SANDRA RITA DE OLIVEIRA

O ALGORITMO GENÉTICO NA OTIMIZAÇÃO DO PROJETO ESTRUTURAL DE EMBARCAÇÕES.

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Mestre em Engenharia.

São Paulo
1998

SANDRA RITA DE OLIVEIRA

O ALGORITMO GENÉTICO NA OTIMIZAÇÃO DO PROJETO ESTRUTURAL DE EMBARCAÇÕES.

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Mestre em Engenharia.

Área de Concentração:
Engenharia Naval e Oceânica

Orientador:
Oscar Brito Augusto

São Paulo
1998

À minha mãe e minha irmã, Nanci, que estiveram presentes em todos os momentos da minha vida me incentivando, sempre com muito amor.

AGRADECIMENTOS

Ao amigo e orientador Prof. Dr. Oscar Brito Augusto pela confiança que depositou em mim e em meu trabalho. Seu constante incentivo, não somente durante o período da pesquisa, mas desde os tempos da graduação, quando eu o conheci como um excelente mestre, veio de encontro às minhas aspirações, colocando-me nos caminhos que eu havia almejado.

À minha família que me apoiou e orientou da melhor forma possível para que eu nunca desistisse perante os obstáculos que viessem a surgir e pudesse terminar mais esta etapa da minha vida.

À FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo) pelo apoio concedido a este trabalho de pesquisa.

A todos aqueles, que em algum momento, me incentivaram através de uma palavra amiga ou de um conselho oportuno.

À Deus que sempre colocou inúmeras oportunidades no meu caminho e que me deu forças para poder aproveitá-las.

Índice

Índice de Figuras

Índice de Tabelas

Nomenclatura

A_j	*string j*, elemento da geração i (j = 1, 2, ..., n) (i = 1, 2, ..., n).
A_j'	*string j*, produzido após reprodução e crossover, da geração i + 1.
AG	Algoritmo Genético.
$A(t)$	população de *strings* A_j no tempo (ou geração) t.
$\overline{f}$	média das performances de toda a população.
f_i	performance da *string i*.
f_t	somatória das performances de todos os membros da população.
fac_i	soma acumulada da performance dos i primeiros membros da população.
$f(H)$	média das performances das *strings* representantes do schema H no tempo t.
$f(\mathbf{X})$	função de mérito ou função objetivo.
$g_j(\mathbf{X})$	*j-ésima* função de restrição de desigualdade.
H	schema formado a partir dos elementos 0, 1, *.
$h_j(\mathbf{X})$	*j-ésima* função de restrição homogênea.
l	comprimento da string
$m(H, t)$	número de exemplos do schema H, contidos na população $A(t)$.
n	número de variáveis de projeto.
n_g	número de (funções) restrições não homogêneas ou de desigualdades.
n_h	número de (funções) restrições homogêneas.
o(H)	ordem do schema H.
p_c	probabilidade de crossover.

p_d	probabilidade de destruição do schema H_i.
p_i	probabilidade de seleção da string A_i.
p_m	probabilidade de mutação.
p_s	probabilidade de sobrevivência do schema H_i.
$p_c[i]$	probabilidade de *crossover* da *string* representativa da variável *i*.
$p_m[i]$	probabilidade de *mutação* da *string* representativa da variável *i*.
p_{co} , p_{mo}	valores iniciais do controle de probabilidades.
p_{1m}, p_{2m}	penalização da *m-ésima* função de restrição.
$p(\mathbf{X},r_k)$	função de penalização na *k-ésima* busca seqüencial.
r_k	fator de respostal na *k-ésima* busca seqüencial.
R	geração atual/número máximo de gerações.
V	conjunto binário formado pelos elementos 0 e 1.
V+	conjunto ternário formado pelos elementos 0, 1 e *.
x_i	*i-ésima* variável de projeto.
$x_i^{(k)}$	*i-ésima* variável de projeto contida no *k-ésimo* vetor **X**.
X	vetor contendo as variáveis de projeto.
α	coeficiente de reflexão.
α_m, β_m	parâmetros de penalização da *m-ésima* função de restrição
δ	deslocamento
$\delta(H)$	comprimento significativo do schema H.
ε, ε_i	tolerância ou número comparativamente pequeno.
$\phi(\mathbf{X},r_k)$	função de mérito penalizada ou função de resposta.
σ	tensão normal
τ	tensão de cisalhamento

Glossário

Bit	é cada posição dentro de uma string, que em um código binário é ocupado por um 1 ou um 0. Da mesma forma que os cromossomos são formados por genes, as strings são formadas por bits. Nos sistemas naturais, um gene tem a capacidade de armazenar e transportar a informação genética para um caráter biológico (qualquer aspecto anatômico-fisiológico de um ser vivo). Um bit, armazena e transporta informações de uma variável de uma geração a outra no processo de otimização.
Building blocks	schemata de curto comprimento significativo e alta performance.
Crossover	cruzamento entre duas strings, escolhidas aleatóriamente entre as strings de uma população num tempo (ou geração) t. É uma recombinação de parte de cada string (substring) do par.
Defining Length	é o comprimento significativo de uma string, ou seja, a distância entre a primeira e a última posição específica da string.
Mating Pool	são todos os membros (strings) de uma população, que após a reprodução, são escolhidos dois a dois para que tenham parte de seu material (bits) recombinado pelo crossover.
Mutação	é uma alteração aleatória ocasional do valor da posição da string. Em um código binário significa mudar o valor de um 1 para um 0 e vice - versa.
Reprodução	é um processo no qual strings individuais são copiadas de acordo com os valores da função objetivo.

Schema	subconjunto de strings com similaridades em certas posições. Um schema é criado a partir de um conjunto ternário, V+ = {0, 1, *}. Um schema é igual a uma particular string se em toda posição no schema um 1 é igual a um 1, um 0 é igual a um 0, ou um * é igual a um 1 ou um 0, de forma indiferente.
Schemata	plural de schema.
Schema order	é a ordem do schema. Número de posições fixas (em um conjunto binário, o número de 1's e 0's) presentes no schema.
String	é a estrutura que representa uma variável de projeto, decodificada num sistema binário. Em analogia com a genética natural, tem a função de uma cromossomo que transporta informações genéticas de uma geração para outra.

Resumo

Neste trabalho apresenta-se o Algoritmo Genético (AG), um algoritmo de programação matemática não linear, a ser utilizado em projetos otimizados de estruturas navais e oceânicas. O algoritmo foi aplicado ao projeto de uma Tampa de Escotilha e de um Vaso de Pressão. Os valores obtidos foram comparados com os resultados apresentados nos mesmos projetos, através da aplicação de alguns algoritmos consagrados da literatura. Embora os algoritmos utilizados no trabalho comparativo, em sua forma original, tenham sido propostos para operarem com variáveis contínuas, eles foram adaptados para o tratamento de problemas com variáveis discretas, uma vez que nos projetos estruturais em geral, é comum encontrar-se este tipo de variável. Os testes executados mostram o Algoritmo Genético como sendo um eficaz algoritmo exploratório, nas regiões do espaço de busca, devido a sua globalidade e flexibilidade.

Abstract

In this work, the Genetic Algorithm (GA), a non linear programming algorithm, is used for the computer aided design of ocean structures. It was applied in a Hatch Cover and a Pressure Vessel design. The results were compared to those obtained with some classical algorithms found in technical literature. However, these algorithms, originally, work with continuous variables, then some improvements were incorporated in order to make possible its application to discrete variables. Tests show the Genetic Algorithm approach to be a good algorithm for exploring search spaces due to its globality and flexibility.

1 - Introdução

Estão muito enraizados na engenharia oceânica, em particular, e na engenharia, em geral, os procedimentos de projeto baseados no método conhecido como *espiral de projeto*, onde os valores a serem definidos para vários parâmetros de projeto são, a princípio, admitidos com base em experiências passadas e melhorados, segundo critérios específicos de performance, através de repetidos exercícios de modificações por tentativa e erro.

Com a introdução de formas estruturais inovadoras e o crescimento da complexidade dos requisitos de projeto surge a necessidade de novos procedimentos capazes de satisfazer os requisitos de projeto em espaços mais curtos de tempo.

Conforme salientava Cella et al. (1973), faz se necessária a substituição da prática convencional, que depende em larga escala de experiência passada, por um sistema automatizado de projeto no qual o problema em questão é formulado numericamente e, em conseqüência disto, tratado por técnicas puramente matemáticas, inclusive de otimização.

Atualmente, os computadores, cada vez mais velozes, permitem ao engenheiro considerar uma faixa maior de possibilidades de projeto e, aliado a isso, os procedimentos de otimização propiciam um meio sistemático de escolha entre estas alternativas, desde que baseadas em algum critério. Se utilizados adequadamente, esses procedimentos podem, na maioria dos casos, melhorar os resultados finais de projeto.

Estudos de otimização em projetos de estruturas de navios datam dos meados da década de 70, Kitamura (1972), e um certo grau de sucesso tem sido atingido pelos

diversos modelos propostos que utilizam principalmente a técnica de minimização seqüencial sem restrições (SUMT) envolvendo essencialmente os parâmetros de projeto tratados como variáveis contínuas. No entanto, os parâmetros de projeto são inerentemente discretos - as espessuras do chapeamento do casco do navio, por exemplo, são padronizadas em valores discretos de acordo com os processos dos fabricantes de aço - indicando que uma abordagem mais adequada para problemas de projeto deveria ser encontrada na otimização global, permitindo a combinação de variáveis contínuas e discretas.

Devido à constante busca de melhorias das técnicas de projeto, deve ser considerada a possibilidade de uso de ferramentas mais modernas, mais robustas e mais rápidas, baseadas em conceitos heterodoxos de otimização.

As considerações anteriores estimulam o presente estudo de otimização de estruturas de embarcações através do uso do Algoritmo Genético, conforme Goldberg (1989).

Como o nome indica, o Algoritmo Genético, que é um algoritmo de busca, tenta simular, em computador, os processos de evolução observados na natureza, como pode ser observado em Hillis (1987) e Davis (1991), muito embora o processo de evolução natural não seja um processo de busca, mas sim um processo que garante a sobrevivência dos melhores adaptados a um determinado meio. Diversos biólogos ainda estão intrigados com os mecanismos de evolução desde que a Teoria da Evolução Biológica ganhou maior aceitação. Os mecanismos que governam esta evolução não estão totalmente desvendados, porém, algumas de suas características, que serão discutidas posteriormente neste capítulo, são perfeitamente conhecidas e aceitas.

As características do processo evolutivo intrigavam John Holland (1975), criador de um novo ramo da programação matemática, naturalmente batizado de *algoritmo genético.* Holland acreditava que estas características se, adequadamente, incorporadas em um algoritmo matemático gerariam uma técnica para resolver problemas de difícil solução, da mesma forma que a natureza tem resolvido os seus, isto é, através da evolução natural.

Esta nova abordagem difere das várias técnicas convencionais de otimização, as quais procuram melhorar sucessivamente, conjunto por conjunto, as variáveis independentes de um problema, gerando uma única solução com ótima performance. O algoritmo genético objetiva melhorar em paralelo um número de conjuntos de variáveis independentes e, deste modo, simultaneamente melhorar os valores estimados de performance para um grupo de soluções.

Para que se possa ter um melhor entendimento de como funciona o Algoritmo Genético é importante que haja um certo conhecimento do processo de evolução natural. As próximas páginas têm a pretensão de fazer um pequeno resumo sobre um assunto de tão grande controvérsia entre os próprios biólogos, estudiosos do assunto.

1.1 - A teoria científica que provoca controvérsias.

A teoria científica da evolução das espécies teve origem com a obra de Charles Darwin, *A origem das espécies* (1859). Darwin e sua teoria suscitaram enormes controvérsias no século XIX. Uma das questões que mais excitavam os ânimos, na época, era saber se o homem descendia do macaco ou não. Para os religiosos que acreditavam na mensagem bíblica (que era aceita como única verdade) de que o homem fora criado por Deus, essa concepção era inaceitável.

Mais de cem anos depois da morte de Darwin (1809 - 1882), pode se pensar que a sua teoria é completamente aceita por todos e que já não excita mais os ânimos. Com efeito, os religiosos, de forma geral, concordam que a história da criação que aparece na Bíblia, no capítulo entitulado Gênesis, não pode ser considerada no seu sentido literal. Na França, no entanto, algumas seitas religiosas, tais como as Testemunhas de Jeová, continuam a negar a teoria da evolução das espécies e a afirmar que o mundo e o homem foram realmente criados no início dos tempos, como descreve o texto do Gênesis. Nos Estados Unidos, durante a década de 80, um pequeno grupo de protestantes, conhecidos como *fundamentalistas* ou *criacionistas*, exerceram certa pressão para que as escolas americanas ensinassem, e atribuíssem o mesmo peso à teoria da evolução e ao relato do Gênesis.

Contudo, pode-se acreditar que, pelo menos entre os cientistas, há total consenso para se reconhecer que a evolução das espécies é um fato. Julian Huxley, um dos fundadores da teoria neodarwinista (que é a continuidade à teoria de Darwin no século XX), declarou, em 1959, que a teoria Darwinista já não era uma hipótese, mas sim um fato. Em 1976, a geneticista Claudine Petit e o bioquímico Emile Zuckerkandl concordam com Huxley, em um manual destinado ao ensino superior, onde diziam: *A noção de evolução dos seres vivos é hoje tão comumente aceita que não sentimos mais a necessidade de demonstrar sua existência*. Na década de 80 a unanimidade dos cientistas com relação à realidade da evolução foi rompida. Michael Denton, diretor do Centro de Pesquisa em Genética Humana de Sydney, na Austrália, escreveu todo um livro, intitulado *Evolution: une théorie en crise*, a favor do criacionismo, negando a realidade da evolução.

A posição de Denton revela o clima de críticas que surge na virada dos anos 80 no mundo da biologia evolucionista. Seu argumento apoia-se, em parte, nas falha reais dos mecanismos da evolução das espécies, propostos pelos defensores do neodarwinismo (teoria ainda hoje dominante), falhas que foram evidenciadas principalmente pelo geneticista japonês, Motoo Kimura (*Evolutionary rate at the molecular level*, *Nature*, 1968), e pelo paleontólogo americano, Stephen J. Gould, que escreveu *Knight Takes Bishop* em *Natural History*, 1986. O grande erro de Denton foi simplesmente acreditar que a partir do momento em que os mecanismos neodarwinistas da evolução não são satisfatórios para todos os biólogos, a própria evolução possa ser posta em dúvida. Gould e Kimura afirmam que a teoria da evolução neodarwinista é incompleta ou falsa em certos aspectos, como por exemplo, o papel específico da seleção natural no processo de nascimento das espécies. Contudo, ao contrário de Denton, eles não negam que a evolução exista.

Na verdade, a discussão científica sobre a teoria da evolução em nossa época não tem como objetivo a realidade da evolução, nem mesmo a reconstituição das árvores genealógicas deste ou daquele grupo de animais ou plantas. Hoje, a principal controvérsia científica tampouco tem como objetivo o confronto entre *lamarckismo* e darwinismo. *Lamarckismo* é o nome dado a primeira teoria evolucionista argumentada cientificamente, pelo grande zoólogo francês Jean-Baptiste de Lamarck (1744 - 1829). Atualmente esta teoria é apresentada com um quê de compaixão para com seu autor, uma vez que foi infelizmente fundamentada numa noção científica errônea: *a hereditariedade dos caracteres adquiridos*. Por exemplo, segundo a visão atual e comumente dada do *lamarckismo*, a girafa teria nascido de uma espécie de proto-girafa ancestral com pescoço curto. Para comerem as folhas das árvores mais altas, esses animais teriam progressivamente esticado o pescoço e transmitido esse caracter a

seus descendentes, assim estes nasceriam com um pescoço já mais alongado que seus pais. Com a repetição do processo ao longo das gerações, a girafa de pescoço longo seria finalmente formada. Contudo a transmissão de caracteres, adquiridos durante a existência, nunca foi verificada experimentalmente, desta forma, é hoje considerada, quase com unanimidade, uma noção científica errônea. Contudo Lamarck não é o autor desta noção, que já existia desde a Antigüidade, e que persistiu depois dele, até o final do século passado.

Atualmente, em matéria de evolução, a controvérsia científica apoia-se essencialmente em dois aspectos. Em primeiro lugar, que posição ocupa realmente a seleção natural de Darwin na evolução? O acaso (representado pelas mutações) não intervém da mesma forma ou até mais? Essa é a opinião do geneticista Motoo Kitamura, que defende uma teoria não darwinista. Em segundo, como as espécies se originam? Já que se há *descendência* e se algumas espécies são ancestrais de outras, é necessário que, em um dado momento, umas dêem origem às outras. Mas como se dá concretamente este nascimento? Curiosamente, o importante livro de Darwin, *A origem das espécies*, fala extensamente da evolução das espécies no sentido geral da descendência de uma espécie da outra, mas não fala realmente da sua origem, ou seja, de como uma nova espécie vem ao mundo. Tais questões, apesar de puramente científicas (papel que diz respeito ao acaso e à seleção natural; como se originam as espécies) têm importantes conseqüências na forma de se representar o lugar do homem no mundo, pois as condições em que se dá o aparecimento da espécie humana podem, sem dúvida alguma, fornecer elementos para a resposta a essa velha questão filosófica.

1.2 - A seleção natural como um mecanismo da evolução

Mesmo quando se assemelham, as espécies distintas apresentam, contudo, diferenças. A teoria de Darwin é uma teoria de *descendência com modificação* . E as modificações correspondem à adaptação dos órgãos ao modo de vida particular de cada espécie. Assim como Lamarck, Darwin baseou-se na constatação fundamental entre a forma e o desempenho dos órgãos dos seres vivos e seu modo de vida, mas ao tentar explicar a graduação entre a diferença individual, variedade, subespécie e espécie, o biólogo inglês voltou-se para outro mecanismo de evolução que está além da hereditariedade dos caracteres adquiridos: a seleção natural.

Tudo leva a crer que Darwin teve como ponto de partida para a sua teoria, a observação das práticas dos criadores de animais, uma vez que estes sabiam aproveitar da melhor forma as diferenças entre os indivíduos de um rebanho. Por exemplo, em um rebanho de carneiros, alguns apresentam melhor lã, outros mais carne, outros são, ainda, mais férteis. Os criadores conseguem gerar boas linhagens (de lã, carne, leite, etc...), cruzando entre si, de preferência, os melhores produtores, que é chamado de *a seleção dos reprodutores.*

Em 1838, além de ler vários livros sobre criação de animais, Darwin leu também várias obras filosóficas, entre outras, a qual merece destaque a obra do economista inglês Thomas Malthus, *Ensaio sobre o princípio populacional.* Malthus discorre sobre a idéia de que, sem interferências (fome, epidemias, etc...), a população humana se duplicaria a cada vinte e cinco anos, através, simplesmente, da reprodução. Darwin transpôs esta noção para o domínio animal, raciocinando da seguinte forma: em tempos normais, a população de uma dada espécie não cresce sensivelmente; o que significa que todos os animais capazes de se reproduzir dentro desta população não

alcançam o estágio da reprodução. Calculou que estes morrem antes de atingir esse momento, seja por não terem sido capazes de enfrentar certas condições do meio, seja por terem perecido na garra dos predadores ou devido a doenças ou ainda, por não terem encontrado condições ideais de subsistência, tendo sido precedidos por seus congêneres melhores preparados para as condições apresentadas pelo meio. Assim, tudo se passaria como se na natureza houvesse uma seleção dos reprodutores análoga àquela feita artificialmente[1] pelos criadores em seus rebanhos.

No caso da *seleção artificial* (realizada pelos produtores), os animais a serem escolhidos para a reprodução são os que apresentam melhores desempenhos relacionados a certos traços desejáveis pelos criadores. No caso da *seleção natural*, os *reprodutores eleitos* seriam aqueles que melhor se adaptassem às condições de existência do meio; em outras palavras, "os mais aptos para sobreviver" na luta pela existência. Neste momento Darwin enuncia o seguinte argumento: se se produzem entre certos indivíduos de uma determinada espécie

> *variações úteis (...) na grande e terrível batalha pela vida, (...) podemos duvidar (é preciso sempre lembrar que o número de indivíduos que nascem é muito maior do que os que sobrevivem) de que os indivíduos, ao possuírem uma vantagem qualquer, por mais imperceptível que possa ser, tenham uma chance maior de sobreviver e de reproduzir-se? Podemos estar certos, por outro lado, de que qualquer variação, por mais nociva que possa ser ao indivíduo, dá início, forçosamente, ao desaparecimento deste. Dei o nome de seleção natural ou de persistência*

[1] Em capítulos posteriores será utilizado o termo artificial para a seleção realizada pelo AG.

do mais apto a essa conservação das diferenças e das variações individuais favoráveis e a essa eliminação das variáveis nocivas.

A seleção natural tornou-se um mecanismo plausível para a explicação da evolução. Após enunciar a seleção natural, Darwin expôs em *A origem das espécies* as provas da evolução, que são ainda hoje reconhecidas, universalmente, como exatas (exceto por Denton), reunidas em cinco grandes categorias: provas paleontológicas, biogeográficas, sistemáticas, morfológicas e embriológicas.

1.3 - O Neodarwinismo

Em sua obra, Darwin praticamente não oferece exemplos concretos de seleção natural, realmente observados na natureza. Seu argumento repousa em dedução teórica, baseada na observação da seleção artificial praticada pelos criadores. Ele utilizou o fato de que os criadores obtêm extraordinárias modificações nos caracteres de uma espécie, ao escolherem judiciosamente os reprodutores de cada geração. Contudo, nas ciências em geral, não se pode utilizar apenas deduções teóricas para se demonstrar a existência de uma dado fenômeno.

Quando escreveu *A origem das espécies* (1859), Darwin até então não tinha provas quanto à realidade da seleção natural *in loco*. A primeira lhe seria oferecida três anos depois, por um naturalista inglês, H. W. Bates, que havia sido, na década precedente, o companheiro de expedição na Amazônia, do co-descobridor da teoria da evolução pela seleção natural, Alfred Russel Wallace. Bates observou, na América do Sul, que certas espécies de borboletas apresentavam em suas asas motivos coloridos iguais aos de outra espécie, da mesma região, que são enjoativas e até mesmo venenosas quando ingeridas por pássaros. Consequentemente, estes evitavam comer as

borboletas da espécie que lhes causavam náuseas, assim como as que as *mimetizavam*. Essa exatidão na reprodução dos motivos coloridos das borboletas *enjoativas* pelas borboletas miméticas só poderia ser explicada pela seleção natural, segundo Bates. As borboletas que não conseguiam uma boa reprodução do modelo eram devoradas e não deixavam descendentes . Inversamente, as borboletas que reproduziam melhor o modelo, tinham ainda mais chances de sobrevivência e de deixar descendentes. Logo, com o tempo, as espécies de borboletas miméticas seriam compostas apenas por indivíduos que imitassem cada vez melhor as borboletas não-palatáveis.

