada, mostrado na **Figura B.8**, soluções boas se espalharam mais lentamente e as sub-populações estarão mais isoladas entre si permitindo soluções diferentes em cada computador.

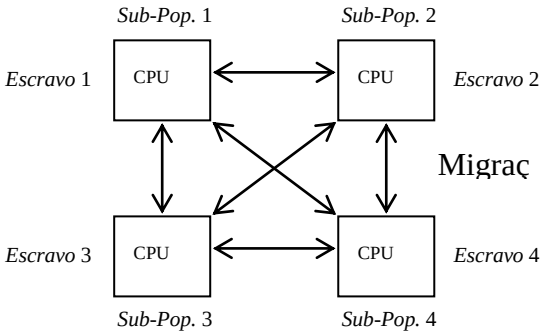



**Figura B.7:** Topologia de Comunicação usada nos AGP de *granularidade* grossa e conectividade densa.

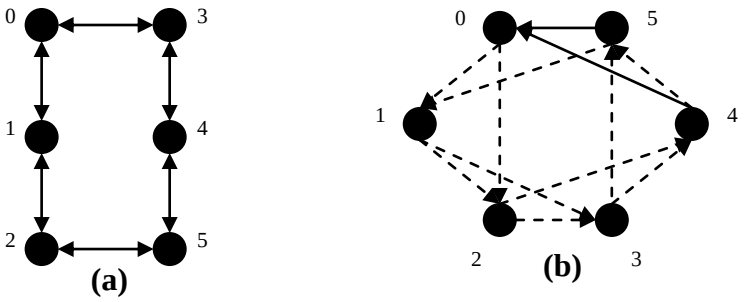



**Figura B.8:** Diferentes topologias de comunicação usada nos AGP de *granularidade* grossa e conectividade = 2.

### B.4.4 *AGP* Hierárquicos.

Utilizam a combinação de dois métodos de paralelização do *AG* descritos anteriormente. Esses algoritmos, também denominados de algoritmos mistos ou híbridos, adicionam um novo grau de complexidade à técnica de paralelização dos *AG*.

A combinação de dois métodos de paralelização de *AG* exige uma hierarquia para a comunicação e execução do algoritmo. Geralmente são implementados dividindo a população global em um determinado número de sub-populações igual ao número de processadores disponíveis no sistema. Em cada uma dessas sub-populações é executado o *AG*, mas no momento da avaliação de cada *indivíduo* ela é também paralelizada usando o método *Mestre/Escravo*. A migração de *indivíduos* ocorre entre as sub-populações. Essa técnica pode ser útil para aplicações complexas que necessitem de considerável quantidade de cálculos para obter a avaliação.

A **Figura B.9** e a **Figura B.10** mostram as topologias de dois tipos de *AGP* hierárquicos, que podem ser usados de acordo com a disponibilidade dos recursos computacionais.

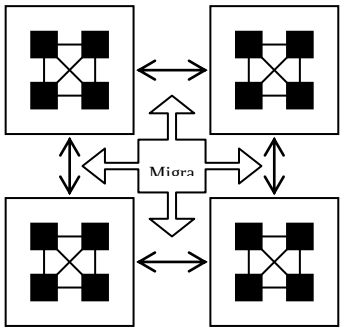



**Figura B.9:** *AGP* Hierárquico onde são combinados dois *AGP* com *granularidade* grossa.

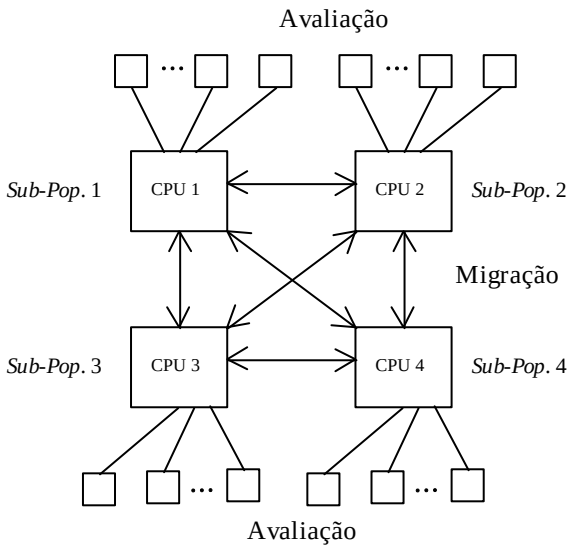



**Figura B.10:** AGP Hierárquico onde são combinados o AGP de *granularidade* grossa para a migração de *indivíduos* entre as subpopulações, e o AGP *Mestre/Escravo* para fazer a avaliação.

