Trabalho Prático 2 - Problema do Clique Máximo (ACO)

André Luiz Moreira Dutra - 2019006345
Computação Natural

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

cienciandre@ufmg.br

1. Introdução

O problema do clique máximo consiste em, dado um grafo, encontrar o maior dentre os subgrafos deste grafo que é completo (todos os vértices têm arestas entre si). Embora este clássico problema seja NP-difícil, podemos tentar aproximar a solução ótima usando heurísticas de busca global em espaço de estados, destacando-se entre elas técnicas baseadas em ACO (Ant Colony Optimization).

Neste contexto, este trabalho tem como objetivo propor uma heurística para o problema da clique máxima por meio de uma implementação baseada em ACOs. Serão apresentados detalhes teóricos e práticos da implementação, bem como uma análise comparativa dos efeitos de diferentes hiperparâmetros do algoritmo no resultado obtido para um determinado conjunto pré-definido de problemas.

2. Implementação do Algoritmo

2.1. Decisões teóricas para o ACO

Conforme mencionado, o ACO se trata de uma heurística de busca global em espaço de estados para resolver problemas difíceis. Desse modo, as execuções tendem a não ter eficiência equivalente à de algoritmos polinomiais. Por isso, embora a validação de alguns parâmetros tenha sido feita com múltiplas execuções, alguns foram pré-definidos com um valor padrão para diminuir o número de variantes de validação possíveis. Nesta seção as escolhas feitas para cada parâmetro não testado serão mencionadas.

- **Mapeamento da solução:** A solução foi mapeada utilizando o próprio grafo da clique para o caminhamento das formigas do ACO. O caminho de cada formiga é então definido escolhendo um nó aleatório no grafo e escolhendo o próximo nó do caminho segundo o feromônio dos nós vizinhos. Para que a

solução determine uma clique, só são oferecidos como candidatos os vizinhos que formam uma clique com os nós já presentes no caminho da formiga. A formiga caminha aumentando a sua solução candidata enquanto ainda houver nós possíveis que atingem as condições de formar uma nova clique. Este mapeamento funciona porque as cliques apresentam padrões de subestrutura: todo subgrafo de uma clique também é uma clique. Assim, ao aumentar o feromônio de um subgrafo, uma formiga tem chances de estar aumentando o feromônio de nós da solução ótima, que é o princípio do qual os ACOs partem.

- **Mapeamento dos feromônios:** os feromônios foram mapeados para os vértices do grafo. Além de oferecerem uma solução mais simples, são mais coerentes para o problema, uma vez que a seleção de arestas específicas não é muito efetiva considerando que todo clique é um subgrafo completo.
- **Feromônio máximo:** O feromônio máximo foi escolhido com quantidade 10. Nós atualizados com mais feromônio que isso se fixam no máximo. A escolha deste valor não foi calibrada como os demais parâmetros.
- **Feromônio mínimo:** De maneira semelhante, o feromônio mínimo foi escolhido com quantidade 0.5. A cada iteração, o feromônio não pode cair mais que isso por evaporação. A escolha do valor também não foi calibrada.
- **Inicialização dos feromônios:** Os feromônios de todos os nós são inicializados com quantidade máxima, partindo do pressuposto de que, inicialmente, isso favorece uma maior exploração de novas soluções (exploration).
- **Atualização dos Feromônios:** Os feromônios de cada nó foram atualizados de forma elitista, de modo que, a cada iteração, apenas os nós caminhados pela formiga representante da melhor solução são atualizados. A quantidade de feromônio adicionada é um valor entre 0 e 1 inversamente proporcional à distância da solução local da solução ótima. Note que, no caso em que a solução local é a melhor, a diferença é 0 e o feromônio adicionado é de 1. Esta implementação foi baseada no artigo deixado como referência na especificação.
- **alfa:** O valor de alfa, o fator de potencialização dos feromônios, foi definido como 1. Escolhas mais conservadoras para os parâmetros não validados foram escolhidas para que não houvesse grandes influências na validação.