É importante salientar que no exemplo acima não se deve imaginar que as borboletas reproduzem o modelo por uma esforço individual e, em seguida, o transmitem a seus descendentes, como no modelo evolucionista proposto por Lamarck. Darwin pensava que a reprodução do modelo, no caso das borboletas, se devia a variações aleatórias inatas: certo indivíduo nascia com um dado motivo colorido; se não estivesse próximo do modelo, era eliminado; se estivesse, sobrevivia e reproduzia-se. Contudo Darwin nunca foi hostil à idéia de hereditariedade dos caracteres adquiridos. Sob o ponto de vista da seleção natural, pouco importa saber por qual mecanismo as borboletas copiam seu modelo e transmitem-no a seus descendentes; ou seja: por um *esforço individual* ou por *variação inata ao acaso*. O essencial é que certos indivíduos realizam melhor esta reprodução do que outros e, como conseqüência, deixam mais descendentes do que estes. Teóricos da evolução, posteriores a Darwin, no século XIX, rejeitaram categoricamente o princípio da hereditariedade dos caracteres adquiridos. No século XX, este ponto de vista foi incorporado às concepções genéticas levadas em conta pela teoria neodarwinista da evolução.

Na década de 30, vários biólogos começaram a estudar a genética das populações naturais (populações que não são estudadas em laboratório) e principalmente a genética das diferenças entre as raças. Desta forma, em 1937, o biólogo russo Th. Dobzhansky reuniu todos os dados da genética de populações: modelos matemáticos, observação em laboratório da seleção de diferentes variações genéticas, observação da diferenciação genética de populações representando raças geográficas... Todo este estudo permitia considerar que a evolução se desenrolava como previu Darwin: sob a ação da seleção natural, variantes hereditárias tornavam-se mais freqüentes em determinadas populações, que representavam variedades ou raças de uma espécie. Algumas destas atingiam, em seguida, um nível de diferenciação genética ainda maior, o que lhe garantia o *status* de espécie[2]. Em sua obra, *Genetics and The Origin of Species* (1937), Dobzhansky formulou o postulado de base do neodarwinismo, no século XX: *A evolução consiste numa mudança na composição genética das populações*. Atualmente, este postulado foi um pouco modificado pelos neodarwinistas, para torná-lo mais explícito: *A evolução é uma mudança na freqüência dos alelos*[3] *nas populações*.

No final da década de 40, a teoria neodarwinista da evolução foi edificada por contribuições provenientes das diversas disciplinas da biologia, sendo assim batizada de *teoria sintética da evolução* por Julian Huxley (neto de T.H. Huxley, porta-voz de Darwin). Em 1942 ele publicou *Evolution: The Modern Synthesys*, em resumo, a síntese repousa nestas duas proposições: em primeiro lugar, a evolução consiste no surgimento de novas variantes de genes por mutação ao acaso, nas populações,

[2] Alguns naturalistas estabeleceram, graças às suas observações das populações naturais, uma definição biológica de espécie: uma espécie é formada pelas populações cujos membros entrecruzam-se de maneira habitual (é uma comunidade sexual ativa).

[3] Os alelos são, em termos técnicos, as diferentes variantes de um mesmo gene.

seguida de substituição gradual, sob a ação da seleção natural, das variantes menos apropriadas para as mais apropriadas; em segundo, o mecanismo de modificação da composição genética das populações permite explicar como uma espécie dá origem a outra, em conseqüência da diferenciação genética acrescida de uma de suas subespécies.

Em 1956, Julian Huxley escreveu:

> *A descoberta do princípio da seleção natural tornou a evolução compreensível; junto à da genética moderna, tornou insustentável qualquer outra explicação para a evolução ... não somente é um fator eficaz e efetivo da evolução, como é o único fator efetivo.*

1.4 - O desenvolvimento deste trabalho

A partir da introdução ao estudo dos processos de evolução natural, apresentados neste capítulo, segue-se o capítulo 2, onde se apresenta o Algoritmo Genético, um algoritmo de programação matemática não linear, que apresenta como singularidade operadores matemáticos que utilizam conceitos básicos da genética.

No capítulo 2, muitos dos conceitos são mostrados de forma quase intuitiva, sendo necessária a apresentação do capítulo 3, Fundamentos Matemáticos, que vem provar matematicamente as importantes características do Algoritmo Genético em problemas de otimização.

O capítulo 4 apresenta de forma genérica a Formulação do Problema de Otimização, que se faz necessária, uma vez que qualquer problema de programação matemática pode ser formulado de maneira única.

Com a implementação do algoritmo, em um programa em linguagem Pascal, pode-se fazer aplicações a alguns exemplos simples, assim como comparar o desempenho do Algoritmo Genético com outros já consagrados da literatura. O capítulo 5 apresenta os resultados destas aplicações, assim como a discussão destes.

O desenvolvimento deste trabalho mostra o Algoritmo Genético (AG) como sendo um algoritmo exploratório, eficaz e robusto, nas regiões do espaço de busca, devido a sua globalidade e flexibilidade.

2 - O Algoritmo Genético

Conforme foi apresentado no capítulo anterior, os organismos vivos têm a intrínseca capacidade de solucionar problemas. Eles exibem uma versatilidade que deixam o melhor programa de computador envergonhado. Esta observação é especialmente irritante para cientistas da computação, que podem gastar meses, ou mesmo anos, de esforço intelectual no desenvolvimento de um algoritmo, ao passo que os organismos conseguem suas habilidades através de mecanismos, aparentemente indiretos, de evolução e seleção natural.

Pesquisadores pragmáticos vêem o extraordinário poder de evolução como algo a ser emulado ao invés de invejado. Através do aproveitamento dos mecanismos da evolução, pesquisadores podem ser capazes de "gerar" programas que solucionam problemas, mesmo quando nenhuma pessoa possa entender completamente sua estrutura. De fato, este então chamado Algoritmo Genético já tem demonstrado a habilidade de fazer avanços em projetos de sistemas extremamente complexos.

O algoritmo genético faz com que seja possível explorar a mais ampla gama de soluções potenciais para um problema, de melhor forma que os programas convencionais.

Segundo a teoria da evolução, tomando-se como base o neodarwinismo, a maioria dos organismos evolui principalmente através de dois processos primários: seleção natural e reprodução sexuada. O primeiro determina quais membros da população sobrevivem para a reprodução e o segundo, assegura a mistura e a recombinação entre os genes dos seus descendentes. Ocorre a troca de material

genético, quando há uma fusão entre o espermatozóide e o óvulo, combinando os cromossomos de um gameta com os do outro. A recombinação ou mistura (*crossing over)* de parte destes cromossomos permite às criaturas evoluírem muito mais rapidamente do que se cada descendente tivesse uma simples cópia dos genes de um dos pais, modificado ocasionalmente através de mutação.

As pessoas têm empregado uma combinação de cruzamento e seleção de raças por milênios, para criar melhores colheitas, apurar qualidades em cavalos, produzir flores ornamentais. Contudo, este não é um processo simples para ser traduzido para um programa de computador. O problema chave é a construção de um *código genético* que possa representar a estrutura de diferentes problemas, assim como o DNA representa a estrutura de uma pessoa ou de um rato.

Pode-se neste momento considerar-se uma importante questão:

O que são na realidade os Algoritmos Genéticos?

Os Algoritmos Genéticos são algoritmos de pesquisa baseados nos mecanismos de seleção e da genética natural. Eles combinam a sobrevivência do mais apto entre, *indivíduos* ou projetos, caracterizados por estruturas de *strings,* com uma elaborada troca de informação aleatória, para formar um algoritmo de busca com as mesmas habilidades encontradas nos mecanismos da evolução natural. Em cada geração, um novo conjunto de criaturas artificiais (*strings*) é criado com base em subgrupos de melhor performance da geração anterior. Apesar da característica aleatória, algoritmos genéticos não fazem simplesmente uma busca aleatória. Eles eficientemente exploram informações históricas para especular novos pontos de busca com uma já esperada melhora na performance.

Desde a sua criação por Jonh Holland (1975), as principais metas nas pesquisas realizadas com os Algoritmos Genéticos são:

1. Abstrair e explicar rigorosamente os processos adaptativos dos sistemas naturais.
2. Projetar sistemas artificiais, *softwares*, que retenham os importantes mecanismos dos sistemas naturais.

Este enfoque tem levado a importantes descobertas nos dois ramos da ciência, natural e artificial.

O tema central da pesquisa em algoritmo genético tem sido *robustez*, o equilíbrio entre eficiência e eficácia necessária para a sobrevivência nos mais variados meios. As implicações da robustez para sistemas artificiais são múltiplas. Caso sistemas artificiais possam ser feitos mais robustos, o custo de um novo projeto pode ter uma redução considerável. Caso maiores níveis de adaptação possam ser obtidos, sistemas existentes poderiam realizar suas funções de melhor forma e por mais tempo. Projetistas de sistemas artificiais - ambos *software* e *hardware*, em sistemas de engenharia, computação ou negócios - somente podem maravilhar-se com a robustez, a eficiência e a flexibilidade dos sistemas biológicos. Características de auto - reparo, auto - orientação, e reprodução são regras nos sistemas biológicos, ao passo que elas apenas existem nos mais sofisticados sistemas artificiais.

Desta maneira, pode-se construir uma interessante conclusão: onde uma performance robusta é desejada (e onde ela não é?), a natureza faz melhor; os segredos da adaptação e sobrevivência são melhor aprendidos através de um cuidadoso estudo de exemplos biológicos. Não se pode aceitar o Algoritmo Genético somente pelo apelo ao argumento da beleza da natureza. Algoritmos Genéticos são teoricamente e

empiricamente comprovados adequados em efetuar exploração robusta em espaços complexos. A primeira monografia no tópico é de John Holland "Adaptation in Natural and Artificial Systems" (1975). Vários *papers* e dissertações estabelecem a validade da técnica em funções de otimização e aplicações de controle. Tendo sido estabelecido como uma válida abordagem para problemas que requerem uma busca eficiente e efetiva, os algoritmos genéticos estão agora sendo mais comumente encontrados em aplicações científicas, de negócios e de engenharia. As razões pelas quais se encontram crescentes os números de aplicações do algoritmo são claras. Estes são computacionalmente simples e poderosos na busca de pontos de ótimo, nos diversos tipos de problemas de otimização. Além do mais, eles não são fundamentalmente limitados por hipóteses restritas sobre o espaço de busca (hipóteses acerca de continuidade, unimodalidade, existência de derivadas, etc),como os tradicionais algoritmos da programação matemática.

2.1 - Diferença entre o Algoritmo Genético e os métodos tradicionais

De modo a superar os métodos mais tradicionais, em termos de robustez, os Algoritmos Genéticos são diferentes dos procedimentos mais comuns de otimização e de busca , da seguinte forma:

1. AGs trabalham com a codificação de um conjunto de parâmetros e não com parâmetros isoladamente;
2. AGs promovem busca em uma população de pontos e não em um único ponto;
3. AGs utilizam resgate de informações da própria função objetivo e não informações derivadas ou outros conhecimentos auxiliares;

4.AGs utilizam regras de transição probabilísticas e não regras determinísticas.

Os algoritmos genéticos requerem um conjunto de parâmetros naturais do problema a ser codificado como uma *string* de comprimento finito sobre algum conjunto finito. Vejamos agora um problema de otimização onde a codificação surge de forma natural.

Considere o problema de uma caixa de interruptores. Ele diz respeito a um dispositivo de caixa preta com um banco de cinco interruptores de entrada. Para cada conjunto de cinco interruptores, há um sinal de saída *f*, matematicamente $f = f(s)$, onde *s* é um conjunto particular dos sinais dos cinco interruptores. O objetivo do problema é formar conjuntos de interruptores para obter o máximo valor possível de *f*. Com outros métodos de otimização poderia-se trabalhar diretamente com o conjunto de parâmetros (conjunto de interruptores) e realizar a busca, indo de um conjunto para outro, usando regras de transição do particular método. Com os Algoritmos Genéticos, primeiro codifica-se os interruptores como uma *string* de comprimento finito. Um código simples pode ser gerado considerando-se uma *string* de cinco 1's e 0's onde cada um dos cinco interruptores é representado por um 1 se o interruptor estiver ligado e um 0 se desligado. Com este código, a *string* 01000 codifica o conjunto onde somente o segundo interruptor está ligado e os demais, desligados. Um fato importante é que com a utilização do algoritmo genético não é necessário que se conheça o funcionamento da caixa preta. No desenvolvimento do trabalho ficará aparente que os algoritmos genéticos exploram codificações similares de um modo geral, como resultado, eles não apresentam restrições como limitações verificadas em outros métodos (continuidade, existência de derivada, unimodalidade, etc)

Em vários métodos de otimização, move-se cuidadosamente de um único ponto no espaço de variáveis para o próximo, utilizando-se algumas regras de transição para a determinação do próximo ponto. Este método "ponto - a - ponto" é perigoso porque ele é uma prescrição perfeita para a localização de falsos pontos de máximo em espaços de busca multimodal. Em contraste, algoritmos genéticos trabalham a partir de um rico banco de dados de pontos simultâneos (uma população de *strings*), localizando muitos pontos de máximo em paralelo; desta maneira a probabilidade de se encontrar um falso ponto de máximo é reduzida em relação aos métodos que vão de ponto a ponto. Como exemplo considere o problema de otimização da caixa preta supracitado. Outras técnicas para solucionar tal problema podem começar com um conjunto de possíveis posições dos interruptores, aplicar algumas regras de transição, e gerar um novo processo de possíveis posições para os interruptores. Um algoritmo genético começa com uma população de *strings* e depois disto gera sucessivas populações de *strings*. No problema em questão, um começo aleatório utilizando sucessivo jogo de moedas (cara = 1, coroa = 0) pode gerar a população inicial de tamanho $n = 4$ (pequeno paro o padrão do algoritmo genético):

01110

01000

10101

11010

Depois deste início, sucessivas populações são geradas procurando-se processar o cruzamento dos indivíduos melhores adaptados em toda a população.

Muitas técnicas de busca requerem uma grande quantidade de informações auxiliares no processo de otimização propriamente dito. Por exemplo a técnica do

Gradiente necessita derivadas (calculadas numericamente ou analiticamente) para que se possa localizar o ponto de máximo. Ao contrário, o Algoritmo Genético não tem necessidade destas informações auxiliares. Para realizar uma busca efetiva na obtenção de melhores e melhores estruturas, ele necessita somente dos valores da função objetivo associados com *strings* individuais. Esta característica faz do Algoritmo Genético o método mais direto entre muitos outros esquemas de busca.

Diferente de muitos métodos, os AGs usam regras de transição probabilísticas para guiar suas buscas. Para pessoas familiarizadas com os métodos determinísticos isto parece estranho, mas o uso de probabilidade não sugere que o método é simplesmente aleatório. Os algoritmos genéticos usam a escolha aleatória como uma ferramenta para guiar a pesquisa através das regiões do espaço de busca com um provável melhoramento.

Estas quatro diferenças - o uso direto da codificação, a busca dentro de uma população, a não necessidade de informações auxiliares, e os operadores aleatórios - em conjunto, contribuem para a robustez do algoritmo genético e apresentam, como resultado, vantagem sobre outras técnicas mais comumente utilizadas.

2.2 - Um Algoritmo Genético simples

Os mecanismos de um algoritmo genético simples são surpreendentemente fáceis, envolvendo nada mais complexo do que cópias de *strings* e a troca parcial de *strings*. A explicação de como este poderoso processo funciona é muito mais sutil. Simplicidade de operação e eficiência são duas das principais atrações do algoritmo genético.

O exemplo da seção anterior mostrou, de forma bastante simplificada, como o algoritmo genético processa uma população de *strings*. Retomando-se o problema da caixa de interruptores, recorda-se que a população inicial era formada por quatro *strings*:

01110

01000

10101

11010

Lembrando-se também que esta população foi escolhida aleatoriamente através de 20 jogadas sucessivas de uma moeda imparcial. Agora deve-se definir um conjunto de simples operações que, utilizando a população inicial, gerem sucessivas populações que evoluem com o tempo.

Um algoritmo genético simples, que produz bons resultados em muitos problemas práticos, é composto de três operadores:

1. Reprodução

2. *Crossover*

3.Mutação

Reprodução é um processo no qual *strings* individuais são copiadas de acordo com os valores da função objetivo, f (biólogos chamam esta função *performance*). Intuitivamente, pode-se pensar na função f como uma medida de lucro ou utilidade que se quer maximizar. Copiar *strings* de acordo com seus valores de performance significa que *strings* com um maior valor têm uma maior probabilidade de contribuir com um ou mais descendentes nas próximas gerações. Este operador é uma versão artificial da

seleção natural, um sobrevivente Darwiniano mais apto entre as *strings*. Em populações naturais o mais apto é determinado pela habilidade da criatura de sobreviver aos seus predadores, pestes e outros obstáculos para atingir a fase adulta e, consequentemente, a reprodução. Neste caso artificial, a função objetivo é o arbitro final que decidirá a "vida ou morte" das criaturas formadas por *strings*, de acordo com a sua performance.

O operador reprodução pode ser implementado, em forma de algoritmo, de várias maneiras. Talvez a mais simples seja a *roleta da sorte*, que apresenta-se de forma detalhada, adiante, neste capítulo.

Depois da Reprodução, um simples *Crossover* pode proceder em dois passos. Primeiro, os membros de uma *string* recém reproduzida no *mating pool* são cruzados aleatoriamente. Segundo, cada par de *strings* sofre o cruzamento da seguinte forma: um inteiro de posição k ao longo da *string* é selecionado de forma aleatória, uniformemente distribuído, entre 1 e o comprimento de *string* menos 1 $[1, l - 1]$. Duas novas *strings* são criadas através da troca de todos os caracteres entre as posições $k + 1$ e l, inclusive. Por exemplo, considere as *strings* A_1 e A_2 da população do problema da caixa de interruptores:

$A_1 = 0\ 1 \mid 1\ 1\ 0$

$A_2 = 1\ 1 \mid 0\ 1\ 0$

Supondo-se que a escolha de um número aleatório entre 1 e 4, obtém-se um $k = 2$ (como indicado pelo símbolo de separação |). Como resultado o *crossover* produz duas novas *strings* onde (') significa que as *strings* são parte de uma nova geração:

$A'_1 = 0\ 1\ 0\ 1\ 0$

$A'_2 = 1\ 1\ 1\ 1\ 0$

Os mecanismos de reprodução e *crossover* são surpreendentemente simples, envolvendo a geração de números aleatórios, cópias de *strings*, e a troca parcial de *strings*. Contudo, a ênfase combinada de reprodução e a estruturação, através da aleatoriedade, da troca de informação do *crossover*, dá aos algoritmos genéticos muito do seu poder.

Neste caso, então, qual é o objetivo do operador Mutação? Sem trazer surpresas, há muita confusão sobre o papel da mutação em genética (tanto natural, quanto artificial), mas qualquer que seja a causa desta confusão, os estudiosos do assunto vêem a mutação como um papel secundário na operação do algoritmo genético. A mutação é necessária, porque mesmo com a reprodução e o *crossover*, efetivamente buscando e recombinando parte do material, ocasionalmente eles podem tornar-se por demais zelosos e perder material genético potencialmente útil (1's ou 0's em uma particular locação). Em sistemas genéticos artificiais, o operador mutação protege o algoritmo de uma perda irrecuperável. No Algoritmo Genético, mutação é , com pequena probabilidade, uma alteração aleatória ocasional do valor da posição da *string*. No código binário do problema da caixa preta, isto simplesmente significa mudar de um 1 para um 0 ou vice-versa. A mutação propriamente dita, é uma *"caminhada"* aleatória no espaço da *string*. Quando escassamente usada com reprodução e *crossover*, ela é uma medida de segurança contra prematuras perdas de informações importantes.

Sabendo-se que o operador mutação tem um papel secundário no simples AG, nota-se que a freqüência de mutação para se obter bons resultados nos estudos empíricos do algoritmo genético é da ordem de uma mutação por mil bits (posição) de

transferência, Goldberg (1989). A porcentagem de mutação é similarmente pequena (ou menor ainda) em populações naturais, levando-se a conclusão de que mutação é apropriadamente considerada como um mecanismo secundário da adaptação no algoritmo genético. Em seu trabalho, Blanc (1994), trata sobre a freqüência das mutações nas populações naturais:

> *... Logo, tínhamos aqui um dos dados objetivos que faltaram a Darwin: as novas variações dos caracteres apareciam de súbito, por mutação espontânea nos fatores genéticos. A questão que se fazia então era saber com que freqüência ocorriam essas mutações nas populações naturais (e não somente nas populações em laboratório). Foram inicialmente pesquisadores russos que, a partir da década de 20, tentaram resolver este problema, (...). Por volta do final dos anos 30, todas essas pesquisas conduziram à noção de que, entre organismos multicelulares (animais e vegetais), a taxa de mutação por gene é de mais ou menos 1/100.000 por célula sexual. Em outros termos, encontraremos uma nova variação para um dado gene em cada geração que compreenda 50 mil indivíduos. (Cada indivíduo é o produto da reunião de duas células sexuais.) É uma cifra consideravelmente inferior à fornecida por Darwin (este calculou, como vimos, que uma nova variação apareceria no décimo ou mesmo no terceiro indivíduo!). De qualquer modo, hoje sabemos que um ser vivo multicelular compreende várias dezenas de milhares de genes em seu patrimônio genético. Na espécie humana, há por volta de 10 mil genes. O que significa, dada uma taxa de mutação de 1 para 100 mil, que cada ser humano é portador, em média, de duas mutações! ...*

Outros operadores genéticos e planos reprodutivos têm sido abstraídos do estudo de exemplos biológicos. Contudo, os três operadores examinados até aqui

(reprodução, *crossover* e mutação) têm provado ser tanto computacionalmente simples, como efetivos na solução de um grande número de importantes problemas de otimização.

2.2.1 - Seleção dos pais através da roleta da sorte

O objetivo da seleção dos pais no algoritmo genético é dar maiores chances de reprodutividade aos membros da população que têm a melhor performance. A seguir, descrevem-se os passos percorridos na seleção de pais através da *roleta da sorte*.

1. Seja f_i o valor da performance do *i-ésimo* membro da população.
2. Seja f_t a somatória da performance de todos os membros da população, isto é:

$$f_t = \sum_{i=1}^{n} f_i$$

3. Seja fac_i a soma acumulada da performance dos primeiros i elementos da população, isto é:

$$fac_i = \sum_{j=1}^{i} f_j$$

4. Seja r , um número aleatório entre 0 e f_t.
5. A roleta retorna o primeiro membro da população, diga-se m, cuja performance acumulada seja maior ou igual a r, isto é:

$$r <= fac_m$$

Na figura 2.1 apresenta-se uma população de dez cromossomos cuja soma total das performances totaliza 130. A primeira linha da figura contém o índice de cada cromossomo, a segunda contém a performance de cada cromossomo, e a terceira, a soma parcial acumulada das performances. Na figura mostram-se sete números

gerados aleatoriamente no intervalo entre 0 e 130, juntamente com o índice do cromossomo que seria escolhido pela *roleta da sorte* para cada um destes números. Em cada caso, o cromossomo escolhido é o primeiro no qual a soma parcial das performances é maior ou igual ao número aleatório.

