Instituto Superior de Engenharia

Politécnico de Coimbra

Introdução à Inteligência Artificial

Engenharia Informática 2023/2024

Trabalho Prático nº 2

Alexandre Ferreira – 2021138219

Carlos Pinto – 2021155627

Representação do Problema: 3
Descrição da Função de Avaliação: 3
Algoritmos Implementados 4
Trepa Colinas (Hill Climbing): 4
Algoritmo Evolutivo: 4
Algoritmo Híbrido 1 e Algoritmo Híbrido 2: 5
Justificações: 5
Representação Binária: 5
Função de Avaliação: 5
Tipo de Vizinhança: 5
Estratégia de Seleção: 5
Estratégia de Crossover: 6
Estratégia de Mutação: 6
Estratégia de Tratamento de Soluções Inválidas: 6
Análise de resultados: 7
Trepa Colinas 7
Vizinhança para Minimização de Arestas: 7
Aceitar Soluções de Custo Igual: 7
Método Evolutivo 8
Torneio vs. Roleta: 8
Crossover de Ponto vs. Crossover de Dois Pontos: 8
Mutação de Bit vs. Inversão de Sequência: 8
Penalização vs. Reparação: 8
Método híbrido 1 8
Seleção: Torneio vs. Roleta: 8
Crossover: 1 Ponto vs. 2 Pontos: 9
Mutação: Bit vs. Inversão: 9
Estratégia de Tratamento de Soluções Inválidas: Penalização vs. Reparação: . 9
Inversão de Sequência vs. Reparação de Arestas: 9
Minimização de Arestas vs Troca de vértices: 9

Método Híbrido 2 9
Torneio vs. Roleta: 9
Crossover 1 Ponto vs. 2 Pontos: 10
Mutação Bit vs. Inversão: 10
Penalização vs. Reparação: 10
Inversão de Sequência vs. Reparação de Arestas: 10
Minimização de Arestas Vs Troca de Vértices: 10
Conclusões 11

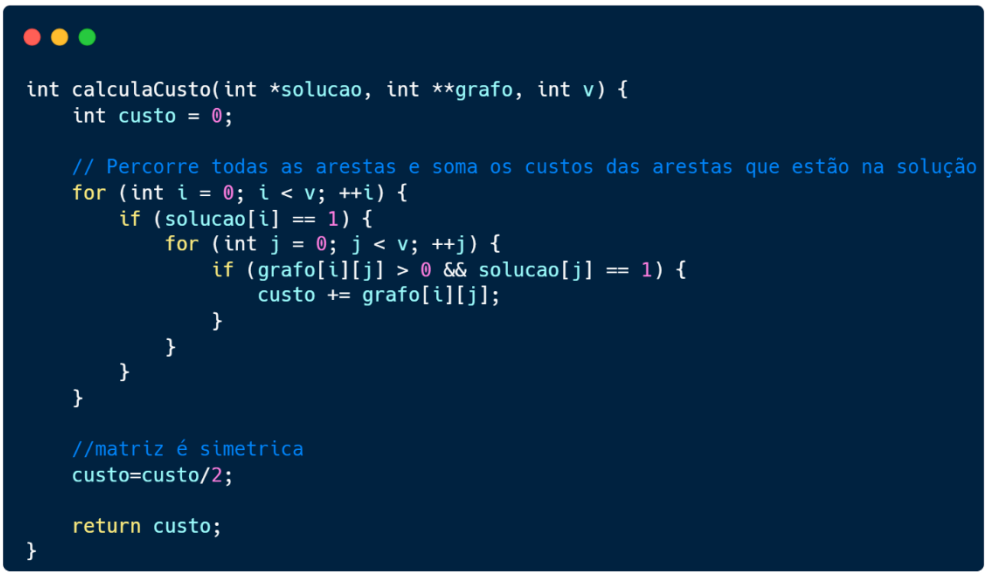
Representação do Problema:

O problema abordado neste código demonstra uma resolução para um problema da Cobertura de Vértices (Vertex Cover), no qual a meta é identificar o menor conjunto de vértices capaz de cobrir todas as arestas de um grafo.