As implementações que usam o método do *AGP* Hierárquico podem reduzir ainda mais o tempo de computação do que o uso de um de seus componentes sozinho. Além disso, pode-se encontrar melhores soluções comparado com cada uma das técnicas de paralelização isoladamente.

# Anexo C

# Dados do Sistema de Transporte Argentino

## C.1 Introdução

Neste Apêndice serão apresentados os dados utilizados no sistema de Transporte Argentino [20] para validação do algoritmo de *Roteamento* proposto. O sistema argentino de transporte pode ser mais bem descrito através do seguinte mapa:


**Figura C. 1:** Mapa da Argentina para descrever o Sistema de Transporte correspondente.

Através da **Figura C.2** é apresentada a matriz de distâncias no sistema de transporte argentino.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 1543 | 1510 | 1203 | 1043 | 1191 | 1023 | 478 | 940 | 1040 | 480 | 715 | 1150 | 1110 | 790 | 1155 | 1050 | 620 | 1158 | 960 | 1455 | 2635 | 3228 |
| **2** | 1543 | 0 | 99 | 340 | 500 | 960 | 860 | 1107 | 883 | 1198 | 1138 | 930 | 770 | 1220 | 1320 | 572 | 1345 | 1530 | 2200 | 2124 | 2385 | 3565 | 4158 |
| **3** | 1510 | 99 | 0 | 307 | 467 | 948 | 780 | 1074 | 803 | 1118 | 1105 | 897 | 695 | 1145 | 1245 | 539 | 1227 | 1497 | 2082 | 2091 | 2352 | 3532 | 4125 |
| **4** | 1203 | 340 | 307 | 0 | 160 | 936 | 768 | 767 | 791 | 1106 | 798 | 590 | 338 | 838 | 938 | 232 | 1005 | 1190 | 1860 | 1784 | 2045 | 3225 | 3818 |
| **5** | 1043 | 500 | 467 | 160 | 0 | 776 | 610 | 607 | 633 | 948 | 638 | 430 | 360 | 810 | 850 | 212 | 977 | 1030 | 1567 | 1624 | 1885 | 3065 | 3658 |
| **6** | 1191 | 960 | 948 | 936 | 776 | 0 | 168 | 713 | 191 | 506 | 744 | 1043 | 1136 | 1543 | 1463 | 988 | 1710 | 1523 | 2060 | 2117 | 2378 | 3558 | 4151 |
| **7** | 1023 | 860 | 780 | 765 | 610 | 168 | 0 | 545 | 23 | 338 | 576 | 875 | 970 | 1420 | 1295 | 822 | 1587 | 1475 | 2012 | 2069 | 2210 | 3390 | 3983 |
| **8** | 478 | 1107 | 1074 | 767 | 607 | 713 | 545 | 0 | 568 | 883 | 31 | 330 | 765 | 830 | 625 | 770 | 885 | 810 | 1347 | 1404 | 1665 | 2845 | 3438 |
| **9** | 940 | 883 | 803 | 791 | 633 | 191 | 23 | 568 | 0 | 315 | 590 | 898 | 993 | 1398 | 1318 | 845 | 1565 | 1378 | 1989 | 2046 | 2187 | 3367 | 3960 |
| **10** | 1040 | 1198 | 1118 | 1106 | 948 | 506 | 338 | 883 | 315 | 0 | 820 | 1213 | 1308 | 1758 | 1633 | 1160 | 1925 | 1660 | 2198 | 2000 | 2498 | 3675 | 4268 |
| **11** | 480 | 1138 | 1105 | 798 | 638 | 744 | 576 | 31 | 590 | 820 | 0 | 361 | 796 | 861 | 656 | 801 | 916 | 841 | 1378 | 1435 | 1696 | 2876 | 3469 |
| **12** | 715 | 930 | 897 | 590 | 430 | 1043 | 875 | 330 | 898 | 1213 | 361 | 0 | 435 | 500 | 420 | 440 | 670 | 600 | 1137 | 1194 | 1455 | 3635 | 3228 |
