PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
BACHARELADO EM ENGENHARIA DE SOFTWARE
DISCIPLINA DE ALGORITMOS AVANÇADOS
PROF. DIEGO VRAGUE NOBLE

Otimização da Colônia de Formigas e o Problema do Caixeiro Viajante

Trabalho 2 da Disciplina

Felipe G. Guedes
felipe.guedes@acad.pucrs.br

Gabriel F. Kurtz
gabriel.kurtz@acad.pucrs.br

Porto Alegre, 26 de Junho de 2018

Abstract

This paper is a study on solving NP-hard algorithms with the use of heuristics and probabilistic techniques. It is focused in the Ant Colony Optimization (ACO) algorithm applied to the Travelling Salesman Problem (TSP).

It was developed as the second evaluated task for the Advanced Algorithms class of the Software Engineering course at PUCRS.

1. Introdução

Neste trabalho, iremos verificar algumas propriedades dos algoritmos NP-difíceis e como podemos aplicar heurísticas para encontrar soluções que, embora não sejam ideais para todas as instâncias, podem ser mais eficientes do que as soluções tradicionais.

Será feita uma breve apresentação teórica dos assunto envolvidos e, posteriormente, demonstraremos um exemplo prático do algoritmo de Otimização da Colônia de Formigas (ACO, em inglês) para encontrar boas soluções para o problema do Caixeiro Viajante (TSP).

2. Problemas NP-difíceis

A classe de problemas NP(Non-deterministic Polynomial) é composta pelos problemas para os quais, atualmente, não se conhece um algoritmo que encontre a solução ótima para todas as instâncias em tempo polinomial. Caso haja um algoritmo que encontre esta solução para todas as instâncias de um problema em tempo polinomial, ele é classificado como P.

Entre os problemas NP, existe um conjunto de problemas que podem ser resolvidos com a mesma dificuldade dos problemas mais difíceis desta classe. Estes problemas são

classificados como NP-completo. Ou seja, caso houvesse a possibilidade de resolver um problema NP-completo em tempo polinomial, poder-se-ia utilizar o mesmo algoritmo para resolver todos eles em tempo polinomial.

O grupo NP-difícil é composto pelos problemas que são pelo menos tão difíceis quanto os NP-completos. Ou seja, ele contém todos os problemas NP-completos e ainda outros problemas que não seriam garantidamente resolvidos por um eventual algoritmo que solucionasse NPC.

O fato de não ser possível encontrar soluções ótimas certamente não impede de trabalhar estes problemas, pois muitos deles são de grande importância hoje. Abre-se mão, portanto, de alguma das premissas que se espera, para aceitar soluções que embora não sejam ótimas em algum aspecto (resposta ótima, tempo de execução), sejam satisfatórias e relevantes.

3. Heurísticas

Heurísticas são técnicas utilizadas para solucionar problemas através da aproximação. Elas não realizam necessariamente um cálculo determinístico, de modo não encontram a solução ótima para o problema, mas tentam utilizar premissas corretas para obter resultados cada vez mais próximos do ideal conforme o algoritmo é executado.

Através das heurísticas, podemos trocar a capacidade de obter respostas sempre ótimas pela capacidade de encontrar respostas com muito menos processamento e, portanto, mais velocidade.

Como verificamos que os problemas NP não possuem algoritmo que os resolva em tempo polinomial, as heurísticas são particularmente importantes neste caso. Há situações em que encontrar uma boa resposta que não seja ótima, porém executando em um tempo viável, torna-se uma alternativa muito melhor do que utilizar algoritmos não-polinomiais como por exemplo de força-bruta, cujo tempo de execução poderia tornar-se inviável para problemas maiores.

Além disso, podem ser utilizadas para encontrar aproximações em problemas para os quais não se conhece nenhum algoritmo que encontre a resposta ótima.

4. Algoritmo de Otimização da Colônia de Formigas (ACO)

O algoritmo de Otimização da Colônia de Formigas foi apresentado pelo italiano Marco Dorigo em sua tese de PhD, no ano de 1992. Trata-se de uma heurística baseada na movimentação de colônias de formigas para encontrar bons caminhos de locomoção.

Ao caminhar, as formigas deixam feromônios no caminho que são detectados pelas formigas que irão utilizar as mesmas rotas posteriormente em busca de recursos. Assim, a probabilidade de uma formiga seguir um caminho onde houver mais feromônios é maior, embora ela possa seguir caminhos diferentes.

