Trabalho Prático nº2

Introdução à Inteligência Artificial

Relatório

Criado por: Francisco Rodrigues - 2021130296
Criado por: Ricardo Tavares – 2021144652
9 de janeiro de 2023

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 maximizar alguma 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, o Maximum Edge Subgraph Problem consiste em encontrar um subconjunto de k-vértices tal que o número de arestas dentro do subconjunto seja máximo.

Formalmente o problema é definido com um grafo não direcionado G = (V, A), composto por um conjunto V de vértices ligados entre si por arestas A e um inteiro k. Dado este grafo o objetivo é encontrar um subconjunto de vértices S, de tamanho k, tal que $S \subseteq V$, de forma a maximizar o número de arestas desse subconjunto.

Interface do programa

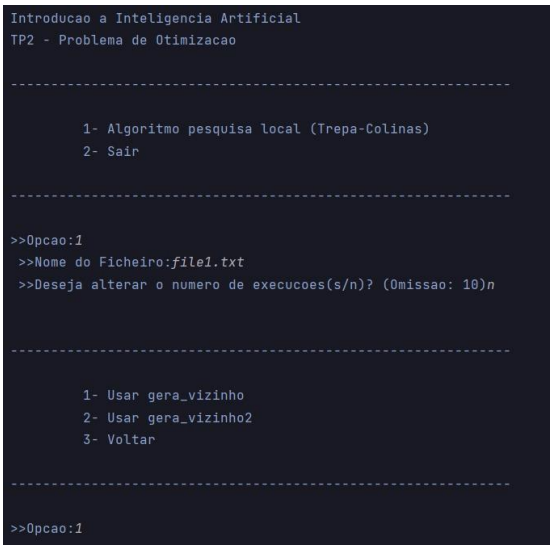

Figura 1- Método Trapa-Colinas

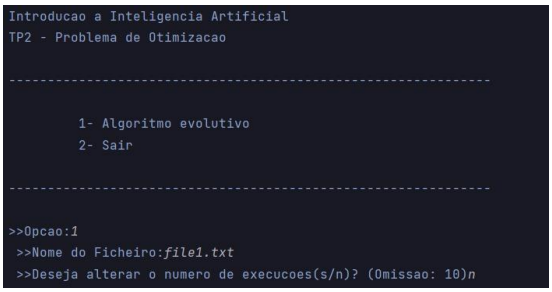

Figura 2- Método Evolutivo

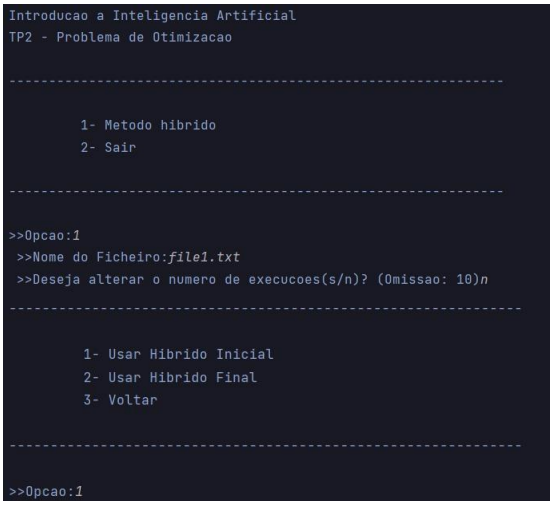

Figura 3- Método Híbrido

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.

Com o algoritmo Trepa-colinas implementado e gerada uma solução vizinha (“Vizinhança 1”) em todos os ficheiros de teste fornecidos, não podemos tirar grandes conclusões pois não conseguimos chegar aos resultados expectáveis.

Trepa-Colinas com Vizinhança 1

Ficheiro		100 it	1000 it	5000 it	10000 it
teste.txt	Melhor	5	5	5	
	MBF	2.98	3.03	2.98	
file1.txt	Melhor	27			
	MBF	12			
file2.txt	Melhor				
	MBF				
file3.txt	Melhor	117			
	MBF	59.77			
file4.txt	Melhor	15			
	MBF	9.68			
file5.txt	Melhor	7			
	MBF	5.98			

Com o algoritmo Trepa-colinas implementado e gerada uma solução vizinha (“Vizinhança 2”) em todos os ficheiros de teste fornecidos, podemos concluir que quanto maior o número de vértices, maior será o valor da melhor

solução. O número de iterações mostrou não ter grande relevância para a alteração significativa da melhor solução.

Trepa-Colinas com Vizinhança 2

Ficheiro		100 it	1000 it	5000 it	10000 it
teste.txt	Melhor	6	6	6	6
	MBF	6	6	6	6
file1.txt	Melhor	110	110	110	110
	MBF	110	110	110	110
file2.txt	Melhor	460	476	492	512
	MBF	414.32	415.08	414	415
file3.txt	Melhor	962	964	966	968
	MBF	953,65	953,81	953,73	953,8
file4.txt	Melhor	1024	1090	1096	1112
	MBF	954,47	950,9	950,69	950,8
file5.txt	Melhor	2596	2458,15	2454	2462
	MBF	2458,9	2676	2620	2670

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 “teste.txt”, “file1.txt”, “file2.txt” e “file3.txt” concluímos que o espectável seria que quanto maior o número de iterações mais próximo estaria da solução ótima o que não é completamente visível através dos testes efetuados.

