UGF - Universidade Gama Filho

Campus Piedade – T.305/2012.2 – Período da Noite

Prof. Vitor Gonçalves INF432 – Complexidade de Algoritmo

ANÁLISE DE COMPLEXIDADE

:

Aluno Sérgio da Silva Pereira Mat. 2010160941-8

Rio de Janeiro – Outubro de 2012

1 – Objetivo:

Demonstrar de forma prática, empregando o aplicativo Matlab da The MathWorks na construção de um gráfico comparativo das funções de maior complexidade do arquivo psofinal1.c utilizado na resolução de problemas relacionados a IA – Inteligência Artificial, considerando o pior caso.

2 – Algoritmo em linguagem C do arquivo psofinal1.c analisado:

```
#include <windows.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <malloc.h>
#include <stdio.h>

#define particles (unsigned int) 20
#define max_gen (unsigned int)500
#define C1 (float)2.0 // constante congnitiva
#define C2 (float)2.0 // constante social
//#define w_ini (float)0.8
//#define w_fim (float)0.2
#define dimension (unsigned int) 2
#define dw (float) 0.01

unsigned long Seed = 176237;

//*****************************************************************************
// Random functions from Genesis (by Grefensttete)
//*****************************************************************************
#define MASK 2147483647
#define PRIME 65539
#define SCALE 0.4656612875e-9
#define Rand()  (( Seed = ( (Seed * PRIME) & MASK) ) * SCALE )
#define Randint(low,high) ( (int) (low + (high-low+1) * Rand())

//*****************************************************************************
//descrição das variaveis
//*****************************************************************************
```

```
main()
{
	int i,j,d,g,p,n,r,numero,v,z,c,b,m,k,l;
	int dim;
	float w;
	float c1,c2,r1,r2,fitmednormalizada;
	float fitness[particles];
	float min,max;
	float pbest[particles][dimension],gbest[dimension];
	float pbestfit[particles],gbestfit;
	float MIN[dimension],MAX[dimension];
	float gbestfit_anterior;
	double sumquad;
	double vmax;
	float  vmax_d;
	double v_mod;
	float gw_atual;
	float dwini,fitmed;
	float X[particles][dimension];
	double V[particles][dimension];
	float Pbest[particles][dimension];
	float w_ini,w_fim;

	// FILE *fp;
	// FILE *out ;
	// FILE *out2;
	// FILE *out3;
	FILE *out4;
	FILE *out5;
	FILE *out6;
	FILE *out7;

	numero=0;
	c1 = 2.0;
	c2 = 2.0;
	w_ini = 0.00000000000000000000000000000000000;
	w_fim = 0.00000000000000000000000000000000000;
	fitmed = -1.7E308;
	gbestfit = -1.7E308;
```

```
fitmednormalizada=-1.7E308;
dim = 2;
vmax_d = 0;
w = 0;

v_mod = 1;
// fp = fopen("pos&vel&ini.dat", "w");
// out = fopen("fitness.dat", "w");
// out2 = fopen("pbest&fitness&numero.dat", "w");
// out3 = fopen("pos&vel&final.dat", "w");
out4 = fopen("xiy.dat", "w");
out5 = fopen("xix.dat", "w");

out6 = fopen("posicao.dat", "w");
out7 = fopen("gbest.dat", "w");

// Inicializa minimos e maximos
for(  j=0; j<dimension; j++)
{
        MIN[j]=-10;
        MAX[j]=10;

       for(  i=0; i<particles; i++)  //-------------------------------------------------------------------n²
        {
                pbest[i][j]=0.0;
                pbestfit[i] = -1.7E308;
                fitness[i] = -1.7E308;
        }
}

// calculate vmax
for (numero=0; numero<particles;numero++)
{
        //fprintf( out, " %i \n  ",numero);

        sumquad = 0;

        for( j=0; j<dimension; j++) //-------------------------------------------------------------------n²
        {
```

```
                sumquad += ( MAX[j]-MIN[j])*( MAX[j]-MIN[j]);
        }

        vmax = (pow(sumquad, 0.5))/4;
}

 // inicializa velocidade e posição
 for ( i=0; i<particles;i++)
 {
        for( d=0; d<dimension; d++) //-----------------------------------------------------------------n²
        {
                vmax_d = pow( (vmax * (vmax/dim)), 0.5);
                V[i][d] = (vmax_d) * (( Rand()*2)-1);
                X[i][d] = MIN[d] + (MAX[d]-MIN[d]) * Rand();
                //fprintf( fp, " %f \t \t \t \t %f \n  ",X[i][d], V[i][d]);
        }
}

.for( g=0; g <= max_gen; g++)
 {
        //fitness
        for(i=0;i<particles; i++) //-------------------------------------------------------------------n²
        {
                fprintf( out4, " %f\t   ",X[i][0]);
                fprintf( out5, " %f\t   ",X[i][1]);
                fprintf( out4, " \n ");
                fprintf( out5, " \n ");
                for(d=0;d<dimension; d++)
                {
                        fitness[i] =  (-((X[i][0]-2)*(X[i][0]-2))-((X[i][1])*(X[i][1]))+5);//---------n³
                }

                if( fitness[i] > pbestfit[i])
                {
                        for(z=0;z < dimension;z++) //-----------------------------------------------n³
                        {
                                pbest[i][z]= X[i][z];
                        }
                        pbestfit[i] = fitness[i];
```

```
        }
}

for (i=0;i<particles;i++)
{
        if (fitness[i] > gbestfit)
        {
                for(c=0; c < dimension; c++) //-----------------------------------------n³
                {
                        gbest[c] = pbest[i][c];
                }
                gbestfit = fitness[i];
        }
}

// move
w = w_ini - ((w_ini - w_fim)/max_gen) * g ;
if (w <= w_fim)
{
        w = w_fim;
}

// Update Velocity
for (i=0; i<particles;i++)
{
        for ( l = 0; l < dimension; l++) //-----------------------------------------------------n³
        {
                r1 = Rand();
                r2 = Rand();
                V[i][l]= w*V[i][l] + r1*c1*(pbest[i][l]- X[i][l]) + r2*c2*(gbest[l] - X[i][l]);
                //fprintf( out4, " %f  \n  ",V[l]);
        }

        // Calcule v_mod
        for(m=0; m < dimension;m++ ) //-----------------------------------------------------n³
        {
                v_mod += pow(V[i][m],2.0);
        }
```

```
            v_mod = pow(v_mod,0.5);

            if ( v_mod > vmax)
            {
                  for(k=0; k < dimension; k++) //----------------------------------------------n³
                  {
                        V[i][k] = V[i][k] * (vmax/v_mod);
                  }
            }

            // Update position
            for( r=0; r < dimension; r++) //-----------------------------------------------------n³
            {
                  X[i][r] = X[i][r] + V[i][r];
                  //fprintf( out3, " %f \t\t\t %f \n  ", X[i][r],V[i][r]);

                  // condição de borda
                  while ((X[i][r]> MAX[r]) || (X[i][r] < MIN[r])) //--------------------------- n⁴
                  {
                        if (X[i][r]>MAX[r])
                        {
                              X[i][r]=2*MAX[r]-X[i][r];
                              V[i][r]=-V[i][r];
                        }

                        if (X[i][r]<MIN[r])
                        {
                              X[i][r]=2*MIN[r]-X[i][r];
                              V[i][r]=-V[i][r];
                        }
                  }
            }
      }

      fprintf( out6, " %f \t\t\t   \n  ",gbestfit);

      for(d=0;d<dimension; d++) //-----------------------------------------------------------------n²
      {
```

