Raihan Kus Putranto | 2106703821 | Kode Asdos 1

TE 2 DAA

Dengan ini saya menyatakan bahwa TE ini adalah hasil pekerjaan saya sendiri

Deskripsi Singkat Algoritma Baru (Branch and Bound)

MVC, atau Minimum Vertex Cover, merupakan suatu permasalah yang memiliki tujuan untuk mencari vertex cover dengan jumlah simpul minimum. Misal, diberikan sebuah tree, yaitu graf G = (V, E), dengan V adalah himpunan simpul (vertex) dan E adalah himpunan sisi (edge), yang terkoneksi dan tidak memiliki loop. Sebuah vertex cover dari graf tersebut adalah himpunan simpul V` ⊆ V sedemikian sehingga setiap sisi dalam graf memiliki setidaknya satu ujungnya yang terhubung ke salah satu simpul dalam V`. Pada paper yang telah disediakan, diberikan empat algoritma yang akan digunakan untuk memecahkan permasalahan MVC (Minimum Vertex Cover). Saya akan menggunakan algoritma Branch and Bound yang merupakan salah satu dari keempat algoritma pada paper.

Pada dasarnya, ide algoritma Branch and Bound adalah dengan mencari himpunan simpul dari graf G untuk mencari vertex cover dengan memperhatikan batasan (bound) yang telah dibuat sebelumnya. Batasan tersebut harus terus menerus memperkecil solusi yang mungkin benar menjadi solusi yang optimal. Terdapat dua batasan yang perlu diinisiasi, yaitu *upper bound* dan *lower bound*. Upper Bound yang diinisiasikan merupakan solusi terbaik (vertex cover) yang terhitung pada setiap eksplorasi yang sedang dilakukan. Sedangkan, Lower bound yang diinisiasikan merupakan perbandingan antara jumlah *edge* di G` dengan *maximum node degree* di G` dimana G` merupakan graf G yang belum tereksplorasi. Algoritma ini mengkonsiderasi vertex secara *decreasing order* terhadap *vertex degree* sebagai kandidat yang paling cocok untuk mencari solusi optimal. Setiap kandidat vertex akan diberikan angka 1 atau 0 untuk mengindikasikan apakah vertex tersebut masuk ke dalam solusi kandidat Vertex Cover atau tidak, dan vertex tersebut akan ditambahkan ke Frontier Set. Frontier Set berisi semua kandidat vertex yang akan deksplorasi. Dengan begitu, jumlah skenario yang akan dilakukan adalah sebanyak $2^{|V|}$, sehingga time complexity-nya adalah $O(2^{|V|})$.

Berdasarkan eksperimen yang dilakukan pada paper tersebut, algoritma Branch and Bound ini berjalan dengan cukup cepat untuk mencari solusi pertama dengan waktu kurang dari 600 detik, tetapi berjalan sangat lama untuk mencari solusi kedua dalam waktu berjam-jam. Walaupun algoritma Branch and Bound berjalan dengan cukup lama dibandingkan algoritma lainnya yang tersedia di paper tersebut, algoritma ini tetap lebih unggul dalam hal akurasi dibandingkan dengan tiga algoritma lain yang digunakan pada paper tersebut.

Pemilihan algoritma untuk menyelesaikan permasalahan MVC harus didasarkan dengan *resource* yang tersedia dan akurasi yang diinginkan. Apabila hasil yang diinginkan berfokus pada akurasi dengan tidak terlalu mempedulikan running time yang lama, maka Branch and Bound adalah pilihannya.

Pseudocode Implementasi dan Dataset

```
Algorithm 2: Branch-and-Bound

Input  :Graph G of (V, E), Cutoff Time
Output :Vertex Cover of G
1 VC ← ∅ # will contain tuples of (vertex,state),state=1 if in VC, 0 otherwise
2 v ← vertex with highest degree
3 FrontierSet = [(v, 0, (-1, -1)), (v, 1, (-1, -1))]
# contains subproblems, each being tuple of (vertex,state,(parent
vertex,parent vertexstate))
4 # state of a vertex represents whether it is in VC; 1 if yes, 0 otherwise
5 UpperBound = |V | # initial upper bound=number of nodes in G
6 while FrontierSet , ∅ do
7   (vi,state,parent)=FrontierSet.pop()
8   VC ← (vi,state)
9   backtrack = false
10  if state == 0 then
11    Set state=1 for all neighbors of vi
12    VC ← All neighbors of vi
13    G ← G / (all neighbors of vi)
14  else if state == 1 then
15    G ⇐ G / vi
16  if G.edges== ∅ then
17    # end of exploring
18    if |VC| < UpperBound then
19      opt ← VC
20      UpperBound ← |VC|
21    backtrack = true
22    Move to next subproblem in FrontierSet
23  else
24    if LowerBound(G) + |VC| < UpperBound then
25      # worth exploring
26      vj ← vertex with highest degree among unexplored nodes
27      FrontierSet ← FrontierSet U [(vj, 0, (vi,state)), (vj, 1, (vi,state))]
28    else
29      # not worth exploring
30      backtrack = true
31      Move to next subproblem in FrontierSet
32    end
33  if backtrack == true and FrontierSet , ∅ then
34    # backtracking step
35    nextparent ⇐ last element in FrontierSet
36    if nextparent ∈ VC then
37      # remove current node from VC and add back to G
38      index ←index of nextparent in VC
39      for i=index to |VC| do
```

```
40        remove VC(i) from VC
41        add VC(i) and its edges back to G
42      end
43    else
44  reset VC ← empty
45  reset G ← original G
46  end
47 return opt
```

Link Github: https://github.com/raihankp/te2_daa

Hasil Eksperimen dan Analisis

Berikut ini adalah tabel yang berisi perbandingan waktu eksekusi (dalam miliseconds [ms]) dari algoritma DP dan BnB dengan 3 dataset yang berbeda. Dataset yang dibuat merupakan suatu binary tree yang diubah menjadi Undirected Graph dalam bentuk Adjacent List. Pada program Dynamic Programming, terdapat beberapa bagian yang saya ubah karena mendapatkan error maximum recursion depth

Dataset	Dynamic Programming Running Time	Branch and Bound Running Time
10^4 (DP) dan 100 (BnB)	3.6521000000000052	41.1937
10^5 (DP) dan 300 (BnB)	4.041400000000028	1000000.9043
10^6 (DP) dan 900 (BnB)	6.751900000000255	1000003.0302

Berdasarkan tabel di atas, terlihat bahwa *running time* algoritma Dynamic Programming cenderung lebih cepat dibandingkan algoritma Branch and Bound untuk mencari Minimum Vertex Cover. Akan tetapi, akurasi pada algoritma Branch and Bound lebih baik. Selain itu, terdapat perbedaan kompleksitas dari kedua algoritma tersebut secara teori (dalam notasi asimtotik). Dynamic Programming memiliki kompleksitas O(n) dengan n adalah jumlah vertexnya. Sedangkan, Branch and Bound memiliki kompleksitas $O(2^{|V|})$.

Kesimpulan

Algoritma baru Branch and Bound memiliki keunggulan dalam mencari Minimum Vertex Cover, yaitu memiliki akurasi yang lebih baik. Namun, algoritma tersebut memiliki kekurangan dalam hal *running time* yang cukup lama dibandingkan Dynamic Programming. Dynamic Programming bisa mendapatkan hasilnya dengan lebih cepat. Hal ini disebabkan oleh kompleksitas yang dimiliki kedua algoritma tersebut. Dynamic Programming memiliki kompleksitas O(n), sedangkan Branch and Bound memiliki kompleksitas $O(2^{|V|})$.