RELATÓRIO PARCIAL DE INICIAÇÃO CIENTÍFICA

INSTITUTO NACIONAL
DE
PESQUISAS ESPACIAIS

ALGORITMOS GENÉTICOS APLICADOS À OTIMIZAÇÃO DE ESTRUTURAS "ISOGRID"

Leonardo de Azevedo Sanchez

ORIENTADOR : Mário Kataoka Filho

Índice

Capítulo 1

1 . Estação espacial internacional (ISS) :

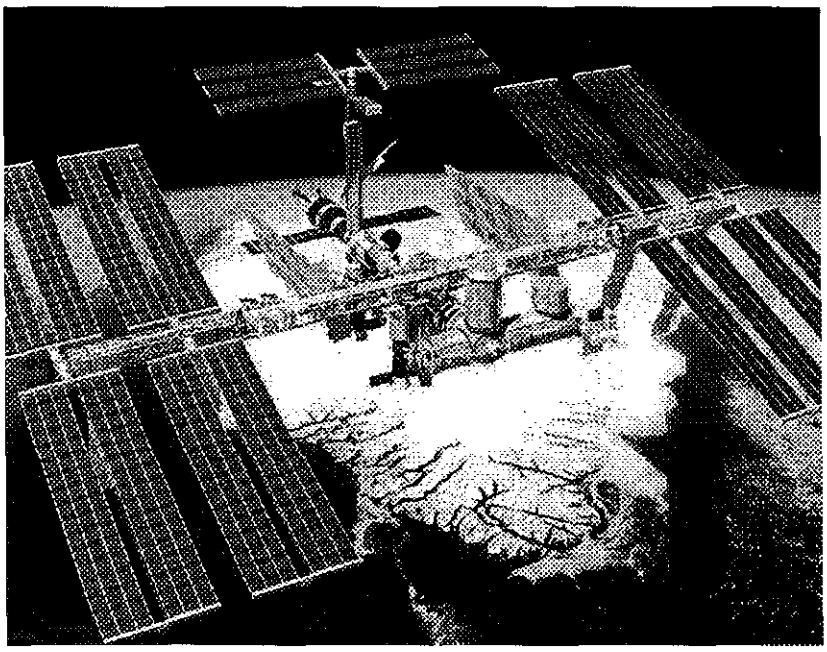

Figura 1.1 – Estação espacial internacional

A estação espacial internacional representa o maior projeto em engenharia da história da humanidade, não somente em tempo gasto com um projeto mas também em tecnologia utilizada. A ISS consistirá de centenas de elementos individuais vindos de várias partes do globo, já que o projeto está dividido entre 16 países, dentre eles o Brasil.

Quando a estação estiver completa, por volta do final de 2003 (se a NASA cumprir seu calendário), ela funcionará como um hotel com muitos quartos e como um centro de pesquisas orbitando a Terra a cada 90 minutos. Sua população permanente será de seis ou sete astronautas alternando e misturando americanos, russos, europeus e japoneses.

A ISS terá cinco vezes o tamanho da estação espacial russa Mir e terá painéis solares e radiadores "brotando" da treliça principal abrangendo aproximadamente 107 metros de comprimento.

Para esse projeto, a NASA está confiando à Boeing, sua principal empreiteira, a tarefa de assegurar de que todas as partes vão se encaixar e funcionar perfeitamente como planejado. Além disso, as outras nações participantes do projeto estão contribuindo com as "ferragens": Os canadenses estão construindo um braço e mão robóticos que serão de valor inestimável para trabalhos de montagem e transporte de cargas pesadas; a agência espacial do Japão, NASDA, está fornecendo um módulo-laboratório que inclui uma "varanda de fundos" onde os experimentos podem ser expostos ao espaço; astronautas usarão um outro braço robótico desenvolvido pelos próprios japoneses para levar os experimentos do compartimento de descompressurização à "varanda". A agência espacial européia também está fornecendo um módulo – laboratório.

Nesses módulos – laboratório serão desenvolvidas pesquisas em que a ausência de gravidade, ou melhor, a presença de microgravidade (um milionésimo da força gravitacional sentida na Terra) é essencial. Os pesquisadores da NASA acreditam que a pesquisa desenvolvida na estação pode levar a novos tratamentos para doenças, melhores materiais e ligas e até mesmo a um uso mais eficiente de combustíveis.

