Estratégias de Paralelização do Algoritmo Genético para Problemas de Roteamento

Felipe Nathan Welter[1]

[1]Centro de Ciências Tecnológicas – Universidade do Estado de Santa Catarina (UDESC)
89.219-710 – Joinville – SC – Brazil

felipenwelter@gmail.com

Resumo. *A aplicação do algoritmo genético como técnica para solução do problema de roteamento de veículos permite a ampla utilização de estratégias de paralelismo, assim como flexibilidade para atender a diferentes ambientes computacionais. A partir de um estudo de caso realizado em uma mineradora, esse artigo tem por objetivo apresentar uma breve contextualização dos conceitos aplicados e realizar o comparativo com outros modelos de implementação a fim de se identificar possibilidades de melhoria ao modelo proposto.*

1. Introdução

O Problema de Roteamento de Veículos (VRP, do inglês Vehicle Routing Problem) é um dos problemas mais conhecidos em otimização combinatória. Sua solução consiste em encontrar o melhor conjunto de rotas a serem percorridas por uma frota de veículos atendendo à algumas restrições específicas tal como janela de tempo ou capacidade de carga. Considerado um problema da classe NP-difícil, a busca pela melhor resposta exige grande tempo de processamento e por esse motivo normalmente é solucionado por meio de um algoritmo probabilístico ou pela aplicação de heurísticas que encontram soluções boas o suficiente em tempo reduzido. Na prática, o VRP pode ter uma série de restrições que o distingue em categorias mais específicas, assim como uma série de abordagens possíveis para solucioná-lo, que variam desde métodos exatos como o algoritmo branch-and-bound, heurísticas construtivas ou ainda metaheurísticas, como é o caso do Algoritmo Genético, solução adotada pelo autor no estudo de caso analisado.

2. Algoritmo Genético

Métodos heurísticos são algoritmos exploratórios que buscam a solução de problemas. Em Ciência da Computação, o desenvolvimento de um algoritmo busca atender a duas propriedades: ter um tempo de execução aceitável e encontrar a solução ótima, ou provavelmente boa, para o problema em todos os casos. [Bueno 2009].

O algoritmo genético é uma das técnicas da computação evolucionária de busca e otimização inspirada na evolução natural das espécies de Darwin. A idéia é criar uma população de indivíduos que vão se reproduzir e competir pela sobrevivência, onde os melhores indivíduos sobrevivem e transferem suas características para novas gerações, até que se alcance uma solução próxima ao resultado ótimo [Pozo et al. 2003]. No VRP, um algoritmo genético consiste nas seguintes etapas: (1) criação da população onde cada indivíduo representa um ponto de origem, destino e rota; (2) avaliação de aptidão (fitness), que visa minimizar a distância média de transporte, diminuir a formação de filas e o reduzir a ociosidade das máquinas; (3) reprodução, que se dá pela seleção de indivíduos aptos e crossover e (4) mutação para evolução dos indivíduos.

3. Estudo de caso

A operação da frota de mineração consiste no transporte do produto de um ponto de carregamento onde o material é extraído até um ponto de descarregamento. O sistema de despacho é que realiza o monitoramento e controle de indicadores como posição dos equipamentos, percurso, velocidade, carga entre diversos outros fatores, buscando otimizar o tempo e capacidade de operação e reduzindo custos de manutenção.

O estudo de caso busca aplicar uma solução embarcada de menor custo focada especialmente na otimização de rotas, através da aplicação de um algoritmo genético paralelizado com modelo de ilhas através de programação OpenMP [Neto 2018]. Segundo o autor, a aplicação de inteligência artificial é pouco explorada em sistemas embarcados. O crescimento do IOT aliado ao baixo custo dos dispositivos é que permite sua aplicação de forma cada vez mais abrangente.

Cada veículo é equipamento com um sistema GPS que fornece suas cordenadas de latitude e longitude. O gatilho para cálculo do percurso é feito em três momentos: fim do carregamento, fim do descarregamento e no cruzamento de rotas, com a limitação de 1 segundo para execução. No estudo de caso o autor utilizou-se de dispositivos distintos para comparativo: Raspberry Pi2 4-core, Raspberry Pi3 4-core e Banana Pi 8-core. O processamento é realizado localmente no dispositivo embarcado, havendo tráfego de informações para um servidor central apenas para coleta de informações de localização. A classificação das minas segue a seguinte configuração: pequeno porte - 5 a 40 equipamentos, 4 origens, 2 destinos e 8 cruzamentos; médio porte - 1 a 110 equipamentos, 6 origens, 3 destinos e 32 cruzamentos; e grande porte - 1 a 170 equipamentos, 8 origens, 5 destinos e 62 cruzamentos.

Os resultados apresentados indicam que, conforme o número de equipamentos aumenta, maior é o tempo de execução do algoritmo, mas também maior é o ganho da execução paralela em relação à sequencial. O hardware utilizado é um fator determinante e os melhores resultados foram obtidos no equipamento com maior número de núcleos, o BananaPI M3, alcançando-se speedups de 1.08, 1.30 e 1.20 em minas de pequeno, médio e grande porte respectivamente. Em determinadas situações não foi possível atender à restrição de tempo, o que não permite ao motorista a tomada de ação em tempo hábil.

4. Análise de implementações alternativas

Visando uma contribuição à proposta apresentada nesse estudo de caso, algumas considerações podem ser discutidas visando a resolução de algumas das limitações apresentadas e o aumento da eficiência da solução.