O efeito da seleção de pais através da *roleta da sorte* é retornar uma seleção de pais aleatória. Embora este procedimento de seleção seja aleatório, a chance de cada pai ser escolhido é proporcional à sua performance, ou seja, quanto maior for a sua performance, maior será a sua probabilidade de escolha, o que será provado matematicamente no capítulo 3.

Figura 2.1: Exemplo de seleção de pais através da roleta da sorte.

Cromossomo	1	2	3	4	5	6	7	8	9	10
Performance	6	10	18	24	11	22	6	17	10	6
Soma Acumulada	6	16	34	58	69	91	97	114	124	130

Número Aleatório	7	83	29	72	130	58	42
Cromossomo Escolhido	2	6	3	6	10	4	4

A primeira tabela mostra a performance de dez cromossomos e a soma parcial das performances. A segunda tabela mostra o cromossomo que seria escolhido pelo algoritmo *roleta da sorte* utilizando-se os valores de performance para cada um dos sete números gerados aleatoriamente.

A seguir será realizada uma simulação manual do algoritmo genético para demonstrar sua robustez e o seu mecanismo de operação.

2.3 - Simulação manual do Algoritmo Genético

Será feito, passo - a - passo, a aplicação do algoritmo genético para um problema particular de otimização. Considere o problema de maximização da função:

$$f(x) = | x^2 - 2x |$$

no intervalo inteiro [0 , 31]

Para a utilização do AG deve-se, em primeiro lugar, codificar as variáveis de decisão do problema como alguma *string* de comprimento finito. Para este problema, a variável x será codificada simplesmente como um binário inteiro de comprimento 5. Generalizando, dado um intervalo de definição $x \supset [x_{min}, x_{max}]$ e uma resolução Δx para a representação de uma variável x, o comprimento mínimo necessário para a representação binária de x é dado por

$$l \approx \log_2 \frac{x_{max} - x_{min}}{\Delta x} \tag{2.1}$$

Antes de realizar a simulação, se fará uma breve revisão da noção de binário, sendo mais fácil a princípio pensar em números na base 10, onde se utilizam 10 dígitos {1, 2, ... , 9, 0}. Por exemplo, o número de cinco dígitos 19970 pode ser pensado como:

$$1.10^4 + 9.10^3 + 9.10^2 + 7.10^1 + 0.10^0 = 19970$$

Na aritmética na base 2 , há somente dois dígitos para se trabalhar, 0 e 1, e como um exemplo o número 11010 codificado para um número da base 10:

$$1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 = 16 + 8 + 2 = 26$$

Com um *byte* de comprimento 5, formado por dígitos, binários, pode-se obter números entre 0 (00000) e 31 (11111). Com a função objetivo definida e codificada será simulada uma única geração de um Algoritmo Genético contemplando reprodução, *crossover* e mutação.

A princípio, uma população inicial é selecionada aleatoriamente. Foi selecionada uma população de tamanho 4 pelo sorteio (cara ou coroa) de uma moeda

lançada 20 vezes. Pode-se ignorar este "passo", utilizando-se a população inicial criada anteriormente, desta mesma forma, para o problema da caixa de interruptores. Esta população é mostrada na segunda coluna da tabela 2.1, observa-se que os valores decodificados de x são apresentados, assim como a performance ou valor da função objetivo $f(x)$. Os valores da performance $f(x)$ são calculados a partir da representação da *string*, veja a segunda *string* da população inicial, 01000. Decodificando-o como um binário inteiro, sem sinal, tem-se:

$$decode\ (01000) = x$$

$$x = 0.2^4 + 1.2^3 + 0.2^2 + 0.2^1 + 0.2^0 = 1.2^3 = 8$$

Para calcular a performance ou função objetivo, simplesmente eleva-se o valor de x ao quadrado, do resultado se subtrai o valor de x multiplicado por 2 e por fim calcula-se o módulo do número obtido, resultando $f(x) = 48$, o valor da performance. Os demais valores de x e $f(x)$ são obtidos de forma similar.

Tabela 2.1 Um Algoritmo Genético através da simulação manual

String No.	População inicial	valor de x	$f_i = f(x) = \lvert x^2 - 2x \rvert$	$pselect_i$ $\frac{f_i}{f_t}$	Seleção Esperada $\frac{f_i}{\bar{f}}$	Seleção Real (*Roleta da sorte*)
1	0 1 1 1 0	14	168	0.14	0.54	1
2	0 1 0 0 0	8	48	0.04	0.16	0
3	1 0 1 0 1	21	399	0.32	1.29	1
4	1 1 0 1 0	26	624	0.50	2.01	2
Soma (f_t)			1239	1.00	4.00	
Média ($\bar{f}$)			310	0.25	1.00	
Máximo			624	0.50	2.01	

Seleção Esperada: $\frac{f_i}{\bar{f}} = \frac{f_i}{\frac{f_t}{n}} = n \cdot \frac{f_i}{f_t} = n \cdot pselect_i$.

Tabela 2.1 (Continuação)

Mating Pool depois da reprodução (| = posição de Troca)	Cruzamento (seleção aleatória)	Posição do *crossover* (seleção aleatória)	Nova População	Valor de x	$f_i = f(x) = \| x^2 - 2x \|$
0 1 | 1 1 0	1	2	0 1 0 1 0	10	80
1 1 | 0 1 0	2	2	1 1 1 1 0	30	840
1 0 1 | 0 1	4	3	1 0 1 1 0	22	440
1 1 0 | 1 0	3	3	1 1 0 0 1	25	575
Soma (f_t)					1935
Média ($\bar{f}$)					484
Máximo					840

Uma geração do algoritmo genético começa com a reprodução. Seleciona-se o *mating pool* da próxima geração através da *roleta da sorte*. A simulação real deste processo tem resultado nas *strings* 1 e 3 recebendo 1 cópia no *mating pool*, a *string* 4 recebendo 2 cópias e a *string* 2 não recebendo cópias. Como pode ser visto na coluna 7 da tabela 2.1. Comparando-se este resultado com o número esperado de cópias ($n.pselect_i$), tem-se o que era esperado: os melhores recebem mais cópias, a média continua a mesma, e os piores "morrem".

Com um ativo grupo de *strings* procurando por pares, um *crossover* simples se procede em dois passos:

1. *strings* são cruzadas aleatoriamente, usando lançamento de moedas para definir os pares do "feliz casal",
2. é feita a troca de material de parte das *strings*, utilizando-se lançamento de moedas para selecionar a posição (*crossing*) a partir da qual será feito a troca.

Retornando à tabela 2.1, a escolha ao acaso dos pares promove o cruzamento da segunda *string* com a primeira. Sendo 2 a posição de *crossing*, as duas *strings* 01110 e 11010 cruzam-se, trocando parte do material, produzindo duas novas *strings*, 11110 e 01010. As duas *strings* remanescentes no *mating pool* são cruzados na posição 3; as *strings* resultantes podem ser checadas na tabela 2.1 .

O último operador, mutação, é feito em base de bit - a - bit. Neste exemplo foi adotada a probabilidade de mutação de 0,1%. Tendo-se nesta geração 4 *strings* de 5 bits, espera-se que a mutação seja 4 . 5 . 0,001 = 0,02 bits a ser submetido à mutação durante a dada geração. A simulação deste processo indica que nenhum bit se submeterá à mutação para este valor de probabilidade. Como resultado, em nenhuma posição, os bits mudarão de 0 para 1 ou vice - versa, nesta geração.

Seguindo-se às operações de reprodução, *crossover* e mutação, a nova população está pronta para ser testada. Para fazer isto, simplesmente decodificam-se as novas *strings* criadas, e calcula-se os valores de performance de cada *string x*. Os resultados de uma única geração da simulação podem ser verificados à direita na continuação da tabela 2.1. Enquanto são tiradas conclusões concretas de um único teste de um processo estocástico que é, na melhor das hipóteses, um negócio muito arriscado, começaremos a ver como Algoritmos Genéticos combinam noções de alta performance para atingir melhores performances. Pode-se notar (ver tab. 2.1) como o máximo valor de performance e a média melhoraram na nova população. A média cresceu de 310 para 484 em uma única geração. A máxima performance subiu de 624 para 840 no mesmo período. Apesar da ajuda do processo aleatório causar estas felizes circunstâncias, começa-se a ver que esta melhora não acontece por pura sorte. A melhor *string* da primeira geração (11010) recebe duas cópias por sua alta (acima da

média) performance. Quando ela é combinada ao acaso com a terceira melhor *string* (01110) e é cruzado na posição 2 (novamente ao acaso), uma das *strings* resultantes (11110) prova ser uma, realmente, muito boa escolha.

Este evento é uma excelente ilustração das idéias e noções de analogia desenvolvidas nas seções precedentes. Neste caso, a boa idéia resultante é a combinação de duas noções acima da média, isto, é as *substrings* 1 1 - - - e - - - 1 1. Apesar do argumento ser um tanto heurístico, começou-se a ver como os Algoritmos Genéticos realizam uma busca robusta. Na seção seguinte o entendimento destes conceitos será expandido através da análise dos Algoritmos Genéticos em termos de schemata.

Tem-se comparado o Algoritmo Genético com um certo processo de busca humano comumente chamado de inovativo ou criativo. Além do mais, a simulação manual de um algoritmo genético simples permitiu a constatação de que algo interessante está acontecendo por aqui. Até este estágio do trabalho, preocupou-se apenas com os resultados superficiais apresentados, deixando-se de lado aspectos importantes, porém implícitos, que estão sendo processados pelos Algoritmos Genéticos. Como saber se com o processamento destes "aspectos importantes" se conseguirá atingir um ótimo em um problema particular? Para a obtenção da resposta é preciso entender como é o desempenho de um algoritmo genético.

Para obter este entendimento, examinam-se os dados disponíveis, em seu estado natural, tenta-se descobrir quais são as principais similaridades entre eles, e qual a melhor forma de se explorar estas similaridades, para se atingir melhores regiões no espaço de busca. Isto permite desenvolver a importante noção de *schema* e,

Genéticos. Como saber se com o processamento destes "aspectos importantes" se conseguirá atingir um ótimo em um problema particular? Para a obtenção da resposta é preciso entender como é o desempenho de um algoritmo genético.

Para obter este entendimento, examinam-se os dados disponíveis, em seu estado natural, tenta-se descobrir quais são as principais similaridades entre eles, e qual a melhor forma de se explorar estas similaridades, para se atingir melhores regiões no espaço de busca. Isto permite desenvolver a importante noção de *schema* e, consequentemente, a parte mais importante da abordagem do Algoritmo Genético, a *"building block hypothesis"*.

2.4 - Similaridades

Desde o princípio do trabalho tem-se ignorado uma questão fundamental. Em um processo de busca que apresenta somente dados de saída (valores de performance), quais informações estão contidas em uma população de strings e em seus valores da função objetivo para ajudar a guiar uma busca direta para o melhoramento? Para se fazer esta pergunta de forma mais clara, considere as strings e valores de performance originalmente mostrados na tabela 2.1 da simulação anterior (o problema da caixa preta), reproduzidos abaixo para maior conveniência:

String	Performance
01110	168
01000	48
10101	399
11010	624

importante ingrediente na otimização desta função? No caso da função utilizada ($f(x) = |x^2 - 2x|$) e a codificação em questão (cinco bits binários inteiros), eles certamente são.

Duas coisas estão sendo feitas, separadamente, aqui. Primeiro, estão sendo buscadas similaridades entre *strings* na população. Segundo, procura-se por relações casuais entre estas similaridades e alta performance. Fazendo-se isto, admite-se uma riqueza de novas informações para ajudar a guiar uma busca. Para ver o quanto, quão precisamente e quais informações admitir, deve-se considerar o importante conceito de um schema (plural = schemata).

2.5 - Schemata

Há pouco, havia interesse simplesmente em *strings*. Desde que similaridades entre *strings* de alta performance podem ajudar a guiar uma busca, pergunta-se como uma *string* pode ser similar aos seus companheiros? Especificamente pergunta-se, de que maneira uma *string* é representativo de outras classes de *strings* com similaridades em certas posições? A estrutura dos schemata provê a ferramenta para a resposta destas questões.

Um schema (Holland, 1975) descreve um subconjunto de *strings* com similaridades em certas posições. Para esta discussão, limita-se mais uma vez, sem perda de generalidade, ao conjunto binário {0, 1}. Representa-se um schema mais facilmente através da adição de um símbolo especial a este conjunto, adiciona-se (*). Com este conjunto expandido, agora pode-se criar *strings* (schemata) sobre um conjunto ternário {0, 1, *}. Um schema é igual a uma particular *string* se em toda posição no schema um 1 é igual a um 1 na *string*, um 0 é igual a um 0, ou um * é igual

a um 1 ou um 0, de forma indiferente. Como exemplo, considere as *strings* e schemata de comprimento 5. O schema *0000 é igual a duas *strings*, isto é 10000 e 00000. Como um outro exemplo, o schema *111* descreve um subconjunto com quatro membros {01110, 01111, 11110, 11111}. Como um último exemplo, o schema 0*1** é igual a qualquer uma das oito *strings* de comprimento 5 que começam com um 0 e têm um 1 na terceira posição. A idéia de schema dá um modo compacto e poderoso de se falar sobre todas as similaridades bem definidas entre *strings* de comprimentos finitos sobre um conjunto finito. Deve-se enfatizar que o * é somente um metasímbolo (um símbolo que representa outros símbolos); ele nunca é processado pelo algoritmo genético. Ele é simplesmente uma notação que permite a descrição de todas as possíveis similaridades entre *strings* de um particular comprimento e conjunto.

Contando o número total de possíveis *schemata*, em exercício ilustrativo, tem-se:

No exemplo prévio, com $l = 5$, há $3.3.3.3.3 = 3^5 = 243$ diferentes schemata, porque cada uma das cinco posições pode ser um 0, 1 ou *. Em geral, para conjuntos de cardinalidade (número de caracteres do conjunto) k, há $(k + 1)^l$ schemata. A primeira vista, tem-se a impressão de que os schemata estão tornando a busca mais difícil. Para um conjunto com k elementos há somente (somente?) k^l diferentes *strings* de comprimento l. Por que considerar $(k + 1)^l$ schemata e aumentar o espaço de interesse? O exemplo de comprimento 5 tem somente $2^5 = 32$ diferentes alternativas de *strings*. Porque fazer uma questão mais difícil considerando-se $3^5 = 243$ schemata? De fato, as razões discutidas na seção anterior tornam as coisas mais fáceis . Retomando-se a lista de quatro *strings* e seus valores de performance, caso se considere as *strings* separadamente, então tem-se somente quatro pedaços de informação; contudo quando

se consideram as *strings*, seus valores de performance, e as similaridades entre eles na população, admite-se a riqueza de novas informações para ajudar diretamente na pesquisa. Quanta informação é admitida considerando-se as similaridades? A resposta para esta pergunta está relacionada com *schemata* únicos contidos na população. Contar exatamente esta quantidade requer conhecimento das *strings* em uma particular população.

Considerando-se uma única *string* de comprimento 5: 11111, por exemplo, ele é um membro de 2^5 schemata, pois cada posição pode ser tomada com seu valor real, ou um símbolo *. Em geral uma *string* tem 2^l schemata. Como resultado, uma população de tamanho n contém algo entre 2^l e $n.2^l$ schemata, dependendo da diversidade da população.

Dos 2^l a $n.2^l$ schemata contidos em uma população, quantos são realmente processados de forma útil pelo algoritmo genético? Para obter-se a resposta desta questão, é necessário a consideração do efeito da reprodução, crossover e mutação no crescimento ou decaimento de importantes schemata de geração a geração. O efeito de reprodução em um determinado schema é fácil de se determinar, desde que com melhor performance há maior probabilidade de seleção, em média é dado um crescente número de amostras para os melhores tipos de similaridades observados. *Crossover* pode deixar um schema ileso, se a posição de *crossing over* for interior a duas posições específicas (0 ou 1) do schema, e pode destruí-lo em caso contrário. Por exemplo, considerando-se os dois schemata 1***0 e **11*, o primeiro mostra-se favorável à ruptura através do *crossover*, enquanto o segundo é mais difícil de ser destruído. Como resultado, schemata de curto comprimento definido (distância entre a primeira e a última posição específica da *string*) são deixados intactos pelo *crossover* e

reproduzidos com uma boa taxa de amostragem pelo operador reprodução. Mutação, com baixas taxas, em geral, não promovem a ruptura de um schema. Chega-se aqui, a uma importante conclusão:

Alta performance, schemata de curto comprimento definido (chamados "building blocks") são propagados geração a geração, dando amostras exponencialmente crescentes dos melhores observados.

O capítulo 3 apresenta a demonstração matemática, assim como uma análise detalhada da conclusão anterior.

2.6 - Aprendendo a Linguagem do Algoritmo Genético

É importante conhecer a terminologia normalmente usada por pesquisadores que trabalham com algoritmos genéticos, porque eles direcionam-se tanto para a genética natural quanto para as ciências da computação. A terminologia usada na literatura é uma mistura do natural e do artificial. Até aqui tem-se citado apenas *strings*, bits, conjuntos e assim por diante. Será feito uma revisão entre estes termos e seus correspondentes na genética natural devido a necessidade apresentada no estudo da literatura dos Algoritmos Genéticos.

As *strings* dos sistemas genéticos artificiais são análogos aos cromossomos nos sistemas biológicos. Nos sistemas naturais, os cromossomos são formados por genes, sendo que um gene é um segmento de DNA que tem a capacidade de armazenar e transportar a informação genética para um *caráter biológico* (qualquer aspecto anatômico-fisiológico de um ser vivo), de uma célula para outra e de uma geração para outra. Um ou mais cromossomos se combinam para formar a prescrição genética total para a construção e operação do organismo. Nos sistemas naturais, a bagagem

genética ou hereditária do indivíduo é chamada *genótipo.* Nos sistemas genéticos artificiais o pacote total de *strings* é chamado de *estrutura.*

Nos sistemas naturais, a expressão do genótipo associada à ação do meio ambiente (manifestação do caráter biológico) é chamada *fenótipo.* Por exemplo, a quantidade de pigmento na pele humana é determinada geneticamente, mas depende também da quantidade de radiação ultravioleta à qual a pele está exposta. Caracteres como a estatura e o peso são também genéticos, porém o fenótipo definitivo dependerá obviamente de fatores tais como a nutrição. Há, no entanto, caracteres que não sofrem influência do ambiente, como por exemplo, o tipo sangüíneo. Nos sistemas genéticos artificiais, as *estruturas* são decodificadas para formar um particular *conjunto de parâmetros*, *soluções alternativas* ou *pontos* (no espaço de solução).O sistema genético artificial tem uma variedade de alternativas para codificar parâmetros tanto numéricos quanto não-numéricos.

Na terminologia natural, genes para um mesmo caráter biológico, localizados no mesmo "locus" em um par de cromossomos homólogos são chamados *genes alelos.* É importante notar que a posição do gene ("locus") é identificada separadamente da sua função, isto é, pode-se falar sobre um particular gene, por exemplo, o gene que determina a cor dos olhos de um animal, sendo seu "locus" = 10 e seu valor alelo = olhos azuis. Em genética artificial, *strings* são compostas de *características* ou *detectores*, os quais tomam diferentes valores. Determinadas características podem ser localizadas em diferentes *posições* na *string.* Por exemplo, a *string* 10000 é codificado como um binário inteiro sem sinal, 16 na base 10, porque implicitamente o 1 está na posição de valor 16.

Tabela 2.2 - Comparação entre as terminologias naturais e do Algoritmo Genético

Natural	***Algoritmo Genético***
cromossomo	*string*
gene	bit, característica, caráter, ou detector
alelo	valor da característica
locus	posição na *string*
genótipo	estrutura
fenótipo	conjunto de parâmetros, solução alternativa, uma estrutura decodificada

3 - Fundamentos Matemáticos

No capítulo anterior o Algoritmo Genético foi apresentado, assim como seus mecanismos de otimização e seu poder, de forma quase intuitiva, apelando para o senso de pesquisa e descoberta humana. Todavia, nesta etapa do trabalho, é necessário retornar a todos os conceitos apresentados sobre o Algoritmo Genético, utilizando fundamentos matemáticos que comprovem tudo que a intuição tem mostrado até aqui.

Na realidade, já se iniciou uma mais rigorosa avaliação do Algoritmo Genético. Na parte final do capítulo 2, foi introduzido o conceito fundamental de um *schema.* Quantitativamente, determinou-se que havia um número realmente muito grande de similaridades a serem exploradas em uma população de *strings*. Intuitivamente viu-se como os AGs exploram em paralelo as muitas similaridades contidas em *"building blocks"* (*schemata* curtos de alta performance). Serão feitas agora observações mais rigorosas. Primeiro conta-se os *schemata* representados dentro de uma população de *strings* e como ocorre o crescimento e o decaimento destes, durante uma dada geração. Para tanto considera-se o efeito de reprodução, *crossover* e mutação em um particular *schema*. Esta análise leva ao teorema fundamental dos Algoritmos Genéticos, que quantifica as taxas de crescimento e decaimento de forma mais precisa.

3.1 - O Teorema Fundamental

A operação dos algoritmos genéticos é notavelmente honesta. Começa-se com uma população aleatória de *n strings*, copiam-se *strings* com alguma tendência em direção ao melhor, efetuam-se cruzamentos e troca parcial de material de *substrings*, e mutação em valores ocasionais de *bits* para uma boa medida de performance. Apesar de os algoritmos genéticos manipularem diretamente uma população de *strings* de maneira honesta, no capítulo 2, reconhece-se que este processamento explícito de *strings* realmente causa o processamento implícito de muitos *schemata* durante cada geração. Para analisar o crescimento e o decaimento de muitos *schemata* contidos em uma população, são necessárias algumas simples notações para dar maior rigor à discussão. Serão consideradas as seguintes operações nos *schemata* contidos na população: reprodução, *crossover* e mutação.

Serão consideradas *strings*, sem perda de generalidade, conjuntos a serem construídos a partir de um conjunto binário V = {0, 1}. Como uma conveniência notacional, as *strings* serão representadas por letras maiúsculas e os caracteres individuais, por letras minúsculas, sendo a posição subscrita. Por exemplo, a *string* de 7 *bits*, A = 1001100, pode ser representada simbolicamente da seguinte forma:

$$A = a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7$$

Cada um dos a_i representam uma única característica binária ou detector (de acordo com a analogia natural, são algumas vezes chamados de genes a_i's), onde cada

característica pode ter um valor 1 ou 0 (algumas vezes os a_i são chamados de valores alelos). Na particular *string* 1001100, tem-se:

$a_1 = 1$; $a_2 = 0$; $a_3 = 0$; etc...

É possível ter-se *strings* onde os detectores não são ordenados seqüencialmente como na *string* A . Por exemplo, uma *string* A' poderia ter a seguinte ordem:

$A' = a_6\ a_4\ a_2\ a_1\ a_3\ a_5\ a_7$.