2. ULC e estruturas Isogrid

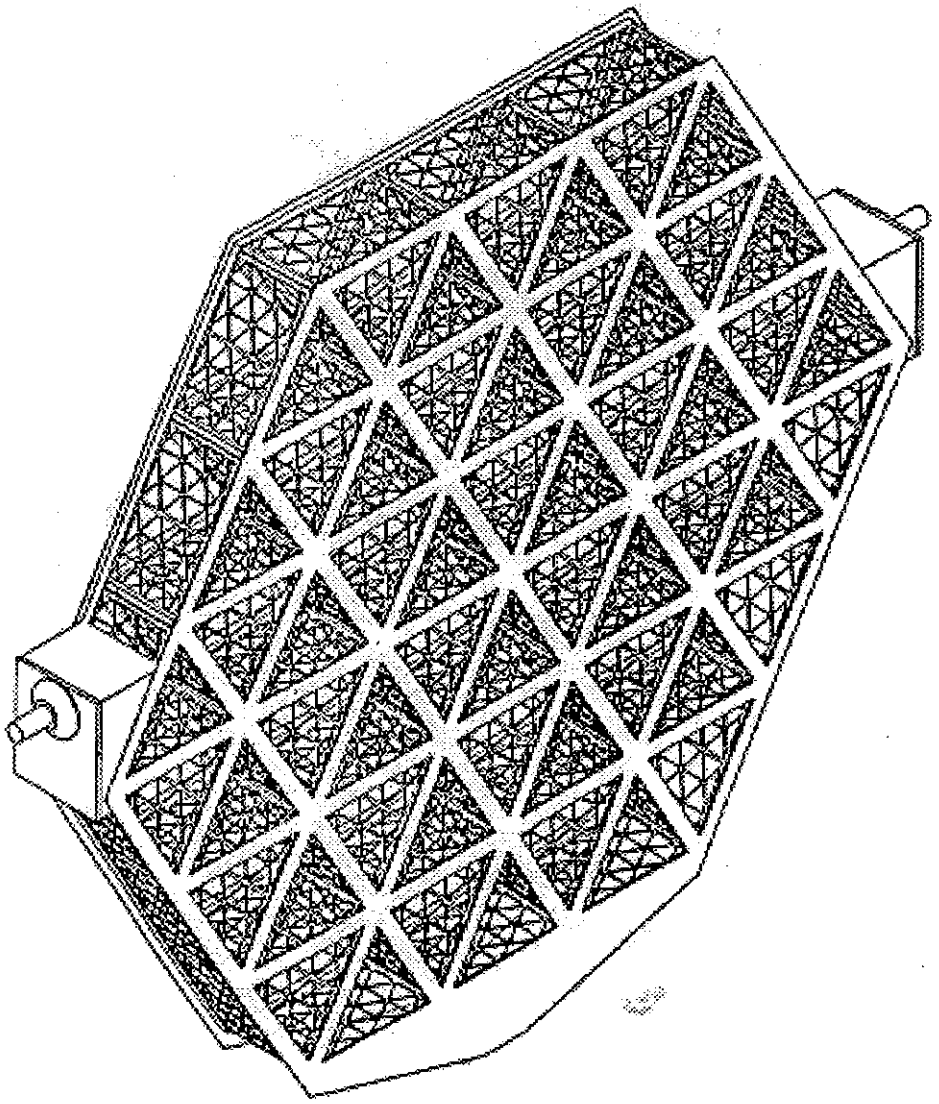

Figura 1.2 – Vista tridimensional do ULC

Este trabalho se propõe a estudar a otimização de uma estrutura da estação espacial internacional mais conhecida como ULC (Unpressurized Logistics Carrier).

Por sugestão da Boeing North American (BNA), decidiu-se utilizar estruturas isogrid para a composição do ULC, tendo em vista o seu sucesso em projetos anteriores. Isogrids são treliças espaciais formadas por triângulos isósceles que lidam com as condições de carregamento de forma mais eficiente, permitindo a minimização do peso total da estrutura.

O ULC tem a função de carregar os experimentos (cargas úteis) para a estação, como uma pallet.

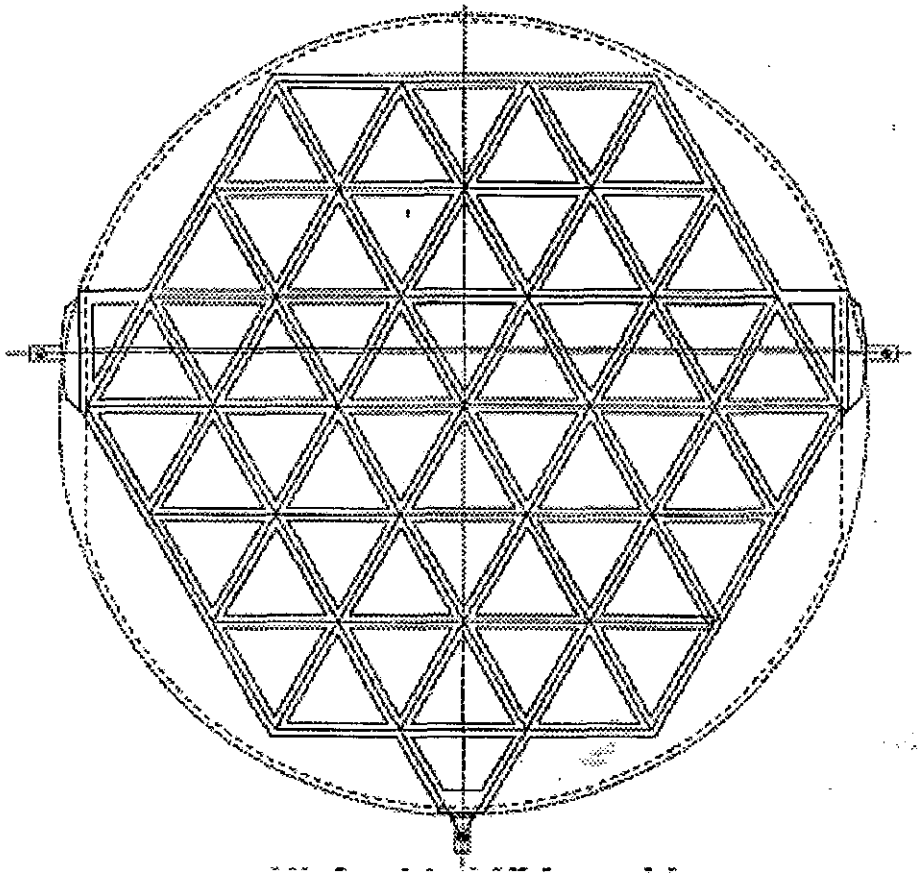

Figura 1.3 – ULC com isogrids de 28 polegadas

Primeiramente, a estrutura, que será transportada dentro do "cargo bay" do ônibus espacial americano (responsável pelo transporte de boa parte dos componentes da estação), deve suportar as condições de

lançamento do mesmo já que as "caixas" contendo os experimentos exercerão um carregamento sobre a estrutura.

Em segundo lugar, o braço mecânico do ônibus espacial pegará o ULC e entregará para o braço mecânico da estação. Este último fixará o ULC na estação para que os experimentos possam ficar em um ambiente de microgravidade durante um tempo pré-determinado, ao final do qual serão recolhidos pelo ônibus espacial e trazidos de volta à Terra para análise dos resultados.

3.História da Otimização Estrutural

Pesquisas na área de otimização estrutural vêm desde Galileo Galilei, 1638. Sua "mais resistente barra em engaste" e a formulação de cisalhamento constante podem ser considerados uma configuração ótima para peso mínimo sob restrição quanto a um valor constante de tensão. Porém, somente após a introdução do cálculo (Newton e/ou Leibnitz) no final do século XVII é que foi possível apresentar resultados razoáveis com a matemática da otimização.

O desenvolvimento de computadores digitais de alta velocidade teve estrondoso impacto no campo das engenharias. Capacidades de cálculo cada vez maiores proporcionaram soluções nunca antes atingidas na resolução de problemas complexos de engenharia e matemática. O computador tem sido responsável pelo desenvolvimento de poderosos métodos numéricos tais como o método dos elementos finitos e o método das diferenças finitas. Entre os métodos numéricos, alguns tópicos têm merecido a atenção dos pesquisadores, dentre eles: Análise de sensibilidade, design multidisciplinar, métodos heurísticos, otimização por decomposição e otimização geométrica. Neste trabalho serão tratados somente os métodos heurísticos de busca.

Os métodos heurísticos foram desenvolvidos para resolver problemas complexos de combinatória, sendo ferramentas poderosas na localização de ótimos globais em problemas razoavelmente difíceis, uma vez que não se faz necessário o cálculo do gradiente da função objetivo e restrições. Entre os métodos heurísticos, quatro são mais comumente usados: algoritmos genéticos, redes neurais, que tem seus fundamentos nas ciências biológicas; simulated annealing, que é inspirado na segunda lei da termodinâmica e por fim o método tabu que procura resultados através de procedimentos atrativos.

