Algoritmo Heurístico Otimização de Colônia de Formigas e Programação Paralela para a resolução do Problema do Máximo Conjunto Independente

Marco Aurélio Deoldoto Paulino[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brazil

marco23_aurelio@hotmail.com

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Resumo. *Este trabalho tem como objetivo na elaboração de algoritmo sequencial e paralelizado para encontrar soluções viáveis do problema do Máximo Conjunto Independente utilizando algoritmos da família Colônia de Formigas introduzida por Marco Dorigo. Além da elaboração, é realizada a análise e comparação entre os algoritmos confeccionados, proporcionando dados sobre a efetividade na utilização de múltiplas threads para o algoritmo apresentado e também ao problema solucionado.*

Introdução

O Problema do Máximo Conjunto Independente, traduzido de Maximum Set Independent Problem (MISP), pode ser considerado um problema de importância para a computação devido a sua aplicabilidade a várias áreas, tais como, Reconhecimento de Padrões, Escalonamento,Biologia Molecular e *Map Labeling*.

Para a elaboração do algoritmo solucionador do MISP, foi utilizado a metaheurística de propósito geral Otimização de Colônia de Formigas, do inglês, Ant Colony Optimization (ACO), introduzido por [Dorigo et al. 1996]. O ACO se propõe a resolver aos mais variados problemas, tais como, os problemas do Caixeiro Viajante, Coloração de Grafos e Mochila Binária e Mochila Fracionária.

Como propósito do artigo, além do algortimo sequencial, foi produzido o algortimo em paralelo. Os algoritmos em paralelo vem com o objetivo de utilizar toda a potência de processadores multicores, visto que ao criar threads, divide-se o problema em partes, o que permite terminar o problema em menor tempo e utiliza todo o potencial a seu serviço. Por outro lado, é necessário haver controles e uma maior atenção para a produção de códigos corretos, principalmente que pensar em criar códigos paralelos é de maior dificuldade que códigos sequenciais.

Este artigo é organizado da seguinte forma. Na Seção 2, será descrito de forma detalhada o Problema do Máximo Conjunto Independente. Na Seção 3, será apresentado

trabalho relacionados, seja pela resolução do Problema do Máximo Conjunto Independente, a utilização da Otimização de Colônia de Formigas ou por trabalhos utilizando algoritmo paralelos. Após, será apresentado na Seção 4 de resolução do MISP e a partir da proposta, será analisado na Seção 5 os resultados obtidos. Por fim, na Seção 6 será descrito as conclusões obtidas e possíveis trabalhos futuros.

Problema

Dado um Grafo G = (V, E), em V representa o conjunto de vértices e E representa o conjunto de arestas do grafo. O Problema do Máximo Conjunto Independente tem como objetivo encontrar um subconjunto V* $\subseteq$ V, em que $\forall$i, j $\in$ V*, a aresta (i, j) $\notin$ E, além de que V* deve ser máximo.

A formulação da programação inteira para o trabalho, pode ser definido da seguinte maneira:

$$\max \sum_{i=1}^{|V|} \text{ci*xi}$$

$$x_i + x_i \leq 1, \qquad \forall \text{ (i, j)} \in \text{E}$$

$$x_i \; in \; 0, 1, \qquad \text{i = 1, 2, ..., } |V|$$

O problema de decisão para verificar se há um conjunto de vértices independentes de tamanho n é classificado como NP-Completo. Enquanto o Problema Máximo Conjunto Independente é classificado como NP-Difícil.

O MISP é um problema muito similar ao problema do clique máximo, que resumidamente, busca encontrar o maior subconjunto de vértices adjacentes, em outras palavras, a cada par de vértices dentro do subconjunto, é necessário que haja uma aresta os interligando. É possível resolver o MISP através do clique máximo, dado o grafo de entrada no MISP, basta acharmos o seu complemento e utilizá-lo como entra para o clique máximo. Também é válido a operação inversa, utilizarmos o MISP para resolvermos o clique máximo. Com isso, dado um grafo G qualquer e seu complemento o grafo X, temos que:

MISP(G) = CliqueMaximo(X) ou então,

CliqueMaximo(G) = MISP(X)

//falar do clique

Trabalhos Relacionados

A meta-heurística Ant Colony Optimization foi apresentado por [Dorigo et al. 1996] a partir de observações sobre o comportamento de formigas reais a fim de descobrir como animais de pouca visão conseguiam se locomover e atingir seus destinos. Como o resultado das observações foi modelada e desenvolvida a ACO. Ao longo do tempo foi desenvolvida extensões para a ACO, como por exemplo, Recursive Ant Colony Optimization e Elitist Ant System.

