



**UNIVERSIDADE FEDERAL DA BAHIA**

**LABORATÓRIO DE SISTEMAS DISTRIBUÍDOS**

**ESPECIALIZAÇÃO AVANÇADA EM SISTEMAS DISTRIBUÍDOS**

**ISAC VELOZO DE C. AGUIAR**

**AGMAX**

**Ferramenta para gerenciamento de  
algoritmos genéticos distribuídos em ilhas de evolução**

Salvador

2009

**ISAC VELOZO DE C. AGUIAR**

**AGMAX**

**Ferramenta para gerenciamento de  
algoritmos genéticos distribuídos em ilhas de evolução**

Monografia apresentada ao Curso de  
Especialização Avançada em Sistemas  
Distribuídos do Laboratório de Sistemas  
Distribuídos, Universidade Federal da  
Bahia.

Orientador: Prof. Frederico Barboza

Salvador

2009

**ISAC VELOZO DE C. AGUIAR**

**AGMAX**

**Ferramenta para gerenciamento de  
algoritmo genético distribuídos em ilhas de evolução**

**UNIVERSIDADE FEDERAL DA BAHIA**

Monografia apresentada ao Curso de Especialização Avançada em Sistemas  
Distribuídos do Laboratório de Sistemas Distribuídos, Universidade Federal da Bahia

Aprovada em 20 de maio de 2010

---

Banca Examinadora: Sandro Andrade, LaSiD/UFBA

---

Orientador: Frederico Barboza, LaSiD/UFBA

Salvador

2009

Aos,

Meus pais: Jurimar e Izabel, minha esposa Norma e meus filhos: Ian e Júlia,  
pelo apoio incentivo, conselhos e participação.

## **AGRADECIMENTOS**

Ao Prof. Frederico Barboza, por todo apoio que me deu, pelas demonstrações; por entender e com isso ter sabido orientar-me com excelência.

Aos professores que contribuíram para o acúmulo de conhecimento ao longo do curso, proporcionando um amadurecimento adequado para a conclusão do curso com este trabalho, em especial a professora Aline coordenadora do curso.

A meus colegas de curso, os quais compartilharam conhecimentos e emoções.

A meus amigos que me incentivaram nos momentos difíceis.

E enfim a todos que direta ou indiretamente me auxiliaram durante todo este período, que considero uma grande conquista.

## RESUMO

Os algoritmos genéticos são utilizados como mecanismos de otimização computacional, porém a sua utilização requer um elevado custo dos recursos computacionais, limitando bastante o contexto de sua utilização. A solução centralizada deste tipo de algoritmo pode limitar o seu processo de execução à apenas os recursos disponíveis no *host*, que a depender da complexidade da busca pela solução esperada utilize grande parte dos recursos disponíveis, conseqüentemente surge a necessidade da distribuição. Para um ambiente distribuído ideal é necessário ter um ambiente computacional que dê o suporte necessário ao gerenciamento das informações. Visando oferecer um ambiente adequado para o gerenciamento de AGs em um cenário distribuído, neste trabalho será apresentada uma ferramenta de gerenciamento de AGs distribuído em ilhas de evolução, denominada AGMAX.

**Palavras-chaves:** algoritmos genéticos, algoritmos genéticos em ilhas de evolução.

## LISTA DE FIGURAS

Figura 01: Código básico de execução de um AG .....	13
Figura 02: 1ª etapa do cruzamento .....	15
Figura 03: 2º etapa cruzamento .....	16
Figura 04: Modelo Global.....	21
Figura 05: Modelo de Granularidade Fina.....	22
Figura 06: Modelo de Granularidade Grossa .....	23
Figura 07: Ilhas de Evolução.....	25
Figura 08: Modelo Arquitetural AGMAX .....	30
Figura 09: Componentes de software do AGMAX.....	32
Figura 10: Modelo de Caso de Uso do Sistema .....	33
Figura 11: Interface Gráfica .....	35
Figura 12: Painel Global .....	36
Figura 13: Painel Individual.....	36
Figura 14: Classe ComponenteUdpUrl .....	41
Figura 15: Classe GerenciadorUdpUrl .....	41

## SUMÁRIO

1	INTRODUÇÃO.....	9
2	ALGORITMOS GENÉTICOS.....	11
2.1	ESTRUTURA BÁSICA DE UM ALGORITMO GENÉTICO.....	12
2.2	OPERADORES GENÉTICOS.....	13
2.2.1	SELEÇÃO.....	14
2.2.2	CRUZAMENTO.....	15
2.2.3	MUTAÇÃO.....	16
2.2.4	PARÂMETROS.....	17
3	ALGORITMOS GENÉTICOS PARALELOS.....	20
3.1	MODELOS DE ALGORITMOS GENÉTICOS PARALELOS.....	20
3.1.1	GLOBAL COM UMA ÚNICA POPULAÇÃO.....	20
3.1.2	POPULAÇÃO ÚNICA – GRANULARIDADE FINA.....	21
3.1.3	GRANULARIDADE GROSSA.....	22
3.1.4	ALGORITMOS GENÉTICOS EM ILHAS DE EVOLUÇÃO.....	23
4	GERENCIAMENTO DE AGS DISTRIBUÍDOS.....	26
4.1	SISTEMAS DISTRIBUÍDOS.....	26
4.2	ALGORITMOS GENÉTICOS EM SISTEMAS DISTRIBUÍDOS.....	29
5	AGMAX – FERRAMENTA DE GERENCIAMENTO DE AGS DISTRIBUÍDOS EM ILHAS DE EVOLUÇÃO.....	30
5.1	ARQUITETURA DO SISTEMA.....	30
5.1.1	COMPONENTES AGMAX.....	31
5.1.2	REQUISITOS FUNCIONAIS.....	32
5.2	INTERAÇÃO E USABILIDADE.....	34
	CONSIDERAÇÕES FINAIS.....	38
	REFERÊNCIAS BIBLIOGRÁFICAS.....	39
	ANEXOS A – CLASSES DO PROTOCOLO DE COMUNICAÇÃO.....	41
	ANEXOS B – CASOS DE USO.....	42



CASO DE USO: INICIAR COMUNICAÇÃO.....	42
CASO DE USO: INICIAR EXECUÇÃO.....	42
CASO DE USO: PARAR EXECUÇÃO.....	43
CASO DE USO: MUDAR PARÂMETROS.....	44
CASO DE USO: ENVIAR PARÂMETROS.....	45
CASO DE USO: NOTIFICAR.....	46

# 1 INTRODUÇÃO

Os algoritmos evolucionários formam uma classe de algoritmos de busca heurística, que baseiam-se na teoria evolutiva. Eles são geralmente aplicados em cenários computacionais onde se procura conseguir uma boa resultado para problemas com amplo espaço de soluções, para os quais não se conheça algoritmo eficiente.

Os principais modelos de algoritmos evolucionários são as *Estratégias Evolutivas*, *Programação Evolucionária* e *Algoritmos Genéticos* (Barcellos, 2000). Em comum, estes modelos usam a natureza como fonte de inspiração, baseiando-se na Teoria da Seleção Natural e da Evolução das Espécies. Seus funcionamentos simulam a evolução através da seleção, mutação e reprodução (Linden, 2008), com vistas à solução de problemas computacionais.

Os algoritmos genéticos (AGs) mantêm uma população de possíveis soluções que são avaliadas de forma simultânea, o que permite tratar problemas com múltiplos objetivos. Adicionalmente, AGs utilizam técnicas de busca global, não se limitando a máximos locais, como acontecem com alguns métodos de busca. Também não é uma forma de busca totalmente aleatória, pois apesar dos métodos aleatórios existentes, utilizam a informação da população corrente para guiar o processo de busca (Linden, 2008).

As técnicas utilizadas pelos AGs foram desenvolvidas para percorrer grandes espaços na busca de soluções, o que pode exigir um grande consumo dos recursos computacionais.

Contudo, o consumo computacional pode ser reduzido com os AGs Paralelos, que utilizam uma estratégia para obter, mais rapidamente, resultados de tarefas grandes e complexas. Dentre os modelos dos AGs Paralelos, temos o modelo em ilhas.

No modelo em ilhas, também conhecidos como AGs em Ilhas de Evolução. O algoritmo sequencial se divide numa série de ilhas (populações), similares aos ninchos ecológicos da natureza. Isto permite encontrar melhores soluções internas nas ilhas, como também permite a migração destas soluções para

outras ilhas existentes, bem como a criação de novas ilhas, desta forma ampliando as boas e distintas soluções para os problemas.

Apesar de todas as vantagens asseguradas no modelo em ilhas, a solução centralizada desta técnica, pode limitar bastante a sua utilização, devido a grande capacidade de consumo dos recursos computacionais. Isto favorece a necessidade da distribuição, que visa principalmente compartilhar recursos, desde componentes de hardware, até as entidades definidas pelo software.