4. Algoritmos Genéticos

A idéia central por trás dos algoritmos genéticos foi primeiramente proposta por John Holland. Esses algoritmos são computacionalmente simples de serem implementados e são ao mesmo tempo poderosos na sua busca pela solução ótima. Além do mais, eles não são limitados por restrições a respeito do espaço de busca (restrições com relação à continuidade, existência de derivadas, unimodalidade e outras). Sendo assim, a busca por soluções utilizando – se os algoritmos genéticos pode se dar em conjuntos não – convexos e mesmo disjuntos, com funções objetivo também não – convexas e não-diferenciáveis podendo-se trabalhar simultaneamente com variáveis reais, lógicas e inteiras. É de se ressaltar também que os algoritmos genéticos não são facilmente presos a mínimos locais como os algoritmos usuais de programação matemática. Quando utilizados na busca de soluções, tais características dos AGs podem levar à descoberta de soluções não convencionais que não seriam vislumbradas pelo projetista por serem contra-intuitivas.

Os interesses de Holland, entretanto, não se limitavam somente a problemas de otimização. Holland estava mais interessado no estudo do

comportamento de sistemas adaptativos complexos, sejam eles biológicos (como o nosso sistema imunológico) ou não (como a economia global ou setorial, por exemplo).

Em seu "Adaptation in Natural and Artificial Systems" de 1975, possíveis soluções para um dado problema foram codificadas por uma representação de uma cadeia de "bits" e transformações análogas às reproduções e evoluções biológicas foram usadas para variar e melhorar essas soluções codificadas. Foi estabelecida uma analogia com o processo natural de reprodução em populações biológicas, onde informação genética armazenada em cadeias cromossomiais evoluem através das gerações para adaptar favoravelmente a um ambiente em mudança. Essa estrutura cromossomial representa a memória das gerações e é alterada quando membros da população reproduzem. Os três processos básicos que afetam a caracterização do cromossomo na evolução natural são as inversões das cadeias cromossomiais, uma mutação ocasional da informação genética e um cruzamento de informações genéticas entre os reprodutores. O último processo é uma troca de material genético entre os pais e permite que genes melhores sejam apresentados pelos seus "filhos". Uma simulação matemática desses processos caracterizam os componentes principais dos algoritmos genéticos.

5. Programa de Elementos Finitos

O programa de elementos finitos utilizado neste trabalho foi desenvolvido pelo professor Edgar Mamiya e sua equipe da UnB (Universidade de Brasília).

Para utilizar o programa é necessário escrever o arquivo fonte em uma pseudo linguagem própria do programa e dar um nome ao arquivo

que comece com a letra "i" (input). Após carregar o arquivo no ambiente do programa e realizar a simulação, o programa fornece um arquivo de saída com os resultados da simulação e este arquivo possui a inicial "o" (output) no seu nome.

Exemplo :

Arquivo de entrada ⇒ iviga
Arquivo de saída ⇒ oviga

Capítulo 2

1.Introdução :

A literatura atual identifica três tipos principais de métodos de busca: os baseados em cálculos, os enumerativos e os aleatórios. Examinaremos cada um para ver que conclusões podem ser tiradas sem que se faça um teste formal.

Dentre as estratégias mais simples para se otimizar uma função estão os métodos de busca enumerativos, onde a função é determinada em um número muito grande de pontos pré–determinados. Refinamentos sucessivos e avaliações revelam a solução ótima. Uma melhora na enumeração simples são métodos tais como "caminhada aleatória" e "caminhada aleatória com exploração da direção". O inconveniente de tais métodos é que para problemas maiores, tais não serão melhores no desenvolvimento da solução do que uma enumeração exaustiva. Tais estratégias são computacionalmente pesadas, ainda mais em projetos estruturais onde o comportamento das respostas a restrições do sistema requerem frequentemente soluções de sistemas de equações muito grandes.

Os métodos de busca baseados em cálculo foram estudados exaustivamente. Estes são subdivididos em duas classes principais: indiretos e diretos. Métodos indiretos procuram o extremo local através da solução do sistema de equações não-lineares resultantes quando se iguala o gradiente da função objetivo a zero. Essa é a generalização multidimensional da noção elementar de cálculo de pontos extremos, como ilustrado na figura 1.

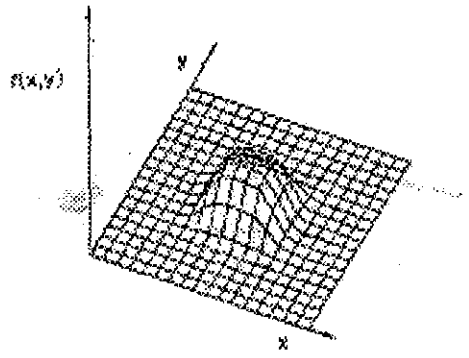

Figura 2.1 – A função com um pico simples é fácil para métodos baseados em cálculos

Dada uma função monótona e irrestrita, para se encontrar um possível pico devemos restringir a busca àqueles pontos com derivada zero em todas as direções. Por outro lado, métodos de busca diretos procuram ótimos locais galgando a função e movendo numa direção relacionada ao gradiente local. Essa é exatamente a noção de "escalada de morro": para achar o ótimo local, escalar a função na direção mais suave possível. Enquanto ambos desse métodos baseados em cálculo foram melhorados, extendidos, estudados e re-estudados, algumas razões simples mostram sua falta de robustez.

Em primeiro lugar, ambos os métodos são de alcance local; os ótimos que eles procuram são os melhores numa vizinhança do ponto atual. Por exemplo, suponha que a figura 1 mostre apenas uma porção do domínio que estamos interessados em analisar; uma figura mais completa é mostrada na figura 2.

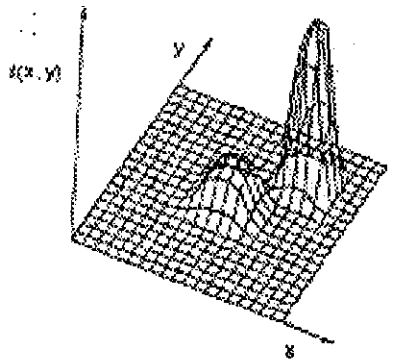

Figura 2.2 – A função com picos múltiplos causa um dilema. Que morro devemos escalar ?

Obviamente que se começarmos a busca ou aplicarmos os procedimentos para encontrarmos o "zero" da função numa vizinhança do pico menor, isso implicará no desprezo do evento principal (pico maior). Além do mais, uma vez que o pico menor é alcançado, para melhorar-se o resultado deve-se procurá-lo através de uma inicialização aleatória ou outros recursos.

