

MAXIMUM EDGE SUBGRAPH PROBLEM

Licenciatura em Engenharia Informática

Introdução à Inteligência Artificial

David José Nobre Pires (P2) / Dinis Meireles de Sousa Falcão (P5)

2019129618 / 2020130403

a2019129618@isec.pt / a2020130403@isec.pt

ÍNDICE

INTRODUÇÃO	2
OBJETIVO	3
DESENVOLVIMENTO	4
INTERFACE.....	5
FUNÇÕES UTILIZADAS	5
RESULTADOS	8
CONCLUSÃO	11
ANEXOS.....	13

INTRODUÇÃO

Este trabalho consiste em conceber, implementar e testar os métodos de otimização que encontrem soluções de boa qualidade para diferentes instâncias do *Maximum Edge Subgraph Problem*.

Dado um grafo e um valor inteiro k , o *Maximum Edge Subgraph Problem* consiste em encontrar um subconjunto de k -vértices, tal que o número de arestas dentro do subconjunto seja máximo.

Dados iniciais:

- Um grafo não direcionado $G = (V, A)$.
- Um inteiro k .

Problema:

- Encontrar um conjunto de vértices S , de tamanho k , tal que $S \subseteq V$, de forma a maximizar o número de arestas desse subconjunto.
- Problema de maximização.

OBJETIVO

Neste trabalho pretende-se implementar e avaliar a capacidade de diferentes algoritmos de otimização para encontrar soluções de boa qualidade para o problema descrito. Sendo assim, neste estudo foi implementado 3 métodos (Algoritmo de pesquisa local, algoritmo evolutivo e método híbrido combinando as duas abordagens anteriores) e efetuar um estudo comparativo aprofundado sobre o desempenho da otimização.



Figura 1 – Imagem CLion

DESENVOLVIMENTO

Como referido anteriormente, neste trabalho decidimos escolher o algoritmo Trepa-Colinas como algoritmo de pesquisa local. O nome e ideia base provem de uma analogia com a decisão tomada por um agente que, perdido numa encosta, pretende atingir o topo. Este parte de um estado inicial dado (ou gerado aleatoriamente), gera os estados sucessores do estado atual e, através de uma função de avaliação, avalia cada estado assim gerado e escolhe o de maior valor.

O Algoritmo Evolutivo é um algoritmo baseado numa gama de mecanismos de evolução biológica e serve para originar conceitos um pouco mais recentes, como o dos Algoritmos Genéticos. Este busca tratar estruturas de objetos abstratos de uma população, como, por exemplo, variáveis de um problema de otimização.

Métodos puramente heurísticos incorporam pouca informação do sistema de equações, e consequentemente não são os mais eficientes do ponto de vista computacional (especialmente nas “cercanias” da solução, onde métodos de busca local são capazes de resolver o problema). Desta forma, Métodos Híbridos de algoritmos de busca local e heurísticos de otimização parecem mais apropriados.

INTERFACE

Com base na formatação utilizada nas aulas, utilizámos a mesma para os vários métodos de otimização implementados. O programa começa pela impressão do nome do problema, ou seja, “*Maximum edge subgraph problem*”. Consoante o método utilizado, os valores solicitados serão diferentes, devido às necessidades dos diferentes métodos. Na figura 2 encontra-se um exemplo de uma das interface desses mesmos métodos, a interface da aplicação que executa o método Trepa Colinas.

```
MAXIMUM EDGE SUBRGRAPH PROBLEM!  
Nome do Ficheiro:file1.txt  
  
Numero de runs:10  
  
Numero de iteracoes:100  
  
Opcao de vizinhanca (1 = 2vz ; 2 = 4vz ; 3 = 6vz):1
```

Figura 2 – Interface da aplicação (Trepa Colinas)

FUNÇÕES UTILIZADAS

Algoritmo de pesquisa local (Trepac-Colinas):

```
int* init_dados(char *nome, int *n, int *kV);
void gera_sol_inicial(int *sol, int v, int kV);
void escreve_sol(int *sol, int vert);
void substitui(int a[], int b[], int n);
void init_rand();
int random_l_h(int min, int max);
float rand_01();
void escreve_matriz(int *matriz, int n);
bool digit_check(char key[]);
```

```
int trepa_colinas(int sol[], int *mat, int vert, int num_iter);
```

```
int calcula_fit(int a[], int *mat, int vert);
```

Algoritmo evolutivo:

```
struct info init_data(char *s, int mat[][2], int popsize, float pm, float pr, int tsize, int
numGenerations, int *grafo);
pchrom init_pop(struct info d);
void print_pop(pchrom pop, struct info d);
chrom get_best(pchrom pop, struct info d, chrom best);
void write_best(chrom x, struct info d);
void write_best_profit(pchrom x, struct info d);
void init_rand();
int random_l_h(int min, int max);
float rand_01();
int flip();
void escreve_matriz(int *matriz, int n);
bool digit_check(char key[]);
bool float_check(char key[]);
```

```
void evaluate(pchrom pop, struct info d, int mat[][2], int * grafo);
float quality(int * sol, struct info d, int mat[][2], int * grafo);

void trepa_colinas(pchrom pop, struct info d, int mat[][2]);
```