A representação da solução é efetuada através de um vetor binário denominado "solução", onde cada posição indica se um vértice está incluído (1) ou não (0) na solução final. Essa abordagem binária revela-se eficiente e intuitiva, possibilitando a manipulação direta dos conjuntos de vértices selecionados.

Descrição da Função de Avaliação:

A função de avaliação foi nomeada como calculaCusto. Calcula o custo de uma solução ao somar os pesos das arestas cobertas pelos vértices selecionados.



```
int calculaCusto(int *solucao, int **grafo, int v) {
    int custo = 0;

    // Percorre todas as arestas e soma os custos das arestas que estão na solução
    for (int i = 0; i < v; ++i) {
        if (solucao[i] == 1) {
            for (int j = 0; j < v; ++j) {
                if (grafo[i][j] > 0 && solucao[j] == 1) {
                    custo += grafo[i][j];
                }
            }
        }
    }

    //matriz é simetrica
    custo=custo/2;

    return custo;
}
```

Algoritmos Implementados

Trepa Colinas (Hill Climbing):

O algoritmo de Trepa Colinas é a ferramenta utilizada para procurar soluções ótimas por pesquisa local. Ao explorar vizinhanças próximas da solução atual, mediante trocas de vértices e minimização de arestas, este algoritmo procura melhorar gradualmente a solução. O critério de aceitação para uma solução vizinha é duplo, envolvendo tanto um custo menor quanto a validação da solução.

Algoritmo Evolutivo:

O Algoritmo Evolutivo introduz uma perspetiva evolutiva, baseada em populações, na resolução do problema. Utilizando seleção por torneio ou proporcional, crossover de um ponto ou dois e mutação de bits ou inversão de sequência, este algoritmo tenta evoluir soluções ao longo de várias gerações. A sua finalidade é explorar de maneira abrangente o espaço de soluções em busca de configurações de baixo custo. A operação de mutação, além de adicionar diversidade à população, esforça-se para manter a validade das soluções resultantes. Sendo que além disto também são usados métodos de penalização e reparação para gerar novas soluções.

Algoritmo Híbrido 1 e Algoritmo Híbrido 2:

Estes algoritmos combinam as abordagens de Trepa Colinas e Evolutivo para otimização.

O utilizador pode escolher o tipo de estrutura de vizinhança para a Trepa Colinas e várias estratégias para o Algoritmo Evolutivo.

Os algoritmos alternam entre fases de Trepa Colinas e Evolutivo por um número especificado de iterações.

Justificações:

Representação Binária:

A escolha da representação binária para as soluções é justificada pela eficiência e facilidade de manipulação direta dos conjuntos de vértices selecionados.

Função de Avaliação:

A função de avaliação reflete precisamente o objetivo da otimização, que é a minimização do custo da solução.

Tipo de Vizinhança:

Troca de Vértices: Esta vizinhança promove a exploração local ao trocar aleatoriamente dois vértices na solução, introduzindo variações pontuais e ajustes .

Minimização de Arestas: Ao minimizar arestas na solução, procura-se uma abordagem que otimize a conectividade, influenciando a topologia da solução.

Estratégia de Seleção:

Seleção por Torneio: Favorece a escolha dos melhores indivíduos, promovendo a convergência para soluções de alta qualidade.

Seleção Proporcional à Aptidão (Roleta): Permite uma seleção mais diversificada, considerando a aptidão relativa dos indivíduos e explorando diferentes nichos do espaço de busca.

Estratégia de Crossover:

Crossover de Um Ponto: Introduz diversidade por meio da troca de informações num ponto específico, incentivando a recombinação de características promissoras.

Crossover de Dois Pontos: Oferece uma abordagem mais complexa, trocando segmentos entre dois pontos, permitindo uma maior mistura de informações genéticas.

Estratégia de Mutação:

Mutação de Bit: Introduz mudanças pontuais nos indivíduos, explorando o espaço de busca local de maneira incremental.

Inversão de Sequência: Provoca alterações mais drásticas, invertendo sequências de genes, o que pode resultar em modificações mais significativas nas soluções.

Estratégia de Tratamento de Soluções Inválidas:

Penalização: Penaliza soluções inválidas, incentivando o algoritmo a evitar essas configurações durante a evolução.