Em segundo lugar, métodos baseados em cálculos dependem da existência de derivadas (valores de inclinação bem definidos). Mesmo se utilizarmos aproximação numérica de derivadas estaremos incorrendo em um erro grave. Muitos parâmetros espaciais práticos têm pouco "respeito" pela noção de uma derivada e a suavidade que isso implica. Teóricos interessados em otimização têm relutado em aceitar o legado dos grandes matemáticos dos séculos XVIII e XIX que "pintaram" um mundo limpo de funções objetivas quadráticas, restrições ideais e derivadas sempre presentes. O mundo real de procura de soluções é repleto de descontinuidades e espaços de busca vastos, multimodais e aleatórios. Concluímos sem surpresa que métodos que necessitam da

existência de restrições de continuidade e também de derivadas não são adequados para todos os problemas a não ser para problemas com domínio muito limitado. Por essa razão e por causa de seu alcance local inerente, devemos rejeitar os métodos baseados em cálculos. Eles são insuficientemente robustos em domínios não próprios.

Os métodos de busca aleatórios têm atingido popularidade crescente à medida que os pesquisadores reconheceram as falhas dos métodos enumerativos e dos baseados em cálculos. O Algoritmo Genético é um exemplo de um procedimento de busca aleatório que usa escolha aleatória como uma ferramenta para guiar uma busca altamente eficiente através de uma codificação de um parâmetro espacial.

2. O Método

Criar novos cromossomos derivados de velhos cromossomos representa a busca no espaço de soluções. Em um GA simples, um cromossomo representa um ponto no espaço de soluções e em geral, mas não necessariamente, uma representação binária é usada. Entretanto, como um exemplo, um cromossomo de tamanho 8 corresponde à seguinte cadeia,

C: 1 0 0 1 0 0 1 1

, a qual pode ser transformada em um valor em qualquer base de interesse. Os novos cromossomos na geração seguinte são resultantes dos seguintes operadores:

2.1 Seleção

Cadeias são copiadas para a câmara de acasalamento com uma probabilidade proporcional a seus valores de adaptação, assegurando então uma maior proporção das cadeias mais adaptadas.

2.2 Crossover

Dois cromossomos são selecionados aleatoriamente, e as características dessas duas cadeias são trocadas, gerando dois novos cromossomos. Sendo assim, consideremos os cromossomos A_1 e A_2, de tamanho 8, com pontos de crossover entre a terceira e a quarta posições, e entre a sexta e a sétima posições, respectivamente.

Pais :

A_1	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
B_1	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8

Filhos :

A_2	a_1	a_2	a_3	b_4	b_5	b_6	a_7	a_8
B_2	b_1	b_2	b_3	a_4	a_5	a_6	b_7	b_8

Portanto, como podemos observar, o crossover não envolve nenhuma troca nos valores atuais do cromossomo, mas simplesmente uma troca nos seus bits.

2.3 Mutação

Uma posição aleatória ao longo do cromossomo é selecionada aleatoriamente e o valor correspondente é mudado de acordo com uma dada regra probabilística. Consideremos o cromossomo C, o qual após sofrer mutação, em uma posição aleatória, por exemplo a terceira posição, origina o seguinte cromossomo C_1,

C_1 : 1 0 1 1 0 0 1 1

3. Exemplo resolvido à mão

Para fins ilustrativos e elucidativos da implementação de um algoritmo genético, lançamos mão do seguinte exemplo simulado a mão passo a passo.

Consideremos o exemplo de uma caixa preta contendo 6 interruptores que podem assumir as posições liga/desliga. Para cada arranjo efetuado com as posições dos interruptores teremos uma saída diferente, f = f(i), onde i é um arranjo particular dos interruptores. Para efeito de estudo seja $f = f(x) = x^2$, onde $0 \leq x \leq 63$.

O objetivo desse problema é arranjar os interruptores em posições liga/desliga tais que a saída assuma o valor máximo. Supondo que se utilize uma cadeia binária para representar a posição dos interruptores teremos que a cadeia 101000 corresponde à situação em que os interruptores 1 e 3 estão na posição liga e os quatro restantes na posição desliga.

Começaremos então com uma população de quatro cadeias geradas por lançamentos sucessivos de uma moeda não viciada sendo que cara representa 1 e coroa representa 0. Com esse método obtivemos as seguintes cadeias :

000110

010101

011110

100001

Para que nosso AG alcance um resultado razoável devemos utilizar os operadores básicos : reprodução, crossover e mutação.

Na reprodução, cada um dos cromossomos é copiado de acordo com o valor da sua função objetivo que mede a adaptabilidade. Copiar as cadeias de acordo com a adaptabilidade significa que cadeias com maior Fa (função objetivo) contribuirão mais para o surgimento das gerações seguintes.

Tabela I : geração da população inicial

Número	Cadeia	Fa	% do total
1	000110	36	1.46
2	010101	441	17.88
3	011110	900	36.49
4	100001	1089	44.16
Total	-	2466	100.00

O operador reprodução pode ser implementado de várias formas. Para o nosso exemplo resolvido à mão, talvez a melhor forma seja criar uma roleta em que cada indivíduo terá uma "fatia" da mesma correspondente em proporção à sua adaptabilidade. Somando os Fa das cadeias, encontramos um total de 2466. A roleta correspondente a reprodução dessa geração encontra – se ilustrada abaixo :

mais copias, os med
000110
da mesma forma e os meno
010101
podemos observar a seguir.
011110

100001

Para que nosso AG alcance um resultado razoável devemos utilizar os operadores básicos : reprodução, crossover e mutação.

Na reprodução, cada um dos cromossomos é copiado de acordo com o valor da sua função objetivo que mede a adaptabilidade. Copiar as cadeias de acordo com a adaptabilidade significa que cadeias com maior Fa (função objetivo) contribuirão mais para o surgimento das gerações seguintes.

