Trabalho Final de Projeto de Análise de Algoritmos

Felipe Tadeu Góes Guimarães

Instituto de Ciências Exatas e Informática (ICEI) Pontifícia Universidade Católica de Minas Gerais Belo Horizonte, Brasil ftgguimaraes@sga.pucminas.br Matheus Rangel de Figueiredo
Instituto de Ciências Exatas e Informática (ICEI)
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Brasil
matheus.figueiredo.1275135@sga.pucminas.br

I. INTRODUÇÃO

Problemas de otimização são bastante comuns em vários setores produtivos. Um exemplo de aplicação é o planejamento de recorte de peças de tecido para confecção de roupas. Uma vez que o custo total do processo depende do grau de utilização da matéria prima, um problema relevante consiste em definir a sequência de peças a serem recortadas, de modo a evitar ao máximo o desperdício de tecido.

Este trabalho tem o objetivo de implementar uma solução para esse problema de corte de peças e que minimize o desperdício total de tecido, fazendo-nos aprender, praticar e aplicar problemas de otimização ensinados ao longo do período.

II. SOLUÇÃO PROPOSTA

Os algoritmos utilizados nesse trabalho foi Brute Force com sua complexidade em n!, Branch and Bound também com complexidade em n!, porem com seu caso médio reduzido.

III. IMPLEMENTAÇÃO

Nessa secção, vamos passar por todos os detalhes dos programas usados nesse trabalho.

A. Questão 1

Nessa primeira questão, nós construímos o trapézio utilizando sua representação x1,x2,x3, também fizemos funções para calcular a areá do trapézio, o encaixe de dois trapézios, a representação cartesianas dos pontos do trapézio, o tamanho do retângulo formado pelo conjunto de trapézios, a distancia dos seus vértices e a área do desperdício.

B. Questão 2

No leitor (leitor,py) retorna a lista de peças a partir de um arquivo de texto. Sendo a primeira linha desse arquivo de texto o numero de peças a serem lidas e o resto de suas linhas a medida de x1,x2,x3 de suas peças.

33.26 -15.34 -26.5

Na (interface.py) foi feita a interface gráfica para o programa, utilizando a biblioteca gráfica OpenCV. Nela formamos uma imagem de fundo preto com altura = 100, que também é a altura padrão das peças e utilizamos a distancia de vértices para definir a posição cartesiana de cada peças.

Ai esta um exemplo de uma imagem formada a partir do arquivo lido anteriormente mostrado.

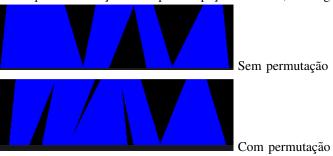


Na interface também conseguimos calcular o desperdício de múltiplas peças.

C. Questão 3

O programa calcula o desperdício total de cada permutação e verifica qual das permutações terá o menor desperdício, assim, garantindo o melhor resultado possível, porem, por fazer todas as permutações o programa fica lento. E a partir de 9 trapézios o tempo de execução começa a fica inviável. O programa pega um conjunto de peças e vê qual a permutação que traz o menor desperdício.

O tempo de execução total para 5 peças foi de 0,008 seg.



D. Questão 4

Utilizamos o método de Branch and Bound a partir da solução da forca bruta, que é um algoritmo que encontra soluções ótimas para diversos problemas de otimização, em especial combinatória. Nessa questão, utilizamos o método de

Branch and Bound a partir da solução da forca bruta, utilizando o desperdício dos nos como parâmetro para podarmos a arvore. Após alguns testas, percebemos que o branch and bound deve resultados mais rápidos, como esperado contra a permutação.

E. Questão 5

Na questão 5 é necessário criar uma heurística para ordenar um vetor de peças de forma a termos um resultado. A heurística que utilizamos foi definida através das relações geométricas, onde se a peça tem x3 negativo ela começará o encaixe e tentaremos achar uma peça que x1 seja maior do que x2 para termos um menor desperdício, já que as peças vão se encaixar de forma a gerar pequenos espaços.

IV. RELATÓRIO DE TESTES

	Número de peças	Tempo de execução	
Permutação	3	0.00	
	5	0.00300	
	7	0.60	
	8	22.381	
BranchAndBound	3	0.00099	
	5	0.004997	
	7	0.22	
	9	3.0460007190	
			taxa em de desperdício em relação ao método
			de força bruta
Heuristico	3		78.52611688486975 %
	6		68.31427292336494 %
	7		68.13200096704006 %

Essa foi a tabela de comparação que fizemos a partir de testes.

V. CONCLUSÃO

Utilizar branch and bound é sempre melhor do que permutação pois exige muito menos processamento por eliminar nós a não ser que caia no pior caso, o que é raro. E utilizar heurística mesmo não sendo ótima, ainda é uma boa forma de definir custos de desperdício.