Contudo, distribuição dificulta a gerência e o controle da execução do algoritmo. Isto torna importante a adoção de técnicas que permitam o monitoramento e coleta dos dados, neste caso dos dados processados pelos AGs. Preferencialmente, isto deve ser feito de forma transparente para o usuário, como se o mesmo estivesse utilizando uma solução centralizada.

Buscando satisfazer estes requisitos, foi desenvolvido no contexto deste trabalho, uma ferramenta para gerenciamento de AGs distribuídos em ilhas de evolução.

A ferramenta AGMAX permite o controle dos AGs distribuídos em uma rede, através da parametrização, do início da execução, do acompanhamento do processo de execução (durante a busca da solução) e da parada da execução (solicitada ou terminada) acionados de forma remota.

A ferramenta de gerência foi integrada ao framework JGA Net (PEREIRA, BORGES e FIGUEREDO, 2008), responsável pelo processamento dos AGs em Ilhas de Evolução.

Esta monografia está organizada da seguinte forma: capítulo 2 apresenta conceitos referentes aos AGs; capítulo 3 apresenta os conceitos acerca dos AGs Paralelos em especial os AGs em ilhas de evolução; capítulo 4 descreve o gerenciamento de AGs em um ambiente distribuído; capítulo 5 apresenta a ferramenta de gerenciamento, denominada AGMAX; por fim, são apresentadas as considerações finais e trabalhos futuros.

## 2 ALGORITMOS GENÉTICOS

Os AGs são um ramo dos algoritmos evolucionários que utilizam modelos computacionais dos processos naturais de evolução como uma ferramenta para solução de problemas (LINDEN, 2008). Apesar da grande variedade de modelos, todos se baseiam no conceito de simulação da evolução das espécies através da seleção, mutação e reprodução, dependente da qualidade dos indivíduos dessa espécie dentro do ambiente. Assim, AGs podem ser definidos como uma técnica de busca baseada numa metáfora do processo biológico de evolução natural.

Os AGs utilizam os cromossomos como representações das soluções. Os cromossomos podem ser codificados de diversas maneiras (binários, inteiros, reais e etc), de acordo com a forma mais adequada a representação da solução para o problema em questão. Por exemplo, pode-se utilizar uma cadeia de inteiros, para representar a ordem de visitação dos vértices de um grafo, como solução para o problema do caixeiro viajante (*Travelling Salesman Problem* – TSP).

Os AGs tendem a direcionar a busca para regiões do espaço onde é mais provável que o resultado seja o ótimo. Este espaço, definido como espaço de busca, é o conjunto de todas as possíveis soluções que podem ser encontradas para um determinado problema.

Para a execução do AG, é necessário que seja definido os parâmetros (probabilidade de recombinação e mutação, tamanho da população, etc). A partir destas informações é gerada uma população inicial de indivíduos, ela é submetida aos operadores genéticos (a seleção, a cruzamento e a mutação). Estes operadores simulam o processo de evolução natural da população, que eventualmente deverá resultar em um indivíduo que caracterizará uma boa solução (talvez até a melhor possível) para o problema (LINDEN, 2008).

Apesar de ser uma técnica de busca extremamente eficiente no seu objetivo de varrer o espaço de busca tentando encontrar a solução ideal, esta eficiência depende das informações utilizadas nos parâmetros de configuração. Consequentemente, nem sempre os parâmetros de configuração podem

oferecer a solução mais próxima da esperada. Além disto, a parametrização dos AGs interfere diretamente no processamento da máquina (que executa o algoritmo), ou seja, ela pode influenciar diretamente na utilização dos recursos computacionais disponíveis.

## 2.1 ESTRUTURA BÁSICA DE UM ALGORITMO GENÉTICO

O funcionamento do AG geralmente tem início com a criação aleatória dos indivíduos que irão fazer parte da população inicial. Após o processo de seleção, baseada na qualidade da solução, são escolhidos indivíduos para a fase de reprodução. Esta fase cria novas soluções utilizando os operadores genéticos (cruzamentos e mutações). Assim, a aptidão do indivíduo determina a probabilidade de sua sobrevivência e, assim, a possibilidade que os caracteres do cromossomo façam parte das futuras gerações.

Tradicionalmente, o genótipo de um indivíduo é formado por um vetor binário de tamanho fixo,  $\mathbf{N}$ , e os AGs exploram um espaço de busca formado por  $2^{\mathbf{N}}$  pontos. Esses pontos formam a população de maneira aleatória, a não ser que exista uma heurística para gerar boas soluções para o domínio. Mesmo assim, uma parte da população é gerada aleatoriamente para assegurar que exista diversidade nas soluções (CANTÚ-PAZ, 1997).

O tamanho da população inicial é muito importante, pois determina se o AG irá encontrar boas soluções, como também influencia no tempo necessário para encontrar a solução. Baseado no conhecimento do problema o usuário deverá escolher o tamanho da população inicial.

A precisão da escolha no tamanho da população refletirá na velocidade na qual se chegará a uma convergência. Quanto maior a população a chance de encontrar a solução aumenta devido à probabilidade da solução desejada estar entre os indivíduos, contudo também haverá um maior consumo de recursos computacionais tornando a execução mais lenta. Outro fato importante é que se a população for muito pequena pode haver uma baixa diversidade de genes tornando difícil a identificação de boas soluções.

Independentemente da população inicial o AG possui uma sequência de passos a serem realizados, assim a sua implementação deve determinar qual a representação ideal para o problema em questão, de forma que este possa ser executado corretamente. Com isto, em situações diferentes alguns operadores se tornam mais adequados do que outros. Na figura 01 é exibido o código clássico de um AG.

```

Seja  $S(t)$  a população de cromossomos na geração  $t$ .

 $t \leftarrow 0$ 
inicializ ar  $S(t)$ 
avaliar  $S(t)$ 
enquanto o critério de parada não for satisfeito faça
     $t \leftarrow t+1$ 
    selecionar  $S(t)$  a partir de  $S(t-1)$ 
    aplicar crossover sobre  $S(t)$ 
    aplicar mutação sobre  $S(t)$ 
    avaliar  $S(t)$ 
fim enquanto

```

Figura 01: Código básico de execução de um AG

Em modo geral, o funcionamento do AG é descrito a seguir:

Especificam-se os parâmetros iniciais (por exemplo, os limites do universo de busca) e define-se a população inicial de indivíduos dentro destes limites. Em seguida verifica-se através da equação de mérito (a aptidão de cada indivíduo). Aplicam-se então os operadores genéticos que modificam a população no intuito de melhorá-la. Este processo iterativo, correspondente às sucessivas gerações. O esquema é repetido até que se obtenha a convergência (baseada em algum critério pré-estabelecido), (conforme ilustrado na Figura 01).

Os operadores genéticos são responsáveis pelo melhoramento da população. Eles estão descritos na próxima seção.

## 2.2 OPERADORES GENÉTICOS

Os operadores genéticos têm como objetivo transformar a população através de sucessivas gerações, buscando melhorar a aptidão dos indivíduos. Estes operadores são necessários para que a população se diversifique e mantenha as características de adaptação adquirida pelas gerações anteriores. Na maior

parte dos casos, os AGs utilizam três operadores: seleção, cruzamento e mutação.

### 2.2.1 SELEÇÃO

O operador de seleção é aplicado logo após a geração da população inicial ou após a criação de uma nova geração. Ele deve simular o mecanismo de seleção natural que atua sobre as espécies biológicas, nas quais os melhores cromossomos da população possuem maior capacidade de reprodução, da mesma forma que os menos aptos também terão menos oportunidade de reprodução.

O fator determinante para a seleção do indivíduo é obtido através do valor de mérito. Quem tiver este valor mais alto terá maiores chances de ser selecionado. O valor é originado a partir de uma expressão matemática de mérito utilizada para medir o quanto uma solução aproxima-se da solução desejada.

Depois de avaliados, alguns indivíduos são selecionados, utilizando o fator mérito, e geram populações intermediárias. Vários são os métodos responsáveis por realizarem a seleção. Abaixo serão descritos os mais comuns:

- Seleção determinística: Os melhores indivíduos de acordo com a função de mérito são selecionados;
- Seleção por torneio: Um grupo de indivíduos da população é escolhido de forma aleatória. Em seguida, aquele com o maior mérito dentre eles, é selecionado (SILVA, 2003);
- Seleção aleatória: Cada indivíduo tem a mesma probabilidade de ser selecionado. Desta forma a seleção de indivíduos não aptos pode ocorrer com maior probabilidade, provocando estagnação;

- Seleção elitista: a preservação das melhores soluções (soluções elite) da geração anterior assegura que as melhores soluções conhecidas até então continuarão na população. Geralmente este tipo de seleção é combinado com outras estratégias de seleção (SILVA, 2003);
- Seleção por roleta giratória: cada indivíduo recebe um valor, que analogamente seria sua porção na roleta. Esse valor é uma proporção entre seu mérito e a soma dos méritos da população. Isso faz com que o indivíduo com maior mérito tenha maior chance de ser escolhido (GOLDBERG, 1989).