Tabela I : geração da população inicial

Número	Cadeia	Fa	% do total
1	000110	36	1.46
2	010101	441	17.88
3	011110	900	36.49
4	100001	1089	44.16
Total	-	2466	100.00

O operador reprodução pode ser implementado de várias formas. Para o nosso exemplo resolvido à mão, talvez a melhor forma seja criar uma roleta em que cada indivíduo terá uma "fatia" da mesma correspondente em proporção à sua adaptabilidade. Somando os Fa das cadeias, encontramos um total de 2466. A roleta correspondente a reprodução dessa geração encontra – se ilustrada abaixo :

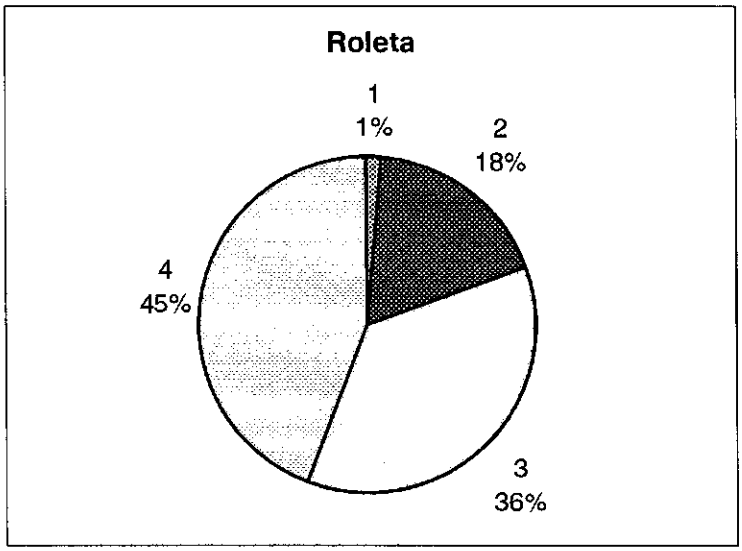

Figura 2.3 – representação esquemática da roleta

A cadeia 3, por exemplo, com um fa de 900 que representa 36.49 % do total de 2466 tem um espaço equivalente a 36.49 % da área total da roleta.

Para realizarmos a reprodução, rodamos a roleta 4 vezes. Cada rodada da roleta selecionará uma cadeia. Nesse exemplo, com a utilização de moedas, obtivemos os seguintes resultados :

- As cadeias 2 e 3 receberam 1 cópia cada ;
- A cadeia 4 recebeu 2 cópias ;
- A cadeia 1 não foi copiada.

Comparando – se os resultados obtidos com os valores esperados podemos constatar que os resultados foram coerentes, ou seja, os mais adaptados receberam mais cópias, os medianamente adaptados permaneceram da mesma forma e os menos adaptados foram excluídos, como podemos observar a seguir.

Tabela II – resultados da simulação de um AG à mão

Nº da cadeia	População inicial	Valor de x	F (x)	$\frac{f_i}{\sum f_i}$	Cadeias esperadas	Nº real de cadeias
1	000110	6	36	0.01	0.05	0
2	010101	21	441	0.18	0.71	1
3	011110	30	900	0.37	1.45	1
4	100001	33	1089	0.44	1.76	2
Soma	-	-	2466	1.00	4.00	4.00
Média	-	-	616.5	0.25	1.00	1.00
Máximo	-	-	1089	0.44	1.76	2.00

Tendo então as quatro cadeias selecionadas para a reprodução, um crossover simples é realizado em duas etapas :

1. Cadeias são "acasaladas" duas a duas de forma aleatória, através do lançamento de moedas ;
2. O crossover dos casais é realizado lançando – se moedas para determinação das posições de intercâmbio.

Observando a tabela II (continuação), constatamos que a seleção aleatória determinou que as cadeias 1 e 2 da câmara de reprodução fossem acasaladas e efetuassem crossover na posição 2. As duas cadeias (1, 2) 010101 e 100001 realizaram o crossover gerando duas cadeias 010001 e 100101. As duas cadeias restantes (3,4) realizaram crossover na posição 4 resultando duas outras cadeias que podem ser verificadas na mesma tabela.

Realizadas a reprodução, o crossover e a mutação, a nova população encontra–se pronta para ser testada. Decodificamos, então, cada cadeia da nova geração e calculamos o valor da função objetivo para cada indivíduo de acordo com os valores encontrados para x. Os resultados

obtidos podem ser melhor analisados na tabela II (continuação) disposta a seguir.

Tabela II – Continuação

Cadeias selecionadas	Par de acasalamento	Posição do crossover	Nova população	Valor de x	F (x)
010101	2	2	010001	17	289
100001	1	2	100101	37	1369
100001	4	4	100010	34	1156
011110	3	4	011101	29	841
Soma	-	-	-	-	3655
Média	-	-	-	-	913.75
Máximo	-	-	-	-	1369

Podemos ressaltar observando as duas tabelas anteriores que com apenas um salto de geração, obtivemos uma melhora no Fa médio de 616.5 para 913.75. Além disso, podemos observar também que o Fa máximo aumentou de 1089 para 1369. A cadeia que possuía o maior Fa na primeira geração, 100001, recebeu duas cópias e foi cruzada com 010101 na posição 2 e forneceu a cadeia 100101 que mostrou ser um bom resultado.

4.Implementação de um AG através do software Galopps 3.2

Antes da aplicação dos AGs na otimização do ULC da estação espacial, que é o objetivo final deste trabalho, resolveremos um problema mais simples para uma maior familiarização com o Galopps. O exemplo a seguir consiste da otimização da função $f(x)=x^{10}$ normalizada ao intervalo [0,1] e foi resolvido através do Galopps 3.2.