Ocorre que os feromônios evaporam ao longo do tempo. Desta forma, uma formiga que demorou mais para chegar até o destino e voltar terá os seus feromônios evaporados relativamente mais rápido do que uma formiga que cumpriu o mesmo trajeto de forma mais eficiente pois, até fazer a rota completa, os feromônios dos setores ineficientes já terá passado mais tempo evaporando.

Isto faz com que, cada vez que uma formiga inicia uma rota seguindo os feromônios mais fortes, é provável que ela siga um dos caminhos mais eficientes mais localizados. Simultaneamente, quando uma formiga desvia da rota, existe a possibilidade de que ela encontre um caminho ainda melhor. Neste caso, seus feromônios depositados terão relativamente mais peso que os das outras formigas, pois como ela foi mais veloz, os feromônios da nova rota terão menos tempo para evaporar (ou mais tempo caso a rota seja pior).

O ACO simula exatamente isto. Em um grafo, são depositados feromônios artificiais que “evaporam” e são instanciados pequenos objetos “inteligentes” para simular as formigas. Através da repetição do algoritmo, espera-se que os objetos formigas encontrem bons resultados para determinados problemas em um tempo aceitável, sem a necessidade de cálculos determinísticos que busquem a resposta ótima.

5. Problema do Caixeiro Viajante (TSP)

O Problema do Caixeiro Viajante é um problema da classe NP-difícil. Ele propõe solucionar a seguinte situação: dada uma lista de cidades e a distância entre elas, e uma cidade inicial, qual o menor caminho possível para passar por todas as cidades e retornar à cidade inicial?

Não se sabe ao certo qual a origem deste problema. Há indícios dele em livros do século XIX como, por exemplo, um livro para caixeiros viajantes de 1832 que inclui o problema e exemplos com cidades da Alemanha e Suíça.

Foi extensivamente estudado ao longo do século XX e até hoje, sendo um problema de grande importância na computação. Mesmo em sua forma original, o problema pode ser aplicado a diversos problemas da vida real como, por exemplo, logística e fabricação de chips. Com algumas transformações, pode ser utilizado abstrair problemas ainda mais distintos. Foram desenvolvidas inúmeras heurísticas e algoritmos de aproximação que são capazes de encontrar respostas com até 3% de precisão em tempo hábil.

O problema pode ser representado através de um grafo onde cada nodo é uma cidade e cada aresta com peso representa o caminho entre as cidades, e o peso das arestas a distância entre aquele par de cidades. É a partir de representações como estas que aplicamos o algoritmo da Otimização da Colônia de Formigas para verificar seu funcionamento no TSP.

6. Modelando o Problema

Para aplicar o algoritmo da Colônia de Formigas ao problema do Caixeiro Viajante, optamos por desenvolver um script em Python. O script possui quatro arquivos de código, sendo:

- main.py: Contém o controle do fluxo do código e as instâncias das variáveis do problema como, por exemplo, o grafo.
- ant_colony_optimization: Contém o algoritmo de Colônia de formigas propriamente dito, onde ocorre o cálculo de aproximação que irá obter a resposta desejada.
- graph.py: A classe utilizada para gerar e manusear os grafos.

- ant.py: A classe que representa uma formiga “inteligente”.
- plot.py: Elabora uma representação visual do caminho percorrido no melhor resultado.

Vejamos com mais detalhes o funcionamento da solução. No momento em que rodamos o main, ocorre a leitura do arquivo de input com os dados do problema. Foram gerados especificamente quatro arquivos que estão inclusos junto com o código, além de uma solução gerada manualmente que está comentada no código e pode ser usada para verificar o funcionamento adequado do algoritmo. No main, ocorre a instanciação do grafo e do ant_colony_optimization.py(ACO).

O ACO, por sua vez, define algumas variáveis do problema, instancia as formigas e manipula as estruturas de dados para poder acessar com mais facilidade os dados do grafo. Inicialmente, joga as formigas em locais diferentes. Inicializa posteriormente o feromônio. A partir disso, começa a rodar o algoritmo que simula as formigas andando pelas arestas atrás dos caminhos mais eficientes.

Ao encontrar o melhor resultado, passa ele para o plot.py que se encarrega de criar a visualização do resultado e exibí-lo na tela de forma gráfica.