### 2.2.2 CRUZAMENTO

O cruzamento, na biologia, é um processo sexuado que envolve dois indivíduos e promove o fenômeno de *crossover*, a troca de fragmentos entre pares de cromossomos. Em AGs, é um processo aleatório que ocorre com uma probabilidade fixa, especificada como um parâmetro de entrada pelo usuário.

Após a seleção e geração da população intermediária, é iniciada a operação, que pode ser visto como a “criação” de uma nova população a partir da população intermediária.

Esta operação permite que novos indivíduos sejam criados a partir do cruzamento de genes entre os cromossomos. Escolhem-se dois indivíduos aleatórios de um grupo já selecionado para formar a próxima geração. A figura 3 ilustra um exemplo de realização do cruzamento, sobre duas cadeias de tamanho  $t$ :

$$\begin{array}{l} A^1 = 0110 | 1111 \\ A^2 = 1110 | 0000 \end{array}$$

Figura 02: 1ª etapa do cruzamento

Existem  $t-1$  possíveis pontos de cruzamento em cadeias de tamanho  $t$ , devido a existência do ponto de cruzamento que representa a cadeia inicial. No exemplo acima um único ponto de cruzamento é escolhido, subdividindo a



cadeia em duas partes com  $t=4$ . Trocando as cadeias delimitadas pelo ponto de cruzamento obtêm-se 2 (duas) novas *strings* (dois novos indivíduos modificados):

$$\begin{aligned} A^{t_1} &= 01100000 \\ A^{t_2} &= 11101111 \end{aligned}$$

Figura 03: 2ª etapa cruzamento

Existem várias formas de cruzamento de genes. No exemplo acima foi usado apenas um ponto de cruzamento, mas podem ser utilizados mais pontos. Quanto mais pontos forem usados, mais exploratória será a busca (CANTÚ-PAZ, 1997).

### 2.2.3 MUTAÇÃO

A operação de mutação permite restaurar a variedade que pode ter sido perdida durante as operações de seleção e cruzamento. Ele altera alguns valores dos indivíduos. Essa modificação se dá pela alteração de alguns *bits* do cromossomo aleatoriamente. A idéia intuitiva por trás deste operador é a criação de diversidade e variabilidade extra na população, sem atrapalhar o progresso já alcançado pelo AG.

Considerando que a codificação utilizada seja a binária, a cardinalidade é 2 (existem somente zero e um no alfabeto). Desta forma pode-se dizer que ocorre a clássica troca dos valores dos alelos selecionados.

O operador de mutação é mais importante nas gerações finais quando a maioria dos indivíduos apresenta uma qualidade similar. Assim evitando a convergência prematura em mínimos locais.

## 2.2.4 PARÂMETROS

Os parâmetros de configuração para a execução dos AGs atuam diretamente sobre o seu comportamento. Eles podem ser escolhidos de forma aleatória ou através da utilização de alguma heurística. Neste último caso, o intuito é verificar a maneira que eles podem influenciar o comportamento do AG. Assim busca-se melhorar a definição de acordo com as necessidades do problema e também dos recursos disponíveis.

Os principais parâmetros utilizados em um AG são geralmente o tamanho da população inicial, as taxas de cruzamento e mutação, a condição de parada:

- **Tamanho da população:** determina o número de elementos simultaneamente considerados dentro do espaço da busca de um AG, ou seja, a quantidade de soluções pertinentes a serem avaliadas. O tamanho da população interfere diretamente no desempenho e na eficiência do AG. Com uma população muito pequena, a cobertura de espaço de busca é reduzida, com isto pode-se não chegar a uma solução ótima. Por outro lado, com uma maior população, há uma representação significativa do espaço de soluções, ou seja, maiores chances de chegar à solução mais adequada, assim evitando uma convergência prematura por falta de possibilidades. AGs com grande população são mais lentos e demandam um grande poder computacional por causa do grande número de combinações a serem testadas.
- **Taxa de Cruzamento:** responsável pela inserção de novos indivíduos na população. Quanto maior for essa taxa, mais rapidamente novas estruturas serão inseridas na população. Isto pode favorecer a perda da variedade genética, já que um número maior de estruturas será recombinado. Com a taxa de cruzamento baixa, o algoritmo pode se tornar lento por causa do grande número de vezes que o cruzamento se repetirá para alcançar o resultado desejado.

- **Taxa de Mutação:** tem a função de introduzir material genético novo na população restaurando alelos que tenham sido perdidos durante o cruzamento ou mesmo durante a mutação. Em linhas gerais são estes os objetivos influenciados por este parâmetro:
  - restaurar a diversidade da população que pode ter sido perdida durante as operações de cruzamento;
  - prevenir busca estagnada, introduzindo características que podem ser essenciais na solução do problema;
  - definir a probabilidade que o conteúdo de um gene de cromossomo seja alterado;
  - prevenir que uma determinada parte deste conteúdo possua sempre o mesmo valor;
  - possibilitar a cobertura de todo o espaço de busca permitido.

Uma situação a ser evitada neste parâmetro é que com uma taxa de mutação muito alta a busca pode se tornar essencialmente aleatória.

- **Condição de Parada:** esse parâmetro é o responsável por interromper a execução do AG. Quando essa condição é atendida, o AG encerra sua busca e retorna a melhor solução da população em questão. Porém é preciso definir o critério que será utilizado como ponto de parada, pois em alguns casos, a condição é satisfeita quando é encontrado o ponto ótimo, porém, na maioria dos casos não temos como afirmar que o ponto encontrado é um ótimo global. Como consequência desse fato, diferentes tipos de critérios são adotados para resolver tal situação. Sendo eles:
  - Número máximo de gerações: encerra a busca quando o número máximo gerações estipulado for alcançado;
  - Tempo máximo de processamento: encerra a busca quando o tempo máximo do processamento estipulado for alcançado;

- Estagnação da população: é observado o comportamento da população durante gerações consecutivas. Se este não for alterado, ou seja, se não acontecer melhorias na aptidão do melhor indivíduo ou na média da população ou a aptidão dos indivíduos estiverem parecidas, a busca é encerrada.

### 3 ALGORITMOS GENÉTICOS PARALELOS

O paralelismo é uma estratégia utilizada em computação para obter-se, mais rapidamente, soluções para problemas grandes e complexos. Para isto é necessário quebrar um problema em partes independentes, de forma que cada elemento de processamento possa executar sua parte do algoritmo simultaneamente com os outros. Atualmente o paralelismo em *hardware* está se tornando mais fácil e mais barato, devido à evolução da tecnologia de chips de computador.

#### 3.1 MODELOS DE ALGORITMOS GENÉTICOS PARALELOS

Os AGs são excelentes candidatos para paralelização, pois têm como princípio básico evoluir uma grande população de indivíduos. Além disto, normalmente requerem um grande tempo de execução.

De acordo com (CANTÚ-PAZ, 1997), algoritmos paralelos seguem a premissa de dividir uma tarefa grande em tarefas menores para que sejam resolvidas simultaneamente. Seguindo esta abordagem de dividir para conquistar, foi criada uma série de modelos capazes de tornar um algoritmo genético mais paralelo do que naturalmente ele já é. Diversos são os modelos de AGs paralelos propostos.

##### 3.1.1 GLOBAL COM UMA ÚNICA POPULAÇÃO

Este modelo de implementação de AGs paralelos constitui a classe mais simples de todas (LINDEN, 2008). Consiste em vários AGs simples, cada um executado em seu processador (processadores distintos), porém todos operando sobre uma única população global.

Neste modelo toda população é concentrada em um ponto único de processamento e os operadores genéticos são aplicados de forma paralela em partes da população que é enviada por este ponto único de processamento. Ou

seja, em uma máquina central (Mestre) é definida toda a população do AG e esta enviaria partes desta população para máquinas posteriores ou secundárias (Escravas), para que nelas sejam executadas as operações genéticas.

Apesar da simplicidade, este modelo possui alguns pontos negativos. O primeiro seria a exigência de uma infra-estrutura de rede capaz de atender e suportar toda a demanda de tráfego de dados que passará por ela, como também deverá gerenciar a comunicação que será realizada entre as máquinas no momento da transferência de indivíduos. Em seguida, nos casos em que a população é muito grande será necessária uma grande quantidade de memória para poder armazená-la, podendo tornar inviável a sua aplicação (CANTÚ-PAZ, 1997).