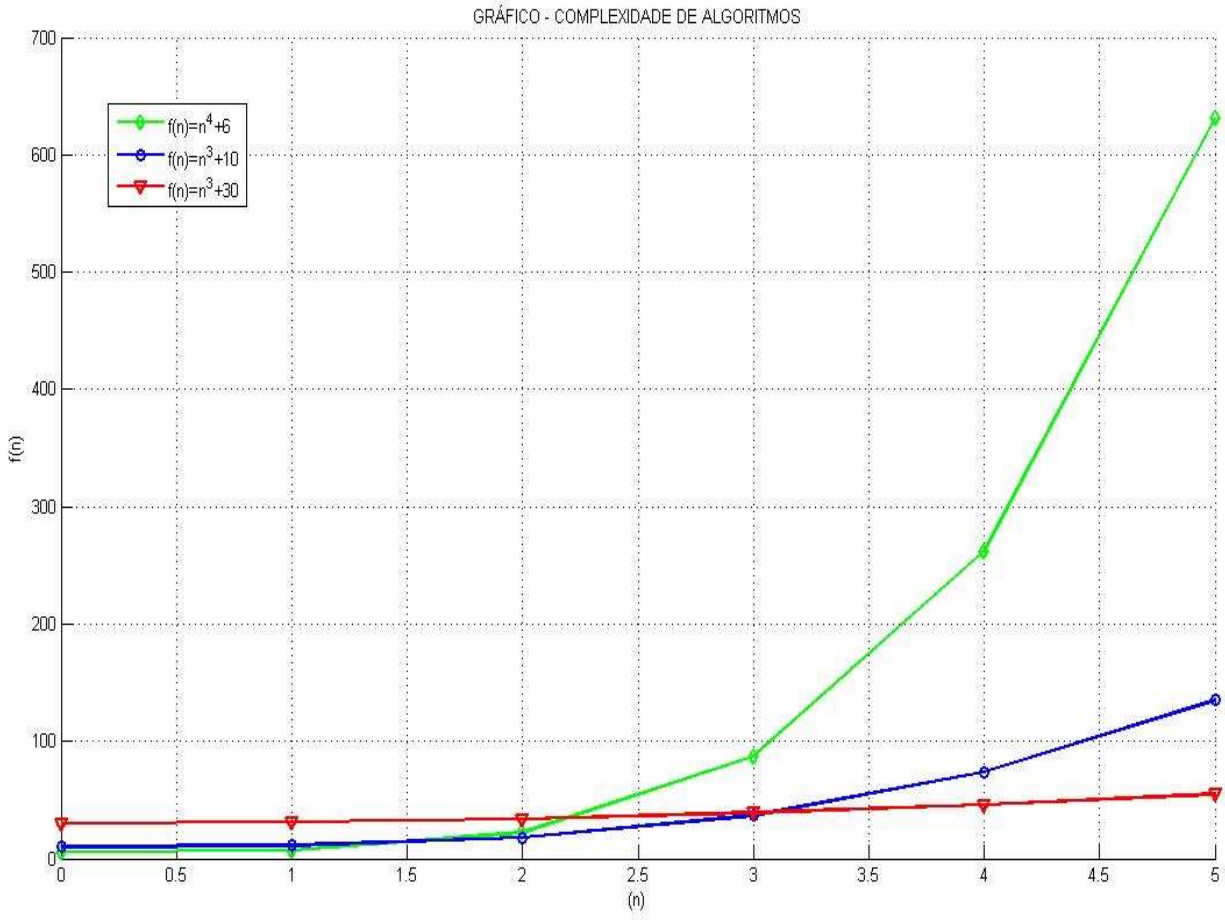
```
                    fprintf( out7, " %f \t   ", gbest[d]);
                }
                fprintf( out7, " \n\n\n");
                fprintf( out4, " \n\n ");
                fprintf( out5, " \n\n ");
        }
}
```