Reparação com Geração de Solução Inicial: Opta por corrigir soluções inválidas gerando novas soluções iniciais, permitindo uma abordagem mais dinâmica na adaptação do algoritmo.

Análise de resultados:

Trepa Colinas

Numero de iteracoes	100	100	100	100	100	100	1000	1000	1000	1000	1000	1000	Otima
Vizinhanca Troca de vertices	X	X	X	X			X	X	X	X			
Vizinhanca Minimização de arestas	X	X			X	X	X	X			X	X	
Aceitar solucoes de custo igual		X		X		X		X		X		X	
file1													
Melhor custo(20 rondas)	48	55	45	51	99	131	45	50	45	48	123	161	**45**
Média melhores custos(20 rondas)	68,25	79	73,25	73,85	225,85	226,4	59,85	62,35	59,75	55,75	225,1	217,6	
file2													
Melhor custo(20 rondas)	26	26	31	26	161	143	15	14	15	14	129	152	**8**
Média melhores custos(20 rondas)	59	64,65	60,95	60,4	258,75	222,75	26,2	25,6	30,3	26,45	278,2	255,75	
file3													
Melhor custo(20 rondas)	428	456	425	415	617	651	351	336	346	336	646	626	**336**
Média melhores custos(20 rondas)	499,25	496,9	488,85	483,25	730,55	753,05	399,8	382,8	393,65	384,35	740,05	745,95	
file4													
Melhor custo(20 rondas)	21	42	41	51	79	73	15	12	10	13	76	64	**7**
Média melhores custos(20 rondas)	70,65	76,15	71,8	78,2	140,1	147,25	21,35	21,65	20,2	22,1	161,65	142,75	
file5													
Melhor custo(20 rondas)	110	144	117	141	196	230	23	52	30	44	197	210	**8**
Média melhores custos(20 rondas)	225,5	246,95	261,65	250,3	336,8	358,25	78,1	83,7	84,3	87,8	308,45	324,55	
test													
Melhor custo(20 rondas)	6	6	6	6	6	6	6	6	6	6	6	6	**6**
Média melhores custos(20 rondas)	6,4	6	6,4	6	7,4	7,45	6,2	6	6,4	6	7,8	7,25	

Vizinhança para Minimização de Arestas:

A eficácia desta estratégia varia entre os conjuntos de dados. Em casos como "**file2**" e "**test**", a variação no desempenho é menor, sugerindo que a estratégia de minimização de arestas é mais robusta nestes casos. No entanto, em "**file1**" e "**file3**", a variação é mais pronunciada, indicando uma sensibilidade maior a esta escolha.

Aceitar Soluções de Custo Igual:

A estratégia de **aceitar soluções de custo igual** parece proporcionar **estabilidade em todos os testes**. Em cenários como "**test**", onde o melhor custo permanece constante, essa estratégia é eficaz. No entanto, em casos como "**file1**", onde há variação no melhor custo, a aceitação de soluções equivalentes pode impedir que o algoritmo fique preso.

Método Evolutivo

Torneio vs. Roleta:

Em geral, os testes utilizando roleta parecem oferecer melhores resultados em termos de melhores custos médios para a maioria dos conjuntos de dados.

Crossover de Ponto vs. Crossover de Dois Pontos:

Não há uma clara tendência que favoreça fortemente um tipo de crossover sobre o outro. Ambos mostraram resultados competitivos. Crossover de um ponto é mais simples, enquanto o de dois pontos permite uma maior diversidade genética.

Mutação de Bit vs. Inversão de Sequência:

A mutação de bit parece fornecer resultados mais estáveis e consistentes em comparação com a inversão de sequência. A mutação de bit é mais simples e mantém a estrutura básica da solução, enquanto a inversão de sequência pode introduzir mudanças mais drásticas.

Penalização vs. Reparação:

Ambas as estratégias de tratamento de soluções inválidas mostraram bons resultados. A penalização pode ser mais eficaz em alguns cenários, enquanto a reparação com geração de solução inicial também se mostrou eficaz.

Método híbrido 1

Seleção: Torneio vs. Roleta:

A seleção por roleta geralmente apresenta resultados ligeiramente superiores em termos de melhor custo e média de melhores custos em comparação com o torneio. Isso sugere que a seleção por roleta favorece uma exploração mais eficaz do espaço de soluções.