A figura abaixo ilustra este modelo, nele o mestre armazena toda a população e fornece aos seus escravos parte da população, para que eles realizem as operações genéticas.

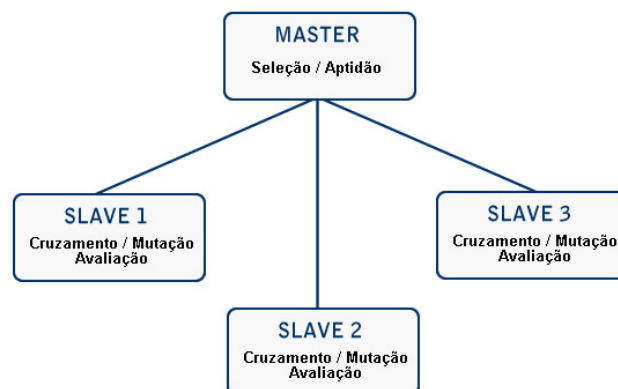


Figura 04: Modelo Global

### 3.1.2 POPULAÇÃO ÚNICA – GRANULARIDADE FINA

Neste modelo uma única população evolui e cada indivíduo é colocado em uma célula de uma grade planar. O processo de seleção e do cruzamento são

aplicado somente entre indivíduos vizinhos na grade (de acordo com a topologia definida).

Este modelo é mais adequado a arquiteturas SIMD (*Single Instruction Multiple Data*) paralelas, que se refere a um conjunto de operações para manipulação eficiente de uma grande quantidade de dados em paralelo, usando um processador vetorial ou um processador matricial.

O modelo de granularidade fina é o intermediário entre o modelo global e o modelo em ilhas que será apresentado posteriormente.

A figura 05 representa a ilustração do modelo de granularidade fina, nele cada ponto são indivíduos de população que está distribuída por todo o espaço.

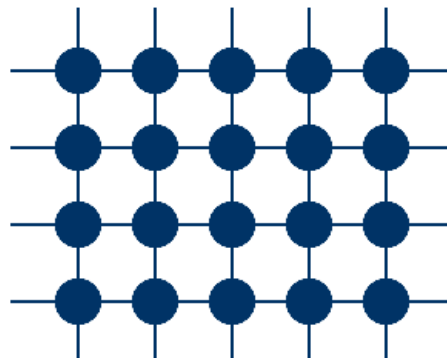


Figura 05: Modelo de Granularidade Fina

### 3.1.3 GRANULARIDADE GROSSA

No modelo de granularidade grossa são criadas várias sub-populações e estas evoluem isoladamente e em paralelo. A partir do critério utilizado, os melhores indivíduos de cada população são enviados para outras sub-populações. Assim, a competição não acontece apenas com os indivíduos de sua respectiva população.

Este modelo é mais adequado às arquiteturas MIMD (*Multiple Instruction Multiple Data*). Sua característica é a execução simultânea de múltiplos fluxos de instruções. Essa capacidade deve-se ao fato de que são construídas a partir

de vários processadores operando de forma cooperativa ou concorrente, na execução de um ou vários aplicativos. Essa definição deixa margem para que várias topologias de máquinas paralelas e de redes de computadores sejam enquadradas como MIMD.

A figura 06 ilustra o modelo de granularidade grossa.

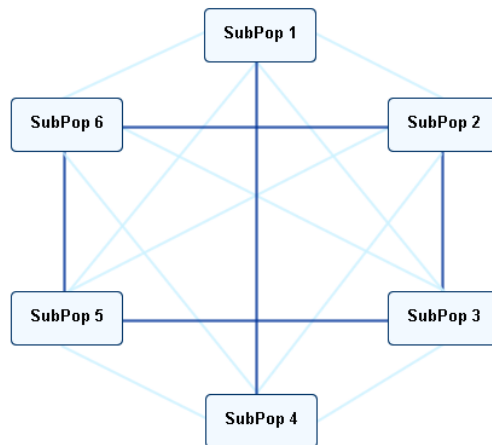


Figura 06: Modelo de Granularidade Grossa

### 3.1.4 ALGORITMOS GENÉTICOS EM ILHAS DE EVOLUÇÃO

A abordagem teórica da seleção natural proposta por Charles Darwin é o pilar da Teoria Moderna da Evolução. Nela Darwin explica que os organismos mais bem adaptados ao meio têm maiores chances de sobrevivência do que os menos adaptados. Os menos adaptados tendem a não deixar um número maior de descendentes, o que ocasionará na sua extinção ao longo do tempo, já que as características favoráveis são hereditárias e passarão para os seus novos descendentes.

Entretanto, foi verificado que em determinadas regiões existem espécies que evoluem de forma isolada para tentar se adaptar a um ambiente isolado, com características específicas. Este processo gera as chamadas espécies endêmicas, ou seja, elas possuem determinadas características que só são encontradas naquele ambiente. Como exemplo, pode-se citar as tartarugas



gigantes de Galápagos, que são espécies de tartarugas que só existem nesta região (DARWIN, 1875).

Seguindo estes princípios e fazendo mais uma vez analogia às soluções encontradas pela natureza, os AGs em ilhas de evolução tem sua definição e implementação fundamentada. Utilizam a premissa de que cada nodo de processamento é enxergado como uma ilha, que contém uma população de indivíduos evoluindo isoladamente a cada geração. No resto do tempo em que as ilhas não estão se comunicando, as mesmas estão trabalhando para encontrar soluções melhores que as existentes.

Um detalhe interessante neste modelo está relacionado ao critério de parada escolhido. Ele deve ser baseado em uma condição que envolva todas as ilhas que compõem o algoritmo genético, evitando que em algum momento, uma ilha com maior capacidade computacional chegue ao critério de parada antes que as demais. Podendo assim deixar alguma ilha parte do tempo ociosa (CANTÚ-PAZ, 1997).

Para facilitar a compreensão do que foi descrito acima, a figura 07 ilustra o modelo de ilhas evolutivas. Na imagem os círculos nas extremidades representam as ilhas e o que está contido em seu interior são os indivíduos que contemplam a sua população. Em cada ilha ocorre o processo de geração de novos indivíduos e de acordo com os parâmetros escolhidos os melhores da população migram para as ilhas vizinhas, construindo um ciclo de evolução e estabelecendo uma comunicação entre todas as ilhas.

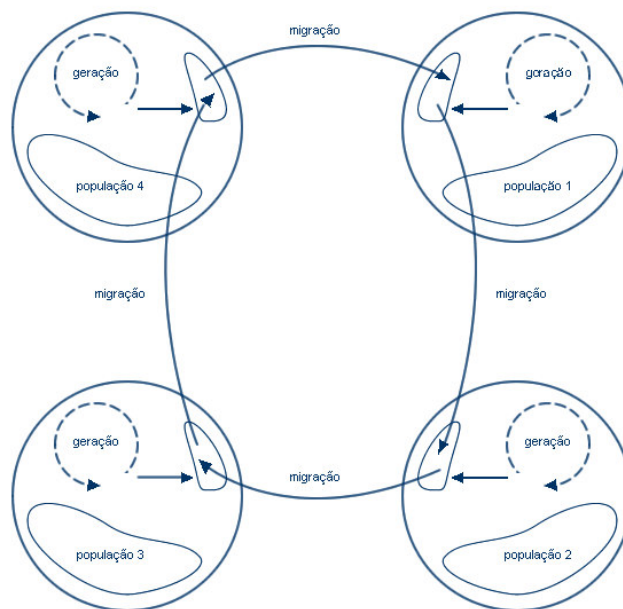


Figura 07: Ilhas de Evolução

Os parâmetros e operadores utilizados em AGs em ilha de evolução são similares aos AGs centralizados diferenciando apenas em alguns parâmetros.

Neste modelo são acrescentados alguns novos parâmetros, um deles é a taxa de migração que serve para indicar a periodicidade com que os indivíduos de uma determinada ilha migrem para as ilhas vizinhas. Outro, seria a quantidade de indivíduos que migram, durante o processo de migração. Por fim, um terceiro parâmetro, indica qual ilha o indivíduo irá durante o momento da migração.

Devido a grande quantidade de comunicação que é feita neste modelo, a frequência com que os indivíduos migram de uma ilha para outra não poderá ser alta, o que limita a quantidade de vezes que o processo de migração pode ser executado. Assim, a migração deve ser executada quando houver necessidade de uma renovação de indivíduos numa ilha.

## 4 GERENCIAMENTO DE AGS DISTRIBUÍDOS

Os sistemas distribuídos (SD) têm como principal fator de motivação a cooperação e o compartilhamento de recursos (componentes de *hardware*, e de *software*). Eles permitem que os componentes se tornem distribuídos de máneira útil.