As rotinas de seleção para reprodução, crossover e mutação são descritas a seguir.

```
/*-------------------------------------------------------------------------*/
/* suselect.c – seleciona para reprodução                                  */
/* Codificado para SGA por Erik Goodman (EDG)                              */
/* 30 de outubro, 1993                                                     */
/*-------------------------------------------------------------------------*/

#include "external.h"

extern float randomperc();
static int *choices, nremain;

char which_selection[] = "suselect.c";

void
select_memory()
{

/* aloca memória auxiliar para seleção através de amostragem estocástica */
/* universal  */

  int nbytes;

  nbytes = popsize * sizeof(int);
  if ((choices = (int *) malloc(nbytes)) == NULL)
    nomemory("choices");
}
```

```
void
select_free()
{

/* libera memória auxiliar para seleção através de amostragem
estocástica */
/* universal */

 free(choices);
 choices = NULL;
}

void
preselect()

/* pré-seleção para o método de amostragem estocástica universal */

{
 int    j, k;
 float  pointer, sum;

 if (fit_avg == 0.) {
  for (j = 0; j < popsize; j++)
    choices[j] = j;
 }
 else {
  k = 0;
  pointer = randomperc();
  sum = 0.0;
```

```
    for (j = 0; j < popsize; j++) {
      for (sum += (oldpop[j].fitness / fit_avg); sum > pointer; pointer++) {
        choices[k++] = j;
        if(k >= popsize)
         break;
      }
      if(k >= popsize)
        break;
    }
  }
  nremain = popsize - 1;
}


int
selection()

/* seleção usando o método de amostragem estocástica universal */

{
  int    jpick, slect;

  jpick = rnd(0, nremain);
  slect = choices[jpick];
  choices[jpick] = choices[nremain];
  nremain--;
  return (slect);
}

/*-------------------------------------------------------------------------------*/
```

```
/* multimut.c – Operador de mutação Multi-bit, espacialmente    */
/* correlacionado, com probabilidade p_multimut. Ele arranja    */
/* aleatoriamente conjuntos de uma ou mais posições (bits ou    */
/* campos) onde o número de conjuntos de posições segue         */
/* aleatoriamente uma distribuição tipo Poisson com             */
/* probabilidade 1./avg_mut_width o mesmo que dizer esperada, uma */
/* vez que o número de mutações adjacentes é 1./avg_mut_width.  */
/* Também trata o cromossomo como circular. Escrito em 6/96 por */
/* EDG                                                          */
/*--------------------------------------------------------------------------*/

#include "external.h"
#include <math.h>

char which_mutation[] = "multimut.c";

void
mutation(child)

/* Essa função é chamada para o caso de cromossomo binário :
alpha_size = 2 */

unsigned *child;

{
   int start, end, change_length, mut_locus, wordnumber, bitinword, i;
   float prob_wider;
   unsigned int mask;
   p_multimut = pmutation;
```

```
/* Primeiro, decide se faz ou não a mutação para esse cromossomo */
/* usando p-multimut */

   if (flip(1. - p_multimut))
       return;
   else {

/* Fazendo uma vez com avg_mut_width, descobre-se quantos */
/* bits serão mudados (talvez 0)    */

       prob_wider = (1. - 1./avg_mut_width);
       change_length = 0;
      while (flip(prob_wider)) {
        change_length++;
       }

/* Cromossomo recebe tratamento circular */

       start = rnd(0, numfields-1);
      end = start + change_length - 1;

/* Realiza a mutação */
/* escreve cromossomo filho – chrom (child); */

      for (i=start; i<=end;i++) {
        if (flip(0.5)) {

        mut_locus = i % numfields;
          wordnumber = mut_locus/UINTSIZE;
          bitinword = mut_locus%UINTSIZE;
```

```
            mask = 1 << bitinword;
            child[wordnumber] ^= mask;
          }
        }
        nmutation++;

/* Escreve cromossomo filho – chrom (child); */
/* fprintf(outfp, "\n"); */

    }
}
```

Agora apresentamos a rotina mãe que inclui a função objetivo, decodifica cada cromossomo e avalia (calcula) a função objetivo para cada indivíduo. No Galopps 3.2, essa é a única parte que o usuário precisa implementar, uma vez que é intrínseca de cada problema. Segue a seguir a rotina.

```
/*-------------------------------------------------------------------------------*/
/* app.c – exemplo retirado do livro do Goldberg [ 6 ] e que consta no   */
/* subdiretório "examples" do Galopps 3.2, tratando – se da otimização*/
/* da função x*10, normalizada ao intervalo [0,1] usando                 */
/* cromossomos de 10bits                                                 */
/*-------------------------------------------------------------------------------*/

#include <math.h>
#include "external.h"

void
objfunc(critter)
```

```
/* Função objetivo f(x) = x**n, normalizada ao intervalo entre 0 e 1,     */
/* onde x é o cromossomo interpretado como um inteiro, e n = 10          */

struct individual *critter;

{

    unsigned mask = 1;
    unsigned bitpos;

    unsigned tp;

    double  pow(), bitpow, coef;

    int     j, k, stop;

    int     n = 10;
    neval++;
    local_cycle_neval++;

    critter->neval = neval;
    critter->init_fitness = 0.0;
    critter->fitness = 0.0;
    coef = pow(2.0, (double) lchrom) - 1.0;
    coef = pow(coef, (double) n);

    for (k = 0; k < chromsize; k++) {
        if (k == (chromsize - 1))
            stop = lchrom - (k * UINTSIZE);
```

```
        else
            stop = UINTSIZE;

        tp = critter->chrom[k];
        for (j = 0; j < stop; j++) {
            bitpos = j + UINTSIZE * k;

            if ((tp & mask) == 1) {

                bitpow = pow(2.0, (double) bitpos);

                critter->init_fitness = critter->init_fitness + bitpow;
            }
            tp = tp >> 1;
        }
    }

/* Nesse ponto o Fa é – fitness = x */
/* devemos agora aumentar x para n-ésima potência */

    critter->init_fitness = pow(critter->init_fitness, (double) n);

    /* normaliza o Fa - fitness */

    critter->init_fitness = critter->init_fitness / coef;
}
```

Com o auxílio de um **makefile**, transformamos as rotinas acima em um programa executável e através da linha de comando "Make

Onepop", o programa realiza uma iteração equivalente a uma geração, partindo da geração 0 para a geração 1. Os parâmetros usados foram :

- p.mutation = 0.0333 (probabilidade de mutação)
- p.cross = 0.6 (probabilidade de crossover)
- popsize = 30 (tamanho da população)
- length = 10 (tamanho da cadeia)

De acordo com De Jong [4], esses valores costumam funcionar bem para problemas matemáticos simples, pois combinam uma probabilidade alta de crossover, uma probabilidade de mutação baixa (inversamente proporcional ao tamanho da população) e um tamanho moderado da população.

Tabela III – resultados da simulação com o Galopps 3.2

Geração 0		Pais	Geração 1	
cadeia	Fa (fitness)		Cadeia	Fa (fitness)
1 – 111000011	0.2824	1 – (1,19)	1110000110	0.2824
2 – 110011001	0.1069	2 – (1, 9)	1101010111	0.1658
3 – 010101111	0.0000	3 – (23,19)	1101010111	0.1647
4 – 011001111	0.0001	4 – (23,19)	1111110010	0.8715
5 – 011111111	0.0009	5 – (19,1)	1110000110	0.2824
6 – 100001111	0.0359	6 – (19,1)	1101010111	0.1657
7 – 000110101	0.0000	7 – (16,1)	1110000110	0.2824
8 – 010100111	0.0000	8 – (16,1)	1100110101	0.1103
9 – 011011001	0.0002	9 – (23,17)	1011110010	0.0469
10 – 010011010	0.0000	10 – (23,17)	1100111010	0.1170
11 – 010111011	0.0000	11 – (6,26)	1011011100	0.0350
12 – 010011010	0.0000	12 – (6,26)	1100100001	0.0861
13 – 110000100	0.0636	13 – (2,19)	1100101111	0.1659
14 – 010110001	0.0000	14 – (2,19)	1100110110	0.1122
15 – 010110111	0.0000	15 – (17,17)	1100111110	0.1228
16 – 110011010	0.1103	16 – (17,17)	1100101111	0.1023
17 – 110011111	0.1243	17 – (19,17)	1101010111	0.1657
18 – 100000000	0.0010	18 – (19,17)	1100111111	0.1243
19 – 110101011	0.1657	19 – (19,19)	1101010111	0.1657
20 – 010011111	0.0000	20 – (19,19)	1101010111	0.1657