A busca genética considerada requer uma população de *strings*, onde são consideradas *strings* individuais A_j (j = 1, 2, 3, ..., n), contidos na população **A**(t) no tempo (ou geração) t, onde o negrito é usado para denotar uma população.

Ao lado de notações para descrever populações, *strings*, posição dos *bits* e alelos, é necessário uma notação conveniente para descrever os *schemata* em *strings* individuais e populações. Considerando-se um esquema H, tomado a partir de um conjunto de 3 elementos V+ = {0, 1, *}. Como discutido em capítulo prévio, o asterisco (*) é um símbolo que tanto representa um 0 quanto um 1, em uma posição particular. Por exemplo, considere o *schema* de comprimento 7:

H = 10**1**

Note que a *string* A = 1001100 discutida acima é um exemplo do *schema* H, porque os alelos a_i são iguais aos alelos h_i nas posições fixas 1, 2 e 5.

Dos resultados do capítulo anterior, alguns são extremamente importantes nesta fase do trabalho. Há 3^l *schemata* ou similaridades definidos a partir de uma *string* binário de comprimento *l*. Em geral, para conjuntos de cardinalidade *k*, há $(k + 1)^l$ *schemata*. Além do mais é necessário lembrar que em uma população de *strings* com *n* membros há pelo menos $n.2^l$ *schema* contidos na população porque cada *string*

é, ele próprio, um representante dos 2^l *schemata*. Estes argumentos de contagem dão algum sentimento sobre a magnitude de informação sendo processada pelos algoritmos genéticos; contudo, para realmente se entender os importantes *building blocks* de futuras soluções, é necessário que se perceba as diferenças entre os tipos de *schemata*.

Os *schemata* não são criados iguais, alguns são mais específicos que outros. Por exemplo, o *schema* 10**1** é mais específico sobre importantes similaridades que o schema 1******. Além do mais, certos *schemata* têm uma distância maior entre elementos significativos que outros. Por exemplo, o *schema* 1****0* tem uma maior distância entre elementos fixos (a_1 e a_6) que o *schema* 1*0**** (a_1 e a_3). Para quantificar estas idéias, serão introduzidas duas importantes propriedades dos *schemata*: "*schema order*" e "*defining length*".

A ordem do *schema* H, representado por o(H), é simplesmente o número de posições fixas (em um conjunto binário, o número de 1's e 0's) presentes no *schema*. Por exemplo a ordem do *schema* 10**1** é 3 (simbolicamente, o(10**1**) = 3), como outro exemplo a ordem do *schema* 1****** é 1.

O *defining length* do *schema* H, representado por $\delta(H)$, é a distância entre a primeira e a última posição específica da *string*. Por exemplo, o *schema* 10**1** tem um *defining length* $\delta = 4$, porque a última posição específica é 5 e a primeira posição específica é 1, logo a distância entre eles é $\delta(H) = 5 - 1 = 4$. No outro exemplo (o *schema* 1******), o *defining length* é particularmente fácil de ser calculado, uma vez que há somente uma única posição fixa, a primeira e a última posição específica são iguais, logo $\delta = 0$.

Os *schemata* e suas propriedades são interessantes estratagemas notacionais para discussões rigorosas e classificação das similaridades entre *strings*. Mais que isto,

eles fornecem meios básicos para analisar os efeitos de rede, da reprodução e demais operadores genéticos, nos *building blocks* contidos dentro da população.

O efeito da reprodução no número esperado de *schemata* na população é particularmente fácil de se determinar. Supondo-se que em um dado tempo t há m elementos de um particular *schema* H contido dentro da população $\mathbf{A}(t)$, onde $m = m(\mathrm{H}, t)$ (há possivelmente diferentes quantidades de diferentes *schemata* H em diferentes tempos t). Durante a reprodução, uma *string* é copiada de acordo com a sua performance, ou mais precisamente uma *string* A_i é selecionada com uma probabilidade $p_i = f_i/\Sigma f_i$. Após selecionar uma população, sem sobreposição, com tamanho n, com reposição de indivíduos, de uma população $\mathbf{A}(t)$, espera-se ter $m(\mathrm{H}, t + 1)$ representantes do *schema* H na população no tempo $t + 1$, como dado na equação abaixo:

$$m(H,t+1) = m(H,t)n\frac{f(H)}{\sum f_i} \tag{3.1}$$

onde $f(\mathrm{H})$ é a média das performances das *strings* representantes do *schema* H no tempo t.

Reconhecendo-se que a média das performances de toda a população pode ser escrita como:

$$\overline{f} = \frac{\sum f_i}{n} \tag{3.2}$$

pode-se então, reescrever o crescimento reprodutivo do *schema*, utilizando-se as equações acima, da seguinte forma:

$$m(H,t+1) = m(H,t)\frac{f(H)}{\overline{f}} \tag{3.3}$$

Um particular *schema* cresce de forma diretamente proporcional à média das performances do *schema*, e de forma inversamente proporcional à média das performances da população. Em outras palavras, *schemata* com valores de performance acima da média da população receberão um incremento no número de amostras na próxima geração, enquanto *schemata* com valores de performance abaixo da média da população receberão um menor número de amostras. É interessante notar que este comportamento esperado é observado em todo *schema* H contido em uma particular população **A** em paralelo. Ou seja, todos os *schemata* em uma população crescem ou decaem de acordo com a média do *schema* sob a operação de reprodução.

Pode-se aprofundar mais sobre a forma matemática do crescimento ou decaimento do *schema* na população. Supondo-se que um determinado *schema* H esteja acima da média da população em um valor $c\overline{f}$, onde c é um valor constante. A partir da equação (3.3), obtém-se:

$$m(H,t+1) = m(H,t)\frac{(\overline{f}+c\overline{f})}{\overline{f}} = (1+c)m(H,t) \tag{3.4}$$

Iniciando-se em $t = 0$ e assumindo c constante, obtém-se:

$$m(H, t) = m(H, 0)(1 + c)^t \tag{3.5}$$

Reconhece-se (3.5) como uma progressão geométrica ou uma forma discreta de uma equação exponencial. O efeito da reprodução é agora quantitativamente claro; ela

promove um crescimento (decaimento) exponencial dos números de amostras de *schemata* acima (abaixo) da média.

Serão verificadas agora os efeitos do *crossover* e da mutação. Até certo ponto é curioso que o operador reprodução possa alocar exponencialmente números crescentes e decrescentes de *schemata* para futuras gerações em paralelo; são feitas amostras de muitos *schemata* diferentes em paralelo de acordo com as mesmas regras através do uso de n operações simples de reprodução. Por outro lado, a reprodução sozinha não faz nada para promover a exploração de novas regiões no espaço de busca, se somente velhas estruturas são copiadas sem alteração, então como algo novo pode ser tentado? Exatamente neste ponto se observa o trabalho do *crossover*. *Crossover* é uma troca estruturada e aleatória de informações entre *strings*. Ele cria novas estruturas com um mínimo de ruptura para a alocação estratégica ditada pelo operador reprodução.

Para saber quais *schemata* são afetados por *crossover* e quais não são, considera-se uma *string* particular de comprimento $l = 7$ e dois *schemata* representativos deste:

A = 1 0 0 1 1 0 0

H_1 = * 0 * * * * 0

H_2 = * * * * 1 0 *

Para se notar o efeito do *crossover* nos *schemata*, é necessário primeiro recordar que o *crossover* simples seleciona aleatoriamente um par, assim como a posição do *crossover*, e a troca da *substring* se dá a partir da posição do *crossover* até o final da *string* do par escolhido. Supondo-se que a *string* A tenha sido escolhido e que se jogue

um dado para a escolha da posição do *crossover* (há 6 diferentes posições em uma *string* de comprimento 7). Supondo-se também que a jogada do dado tenha apresentado como resultado o número 3, o que significa que a posição do *crossover* se dará entre as posições 3 e 4 da *string*. O efeito desta operação nos dois *schemata* H_1 e H_2 pode ser visto facilmente no exemplo a seguir, onde a posição do *crossover* é marcada com o símbolo separador | :

A = 1 0 0 | 1 1 0 0

H_1 = * 0 * | * * * 0

H_2 = * * * | * 1 0 *

A menos que o par da *string* A seja idêntico a ela nas posições fixas do *schema* (uma possibilidade que será ignorada) o *schema* H_1 será destruído, porque o 0 na posição 2 e o 0 na posição 7 serão enviados para diferentes descendentes (eles estão em lados opostos do símbolo separador que marca o ponto de corte da *string*). Está igualmente claro que com o mesmo ponto de corte, o *schema* H_2 sobreviverá, porque o 1 na posição 5 e o 0 na posição 6 serão levados intactos para um único descendente. Apesar de ter sido usado um ponto específico para ilustração do exemplo, fica claro que o *schema* H_1 tem menor probabilidade de sobreviver ao *crossover* que o *schema* H_2, porque em média o ponto de corte cairá mais facilmente entre os extremos das posições fixas. Para quantificar esta observação, nota-se que o *schema* H_1 tem um *defining length* de 5 ($\delta(H) = 5$). Se a posição de *crossover* é selecionada aleatoriamente de forma uniforme entre as 6 ($l - 1 = 7 - 1 = 6$) possíveis posições, então claramente o *schema* H_1 é destruído com a seguinte probabilidade:

$$p_d = \frac{\delta(H_1)}{(l-1)} = \frac{5}{6}$$

A probabilidade de sobrevivência é dada por:

$$p_s = 1 - p_d = \frac{1}{6}$$

De forma similar, o *schema* H_2 tem um *defining length* $\delta(H_2) = 1$, ou seja, ele é destruído em apenas 1 evento entre 6, onde a posição de corte é selecionada para ocorrer entre as posições 5 e 6, logo $p_d = 1/6$ ou a probabilidade de sobrevivência é $p_s = 1 - p_d = 5/6$.

De forma geral, pode-se calcular a probabilidade de sobrevivência para todo *schema*. Porque um *schema* sobrevive quando a posição de cross cai fora do *defining length*, a probabilidade de sobrevivência sob um *crossover* simples é:

$$p_s = 1 - \frac{\delta(H)}{(l-1)} \tag{3.6}$$

desde que o *schema* sofre ruptura sempre que uma posição dentro do *defining length* é selecionada de $l-1$ possíveis posições. Se o próprio *crossover* é feito através de uma escolha aleatória, com uma probabilidade p_c em um particular par, a probabilidade de sobrevivência é dada pela expressão:

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{l-1} \tag{3.7}$$

a qual se reduz a expressão (3.6) quando $p_c = 1.0$.

O efeito combinado da reprodução e *crossover* pode ser agora considerado. Como quando foi considerado somente reprodução, havia o interesse em calcular o

número de um particular *schema* H esperado na próxima geração. Assumindo-se independência entre as operações de reprodução e *crossover*, obtém-se:

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\overline{f}} \left[1 - p_c \cdot \frac{\delta(H)}{l-1} \right] \tag{3.8}$$

Comparando-se esta expressão com a de número (3.3), somente para o operador reprodução, o efeito combinado do *crossover* e reprodução é obtido através da multiplicação do número esperado do *schemata*, somente para reprodução, pela probabilidade de sobrevivência sob *crossover*, p_s. O *schema* H cresce ou decai dependendo de um fator de multiplicação. Com ambos, *crossover* e reprodução, este fator depende de duas coisas: se o *schema* é acima ou abaixo da média da população e se o *schema* tem relativamente curto ou longo *defining length*. Claramente, aqueles *schemata* com ambos, performance observada acima da média e curto *defining length*, apresentarão amostras com proporções exponencialmente crescentes.

O último operador a ser considerado será mutação. Utilizando as definições prévias, mutação é uma alteração aleatória de uma única posição com probabilidade p_m. Para que um *schema* H sobreviva, todas as posições especificadas devem sobreviver. Além disto, desde que um único alelo sobreviva com probabilidade (1 - p_m), e desde que cada uma das mutações é estatisticamente independente, um particular *schema* sobrevive quando cada uma das *o(H)* posições fixas dentro do *schema* sobrevive. Multiplicando a probabilidade de sobrevivência (1 - p_m) por ela mesma *o(H)* vezes, obtém-se a probabilidade de sobrevivência, considerando-se somente a mutação, $(1 - p_m)^{o(H)}$. Para pequenos valores de p_m ($p_m << 1$), a probabilidade de sobrevivência do *schema* pode ser linearizada pela expressão $(1 - o(H) \cdot p_m)$. Desta forma pode-se concluir que um particular *schema* H recebe um número esperado de cópias na

próxima geração sob reprodução, *crossover* e mutação como é dado na seguinte equação (ignorando-se termos muito pequenos, quando comparados com os demais):

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \frac{\delta(H)}{l-1} - o(H) p_m \right] \tag{3.9}$$

A adição da mutação promove uma pequena alteração nas conclusões prévias. *Schemata* acima da média da população, curto e de baixa ordem, recebem amostras exponencialmente crescentes em gerações subsequentes. Esta é uma conclusão muito importante, tanto é que recebe um nome especial: O *Teorema do Schema* ou o *Teorema Fundamental dos Algoritmos Genéticos.* Embora os cálculos que levaram à prova do Teorema do *Schema* não foram muito trabalhosos, as implicações do Teorema são muito mais extensas e sutis. Pode-se examinar o efeito dos três operadores nos *schemata*, dentro de uma população, através da retomada do exemplo da simulação manual do algoritmo genético (capítulo 2).

3.2 - Processamento de um Schema: um exemplo manual

No capítulo 2 foi demonstrado os mecanismos de um Algoritmo Genético simples através do cálculo manual de uma única geração. Pretende-se retornar àquele exemplo, sendo que no momento será mostrado como o Algoritmo Genético processa os *schemata* (não *strings* individuais) dentro da população. O cálculo manual do capítulo anterior está reproduzido na tabela 3.1 . Além da informação apresentada anteriormente, será adicionado o processamento de três *schemata* particulares, chamados H_1, H_2 e H_3, onde:

$H_1 = 1****$

$H_2 = *10**$

$H_3 = 1***0$.

Pode-se observar os efeitos da reprodução, *crossover* e mutação no primeiro *schema*, H_1. Durante a fase de reprodução, as *strings* são copiadas probabilisticamente de acordo com os seus valores de performance. Olhando-se para a primeira coluna da tabela nota-se que as *strings* 3 e 4 são ambas representativas do *schema* 1****. Após a reprodução, verifica-se que foram produzidas três cópias do *schema* (*strings* 2, 3, 4 na coluna do "*Mating Pool*"). Este valor corresponde ao valor esperado pelo Teorema do *Schema*? Pelo teorema espera-se ter :

$$m \cdot \frac{f(H)}{\overline{f}} \text{ cópias.}$$

Calculando a performance média do *schema* ($f(H_1)$), obtém-se:

$$\frac{(399 + 624)}{2} = 511.5$$

Dividindo-se este valor pela média da população ($\overline{f} = 310$) e multiplicando-se pelo número de *schemata* H_1 no tempo t ($m(H_1, t) = 2$), obtém-se o número esperado de *schemata* H_1 no tempo $t + 1$ (3.3):

$$m(H, t+1) = 2 \cdot \frac{511.5}{310} = 3.30$$

Comparando-se este valor com o número real de *schemata* (três), percebe-se que o número de cópias está correto. Avançando-se um passo, nota-se também que o *crossover* não tem nenhum efeito neste *schema* porque um *defining length* $\delta(H_1) = 0$ prevê disruptura de um único bit. Além do mais, com a probabilidade de mutação dada ($p_m = 0.001$) espera-se que se tenha $m \cdot p_m = 3 \cdot 0.001 = 0.003$, ou seja nenhum bit muda dentro das cópias dos três *schemata* nas três *strings*. Como resultado, observa-se que para o *schema* H_1, obtém-se o esperado, número exponencialmente crescente de *schemata*, como foi predito no Teorema do *Schema*.

Tabela 3.1 - O Algoritmo Genético processando *schemata* através da simulação manual

String No.	População inicial	valor de x	$f_i = f(x) = \lvert x^2 - 2x \rvert$	$pselect_i$ $\frac{f_i}{f_t}$	Seleção Esperada $\frac{f_i}{\bar{f}}$	Seleção Real (*Roleta da sorte*)
1	0 1 1 1 0	14	168	0.14	0.54	1
2	0 1 0 0 0	8	48	0.04	0.16	0
3	1 0 1 0 1	21	399	0.32	1.29	1
4	1 1 0 1 0	26	624	0.50	2.01	2
Soma (f_t)			1239	1.00	4.00	
Média ($\bar{f}$)			310	0.25	1.00	
Máximo			624	0.50	2.01	

Processamento dos *schemata*			
Antes da Reprodução			
		Strings Representativas	Mérito médio do *Schema* -f(H)
H_1	1 * * * *	3,4	511.5
H_2	* 1 0 * *	2,4	336
H_3	1 * * * 0	4	624

Tabela 3.1 (Continuação)

Mating Pool depois da reprodução (| = posição de Troca)	Cruzamento (seleção aleatória)	Posição do *crossover* (seleção aleatória)	Nova População	Valor de x	$f_i = f(x) = \lvert x^2 - 2x \rvert$
0 1 | 1 1 0	1	2	0 1 0 1 0	10	80
1 1 | 0 1 0	2	2	1 1 1 1 0	30	840
1 0 1 | 0 1	4	3	1 0 1 1 0	22	440
1 1 0 | 1 0	3	3	1 1 0 0 1	25	575
Soma (f_t)					1935
Média ($\bar{f}$)					484
Máximo					840

Processamento dos *Schemata*					
Depois da Reprodução			Depois de todas as operações		
Valor Esperado	Valor Real	Strings Representativas	Valor Esperado	Valor Real	Strings Representativas
3.30	3	2,3,4	3.30	3	2,3,4
2.17	2	2,4	1.63	2	1,4
2.01	2	2,4	0.0	2	2,3

Até aqui houve sucesso, uma vez que a simulação manual do algoritmo no processamento dos schemata apresentou o resultado que era esperado pelo Teorema do *Schema*, mas o *schema* H_1 com um único bit fixo, dá a impressão de um caso especial. O que acontece com a propagação de importantes similaridades com longos *defining length*? Por exemplo considerando-se a propagação do *schema* H_2 = *10** e do *schema* H_3 = 1***0. O caso de H_2 começa com dois exemplos na população inicial e termina com duas cópias após a reprodução. Isto concorda com o esperado número de cópias, $m(H_2) = 2 \cdot 336/310 = 2.17$, onde 336 é a média do *schema* e 310 é a média do fitness da população. O caso de H_3 começa com um único exemplo (string 4) e termina com duas cópias após a reprodução (strings 2 e 4 na coluna de strings após reprodução). Isto concorda com o número esperado de cópias, $m(H_3) = 1 \cdot 624/310 = 2.01$, onde 624 é a média da performance do *schema* e 310 é a média da performance da população. As circunstâncias que se seguem ao *crossover* são um pouco diferentes. Nota-se que para o *schema* curto, *schema* H_2, as duas cópias são mantidas mesmo com a ocorrência do *crossover*. Por causa do *defining length* curto, espera-se que o *crossover* interrompa o processo somente uma vez em quatro (l - 1 = 5 - 1 = 4). Como resultado o *schema* H_2 sobrevive com alta probabilidade. O número esperado de *schemata* H_2 é $m(H_2, t+1) = 2.17 \cdot 0.75 = 1.63$, (3.8), que é uma boa comparação com o número de representantes do *schemata* obtido na simulação. H_3 , devido ao seu longo *defining length* ($\delta(H_3)$ = 4), usualmente é destruído pelo *crossover*, $m(H_3, t+1) = 2.01 \cdot 0 = 0$.

O cálculo manual acaba de confirmar: *Schemata* curtos e de baixa ordem recebem números exponencialmente crescentes ou decrescentes de amostras,

dependendo da média da performance do *schema*, quando comparada com a média da população.

4 - A Formulação do Problema de Otimização

O capítulo 5 tem o objetivo de aplicar o Algoritmo Genético a alguns exemplos simples de engenharia, que já foram previamente estudados, e comparar os resultados apresentados com os obtidos através da aplicação de outros métodos de otimização.

Antes, porém, da aplicação do algoritmo aos referidos exemplos, faz-se necessária a familiarização com o problema de programação matemática, assim como com os métodos que serão empregados na resolução desses exemplos.

Qualquer problema de programação matemática pode ser formulado de maneira única, diga-se *standard*, o que, sob o ponto de vista de aplicação, é particularmente desejável, pois seus métodos de solução podem ser utilizados em diferentes problemas sem quaisquer modificações.

4.1 - Variáveis de projeto

As variáveis de projeto ou variáveis de decisão são aquelas que caracterizam o projeto e precisam ser definidas pelo engenheiro ou projetista. No projeto estrutural de um navio, elas podem incluir o módulo de secção de um reforçador, a espessura de uma unidade de chapeamento, um espaçamento de cavernas, etc. Agrupam-se em um vetor, constituído de n variáveis independentes, que pode ser definido como

$$X = (x_1, x_2, x_3, \ldots x_n)^T \tag{4.1}$$

Geralmente, estas variáveis são consideradas determinísticas, isto é, não estão sujeitas a uma distribuição de probabilidade. Podem ter um espectro contínuo ao longo de um intervalo, como no caso da largura de flange de uma viga fabricada, ou podem

estar restritas a valores discretos, como no caso do número de reforçadores em um painel reforçado. O procedimento usual para o tratamento de variáveis discretas é assumir que elas são contínuas, em uma primeira abordagem, e posteriormente pesquisarem-se os projetos com os valores discretos mais próximos à solução contínua anteriormente encontrada. Porém, há consenso entre os pesquisadores nesse exemplo simples que, em muitos casos, o ótimo discreto pode estar longe do ótimo contínuo arredondado para o discreto, veja Fu et al. (1991).

4.2 - Função Objetivo

A função objetivo, ou função de custo, ou medida de mérito, é uma função escalar das variáveis de projeto definida, genericamente, como

$$f(\mathrm{X}) = f(x_1, x_2, \ldots, x_n)^T \tag{4.2}$$

e é a função a ser otimizada.

A função objetivo pode ser uma simples equação linear envolvendo as variáveis de decisão x_i, ou outra qualquer. Para a maioria dos problemas estruturais práticos, no entanto, a função objetivo é não linear e não explícita, pois os modelos de projeto não só envolvem equações como também tabelas ou outras formas implícitas de sistematização.

A função de mérito nos problemas estruturais pode ser definida com base tanto no peso final da estrutura quanto no seu custo de fabricação, embora, sob o ponto de vista prático, a primeira seja de imediata quantificação e a segunda, um pouco mais complexa por envolver dados associados às variáveis de mercado.