Método híbrido combinando as duas abordagens:

```
void tournament(pchrom pop, struct info d, pchrom parents);  
void genetic_operators(pchrom parents, struct info d, pchrom offspring);  
void crossover(pchrom parents, struct info d, pchrom offspring);  
void mutation(pchrom offspring, struct info d);
```

```
void evaluate(pchrom pop, struct info d, int mat[][2], int * grafo);  
float quality(int * sol, struct info d, int mat[][2], int * grafo);  
  
void trepa_colinas(pchrom pop, struct info d, int mat[][2]);
```

```
struct info init_data(char *s, int mat[][2], int popsize, float pm, float pr, int tsize, int  
numGenerations, int *grafo);  
pchrom init_pop(struct info d);  
void print_pop(pchrom pop, struct info d);  
chrom get_best(pchrom pop, struct info d, chrom best);  
void write_best(chrom x, struct info d);  
void write_best_profit(pchrom x, struct info d);  
void init_rand();  
int random_l_h(int min, int max);  
float rand_01();  
int flip();  
void escreve_matriz(int *matriz, int n);  
bool digit_check(char key[]);  
bool float_check(char key[]);
```


RESULTADOS

Algoritmo de pesquisa local (Trepa-Colinas):

Trepa-Colinas com Vizinhança 1					
NUM VERT		100 it	1000 it	5000 it	10000 it
file1.txt	Melhor	20	20	20	20
	MBF	19,6	20	20	20
file2.txt	Melhor	15	15	15	15
	MBF	13,7	15	15	14,7
file3.txt	Melhor	108	112	112	112
	MBF	104,8	109,69	109,5	109,5
file4.txt	Melhor	40	79	79	79
	MBF	33	70,4	75,8	75,5
file5.txt	Melhor	29	68	98	98
	MBF	19,5	61,79	88,59	93,09

Figura 3 – Trepa-Colinas com Vizinhança 1

Conclusão: Nesta experiência verifica se que o file3.txt obteve maior sucesso, nas melhores soluções e MBF.

Trepa-Colinas com Vizinhança 1 e aceitando soluções de custo igual					
NUM VERT		100 it	1000 it	5000 it	10000 it
file1.txt	Melhor	20	20	20	20
	MBF	19,9	20	20	20
file2.txt	Melhor	15	15	15	15
	MBF	14,9	15	15	15
file3.txt	Melhor	109	112	112	112
	MBF	105,19	112	112	112
file4.txt	Melhor	38	73	79	79
	MBF	29,4	66,09	76,3	76
file5.txt	Melhor	26	73	98	98
	MBF	19,9	63,5	91,5	91

Figura 4 – Trepa-Colinas com Vizinhança 1 e aceitando soluções de custo igual

Conclusão: Quando comparando as soluções com maior ou custo igual, é possível verificar que novamente o ficheiro file3.txt obtém as melhores soluções.

Trepa-Colinas com Vizinhaça 2

NUM VERT		100 it	1000 it	5000 it	10000 it
file1.txt	Melhor	20	20	20	20
	MBF	19,79	20	20	20
file2.txt	Melhor	15	15	15	15
	MBF	14,1	14,7	14,7	15
file3.txt	Melhor	107	112	112	112
	MBF	104,69	110,4	111,3	111,69
file4.txt	Melhor	40	79	79	79
	MBF	32,9	69	76,59	76,59
file5.txt	Melhor	32	74	93	98
	MBF	20,7	66,69	87,4	93,09

Figura 4 – Trepa-Colinas com Vizinhaça 2

Conclusão: Nesta experiência, comparando desta vez com vizinhaça dois (ou seja quatro vizinhos), é possível verificar mais uma vez que o ficheiro file3.txt se adapta bem e obtém as melhores soluções.

Algoritmo evolutivo:

file1.txt															
		Algoritmo base		com penalização		com reparação 1		com reparação 2		com mutação por troca		com recombinação 2 pontos		com recombinação uniforme	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF
ger = 2500	pr = 0.3	20	19,1												
pop = 100	pr = 0.5	20	19,1												
pm = 0.01	pr = 0.7	20	19,2												
ger = 2500	pm = 0.0	20	19,4												
pop = 100	pm = 0.001	19	19												
	pm = 0.01	19	18,9												
pr = 0.7	pm = 0.05	19	18,79												
pr = 0.7	pop = 10 (ger = 25K)	19	18,2												
pm = melhor valor obtido	pop = 50 (ger = 5K)	20	19												
pr = 0,7	pop = 100 (ger = 2.5K)	20	19,1												

Figura 5 – Algoritmo evolutivo

Conclusão: Nota-se uma pequena variância nos resultados quando o parametro “pop” e “pm” é alterado.