Para a utilização de um sistema distribuído é necessário um maior esforço para o gerenciamento do sistema. Já que os dados são manipulados em diferentes componentes do sistema. Nesta seção serão abordadas questões pertinentes aos SD e a utilização de AGs em sistemas distribuídos.

### 4.1 SISTEMAS DISTRIBUÍDOS

Segundo Colouris (2007), um sistema é distribuído quando os seus componentes (de *hardware* e *software*) são interligados em uma rede de computadores, eles se comunicam e coordenam suas ações através de mensagens.

Os sistemas distribuídos se diferem dos tradicionais, basicamente devido às seguintes características:

- **Compartilhamento de recursos**

Permite que recursos possam ser compartilhados. Estes recursos podem ser componentes de *hardware*, tais como impressoras, discos, *scanners*, ou componentes de *software*, tais como banco de dados, arquivos e outros objetos de dados.

O compartilhamento dos recursos é oferecido aos usuários através de sistemas. Os sistemas são responsáveis pela comunicação, troca e apresentação dos dados, entre os componentes.

Com isto os recursos presentes fisicamente em um computador, podem ser acessados por outros através da comunicação. Para um compartilhamento eficaz, cada recurso deve ser gerenciado através de um programa que ofereça uma interface de comunicação, tornando possível a manipulação, acesso e atualização de forma confiável e consistente.

- **Concorrência**

A concorrência ocorre quando dois processos são executados simultaneamente, em especial, quando eles disputam acesso a algum recurso compartilhado.

Nos sistemas distribuídos é comum os processos disputarem acesso a algum recurso. Isto permite a execução de vários programas em paralelo sem afetar o desempenho do sistema.

Por outro lado devem ser observados e tratados aspectos relacionados à sincronização. Assim, os acessos característicos de concorrência devem ser realizados em sincronia.

- **Escalabilidade**

Os sistemas distribuídos são capazes de funcionar em diversas escalas eficientemente. Ele pode ser composto por apenas duas *workstations* (estações de trabalhos) e um servidor de arquivos, ou até centenas de delas e vários servidores de arquivos, de impressão, dentre outros, permitindo que os recursos sejam compartilhados na diversidade de estações que fazem parte dos sistemas. Esta característica visa garantir que o sistema e a aplicação não necessite de mudanças quando houver um aumento da escala do sistema.

- **Tolerância a falhas**

As falhas (em *hardware* ou *software*) podem gerar resultados incorretos, como também podem interromper atividade antes da sua conclusão. O projeto de sistemas tolerante a falhas podem seguir duas abordagens:

redundância de *hardware* (uso de componentes redundantes ou em excesso) e restabelecimento de *software* (programas que recuperam as falhas ocorridas).

Um sistema distribuído não se torna indisponível quando ocorre uma falha de *hardware*, como ocorre nos sistemas centralizados. Basta que ocorra a mudança para outra *workstation*, onde será possível continuar com as atividades a partir do ponto o qual já estava antes de ocorrer a falha.

- **Transparência**

A transparência permite que um sistema distribuído seja visualizado como um sistema centralizado, portanto, é necessário abstrair os componentes e a comunicação entre eles.

Diversas são as formas de transparência, elas são:

- **Acesso:** esconde diferenças na representação de dados e como um recurso é acessado;
- **Localização:** esconde onde o recurso está localizado;
- **Migração:** esconde que um recurso pode ser mover para outra localização;
- **Relocação:** esconde que um recurso possa ser movido para outra localização enquanto está sendo usado;
- **Replicação:** esconde que vários recursos podem ser compartilhados por vários usuários concorrentes;
- **Falha:** esconde a falha e recuperação de um recurso
- **Persistência:** esconde que um recurso (*software*) está em memória ou em disco.

## 4.2 ALGORITMOS GENÉTICOS EM SISTEMAS DISTRIBUÍDOS

Os sistemas distribuídos e os sistemas paralelos estão sendo bastante utilizados, de forma independente e também em conjunto. Ambos têm vantagens que favorecem ao usuário o monitoramento da aplicação e distribuição das tarefas durante o seu processo de execução.

O paralelismo tem como objetivo melhorar o desempenho e disponibilidade da aplicação durante o seu funcionamento. Ele é responsável por distribuir a execução do processamento dos algoritmos, o que muitas vezes é necessário, devido ao fato dos aplicativos requererem, ocasionalmente, uma grande disponibilidade dos recursos computacionais.

Conquanto, a distribuição possui diversas vantagens, ela traz consigo uma maior necessidade de gerenciamento. Isto acontece, devido ao fato das informações estarem dispersas em diferentes componentes da rede, embora os SD devam disponibilizá-las para os usuários, de forma similar ao que ocorre.

Em AGs distribuídos esta premissa continua aplicável. Portanto para a utilização destas tecnologias em conjunto é necessário que seja montada uma infra-estrutura que permita ao usuário manipular as informações pertinentes aos AGs de forma transparente, como se estivesse utilizando um sistema centralizado.

O próximo capítulo apresenta a ferramenta AGMAX, responsável por garantir a infra-estrutura necessária para o gerenciamento da execução de AGs em Ilhas de Evolução.

## 5 AGMAX – FERRAMENTA DE GERENCIAMENTO DE AGS DISTRIBUÍDOS EM ILHAS DE EVOLUÇÃO

Em função da necessidade do gerenciamento da execução de AGs distribuídos, foi desenvolvida a ferramenta AGMAX. Portanto, ela permite controlar e gerenciar as estações responsáveis pela execução dos AGs em ilhas de evolução.

Neste capítulo será apresentada a ferramenta. Em particular, serão descritas: a arquitetura do sistema, o modelo de componentes, a apresentação das funcionalidades, o protocolo de comunicação, e a interface da aplicação.

### 5.1 ARQUITETURA DO SISTEMA

O modelo arquitetural adotado para a ferramenta é baseada no tipo Cliente/Servidor.

Os módulos Cliente e Servidor compõe este modelo de arquitetura, eles são interligados por uma rede de computadores. O cliente (Gerenciador AG), emite um pedido para o Servidor (Estação AG), que oferece serviços. Estes serviços são responsáveis por solicitar ações para manipulação do AG. A figura 08 representa o esquema arquitetural da AGMax.

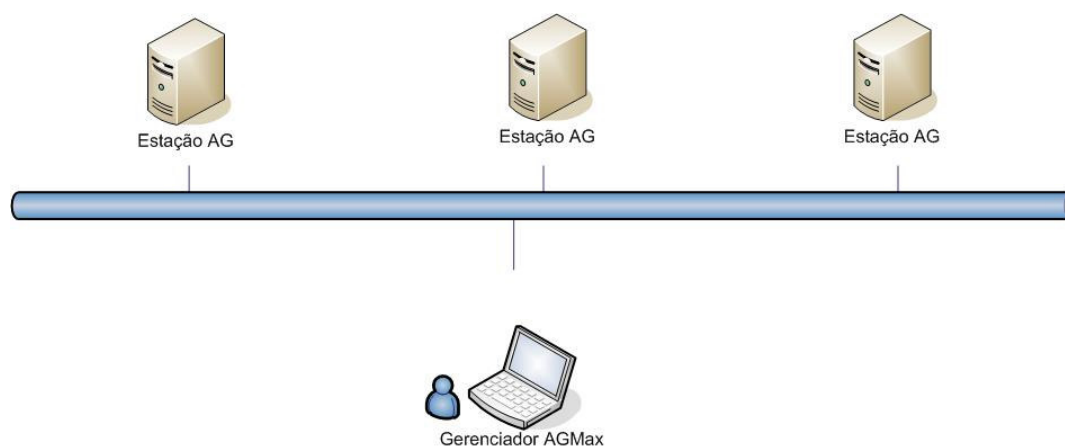


Figura 08: Esquema da Arquitetura AGMAX

A arquitetura permite que novas Estações AG sejam adicionadas a qualquer momento na rede. Porém para que sejam gerenciadas é necessário que o Gerenciador AGMax realize a busca das estações disponível na rede.

As **Estações AG** são responsáveis pela execução dos AGs, já o **Gerenciador AGMax** é responsável pelo controle das estações, eles comandam as ações e realizam a coleta das informações que fornecidas pelas estações.

O Gerenciador AGMAX comunica-se com as estações AGs através do protocolo UDP (User Datagram Protocol). O protocolo UDP foi escolhido para permitir que o processo de comunicação do Gerenciador para as Estações possa ser realizado sem que o primeiro conheça previamente os demais, dado que o UDP é um protocolo não orientado à conexão. Contudo, para garantir que todas as estações ativas possam ser alcançadas pelas mensagens disparadas pelo Gerenciador fixou-se a porta 7072 como padrão para o protocolo AGMAX.