O custo de uma construção é, geralmente, dividido em itens como custos de material, de fabricação e de pintura, entre outros. Os custos de material podem ser estabelecidos separadamente para os diferentes tipos de material utilizados na construção (como placas, perfis laminados, elementos forjados, etc.) e de diferentes tipos ou graus do material empregado. Os custos de fabricação são geralmente estimados por unidade ou por comprimento de solda, em que se aplicam corretores para contabilizar a eficiência, estimados separadamente para os diversos estágios, como fabricação, montagem, e ereção do bloco, no dique ou carreira, considerando-se também as partes planas e curvas do casco e os processos de soldagem. Dentro das operações que envolvem a fabricação são considerados o corte, a moldagem a quente e a frio, a montagem das partes, a soldagem, a inspeção e o acabamento. Finalmente, os custos de pintura que dependem, de um modo geral, da área superficial a ser coberta pelas camadas de tinta.

Como se vê, estabelecer uma função de mérito envolvendo custos não é uma tarefa elementar e, fatalmente, será característica própria de cada estaleiro. Moe (1970), em uma tentativa de quantificar o custo como medida de mérito, utilizou um modelo de custo que incluiu mais de 150 valores unitários baseados em informações detalhadas do custo de fabricação e de instalação de cada elemento estrutural em um bloco de um navio petroleiro.

Em face a esses agravantes, muitas vezes se adota, como função de mérito, ou objetivo a ser atingido durante o projeto estrutural, o mínimo peso.

4.3 - Funções de Restrição

Problemas práticos geralmente estão sujeitos a uma série de restrições de fronteira que podem ser representadas por

$$g_j(X) \geq 0 \tag{4.3}$$

para j = 1, 2, ...n_g. Estas restrições podem ser lineares ou não lineares nas variáveis x_i. Os níveis de tensões em uma estrutura, por exemplo, funções das variáveis geométricas que definem essa estrutura, devem estar restritos a um valor máximo admissível.

4.4 - Restrições Homogêneas

Problemas práticos também podem estar sujeitos a uma série de restrições homogêneas ou de igualdade

$$h_n(\mathbf{X}) = 0 \tag{4.4}$$

para j = 1, 2, ...n_h. Analogamente, tais restrições podem ser lineares ou não lineares nas variáveis x_i. Elas podem ser utilizadas para eliminar uma ou mais variáveis de decisão uma vez que as variáveis x_i não são mais independentes.

Em alguns casos, onde não se deseja trabalhar com restrições homogêneas, pode-se contornar o problema transformando-se uma restrição homogênea em duas restrições de desigualdades isto é, impondo-se $|h(X)| \leq \varepsilon$, onde ε é um número pequeno.

O problema de otimização pode, agora, ser expresso pelo ato de selecionar o vetor X_o, de variáveis de projeto, que minimizar f(X) sujeita às restrições impostas, resultando num valor ótimo para a função objetivo, $f_0(X_o)$. Usa-se, de modo geral, o

mínimo da função de mérito como seu ótimo porque a obtenção do máximo sempre pode tratada como minimização do negativo de f(X).

Neste trabalho, concentrar-se-á no desenvolvimento de um método para o qual f(X), g(X) e h(X)possam ser funções não lineares das variáveis de decisão.

4.5 - Soluções Locais x Soluções Globais

Um mínimo da função f(X) pode ser um mínimo global, isto é, o menor valor de f(X) para qualquer X que satisfaça às funções de restrição, ou um mínimo também pode ser local, isto é, o menor valor de f(X) em alguma região local viável dos vetores X.

4.6 - Classificação dos Métodos de Otimização

Os métodos de otimização podem ser classificados em indiretos, ou de gradiente, se independem da comparação direta dos valores numéricos da função de mérito calculados em dois ou mais pontos. Os métodos indiretos fazem uso de condições necessárias para que um ponto seja de mínimo ou de máximo; condições essas expressas através de relações matemáticas, que são, por sua própria natureza, indiretas. Os métodos diretos, ou de busca, pressupõem a determinação e comparação dos valores da função a otimizar em diversos pontos situados dentro do campo de definição das variáveis independentes.

Ambos, podem ser sem restrições, se não há nenhuma restrição no processo, ou com restrições, caso contrário.

Se o problema de otimização pode ser expresso completamente através de uma formulação matemática, de sorte que as derivadas da função objetivo e das funções de restrição possam ser calculadas analiticamente, então o cálculo diferencial fornece meios para se obter, analiticamente, um ponto extremo. Para problemas complexos, com restrições não lineares, típicos da maioria dos projetos estruturais, esses meios não podem ser aplicados com a precisão desejável, habilitando-se, portanto, os métodos numéricos do tipo iterativo, que geralmente requerem o emprego de um computador.

Neste trabalho apresenta-se como método alternativo para o problema da síntese de estruturas, o Algoritmo Genético, não com o intuito de concorrer com os métodos tradicionais, mas no sentido de se explorar uma ferramenta que possa ser aplicada em diversos problemas sem que se restrinja suas características quanto à continuidade, à existência de derivadas ou de qualquer outro empecilho gerado por imposições matemáticas.

4.7 - A busca com restrições

Sem muito esforço, pode-se encontrar na literatura muitos algoritmos de programação matemática que se habilitam a resolver problemas com a existência de restrições. Porém, a quase totalidade deles, exige que o usuário encarregue-se de fornecer um ponto de partida viável. Em muitos problemas não elementares, como no caso de projeto estrutural de navios, conseguir um vetor, ou seja um projeto, que satisfaça todas as restrições é uma tarefa complexa e, uma vez obtido, freqüentemente leva o engenheiro a dar por encerrado o próprio projeto.

Para contornar esse problema, Fletcher (1973), sugere que se utilize o método de busca com penalidades. No método das penalidades, define-se uma função auxiliar, que incorpora a função objetivo original e as de restrições, convertendo-se o problema com restrições num processo de otimização sem restrições.

Seja o problema de minimização como o definido anteriormente. No método das penalidades define-se uma função auxiliar $\phi(X, r_k)$

$$\phi(\mathbf{X},r_k) = f(\mathbf{X}) + p(\mathbf{X},r_k) \tag{4.5}$$

onde $p(\mathbf{X},r_k)$, é chamada de função de penalização e r_k é uma variável escalar positiva. A variável r_k é usualmente denominada de *fator de resposta* e a superfície representativa da função $\phi(\mathbf{X},r_k)$ é chamada de *superfície de resposta.*

Como na natureza há uma constante negociação entre vantagens e desvantagens, o problema, antes com restrições, se transforma em diversos problemas sem restrições, já que para diferentes valores do fator de resposta, r_k, pode-se chegar a pontos diferentes na minimização da função $\phi(X,r_k)$. O processo global de busca termina quando, entre dois valores distintos de r_k, problemas k e k+1, chega-se ao mesmo ponto ótimo de $\phi(X,r_k)$. Esse processo foi batizado de SUMT *Sequential Unconstrained Minimization Technique*, e vem sendo aplicado com sucesso em diversos problemas de engenharia, a exemplo de EXMOOR, conforme Dias et al. (1998).

Existem algumas formas de se definir a função de penalização e a escolha, por uma ou outra forma, afetará a maneira pela qual o fator de resposta deverá ser seqüencialmente avaliado e como deve ser estabelecido o ponto de partida. As usuais são

$$\phi(\mathbf{X},r_k) = f(\mathbf{X}) + r_k \sum_{j=1}^{m} \frac{1}{g_j(\mathbf{X})} \tag{4.6}$$

$$\phi(\mathbf{X},r_k) = f(\mathbf{X}) - r_k \sum_{j=1}^{m} Ln\left[g_j(\mathbf{X})\right] \tag{4.7}$$

$$\phi(\mathbf{X},r_k) = f(\mathbf{X}) - r_k \sum_{j=1}^{m} min\left[0, g_j(\mathbf{X})\right] \tag{4.8}$$

$$\phi(\mathbf{X},r_k) = f(\mathbf{X}) + r_k \sum_{j=1}^{m} min\left[0, g_j(\mathbf{X})\right]^2 \tag{4.9}$$

Obviamente, nas funções (4.6) e (4.7), o termo de penalização $p(X, r_k)$, é definido somente para valores de $g_j(X)$ não nulos e no caso de (4.7), somente para valores estritamente positivos. Essas funções são classificadas de funções de penalização interior, porque, partindo-se de um ponto viável, elas não permitem que o ponto de mínimo da função $\phi(X,r_k)$, se posicione fora da região viável. Para essas funções, a busca deve ser feita com valores de r_k decrescentes. Fletcher (1973) sugere, 0.1, 0.01, 0.001....

As funções (4.8) e (4.9) são classificadas como funções de penalização exterior, já que elas permitem que o ponto de mínimo da função $\phi(X,r_k)$ se posicione fora da região viável, o que as torna atrativas, pois livram de iniciar o processo com um ponto de partida viável. Nestes casos a busca deve ser feita para valores de r_k crescentes, ou seja, 1, 10, 100,...

Ainda em relação as equações (4.8) e (4.9), particularmente a equação (4.8), conduz a um mal condicionamento da superficie $\phi(X,r_k)$ devido à descontinuidade na fronteira com as restrições. As técnicas de otimização que dependem da continuidade da superficie $\phi(X,r_k)$ para obterem um bom desempenho, podem ser pouco eficientes

ou até falharem em situações como essa. Por outro lado, aqueles métodos que comparam valores de $\phi(X,r_k)$, em pontos no universo de busca, podem ser extremamente eficazes na solução de problemas com penalizações exteriores.

Apesar de muito difundido, no método das penalidades encontram-se algumas potenciais dificuldades. Problemas de escala adequada entre a função de mérito e as penalizações, seleção adequada do fator de resposta inicial, e a determinação da variação deste entre ciclos, são problemas para os quais não se encontraram soluções generalizadas. Escalas inadequadas, entre as penalizações ou entre a função de mérito e as de restrições, podem resultar em convergência para uma solução não ótima.

Tendo consciência de todos esses percalços, ainda assim se optou por trabalhar com penalizações, envolvendo o Algoritmo Genético que se mostrou eficiente e robusto na solução dos problemas propostos. Entre as características implantadas destacam-se: a busca com penalização exterior, evitando-se a necessidade do usuário fornecer um vetor de partida viável, a eliminação da busca seqüencial pela dinamização das modificações do fator de resposta r_k e o tratamento de variáveis contínuas e discretas.

4.8 - A Penalização

Por razões óbvias, num método com penalização é importante normalizar os valores das funções de mérito e de restrição. Quando as penalizações associadas com várias restrições são combinadas, para que as parcelas de diferentes unidades e ordem de grandeza não sejam diretamente somadas, a normalização se faz necessária e pode ser facilmente implementada adotando-se o seguinte enfoque: se uma particular função

das variáveis de projeto, por exemplo $g_j(\mathrm{x})$, deve estar restrita a um intervalo fechado $[g_{min}; g_{max}]$, então adotam-se as seguintes definições para as restrições:

$$g_j(\mathrm{X}) = 1 - \frac{\tilde{g}_j(\mathrm{X})}{g_{j\,max}} \geq 0 \quad \Leftrightarrow \quad 0 \leq \tilde{g}_j(\mathrm{X}) \leq g_{j\,max} \tag{4.10}$$

$$g_{j+1}(\mathrm{X}) = \frac{\tilde{g}_j(\mathrm{X})}{\mathrm{g_{j\ min}}} - 1 \geq 0 \quad \Rightarrow \quad \tilde{\mathrm{g}}_{\mathrm{j}}(\mathrm{X}) \geq g_{j\,min} \geq 0 \tag{4.11}$$

tendo-se, com isso, todas as restrições na forma não negativa desejada, assegurando-se sua satisfação de modo paritário.

Por outro lado, a penalização pode ser diferenciada pela definição da função de penalização na forma

$$\text{onde:} \quad \begin{aligned} p(\mathbf{X}, r_k) &= -r_k \sum_{m=1}^{n_g} < g_m(\mathbf{X}) > \\ < g_m(\mathbf{X}) > &= \begin{cases} 0, \text{ se } g_m(\mathbf{X}) \geq 0 \\ \alpha_m \left| g_m(\mathbf{X}) \right|^{\beta_m}, \text{ se } g_m(\mathbf{X}) < 0 \end{cases} \end{aligned} \tag{4.12}$$

Os coeficientes α_m e β_m permitem controlar o quanto uma violação de uma função de restrição penaliza a função de mérito. Suponha-se um projeto estrutural que tenha como restrição máxima de deslocamentos g_{max}. Na prática é usual um certo nível de tolerância na excedência, por exemplo, um projetista poderia aplicar uma pequena penalidade, diga-se p_{1m}, para um deslocamento de $g_{max} + \varepsilon_{1m}$ mas uma significativa penalidade, diga-se p_{2m}, para um deslocamento $g_{max} + \varepsilon_{2m}$. Nestas condições o par (α_m, β_m) é calculado como

$$\beta_m = \frac{\ln \frac{p_2}{p_1}}{\ln \frac{g_{max} + \varepsilon_2}{g_{max} + \varepsilon_1}}, \qquad \alpha_m = \frac{p_2}{\left(g_{max} + \varepsilon_2\right)^{\beta_m}} \tag{4.13}$$

Os valores usuais encontrados na literatura são $(\alpha_m, \beta_m) = (1,1)$, penalização linear, e $(\alpha_m, \beta_m) = (1,2)$, penalização quadrática.

5 - Testes de Confiabilidade

O Algoritmo Genético foi utilizado nos dois problemas estruturais simples descritos a seguir. O objetivo maior dos testes foi o de verificar o comportamento do algoritmo e compará-lo com outros algoritmos tradicionais de programação matemática.

Conseguiu-se uma melhor convergência na operação do AG, adaptando-se as probabilidades de mutação e *crossover* , variável por variável, de acordo com as relações:

$$p_c[i] = p_{co} + (1-p_{co})*R \tag{5.1}$$

$$p_m[i] = p_{mo}*(1-R) \tag{5.2}$$

onde:

$p_c[i]$ probabilidade de *crossover* da *string* representativa da variável *i*

$p_m[i]$ probabilidade de mutação da *string* representativa da variável *i*

p_{co} , p_{mo} valores iniciais do controle de probabilidades

R geração atual/número máximo de gerações

O principal intuito disso foi o de minimizar o efeito da mutação à medida em que as gerações vão sendo apuradas.

5.1 - Problema 1 - Tampa de Escotilha

Visando o mínimo peso, deseja-se obter as dimensões h e t_b , mostrados na figura 5.1, para a tampa de escotilha de porão de carga de uma embarcação. A largura da escotilha é $l_0 = 6.0m$ e a tampa é formada por segmentos de viga caixão, fabricada

em alumínio - módulo de elasticidade E = 70KN/mm^2, tensão de escoamento, σ_e = 70N/mm^2, coeficiente de Poisson, μ = 0.3 - e largura de segmento b = 600mm. A pressão, a ser suportada pela escotilha, é de p = 10KN/m^2. Devem ser verificadas as restrições, a seguir, mencionadas:

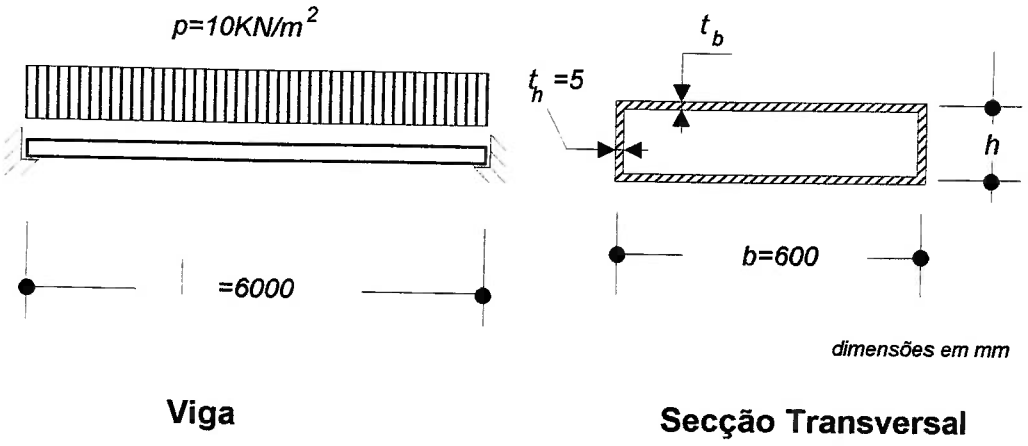


Figura 5.1 - Segmento de tampa de escotilha

A - Máximo deslocamento, $g_1(h, t_b)$:

$$\delta = \frac{5(pb)l^4}{384EI} \leq \delta_{lim} = \frac{l}{500} = 12.0mm$$

$g_1(h, t_b) = \delta_{lim} - \delta$

B - Resistência nos pontos de máxima solicitação:

B1 - flange, fibra externa, $g_2(h, t_b)$:

$$\sigma = \frac{(pb)l^2}{12W} \leq \sigma_{max} = 70MPa$$

$g_2(h, t_b) = \sigma_{max} - \sigma$

B2 - alma, linha neutra, $g_3(h, t_b)$:

$$\tau = \frac{\left(\frac{pbl}{2}\right)m}{t_h I} \leq \tau_{max} = \frac{70}{\sqrt{3}} MPa$$

$g_3(h, t_b) = \tau_{max} - \tau$

C - Não ocorrência de flambagem localizada dos flanges, $g_4(h, t_b)$:

$$\frac{b}{t_b} \leq \frac{4\pi}{\sqrt{12(1-\mu^2)}}\sqrt{\frac{E}{\sigma}}$$

$$g_4(h, t_b) = \frac{4\pi}{\sqrt{12(1-\mu^2)}} \cdot \sqrt{\frac{E}{\sigma}} - \frac{b}{t_b}, \quad \text{onde } \sigma = \frac{(pb)l^2}{12w}$$

Propriedades úteis:

- Área: $A = 2(ht_h + bt_b)$
- Inércia à flexão: $I = (t_h h + 3t_b b)\frac{h^2}{6}$
- Módulo de Resistência nos flanges: $w = 2\frac{I}{h}$
- Momento estático de área na LN: $m = \frac{1}{4}bt_b h + \frac{1}{8}h^2 t_h$
- Função de Mérito: $f(h, t_b) = A$

Por simplicidade, o peso das anteparas transversais e dos reforçadores não foram considerados no problema.

Na figura 5.2 mostram-se as funções de restrição e as curvas de nível para a superfície de resposta, utilizando-se o fator de resposta r_k = 1000, com as funções não normalizadas. Com este fator de resposta, correlacionado ao fato de as funções de mérito e de restrições não estarem normalizadas, não se garante que o mínimo da superfície de resposta se localize no interior da região viável, conforme pode-se notar na figura.

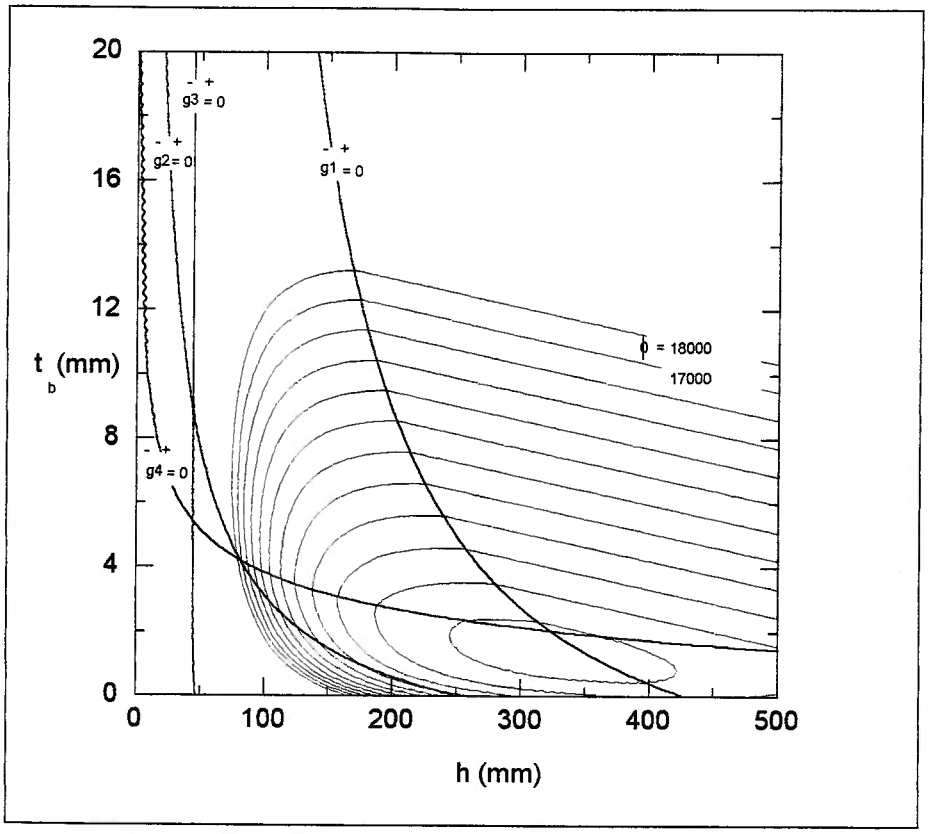


Figura 5.2 - Função de resposta [ϕ] - Tampa de Escotilha
Fator de resposta r_k =1000. Funções não normalizadas

Várias simulações foram feitas, utilizando-se o Algoritmo Genético. Algumas das quais foram selecionadas e podem ser vistas a seguir.

As figuras seguintes, apresentam os resultados obtidos, representando graficamente a performance da população em função das sucessivas gerações desta.

Nas aplicações de 1 a 3 (figuras 5.3 - 5.5) fixou-se um tamanho de população (50, 70 e 90 indivíduos, respectivamente) e para cada uma delas variou-se o número de gerações (50, 70 e 90 gerações), podendo-se assim iniciar uma análise qualitativa da influência do tamanho da população no algoritmo, o que será discutido mais adiante neste mesmo capítulo, no item 5.3 - Análise dos resultados.

Nas aplicações de 4 a 6 (figuras 5.6 - 5.8) fixou-se o número de gerações (50, 70 e 90 gerações, respectivamente) e para cada uma delas variou-se o tamanho da população (50, 70 e 90 indivíduos), o que permitiu uma analise qualitativa da influência das probabilidades de mutação e *crossover* no algoritmo, que também será discutido no item 5.3.

Observação: As funções de mérito e objetivo (performance) encontram-se normalizadas e as probabilidades de mutação e *crossover* foram escaladas, conforme as equações (5.1) e (5.2).