---

3 – Conclusão

Foram identificados três níveis de complexidade de maior magnitude no código fonte analisado, onde:

O gráfico abaixo exibe a comparação das funções de complexidade verificadas em maior número de ocorrências no algoritmo, $f(n^2)$ e $f(n^3)$ além da função de maior complexidade do pior caso, $f(n^4)$, a faixa de n foi escolhida em função do ponto de transição entre as curvas para uma melhor compreensão da dinâmica que as complexidades assumem, uma em relação a outra.



Pode-se verificar as expressões inseridas no MATLAB em seguida:

Lista – Comandos do MatLab

```
x = 0:1:5
y1 = x.^4+6
y2 = x.^3+10
y3 = x.^2+30
plot(x, y1, '-gd', x, y2, '-bo', x, y3, '-rv')
title('GRÁFICO - COMPLEXIDADE DE ALGORITMOS')
xlabel('(n)'); ylabel('f(n)')
legend('f(n)=n^4+6', 'f(n)=n^3+10', 'f(n)=n^3+30',2)
axis('auto')
axis('on')
grid('on')
```

Nessa analise pode-se constatar que o pior caso é $\eta = f(n^4) = O(n^4)$ mais as constantes do algoritmo, consideradas no cálculo e desprezadas aqui por não serem representativas no resultado do comportamento das funções com o crescimento da complexidade em função de n no pior caso.

4 – Referências

GONÇALVES, Vitor. **Notas de aulas de Complexidade de Algoritmo da matéria INF432**. Laboratórios da UGF – Universidade Gama Filho, campus da Piedade, Rio de Janeiro: 2º Período de 2012.

MATSUMOTO, Élia Yathie. MATLAB 6: Fundamentos de Programação. 2ª ed. São Paulo: Editora Érica, 2001.

ZIVIANI, Nivio. Projeto de Algoritmos com implementação em Pascal e C. 2 ed. Ver. Ampl. São Paulo: Pioneira Thompson Learning. 2004.

PUC – Pontifica Universidade Católica. **Estrutura e Dados Avançados.** Rio de Janeiro - 2010. Disponível em: < http://www.tecgraf.puc-rio.br/~mgattass/EDA/EDA_04_Complexidade.pdf> . Acessado em: 25/08/2012.