Trabalho Prático Nº2
Introdução à Inteligência Artificial
23/24

Realizado por:

Bruno Tiago Ferreira Martins – 2022147149
João Choupina Ferreira da Mota – 2020151878

Índice

Objetivo

Este trabalho prático consiste na implementação e teste dos métodos de otimização que encontram soluções de boa qualidade para diferentes instâncias do problema. Sendo estes métodos técnicas utilizadas para encontrar o valor mais adequado de uma determinada variável, de modo a neste caso minimizar xalguma medida de interesse.

Para isso implementamos três métodos neste trabalho de forma a encontrar as melhores soluções deste problema.

1. Algoritmo de pesquisa local(trepa-colinas);
2. Algoritmo evolutivo;
3. Método híbrido combinando as duas abordagens anteriores.

Problema apresentado

Dado um grafo e um valor inteiro k, este problema consiste em encontrar um subconjunto de vértices de tamanho k, tal que todos os vértices possuam pelo menos uma ligação e o custo das arestas dentro do subconjunto seja mínimo.

Formalmente o problema é definido:

Dados:

- um grafo não direcionado G = (V, A), composto por um conjunto V de vértices ligados entre si por arestas pesadas A
- um inteiro k

Problema:

- encontrar um subconjunto de vértices S, de tamanho k, tal que $S \subseteq V$, com todos os vértices de S com pelo menos uma ligação, de forma a minimizar o custo total das arestas desse subconjunto
- o objetivo deste problema é, portanto, de minimização.

Interface do programa

```
PS C:\Users\João Mota\Desktop\TPIIA2_TOP\Algoritmos\trepa\trepa> ./final file1.txt


------------------------------------------------------------

        1- Usar gera_vizinho1
        2- Usar gera_vizinho2
        3- Voltar

------------------------------------------------------------

>>Opcao:
```

Figura 1- Método Trepa-Colinas

```
>>Opcao: 1


------------------------------------------------------------

         1- Usar metodo normal
         2- Usar metodo probabilistico
         3- Voltar

------------------------------------------------------------

>>Opcao:
```

Figura 2- Continuação Trepa-Colinas

Análise de Resultados

Algoritmo Trepa Colinas

O algoritmo de trepa-colinas é um algoritmo de busca que é usado para encontrar uma solução ótima para um problema. Ele funciona movendo-se a partir de uma solução inicial para uma vizinhança mais próxima da solução ótima. Ele continua a iterar sobre esses movimentos até que seja encontrada a solução ótima ou até que seja atingido um ponto de máximo local, o que significa que não há soluções mais próximas da solução ótima.

.

Trepa-Colinas com Vizinhança 1					
NUM VERT		100 it	1000 it	5000 it	10000 it
8: file1.txt	Melhor	45	45	45	48
	MBF	76.570000	59.270000	56.830000	55.940000
7: file2.txt	Melhor	16	8	11	10
	MBF	68.600000	26.760000	25.780000	24.980000
16: file3.txt	Melhor	411	336	336	336
	MBF	497.720000	388.990000	371.690000	372.050000
14: file4.txt	Melhor	57	10	9	8
	MBF	926.100000	902.520000	921.090000	861.580000
15: file5.txt	Melhor	1000	1000	1000	1000
	MBF	1000.000000	1000.000000	1000.000000	1000.000000

Com o algoritmo Trepa-colinas implementado e gerada uma solução vizinha (“Vizinhança 1”) em todos os ficheiros de teste fornecidos, podemos concluir que quanto maior o número de vértices, maior será o valor da melhor solução.

Conforme fomos aumentando o número de iterações notámos um decréscimo no valor médio de todas as repetições. Resumindo, conforme o acréscimo das iterações maior será a aproximação da MBF à melhor solução.

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

NUM VERT		100 it	1000 it	5000 it	10000 it
8 : file1.txt	Melhor	45	45	45	45
	MBF	77.970000	58.360000	58.600000	58.540000
7: file2.txt	Melhor	15	8	11	10
	MBF	64.240000	25.650000	23.780000	23.680000
16: file3.txt	Melhor	423	336	336	336
	MBF	495.900000	386.010000	373.650000	372.220000
14: file4.txt	Melhor	53	14	7	7
	MBF	772.000000	99.950000	11.160000	9.600000
15: file5.txt	Melhor	1000	93	27	20
	MBF	1000.000000	973.910000	952.580000	912.620000

Conforme fomos aceitando soluções de custo igual reparámos que apenas no FILE4, nomeadamente entre as 5000 e 1000 iterações, a média foi ficando cada vez mais próxima da melhor solução.