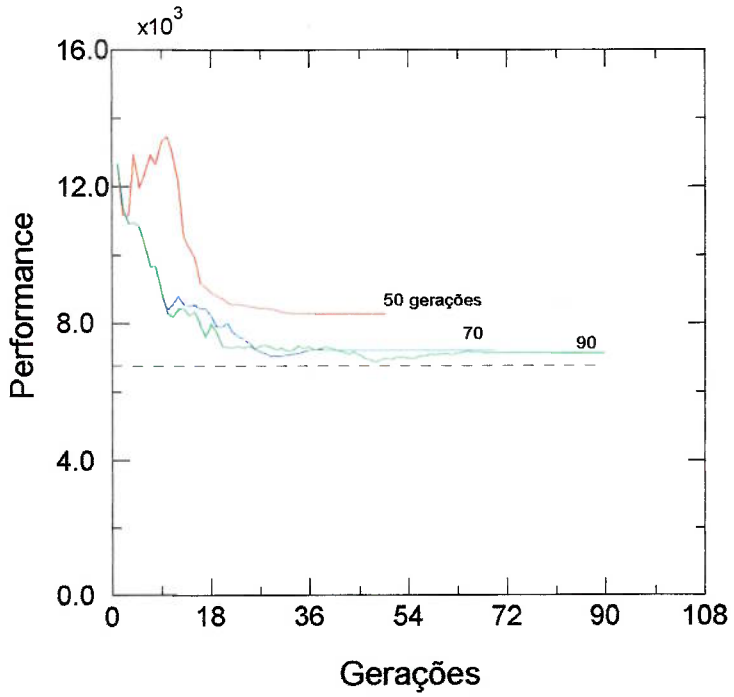


Figura 5.3 - Tampa de Escotilha - Aplicação 1
População 50 indivíduos. Fator de penalização r_k=10000.
Probabilidades escaladas. Funções normalizadas.
Algoritmos de Box e de Nelder-Mead: performance = 6757.

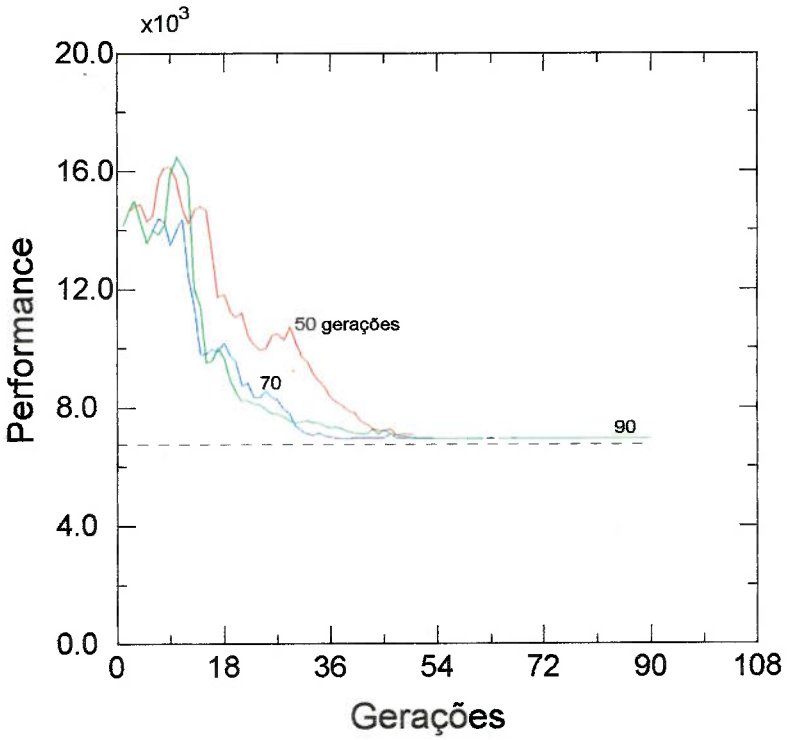


Figura 5.4 - Tampa de Escotilha - Aplicação 2
População 70 indivíduos. Fator de penalização r_k=10000.
Probabilidades escaladas. Funções normalizadas.
Algoritmos de Box e de Nelder-Mead: performance = 6757.

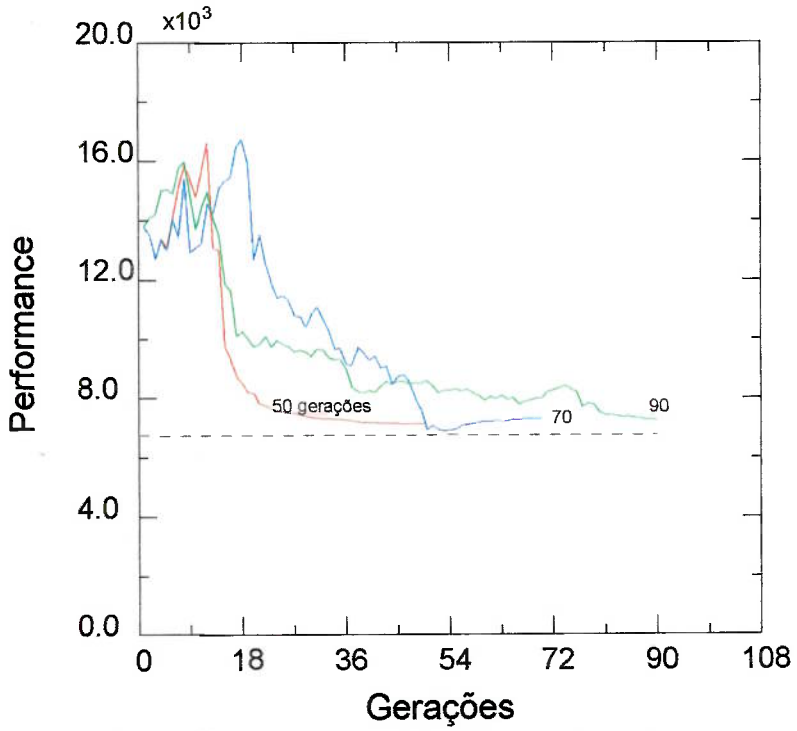


Figura 5.5 - Tampa de Escotilha - Aplicação 3
População 90 indivíduos. Fator de penalização r_k=10000.
Probabilidades escaladas. Funções normalizadas.
Algoritmos de Box e de Nelder-Mead: performance = 6757.

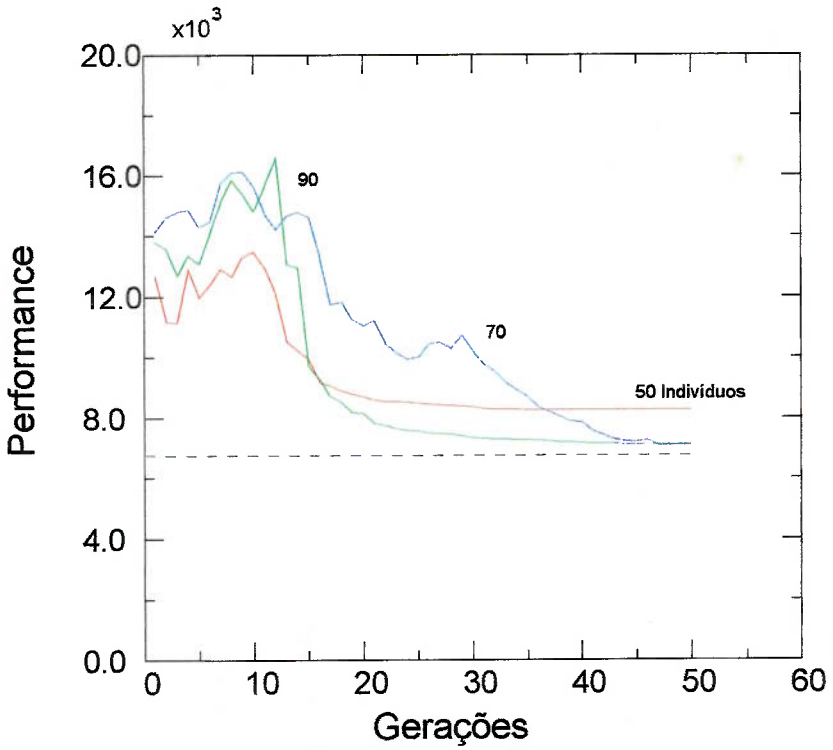


Figura 5.6 - Tampa de Escotilha - Aplicação 4
50 gerações. Fator de penalização r_k=10000.
Probabilidades escaladas. Funções normalizadas.
Algoritmos de Box e de Nelder-Mead: performance = 6757.

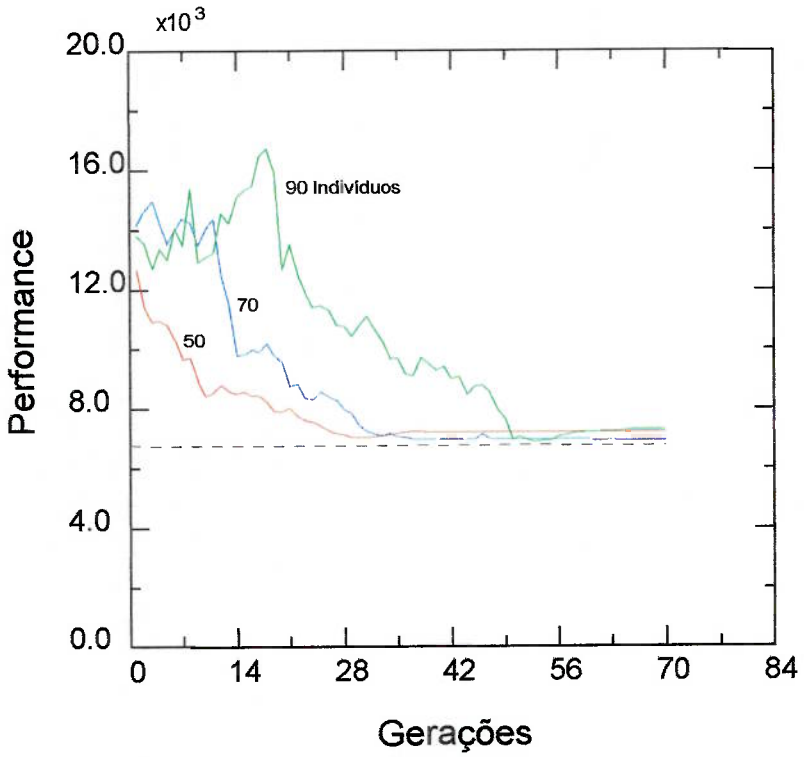


Figura 5.7 - Tampa de Escotilha -Aplicação 5
70 gerações. Fator de penalização r_k=10000.
Probabilidades escaladas. Funções normalizadas.
Algoritmos de Box e de Nelder-Mead: performance = 6757.

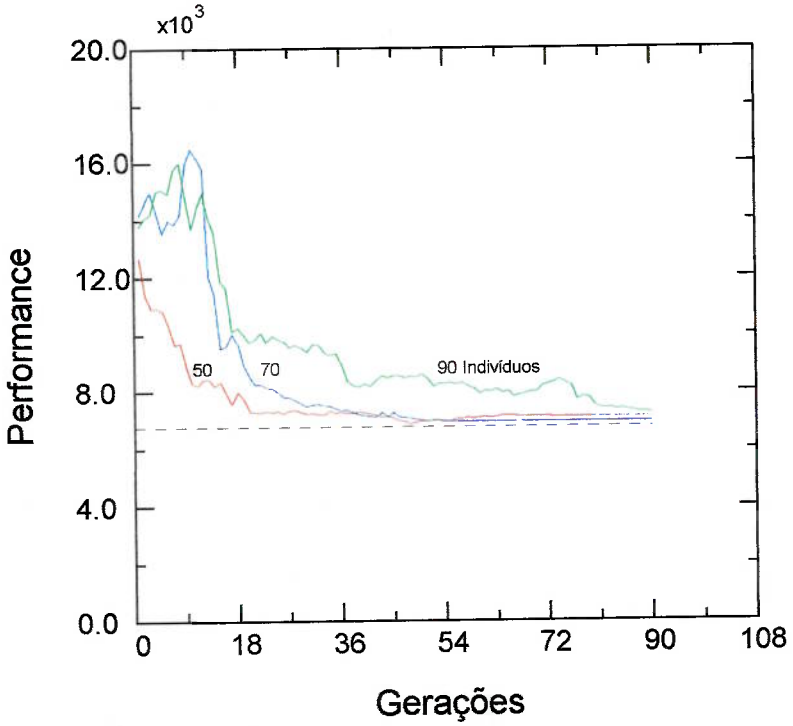


Figura 5.8 - Tampa de Escotilha -Aplicação 6
90 gerações. Fator de penalização r_k=10000.
Probabilidades escaladas. Funções normalizadas.
Algoritmos de Box e de Nelder-Mead: performance = 6757.

Tabela 5.1 - Resultados para o projeto da Tampa de Escotilha

ALGORITMO	PONTO DE PARTIDA		RESULTADOS		FUNÇÃO DE MÉRITO
	h_0	tb_0	h	tb	
AG	-	-	316.414	3.164	6961
BOX	-	-	323.886	2.932	6757
NELDER-MEAD	200	5	323.813	2.932	6757
	300	5	323.813	2.932	6757
	300	2	323.813	2.932	6757
HOOKE-JEEVES	200	5	298.276	3.688	7408
	300	5	298.276	3.688	7408
	300	2	372.704	2.750	7027

Tolerância: 10^{-5} e passo de 20 para a variável t_b , tanto para a montagem do simplex, algoritmo de Nelder-Mead, quanto passo inicial, para o algoritmo de Hooke-Jeaves. Região de busca, $h \subset [1, 500]$ e $t_b \subset [0, 20]$. O Algoritmo Genético foi rodado com uma população de 70 indivíduos e 90 gerações.

5.2 - Problema 2 - Vaso de Pressão

O outro problema refere-se à otimização do projeto estrutural do vaso de pressão mostrado na figura 5.9. Este problema foi resolvido por diferentes métodos, Fu et al. (1991), Li et al. (1994), Thierauf et al. (1997). A função objetivo combina os custos de material, de conformação e de soldagem do vaso de pressão.

O conjunto de restrições está de acordo com os respectivos códigos ASME. O problema de otimização do projeto é formulado a seguir:

- minimizar

$$f(\mathbf{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

- sujeito a

$$g_1(\mathbf{X}) = x_1 - 0.0193x_3 \geq 0$$
$$g_2(\mathbf{X}) = x_2 - 0.00954x_3 \geq 0$$
$$g_3(\mathbf{X}) = \pi x_3^2 x_4 + \tfrac{4}{3}\pi x_3^3 - 750.0 \cdot 1728.0 \geq 0$$
$$g_4(\mathbf{X}) = 240.0 - x_4 \geq 0$$
$$1.000 \leq x_1 \leq 1.375$$
$$0.625 \leq x_2 \leq 1.000$$

As variáveis de projeto x_3 e x_4 são contínuas e as variáveis x_1 e x_2 são valores discretos, múltiplos de 0.0625 in.

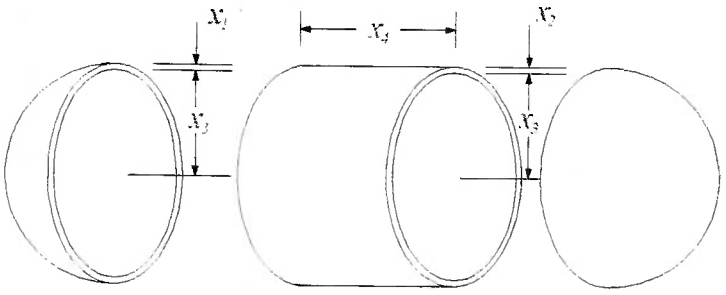


Figura 5.9 - Vaso de pressão

Utilizando-se o Algoritmo Genético, no problema em questão, pode-se efetuar inúmeras simulações. Algumas das quais foram selecionadas e podem ser vistas a seguir.

As próximas figuras, apresentam os resultados obtidos, representando graficamente a performance da população em função das sucessivas gerações da mesma.

Nas aplicações de 1 a 3 (figuras 5.10 - 5.12) manteve-se fixo o tamanho da população (80, 100 e 120 indivíduos, respectivamente) e para cada uma delas variou-se o número de gerações (80, 100 e 130 gerações), o que permitiu o inicio de uma análise qualitativa da influência do tamanho da população no algoritmo, o que será discutido no próximo item.

Nas aplicações de 4 a 6 (figuras 5.13 - 5.15) fixou-se o número de gerações (80, 100 e 130 gerações, respectivamente) e para cada uma delas variou-se o tamanho da população (80, 100 e 120 indivíduos), o que permitiu uma analise qualitativa da influência das probabilidades de mutação e *crossover* no algoritmo, que também será discutido no item 5.3.

Assim como no problema da Tampa de Escotilha, as probabilidades de mutação e *crossover* foram escaladas, conforme as equações (5.1) e (5.2).

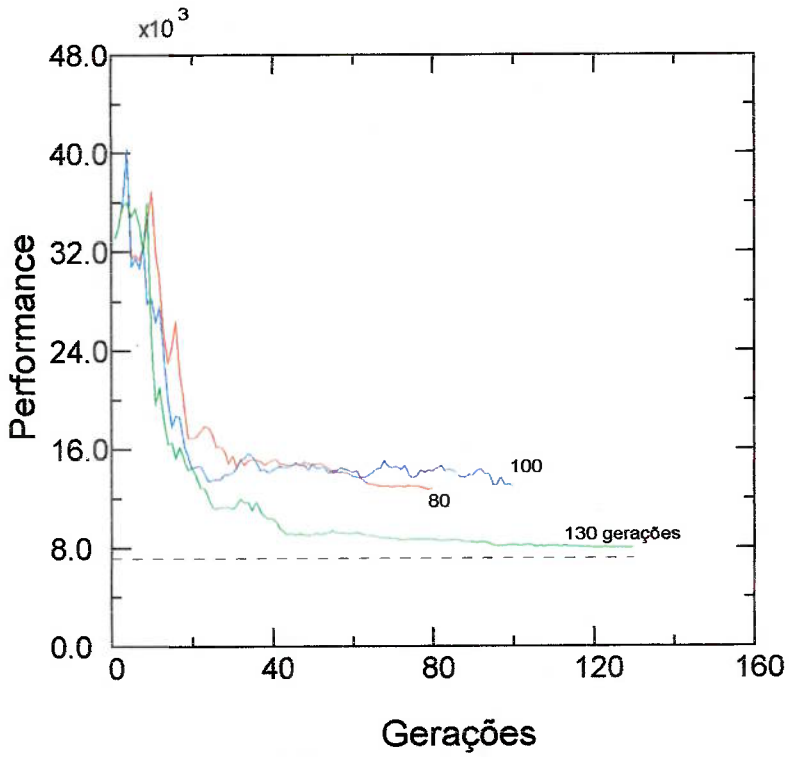


Figura 5.10 - Vaso de Pressão - Aplicação 1
População 80 indivíduos. Fator de penalização r_k=100000.
Probabilidades escaladas.
Algoritmo de Li, H. e Chou: performance = 7127.

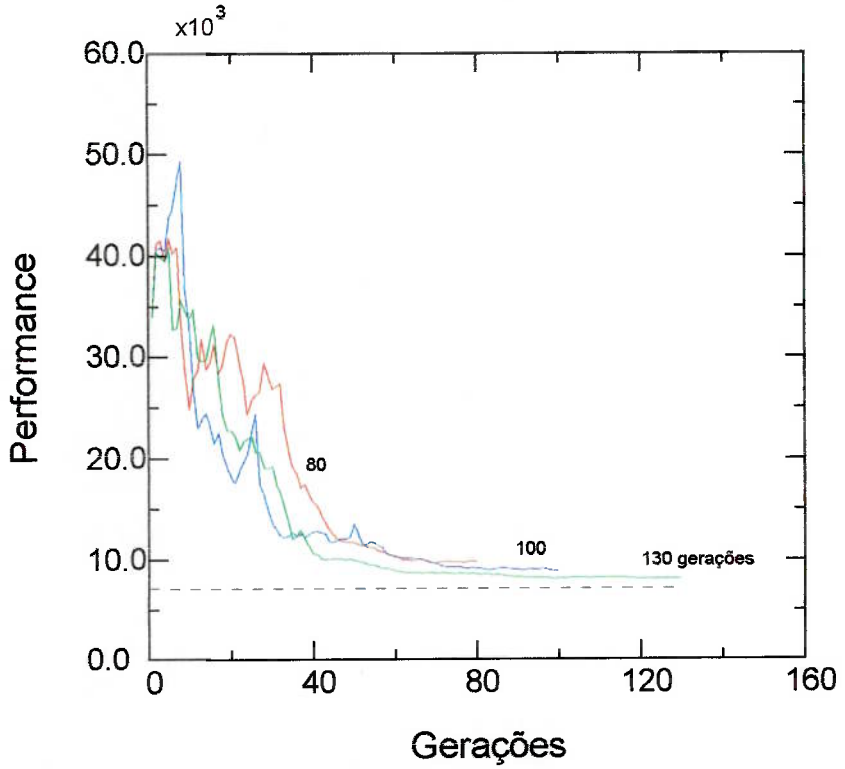


Figura 5.11 - Vaso de Pressão - Aplicação 2
População 100 indivíduos. Fator de penalização r_k=10000.
Probabilidades escaladas.
Algoritmo de Li, H. e Chou: performance = 7127.

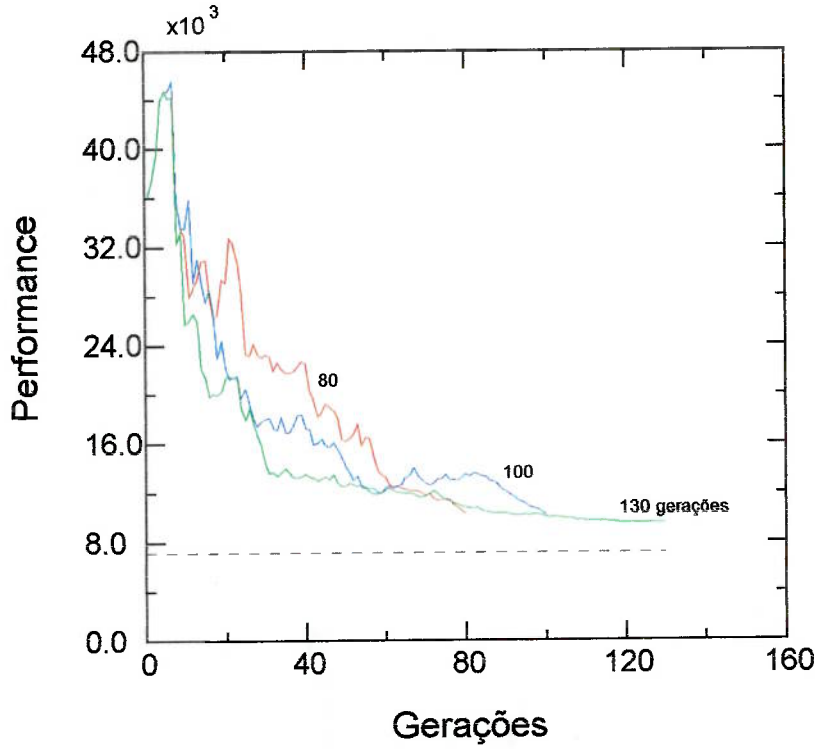


Figura 5.12 - Vaso de Pressão - Aplicação 3
População 120 indivíduos. Fator de penalização r_k=100000.
Probabilidades escaladas.
Algoritmo de Li, H. e Chou: performance = 7127.

