Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática

Implementação de Memória Compartilhada e Distribuída no Algoritmo Genético

Disciplina: Programação Concorrente

Docente: Prof. Igor da Penha Natal

Discentes:
Renan Augusto Leonel ra: 115138
Felipe Roveroni de Lima ra: 112675
José Rafael Silva Hermoso ra: 112685
Pedro Henrique de Melo Costa ra: 112653

Maringá - PR
2022

Sumário

1. Descrição do problema

O problema do caixeiro viajante (PCV) é um problema de otimização que consiste basicamente em determinar o menor caminho possível para percorrer uma série de pontos de forma que cada ponto seja visitado apenas uma vez, e volte ao ponto de origem. Devido a sua complexidade de resolução, a utilização de algoritmos genéticos para sua resolução é comumente empregada.

Um algoritmo genético (AG) é um algoritmo probabilístico inspirado na evolução natural das espécies e na genética. É uma técnica que envolve uma busca adaptativa baseada na sobrevivência dos indivíduos mais aptos da população, juntamente com a reprodução, aplicado para problemas de otimização.

Um AG é constituído dos elementos básicos a seguir:

- Cromossomo: um indivíduo
- Gene: um elemento do cromossomo
- População: conjunto de cromossomos
- Crossover (cruzamento): cruzamento dos cromossomos pais para gerar novos cromossomos filhos
- Seleção: seleciona os cromossomos para o crossover
- Mutação: modificação arbitrária de uma pequena parte do cromossomo

Para o PCV, os cromossomos representam uma solução (ciclo hamiltoniano), um gene representa uma parte deste ciclo, a população representa um conjunto de soluções que são os cromossomos e a mutação se dá pela troca aleatória de genes de uma solução. Para selecionarmos os indivíduos, devemos abstrair para uma roleta de probabilidades, simulando uma escolha onde os melhores indivíduos têm mais chances de serem escolhidos. Em seguida, a reprodução é abstraída para um algoritmo que faz o cruzamento de dois indivíduos pré-selecionados. Além desses elementos, também temos a geração de uma população inicial, o critério de parada, o critério de avaliação da população, o critério para ocorrer uma mutação e a forma de realizar a manutenção da população. Logo abaixo na figura 1 pode-se observar um AG (Algoritmo Genético) em pseudocódigo.

```
Gerar população inicial
enquanto critério-de-parada faça:
    avaliação da qualidade dos cromossomos da população
    seleção para o cruzamento
    cruzamento
    mutação
    atualização da população
```

Imagem 1 - Pseudocódigo de um algoritmo genético.

Para este trabalho, foram desenvolvidas duas implementações de algoritmos genéticos, onde a primeira utiliza memória compartilhada, e a segunda utiliza memória distribuída, que serão detalhadas nas seções seguintes, utilizando como entrada o arquivo '*att48.txt*', um arquivo txt com 48 vértices de um grafo, anexado com o trabalho.

2. Descrição da técnica utilizada

Para este trabalho, foram utilizadas duas técnicas de implementação: utilizando memória compartilhada e distribuída.

Uma implementação por memória compartilhada consiste de um método de comunicação de processos, utilizando uma memória global no sistema que será dividida entre todos os processadores presentes, onde a execução será paralelizada em várias threads. É um método rápido de comunicação entre os processos, mas não escalável, pois os processos estão sendo executados em um só computador. Além disso, a memória compartilhada pode ser local, onde é exclusiva a um único processador, ou global, onde todos os processadores têm acesso.

Uma implementação por memória distribuída considera que cada processo possui sua própria memória, estabelecendo comunicação entre si através de mensagens privadas, oferecendo como vantagens principalmente a escalabilidade, pois ao manter cada processador com sua memória privada, evitamos os gargalos presentes em uma memória compartilhada, por exemplo. Quanto ao processamento na memória distribuída, os mesmos são conectados por uma rede e comunicam-se através do envio de mensagens. Como dito anteriormente, cada processador tem sua própria memória local, portanto um processador não consegue acessar a memória de outro processador, então cabe ao programador sincronizar as tarefas entre os processadores.

Primeiramente, temos uma implementação do algoritmo genético utilizando memória compartilhada. Foi escolhido o método de seleção por roleta juntamente com os cruzamentos oX e cX. Detalhes de como foi desenvolvida a paralelização serão abordados na próxima seção, juntamente com algumas questões de implementação.

