TP1 - Inteligência Coletiva para o problema de P-medianas com restrições de capacidade

Ramon Gonçalves Gonze

07 de novembro de 2017

Resumo

O objetivo deste trabalho é apresentar um algoritmo de Otimização de Colônia de Formigas (ACO) para resolver o problema de p-medianas com restrições de capacidade. Os assuntos sobre inteligência coletiva abordados em sala de aula foram utilizados no desenvolvimento deste trabalho, assim como as sugestões de referências. Os resultados dos experimentos são apresentados ao final, juntamente com a análise sobre a variação dos parâmetros e seus respectivos impactos.

1 Introdução

O problema de P-medianas consiste em, dada uma rede composta de N pontos (podendo ser representada por um grafo não-direcionado), encontrar P centros (medianas) de forma a minimizar a soma das distâncias de cada ponto ao centro mais próximo. Além de localizar os centros, há uma restrição: cada ponto possui uma demanda associada, que restringe a capacidade de atendimento dos centros.

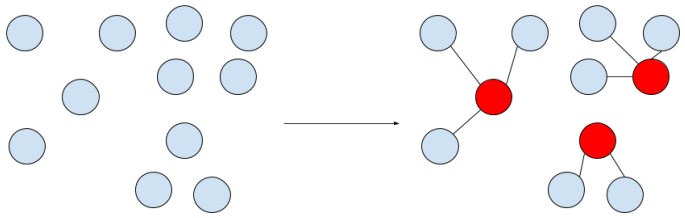

Figura 1: Do lado esquerdo, os pontos no espaço são a entrada do algoritmo. À direita, a melhor solução para $P = 3$, sendo os vértices de cor vermelha, os centros escolhidos.

Um algoritmo de colônia de formigas se baseia no comportamento de formigas na busca de alimento para o formigueiro. Através de uma substância química deixada por onde andam, denominada *feromônio*, as formigas tendem a minimizar o tamanho da rota do formigueiro até a fonte de alimento. A minimização ocorre pois a quantidade de feromônio depositada em caminhos mais curtos, é maior, e a tendência de cada formiga é seguir por um local que contenha mais feromônio. Juntamente com o ACO, foram utilizadas heurísticas para otimizar o funcionamento do algoritmo.

A seção seguinte demonstrará como foi feita a modelagem do algoritmo A solução do problema é descrita na **Seção 2.2**, e em sequência são apresentadas as heurísticas utilizadas para otimizar o algoritmo. A descrição de cada arquivo que executa o algoritmo, juntamente com informações sobre as bases de dados utilizadas para teste, estão contidas na **Seção 3**. A **Seção 4** contém os experimentos e a análise dos resultados. Por fim, a última seção conclui todo o trabalho realizado, exibindo as dificuldades encontadas e os resultados gerais sobre a utilização de um ACO para a resolução do problema de P-medianas com restrição de capacidade.

2 Modelagem

Nesta seção serão descritas as estruturas de dados, técnicas e os métodos utilizados para modelar o problema, assim como suas funcionalidades para o funcionamento do algoritmo.

2.1 Representação

Os pontos da rede foram representados por um grafo não-direcionado $G(V, E)$, sendo V o conjunto de vértices e E o conjunto de arestas. A escolha da implementação do grafo por uma matriz de adjacências se deu pelo fato de o algoritmo fazer acesso constante às distâncias entre os vértices. A distância entre dois vértices quaisquer i e j (uma aresta do grafo) é determinada pela distância euclidiana

$$d_{ij} = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

sendo as coordenadas do vértice $i = (x_0, y_0)$ e as do vértice $j = (x_1, y_1)$.

2.2 Solução do problema

As formigas foram utilizadas para selecionar as medianas, a cada iteração do algoritmo. A partir de P medianas selecionadas, é construída uma solução utilizando uma heurística, descrita na **Seção 2.2.2**. Portanto, cada vértice possuirá uma quantidade de feromônio atrelada a ele, e esta será atualizada a cada iteração de acordo com a qualidade da solução gerada pelas medianas selecionadas naquela iteração.