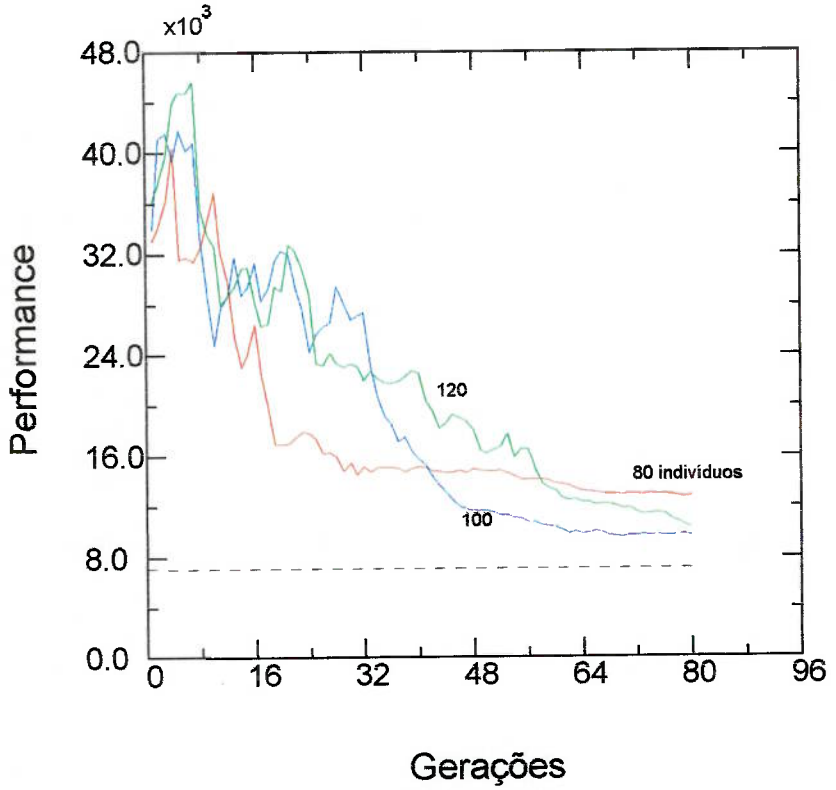


Figura 5.13 - Vaso de Pressão - Aplicação 4
80 gerações. Fator de penalização r_k=10000.
Probabilidades escaladas.
Algoritmo de Li, H. e Chou: performance = 7127.

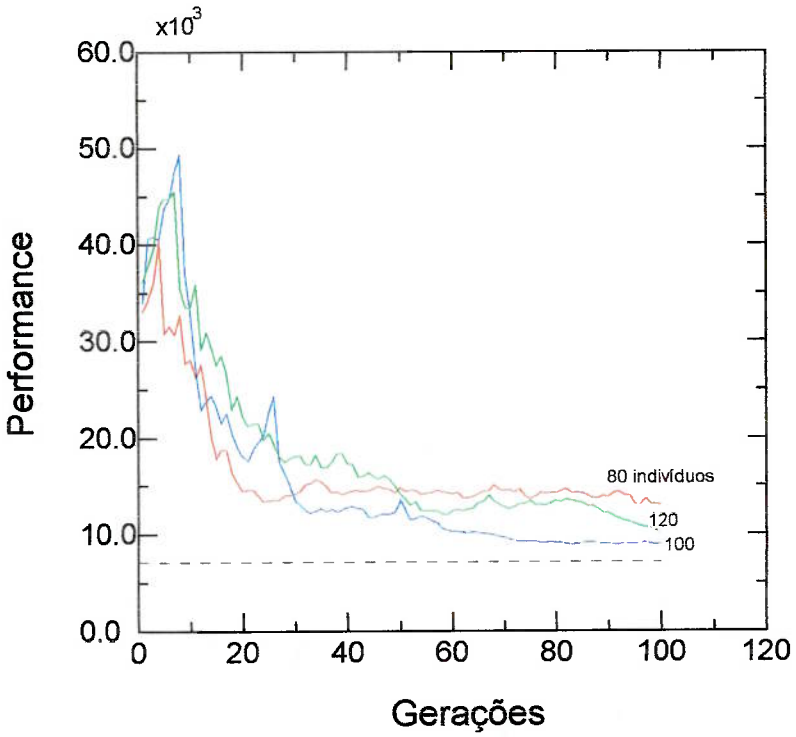


Figura 5.14 - Vaso de Pressão - Aplicação 5
100 gerações. Fator de penalização r_k=10000.
Probabilidades escaladas.
Algoritmo de Li, H. e Chou: performance = 7127.

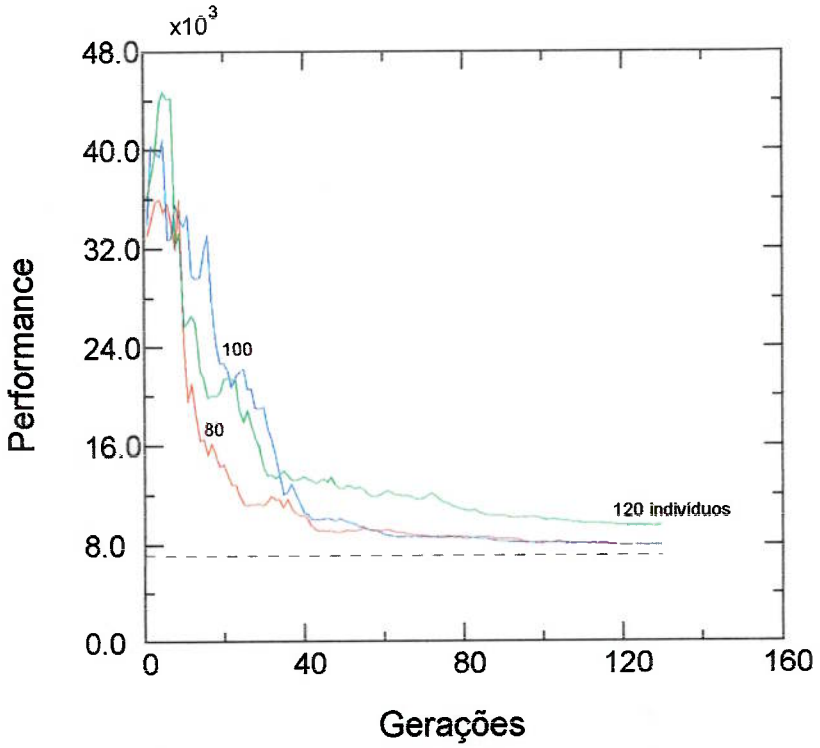


Figura 5.15 - Vaso de Pressão - Aplicação 6
130 gerações. Fator de penalização r_k=10000.
Probabilidades escaladas.
Algoritmo de Li, H. e Chou: performance = 7127.

Tabela 5.2 - Resultados para o projeto do Vaso de Pressão

	Fu,Fenton e Cleghorn	Li, H. e Chou	Thierauf e Cai	AG
f_{min}	8048.6	7127.3	7006.9	7833.6
x_1	1.125	1.000	1.000	1.000
x_2	0.625	0.625	0.625	0.750
x_3	48.380	51.250	51.812	50.044
x_4	111.745	90.991	84.591	102.028

O Algoritmo Genético foi rodado com uma população de 100 indivíduos e 130 gerações.

5.3 - Análise dos resultados

A partir das figuras 5.3, 5.4 e 5.5, obtidas através dos testes realizados com o problema da Tampa de Escotilha e das figuras 5.10, 5.11 e 5.12, obtidas através dos testes realizados com o problema do Vaso de Pressão, pode-se notar que os melhores resultados encontram-se nas figuras 5.4 (Tampa de Escotilha) e 5.11 (Vaso de Pressão). Em ambos os casos a população é moderada, fato que esta de acordo com as observações de De Jong (1975) que garante melhores resultados para um tamanho moderado de população. Se maiores populações permitem uma melhor convergência, uma vez que se pode encontrar um conjunto maior de diferentes *schemata* presentes nesta população, por outro lado populações menores têm a habilidade de mudar mais rapidamente e desta forma exibir, inicialmente, uma melhor performance.

Adicionalmente, nas figuras 5.6, 5.7 e 5.8, obtidas através das aplicações feitas com o problema da Tampa de Escotilha e as figuras 5.13, 5.14 e 5.15, obtidas através das aplicações feitas com o problema do Vaso de Pressão, mostra se que os melhores resultados foram encontrados nas figuras 5.8 e 5.15 , respectivamente. Em ambos os problemas utiliza-se a escala para probabilidades (equações (5.1) e (5.2)), garantindo-se que, com o progredir das gerações, a probabilidade de *crossover* tenda a 1 e a probabilidade de mutação tenda a 0, fato este também observado por De Jong em seus experimentos referentes à probabilidade de *crossover* (alta) e probabilidade de mutação

probabilidade de mutação tenda a 0, fato este também observado por De Jong em seus experimentos referentes à probabilidade de *crossover* (alta) e probabilidade de mutação (baixa). Deve-se lembrar que a probabilidade de mutação se apresenta interessante na prevenção da perda prematura de importantes alelos e a probabilidade de *crossover* tem compromisso entre uma melhora contínua da performance e convergência na busca do ótimo.

6 - Conclusões & Recomendações

Através dos resultados apresentados no capítulo anterior, pode-se verificar a característica fundamental do algoritmo genético, que é a de explorar regiões onde encontram-se os valores ótimos e, eventualmente, os atingir. O comportamento convergente, sem a garantia da optimalidade, é um problema que vem sendo encontrado pelos inúmeros usuários do Algoritmo. Consegue-se constatar que o AG isola, rapidamente, interessantes áreas do espaço de variáveis mas tem se mostrado um método fraco, quando comparado com os demais, na escalada para o ótimo. Isto não reduz sua utilidade. Muito pelo contrário, nos métodos onde se verifica uma melhor convergência não se garante a globalidade e, em muitas situações, não se consegue ter a flexibilidade do AG. Adicionalmente, muitos métodos são limitados a uma estreita classe de problemas, como o tratamento isolado de variáveis discretas e variáveis contínuas. Por outro lado, o Algoritmo Genético pode ser usado quando técnicas convencionais apresentam dificuldades naquele específico problema.

Neste contexto seria interessante utilizar esquemas híbridos, iniciando-se a busca com o algoritmo genético, para o isolamento de importantes regiões do problema e, uma vez conhecidas estas regiões, aplicar, localmente, os métodos convencionais de programação matemática para definir os ótimos locais. Desta forma, pode-se combinar a globalidade e o paralelismo do AG com o comportamento "mais" convergente das técnicas locais.

Alguns parâmetros são de incontestável importância para o desenrolar do algoritmo, como por exemplo, tamanho de população, número de gerações e

probabilidades de mutação e de *crossover*, todos relacionados com o número de variáveis do problema em questão. Neste trabalho estes parâmetros foram tratados qualitativamente, observando as sugestões presentes nas referências, como por exemplo, De Jong (1975), que obteve melhores resultados para populações moderadas, baixa probabilidade de mutação e alta probabilidade de crossover. Contudo, de modo a se obter melhor rendimento do Algoritmo Genético, seria interessante que os mesmos fossem quantificados em função das características do problema. Uma possibilidade seria o teste exaustivo de todas as combinações possíveis de valores para os parâmentros na busca de melhores resultados. A principal desvantagem desta abordagem é o tempo gasto na realização dos testes, além do fato de que a comparação direta dos resultados é pouco significativa quando não há uma combinação dominante dos parâmetros em todas as aplicações realizadas. Isto indica que um método que utilize testes estatísticos é conveniente a fim de se evitar a exploração de todas as combinações dos parâmetros.

Tal método, que fica aqui como sugestão para uma continuidade desta pesquisa em trabalhos futuros, é oportunamente proposto por Xu et al. (1996) e é chamado pelos autores de *The Growing and Pruning Method (GPM)*, o qual é apresentado na figura 6.1 e consiste da construção de uma árvore de decisão, onde cada nó i corresponde a um parâmetro P_i e cada nó k no nível i corresponde ao valor V_{ik} do parâmetro P_i. O nó raiz da árvore é uma versão *default* do algoritmo onde os parâmetros são estabelecidos para valores escolhidos arbitrariamente ou por uma fase preliminar de ajustes. No primeiro nível da árvore de decisão, desde o nó raiz, um número de nós é organizado variando somente um dos parâmetros em seu valor *default*. Em cada um destes nós o mesmo conjunto de problemas teste é resolvido pelo algoritmo e os resultados são comparados pelo teste de Friedman, que pode ser visto

em Xu et al. (1996). Este teste é capaz de determinar se as diferenças entre os resultados para cada valor do parâmetro no mesmo nível são estatisticamente significativas. Neste caso, o teste também pode estabelecer uma relação dominante entre os nós, possibilitando a eliminação dos valores dominantes dos parâmetros que não serão considerados nos próximos testes. Nos demais níveis da árvore o procedimento é quase o mesmo, exceto a comparação entre os nós com diferentes nós de origem, a qual é feita pelo teste de Wilcoxon, que também pode ser encontrado em Xu et al. (1996). Quando o último nível da árvore é atingido e todos os testes feitos, os nós remanescentes no nível correspondente são a melhor combinação dos valores para os parâmetros do algoritmo.

Apesar de o GPM ter sido proposto para um específico algoritmo de busca (*Tabu Search Algorithm*), o procedimento da figura 6.1 é bastante genérico e pode ser aplicado ao Algoritmo Genético, cujos parâmetros devem ser organizados em ordem de importância. Esta organização é possível caso haja um certo grau de independência entre os efeitos gerados pelos parâmetros na performance do algoritmo. Embora isto não seja totalmente verdade para os Algoritmos Genéticos, eles podem ser aproximadamente organizados em ordem decrescente de importância, caso estes efeitos sejam determinados separadamente para cada parâmetro e caso haja uma relação de dominância entre eles.

Passo 1: Supondo-se que k parâmetros sejam determinados e organizados em ordem decrescente de importância, $P = \{P_1, P_2, ..., P_k\}$. Um conjunto de valores, $V_i = \{V_{i1}, V_{i2}, ..., V_{iti}\}$, é escolhido para cada parâmetro P_i.

Passo 2: A árvore de decisão é iniciada considerando-se a partição do parâmetro P_1 do nó raiz o qual é também considerado a única folha inicial do nó. Define nível $i = 0$

Passo 3: Para cada folha existente no nó no nível i, cresce t_{i+1} partições onde cada partição representa um dos valores do parâmetro P_{i+1}. São feitas aplicações dos problemas teste para cada partição e os resultados são coletados, criando uma folha no nível $i + 1$. Os nós do nível i perdem o seu *status* como folhas.

Passo 4: Para cada conjunto de folhas (no nível $i + 1$) que dividem o mesmo nó de origem, é usado o teste de Friedman para a determinação dos melhores valores do parâmetro P_{i+1}. As folhas de valores inferiores ao melhores valores são eliminadas.

Passo 5: Para cada par de folhas remanescentes que não dividem um mesmo nó de origem, e as quais não foram examinadas neste passo, aplica-se o teste Wilcoxon. Caso os membros do par sob verificação são significantemente diferentes, elimina-se o de valor inferior.

Passo 6: $i = i + 1$. Se $i > k$, vá para o passo 7, senão vá para o passo 3.

Passo 7: Cada folha remanescente representa uma das melhores combinações dos valores dos parâmetros para o algoritmo.

Figura 6.1 - Método Estatístico (GPM)

Concluindo, o Algoritmo Genético se mostra como uma poderosa ferramenta para a síntese e otimização de estruturas, em geral, e particularmente da estrutura do navio, onde o processo de síntese ainda se baseia na difundida espiral de projeto.

7 - Referências Bibliográficas

Augusto, O . B., 1996 (1), "Processos de Busca não Linear para o projeto da Estrutura do Navio" BT/PNV/23, Departamento de Engenharia Naval e Oceânica - Escola Politécnica - USP.

Augusto, O . B., 1996 (2), "Projeto Estrutural de um Navio Patrulha Costeiro" BT/PNV/26, Departamento de Engenharia Naval e Oceânica - Escola Politécnica - USP.

Blanc, M., 1994, "Os Herdeiros de Darwin", Página Aberta.

Bouchard, E. E., 1988; Kidwell, G. H. ; Rogan, J. E. "The Application of Artificial Intelligence Technology to Aeronautical System Design" AIAA-88-4426, AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting, Atlanta, September.

Box, M. J., 1965, "A New Method of Constrained Optimization an Comparison with other Methods", Computer Journal, Vol. 8, No. 1, April, pp. 42-52.

Cella, A. ;Soosaar,K., 1973, "Discrete Variables in Structural Optimization" in Optimun Structural Design - Theory and Applications, John Willey & Sons.

Costa Neto, P.L.O, 1974., Cymbalista, M. "Probabilidades: resumos teóricos, exercícios resolvidos, exercícios propostos", Edgard Blücher.

Davidor, Y., 1989, "Genetic Algorithms for Order Dependent Process Applied to Robot Path Planning", Unpublished Ph.D. dissertation, Imperial College, University of London.

Davis, L., 1991, "Handbook of Genetic Algorithms", Van Nostrand Reinhold.

De Jong, K. A ., 1975, "An analysis of the Behavior of a Class of Genetic Adaptative Systems", (Doctoral Dissertation, University of Michigan). Dissertation Abstracts International 36(10), 5140B. (University Microfilms No. 76-9381).

Dias, C. A . N., Augusto, O . B., Rossi, R., 1998, "A Practical Method for Optimum Design of Mooring Systems", PRADS 98 - The 7th International Symposium on Practical Design of Ships and Mobile Units, September.

Fletcher, R., 1973, "Mathematical-Programming Methods" - A Critical Review, in Optimum Structural Design, Eds. Ghallagher, R. H., Zienkiewicz, O. C., John Wiley & Sons, Great Britain, 1973.

Fu, J., Fenton, R. G., Cleghorn, W. L., 1991, "A Mixed Integer-Discrete-Continuous Programming Method and its Application to Engineering Design Optimization", Engineering Optimization, N. 17, pp. 263-280.

Goldberg, D.E., Samtani, M.P., 1986, "Engineering Optimization via Genetic Algorithms" Proceedings of the Ninth Conference on Eletronic Computation, 471 - 482.

Goldberg, D. E., 1989, "Genetics Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley.

Grefenstette, J. J., 1986, "Optimization of Control Parameters for Genetics Algorithms", IEEE Transactions on Systems, Man, and Cybernetics, SMC-16(1), pp. 122-128.

Hillis, W. D., 1987, "The Connection Machine", Scientific American, 255(6).

Holland, J. H., 1975, "Adaptation in Natural and Artificial Systems" Ann Arbor: The University of Michigan Press,

Holland, J. H., 1992, "Genetic Algorithms", Scientific American, 267(1), 66-72.

Hooke, R., Jeeves, T. A., 1961, "Direct Search Solution of Numerical and Statistical Problems", Journal of the Association of Computing Machinery, Vol. 8, April, pp. 212-229.

Jenkins, W.M., 1997, "On the Application of Natural Algorithms to Structural Design Optimization" Engineering Structures, vol. 19, No. 4, 302 -308.

Kitamura, K., 1972, "Studies on Optimization of Ships Structures - Optimum Design of Longitudinal Members of Tanker" J.S.N.A., No. 132.

Li, H. L., Chou, C. T., 1994, "A Global Approach for Nonlinear Mixed Discrete Programming in Design Optimization", Engineering Optimization, N. 22, pp. 109-122.

Mayer-Kress, G., 1989, "A Nonlinear Dynamical System Approach to International Security", Technical Report LA-UR-89-1355, Los Alamos National Laboratory.

Meyer, P.L., 1982, "Probabilidade: aplicações à estatística", tradução do Prof. Ruy de C.B. Lourenço Filho, Livros Técnicos e Científicos.

Moe, J., 1970, "Design of Ship Structures by means of Non-linear Programming Techniques, International Shipbuilding Progress", Vol. 17, No. 187, March, pp. 69-89.

Nelder, J. A., Mead, R., 1965, "A Simplex Method for Function Minimization", Computer Journal, Vol. 7, No. 4, Jan., pp. 308-313.

Okada, T. ;Neki, I., 1992, "Utilization of Genetic Algorithm for Optimization the Design of Ship Hull Structures" J.S.N.A., vol. 171, June.

Thierauf, G., Jianbo, C., 1997, "Parallel Evolution Strategy for Solving Structural Optimization", Engineering Structures, Vol. 19, N.4, pp. 318-324.

Xu, J., Chiu, S. Y., Glover, F., 1996, "Fine- Tuning a Tabu Search Algorithm with Statistical Tests", Working Paper, University of Colorado.

Apêndice A - Revisão de Análise Combinatória e Probabilidade Elementar

O objetivo deste apêndice é fazer uma pequena revisão de Análise Combinatória e Probabilidade Elementar. Apesar de os conceitos aqui apresentados serem simples, eles são de extrema importância para a compreensão dos fundamentos matemáticos, utilizados como ferramenta na construção das bases do Algoritmo Genético. Deve-se ter em mente que estas simples páginas não têm a pretensão de serem uma referência para um curso de probabilidade, ao contrário, aqueles que têm a intenção de se aprofundar no assunto, podem encontrar nas referências desta dissertação um ponto de partida.

A.1 - Análise Combinatória

A.1.1 - Princípio da Indução Finita (PIF)

Seja $P(n)$ uma propriedade associada ao número natural n. Se $P(0)$ é verdadeira e se $P(k) \Rightarrow P(k+1)$, isto é, se do fato de se supor verdadeira $P(k)$ se puder concluir que $P(k+1)$ é verdadeira, então $P(n)$ é verdadeira para todo n natural.

Considere o seguinte problema:

Provar que para todo $n \in \mathrm{N}$, a seguinte afirmação é verdadeira:

$$P(n) : n^2 + n \text{ é divisível por } 2.$$

Solução:

- para $n = 0$, tem-se:

  $0^2 + 0 = 0$, que é divisível por 2, logo a propriedade é verdadeira para $n = 0$

- admitindo-se que:

  $k^2 + k$ é divisível por 2 ($n = k$), prova-se que:

  $(k + 1)^2 + k + 1$ é também divisível por 2 ($n = k + 1$)

  $(k + 1)^2 + k + 1 = k^2 + 2k + 1 + k + 1 = (k^2 + k) + 2(k + 1)$

Nota-se que:

$(k^2 + k)$ é divisível por 2 (Hipótese)

e

$\Rightarrow (k^2 + k) + 2(k + 1)$ é divisível por 2

$2(k + 1)$ é divisível por 2

Logo $(n^2 + n)$ é divisível por 2, qualquer que seja $n \in \mathrm{N}$.

A.1.2 - Binômio de Newton

Através do PIF, é possível mostrar que valem as seguintes relações:

$$(x+y)^n = \sum_{p=0}^{n} \binom{n}{p} x^{n-p} y^p \qquad \text{(A .1)}$$

$$(x-y)^n = \sum_{p=0}^{n} (-1)^p \binom{n}{p} x^{n-p} y^p \qquad \text{(A .2)}$$

Sendo $\binom{n}{p}$, definido como *número binomial de classe p do número n* ou *binomial de n sobre p,* como segue:

$$\binom{n}{p} = \frac{n!}{p!(n-p)!} \Leftrightarrow n \geq p; \quad \forall \mathrm{n,p} \in \mathrm{N} \qquad \text{(A .3)}$$

$$\binom{n}{p} = 0 \Leftrightarrow n < p; \quad \forall n, p \in N \qquad (A.4)$$

O Binômio de Newton é útil em um grande número de cálculos combinatoriais e probabilísticos.

