Relatório - Parte 3 - Teoria dos Grafos (COS242)

Thiago Barroso Perrotta e Guilherme Sales

11 de Dezembro de 2013

Professores: Daniel Ratton Figueiredo e Ricardo Marroquim

1 Do objetivo

O objetivo desta terceira parte do trabalho foi de utilizar a biblioteca desenvolvida anteriormente para aplicá-la ao problema do **clique de peso máximo**, que consiste em encontrar o clique de maior peso, num grafo com pesos nas arestas. Sabe-se que esse problema é NP-difícil, de modo que soluções polinomiais são ainda desconhecidas. Atualmente, podemos apenas encontrar soluções exatas através de força bruta, em tempo exponencial.

No entanto, a proposta desse trabalho é utilizar uma ou mais heurísticas para tentar resolver o problema de forma **aproximada**. A ideia seria utilizar algum algoritmo guloso, ou de programação dinâmica, ou algo similar, para encontrar uma clique com o peso mais próximo do ótimo quanto possível, mas a um custo computacional (tempo e memória) acessível.

2 Da abordagem do problema

Em primeiro lugar, tivemos que reprocessar os grafos de entrada, pois eles continham arestas paralelas ("multigrafo"). Naturalmente, como queremos trabalhar apenas com grafos simples, eliminamos as arestas múltiplas, selecionando apenas uma: as que tinham maior peso. Adicionamos um método para realizar esse pré-processamento (filtragem) na classe *InputHandler*.

Na ausência de soluções exatas eficientes para o problema, buscamos soluções que aproximem bem a solução ótima, e possa ser calculada dispondo de uma quantidade de recursos de tempo e memória razoáveis.

Consultando algumas referências na literatura, encontramos uma solução simples para um problema similar: o de achar um clique com o maior número de vértices no grafo. Notamos que no caso em que todas as arestas têm o mesmo peso, achar o clique de peso máximo se resume a achar o maior clique em número de vértices. Além disso, caso os pesos das arestas não variem muito em torno da média, podemos imaginar que o clique de maior peso seja também o clique com maior número de vértices.

Para resolver então o problema do maior clique, nos baseamos no algoritmo descrito por Carraghan e Pardalos, no artigo *'An exact algorithm for the maximum clique problem'* (Operations Research Letters, 1990).

Esse algoritmo se baseia numa enumeração parcial dos cliques do grafo, que a cada momento registra o maior clique já obtido, e evita entrar em cálculos que sabidamente não levarão a uma resposta melhor (*pruning*). Essa técnica em geral permite encontrar a solução exata em bem menos tempo do que a enumeração total.

2.1 Cotas estimadas

Decidimos, antes de tudo, obter uma cota superior para o peso máximo de um clique do grafo. Trivialmente, a maior cota superior pode ser obtida através da soma de todas as arestas (pesos) do grafo. No entanto, essa é a pior cota que poderíamos tomar; ela só é igual ao peso máximo quando o grafo é completo.

Grafo	Cota Superior #1		Cota Superior #2	
	Grafo		Clique	
	Peso Máx	gmax	Peso Máx	gmax
10_0	72.25	5	67.25	4
10_1	79	5	65	3
100_0	1083.4	16	940.97	11
100_1	1396	20	982	11
1000_0	33453.8	240	23023.2	105
1000_1	220498	174	93132	102
10000_0	82041.2	304	37064.9	127
10000_1	458019	226	149625	125
100000_0	73261.2	217	37159.7	110
100000_1	439447	216	126730	115

Tabela 1: Cotas Superiores

Grafo	Maior clique por Carraghan-Pardalos					
	Tempo Matriz		Tempo Lista		Peso Máx	Tamanho
	c/ O3	s/ O3	c/ O3	s/ O3		
10_0	< 1ms				15.625	3
10_1	< 1ms				60	4
100_0	< 1ms				58.1	10
100_1	< 1ms				398	10
1000_0	30 ms	80 ms	90 ms	2.53 s	73.896	50
1000_1				2.29 s	11969	49
10000_0	1.07s	3.33 s	2.59 s	72.73 s	74.4516	80
10000_1		3.20 s		75.71 s	31024	80
100000_0	estoura a memória					
100000_1	estoura a memória					

