Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
DCC059 - Teoria dos Grafos Semestre 2016-1

Conjunto independente máximo

Rafael de Souza Terra

Professor: Stênio Sã Rosáio F. Soares

Relatório do trabalho final de Teoria dos Grafos, parte integrante da avaliação da disciplina.

Juiz de Fora

Agosto de 2016

Sumário

1 Introdução

Seja um grafo $G = (V,E)$, o conjunto independente V_{ind} é subconjunto de vértices de G tal que não existe nenhuma aresta entre qualquer par de elementos do subconjunto. Um conjunto independente é classificado como máximo quando nao existe um conjunto $|V'_{ind}|$ tal que $|V'_{ind}| > |V_{ind}|$. O problema do conjunto independente máximo é classificado como NP. A seguir temos dois exemplos de aplicação:

- Reunir o maior número possível de pessoas do seu círculo de amizades que não se conhecem.
- O conjunto máximo de projetos que podem ser executados em paralelo.

2 Metodologia utilizada

Como o problema do conjunto independente máximo é do tipo NP, foi necessário o uso de um algoritmo construtivo guloso para obter uma solução. A heurística do algorimo guloso consiste em organizar o conjunto de vértices do grafo G pelo grau em ordem crescente.

2.1 Estruturas de dados utilizadas

O grafo do problema está armazenado na forma de uma lista duplamente encadeada onde cada vértice possui um atributo chamado lista de adjacências que armazena todos os IDs dos vértices adjacentes a este. A escolha desta estrutura deve-se à fácil implementação e ao menor gasto de memória comparado com algumas outras estruturas de dados.

2.2 Abordagens algoritmicas usadas na solução

Para resolver o problema do conjunto independente máximo fizemos o uso de um algoritmo construtivo guloso. Seja S o conjunto solução e C_0 o melhor candidato do conjunto de canditados.

Algorithm 1: *algoritmoConstrutivoGuloso*

Data: C - Conjunto de vértices candidatos;
Result: $|S|$

$S \leftarrow \emptyset$;

while $C \neq \emptyset$ **do**
    $S \leftarrow S \cup C_0$;
    $C \leftarrow C - C_0$;
    Remove todos os adjacentes de C_0 do conjunto de candidatos;

return $|S|$

Na tentativa de obter um resultado melhor foi implementado o algoritmo guloso com aleatoriedade passada como parâmetro.

Algorithm 2: *algoritmoConstrutivoGulosoComAleatoriedadePassadaComoParametro*

Data: C - Conjunto de vértices candidatos;
α - parâmetro de restrição da lista de candidatos;
Result: $|S|$

$k \leftarrow$ numero aleatório entre 0 e $\alpha \times |C|$;
$S \leftarrow \emptyset$;

while $C \neq \emptyset$ **do**
    $S \leftarrow S \cup C_k$;
    $C \leftarrow C - C_k$;
    Remove todos os adjacentes de C_k do conjunto de candidatos;

return $|S|$

Outro algoritmo usado para tentar melhorar o resultado obtido foi o algoritmo construtivo guloso com aleatoriedade ajustada automaticamente.

Algorithm 3: *probRandom*

Data: $P \leftarrow$ Lista de probabilidades;
$tam \leftarrow$ Tamanho do conjunto de parâmetros de restrição da lista;
Result: *indice*

$indice \leftarrow$ indice escolhido, inicializado com 0.0;
$soma \leftarrow$ soma das probabilidades, inicializado com 0.0;
$numRand \leftarrow$ valor aleatorio, inicializado com 0.0 **for** $it = 0$ *to* tam **do**
  $soma \leftarrow soma + P_{it} \times 1000$
$numRand \leftarrow$ número aleatório entre 0.0 e $soma$;
$soma \leftarrow 0.0$;
for $it = 0$ *to* tam **do**
  $soma \leftarrow soma + P_{it} \times 1000$;
  if $nRand <= soma$ **then**
    return it

Algorithm 4: *AlgoritmoConstrutivoGulosoComAleatoriedadeAjustadaAutomaticamente*

Data: C - Conjunto de vértices candidatos;
A -conjunto de parâmetros de restrição da lista;
Result: $|S|$