Foram encontrados diversos trabalhos com o Máximo Conjunto Independente, sendo o mais antigo do ano 1977, desenvolvido por [Tarjan and Trojanowski 1976] RE Tarjan AE Trojanowski que apresentou um algoritmo de solução. Também foi encontrado trabalhos que apresentavam soluções utilizando heurísticas, bem como o trabalho

de MAURICIO G.C. RESENDE AND CELSO C. RIBEIRO /*REFERÊNCIA*/ que utilizaram a meta-heurística GRASP e Algortimos Evolucionários por Back e Khuri /*REFERENCIA*/

Direcionando as pesquisas para encontrar trabalhos que utilizaram a meta-heurística Ant Colony Optimization para a resolução do Máximo Conjunto Independente, foi encontrado dois trabalhos que propôs esse desafio [Choi et al. 2007] e [Leguizamon et al. 2011]. Para o presenta trabalho, utilizaremos propostas enunciadas neste trabalho, bem como a função probabilidade /ver nome e parâmetros.

Proposta

Para a resolução do problema, poderíamos utilizar algoritmos determinísticos, porém ao utilizar algoritmos heurísticos provavelmente encontraremos boas soluções com menor tempo em relação aos algoritmos determinísticos e com o uso de heurísticas adequadas a queda na qualidade das soluções não será tão sentida.

O único conhecimento que temos para resolver o problema é o grafo, e assim temos acesso a sua constituição, como por exemplo, o número de vértices e arestas, densidade e quantidade de cada vértices adjacentes de cada vértice.

Para resolvermos o problema, precisamos utilizar três conjuntos ao longo do algoritmo, os conjuntos são:

S(t): Conjunto de Vértices presente na resposta em um determinado tempo t.

I(t): Conjunto de Vértices que **não** podem estar mais na solução em um determinado tempo t.

D(t): Conjunto de Vértices que **ainda** podem estar solução em um determinado tempo t.

Com nos três conjuntos, sabemos que S(t) + I(t) + D(t) = V para qualquer instante de tempo, e que a S(t) $\subseteq$ V*. //Figuras com os três conjuntos dentro de V e da figura de S(t) dentro de V*

A partir do grafo abaixo, explicaremos como funciona a heurística utilizada, que foi abstraída do trabalho /*REFERÊNCIA*/.

//Grafo

//explicar que pega o vertice com mais vert não adjacentes

Avaliação

Para a realização dos testes foi utilizado um computador com processador XXX de xx núcleos, com memória RAM xx, Memória cache e Sistema Operacional Ubuntu 16.04 64 bits.

No código sequencial ocorreu tantas miss caches, uso de memória. Enquanto no código em paralelo

//gráfico

Os resultados que foram obtidos, mostra que houve..

Conclusões e Trabalhos Futuros

Os resultados desse trabalho ....

A utilização de programação paralela se mostra bastante útil quando utilizada de forma correta e sua utilização nos mostra o ganho de perfomance na execução dos problemas e ainda é uma forma eficiente de utilizar toda potência de um computador multi-core. /verificar

Devido a inexperiência com a programação paralela, um trabalho futuro desejado seria a revisão do código e a melhoria do mesmo, devido ao aumento de conhecimento na área.

Outra hipótese envolvendo programação paralela é a utilização de placas gráficas para a execução de códigos paralelos, área crescente no cenário e que possui resultados animadores. A junção da programação paralela entre CPU e GPU já foi proposta e deve ser bastante explorada nos tempos atuais.

Referências

Choi, H., Ahn, N., and Park, S. (2007). An ant colony optimization approach for the maximum independent set problem. *Journal of the Korean Institute of Industrial Engineers*, 33(4).

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41.

Leguizamon, G., Schutz, M., and Michalewicz, Z. (2011). An ant system for the maximum independent set problem. Disponivel em: http://ls11-www.cs.tu-dortmund.de/downloads/papers/LeSz02.pdf.

Tarjan, R. E. and Trojanowski, A. E. (1976). Finding a maximum independent set. *STAN-CS-76-550*.