2.2.1 Seleção de medianas

A probabilidade de um ponto ser escolhido como mediana é derivada de dois fatores: feromônio e *densidade*. Essa última foi adicionada para que a escolha de um vértice fosse mais direcionada para aqueles que estão em um espaço com densidade de vértices (possui uma grande quantidade de vizinhos próximos a ele) maior. A densidade de um vértice i é calculada conforme o Algoritmo 1 a seguir:

Algorithm 1

```
1: procedure CALCULADENSIDADE(GRAFO)
2:     for i := 1 to n do
3:         vertices_ord = ordenaVertices(i)
4:         [num_vizinhos, soma_vizinhos] = alocaPontos(i, vertices_ord)
5:         i.densidade = num_vizinhos / soma_vizinhos
```

onde n é o número de vértices, e $i.densidade$ representa a densidade do vértice i. A função ordenaVertices(i) ordena todos os vértices baseado nas suas distâncias ao vértice i. A função alocaPontos($i, vertices\_ord$) associa cada vértice de $vertices\_ord$ a i, até que a capacidade deste seja alcançada, retornando dois valores: $num\_vizinhos$, que é o número de vértices vizinhos alocados para aquele ponto; e $soma\_vizinhos$, que é a soma da distânca entre cada vértice alocado e o vértice i.

Seja p_i a probabilidade de um ponto i ser escolhido como mediana:

$$p_i = \frac{(\tau_i)^\alpha . (d_i)^\beta}{\sum_{j \in V} (\tau_j)^\alpha . (d_j)^\beta}$$

Os parâmetros α e β são, respectivamente, utilizados para regular a influência do feromônio e da densidade na probabilidade de escolha de um vértice como mediana.

2.2.2 Heurística para construção da solução

Após as medianas serem escolhidas, uma heurística é utilizada para alocar os vértices restantes a elas. O método foi proposto por Osman e Christofides [1], e é descrito no Algoritmo 2. Seja $r = |V| - P$ o número de pontos que não foram escolhidos como medianas:

Algorithm 2

```
1: procedure CONSTROISOLUCAO(MEDIANAS[])
2:     vertices_ord = ordenaVerticesRestantes(medianas[])
3:     for i := 1 to r do
4:         medianas_ord = ordenaMedianas(vertices_ord[i])
5:         for j := 1 to P do
6:             if capacidade(medianas_ord[j]) - demanda(vertices_ord[i]) ≥ 0 then
7:                 m_ij = 1
       return m
```

A função ordenaVerticesRestantes(medianas[]) retorna uma lista de todos os vértices que não são medianas em ordem crescente da distância de cada vértice para a mediana mais próxima. A função ordenaMedianas(i) retorna uma lista com todas as medianas ordenadas de forma crescente pela distância até o vértice i. A partir da lista $medianas\_ord$, o vértice i é alocado para a primeira mediana disponível, ou seja, que ainda não tenha esgotado a sua capacidade.

Ao final da execução do algoritmo, a matriz solução (descrita na **Seção 3**) é gerada.

2.2.3 Atualização do feromônio

A cada iteração do algoritmo, ou seja, a cada vez que as formigas são utilizadas para selecionar os centros, uma atualização do feromônio deve ser realizada. Seja S o conjunto de vértices selecionados como medianas para gerar a solução s e τ_i a quantidade de feromônio do vértice i. A atualização do feromônio de cada vértice é dada por:

$$\tau_i = \rho.\tau_i + \Delta\tau_i$$

onde $\rho \in (0, 1]$ é a taxa de evaporação do feromônio, e $\Delta\tau_i$ é o incremento (ou decremento) do feromônio, de acordo com a qualidade da solução. $\Delta\tau_i$ é definido por:

$$\Delta\tau_i = \begin{cases} \frac{1}{q(s)}, & i \in S \\ 0, & \text{caso contrário} \end{cases}$$

onde $q(s)$ é a qualidade da solução s. Quando um conjunto de vértices é escolhido como medianas , mas a soma das capacidades de todos esses vértices não comporta a demanda de todos os vértices, $q(s)$ retorna uma constante negativa, que irá decrementar a quantidade de feromônio dos vértices que estão no conjunto S.

