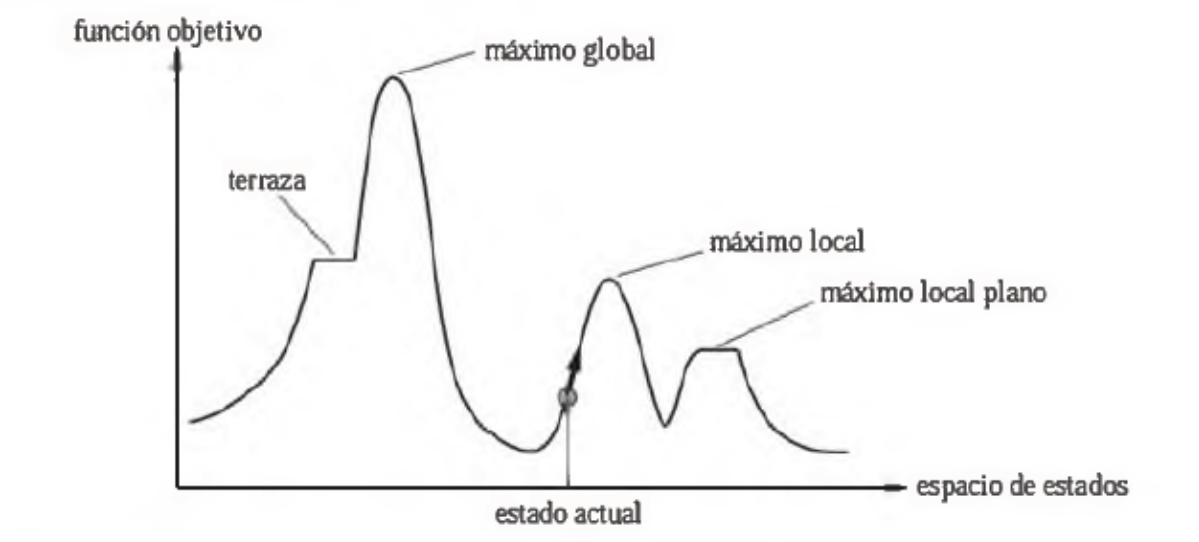
isec
Politécnico de Coimbra

Trabalho Prático

Implementação dos algoritmos de bosta local, evolutivo e híbridos.

IIA



Paulo Guilherme Lopes Sá - 2021142819 - a2021142819@isec.pt
Rúben Alexandre da Silva Martins- 2021141770 - a2021141770@isec.pt

2022/2023

Índice

Proposta de Trabalho e Objetivos:

No âmbito deste trabalho prático, propôs-se a concepção, implementação e teste de métodos de otimização destinados a identificar soluções de alta qualidade para diversas instâncias do Problema de Subgrafo de Borda Mínimo com custo por aresta.

O problema em questão envolve os seguintes elementos:

- Dados, que consistem em um grafo não direcionado composto por um conjunto V de vértices conectados por arestas A que tem um custo cada, e um inteiro K.
- Problema, que se resume a encontrar um subconjunto de vértices S, com tamanho k, pertencente a V, de modo a minimizar o custo das arestas dentro desse subconjunto e só tendo soluções que todas as arestas tenham pelo menos uma aresta a conectar com um vértice do conjunto que pretender ser a solução.

Interação com o utilizador

Iniciamos o trabalho com uma abordagem que prioriza a validação de parâmetros e o controle de entrada. Desenvolvemos um menu interativo que solicita ao usuário informações essenciais: escolher o algoritmo desejado, determinar o número de iterações desejadas e, por fim, selecionar o arquivo a ser processado.

Leitura do arquivo:

A operação de leitura dos arquivos é executada por meio da função inicializar_Dados_Ficheiro. Esta função recebe como parâmetros o nome do arquivo, a quantidade de vértices, arestas e um inteiro k, todos como ponteiros.

O processo inicia-se com a leitura dos dados de vértices, arestas e k, realizada por meio do comando fscanf diretamente do arquivo. Em seguida, é criada uma matriz dinâmica de ponteiros para ponteiros, com dimensões proporcionais ao número de vértices por vértices . Essa matriz é projetada para armazenar os custos associados a cada aresta, assumindo o valor 0 quando não há conexão entre os vértices.

Os valores são então inseridos nas posições correspondentes, configurando adequadamente a estrutura da matriz para refletir as informações extraídas do arquivo. Esse processo é crucial para estabelecer corretamente a representação do grafo em questão.

Algoritmo Trepa-colinas:

O método de Trepa-colinas emprega uma fórmula para a geração de vizinhos, otimizando a solução inicialmente criada. Nessa abordagem, uma solução inicial é gerada por meio de uma fórmula específica e, em seguida, encaminhada para a função trepa colinas, onde a vizinhança é gerada. Na execução dessa função, são criados dois vizinhos, sendo selecionado o melhor entre eles. A solução inicial é substituída por esse vizinho se for considerado superior.

A validação de uma solução é executada pela função designada "cálculo fitness". Nessa função, a verificação é realizada posição por posição da solução, com um ponteiro indicando a posição que está sendo analisada. O algoritmo percorre então a solução em busca de conexões com outros vértices. Se não houver ligação, é atribuído um custo absurdo à solução para que seja rejeitada. Por outro lado, se houver uma ligação, o custo da solução é incrementado com o custo da própria ligação. Essa abordagem permite avaliar a qualidade da solução, penalizando aquelas que não atendem às condições desejadas e recompensando as soluções que mantêm conexões apropriadas entre os vértices.

A solução inicial é gerada a tudo com 0 depois e feito um loop k vez para preencher a solução com “1” aleatoriamente e número de “1” secao igual a k a geração do vizinhança também mente o valor de número de um ao longo da algorítmo.

