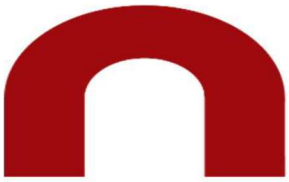

Instituto Superior de Engenharia

Politécnico de Coimbra

Trabalho Prático 2 de Introdução a Inteligência Artificial
Gráfos

Autores:
- Lucas Erardo Alves da Graça Monteiro -> 2022153271
- Timoffy Drohin -> 2022161633

Professores:
- Carlos Manuel Jorge da Silva Pereira
- Anabela Borges Simões

1. ***Representação do Problema e Função de Avaliação***:

- O código trata de um problema de otimização em grafos, onde o objetivo é encontrar um subconjunto de vértices com um custo total mínimo. A função `calculaCustoTotal` calcula o custo total de uma solução, somando os custos das arestas que conectam vértices selecionados.

- A solução para o problema de otimização é representada como um vetor de inteiros, onde cada posição no vetor corresponde a um vértice no grafo. O valor em cada posição indica se o vértice correspondente está incluído no subconjunto ótimo (geralmente representado por 1) ou não (representado por 0).

A função `validateSoluction` no código desempenha um papel crucial na verificação da validade das soluções geradas pelos algoritmos de otimização.

- `validateSoluction` é usada para verificar se uma solução gerada cumpre determinados critérios de validade. Esses critérios estão relacionados à estrutura da solução e às regras específicas do problema de otimização tratado pelo programa.

A função verifica se a solução gerada é diferente da solução inicial. Se for igual ou se o custo for o mesmo que o da solução inicial, a solução é considerada inválida. Essa parte é omitida ou retirada da função quando de pretende aceitar soluções de igual custo.

A função também conta o número de vértices incluídos na solução (indicados pelo valor 1). Se esse número for diferente de `k` (um parâmetro do problema), a solução é considerada inválida.

Para cada vértice incluído na solução, a função verifica se está conectado a pelo menos um outro vértice também incluído na solução. Isso é feito percorrendo as arestas do grafo. Se um vértice selecionado não estiver conectado a outro vértice selecionado, a solução é considerada inválida.

- A função retorna um valor inteiro, geralmente 0 ou 1, indicando se a solução é inválida (0) ou válida (1).

- Esta função é essencial para garantir que as soluções geradas pelos algoritmos de otimização, como Hill Climbing e métodos evolutivos, atendam aos critérios específicos do problema. Isso ajuda a evitar a consideração de soluções de baixa qualidade.

Em resumo, `validateSoluction` serve como um filtro de qualidade, assegurando que as soluções consideradas para futuras operações, como recombinação ou mutação, sejam válidas e atendam às exigências do problema de otimização definido.

2. *Algoritmos e Heurísticas Utilizados*:

- Hill Climbing: Uma técnica de pesquisa local onde a cada iteração se move para a solução vizinha com o melhor custo. Implementações com um e dois vizinhos são usadas (`Hill_Climbing` e `Hill_Climbing_2`).

- Recombinação (Crossover): Utilizada em algoritmos evolutivos, gera novas soluções combinando partes de duas soluções existentes. Há implementações de crossover de um ponto (`algoritmoRecombinacao_Single_Point_Crossover`) e de dois pontos (`algoritmoRecombinacao_Double_Point_Crossover`).

- Mutação: Altera aleatoriamente partes de uma solução para explorar novas regiões do espaço de solução. Implementações incluem mutação por troca (`algoritmoMutacao_Troca`) e inserção (`algoritmoMutacao_Insercao`).

- Abordagens Híbridas: Combina diferentes técnicas, como mutação seguida de Hill Climbing (`Hibrido` e `Hibrido_2`), no caso foi utilizado os 2 operadores de mutação para gerar as soluções hibridas, que gera uma solução pela mutação e faz um numero N de interações para a refinar com o Hill Climbing(Trepa-Colinas);

3. *Justificação das Opções Tomadas*:

- A diversidade de métodos, como Hill Climbing, crossover e mutação, sugere uma tentativa de balancear exploração e explotação. Isso é importante em problemas de otimização para evitar mínimos locais e explorar eficientemente o espaço de solução.

- A utilização de abordagens híbridas é uma tentativa de combinar os pontos fortes de diferentes métodos para melhorar a eficácia geral do algoritmo.

O Hill Climbing pois é o mais simples de entender e implementar e também o mais abordado em aulas, e no algoritmo evolutivo não temos grandes justificativas.

A representação da solução no código analisado é descrita da seguinte forma:

- A função `imprimirSubconjunto` é usada para imprimir o subconjunto de vértices selecionados como parte da solução. Esta função percorre o vetor da solução e imprime os índices dos vértices que fazem parte do subconjunto ótimo.

- Além disso, o custo total da solução (calculado pela função `calculaCustoTotal`) é frequentemente exibido para indicar a eficácia da solução encontrada.

- Se o vetor da solução for `[1, 0, 1, 0, 1]` para um grafo de 5 vértices, a saída será algo como "Vertices: 1 3 5", indicando que os vértices 1, 3 e 5 estão incluídos no subconjunto ótimo. O custo total associado a esta solução também será exibido.