Uma otimização do ACO proposta em Stützle e Hoose [2] e Stützle e Dorigo [3], é de estipular um valor máximo e um mínimo para os feromônios. O valor máximo de um feromônio, τ_{max}, é definidos por:

$$\tau_{max} = \frac{1}{1-\rho}.\frac{1}{F_{ot}}$$

e o valor mínimo, τ_{min}, é definido por:

$$\tau_{min} = \frac{\tau_{max}}{2n}$$

onde ρ é a taxa de evaporação do feromônio, n o número de pontos e F_{ot} é o valor da solução ótima. Contudo, na maioria dos casos, o valor do ótimo não é conhecido. Quando o ótimo não é conhecido, pode-se substituir F_{ot} pelo valor da melhor solução encontrada até o momento.

3 Base de Dados e Execução

Foram utilizadas três bases de dados para a realização dos experimentos:

Base	Pontos	P	Ótimo
SJC1	100	10	17246,53
SJC2	200	15	33225,88
SJC3b	300	30	40635,80

Tabela 1: Bases de dados utilizadas. *Pontos* indica a quantidade de pontos no espaço; P o número de centros que devem ser escolhidos; e *Ótimo* a qualidade da melhor solução possível (somatório das distâncias de todos os pontos até a mediana mais próxima).

O trabalho foi implementado em Python na versão 3.6.1. A função principal do arquivo está no arquivo **main.py** (**Seção 3.1**), que recebe como parâmetro o nome do arquivo que contém a base de dados. O programa deve ser executado segundo o escopo:

python3 <diretório/base_de_dados> <solução_ótima>

Para executar o programa para a base **SJC1** por exemplo, deve ser escrito:

python3 SJC1.dat 17246.53

Caso a solução ótima não seja conhecida, não é necessário escrever o segundo argumento.
As bases de dados devem obedecer o seguinte formato:

1^{a} linha: n, p
i-ésima linha: x, y, c, d

sendo n o número de pontos no espaço; p a número de centros a serem selecionados; (x, y) as coordenadas cartesianas do i-ésimo ponto; c a capacidade do i-ésimo ponto; e d a demanda do i-ésimo ponto.

Os parâmetros do algoritmo como número de iterações, feromônio inicial, taxa de evaporação, entre outros, se encontram no arquivo **aco.py**.

A saída do algoritmo é feita pela saída padrão (**stdout**); A primeira linha contém o valor da qualidade da melhor solução encontrada, e o restante das linhas contém uma matriz binária m no estilo de arquivos *.csv*, que representa quais vértices foram escolhidos como medianas e qual vértice está alocado a qual mediana nesta solução, conforme abaixo:

$$m_{ij} = \begin{cases} 1, & \text{se o vértice } i \text{ é alocado para a mediana } j \\ 0, & \text{caso contrário} \end{cases}$$

Os centros escolhidos são alocados para si mesmos (sua demanda ocupa parte da sua capacidade), logo quando $m_{ij} = 1$ e $i = j$, significa que o vértice é uma mediana.

3.1 Arquivos

Os arquivos que compõe este trabalho, seguidos de suas funções, são:

- **graph.py**: Possui funções que constroem um grafo por uma matriz de adjacências, e também as funções que calculam a densidade de um ponto (**Algoritmo 1**).
- **aco.py**: Possui os parâmetros do ACO, além das funções responsáveis por realizar a escolha das medianas, a construção de uma solução, a atualização do feromônio, entre outras.
- **main.py**: O arquivo que contém a função principal do algoritmo, que é responsável por fazer chamadas de todos os outros métodos implementados. O Algoritmo 3 representa o funcionamento:

Algorithm 3

```
1:  procedure ACO
2:      dados = leituraDeDados()
3:      grafo = constroiGrafo(dados)
4:      calculaDensidade(grafo)
5:      melhor_solucao = ∞
6:      for i := 1 to MAX_IT do
7:          p_medianas[] = escolheMedianas()
8:          s = constroiSolucao(p_medianas[])
9:          if s < melhor_solucao then
10:             melhor_solucao = s
11:         atualizaFeromonio()
12:     return melhorSolucao
```

onde MAX_IT é o número de iterações a serem executadas.