2.2. Implementação

A implementação dos códigos envolvidos neste trabalho se divide em quatro arquivos: ant_clique.py, get_solution.py, get_solution.sh e plot_data.py. Toda a implementação do ACO está concentrada no ant_clique.py, e os demais arquivos são responsáveis pela coleta e plot de dados para a validação.

O arquivo get_solution.py recebe um problema e os parâmetros a serem testados e gera um csv com os resultados de 30 repetições do algoritmo para os dados definidos. O arquivo get_solutions.sh é um bash que itera entre várias combinações de parâmetros diferentes e chama get_solution.py com elas. Por fim, plot-data.py utiliza os dados gerados para gerar os gráficos presentes na seção de validação dos resultados.

Por fim, o arquivo ant_clique.py implementa a solução do problema na função max_clique_ant_solver. Além disso, o arquivo também conta com as funções start_pheromones, gen_solution e update_pheromones, que se ocupam das funções diretas do ACO de inicializar os feromônios, gerar a solução de cada formiga e atualizar os feromônios. Por fim, as funções get_node_probabilities e max_random são auxiliares às funções principais mencionadas anteriormente. O comportamento de cada função será descrito mais detalhadamente a seguir:

- **max_clique_ant_solver:** Implementa o ACO propriamente dito. Segue essencialmente o pseudocódigo padrão de um ACO: inicializa os feromônios com o feromônio máximo em start_pheromones, define uma variável para a melhor solução global e entra em um loop onde, a cada iteração, ele calcula os caminhos de cada formiga com gen_solution, obtém a melhor solução local (dentre as formigas), atualiza a melhor solução global e atualiza os feromônios com essas informações em update_pheromones. Ao fim das iterações, o melhor global é retornado. Além disso, o melhor global é coletado para cada iteração e retornado junto à solução para permitir a visualização dos dados.
- **start_pheromones:** Inicializa os feromônios dos vértices com o valor definido. Os feromônios são definidos como parâmetros dos vértices do grafo dado.
- **gen_solution:** Gera a solução de uma formiga segundo a modelagem mencionada na seção anterior. Começando em um nó aleatório, os candidatos são definidos como os vizinhos desse nó. A formiga então itera escolhendo um candidato segundo as probabilidades definidas com get_node_probability,

adicionando ele à solução e atualizando a lista de candidatos como uma interseção dela com os vizinhos do novo nó. Assim, os candidatos sempre correspondem a vizinhos de todos os nós na solução, condição necessária e suficiente para permitir uma clique.