O algoritmo Evolutivo também foi implementado nos ficheiros “file4.txt” e “file5.txt” mas por algum erro os valores deram todos zero o que não nos deixa tirar nenhumas conclusões.

Método Evolutivo

Ficheiro		100it	1000it	5000 it	1000 it
teste.txt	Melhor	22	22	22	22
	MBF	0,22	0,22	0,22	0,22
file1.txt	Melhor	118	122	117	121
	MBF	1,07	0,99	1,05	1,04
file2.txt	Melhor	297,0	277,0	285,0	285,0
	MBF	2,5	0,0	2,8	2,5
file3.txt	Melhor	286,0	283,0	286,0	294,0
	MBF	2,7	2,8	2,6	2,7
file4.txt	Melhor	0	0	0	0
	MBF	0	0	0	0
file5.txt	Melhor	0	0	0	0
	MBF	0	0	0	0

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 “teste.txt”, “file1.txt”, “file2.txt”, “file3.txt”, “file4.txt” e “file5.txt” concluímos que não houve uma grande alteração nos valores a medida que variamos o pr entre 0.3, 0.5 e 0.7 o pm entre 0.0, 0.001, 0.01 e 0.05 o pop entre 10, 50 e 100 e o tsize entre 3, 10 e 50.

		Algoritmo base (Recombinação de 1 ponto de corte + Mutação binária)											
		teste.txt		file1.txt		file2.txt		file3.txt		file4.txt		file5.txt	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF
pop = 100 (ger = 2500)	pr = 0.3	22,0	0,2	118,0	12,8	268,0	25,6	279,0	27,9				
pm = 0.01	pr = 0.5	22,0	0,2	118,0	12,8	281,0	24,5	275,0	24,8				
tsize = 2	pr = 0.7	22,0	0,2	118,0	12,8	283,0	22,3	274,0	26,6				
pop = 100 (ger = 2500)	pm = 0.0	22,0	0,2	118,0	12,8	265,0	25,5	287,0	26,2				
pop = 100	pm = 0.001	22,0	0,2	118,0	12,8	278,0	26,8	282,0	27,3				
pr = 0.7	pm = 0.01	22,0	0,2	118,0	12,8	283,0	22,3	274,0	26,6				
tsize = 2	pm = 0.05	22,0	0,2	118,0	12,8	271,0	24,3	269,0	25,9				
pr = 0.7	pop = 10 (ger = 25K)	22,0	0,2	118,0	12,8	269,0	25,8	279,0	26,7				
pm = melhor valor obtido	pop = 50 (ger = 5K)	22,0	0,2	118,0	12,8	283,0	27,1	277,0	27,7				
tsize = 2	pop = 100 (ger = 2.5K)	22,0	0,2	118,0	12,8	278,0	26,8	282,0	27,3				

		teste.txt		file1.txt		file2.txt		file3.txt		file4.txt		file5.txt	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF	Best	MBF
pop = 100 (gen = 2500)	tsize = 3	22,0	0,2	128,0	12,8	287,0	28,1	291,0	27,9				
pr = 0.7	tsize = 10	22,0	0,2	128,0	12,8	277,0	27,7	282,0	27,7				
pm = 0.001	tsize = 50	22,0	0,2	128,0	12,8	277,0	27,6	277,0	27,6				

Apenas para "file2.txt"		Algoritmo híbrido inicial		Algoritmo híbrido final	
Parâmetros Fixos	Parâmetros a variar	Best	MBF	Best	MBF
pop = 100 (gen = 2.5k)	PROBGERAVIZ = 1	271,0	27,1	239,0	0,0
pr = 0.7					
pm = 0.001	PROBGERAVIZ = 0.8	261,0	30,8	232,0	22,7
tsize = 2					

Conclusão

Com o método Trepa-colinas, concluímos que quanto maior o número de vértices, maior será o valor da melhor solução e o mesmo para quanto maior o número de iterações.

No método evolutivo, concluímos que quanto maior o número de iterações mais próximo estaria da solução ótima, o que não foi completamente visível através dos nossos testes nos ficheiros

Por fim o método Híbrido permitiu-nos concluir que, não houve uma grande alteração nos valores a medida que variamos o pr entre 0.3, 0.5 e 0.7 o pm entre 0.0, 0.001, 0.01 e 0.05 o pop entre 10, 50 e 100 e o tsize entre 3, 10 e 50.

Concluímos também que neste trabalho cometemos alguns erros de implementação que infelizmente não conseguimos contornar o que não nos levou definitivamente aos resultados esperados.

Anexos

Pasta de Ficheiros “Resultado1” Resultado1.xlsx

Pasta de Ficheiros “Resultado2” Resultado2.xlsx