21 – 001101011	0.0000	21 – (26,17)	1100100001	0.0861
22 – 000110011	0.0000	22 – (26,17)	1100001111	0.0686
23 – 101111001	0.0469	23 – (23,1)	1110000111	0.2824
24 – 011101100	0.0004	24 – (23,1)	1011110010	0.0469
25 – 010110111	0.0000	25 – (27,1)	1010011000	0.0612
26 – 110010000	0.0861	26 – (27,1)	1010011000	0.0132
27 – 101001100	0.0134	27 – (1,17)	1100100010	0.0873
28 – 100011110	0.0031	28 – (1,17)	1110011111	0.3707
29 – 010010100	0.0000	29 – (1,19)	1101010100	0.1597
30 - 001011001	0.0000	30 – (1,19)	1110000111	0.2859

Como pode – se observar, obtivemos o maior valor de adaptabilidade para o indivíduo 4 da geração 1 (Fa = 0.8715). A média de adaptabilidade na geração 1 foi de 0.1732, o que indica que a população ainda mostra baixo desempenho apesar de apresentar alguns indivíduos com alto Fa. De uma forma geral, vemos que o AG rapidamente se direciona para o máximo, pois observa –se que a geração de partida, a geração 0, apresenta baixo desempenho. É de se esperar também que tanto a performance de cada indivíduo quanto da população em geral melhore rapidamente à medida que a simulação avança, ou seja, com um maior número de gerações.

5. Considerações finais

A utilização de AGs na otimização de problemas simples mostra a eficiência do método. Constata – se também que não existem garantias que o método convergirá para uma solução ótima, enquanto que métodos baseados em cálculos garantem a convergência para o ponto estacionário mais próximo. Além disso, os métodos de busca genética são bons em identificar regiões nas quais o ótimo global pode ocorrer.

Outra característica importante é a influência dos parâmetros (probabilidade de crossover, probabilidade de mutação, tamanho da

cadeia e tamanho da população), pois os mesmos constituem uma parcela de dificuldade razoável inerente ao método devido ao seu ajuste via tentativa e erro.

Capítulo 3

1. Simulação de um exemplo no programa de elementos finitos

Mostraremos agora um exemplo de aplicação do programa de elementos finitos. Será considerado uma estrutura formada por três vigas de seção retangular. As juntas estão rigidamente conectadas e a estrutura está engastada. A figura abaixo apresenta mais claramente o problema :

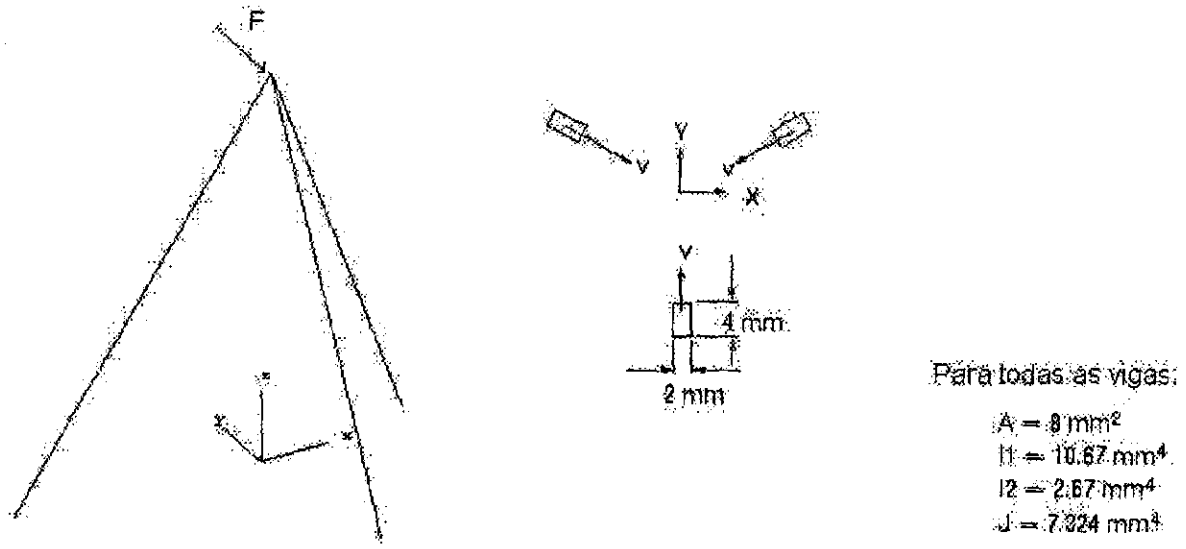

Figura 3.1 - Desenho esquemático do problema. A segunda figura mostra a orientação das vigas.

Para esse problema, o arquivo de entrada de dados é :

ef - i7-02 - teste viga L=1.0m h=4mm b=2mm
estatico

```
4 3 4 3 6
mate
9   19.9e+4  0.3  2.71e+3   1  4.0  2.0  43.3  -25.  0.
9   19.9e+4  0.3  2.71e+3   1  4.0  2.0 -43.3  -25.  0.
9   19.9e+4  0.3  2.71e+3   1  4.0  2.0   0.    1.  0.
coor
grid  1   0  -433.  250.     0.   0
grid  2   0   433.  250.     0.   0
grid  3   0     0. -500.     0.   0
grid  4   0     0.    0.  1000.   0
elem
viga01  1  1   1   4
viga01  2  2   2   4
viga01  3  3   3   4
boun
1    1 1 1 1 1 1   0.0      0.0  0.0  0.0  0.0  0.0  0.0
2    1 1 1 1 1 1   0.0      0.0  0.0  0.0  0.0  0.0  0.0
3    1 1 1 1 1 1   0.0      0.0  0.0  0.0  0.0  0.0  0.0
4    0 0 0 0 0 0   0.0  -5000.0  0.0  0.0  0.0  0.0  0.0
1.0e-08 1. 1. 1 1
```