$P \leftarrow$ Lista de probabilidades com distribuição uniforme de tamanho $|A|$;
$Q \leftarrow$ Lista de tamanho $|A|$ inicializada com 0.0;
$nAlfa \leftarrow$ Lista de somatórios para os alphas inicializados em 0.0;
$valEncontrados \leftarrow$ Lista de somatorios das soluções encontradas para cada α inicializados em 0.0;
$media \leftarrow$ Lista de valores da função objetivo sobre o número de vezes que cada alfa foi usado;
$maxIt \leftarrow$ Número máximo de iterações do algoritmo, inicializado com 1500;
$bIteracao \leftarrow$ Número do tamanho do bloco de atualização, inicializado com 100;
$\alpha_{atual} \leftarrow \alpha$ atual usado em cada iteração, inicializado com 0.0;
$S_{atual} \leftarrow$ conjunto solução da iteração atual, inicializado com $NULL$;
$S_{melhor} \leftarrow$ conjunto da melhor solução, inicializado com $NULL$;
$indice \leftarrow$ valor do índice da iteração atual, inicializado com 0.0;
$emBloco \leftarrow$ Número auxiliar para o bloco de iteração, inicializado com 1.0;
$somQ \leftarrow$ Somatório dos elementos da lista Q, inicializado com 0.0;
for *it = 1 to maxIt* **do**
    if $it <= |A|$ **then**
        $\alpha_{atual} \leftarrow A_{it}$;
        $nAlfa_{it} \leftarrow nAlfa_{it} + 1$;
    else
        $indice \leftarrow probRandom(P, |A|)$;
        $\alpha_{atual} \leftarrow A_{indice}$
    $S_{atual} \leftarrow$
     $AlgoritmoConstrutivoGulosoComAleatoriedadePassadaComoParametro(C, \alpha_{atual})$;

    $valEncontrados_{indice} \leftarrow valEncontrados_{indice}+1$;
    if $S_{atual} > S_{melhor}$ **then**
        $S_{melhor} \leftarrow S_{atual}$;
    if $it == emBloco * bIteracao$ **then**
        $emBloco \leftarrow emBloco + 1$;
        for $i = 0$ *to* $|A|$ **do**
            $media_i \leftarrow valEncontrados_i / nAlfa_i$;
            $Q_i \leftarrow S_{melhor} / media_i$;
        $somQ \leftarrow 0.0$;
        for $(i = 0$ *to* $|A|)$ **do**
            $somQ \leftarrow somQ + Q_i$;
            $P_i \leftarrow Q_i / somQ$;

return $|S|$

3 Experimentos computacionais

Os experimentos foram realizados usando as 10 instâncias a seguir:

Tabela 1: Instâncias utilizadas

instâncias	Nº de vértices	Nº de arestas	Melhor valor (MIS)
Frb30-15-1.mis	450	17827	30
Frb35-17-1.mis	595	27856	35
Frb35-17-2.mis	595	27847	35
Frb40-19-1.mis	760	41314	40
Frb45-21-1.mis	945	59186	45
Frb50-23-1.mis	1150	80072	50
Frb53-24-1.mis	1272	94227	53
Frb56-25-1.mis	1400	109676	56
Frb59-26-1.mis	1534	126555	59
frb100-40.mis	4000	572774	100

Foram realizados 3 experimentos, todos usando um computador com um processador Intel Core i5-4200U CPU@1.60$GHz \times 4$, 8Gb de memória ram e sistema operacional Ubuntu 16.04.1 com arquitetura de 64 bits.

- **Experimento 1:** No primeiro experimento foi realizada a execução do algoritmo construtivo guloso para cada instância. A tabela a seguir exibe os resultados obtidos para este experimento.

Tabela 2: Experimento 1

Instâncias	Resultado obtido	Tempo de processamento(s)	Resultado da literatura	Porcentagem em relação ao resultado da literatura
Frb30-15-1.mis	20	0.002485	30	66,67%
Frb35-17-1.mis	25	0.004468	35	71,43%
Frb35-17-2.mis	25	0.004358	35	71,43%
Frb40-19-1.mis	31	0.007331	40	77,50%
Frb45-21-1.mis	35	0.012242	45	77,78%
Frb50-23-1.mis	36	0.015563	50	72,00%
Frb53-24-1.mis	37	0.020297	53	69,81%
Frb56-25-1.mis	41	0.02367	56	73,21%
Frb59-26-1.mis	45	0.030188	59	76,27%
Frb100-40.mis	72	0.19523	100	72,00%

- **Experimento 2:** No segundo experimento foram realizadas 30 execuções de cada instância, com 500 iterações cada execução, do algoritmo construtivo guloso com aleatoriedade passada como parâmetro para os α: 0.1, 0.2 e 0.3. A tabela a seguir exibe os melhores resultados dentre as 30 execuções para cada α usado.