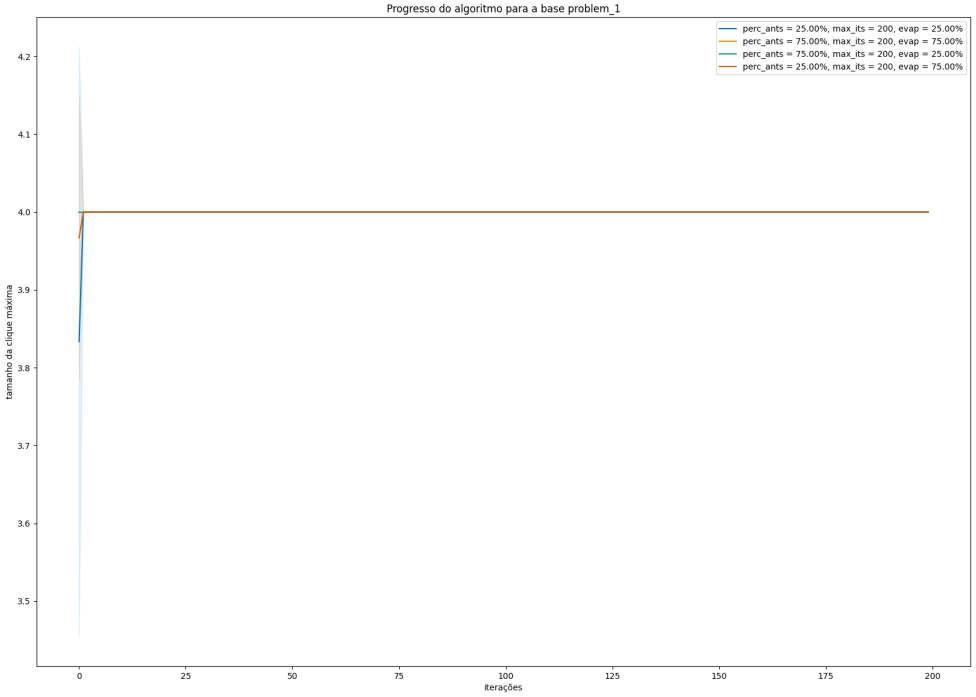
- **get_node_probabilities:** As probabilidades dos candidatos correspondem ao vetor de feromônios de todos eles normalizado para que a soma equivaler a 1. Antes da normalização, os feromônios são elevados a alfa, segundo a definição dos ACOs. Esta função realiza esses cálculos para um conjunto de candidatos dado e retorna as probabilidades.
- **max_random:** Função auxiliar que retorna o valor máximo de uma lista de elementos. Se houver mais de um máximo, escolhe um candidato aleatório e o retorna. Funciona quase de maneira idêntica ao max comum, mas o max sempre retorna o primeiro. Todos os cálculos de máximo das funções acima foram feitos usando este máximo para garantir o caráter estocástico do algoritmo.

3. Validação do Algoritmo

A validação do algoritmo foi feita testando variações de três parâmetros para três problemas selecionados: o número de formigas, o número de iterações e a taxa de evaporação do feromônio, para os problemas 1, 2 e 4 da especificação. Quanto aos parâmetros observados, é importante ressaltar que, como forma de tornar a análise coerente para cada problema, o número de formigas não foi definido fixamente, e sim como uma porcentagem do número de nós do problema. Além disso, com relação aos problemas analisados, o problema 3 da especificação não foi executado porque o link fornecido estava quebrado.

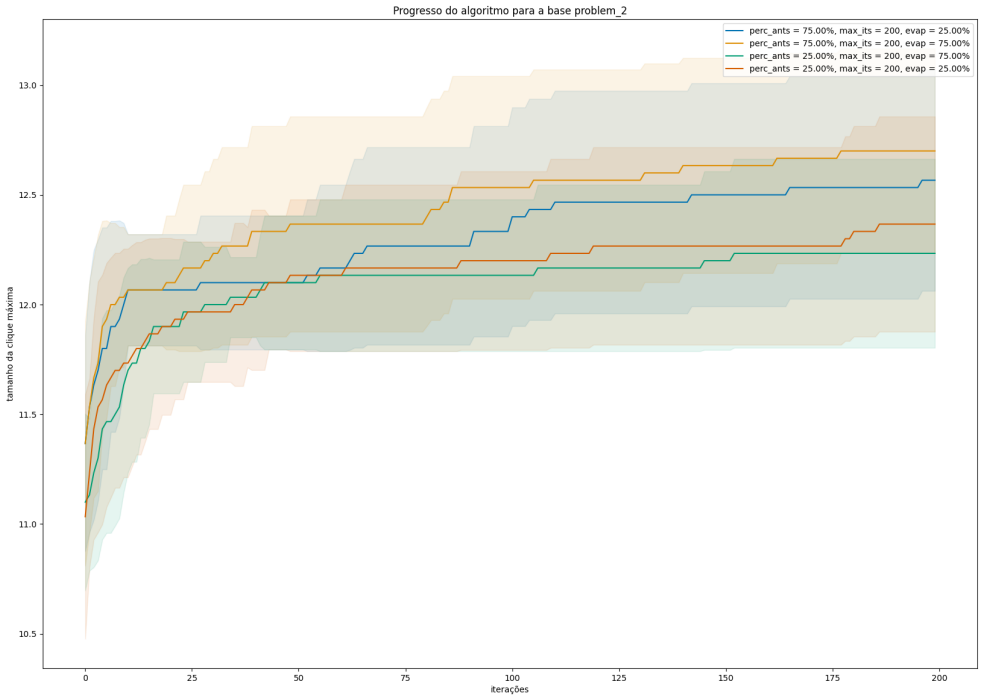
Cada um dos problemas foi testado em quatro configurações de parâmetros, variando o número de formigas entre 25% e 75% e a taxa de evaporação entre 25% e 75%. O número máximo de iterações definido foi de 200, e a melhor clique foi coletada para cada iteração do algoritmo. Por fim, cada configuração possível foi executada 30 vezes, e a evolução do algoritmo em cada iteração foi coletada e será apresentada na forma de intervalos de confiança para cada variação.