Trepa-Colinas com Vizinhança 2

NUM VERT		100 it	1000 it	5000 it	10000 it
8 : file1.txt	Melhor	48	45	45	45
	MBF	77.450000	57.520000	54.120000	53.490000
7: file2.txt	Melhor	23	8	13	13
	MBF	63.850000	26.770000	25.530000	21.750000
16: file3.txt	Melhor	422	340	346	340
	MBF	490.070000	388.240000	396.190000	387.750000
14: file4.txt	Melhor	53	12	13	12
	MBF	944.690000	824.300000	424.400000	450.900000
15: file5.txt	Melhor	1000	1000	87	107
	MBF	1000.000000	1000.000000	973.700000	982.170000

Analisando cuidadosamente os resultados do algoritmo “gera_vizinho2” conseguimos notar uma ligeira descida nos valores da média aproximando-se ainda mais à solução ótima. Isto deve-se à troca dos dois elementos iniciais, realizando mais duas trocas, garantindo uma maior diversificação.

Trepa-Colinas probabilístico, com vizinhança 1 e aceitando soluções de custo igual

NUM VERT		100 iterações Prob = 0.01	5000 iterações Prob = 0.01	100 iterações Prob = 0.0005	5000 iterações Prob = 0.0005
8: file1.txt	Melhor	52	45	52	45
	MBF	108.060000	98.930000	95.680000	66.070000
7: file2.txt	Melhor	31	22	20	10
	MBF	105.370000	96.840000	89.140000	40.460000
16: file3.txt	Melhor	432,0	392,0	435,0	344,0
	MBF	545.460000	504.500000	521.090000	416.080000
14: file4.txt	Melhor	65,0	47,0	70,0	15,0
	MBF	832.110000	722.550000	787.080000	132.830000
15: file5.txt	Melhor	1000	1000	1000	110
	MBF	1000.000000	1000.000000	1000.000000	974.750000

Relativamente a esta tabela onde estudamos a alteração da probabilidade de aceitar uma solução pior bem como a alteração das iterações, podemos dizer que conforme a diminuição da probabilidade os resultados revelam-se melhores.

O aumento das iterações revela-se também importante nessa descida.

Algoritmo Evolutivo

O algoritmo evolutivo é um método de otimização que se inspira no processo de seleção natural darwiniano. Ele funciona criando uma população inicial de soluções possíveis para um problema e, em seguida, usando técnicas de cruzamento e mutação para gerar novas soluções a partir da população existente. As soluções são avaliadas de acordo com uma função de aptidão e aquelas que são mais aptas são selecionadas para criar a próxima geração de soluções. Esse processo é repetido várias vezes até que uma solução ótima seja encontrada ou até que um critério de parada seja atingido.

Com o algoritmo Evolutivo implementado e testado nos ficheiros “file1.txt”, “file2.txt” e “file3.txt” concluímos que o espectável seria que quanto maior o número da variável PR mais próximo estaria da solução ótima o que é completamente visível através dos testes efetuados. Com exceção do FILE3 onde ocorre precisamente o contrário.

Em relação à variável PM é visível que conforme o seu aumento mais próximo fica a média da solução ótima.

A mudança da variável POP quase ou nada impacta nesses mesmos valores.

Por fim, a alteração para maior na variável TSIZE revelou ser uma mais valia no que toca à proximidade com o valor ótimo no algoritmo base, porém quando falamos da recombinação de 2 pontos de corte + Mutação por troca + Reparação1 essa alteração revela tornar o algoritmo um quanto tanto instável.

		Algoritmo base (Recombinação de 1 ponto de corte + Mutação binária + Penalização cega)		Recombinação de 2 ponto de corte + Mutação por troca + Reparação1 (aleatória)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF
pop = 100 (ger = 2500)	pr = 0.3	45,0	57.56	45.0	58.30
pm = 0.01	pr = 0.5	45,0	56.79	45,0	55.09
tsize = 2	pr = 0.7	45,0	54.48	45,0	53.86
pop = 100 (ger = 2500)	pm = 0.0	45,0	54.45	45,0	64.13
pop = 100	pm = 0.001	45,0	55.62	45.0	55.42
pr = 0.7	pm = 0.01	45,0	54.48	45,0	54.74
tsize = 2	pm = 0.05	45,0	57.68	45,0	55.31
pr = 0.7	pop = 10 (ger = 25K)	45,0	56.51	45,0	56.82
pm = melhor valor	pop = 50 (ger = 5K)	45,0	57.72	45,0	56.32
tsize = 2	pop = 100 (ger = 2.5K)	45.0	56.37	45.0	56.47

		Algoritmo base		com reparação1 (aleatória)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF
pop = 100 (gen = 2500)	tsize = 3	45,0	58,3	45,0	55.48
pr = 0.7	tsize = 10	45,0	57.03	45,0	56.23
pm = 0.001	tsize = 50	45,0	49,0	45,0	56.16