O protocolo AGMAX utiliza na camada de aplicação atributos padrões que permitem o cliente enviar mensagens ao servidor durante a solicitação dos serviços. Por sua vez, os servidores realizam as operações e encaminham mensagens aos clientes.

As mensagens enviadas utilizam a estrutura típica de parâmetros da URL (*Uniform Resource Locator*), consistindo de pares nome-valor separados pelo *caracter* '&'. Estas URLs estão definidas em duas classes do projeto, elas são: *ComponenteUdpUrl* e *GerenciadorUdpUrl*. Através dos métodos presentes é possível montar as URLs. Detalhes do protocolo e das classes estão apresentados no Anexo A, deste documento.

### 5.1.1 COMPONENTES AGMAX

Um componente de software é uma unidade independente. Ele permite a utilização juntamente com outros componentes, formando assim sistemas mais complexos e completos.



O AGMAX por exemplo é um componente de software que foi desenvolvido a partir da utilização de outros dois componentes de software o *JGA* e o *JGNet*. Consequentemente existe uma dependência do AGMAX para com estes outros dois componentes.

O JGA é o componente responsável pela execução dos algoritmos, embora ele suporte a execução de AGs na mesma estação. O JGNet é responsável pela migração de um indivíduo de uma determinada ilha para outras ilhas, compondo os AGs em ilhas de evolução. A figura abaixo ilustra os componentes da AGMAX.

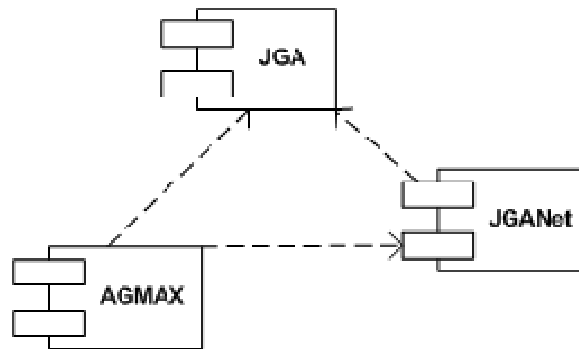


Figura 09: Componentes de software do AGMAX

O AGMAX é o responsável pelo gerenciamento das estações que executam os AGs. Ele prôve uma interface que permite o usuário interagir com os AGs distribuídos na rede como se estivesse em um sistema centralizado.

A próxima seção descreve os requisitos funcionais da aplicação.

### 5.1.2 REQUISITOS FUNCIONAIS

No contexto da engenharia do software, requisito, é uma condição ou capacidade na qual o sistema tem de estar de acordo. Faz parte de uma etapa do processo de análise de sistemas engloba as atividades que auxiliam a elaboração de documentos de requisitos e sua manutenção ao longo do tempo.

Existem vários tipos de requisitos. Porém eles se dividem basicamente em duas categorias, os requisitos funcionais e os requisitos não funcionais. Os requisitos funcionais especificam ações que um sistema deve ser capaz de executar, de forma completa e consistente. Os requisitos não funcionais representam as ações que não são diretamente relacionadas às operações do negócio (ex: utilidade, confiança, desempenho, suporte e escalabilidade). Nesta seção serão apenas apresentados os requisitos funcionais.

Para a descrição dos requisitos da ferramenta foi utilizado o modelo de Caso de Uso. O diagrama descreve como diferentes tipos de usuários interagem com o sistema para resolver os problemas. Casos de Uso descrevem uma sequência de ações que representam um cenário principal (perfeito) e cenários alternativos, assim demonstrando o comportamento do sistema através das interações com os usuários. A figura 10 representa o diagrama de caso de uso do AGMAX.

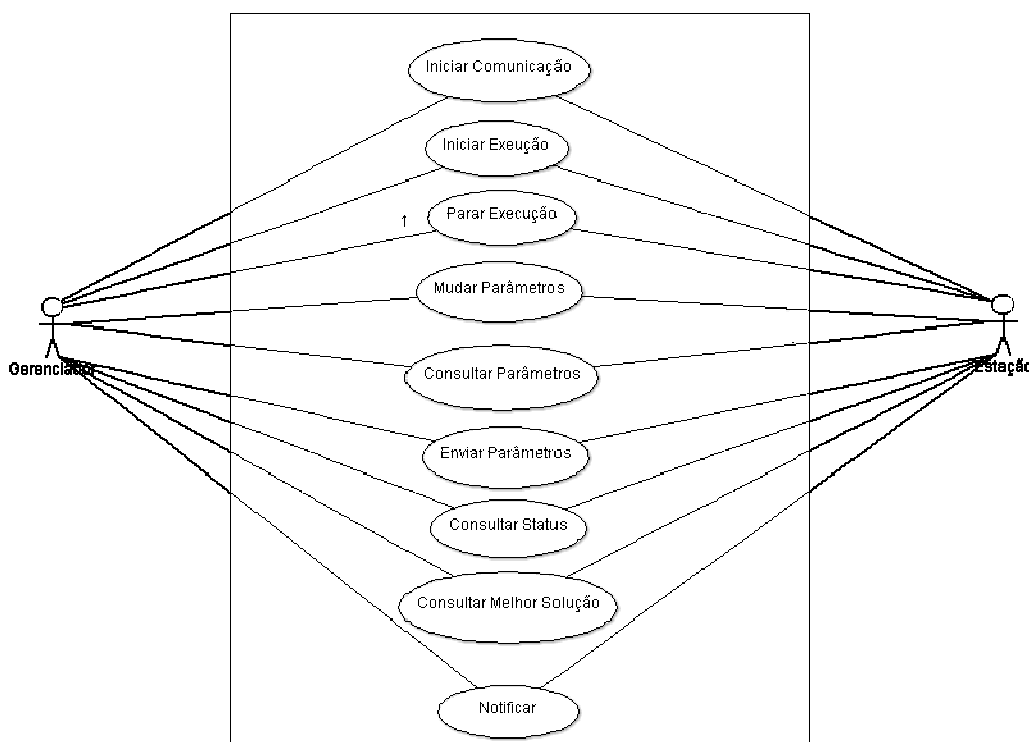


Figura 10: Modelo de Caso de Uso do Sistema

As principais funcionalidades da ferramenta são descritas a seguir:

- **Início da comunicação:** inicia a comunicação do aplicativo gerenciador, com os aplicativos estações.
- **Início da execução:** inicia a execução do AG nas estações.
- **Parada da execução:** interrompe a execução do AG que está sendo executado nas estações.
- **Mudança de parâmetros:** mudar o cenário dos dados de configuração dos parâmetros do AG.
- **Consulta de parâmetros:** consulta o valor dos parâmetros que estão configurados para execução do AG.
- **Envio de parâmetros:** envia o valor dos parâmetros que estão configurados para execução do AG.
- **Consulta de status:** consulta o status atual dos AGs.
- **Consulta melhor solução:** consultar a melhor solução atual do AG.

Outro tipo de requisito considerado importante seria o de interação e usabilidade, estes são um dos principais fatores de aceitação das ferramentas, pois se não forem utilizados componentes de interface adequados, o sistema pode ser difícil de ser manipulado e conseqüentemente não ter a aceitação por parte dos usuários. A próxima seção apresenta a interface e descreve as suas habilidades no processo de interação e usabilidade.

## 5.2 INTERAÇÃO E USABILIDADE

A ferramenta AGMAX foi desenvolvida de forma a permitir que o usuário interagir com os AGs distribuídos pela rede de forma transparente. Apesar de possuir uma interface simples e com poucos recursos, ela foi desenvolvida utilizando os princípios da interação e usabilidade (MINASI, 1994).

A interface gráfica foi desenvolvida para com o intuito de oferecer uma usabilidade simples e eficiente. Ela apresenta componentes visuais básicos que de forma eficiente disponibiliza o usuário interagir com as funcionalidades do sistema. É a partir desta tela que o usuário gerencia as estações. A figura 11 ilustra a interface gráfica da ferramenta AGMAX.