Tabela 3: Experimento 2: $\alpha = 0.1$

Instância	Semente usada	Resultado	Tempo de processamento(s)
Frb30-15-1.mis	120	23	1.27382
Frb35-17-1.mis	128	29	2.32722
Frb35-17-2.mis	100	28	2.53128
Frb40-19-1.mis	100	33	4.40554
Frb45-21-1.mis	100	35	6.36876
Frb50-23-1.mis	100	39	8.96379
Frb53-24-1.mis	100	41	13.2229
Frb56-25-1.mis	100	43	14.5212
Frb59-26-1.mis	100	46	18.5549
Frb100-40.mis	101	76	130.6020

Tabela 4: Experimento 2: $\alpha = 0.2$

Instância	Semente usada	Resultado	Tempo de processamento(s)
Frb30-15-1.mis	108	24	1.27726
Frb35-17-1.mis	112	29	2.33199
Frb35-17-2.mis	100	28	2.32565
Frb40-19-1.mis	100	33	4.08592
Frb45-21-1.mis	100	36	6.28147
Frb50-23-1.mis	100	39	8.61547
Frb53-24-1.mis	100	42	11.7122
Frb56-25-1.mis	100	44	13.4988
Frb59-26-1.mis	100	46	17.2008
Frb100-40.mis	101	76	126.3070

Tabela 5: Experimento 2: $\alpha = 0.3$

Instância	Semente usada	Resultado	Tempo de processamento(s)
Frb30-15-1.mis	116	24	1.2893
Frb35-17-1.mis	121	30	2.34418
Frb35-17-2.mis	100	28	2.34643
Frb40-19-1.mis	100	33	4.09503
Frb45-21-1.mis	100	36	6.31869
Frb50-23-1.mis	102	40	8.62958
Frb53-24-1.mis	100	42	11.7184
Frb56-25-1.mis	100	44	13.6081
Frb59-26-1.mis	100	46	17.2723
Frb100-40.mis	104	76	125.7380

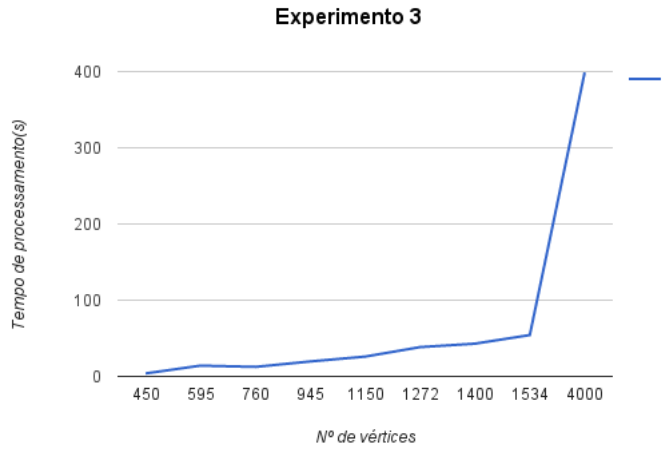
- **Experimento 3:** No terceiro experimento foram realizadas 30 execuções do algoritmo construtivo guloso com aleatoriedade ajustada automaticamente para cada instância, passando o seguinte conjunto de alfas: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0. A tabela a seguir exibe os melhores resultados dentre as 30 execuções.

Tabela 6: Experimento 3

Instância	Semente usada	Resultado	Melhor alfa	Tempo de processamento(s)
Frb30-15-1.mis	101	25	0.6	3.88474
Frb35-17-1.mis	124	30	0.3	7.10511
Frb35-17-2.mis	102	30	0.6	7.04912
Frb40-19-1.mis	107	34	0.4	12.4823
Frb45-21-1.mis	102	37	0.6	19.6122
Frb50-23-1.mis	120	41	0.8	26.0031
Frb53-24-1.mis	114	42	0.3	38.4242
Frb56-25-1.mis	105	45	0.4	42.9640
Frb59-26-1.mis	106	48	1.0	54.2497
Frb100-40.mis	101	78	0.9	398.9270

4 Conclusões

Pelos resultados obtidos, podemos concluir que o algoritmo construtivo guloso com aleatoriedade ajustada automaticamente com o conjunto de alfas escolhidos, foi o algoritmo que obteve melhores resultados, porém ele nao é tão eficiente em relação ao tempo de processamento, dificultando assim, a análise de instâncias com muitos vértices.



Referências

[1] BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring)
`http://www.nlsde.buaa.edu.cn/ kexu/benchmarks/graph-benchmarks.htm`

[2] Wikipédia - Conjunto independente máximo
`https://goo.gl/dVb6z0`