Tabela 2: Heuristica #1: determinando o maior clique do grafo

Uma segunda abordagem seria a de obter o maior grau dentre os nós do grafo, g_{max}, e então observar que o maior clique possui no máximo esse tamanho. Assim, supomos que esse clique existe, observando que ele possui exatamente $q = \frac{g_{max}(g_{max}+1)}{2}$ arestas. Desse modo, a nossa cota superior é obtida através da soma dos pesos das maiores q arestas do grafo. Chamamos essa cota de **Cota Superior #1.**

Uma pequena melhoria nesse raciocínio é a de, em vez de apenas considerar o grau máximo do grafo, considerar o máximo grau que uma clique no grafo poderia ter (teoricamente). Por exemplo, não adianta haver uma estrela cujo maior grau é 1000 e aplicar a **Cota Superior #1** para tomarmos todas as arestas da estrela. O clique máximo de uma estrela tem tamanho 2. Assim, na **Cota Superior #2** tomamos o maior grau r que pudermos encontrar, tal que existam no mínimo r nós com grau maior ou igual a r.

3 Estudo de Caso

Heurística #1: vértices do clique encontrado:

- 10_0 $\Longrightarrow$ 4, 8, 10
- 10_1 $\Longrightarrow$3, 4, 5, 9
- 100_0 $\Longrightarrow$ 16, 34, 36, 47, 52, 66, 72, 82, 94, 97
- 100_1 $\Longrightarrow$ 2, 4, 17, 32, 38, 51, 53, 56, 71, 87

- 1000_0 $\Longrightarrow$ 27, 46, 63, 108, 113, 186, 203, 205, 209, 215, 216, 226, 254, 300, 312, 318, 377, 406, 428, 433, 435, 459, 508, 510, 516, 525, 526, 531, 545, 624, 628, 656, 688, 696, 699, 733, 741, 794, 806, 832, 855, 856, 873, 878, 893, 912, 934, 973, 983, 994
- 1000_1 $\Longrightarrow$ 22, 33, 35, 75, 92, 117, 118, 123, 130, 150, 172, 196, 213, 223, 239, 251, 269, 300, 309, 312, 342, 383, 385, 433, 456, 488, 508, 524, 535, 540, 549, 591, 659, 695, 752, 755, 766, 773, 804, 813, 856, 870, 877, 893, 921, 922, 949, 968, 984
- 10000_0 $\Longrightarrow$ 24, 71, 206, 250, 308, 345, 429, 951, 1075, 1252, 1356, 1583, 1622, 1628, 1901, 1982, 2034, 2095, 2405, 2718, 2791, 2808, 2922, 3154, 3198, 3363, 3395, 3400, 3473, 3547, 3628, 3705, 3764, 4576, 4774, 4779, 4783, 4846, 4847, 4947, 5136, 5187, 5258, 5277, 5404, 5406, 5766, 5906, 6100, 6140, 6300, 6492, 6568, 6617, 7023, 7056, 7384, 7423, 7466, 7541, 7581, 7769, 7867, 7952, 8066, 8179, 8264, 8340, 8507, 8751, 8930, 8948, 8996, 9060, 9162, 9278, 9310, 9361, 9369, 9500
- 10000_1 $\Longrightarrow$ 208, 243, 957, 1075, 1153, 1237, 1362, 1512, 1521, 1570, 1583, 2036, 2069, 2308, 2311, 2453, 2588, 2643, 2825, 2835, 2905, 2952, 3196, 3267, 3383, 3412, 3518, 3775, 3821, 3823, 3897, 4086, 4159, 4277, 4507, 4537, 4650, 4901, 4963, 4993, 5196, 5332, 5475, 5486, 5571, 5795, 5930, 5937, 6096, 6717, 6866, 6910, 7107, 7233, 7568, 7924, 7986, 8014, 8222, 8287, 8300, 8380, 8393, 8775, 8831, 8927, 8933, 9019, 9026, 9071, 9180, 9373, 9429, 9453, 9540, 9693, 9713, 9875, 9922, 9942

4 Conclusão

Note que encontramos boas previsões para os grafos 10_1, 100_1, e 10000_1 com as cotas superiores!

Note que para os dois últimos casos a memória utilizada é relativamente grande (em torno de 7,5 GB). Para 1 milhão de vértices, seria gasto em torno de 750GB (!) de memória, simplesmente tornando essa primeira heurística inviável. A conta é feita assim: $8n^2$, onde 8 bytes é o tamanho de um *double* e n^2 é a quantidade de memória que uma matriz de adjacência gasta.

Compensamos esse custo de memória pela velocidade com que o algoritmo acha o clique de tamanho máximo, e o fato de que em muitos casos esse é exatamente o clique de peso máximo!