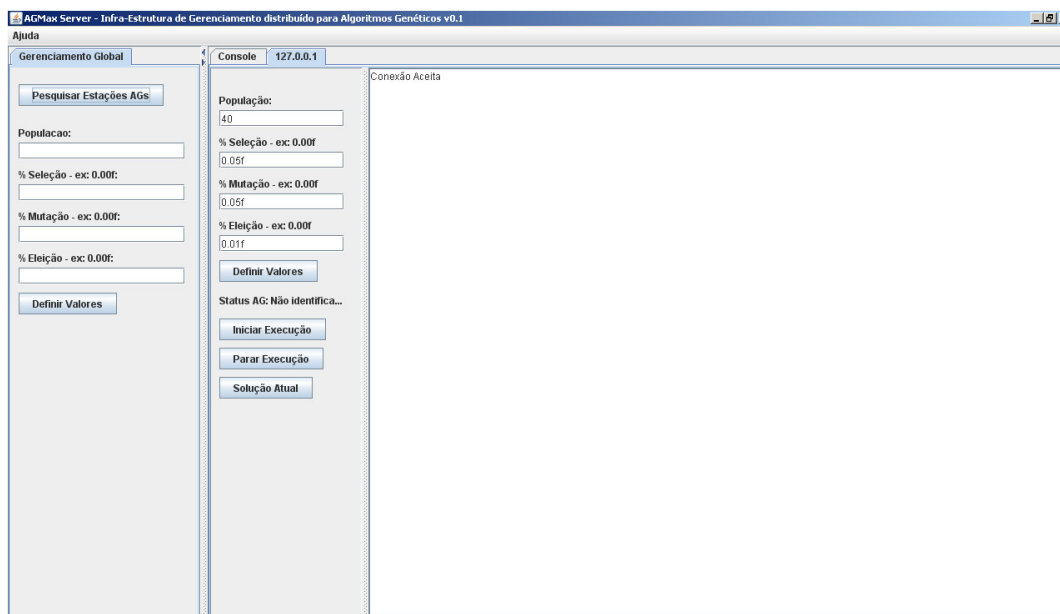
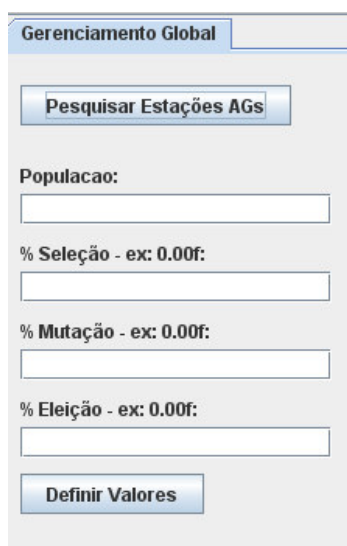


Figura 11: Interface Gráfica

A AGMAX serve como um canal de comunicação do gerenciador com as estações, isto é, o gerenciador é responsável por invocar as operações e por todo o gerenciamento, já as estações são responsáveis pela execução do AG, como também manter o gerenciador informado dos acontecimentos.

Através desta tela também é possível manipular uma ou mais estações de maneira independente ou global. A manipulação independente permite gerenciar as estações AGs individualmente, já a global permite que sejam gerenciadas todas as estações ao mesmo tempo.

A manipulação global da ferramenta é gerenciada conforme ilustra a Figura 12, um painel de propriedades oferece opções de utilização para o usuário. Através deste painel o usuário pode chamar os serviços, permitindo assim a realização das ações em todas as estações, sendo que para isto é necessário que a conexão já tenha sido iniciada.



Gerenciamento Global

Pesquisar Estações AGs

Populacao:

% Seleção - ex: 0.00f:

% Mutação - ex: 0.00f:

% Eleição - ex: 0.00f:

Definir Valores

Figura 12: Painel Global

A manipulação individual da ferramenta é gerenciada conforme ilustra a Figura 13, um painel de propriedades oferece opções de utilização para o usuário. Através deste painel o usuário pode “disparar ações” que são realizadas pela estação.



Console 127.0.0.1

Conexão Aceita

População:

40

% Seleção - ex: 0.00f

0.05f

% Mutação - ex: 0.00f

0.05f

% Eleição - ex: 0.00f

0.01f

Definir Valores

Status AG: Não identifica...

Iniciar Execução

Parar Execução

Solução Atual

Figura 13: Painel Individual

Vale ressaltar que as estações são responsáveis pela execução do AG, isto é, são responsáveis pela execução das operações de seleção, cruzamento e mutação. Adicionalmente as estações mantêm a comunicação com a ferramenta gerenciadora, para informar a situação e ou iniciar alguma operação que tenha sido instruída.

Deve-se destacar, que a ferramenta dá acesso a todas as possíveis operações que são disponibilizadas pelos AGs, de forma que o usuário de forma centralizada e transparente, monitora e controla os AGs que são executados nas estações.

## CONSIDERAÇÕES FINAIS

Os algoritmos genéticos são úteis na resolução de problemas computacionais, para os quais não se conhece soluções computacionalmente eficientes. Contudo durante o processo de execução de um AG, pode ser necessário o uso elevado de recursos computacionais.

Técnicas de paralelização e distribuição têm sido desenvolvidas para diminuir os problemas encontrados durante o processo de execução dos AGs, porém para que este processo seja mantido e monitorado de forma adequada é necessário que haja o gerenciamento do processo de execução.

Visando oferecer um ambiente distribuído que permita o usuário gerenciar diferentes máquinas (em uma rede de computadores) que executam AGs em Ilhas de Evolução, foi desenvolvida a ferramenta AGMAX.

A ferramenta permite ao usuário gerenciar diferentes estações de AGs, a partir de um único ponto. Com ela é possível identificar em quais estações estão sendo executados os AGs, além de se obter um maior controle sobre tais execuções, através do monitoramento e coleta de informações dos AGs.

Para dar continuidade ao trabalho são sugeridas as seguintes abordagens:

- Estudo de viabilidade de desenvolvimento da ferramenta de gerenciamento na web;
- Estudo da Interface Gráfica, com o intuito oferecer componentes visuais que permitam o usuário interagir com a aplicação de uma forma mais amigável.

## REFERÊNCIAS BIBLIOGRÁFICAS

ÁVILA, Sérgio Luciano, Algoritmos Genéticos Aplicados na Otimização de Antenas refletoras, 2002, Dissertação.

BARCELLOS, João Carlos Holland de, Algoritmos Genéticos Adaptativos : Um estudo comparativo - 2000

CANTÚ-PAZ, Erick. A Survey of Parallel Genetic Algorithms – Department of Computer Science and Illinois Genetic Algorithms Laboratory – University of Illinois, 1997.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. Sistemas Distribuídos – Conceitos e Projeto, Porto Alegre-RS: Bookman Companhia Editorial LTDA, 2007.

FELISSÍSIMO, José Roberto; AVANCINE, Sérgio Luis. Em Busca de uma Metodologia: a Pesquisa-Ação, Cadernos FUNDAP, 1981;

FREITAS, Cherze C.; Guimarães, Priscila R. B.; Neto, Manoel C. M.; Barboza, Frederico J. R., Uma ferramenta Baseada em Algoritmos Genéticos para a Geração de Tabela de Horário Escolar

F. K. Miyazawa e E. C. Xavier, Classes de Complexidade e NP-Completeness, 03 de setembro de 2009

GAREY, M. R.; JOHNSON D. S. Computers and Intractability: A Guide to the Theory of NP Completeness (Series of Books in the Mathematical Sciences), 1979;

GOLDBERG, D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Assiscon-Wesley, 1989.

LARMAN, Craig; Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvedor iterativo 3ª edição, 2007.



LENZ, Processamento SIMD (Single Instruction, Multiple data) – FATEC – São Paulo

LINDEN, Ricardo. Algoritmos genéticos, 2º edição, Rio de Janeiro: Brasport, 2008;

LUTZ, R.R. "Analyzing Software Requirements Errors in Safety-Critical Embedded Systems." Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, New York, NY, December 1993, pp. 126-133.

MINASI, Mark. Segredos de projeto de interface gráfica com o usuário. Tradução Flavio Eduardo Morgado. Rio de Janeiro: Infobook, 1994.

MITCHELL, Melanie, An Introduction To Genetic Algorithms (Series – Complex Adaptive Systems Series), Paperback, 1998.

M. R. Garey e D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences), 15 de janeiro de 1979.

PEREIRA, José Antônio; BORGES, Leonardo Costa; FIGUEREDO, Roberto Boscolo. Avaliação do Comportamento de Algoritmos Genéticos em Ilhas de Evolução Paralela no Problema da Sacola, Faculdade Rui Barbosa, 2008.

POZZA, Osvaldo Antonio e PENEDO, Sergio A Máquina de Turing - 2002

SILVA, Edna Lúcia; MENEZES, Estera Muszkat. Metodologia da Pesquisa e Elaboração de Dissertação, 3º edição revisada e atualizada, 2001.