4 Experimentos

Os experimentos abaixo foram realizados ao longo de 10,000 iterações, e aqueles utilizados para comparação da qualidade da melhor solução, foram executados 30 vezes e extraída a média.

4.1 Experimento 1 - Feromônio Inicial

O primeiro experimento feito, utilizando a base **SJC2**, teve o objetivo de verificar qual o impacto da variação do feromônio inicial nos pontos, e como a evolução se comportaria.

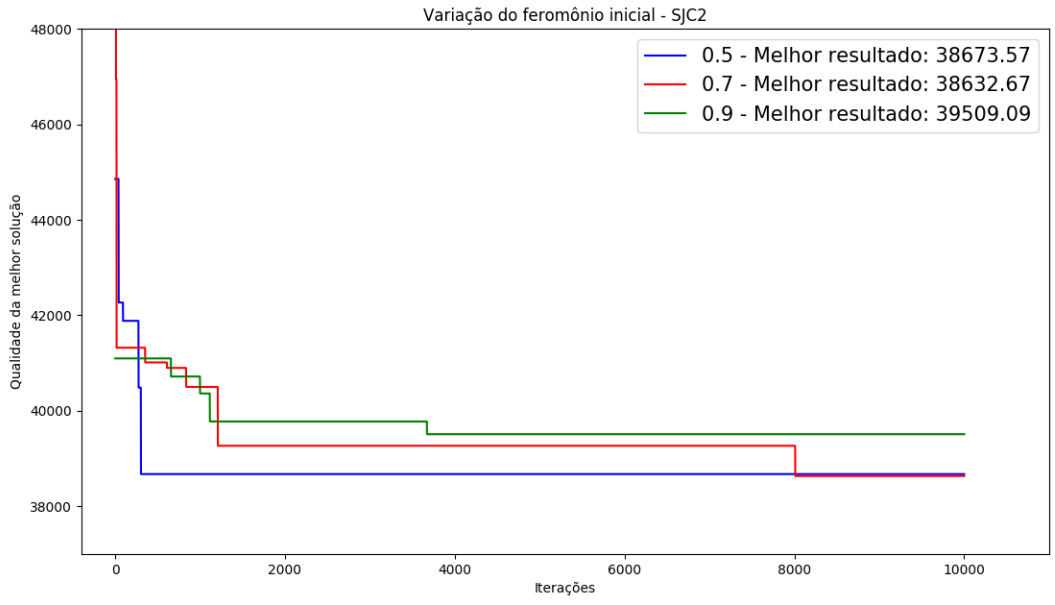

Figura 2: Gráfico que demonstra a variação do feromônio inicial, com os valores de 0.5, 0.7 e 0.9.

Podemos observar a partir do gráfico da Figura 2 que quanto menor o valor do feromônio inicial, a convergência do algoritmo se torna mais rápida, com o valor da melhor solução se estagnando logo a partir da 200ª iteração. Com feromônios iniciais de valor superior, a convergência se torna menos prematura, porém, o valor da melhor solução final é pior, conforme pode ser comparado observando as linhas verde e azul.

4.2 Experimento 2 - Variação dos parâmetros α e β

O valor do feromônio inicial foi fixado em 0.5, pelo resultado obtido no Experimento 1. Aqui são variados os valores de α e β da equação que calcula a probabilidade de um ponto ser escolhido como mediana (**Seção2.2.1**). A base de dados utilizada foi a **SJC3b**.