Exemplo:

Mostrar que $\sum_{p=0}^{n} \binom{n}{p} = 2^n$.

Solução:

$$\sum_{p=0}^{n} \binom{n}{p} (1)^{n-p} (1)^p = (1+1)^n = 2^n$$

A.1.3 - Princípio Fundamental da Contagem

Se um acontecimento A pode ocorrer de *m* maneiras e se, para cada uma delas, um segundo acontecimento B pode ocorrer de *n* maneiras, então o número de maneiras de ocorrer A seguido de B é *m* . *n*.

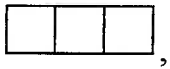
Exemplo:

Quantos números de três algarismos podem ser formados no sistema decimal:

a) podendo repetir os algarismos?

b) sem repetir os algarismos?

Solução:

a) Deve-se considerar o seguinte esquema  , onde o primeiro quadro é preenchido pelo algarismo das centenas, o segundo pelo algarismo das dezenas e o terceiro pelo das unidades. Considera-se também os seguintes acontecimentos e seus respectivos números de ocorrências:

Acontecimento	Número de ocorrência
A: escolha do algarismo para o primeiro quadro	9 (pois o zero não pode ocorrer neste quadro)
B: escolha do algarismo para o segundo quadro, após ter ocorrido A	10 (pode ocorrer qualquer algarismo do sistema decimal)
C: escolha do algarismo para o terceiro quadro, após ter ocorrido A e B	10 (notar que os algarismos estão sendo repetidos)

Então, pelo princípio fundamental da contagem tem-se:

$9 \cdot 10 \cdot 10 = 900$ números de três algarismos.

b) lembrando-se agora que o número deve ter os seus algarismos distintos, tem-se:

Acontecimento	Número de ocorrência
A: escolha do algarismo para o primeiro quadro	9 (pois o zero não pode ocorrer neste quadro)
B: escolha do algarismo para o segundo quadro, após ter ocorrido A	9
C: escolha do algarismo para o terceiro quadro, após ter ocorrido A e B	8

Logo, pelo princípio fundamental da contagem:

$9 \cdot 9 \cdot 8 = 648$ números.

A.1.4 - Arranjos Simples

Seja A um conjunto com n elementos. Um arranjo dos n elementos, tomados p a p, é qualquer subconjuto ordenado de A que tenha p elementos $(0 \leq p \leq n)$.

O número p chama-se *classe* ou *taxa do arranjo*, e o número total de arranjos simples de n elementos tomados p a p $\left(A_{n,p} \text{ ou } \mathrm{A}_{\mathrm{n}}^{\mathrm{p}}\right)$ é calculado através da relação:

$$A_{n,p} = n(n-1)(n-2) \ldots (n-p+1) \qquad \text{ou} \qquad \mathrm{A}_{\mathrm{n,p}} = \frac{n!}{(n-p)!} \qquad \text{(A .5)}$$

Exemplo:

Tendo-se 3 bolas diferentes e 5 gavetas, de quantos modos pode-se guardá-las, sendo uma em cada gaveta?

Solução:

Indique-se por G_1, G_2, G_3, G_4, G_5 essas 5 gavetas. Para guardar as 3 bolas, devem-se escolher 3 dentre essas 5 gavetas. Logo, cada escolha efetuada é um subconjunto que contém 3 dos 5 elementos acima. Como as bolas que devem ser guardadas são diferentes entre si, para um mesmo subconjunto de 3 gavetas pode-se obter maneiras diferentes de guardar as bolas, mudando a ordem das gavetas. Isto significa que os subconjuntos são ordenados, isto é, são arranjos de classe 3. Logo, $A_{5,3} = 5 \cdot 4 \cdot 3 = 60$ maneiras.

A.1.5 - Permutação

Seja A um conjunto com n elementos distintos. Uma permutação dos n elementos é qualquer dos arranjos de classe n, formados com esses n elementos.

O número total das permutações de n elementos distintos, P_n, é dado por:

$$P_n = A_{n,n} = n(n-1)(n-2) \ldots 3 \cdot 2 \cdot 1 \qquad \text{ou} \qquad P_n = n! \qquad \text{(A .6)}$$

Exemplos:

Considere os anagramas da palavra REPÚBLICA; pergunta-se:

a) qual é o total deles?

b) quantos começam por consoante?

c) quantos terminam por vogal?

d) quantos começam por vogal e terminam por consoante?

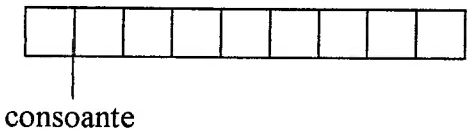
Solução

a) O total de anagramas é igual ao número de permutações simples das letras R, E, P, U, B, L, I, C, A, portanto 9!

b) Considere o esquema abaixo: os 9 quadros representam uma permutação ou anagrama da palavra REPÚBLICA, onde no primeiro quadro tem-se sempre uma consoante.

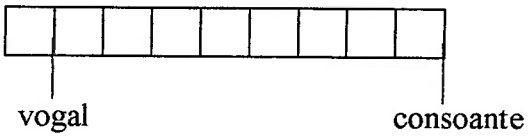
   Para cada consoante do conjunto {R, P, B, L, C} colocada no primeiro quadro, pode-se formar tantos anagramas quantas são as permutações das 8 letras restantes, isto é, 8! .

Como pode-se escolher uma consoante de 5 maneiras diferentes, tem-se 5 . 8! anagramas.



c) Fazendo-se um raciocínio análogo ao item anterior, obtém-se 4 . 8! anagramas.

d) Os anagramas que começam por uma vogal e terminam por uma consoante assumem a forma do esquema abaixo.



Para cada vogal do conjunto {E, U, I, A} colocada no primeiro quadro, e uma consoante do conjunto {R, P, B, L, C} colocada no último quadro, pode-se fazer tantos anagramas quantas são as permutações das 7 letras restantes, isto é, 7! .

Mas, pelo princípio fundamental, tem-se:

escolha de uma vogal para o primeiro quadro: 4 maneiras

escolha de uma consoante para o último quadro: 5 maneiras

Tem-se então, 20 escolhas possíveis e, como para cada escolha pode-se fazer 7! anagramas, resulta um total de 20 . 7! anagramas.

A.1.6 - Permutação com repetição

Considera-se um conjunto com n elementos, entre os quais:

α_1 são iguais a a_1

α_2 são iguais a a_2

.

.

.

α_k são iguais a a_k

Tem-se: $\alpha_1 + \alpha_2 + \ldots + \alpha_k = n$. O número de permutações desse grupo representa-se por:

$$P_n^{(\alpha_1, \alpha_2, \ldots, \alpha_k)} = \frac{n!}{\alpha_1! \cdot \alpha_2! \cdot \ldots \cdot \alpha_k!} \qquad \text{(A .7)}$$

Exemplo:

Quantos são os anagramas da palavra PATRIARCA.?

Solução:

Tem-se:

1 elemento igual a T	2 elementos iguais a R
1 elemento igual a I	1 elemento igual a P
1 elemento igual a C	3 elementos iguais a A

Logo: $P_n^{(3, 2, 1, 1, 1, 1)} = \frac{9!}{3! \cdot 2! \cdot 1! \cdot 1! \cdot 1! \cdot 1!} = 30240$

A.1.7 - Combinações

Uma combinação dos elementos distintos de um conjunto A, tomados p a p, é qualquer subconjunto não-ordenado de A que tenha p elementos ($0 \leq p \leq n$).

O número de combinações dos n elementos tomados p a p

$\left(C_{n,p} \text{ ou } C_n^p \text{ ou } \binom{n}{p} \right)$ é:

$$C_{n,p} = \frac{A_{n,p}}{P_p} \qquad \text{ou} \qquad C_{n,p} = \frac{n!}{p!(n-p)!} \tag{A.8}$$

Exemplo:

Tendo-se 3 bolas iguais e 8 gavetas, de quantos modos pode-se guardar as bolas, sendo uma em cada gaveta?

Solução:

Indica-se por G_1, G_2, G_3, ... , G_8 as 8 gavetas, então para se guardar 3 bolas iguais, sendo uma em cada gaveta, deve-se escolher 3 entre essas 8 gavetas. Portanto, cada escolha é um subconjunto que contém 3 dos 8 elementos acima, isto é, uma combinação das 8 gavetas tomadas 3 a 3 (notar que o que torna o subconjunto não-ordenado é o fato de as bolas serem iguais).

Logo: $C_{8,3} = \frac{8!}{3!(8-3)!} = 56$ modos diferentes

A.2 - Probabilidade

A.2.1 - Espaço Amostral. Eventos

Em um fenômeno aleatório ou probabilístico, isto é, sujeito às leis do acaso, chama-se *espaço amostral* ou *espaço das probabilidades* ao conjunto (em geral o mais detalhado possível) de todos os resultados possíveis de acontecer. Este conjunto denotar-se-á *S*.

Por exemplo, se dois dados forem jogados, o espaço amostral correspondente poderá ser descrito da seguinte forma:

$$S_1 = \begin{Bmatrix} (1,1) & (2,1) & (3,1) & (4,1) & (5,1) & (6,1) \\ (1,2) & (2,2) & (3,2) & (4,2) & (5,2) & (6,2) \\ (1,3) & (2,3) & (3,3) & (4,3) & (5,3) & (6,3) \\ (1,4) & (2,4) & (3,4) & (4,4) & (5,4) & (6,4) \\ (1,5) & (2,5) & (3,5) & (4,5) & (5,5) & (6,5) \\ (1,6) & (2,6) & (3,6) & (4,6) & (5,6) & (6,6) \end{Bmatrix}$$

onde o primeiro número de cada par indica o ponto de um dos dados e o segundo indica o ponto do outro dado.

Da mesma forma, se quatro moedas forem jogadas, o espaço amostral poderá ser descrito por:

$$S_2 = \begin{Bmatrix} CCCC & CCCK & CCKK & CKKK & KKKK \\ & CCKC & CKCK & KCKK & \\ & CKCC & KCCK & KKCK & \\ & KCCC & CKKC & KKKC & \\ & & KCKC & & \\ & & KKCC & & \end{Bmatrix}$$

onde C pode representar coroa e K cara, em cada moeda lançada.

Qualquer subconjunto de um espaço amostral será um *evento*, definindo um resultado bem determinado. Os eventos podem ser simples ou compostos, conforme se constituam de um ou mais resultados de S. Os eventos serão designados por letras maiúsculas.

Dentre os eventos a serem considerados deve-se incluir o próprio S (evento certo) e o conjunto vazio, ϕ (evento impossível).

Por exemplo, sejam em S_1 os eventos:

E = dar 1 nos dois dados;

F = soma dos pontos igual a quatro;

G = soma dos pontos menor ou igual a cinco;

H = dar 2 no primeiro dado.

Tem-se

$E = \{(1,1)\}$; $F = \{(1,3), (2,2), (3,1)\}$; etc.

O evento E é simples e F é composto.

As operações entre os conjuntos podem ser aplicadas aos eventos. Define-se então:

a) Evento intersecção. $E \cap F$, $E \cdot F$

É o evento formado pelos resultados que pertencem a ambos os eventos considerados. Usando um diagrama de Euler-Venn, tem-se simbolicamente:

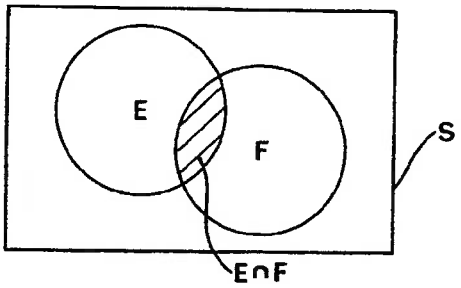


Figura A.1 - Evento intersecção

Exemplo: em S_1

$G \cap H = \{(2,1), (2,2), (2,3)\}$

O evento intersecção significa a ocorrência de ambos os eventos considerados.

b) Evento reunião ou união. $E \cup F$, $E + F$

É o evento formado pelos resultados que pertencem a pelo menos um dos eventos considerados. Simbolicamente:

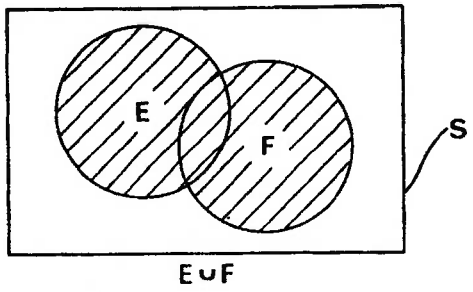


Figura A.2 - Evento união

Exemplo: em S_1

$F \cup H = \{(1,3), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6), (3,1)\}$.

O evento reunião ou união significa a ocorrência de pelo menos um dos eventos considerados.

c) Evento complementar. $\overline{E}$, E^c

É o evento formado pelos resultados que não pertencem ao evento considerado. Simbolicamente:

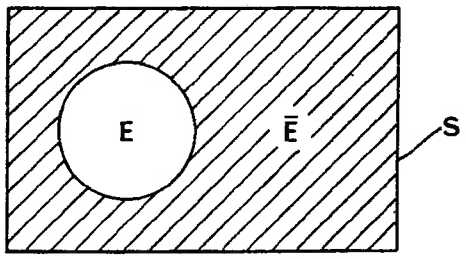


Figura A.3 - Evento complementar

O evento complementar significa a não ocorrência do evento considerado.

As operações acima podem facilmente ser estendidas a mais de dois eventos.

Por outro lado, dizemos que dois eventos E e F são mutuamente exclusivos se $E \cap F = \phi$. Simbolicamente:

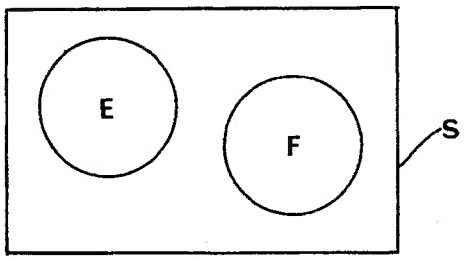


Figura A.4 - Eventos mutuamente exclusivos

Logo, tais eventos não podem ocorrer simultaneamente em uma mesma realização de um experimento aleatório.

Essa definição também pode ser imediatamente generalizada a mais de dois eventos.e a reunião de n eventos mutuamente exclusivos é o próprio S, diz-se que tais eventos são *mutuamente exclusivos e exaustivos*, ou que formam uma *partição* de S.

A.2.2 - Probabilidade e suas propriedades

A maior ou menor possibilidade de ocorrência dos diversos eventos é medida por um número chamado *probabilidade*.

Pode-se de forma simples entender a probabilidade como sendo um número real associado a um evento, gozando das seguintes propriedades:

a) $0 \leq P(E) \leq 1$; (A.9)

b) $P(S) = 1$; (A.10)

c) $P(\phi) = 0$; (A.11)

d) Se $E, F, ..., K$ são eventos mutuamente exclusivos

$$P(E \cup F \cup ... \cup K) = P(E) + P(F) + ... + P(K); \quad (A.12)$$

e) $P(\bar{E}) = 1 - P(E)$; (A.13)

f) $P(E \cup F) = P(E) + P(F) - P(E \cap F)$; (A.14)

g) $P(E \cup F) = P(E) + P(\bar{E} \cap F)$. (A.15)

Deve-se notar que as propriedades acima não fornecem uma maneira objetiva de se atribuir valores à probabilidade. Em verdade, a probabilidade, considerada à luz das propriedades acima, pode ter seus valores atribuídos de forma subjetiva por diversos indivíduos o que, em diversos casos, tem suas vantagens.

Uma regra prática que fornece uma maneira mais objetiva para a atribuição numérica da probabilidade é:

$$P(E) = \frac{m}{n}, \tag{A.16}$$

onde

m = número de resultados favoráveis ao evento E;

n = número de resultados possíveis, *desde que igualmente prováveis*.

A maneira teoricamente mais objetiva de se atribuírem probabilidades seria, em casos onde o experimento pode ser repetido indefinidas vezes, através do valor limite para o qual tende a proporção de vezes em que o evento ocorre, à medida que aumenta o número de repetições do experimento, sempre sob as mesmas condições.

A.2.3 - Probabilidade condicionada e correspondentes propriedades

Muitas vezes, o fato de saber-se que certo evento ocorreu faz com que se modifique a probabilidade que se atribui a outro evento. Denota-se por $P(E|F)$ à probabilidade do evento E sabendo-se que F ocorreu ou, simplesmente, probabilidade de E condicionada a F

Pode-se mostrar a coerência da relação segundo a qual

$$P(E|F) = \frac{P(E \cap F)}{P(F)}, \quad \mathrm{P(F)} \neq 0. \tag{A.17}$$

Analogamente

$$P(F|E) = \frac{P(E \cap F)}{P(E)}, \quad \mathrm{P(E)} \neq 0. \tag{A.18}$$

Das expressões acima resulta a regra do produto, que refere-se ao cálculo da probabilidade do evento intersecção,

$$P(E \cap F) = P(E) \cdot P(F|E) = P(F) \cdot P(E|F). \tag{A.19}$$

Nota-se que a ordem de condicionamento pode ser invertida. Para três eventos pode-se, por exemplo, escrever:

$$P(E \cap F \cap G) = P(E) \cdot P(F|E) \cdot P(G|E \cap F). \tag{A.20}$$

De forma semelhante pode ser generalizada a expressão para diversos eventos.

São importantes os dois seguintes teoremas:

a)*Teorema da probabilidade total.* Seja E_1, E_2, ..., E_n uma partição e F um evento qualquer de S. Então:

$$P(F) = \sum_{i=1}^{n} P(E_i) \cdot P(F|E_i). \tag{A.21}$$

Este resultado pode ser demonstrado considerando-se o evento F subdividido em suas intersecções com os eventos E_i e aplicando-se as probabilidades anteriores.

b) *Teorema de Bayes.* Nas condições do teorema anterior:

$$P(E_j|F) = \frac{P(E_j) \cdot P(F|E_j)}{\sum_{i=1}^{n} P(E_i) \cdot P(F|E_i)} \qquad j = 1, 2, ..., n. \tag{A.22}$$

Consegue-se facilmente este resultado, do teorema anterior e demais propriedades.

Os dois teoremas acima são particularmente úteis no estudo de situações que se processam em duas etapas, a primeira das quais corresponde à ocorrência de um (e somente um, devido à sua natureza) dos eventos da família E_i, dizendo o efeito F respeito à segunda etapa. O primeiro teorema ensina como calcular a propriedade incondicional do evento F, isto é, não importando qual dos eventos da família E_i possa ter ocorrido. O teorema de Bayes por sua vez, mostra como calcular a probabilidade

de que tenha sido o particular evento E_j da família E_i aquele que ocorreu, isto face à informação de que o evento F ocorreu.

A.2.4 - Eventos independentes

Se $P(E|F) = P(E|\ \bar{F}) = P(E)$, o evento E é dito *estatisticamente independente* do evento F. Isto implica em que o evento F também será estatisticamente independente do evento E, o que é fácil provar.

Nas condições de independência os cálculos se simplificam, pois não é mais necessário a preocupação com probabilidades condicionadas.

Sendo independentes os eventos, a regra do produto fica

$$P(E \cap F) = P(E) \cdot P(F), \tag{A.23}$$

sendo de imediata generalização a vários eventos, ou seja:

$$P(E \cap F \cap \ \ldots \ \cap K) = P(E) \cdot P(F) \ \ldots \ \mathrm{P(K)}. \tag{A.24}$$

Exemplo:

Um baralho de 52 cartas é subdividido em 4 naipes: copas, espadas, ouros e paus.

a) Retirando-se uma carta ao acaso, qual a probabilidade de que ela seja de ouros ou de copas?

b) Retirando-se duas cartas ao acaso com reposição da primeira carta, qual a probabilidade de ser a primeira de ouros e a segunda de copas?

c) Recalcular a probabilidade anterior se não houver reposição da primeira carta.

d) Havendo reposição, qual a probabilidade de sair a primeira carta de ouros ou então a segunda de copas?

Solução:

a) Sejam os eventos

E = sair carta de ouros;

F = sair carta de copas.

Os eventos E e F são mutuamente exclusivos e existe interesse na sua reunião, isto é, na ocorrência de *um* deles. Logo, pela propriedade (A.12), tem-se:

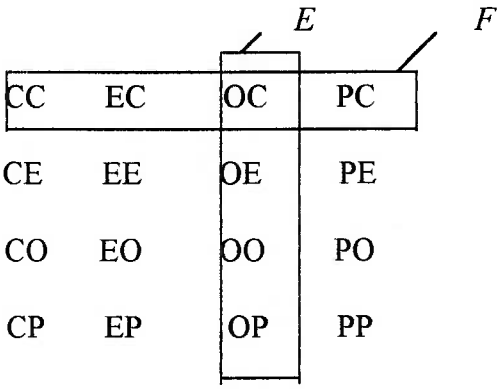
$$P(E \cup F) = P(E) + P(F) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}.$$

b)Sejam redefinidos os eventos E e F:

E = sair a primeira carta de ouros;

F = sair a segunda carta de copas.

O evento desejado é a intersecção dos eventos E e F, conforme pode-se ver no espaço amostral da retirada de duas cartas do baralho, dado abaixo:



Como as retiradas são feitas com reposição, os eventos E e F são independentes, logo, pela regra do produto na sua forma simplificada, dada por (A.23), tem-se:

$$P(E \cap F) = P(E) \cdot P(F) = \frac{1}{4} \cdot \frac{1}{4} = \frac{1}{16}.$$

c) Neste caso, os eventos E e F não são mais independentes, logo deve-se aplicar a regra do produto, conforme é dada em (A.19),

$$P(E \cap F) = P(E) \cdot P(F|E) = \frac{1}{4} \cdot \frac{13}{51} = \frac{13}{204}.$$

Note-se que, se a primeira carta foi de ouros, haverá 13 cartas de copas num total de 51, ao se fazer a segunda retirada.

d) Deseja-se neste momento a reunião dos eventos E e F, que não são mutuamente exclusivos, pois trata-se de cartas diferentes. Logo deve-se usar a propriedade (A .14), obtendo-se:

$$P(E \cup F) = P(E) + P(F) - P(E \cap F) = \frac{1}{4} + \frac{1}{4} - \frac{1}{4} \cdot \frac{1}{4} = \frac{7}{16}$$

Uma solução alternativa seria através do evento complementar, ou seja, a primeira carta não ser de ouros e a segunda não ser de copas. Sendo $G = E \cup F$ o evento cuja probabilidade queremos calcular, tem-se:

$$P(\overline{G}) = P(\overline{E} \cap \overline{F}) = \frac{3}{4} \cdot \frac{3}{4} = \frac{9}{16}.$$

Logo, pela propriedade (A.13), tem-se:

$$P(G) = P(E \cup F) = 1 - P(\overline{G}) = 1 - \frac{9}{16} = \frac{7}{16}.$$