E o arquivo saída do programa de elementos finitos será :

```
EF  - Analises Estatica e Dinamica de Estruturas
        via  Metodo dos Elementos Finitos
        versao 1.03 - agosto de 1998
        (C) UnB-FT-ENM
ef - i7-02 - teste viga L=1.0m h=4mm b=2mm
 analise                = estat
 numero de pontos nodais   =    4
```

numero de elementos = 3

numero de nos de contorno = 4

numero de materiais = 3

numero de g.lib. por no = 6

MATERIAL 1 --> PAR(1) = 0.19900E+06

MATERIAL 1 --> PAR(2) = 0.30000

MATERIAL 1 --> PAR(3) = 2710.0

MATERIAL 1 --> PAR(4) = 1.0000

MATERIAL 1 --> PAR(5) = 4.0000

MATERIAL 1 --> PAR(6) = 2.0000

MATERIAL 1 --> PAR(7) = 43.300

MATERIAL 1 --> PAR(8) = -25.000

MATERIAL 1 --> PAR(9) = 0.0000

MATERIAL 2 --> PAR(1) = 0.19900E+06

MATERIAL 2 --> PAR(2) = 0.30000

MATERIAL 2 --> PAR(3) = 2710.0

MATERIAL 2 --> PAR(4) = 1.0000

MATERIAL 2 --> PAR(5) = 4.0000

MATERIAL 2 --> PAR(6) = 2.0000

MATERIAL 2 --> PAR(7) = -43.300

MATERIAL 2 --> PAR(8) = -25.000

MATERIAL 2 --> PAR(9) = 0.0000

MATERIAL 3 --> PAR(1) = 0.19900E+06

MATERIAL 3 --> PAR(2) = 0.30000

MATERIAL 3 --> PAR(3) = 2710.0

MATERIAL 3 --> PAR(4) = 1.0000

MATERIAL 3 --> PAR(5) = 4.0000

MATERIAL 3 --> PAR(6) = 2.0000

MATERIAL 3 --> PAR(7) = 0.0000

MATERIAL 3 --> PAR(8) = 1.0000

MATERIAL 3 --> PAR(9) = 0.0000

TABELA DE COORDENADAS DOS PONTOS NODAIS:

NO	X	Y	Z
1	-433.0000	250.0000	0.0000
2	433.0000	250.0000	0.0000
3	0.0000	-500.0000	0.0000
4	0.0000	0.0000	1000.0000

TABELA DE CONEXOES NODAIS:

ELEM.	CONEXOES NODAIS			
viga01	1	1	1	4
viga01	2	2	2	4
viga01	3	3	3	4

CONDICOES DE CONTORNO:

NO	X	Y	Z			
1	1 1 1 1 1 1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00 0.00E+00
2	1 1 1 1 1 1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00 0.00E+00
3	1 1 1 1 1 1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00 0.00E+00
4	0 0 0 0 0 0	0.00E+00	-0.50E+04	0.00E+00	0.00E+00	0.00E+00 0.00E+00

tol= 0.10E-07 coef prop= 0.10E+01 dt= 0.10E+01 #passos= 1 #iter= 1

VARIAVEIS PRIMARIAS: Deslocamentos, temperatura...:

-0.4330E+03 0.2500E+03 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

0.4330E+03 0.2500E+03 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

0.0000E+00 -0.5000E+03 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00

0.0000E+00 0.0000E+00 0.1000E+04 0.0000E+00 -0.12821E+02 -0.3447E-04 0.2093E-01 0.0000E+00 0.0000E+00

FORCAS NODAIS:

-0.4330E+03 0.2500E+03 0.0000E+00 -0.1443E+04 0.8333E+03 -0.3333E+04 -0.6612E+02 0.1329E+02 -0.6043E+01

0.4330E+03 0.2500E+03 0.0000E+00 0.1443E+04 0.8333E+03 -0.3333E+04 -0.6612E+02 -0.1329E+02 0.60434E+01

0.0000E+00 -0.5000E+03 0.0000E+00 0.0000E+00 0.3333E+04 0.6666E+04 -0.1430E+02 0.0000E+00 0.0000E+00

0.0000E+00 0.0000E+00 0.1000E+04 0.0000E+00 -0.5000E+04 -0.9095E-12 0.1500E-08 0.0000E+00 0.0000E+00

BIBLIOGRAFIA

[1] Barbosa, Hélio J. C., "Introdução aos Algoritmos Genéticos", SBMAC, Rio de Janeiro, 1997.

[2] Hajela, P., "Genetic Algorithms in Automated Structural Synthesis", *Kluwer Academic Publishers,* Netherlands, 1992.

[3] Hajela, P., "Genetic Search – An Approach to the Nonconvex Optimization Problem", *AIAA Journal,* Vol. 28, NO.7, july 1990.

[4] De Jong, K. A., "Analysis of the Behavior of a class of Genetic Adaptive Systems", Ph. D. Thesis, University of Michigan, Ann Arbor, MI, 1975.

[5] Holland, J. H., "Adaptation in Natural and Artificial Systems", *University of Michigan Press,* Ann Arbor, MI, 1975.

[6] Goldberg, D. E., "Genetic Algorithms in Search Optimization and Machine Learning", *Addison – Wesley Publishing Company Inc.,* New York, 1989.

[7] Dhingra, A. K., and Lee, B. H., "A Genetic Algorithm Approach to Single and Multiobjective Structural Optimization with Discrete-Continuous Variables", *International Journal for Numerical Methods in Engineering,* Vol. 37, pp. 4059 – 4080, 1994.

[8] Lee, J. and Hajela, P., "Parallel Genetic Algorithm Implementation in Multidisciplinary Rotor Blade Design", *Journal of Aircraft,* Vol. 33, NO. 5, september – october 1996.

[9] Rao, S. S., Pan, T., and Venkayya, V. B., "Optimal Placement of Actuators in Actively Controlled Structures Using Genetic Algorithm", *AIAA Journal*, Vol. 29, NO. 6, june 1991.

[10] Fernandes, L. M., Figueiredo, I. N., Júdice, J. J., Costa, L. A. and Oliveira, P. N., "Application of Genetic Algorithms to Plate Optimization".

[11] Winter, G., Berriel, R., Cuesta, P., Galván, B., Greiner, D. and Sánchez, I., "Improving Perfomance of Genetic Algorithms in Transonic Flow".

[12] Annicchiarico, W. and Cerrolaza, M., "Structural Shape Optimization of F.E. Models Using β-Splines and Genetic Algorithms".

[13] Lemonge, A. C. C. and Barbosa, H. J. C., "A Genetic Algorithm for Optimal Bridge Pillar Location".

[14] Burczyæski, T. and Kokot, G., "Topology Optimization Using Boundary Elements and Genetic Algorithms".

[15] Cuesta, P., Galvan, B. and Winter, G., "Probabilistic Risk Assessment and Genetic Algorithms for Crude Oil Dissemination Analysis and Prevention at the Gran Canaria Coast".

[16] Priaux, J., Sefrioui, M. and Mantel, B., "Complex Aerospace Engineering Design by Genetic Algorithms".