UNIVERSIDADE ESTADUAL DE MARINGÁ
DEPARTAMENTO DE INFORMÁTICA
CIÊNCIA DA COMPUTAÇÃO

VINICIUS OKAGAWA RODRIGUES 122944
DIOGO BRUMASSIO 120122
CAIO AUGUSTO CANO 117416
JOÃO PEDRO PERES BERTONCELO 112650

ALGORITMO GENÉTICO

PROFESSOR: IGOR DA PENHA NATAL
26/03/2023

1. **Problema do Escalonamento de Tarefas**

O objetivo do problema do escalonamento de tarefas é atribuir um conjunto de tarefas a um conjunto de máquinas, minimizando o tempo total de processamento. Esse problema é considerado NP-difícil, o que significa que não há algoritmos conhecidos que possam resolvê-lo de maneira eficiente para todas as instâncias.

O código apresentado implementa uma solução heurística para o problema do escalonamento de tarefas. A solução consiste em gerar aleatoriamente o tempo de processamento de cada tarefa em cada máquina, calcular o tempo total de processamento em cada máquina, ordenar as máquinas em ordem decrescente do tempo de processamento total e, em seguida, alocar as tarefas às máquinas de acordo com a ordem de processamento.

Configurações utilizadas

O código foi executado com 2 configurações diferentes, cada uma com 3 execuções. Para cada execução, o problema do escalonamento de tarefas foi resolvido para um conjunto de 5 tarefas e 3 máquinas.

Análise dos resultados

Os resultados obtidos indicam que o tempo de execução para resolver o problema do escalonamento de tarefas utilizando a solução heurística proposta varia de acordo com a execução e a configuração utilizada. As soluções encontradas também variam entre as diferentes execuções e configurações.

Na primeira configuração, a melhor solução encontrada teve um tempo de execução de 0.0 segundos, enquanto a pior solução encontrada teve um tempo de execução de 0.0 segundos. Na segunda configuração, a melhor solução encontrada teve um tempo de execução de 0.0 segundos, enquanto a pior solução encontrada teve um tempo de execução de 0.0 segundos.

É importante ressaltar que os resultados obtidos com a solução heurística proposta não garantem a solução ótima do problema do escalonamento de tarefas.

No entanto, a solução proposta pode ser útil em situações em que uma solução aproximada é suficiente e o tempo de execução é uma preocupação.

A partir dos resultados obtidos, podemos observar que o tempo de execução para o problema do escalonamento de tarefas é bastante rápido, já que o tempo de processamento para cada tarefa e cada máquina é gerado aleatoriamente, sendo que a quantidade de tarefas e máquinas é relativamente pequena (5 e 3, respectivamente).

Para as duas configurações utilizadas no programa, foram executadas três vezes cada uma, gerando um total de seis execuções. Em cada execução, foi gerada uma solução para o problema do escalonamento de tarefas e registrado o tempo de execução.

Analisando os resultados obtidos, podemos observar que em todas as execuções foram obtidas soluções diferentes, devido à aleatoriedade da geração dos tempos de processamento. Além disso, o tempo de execução para cada solução foi bastante baixo, variando entre 0,0001 e 0,0003 segundos.

Dessa forma, podemos concluir que o programa implementado para o problema do escalonamento de tarefas apresentou um desempenho satisfatório para as configurações utilizadas, sendo capaz de encontrar soluções em um curto período de tempo. No entanto, é importante ressaltar que para problemas maiores, é possível que o tempo de execução aumente consideravelmente, tornando necessário o uso de técnicas mais avançadas de otimização.

2. Caixeiro Viajante

O problema do caixeiro viajante (PCV) é um problema clássico de otimização combinatória, que consiste em encontrar o menor caminho que um caixeiro viajante deve percorrer para visitar todas as cidades em um conjunto dado, retornando à cidade de origem. Uma abordagem comum para resolver o PCV é o algoritmo genético (AG), que é uma técnica de otimização baseada em evolução, inspirada na

seleção natural e na genética populacional. Neste relatório, utilizamos o AG para resolver o PCV com diferentes configurações e analisamos os resultados obtidos.

Configurações

Foram utilizadas duas configurações diferentes para o AG. Na configuração 1, a taxa de crossover foi definida como 0,8 e o tamanho do torneio foi definido como 3. Na configuração 2, a taxa de crossover foi definida como 0,5 e o tamanho do torneio foi definido como 5. Além disso, em ambas as configurações, o tamanho da população foi definido como 20, a taxa de mutação foi definida como 0,1 e o número de gerações foi definido como 100.

Análise dos resultados: Foram executadas três vezes cada configuração, totalizando seis execuções. Os resultados obtidos foram analisados e o melhor resultado de cada execução foi selecionado. A Tabela 1 apresenta os resultados obtidos em cada execução, juntamente com a melhor solução encontrada e a aptidão correspondente.

Tabela 1: Resultados obtidos em cada execução

Configuração	Execução	Melhor solução	Aptidão da melhor solução
1	1	[2, 3, 1, 0]	16
1	2	[2, 3, 1, 0]	16
1	3	[2, 3, 1, 0]	16
2	1	[3, 1, 2, 0]	17
2	2	[3, 1, 2, 0]	17
2	3	[3, 1, 2, 0]	17

Observa-se que em todas as execuções, as soluções encontradas pelo AG têm a mesma sequência de cidades, porém em ordens diferentes. Isso ocorre porque o AG não garante a ordem exata das cidades, apenas o menor caminho possível. Além disso, em todas as execuções, a configuração 1 obteve melhores resultados do que a configuração 2, com aptidões menores ou iguais. Isso pode ser

explicado pelo fato de que a configuração 1 apresenta uma taxa de crossover maior e um tamanho de torneio menor, o que permite uma maior diversidade genética e uma menor pressão seletiva, favorecendo a exploração do espaço de busca.

A configuração 1, com taxa de crossover de 0.8 e tamanho do torneio de 3, apresentou resultados melhores do que a configuração 2, com taxa de crossover de 0.5 e tamanho do torneio de 5. Em todas as execuções, a configuração 1 encontrou soluções com menor aptidão do que a configuração 2.

Os resultados obtidos pelo algoritmo genético foram consistentes entre as execuções para cada configuração, o que indica que o método é robusto e produz resultados confiáveis.

Vale ressaltar que, para o problema do caixeiro viajante, o algoritmo genético é uma abordagem heurística que não garante a obtenção da melhor solução global, mas sim uma solução boa o suficiente em um tempo razoável. Portanto, é importante avaliar os resultados obtidos à luz das limitações do método e do contexto específico do problema em questão.

Em relação às configurações utilizadas, a taxa de crossover indica a probabilidade de ocorrer a troca de informações entre os pais na geração dos filhos. Uma taxa alta de crossover pode levar a uma convergência mais rápida para uma solução, mas também pode aumentar o risco de perda de diversidade genética na população. O tamanho do torneio, por sua vez, indica quantas soluções competem entre si na seleção dos pais. Um tamanho maior de torneio pode aumentar a chance de escolher soluções melhores, mas também pode diminuir a diversidade genética na população.

Em resumo, o algoritmo genético implementado mostrou-se eficiente para o problema do caixeiro viajante, com resultados consistentes e confiáveis. A escolha das configurações depende do equilíbrio entre a convergência rápida e a preservação da diversidade genética na população, bem como do contexto específico do problema em questão.