| **13** | 1150 | 770 | 695 | 338 | 360 | 1136 | 970 | 765 | 993 | 1308 | 796 | 435 | 0 | 450 | 550 | 156 | 617 | 1035 | 1472 | 1629 | 1890 | 3070 | 3663 |
| **14** | 1110 | 1220 | 1145 | 838 | 810 | 1543 | 1420 | 830 | 1398 | 1758 | 861 | 500 | 450 | 0 | 320 | 606 | 167 | 825 | 1022 | 1419 | 1680 | 2860 | 3453 |
| **15** | 790 | 1320 | 1245 | 938 | 850 | 1463 | 1295 | 625 | 1318 | 1633 | 656 | 420 | 550 | 320 | 0 | 705 | 260 | 505 | 883 | 1099 | 1360 | 2540 | 3133 |
| **16** | 1145 | 572 | 539 | 232 | 212 | 988 | 822 | 770 | 845 | 1160 | 801 | 440 | 156 | 606 | 705 | 0 | 773 | 1040 | 1588 | 1634 | 1895 | 3075 | 3668 |
| **17** | 1050 | 1345 | 1227 | 1005 | 977 | 1710 | 1587 | 885 | 1565 | 1925 | 916 | 670 | 617 | 167 | 260 | 773 | 0 | 765 | 855 | 1359 | 1620 | 2800 | 3393 |
| **18** | 620 | 1530 | 1497 | 1190 | 1030 | 1523 | 1475 | 810 | 1378 | 1660 | 841 | 600 | 1035 | 825 | 505 | 1040 | 765 | 0 | 537 | 594 | 855 | 2035 | 2628 |
| **19** | 1158 | 2200 | 2082 | 1860 | 1567 | 2060 | 2012 | 1347 | 1989 | 2198 | 1378 | 1137 | 1472 | 1022 | 883 | 1588 | 855 | 537 | 0 | 660 | 750 | 1930 | 2523 |
| **20** | 960 | 2124 | 2091 | 1784 | 1624 | 2117 | 2069 | 1404 | 2046 | 2000 | 1435 | 1194 | 1629 | 1419 | 1099 | 1634 | 1359 | 594 | 660 | 0 | 495 | 1675 | 2268 |
| **21** | 1455 | 2385 | 2352 | 2045 | 1885 | 2378 | 2210 | 1665 | 2187 | 2495 | 1696 | 1455 | 1890 | 1680 | 1360 | 1895 | 1620 | 855 | 750 | 495 | 0 | 1180 | 1773 |
| **22** | 2635 | 3565 | 3532 | 3225 | 3065 | 3558 | 3390 | 2845 | 3367 | 3675 | 2876 | 3635 | 3070 | 2860 | 2540 | 3075 | 2800 | 2035 | 1930 | 1675 | 1180 | 0 | 593 |
| **23** | 3228 | 4158 | 4125 | 3818 | 3658 | 4151 | 3983 | 3438 | 3960 | 4268 | 3469 | 3228 | 3660 | 3453 | 3133 | 3668 | 3393 | 2628 | 2523 | 2268 | 1773 | 593 | 0 |

**Figura C. 2:** Matriz de Distância associada ao sistema de Transporte argentino.

A seguir, na **Tabela C.1** é descrita a nomenclatura utilizada para definir cada nó desse sistema de transporte.

**Tabela C.1** Definição de cada nó.

| Nó | Cidade |
|---|---|
| **1** | BUENOS AIRES |
| **2** | SAN. S. DE JUJUY |
| **3** | SALTA |
| **4** | S. M. DE TUCUMAN |
| **5** | SGO. DEL ESTERO |
| **6** | FORMOSA |
| **7** | RESISTENCIA |
| **8** | SANTA FE |
| **9** | CORRIENTES |
| **10** | POSADAS |
| **11** | PARANA |
| **12** | CORDOBA |
| **13** | LA RIOJA |
| **14** | SAN JUAN |
| **15** | SAN LUIS |
| **16** | CATAMARCA |
| **17** | MENDOZA |
| **18** | SANTA ROSA |
| **19** | NEUQUEN |
| **20** | VIEDMA |
| **21** | RAWSON |
| **22** | RIO GALLEGOS |
| **23** | USHUAIA |