3. Descrição da paralelização da técnica

Como o problema abordado neste trabalho envolve um procedimento iterativo de testes, ou seja, devemos executar várias vezes o mesmo algoritmo para realizar

uma análise do comportamento da solução e dos parâmetros utilizados. A paralelização e distribuição ficou com a tarefa de executar uma instância do algoritmo genético. Quando usamos *threads* compartilhadas, *cada thread* executa um algoritmo genético e quando usamos memória distribuída, cada membro da rede também executa um algoritmo genético. Cada execução tem a sua configuração de parâmetros e retorna o resultado e o tempo de execução.

3.1 Memória Distribuída

Com o memória compartilhada, foi utilizado uma estrutura de *Client/Server* para simular um gerenciador que fornece parâmetros para os *Clients* que se conectarem a ele. A conexão é feita via protocolo *TCP*. O *Server*, possui uma lista de parâmetros, cada parâmetro possui um tamanho de população, taxa de mutação, e critério de parada (número de gerações que será executado.

Cada *Client*, ao se conectar, deve enviar uma mensagem “ok” para o servidor para representar que a conexão foi estabelecida. Em seguida o servidor escolhe um parâmetro na sua lista de parâmetros e envia para esse *client*. Após a execução, o *Client* retorna o resultado final e o tempo de execução e encerra a conexão. Enquanto o servidor estiver ligado qualquer *Client* pode estabelecer uma conexão e receber uma configuração de parâmetros.

3.2 Memória Compartilhada

A forma de aplicar uma memória compartilhada para o algoritmo genético utilizado, foi paralelizar as execuções, por mais que cada execução não vai realizar troca de informações com as outras, sendo assim uma paralelização que não utiliza troca de mensagens.

Foi utilizado *threads,* onde cada um vai executar uma instância do algoritmo genético e retornar o resultado para a *thread main*.

4. Resultados Obtidos

Para a análise dos resultados, foram utilizados testes com a mesma configuração de parâmetros.

4.1 Algoritmo genético *single thread*

Resultados de 8 execuções para os parâmetros:
Resultado final : 42515.9 Tempo de execução: 8.25000 sec

Resultado final : 43787.5 Tempo de execução: 8.14062 sec
Resultado final : 43015.5 Tempo de execução: 8.29688 sec
Resultado final : 49441.5 Tempo de execução: 8.23438 sec
Resultado final : 44405.4 Tempo de execução: 8.26562 sec
Resultado final : 48657.6 Tempo de execução: 8.62500 sec
Resultado final : 48205.8 Tempo de execução: 8.42188 sec
Resultado final : 46559.1 Tempo de execução: 8.73438 sec

4.2 Algoritmo genético com 4 *threads*

Resultado de 8 threads:
Resultado final 1 : 46892.3 Tempo de execução: 10.7812 sec
Resultado final 1 : 55957.9 Tempo de execução: 11.1406 sec
Resultado final 1 : 67804.0 Tempo de execução: 13.3750 sec
Resultado final 1 : 43055.3 Tempo de execução: 22.1094 sec
Resultado final 2 : 43616.5 Tempo de execução: 10.4062 sec
Resultado final 2 : 46823.1 Tempo de execução: 10.9062 sec
Resultado final 2 : 54480.9 Tempo de execução: 13.2500 sec
Resultado final 2 : 54691.6 Tempo de execução: 22.5312 sec

4.3 Algoritmo genético distribuído com dois *clients*

Resultado de 8 execuções dos *clients*:
Resultado final : 40042.9 Tempo de execução: 8.39062 sec
Resultado final : 46337.3 Tempo de execução: 8.20312 sec
Resultado final : 43599.3 Tempo de execução: 8.18750 sec
Resultado final : 42737.7 Tempo de execução: 8.87500 sec
Resultado final : 45073.0 Tempo de execução: 8.71875 sec
Resultado final : 44362.8 Tempo de execução: 9.48438 sec
Resultado final : 49847.1 Tempo de execução: 9.73438 sec
Resultado final : 40731.4 Tempo de execução: 8.87500 sec

5. Análise dos Resultados

Com base nos resultados encontrados, podemos concluir que a memória distribuída e os threads afetam positivamente a execução dos algoritmos genéticos, visto que os mesmos necessitam de diversas execuções dada sua característica heurística de não garantir sempre o mesmo resultado (além também do resultado ótimo). Já quanto ao tempo de execução, notamos que em todos os testes o tempo é bem similar. Portanto, tanto com threads quanto com o uso distribuído, podemos

executar diversas instâncias ao mesmo tempo, resultando num ganho de tempo para a calibragem dos parâmetros e também nos testes dos algoritmos genéticos.