4. *Resultados dos testes efetuados:*

Tal como poderá ver nas tabelas do Estudo dos dados.xlsx vamos mostrar aqui os resultados dos testes obtidos e a respetiva análise:

Trepa-Colinas com Vizinhança 1

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	100	66	52	52
	MBF	115.000000	94.400002	92.333336	94.000000
file2.txt	Melhor	129	129	129	129
	MBF	129.000000	129.000000	129.000000	129.000000
file3.txt	Melhor	431	437	417	427
	MBF	431.000000	442.000000	417.000000	431.500000
file4.txt	Melhor	179	189	189	189
	MBF	179.000000	189.000000	189.000000	189.000000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	539.000000	539.000000

Como se pode ver no estudo realizado conforme-me mais interações mais próximo ele chega da solução ótima, em caso de ficheiros com muitos vértices são necessários muito mais interações para causar alguma mudança significativa para chegar ao ótimo como é o caso dos file 4 e 5.

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

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	101	67	52	52
	MBF	115.000000	92.400002	92.333336	94.000000
file2.txt	Melhor	130	129	129	129
	MBF	129.000000	129.000000	129.000000	129.000000
file3.txt	Melhor	433	437	415	427
	MBF	431.000000	442.000000	415.000000	431.500000
file4.txt	Melhor	179	189	189	189
	MBF	179.000000	189.000000	189.000000	189.000000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	539.000000	539.000000

Não é tão diferente de aceitar apenas custos menores, pois sendo um problema de minimização, este procura sempre o mais baixo possível e se não encontra aceita qualquer solução valida de valor inferior aos demais.

Trepa-Colinas com Vizinhança 2

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	76	73	58	72
	MBF	91.457336	104.714287	93.167664	102.533000
file2.txt	Melhor	129	129	129	129
	MBF	129.000000	129.000000	129.000000	129.000000
file3.txt	Melhor	446	427	417	429
	MBF	446.000000	427.000000	437.250056	430.000000
file4.txt	Melhor	179	189	189	189
	MBF	179.000000	189.000000	189.000000	189.000000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	539.000000	539.000000

Este por gerar mais vizinhos tem uma visão maior do problema e consequentemente pode chegar mais rápido há solução, também quanto mais interações mais próximo do

ideal está. O único problema desta abordagem é a possibilidade de gerar uma solução que já tenha passado da solução ótima.

Recombinação Single point Crossover Reparação

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	93	122	100	91
	MBF	230.032257	230.819397	235.871048	112.533000
file2.txt	Melhor	95	93	83	65
	MBF	254.327682	257.906250	251.177292	124.650000
file3.txt	Melhor	652	593	568	429
	MBF	744.695007	738.958313	748.726929	430.000000
file4.txt	Melhor	180	69	103	98
	MBF	192.750000	79.000000	122.333336	119.000000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	539.000000	539.000000

Durante a recombinação, duas "soluções-pais" são selecionadas e combinadas para produzir "soluções-filhas". Isso é feito trocando-se partes dos cromossomos (ou sequências) dos pais. Neste caso, em um ponto de corte simples, a primeira metade da sequência de um pai é combinada com a segunda metade da sequência do outro pai. A reparação mostrou-se mais eficaz a encontrar soluções mais próximas da ótima.

Recombinação Double point Crossover Reparação

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	99	98	96	86
	MBF	216.252518	226.719803	227.724319	100.653765
file2.txt	Melhor	112	82	55	43
	MBF	244.326080	272.883362	268.070099	124.650000
file3.txt	Melhor	633	540	614	430
	MBF	741.484985	759.809998	756.219238	430.000000
file4.txt	Melhor	180	118	106	96
	MBF	192.750000	118.000000	148.000000	116.340000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	539.000000	539.000000

Neste caso, em dois pontos de corte, 50% da sequência é de um pai é combinada com a metade da sequência do outro pai, a diferença está que se pode reorganizar de mais formas gerando mais diversidade.

Recombinação Single point Crossover Penalisação

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	116	132	106	102
	MBF	214.834167	257.906250	235.871048	112.533000
file2.txt	Melhor	123	110	96	78
	MBF	247.065048	230.819397	235.726000	123.883630
file3.txt	Melhor	646	572	543	430
	MBF	762.554993	738.958313	748.726929	430.000000
file4.txt	Melhor	122	56	98	96
	MBF	122.000000	67.000000	124.672700	111.000000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	539.000000	539.000000

Recombinação Double point Crossover Penalisação

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	104	122	100	91
	MBF	253.804016	227.724319	226.719803	112.533000
file2.txt	Melhor	70	93	83	65
	MBF	242.316177	257.906250	251.177292	124.650000
file3.txt	Melhor	628	593	568	429
	MBF	750.489990	738.958313	748.726929	430.000000
file4.txt	Melhor	206	69	103	98
	MBF	206.000000	79.000000	122.333336	119.000000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	539.000000	539.000000

São as mesma funções só que utiliza o método da penalização que atribui um custo maior a soluções invalidas de forma a nunca serem escolhidas.