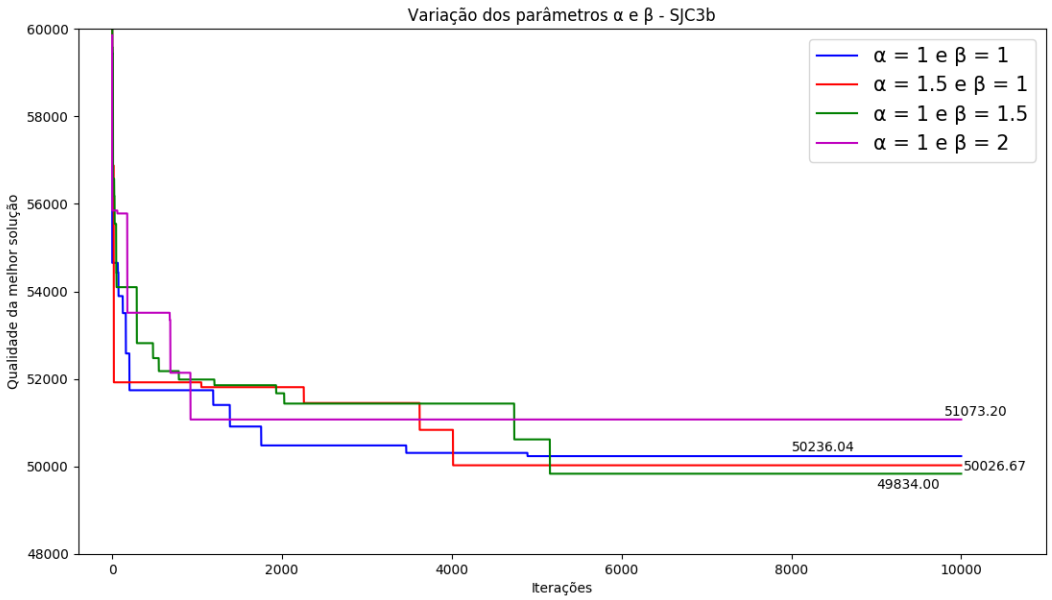

Figura 3: Gráfico que demonstra a variação dos parâmetros que determinam a influência, respectivamente, do feromônio e da densidade de um ponto na probabilidade dele ser selecionado como mediana.

A partir do gráfico da Figura 3, podemos concluir que quando $\beta > \alpha$, melhores resultados são obtidos, como o resultado pela combinação $\alpha = 1$ e $\beta = 1.5$. Porém, a diferença $|\beta - \alpha| \leq 0.5$, caso contrário, a qualidade da melhor solução cai, como observado na linha roxa ($\alpha = 1$ e $\beta = 2$).

4.3 Experimento 3 - Taxa de evaporação (ρ)

Para este experimento, foram mantidos fixos os seguintes valores: feromônio inicial $= 0.5$, $\alpha = 1$ e $\beta = 1.5$. A base de dados utilizada foi a **SJC1**. Alterando a taxa de evaporação, a Figura 4 nos mostra o resultado:

