

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

Marco Aurélio Deoldoto Paulino¹

¹Departamento 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), é um problema na Teoria dos Grafos cujo objetivo é encontrar o maior conjunto possível de vértices que não possuem arestas entre si. O 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*. Neste artigo iremos propor uma solução para o MISP utilizando a meta-heurística Otimização de Colônia de Formigas e programação paralela. Para fins de avaliação de performance, será produzida uma versão sequencial da solução.

A meta-heurística de propósito geral Otimização de Colônia de Formigas, do inglês, Ant Colony Optimization (ACO), introduzido por [Dorigo et al. 1996], se baseou no comportamento das colônias de formigas para a elaboração da meta-heurística. As questões observadas foram, como animais praticamente cegos conseguem chegar em seus destinos, como funciona a substância química feromônio que utilizam para demarcar o caminho. Porém a representação da formiga e do feromônio não é necessariamente fiel a realidade, por exemplo, o tempo é representado discretamente e o feromônio pode ser atualizado ao final do caminho. 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.

Utilizado neste trabalho, o algoritmo em paralelo vem com o objetivo de utilizar toda a potência de processadores multicore, visto que ao dividir o fluxo de execução

em vários fluxos, permite terminar o problema em menor tempo, além de utilizar todo o potencial de sua máquina. Por outro lado, é necessário haver maior controle e atenção para a produção de códigos paralelos corretos, principalmente que a criação de códigos paralelos é de maior dificuldade que códigos sequenciais, visto que a grande maioria dos programadores estão acostumados a apenas programar sequencial e a elaboração de código paralelo possui maior complexidade em relação aos 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

De acordo a Teoria dos Grafos, o Conjunto Independente é um conjunto de vértices em um grafo, em que estes vértices não podem possuir arestas entre si. Enquanto o problema do Máximo Conjunto Independente tem como objetivo encontrar o maior conjunto independente de um grafo. Também existe outros problemas envolvendo conjuntos independentes, por exemplo, o problema de decisão para verificar se há um conjunto independente de tamanho n e o maximal conjunto independente que não são subconjuntos de outros conjuntos independentes. A formulação matemática do MISP será mostrado adiante.

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 MISP, pode ser definido da seguinte maneira:

$$\begin{aligned} \max \sum_{i=1}^{|V|} c_i x_i \\ x_i + x_j &\leq 1, & \forall (i, j) \in E \\ x_i &\in \{0, 1\}, & i = 1, 2, \dots, |V| \end{aligned}$$

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 um problema de Otimização 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 entrada 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:

$$\begin{aligned} \text{MISP}(G) &= \text{CliqueMaximo}(X), \\ &\text{ou então,} \\ \text{CliqueMaximo}(G) &= \text{MISP}(X) \end{aligned}$$

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] apresentou um algoritmo de solução eficiente para o problema. Também foi encontrado trabalhos que apresentavam soluções utilizando heurísticas, bem como o trabalho de [Resense et al.] que utilizaram a meta-heurística GRASP e Algoritmos Evolucionários por [Back and Khuri].

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 presente trabalho, utilizaremos propostas enunciadas neste trabalho, bem como a função probabilidade /ver nome e parâmetros.

Proposta

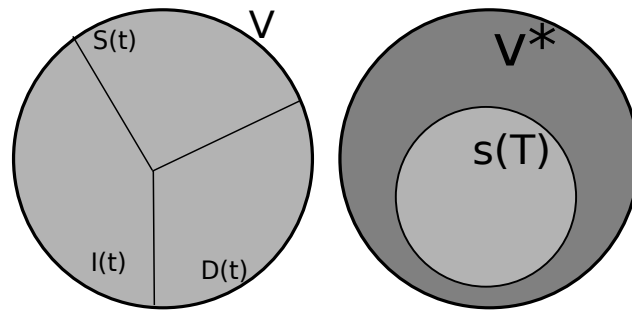
Para a resolução do problema, poderíamos utilizar algoritmos determinísticos, porém ao utilizar algoritmos heurísticos provavelmente encontraremos boas soluções com menor tempo em relação aos algoritmos determinísticos e com o uso de heurísticas adequadas a queda na qualidade das soluções não será tão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

Como vemos na Figura 1(a), os três conjuntos devem compor completamente o conjunto V , ou seja, $S(t) + I(t) + D(t) = V$ em qualquer instante de tempo e um vértice não pode estar contido em mais de um conjunto. Na Figura 1(b), temos que a $S(t) \subseteq V^*$, ou seja, tendo V^* como a solução final, a cada instante de tempo, o conjunto $S(t)$ deve ser



(a) Composição dos conjuntos utilizados na solução. (b) Composição do conjunto solução em um determinado tempo t.