Crossover: 1 Ponto vs. 2 Pontos:

Configurações com crossover de 2 pontos parecem fornecer resultados comparáveis ou ligeiramente melhores em relação ao melhor custo e média de melhores custos em comparação com o crossover de 1 ponto. Crossover de 2 pontos pode permitir uma maior diversidade genética.

Mutação: Bit vs. Inversão:

A mutação de bit e a inversão de sequência apresentam desempenhos semelhantes, com pequenas variações. A escolha entre elas pode depender da natureza específica do problema.

Estratégia de Tratamento de Soluções Inválidas: Penalização vs. Reparação:

Em geral, estratégias que envolvem penalização parecem oferecer resultados mais consistentes em comparação com estratégias de reparação. A adição da minimização de arestas geralmente melhora o desempenho.

Inversão de Sequência vs. Reparação de Arestas:

Configurações que envolvem inversão de sequência e reparação de arestas têm desempenhos comparáveis. A escolha entre elas pode depender da complexidade do problema.

Minimização de Arestas vs Troca de vértices:

A estratégia de minimização de arestas tende a proporcionar soluções com melhores custos, como indicado pelos resultados de melhor custo e média de melhores custos.

Método Híbrido 2

Torneio vs. Roleta:

Torneio: Proporciona uma competição direta entre indivíduos, favorecendo a diversidade genética e, portanto, uma exploração robusta do espaço de soluções.

Roleta: Oferece uma seleção proporcional à aptidão, priorizando os mais aptos, o que pode resultar em uma convergência mais rápida para soluções de melhor qualidade. A eficácia pode depender da natureza do problema.

Crossover 1 Ponto vs. 2 Pontos:

1 Ponto: Introduz diversidade ao trocar segmentos entre dois indivíduos em um ponto específico, mantendo uma abordagem mais simples.

2 Pontos: Permite a troca de dois segmentos, potencialmente introduzindo mais variabilidade genética. Pode ser benéfico para problemas mais complexos.

Mutação Bit vs. Inversão:

Bit: Altera aleatoriamente bits individuais, proporcionando pequenas mudanças. Eficaz para manter a diversidade genética.

Inversão: Inverte sequências, introduzindo mudanças mais significativas. Útil para explorar diferentes configurações genéticas.

Penalização vs. Reparação:

Penalização: Penaliza soluções inválidas, incentivando a população a explorar alternativas. Pode ser eficaz para manter a diversidade.

Reparação: Tenta corrigir soluções inválidas, contribuindo para uma busca mais consistente por soluções válidas. A abordagem pode variar dependendo da complexidade do problema.

Inversão de Sequência vs. Reparação de Arestas:

Inversão de Sequência: Altera a ordem de genes, proporcionando diversidade genética. Pode ser eficaz para explorar diferentes configurações.

Reparação de Arestas: Concentra-se na correção de conexões inválidas. Pode ser útil para problemas em que a validade da solução é crucial.

Minimização de Arestas Vs Troca de Vértices:

Minimização de Arestas: Prioriza a redução do número de arestas na solução, potencialmente simplificando a configuração. Útil para problemas onde a quantidade de arestas impacta significativamente o custo.

Troca de Vértices: Foca na troca de vértices, buscando configurações alternativas. Pode ser eficaz para explorar diferentes combinações de vértices.

Conclusões

Com base na análise dos resultados, o método evolutivo utilizando seleção por roleta, crossover de um ponto, mutação de bit e penalização emergiu como a escolha mais consistente e eficaz. Este método demonstrou uma capacidade notável de produzir soluções de baixo custo em diferentes instâncias de teste, apresentando médias de melhores custos mais estáveis em comparação com outras configurações e métodos.

Outros métodos, como o trepa colinas com vizinhança de minimização de arestas, também apresentaram desempenho notável, especialmente em ficheiros onde a topologia das arestas desempenha um papel crítico. No entanto, o método evolutivo com roleta destacou-se pela sua capacidade de lidar de maneira mais consistente e eficiente com uma variedade de cenários, posicionando-o como a escolha mais sólida nesta análise.