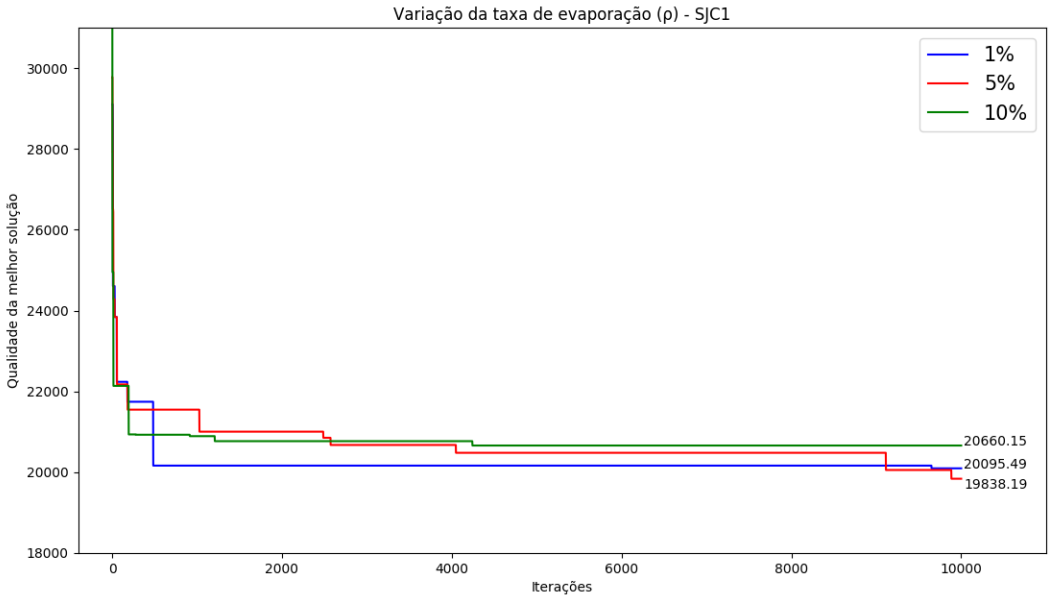

Figura 4: Gráfico que apresenta o comportamento do algoritmo de acordo com diferentes taxas de evaporação do feromônio.

Podemos observar no gráfico da Figura 4 que quando a taxa de evaporação é muito baixa (1%), a convergência é prematura. Por outro lado, com uma taxa de evaporação muito alta (10%), a qualidade da melhor solução final diminui. Portanto, podemos concluir que $\rho = 5\%$ é a taxa mais apropriada.

4.4 Experimento 4 - Densidade

Este experimento tem o objetivo de avaliar a influência da densidade dos pontos na evolução do algoritmo. A Tabela 1 apresenta o resultado das execuções:

Base	Densidade	Melhor resultado	Média	Desvio padrão
SJC1	Com	19011.46	19784.15	351.64
	Sem	19069.75	19979.33	365.33
SJC2	Com	37348.29	38690.29	533.82
	Sem	37612.66	38801.64	649.32
SJC3b	Com	48727.43	50244.46	521.36
	Sem	48450.48	49808.29	573.60

Tabela 2: Execução do algoritmo com e sem a utilização de densidade de um ponto. O Desvio padrão e a Média são em relação às 30 execuções realizadas.

Podemos observar através dos dados da Tabela 2 que não houve uma variância muito grande nas soluções encontradas para todas as bases, utilizando-se ou não a densidade dos pontos.

5 Conclusão

A partir dos experimentos realizados e as análises sobre as mudanças de parâmetros, pode-se concluir que o algoritmo obteve uma boa aproximação da solução ótima. Era esperado que a densidade dos pontos alcançasse uma melhora significativa, porém os experimentos provaram que não há muita diferença em utilizar essa heurística ou não.

Por outro lado, a heurística descrita no Algoritmo 2 se mostrou bem eficiente para a solução do problema em geral, retornando resultados satisfatórios, considerando o âmbito da disciplina.

Referências

[1] OSMAN, I. H.; CHRISTOFIDES, N., Capacitated CLustering Problems by Hybrid Simulated Annealing and Tabu Search. Pergamon Press, England, 1994, Int. Trans. Opl.Res. v. 1, n. 3, pp. 317-336, 1994.

[2] STÜTZLE, T.; HOOS, H.H. The MAX-MIN ant system and local search for the traveling salesman problem. In T. Bäck, Z. Michalewicz, and X. Yao, editors, Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97), IEEE Press, Piscataway, NJ, USA, pp. 309-314, 1997.

[3] STÜTZLE, T.; DORIGO M. ACO algorithms for the Quadratic Assignment Problem. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pp. 33-50. McGraw-Hill, 1999.

[4] FRANÇA, O. F.; VON ZUBEN J.F; CASTRO N. L. Max Min Ant System and Capacitated p-Medians: Extensions and Improved Solutions. Campinas/SP and Santos/SP, Brazil. Informatica, vol. 29, n. 2, pp. 163–171, 2005.