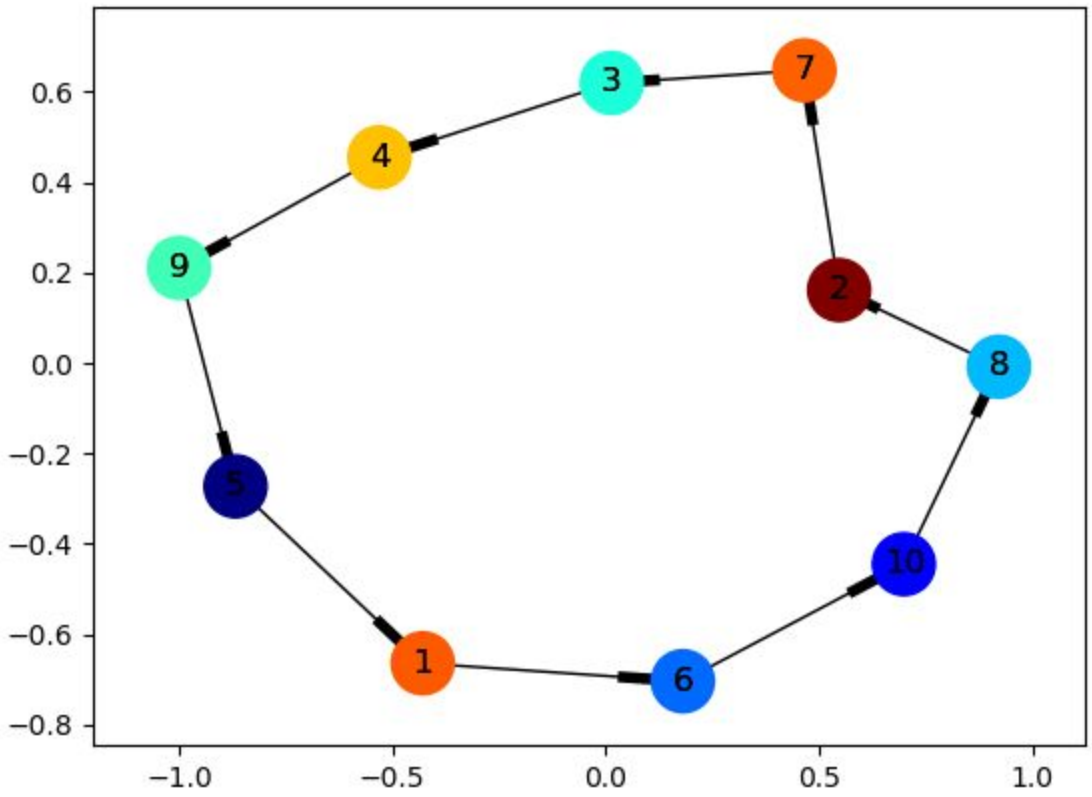
7. Resultados Encontrados

O programa após computar o problema exibe no terminal uma mensagem que informa ao usuário a solução final na forma de texto. Outra informação importante que é exibida é o custo do caminho. No caso mostrado abaixo, o programa foi executado com o arquivo graph10.csv que contem dez nodos. Para esse problema, o algoritmo atingiu a seguinte solução:

“ Solução final: 10 >> 8 >> 2 >> 7 >> 3 >> 4 >> 9 >> 5 >> 1 >> 6 >> 10 | cost: 191 ”

Há também um *output* visual para o usuário que é representado através de um grafo como mostrado na imagem 1 a seguir. Este é o output gerado pelo mesmo problema que o anterior, apenas exibido de forma visual para a melhor compreensão do usuário.

Imagem 1: Plot gerado pelo programa



Fonte: Própria.

8. Conclusão

Consideramos que o programa que desenvolvemos solucionou de forma bastante razoável um problema de alta complexidade. Nos caso em que foi possível verificar manualmente, com amostragens pequenas de cidades, o algoritmo foi capaz de localizar a resposta ótima para as instâncias realizadas.

Lembramos, no entanto, que o algoritmo que implementamos tem a tendencia de encontrar mínimas locais em vez de mínimas globais. Por este e outros motivos, não há como garantir em nenhuma hipótese que o resultado obtido seja efetivamente o resultado ótimo.

De qualquer forma, esta já era uma característica esperada da solução. Lembramos também que, como estudantes de graduação, a nossa versão do algoritmo da Colônia de Formigas é relativamente básica. Existem diversas vertentes muito mais avançadas que utilizam algoritmos como este nos aspectos mais modernos da tecnologia, havendo inclusive empresas dedicadas exclusivamente ao algoritmo da Colônia de Formigas.

Acreditamos ter realizado um bom trabalho e obtido o resultado esperado.