file2.txt															
Parâmetros Fixos	Parâmetros a variar	Algoritmo base (Recombinação de 1 ponto de corte + Mutação binária + Penalização cega)		Recombinação de 1 ponto de corte + Mutação binária + Penalização não cega		Recombinação de 1 ponto de corte + Mutação binária + Reparação1 (aleatória)		Recombinação de 1 ponto de corte + Mutação binária + Reparação2 (heurística sófrega)		Recombinação de 1 ponto de corte + Mutação por troca + Reparação2		com recombinação com 2 pontos de corte + mutação binária + reparação2		com recombinação uniforme + mutação binária + reparação2	
		Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF
pop = 100 (ger = 2500) pm = 0.01 tsize = 2	pr = 0.3	13,0	12,1												
	pr = 0.5	13,0	11,7												
	pr = 0.7	12,0	11,4												
pop = 100 (ger = 2500) pop = 100 pr = 0.7 tsize = 2	pm = 0.0	14,0	12,0												
	pm = 0.001	13,0	11,8												
	pm = 0.01	14,0	12,1												
pr = 0.7 pm = melhor valor obtido tsize = 2	pm = 0.05	14,0	11,7												
	pop = 10 (ger = 25K)	14,0	10,4												
	pop = 50 (ger = 5K)	12,0	11,4												
	pop = 100 (ger = 2.5K)	14,0	12,1												

Figura 6 – Algoritmo evolutivo

Conclusão: Não se verifica diferenças significativas.

file3.txt															
Parâmetros Fixos	Parâmetros a variar	Algoritmo base		com penalização		com reparação 1		com reparação 2		com mutação por troca		com recombinação 2 pontos		com recombinação uniforme	
		Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF
ger = 2500 pop = 100 pm = 0.01	pr = 0.3	103	100												
	pr = 0.5	103	100												
	pr = 0.7	101	99,69												
ger = 2500 pop = 100 pr = 0.7	pm = 0.0	102	100,59												
	pm = 0.001	102	100,3												
	pm = 0.01	103	100,8												
pr = 0.7 pr = 0.7 pm = melhor valor obtido	pm = 0.05	101	100												
	pop = 10 (ger = 25K)	99	96,8												
	pop = 50 (ger = 5K)	103	99,59												
	pop = 100 (ger = 2.5K)	101	99,59												

Figura 7 – Algoritmo evolutivo

Conclusão: Verifica se uma variação nos resultados aquando da alteração do parâmetro “pop”.

file4.txt															
Parâmetros Fixos	Parâmetros a variar	Algoritmo base		com penalização		com reparação 1		com reparação 2		com mutação por troca		com recombinação 2 pontos		com recombinação uniforme	
		Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF
ger = 2500 pop = 100 pm = 0.01	pr = 0.3	23	15,5												
	pr = 0.5	16	14,6												
	pr = 0.7	18	15,2												
ger = 2500 pop = 100 pr = 0.7	pm = 0.0	18	14,5												
	pm = 0.001	21	15,1												
	pm = 0.01	21	15												
pr = 0.7 pr = 0.7 pm = melhor valor obtido	pm = 0.05	20	15,5												
	pop = 10 (ger = 25K)	12	10,4												
	pop = 50 (ger = 5K)	18	14,1												
	pop = 100 (ger = 2.5K)	17	14,1												

Figura 8 – Algoritmo evolutivo

Conclusão: Alterando os parâmetros “pr” e “pop” verifica-se uma pequena diferença entre resultados.

file5.txt															
Parâmetros Fixos	Parâmetros a variar	Algoritmo base		com penalização		com reparação 1		com reparação 2		com mutação por troca		com recombinação 2 pontos		com recombinação uniforme	
		Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF
ger = 2500 pop = 100 pm = 0.01	pr = 0.3	13	9,6												
	pr = 0.5	13	10,5												
	pr = 0.7	13	9,2												
ger = 2500 pop = 100	pm = 0.0	10	8,7												
	pm = 0.001	13	9,3												
	pm = 0.01	10	9,5												
pr = 0.7	pm = 0.05	14	10,2												
	pop = 10 (ger = 25K)	9	6,6												
pm = melhor valor obtido	pop = 50 (ger = 5K)	12	8,8												
	pop = 100 (ger = 2.5K)	11	9,9												

Figura 9 – Algoritmo evolutivo

Conclusão: A diferença mais significativa, verifica se aquando da variação do parâmetro “pop”




Método Híbrido:

Falta de tempo.

CONCLUSÃO

De acordo com os resultados obtidos nos vários testes realizados, podemos comprovar que o melhor método utilizado é o algoritmo de pesquisa local, Trepa-Colinas.

ANEXOS

-  **Resultados_TrepaColinas.xlsx**
-  **Resultados_Evolutivo.xlsx**
-  **Relatório_IIA_Trabalho_Prático_2.pdf**

FIM