Observe os resultados para o primeiro problema, considerado fácil:



Evidentemente, a particularidade deste problema o tornou simples o suficiente para que todas as variações convergissem para o tamanho máximo antes das primeiras 10 iterações. Mesmo as diferentes execuções convergiram, porque o intervalo de confiança é quase fechado na reta máxima. Embora este problema não represente as diferenças entre as variantes observadas, ele é um exemplo um pouco esdrúxulo dos efeitos das diferenças no número de iterações definido para o algoritmo. Como o algoritmo sempre salva a melhor solução, pode parecer que quanto mais iterações melhor, já que garantidamente nunca haverá queda. No entanto, é possível que, como nesse caso, o problema seja mais simples e não requeira tantas iterações, culminando em uma situação de convergência nas primeiras iterações e estagnação desnecessária nas demais.

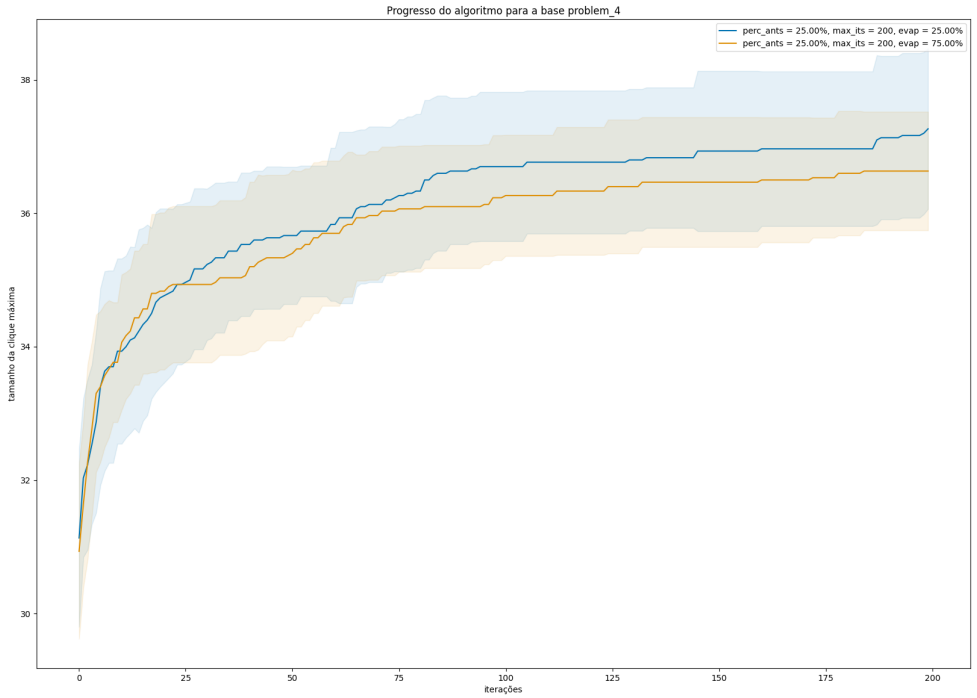
Observe agora a evolução das variantes para o problema 2:



Neste caso, a convergência não foi trivial como no anterior. No entanto, a diferença entre os resultados também não é tão significativa considerando as diferenças entre os parâmetros observados. O resultado médio da melhor e da pior curva têm diferença de aproximadamente um nó na clique, por exemplo, enquanto tanto o número de formigas quanto a taxa de evaporação têm diferença de 50% entre si. De qualquer forma, a diferença entre o desempenho de cada curva é evidente, principalmente ao fim das iterações. Observando o gráfico, vemos que variantes com 75% de formigas se desempenharam melhores que variantes com 25% de formigas. Isso é esperado, considerando que uma maior quantidade de formigas determina uma maior exploração (exploration) de subgrafos aleatórios e, assim, uma maior chance de formar o subgrafo ótimo.

No entanto, quanto à taxa de evaporação, podemos ver que entre variantes com mais formigas a melhor taxa foi a maior (75%) enquanto para as variantes com menos formigas a melhor taxa foi a menor (25%). Isso possivelmente se dá porque, com mais formigas, mais subgrafos diferentes são explorados, então uma alta taxa de evaporação é favorável por fazer com que formigas não se atenham tanto a soluções antigas em novas iterações (exploration), enquanto depois de muitas iterações provavelmente o espaço foi explorado o suficiente por todas as formigas para que, mesmo com a alta taxa de evaporação, as melhores soluções se destaquem (exploitation). Já para a situação com menos formigas, uma alta taxa de evaporação pode fazer com que boas soluções se percam do grafo rapidamente, enquanto o número menor de formigas não consegue explorar o espaço com variação suficiente para mantê-las ou encontrar soluções melhores a tempo. desfavorecendo o resultado final.

Por fim, observe os resultados do algoritmo para o problema 4:



Note que apenas duas variantes foram calculadas. Isso ocorreu porque a execução das variantes com 75% de formigas não terminou a tempo da entrega deste trabalho, já que se trata de um grafo com uma quantidade massiva de nós com muitos subgrafos grandes possíveis. No entanto, podemos observar o mesmo padrão do caso anterior, no qual, para menos formigas, uma taxa de evaporação maior é favorável. No entanto, ao contrário dos outros dois casos, como será apresentado na próxima imagem, este foi o único que não foi capaz de encontrar a solução ótima, encontrando uma clique de 40 nós, diferindo da máxima de 44. Muito provavelmente, mesmo as demais variantes não alcançariam o máximo, devido à curta diferença geral apresentada pelas médias, de cerca de 1 nó. Porém, talvez fosse possível que alguma das execuções atingisse o máximo devido ao tamanho do intervalo de confiança de cada curva. De qualquer forma, é evidente que mais iterações seriam necessárias nesse caso, tornando-o oposto ao primeiro caso, no qual a quantidade de iterações foi excessiva.

Os melhores resultados entre todas as execuções de cada problema foram os seguintes:

```
problem_1:  4
problem_2:  13
problem_4:  40
```

dos quais, conforme mencionado, apenas o último não corresponde ao ótimo.

4. Conclusão

Este trabalho permitiu a consolidação do aprendizado sobre algoritmos baseados em colônias de formigas, por meio da implementação manual do algoritmo para implementação do problema do clique. A implementação permitiu a compreensão do algoritmo em diversos aspectos, desde a modelagem das soluções das formigas à interpretação da influência dos feromônios e dos parâmetros envolvidos no algoritmo. A análise dos resultados também contribuiu para o conhecimento crítico quanto aos passos necessários para calibrar os hiperparâmetros de algoritmos estocásticos, e nos deu indícios quanto à forma como os parâmetros interagem uns com os outros.

Bibliografia

PAPPA, Gisele. *Slides da disciplina Computação Natural. Departamento de Ciência da Computação. Universidade Federal de Minas Gerais. 2023.*

A study of ACO capabilities for solving the Maximum Clique Problem, https://liris.cnrs.fr/Documents/Liris-1847.