Figura 1. Composições dos conjuntos.(a) e (b)

um subconjunto de V^* , assim sendo, seja qual instante de tempo for, todos os elementos de S deverão estar contidos em V^* no final.

Dado o grafo descrito na Figura 2 vamos explicar a heurística utilizada. No grafo temos um total de oito vértices. Como suposição, vamos tomar que em determinado tempo t , no conjunto solução, temos $S(t) = A$. Como o vértice A está no conjunto solução, todos os vértices adjacentes a ele não pode mais estar na solução, logo precisamos descartá-los, assim sendo, $I(t) = B, C$. Para acharmos o conjunto dos vértices em que ainda podem estar na solução, temos $D(t) = V - S(t) - I(t)$, logo $D(t) = D, E, F, G$.

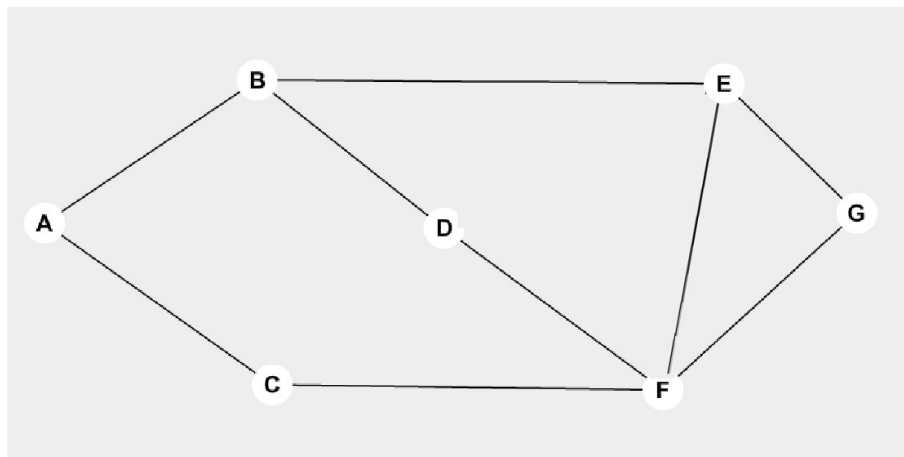


Figura 2. Grafo de Entrada para o MISP

Agora iremos encontrar o próximo vértice a ser adicionado ao conjunto solução. Para escolhermos o vértices, iremos calcular o número de vértices na qual o vértice em questão não são adjacentes, lembrando que estamos apenas trabalhando com os vértices contido no conjunto de vértices que ainda podem estar na solução. Esse número representa o valor da função heurística.

- $\tau_D(S(t)) = |\{E, G\}| = 2$
- $\tau_E(S(t)) = |\{D\}| = 1$
- $\tau_F(S(t)) = |\{\}| = 0$
- $\tau_G(S(t)) = |\{D\}| = 1$

Agora que sabemos o valor da função heurística de cada vértice, iremos escolher o vértice com o maior valor, que no exemplo, foi o vértice D. Com isso, iremos adicionar o vértice D na solução, e os vértices adjacentes a D no conjunto de vértices que não podem estar na solução e repetir o processo com os vértices restantes disponíveis para a solução. O processo irá terminar em um tempo t' em que o conjunto $D(t') =$, lembrando que em qualquer tempo t , temos que ter a seguinte igualdade: $V = S(t) + I(t) + D(t)$.

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 performance 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

- Back, T. and Khuri, S. An evolutionary heuristic for the maximum independent set problem. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=350004>.
- 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 Colormi, 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. Disponível em: <http://ls11-www.cs.tu-dortmund.de/downloads/papers/LeSz02.pdf>.

Resense, M. G., Smith, S. H., and Feo, T. A. A greedy randomized adaptive search procedure for maximum independent set. Disponivel em: <http://pubsonline.informs.org/doi/abs/10.1287/opre.42.5.860>.

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