Figura 3- File1

		Algoritmo base (Recombinação de 1 ponto de corte + Mutação binária + Penalização cega)		Recombinação de 2 ponto de corte + Mutação por troca + Reparação1 (aleatória)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF
pop = 100 (ger = 2500)	pr = 0.3	27,0	99.50	11,0	21.86
pm = 0.01	pr = 0.5	32,0	94.26	11,0	23.48
tsize = 2	pr = 0.7	32,0	89,0	8,0	23.64
pop = 100 (ger = 2500)	pm = 0.0	32,0	89,0	10.0	51.21
pop = 100	pm = 0.001	18,0	53,0	11,0	28.75
pr = 0.7	pm = 0.01	40,0	94.28	8,0	23.05
tsize = 2	pm = 0.05	41,0	108.37	14,0	23.77
pr = 0.7	pop = 10 (ger = 25K)	22,0	65.89	13,0	26.48
pm = melhor valor	pop = 50 (ger = 5K)	15,0	63.91	10,0	23.12
tsize = 2	pop = 100 (ger = 2.5K)	20,0	53.5	10,0	22.99

		Algoritmo base		com reparação1 (aleatória)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF
pop = 100 (gen = 2500)	tsize = 3	18,0	56.65	17,0	42.1
pr = 0.7	tsize = 10	20,0	56.69	14,0	36.69
pm = 0.001	tsize = 50	15,0	55.26	15,0	40.16

Figura 4- File2

		Algoritmo base (Recombinação de 1 ponto de corte + Mutação binária + Penalização cega)		Recombinação de 2 ponto de corte + Mutação por troca + Reparação1 (aleatória)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF
pop = 100 (ger = 2500)	pr = 0.3	429,0	497.54	336,0	376.81
pm = 0.01	pr = 0.5	476,0	530.14	336,0	373.56
tsize = 2	pr = 0.7	489,0	565.54	336,0	372.21
pop = 100 (ger = 2500)	pm = 0.0	477,0	583.27	351,0	432.17
pop = 100	pm = 0.001	385,0	453.60	336,0	400.79
pr = 0.7	pm = 0.01	494,0	565.30	336,0	380.54
tsize = 2	pm = 0.05	528,0	615.26	336,0	372.56
pr = 0.7	pop = 10 (ger = 25K)	573,0	661.93	336,0	381.58
pm = melhor valor	pop = 50 (ger = 5K)	554,0	626.10	336,0	372.41
tsize = 2	pop = 100 (ger = 2.5K)	529,0	611.56	336,0	377.54

		Algoritmo base		com reparação1 (aleatória)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF
pop = 100 (gen = 2500)	tsize = 3	388,0	454.86	340,0	398.24
pr = 0.7	tsize = 10	345,0	456.89	336,0	401.78
pm = 0.001	tsize = 50	375,0	449.02	341,0	398.75

Figura 5- File3

Algoritmo Híbrido

Um algoritmo híbrido é um algoritmo que combina elementos de dois ou mais algoritmos diferentes para resolver um problema. O objetivo de um algoritmo híbrido é aproveitar as vantagens de cada algoritmo individual e superar os limites de cada um deles.

Exemplos de algoritmos híbridos incluem o algoritmo de trepa-colinas evolutivo, que combina o algoritmo de trepa-colinas com o algoritmo evolutivo, e o algoritmo genético simplesmente restringido, que combina o algoritmo genético com o algoritmo de busca em largura.

Com o algoritmo híbrido implementado e testado nos ficheiros “file1.txt”, “file2.txt”, “file3.txt” concluímos que não houve uma grande alteração nos valores a medida que variamos o valor da variável PROBGERAVIZ entre 0.8 e 1.

		Algoritmo base híbrido i)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF
pop = 100 (gen = 2.5k) pr = 0.7	PROBGERAVIZ = 1	45,0	46.65
pm = 0.001 tsize = 2	PROBGERAVIZ = 0.8	45,0	46.26

Figura 6- File1

		Algoritmo base híbrido i)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF
pop = 100 (gen = 2.5k) pr = 0.7	PROBGERAVIZ = 1	14,0	20.33
pm = 0.001 tsize = 2	PROBGERAVIZ = 0.8	11,0	20.77

Figura 7- File2

		Algoritmo base híbrido i)	
Parâmetros Fixos	Parâmetros a variar	Best	MBF
pop = 100 (gen = 2.5k) pr = 0.7	PROBGERAVIZ = 1	350,0	397.36
pm = 0.001 tsize = 2	PROBGERAVIZ = 0.8	344,0	400.32

Figura 8- File3

Conclusão

Com o método Trepa-colinas, podemos concluir que o número de vértices está diretamente relacionado com o valor da solução bem como o número de iterações.

No método evolutivo, concluímos que as variáveis PR, PM e TSIZE são as mais impactantes na procura pela melhor solução, sendo a POP a menos significativa.

Por fim o método Híbrido permitiu-nos concluir que, não houve uma grande alteração nos valores a medida que variamos PROBGERAVIZ entre 0.8 e 1.

Anexos

Pasta de Ficheiros “Resultado1” Resultado1.xlsx

Pasta de Ficheiros “Resultado2” Resultado2.xlsx