Mutação Troca

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	98	123	112	97
	MBF	190.550003	233.463333	221.083466	219.569641
file2.txt	Melhor	129	129	103	129
	MBF	271.049988	269.200012	283.696259	256.190369
file3.txt	Melhor	454	454	454	454
	MBF	686.109985	728.243347	718.341553	745.013733
file4.txt	Melhor	189	189	189	160
	MBF	189.000000	189.000000	189.000000	160.000000
file5.txt	Melhor	539	539	539	539
	MBF	539.000000	539.000000	523.347839	539.000000

Mutação Insersão

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	127	107	107	114
	MBF	209.781250	217.949493	229.534775	228.602280
file2.txt	Melhor	98	129	129	74
	MBF	231.867645	272.485718	242.564041	253.426834
file3.txt	Melhor	454	454	454	454
	MBF	706.530029	723.833313	740.320007	724.255005
file4.txt	Melhor	189	189	189	189
	MBF	182.578949	170.392853	189.769226	179.444443
file5.txt	Melhor	539	539	539	539
	MBF	588.043457	539.000000	522.909119	534.769226

A mutação é um conceito fundamental em Inteligência Artificial (IA), particularmente em algoritmos genéticos e evolutivos. Ela é inspirada no processo biológico de mutação e serve para introduzir variações nas soluções potenciais, ajudando a explorar o espaço de soluções de forma mais ampla e a evitar o estagnamento em ótimos locais. Vou explicar dois tipos específicos de mutação: por troca e por inserção.

Na mutação por troca, dois elementos (genes) dentro de um cromossomo (solução) são selecionados aleatoriamente e trocados de lugar. Por exemplo, em uma sequência [1, 2, 3, 4, 5], se os elementos na segunda e quinta posição são selecionados para mutação, a sequência resultante após a troca seria [1, 5, 3, 4, 2].Este tipo de mutação é útil em

problemas de otimização onde a ordem dos elementos é importante. A mutação por troca pode ter um impacto significativo na solução, especialmente se os elementos trocados estão distantes na sequência. Ela ajuda a explorar novas regiões do espaço de solução, potencialmente levando a melhores soluções.

Na mutação por inserção, um elemento é selecionado e inserido em uma posição diferente dentro da mesma sequência. Por exemplo, na sequência [1, 2, 3, 4, 5], se o elemento '3' é selecionado para mutação e inserido após o '4', a sequência resultante seria [1, 2, 4, 3, 5]. Assim como a mutação por troca, a mutação por inserção é útil em problemas onde a ordem dos elementos é crucial. Ela é particularmente útil em cenários onde a inserção de um elemento em uma posição específica pode levar a uma melhoria significativa da solução. A mutação por inserção pode alterar significativamente a estrutura da solução, permitindo uma exploração mais diversificada do espaço de soluções.

Hibrido 1

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	52	52	52	52
	MBF	57.740002	54.820000	52.850769	53.393749
file2.txt	Melhor	21	35	18	16
	MBF	70.418922	48.468502	32.407528	23.387180
file3.txt	Melhor	429	417	417	417
	MBF	429.429993	418.429993	417.866150	418.343750
file4.txt	Melhor	40	29	19	16
	MBF	87.250000	56.533333	44.187691	37.151249
file5.txt	Melhor	382	219	158	138
	MBF	449.730011	336.829987	250.660004	202.535004

Hibrido 2

NUM VERT		100 it	300 it	650 it	800 it
file1.txt	Melhor	93	67	52	52
	MBF	131.314606	51.456000	52.850769	53.393749
file2.txt	Melhor	21	28	18	16
	MBF	70.418922	48.468502	37.151249	23.387180
file3.txt	Melhor	427	421	417	417
	MBF	466.220001	411.222393	417.866150	418.343750
file4.txt	Melhor	41	36	19	16
	MBF	84.2567000	56.533333	51.456000	37.151249
file5.txt	Melhor	468	213	158	138
	MBF	536.000000	336.829987	250.660004	202.535004

Diferenças-Chave entre os 2:

Tipo de Mutação: Hibrido 1 usa mutação por inserção, enquanto Hibrido 2 usa mutação por troca. Essas abordagens diferentes afetam como as soluções são exploradas e otimizadas. Ambas as funções são úteis para resolver problemas de otimização em grafos, como encontrar o caminho mais curto, o melhor agendamento, etc., mas podem

se sair melhor em diferentes tipos de problemas devido às suas estratégias de mutação distintas.

Melhores Pesquisas

Melhor pesquisa local	Trepa colinas com 2 vizinhança
Melhor evolutivo	Recombinação Double point Crossover Reparação
Melhor Hibrido	Hibrido 1

Por fim para concluir essa é a tabela dos melhores ou os que chegaram mais perto de encontrar a solução ótima. Os motivos já foram explicados acima. Estas são as conclusões do estudo e do desenvolvimento do trabalho prático assim despedimos e esperamos ter cumprido os requisitos.