Um primeiro aspecto a se observar é em relação ao hardware dos dispositivos embarcados, que precisam ser compatíveis às restrições específicas do cenário apresentado e às limitações orçamentárias da empresa. A aquisição de equipamentos mais potentes, de gerações mais atuais e com maior número de núcleos de processamento poderia trazer um ganho considerável de performance e sem inviabilizar o custo do projeto, o que foi comprovado pela análise comparativa entre os modelos RasperryPI 2 e 3 apresentado pelo autor, onde o speedup variou de 1.15 para 1.29 nas minas de grande porte. A customização do hardware através do aumento do clock do CPU, ou ainda a adoção de compilações especiais para os sistemas operacionais, mantidos originais durante o estudo de caso, também poderiam contribuir.

Outro fator limitante e pouco explorado pelo autor é a latência de rede na busca das informações das coordenadas dos outros veículo em um servidor central, que pode comprometer a performance do processo. Realizar a sincronização das informações em intervalos regulares, de forma assíncrona, poderia eliminar a espera pela sincronização durante a execução do algoritmo genético.

A estratégia de paralelização em modelo de ilha aplicada no estudo seria, talvez, o aspecto com maior potencial de ganho. O algoritmo genético pode ser adequado para trabalhar em diferentes estruturas computacionais. No modelo master-slave as operações de selection, crossover e mutation são realizadas pelo mestre, que distribui, entre os nós filhos, cada indivíduo para cálculo do fitness, muito útil quando a função fitness realiza operações mais complexas para atender restrições de negócio. No modelo de ilhas, a população é divida em subpopulações que evoluem de forma independente, contando com uma etapa de migração para replicar os melhores resultados, o que pode ser feito de forma assíncrona, sem barreira para sincronismo entre threads.

No modelo em células, cada indivíduo é executado por um dispositivo independente, havendo troca de informações entre vizinhos nas etapas de crossover e mutation, sendo que a topologia de conexões é um fator determinadnte para o processo e que permite maior nível de paralelismo [Reza 2015]. Há ainda o modelo hierárquico, que permite a mescla de diferentes estruturas em camadas, garantindo grande flexibilidade para se adequar a diferentes ambientes computacionais. A exemplo, poderia se trabalhar com o modelo de ilhas em uma população (através de threads) e ao mesmo tempo realizar a troca de informações com outros nós da rede, no modelo em células.

A escolha do autor pelo modelo de ilhas é assertiva visto as especificidades do cenário de mineração. O modelo master-slave não seria uma boa escolha pois as regras de avaliação fitness não são entendidas como gargalo de processamento. Os modelos em célula e hierárquico teriam sua aplicação mais relacionada a ambientes multi-computadores e não multi-processadores. A função de migração ocasiona perda de desempenho em cenários mais complexos [Neto 2018] e nessa situação Zheng et al. (2014) indica que o modelo assíncrono apresentaria melhor performance. Na prática, os melhores indivíduos poderiam ser dispostos em uma memória compartilhada e conforme cada ilha concluísse a avaliação de sua população utilizaria os melhores indivíduos disponíveis até o momento, o que não acarretaria perdas de performance ou qualidade da solução.

De forma complementar é importante salientar que a definição dos parâmetros do algoritmo genético, tal como o número de populações e a taxa de migração, podem afetar significativamente a velocidade de processamento e a qualidade da solução, portanto precisam ser configurados de acordo com a realidade de cada caso, sugerindo-se ainda a avaliação a partir de experimentações práticas.

Para concluir, a otimização das rotas de transporte descrita no estudo de caso permite a redução de custos operacionais e aumenta a produtividade e eficiência do processo produtivo, tendo como grande diferencial o pequeno custo de implantação através de uma solução embarcada, ao invés das soluções robustas disponíveis no mercado, incompatível com a realidade orçamentária de pequenas mineradoras. A otimização do algoritmo genético, por sua vez, deve considerar não apenas o tempo de processamento, mas principalmente a qualidade da solução [Zheng et al. 2014].

Referências

Bueno, F. (2009). *Métodos Heurísticos: Teoria e Implementações*. IFSC. Disponível em: http://bit.ly/30hooR9. Acesso em: 30 de junho de 2019.

Neto, G. D. (2018). Paralelismo de um algoritmo genético, aplicado a otimização de rotas em mineração de modo escalável. Master's thesis, Universidade Federal de Uberlândia, Uberlândia. Disponível em: https://repositorio.ufu.br/bitstream/123456789/25374/3/ParalelismoAlgoritmoGenetico.pdf. Acesso em: 16 out. 2019.

Pozo, A. et al. (2003). *Computação Evolutiva*. Disponível em: http://www.inf.ufpr.br/aurora/tutoriais/Ceapostila.pdf. Acesso em: 7 de julho de 2019.

Reza, Roshani; Karim Sohrabi, M. (2015). Parallel genetic algorithm for shortest path routing problem with collaborative neighbors. *Ciência e Natura*, 37:328 – 333. Disponível em: https://www.redalyc.org/articulo.oa?id=467547683042. Acesso em: 20 out. 2019.

Zheng, L. et al. (2014). Architecture-based design and optimization of genetic algorithms on multi- and many-core systems. *Future Generations Computer Systems*, 38:75 – 91. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X13002082. Acesso em: 20 out. 2019.