Trepa-colinas probabilístico:

Outro método empregado é o Trepa-colinas probabilístico. Em vez de chamar a função tradicional trepa colinas, invocamos a versão probabilística, que compartilha semelhanças com o método padrão, mas apresenta uma distinção significativa. Nesse caso, há uma probabilidade de aceitar uma solução vizinha mesmo que seja considerada pior. Essa aceitação é determinada por um valor aleatório gerado, que, se for superior à probabilidade de reprodução predefinida, resulta na aceitação da solução vizinha. Essa abordagem probabilística introduz uma variabilidade na seleção de soluções, permitindo a exploração de diferentes caminhos na busca pelo ótimo global.

Modelo Evolutivo:

O modelo evolutivo adota uma abordagem semelhante ao trepa-colinas, mas com distinções fundamentais. Embora ainda gere uma solução inicial, neste método, não é apenas uma solução isolada, mas sim uma população de soluções. Essa população é submetida a torneios para gerar descendentes mais robustos, com maior

qualidade, após um número fixo de iterações e um número definido de mutações e alterações de código realizadas pelos operadores genéticos, levando em consideração as probabilidades de mutação, recombinação e reprodução.

Operadores Genéticos:

No modelo evolutivo, quatro operadores genéticos são empregados: mutação normal, crossover, mutação_swap e crossover com três pontos de corte. Na mutação simples, um bit é invertido, transformando 1 em 0 ou vice-versa. Na mutação_swap, em vez de alterar apenas um elemento, dois são escolhidos aleatoriamente e seus valores são trocados. O crossover divide os pais em dois pontos aleatórios, gerando um filho com uma mistura de ambos. Se o número aleatório não permitir essa troca, os elementos dos pais são copiados para os filhos. No crossover com quatro pontos de corte, a solução é dividida em quatro partes.

Avaliação e Seleção ou Torneios:

Durante o ciclo de gerações, a qualidade de cada solução é avaliada. Se uma solução for superior à anterior, ela é armazenada em um ponteiro, e a qualidade é registrada em uma variável. Dois tipos de torneios foram implementados: o torneio padrão, que escolhe dois indivíduos aleatórios, verifica a qualidade e seleciona o melhor para ser pai, e o torneio5, que permite uma seleção mais ampla, facilitando o aprimoramento da população escolhendo os melhores apenas.

Reparação e penalização

para as inválidas reparamos as soluções para manter o número de k e quando uma solução não está com todas as arestas ligadas demos uma penalização como no algoritmo local.

Modelos híbridos:

Neste modelo, reunimos o aprimorado de ambos os universos: os operadores genéticos mais eficientes, o torneio de maior qualidade e o método de pesquisa local mais eficaz.

No primeiro híbrido foi feito que o evolutivo corria e depois dava a melhor solução e depois vinha o trepa colinha para incrementar esse valor e chegar ao melhor resultado e ideia era complementar e fazer um trapa coluna melhor com a ajuda de um evolutivo.

O segunda foi uma ideia completamente diferente primeiro uma solução aleatória depois o trepa colinha e resultado era posto em todos os espaços da população inicial do evolutivo e é aplicado o evolutivo para ter uma solução final.

Análise dos dados:

O problema que e nos dados temos uma grafo com algumas possibilidades para um algoritmo de busca intensiva para o file5 era preciso 18.877.913.877.607.917.786.274.849.200 tentativa para achar a melhor solução mas com os algoritmos que foram implementados podemos diminuir essa custo computacional para apenas uma fração.

Nome do ficheiro:	cálculo:	possibilidade:
teste	C(6,4) = 6! / (4!(6 - 4)!) = 6! / 4! x 2!	15
file1	C(28,8) = 28! / (8!(28 - 8)!) = 28! / 8! x 20!	3.108.105
file2	C(64,7) = 64! / (7!(64 - 7)!) = 64! / 7! x 57!	621.216.192
file3	C(70,16) = 70! / (16!(70 - 16)!) = 70! / 16! x 54!	2.480.089.880.334.220
file4	C(200,14) = 200! / (14!(200 - 14)!) = 200! / 14! x 186!	1.179.791.641.436.990.551.200
file5	C(500,15)= 500! / (15!(500 - 15)!) = 500! / 15! x 485!	18.877.913.877.607.917.786.274.849.200

Análise dos testes:

Todos os dados de análise dos testes foram retirados das experiências feitas, chegando a assim à conclusão que no modelo de pesquisa local, o modelo que mais se aproximou dos resultados pretendidos foi o Trepa Colinas.
Para o modelo evolutivo concluímos que o melhor foi o modelo evolutivo com uma probabilidade de mutação de 1 e uma probabilidade de reprodução de 1, utilizando os

operadores crossover3p e mutation com o método de seleção tournament, mesmo não chegando aos resultados pretendidos nos ficheiros 4 e 5.
Para o modelo híbrido usamos o melhor do local e o melhor do evolutivo e concluímos que o melhor modelo seria o modelo 2, é o que mais se aproxima dos valores pretendidos. O modelo híbrido 2 utiliza o algoritmo Trepa-Colinas solução Maiores para criar as soluções da população inicial e utiliza as operações Crossover3p e Mutation.

Conclusão

Ao desenvolver e implementar algoritmos como Trepa-colinas, Modelo Evolutivo e híbrido aprendemos a importância da interação eficaz com o utilizador e da representação precisa dos dados do grafo.
Os resultados obtidos mostram que, em determinados contextos, a combinação de abordagens locais e evolutivas pode levar a soluções mais eficientes.
Este trabalho contribui para nosso entendimento mais profundo e prático na área de otimização de algoritmos.