SIMOES, André Luiz Ribeiro, Metodologia de Trabalho, acessado em <http://www.slideshare.net/alsimoes/scrum>, 04 de novembro de 2009

REIS, Luisa Fernanda R. e AKUTSU, Jorge - Estratégias Operacionais para Sistemas de reservatórios via Algoritmos Genéticos (AGs) – 2002

## ANEXOS A – CLASSES DO PROTOCOLO DE COMUNICAÇÃO

O protocolo de comunicação definido está implementado em duas classes do projeto. A classe `ComponenteUdpUrl` (representada na figura 14), contempla as mensagens que são enviadas pelos clientes e a classe `GerenciadorUdpUrl` (representada na figura 15), contempla as mensagens que são enviadas pelo Gerenciador. A

agmax::management::udp::url::ComponenteUdpUrl
url : StringBuffer <u>POPULACAO_INICIAL : String</u> <u>SELECAO_INICIAL : String</u> <u>MUTACAO_INICIAL : String</u> <u>ELEICAO_INICIAL : String</u>
getUrlStart() : String resultadoExecucao(ga : GANetEngine,paramIdHost : String) : String resultadoExecucao(app : ApplicationAg,paramIdHost : String) : String getUrlExecucaoParada(paramIdHost : String) : String getUrlExecucaoNaoParada(paramIdHost : String) : String resultadoFinalExecucao(app : ApplicationAg,paramIdHost : String) : String fimExecucao(ga : GANetEngine,paramIdHost : String) : String resultadoExecucao(paramIdHost : String) : String getUrlExecucaoIniciada(paramIdHost : String) : String getUrlExecucaoParadaSucesso(paramIdHost : String) : String getUrlExecucaoParadaFalha(paramIdHost : String) : String getUrlConexaoAceita() : String getUrlParametrosAlterados(paramIdHost : String) : String

Figura 14: Classe `ComponenteUdpUrl`

agmax::management::udp::url::GerenciadorUdpUrl
url : StringBuffer
resultadoExecucao(ga : GANetEngine,paramIdHost : String) : String aceitarConexao(idHost : int) : String iniciarExecucao(paramIdHost : int) : String solicitarConexao(paramIdHost : int) : String solicitarSolucao(paramIdHost : int) : String pararExecucao(paramIdHost : int) : String alterarParametros(field : JTextField,idHost : int) : String

Figura 15: Classe `GerenciadorUdpUrl`

## ANEXOS B – CASOS DE USO

### CASO DE USO: INICIAR COMUNICAÇÃO

<b>Número</b>	UC001
<b>Nome</b>	Iniciar Comunicação
<b>Ator(es)</b>	Gerenciador e Estação
<b>Descrição</b>	Este UC tem como objetivo iniciar a comunicação do aplicativo gerenciador, com o aplicativo estação.
<b>Pré-Condições</b>	O usuário configura a lista dos endereços IPs.
<b>Pós-Condições</b>	Não há
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1. O gerenciador envia mensagem(ns) solicitando o início da comunicação para a(s) estação(ões);</li> <li>2. A(s) estação(ões) recebe(m) a mensagem de solicitação;</li> <li>3. A(s) estação(ões) aceita(m) a comunicação;</li> <li>4. A(s) estação(ões) envia(m) mensagem confirmando o início da comunicação.</li> </ol>
<b>Cenário Alternativo</b>	Não há.
<b>Exceções</b>	Não há.
<b>Inclusão (includes)</b>	Não há.
<b>Extensões (extend)</b>	Não há.
<b>Regra de Negócio</b>	

### CASO DE USO: INICIAR EXECUÇÃO

<b>Número</b>	UC002
<b>Nome</b>	Iniciar Execução
<b>Ator(es)</b>	Gerenciador e Estação
<b>Descrição</b>	Este UC tem como objetivo iniciar a execução do

	algoritmo genético, na(s) estação(ões).
<b>Pré-Condições</b>	Conexão entre o gerenciador e a(s) estação(ões) já tenha sido iniciada(s).
<b>Pós-Condições</b>	1. Ao terminar a execução do AG a estação envia mensagens para o gerenciador [Inclui Caso de Uso Enviar Status] [ Inclui Caso de Uso Enviar Melhor Solução].
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1. O Gerenciador envia mensagem(ns) solicitando o início da execução de um AG para uma ou todas estações;</li> <li>2. A(s) estação(ões) recebe(m) a mensagem de solicitação</li> <li>3. A(s) estação(ões) inicia(m) a execução</li> <li>4. A(s) estação(ões) envia(m) mensagens confirmando o início da conexão [Inclui Caso de Uso Enviar Status] [ Inclui Caso de Uso Enviar Melhor Solução].</li> </ol>
<b>Cenário Alternativo</b>	
<b>Exceções</b>	
<b>Inclusão (includes)</b>	UC006 – ENVIAR STATUS  UC007 – ENVIAR MELHOR SOLUÇÃO
<b>Extensões (extend)</b>	
<b>Regra de Negócio</b>	

### CASO DE USO: PARAR EXECUÇÃO

<b>Número</b>	UC003
<b>Nome</b>	Parar Execução
<b>Ator(es)</b>	Gerenciador e Estação
<b>Descrição</b>	Este UC tem como objetivo parar a execução do AG,

	que está sendo executado na estação.
<b>Pré-Condições</b>	AG esteja sendo executado por pelo menos uma estação.
<b>Pós-Condições</b>	Não há.
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1. O Gerenciador envia mensagem(ns) solicitando que a execução do AG seja interrompida para a(s) estação(ões);</li> <li>2. A(s) estação(ões) recebe a mensagem de solicitação;</li> <li>3. A(s) estação(ões) interrompe(m) a execução do AG;</li> <li>4. A(s) estação(ões) enviam mensagem confirmando a interrupção do AG.</li> </ol>
<b>Cenário Alternativo</b>	
<b>Exceções</b>	
<b>Inclusão (includes)</b>	
<b>Extensões (extend)</b>	
<b>Regra de Negócio</b>	

## CASO DE USO: MUDAR PARÂMETROS

<b>Número</b>	UC004
<b>Nome</b>	Mudar Parâmetros
<b>Ator(es)</b>	Gerenciador e Estação
<b>Descrição</b>	Este UC tem como objetivo mudar o cenário dos parâmetros de configuração do AG.
<b>Pré-Condições</b>	
<b>Pós-Condições</b>	Não há.
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1. O Gerenciador envia mensagem(ns) informando os parâmetros de configuração do AG, solicitando a alteração;</li> <li>2. A(s) estação(ões) recebe(m) a mensagem de</li> </ol>

	<p>solicitação;</p> <p>3. A(s) estação(ões) atualiza os parâmetros de configuração do AG;</p> <p>4. A(s) estação(ões) envia(m) uma mensagem para o gerenciados, informando a alteração dos parâmetros de configuração do AG.</p>
<b>Cenário Alternativo</b>	
<b>Exceções</b>	
<b>Inclusão (includes)</b>	
<b>Extensões (extend)</b>	
<b>Regra de Negócio</b>	

## CASO DE USO: ENVIAR PARÂMETROS

<b>Número</b>	UC005
<b>Nome</b>	Enviar Parâmetros
<b>Ator(es)</b>	Gerenciador e Estação
<b>Descrição</b>	Este UC tem como objetivo enviar o valor dos parâmetros que estão configurados pelo AG.
<b>Pré-Condições</b>	
<b>Pós-Condições</b>	Não há.
<b>Cenário Principal</b>	<p>1. O Gerenciador envia mensagem(ns) solicitando os parâmetros que estão configurados no(s) AG(s);</p> <p>2. A(s) estação(ões) recebem mensagem de solicitação;</p> <p>3. A(s) estação(ões) envia(m) mensagem com os parâmetros configurados.</p>
<b>Cenário Alternativo</b>	
<b>Exceções</b>	
<b>Inclusão (includes)</b>	
<b>Extensões (extend)</b>	

<b>Regra de Negócio</b>	
-------------------------	--

## CASO DE USO: NOTIFICAR

<b>Número</b>	UC006
<b>Nome</b>	Notificar
<b>Ator(es)</b>	Gerenciador e Estação
<b>Descrição</b>	Este UC tem como objetivo enviar o status e a melhor solução atual do AG.
<b>Pré-Condições</b>	Não há.
<b>Pós-Condições</b>	Não há.
<b>Cenário Principal</b>	<ol style="list-style-type: none"> <li>1. O Gerenciador envia mensagem(ns) de solicitação;</li> <li>2. A(s) estação(ões) recebem mensagem de solicitação;</li> <li>3. A(s) estação(ões) envia(m) mensagem informando o seu status e a melhor solução atual.</li> </ol>
<b>Cenário Alternativo</b>	<ol style="list-style-type: none"> <li>1. O AG em execução é concluído;</li> <li>2. A estação envia mensagem do status e mensagem atual para o gerenciador ao término da execução do AG.</li> </ol>
<b>Exceções</b>	
<b>Inclusão (includes)</b>	
<b>Extensões (extend)</b>	
<b>Regra de Negócio</b>	