Penggunaan Metode *Branch And Bound With Search Tree* Untuk Menyelesaikan Persoalan Pedagang Keliling Pada Graf Lengkap Sebagai Pengganti Metode *Exhaustive Enumeration*

Alfan Farizki Wicaksono - NIM : 13506067

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16067@students.if.itb.ac.id

Abstraksi – *Persoalan pedagang keliling merupakan persoalan yang cukup terkenal di dalam teori graf, Implementasinya sangat banyak digunakan di dalam kehidupan sehari-hari, dari masalah yang berhubungan dengan pengiriman barang sampai yang berhubungan dengan industri pabrik Oleh karena itu, saya memilih persoalan ini sebagai tema dari makalah saya. Saya juga membatasi masalah yang ada yaitu pemecahan TSP graf lengkap (Kn) untuk lebih memberikan kejelasan yang dalam.*

Makalah ini membahas sebuah metode yang dapat digunakan untuk menyelesaikan persoalaan pedagang keliling (TSP) pada graf lengkap. Kita dapat melihat kemampuan dari metode ini dalam memecahkan persoalaan dari segi kompleksitas waktu dan kemudahan dalam penggunaan. Dalam pengerjaannya , metode Branch And Bound menggunakan Search Tree untuk menyimpan berbagai kemungkinan dari solusi yaitu jumlah bobot dari berbagai lintasan, dari bagian ini kemudian diolah untuk mendapatkan solusinya atau jarak dengan bobot yang minimum. Kita juga dapat melihat studi kasus dari persoalaan yang diberikan yaitu pencarian solusi dari TSP pada graf lengkap dengan derajat 5 (K5).

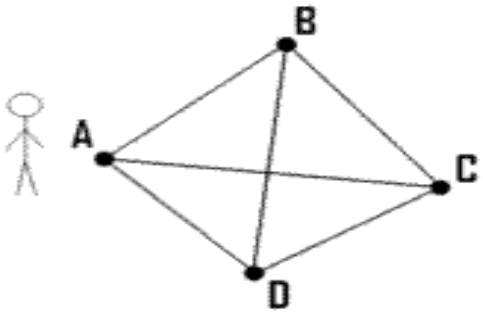
Makalah ini juga membahas mengenai cara dasar yang digunakan untuk pemecahan TSP, yaitu algoritma pemecahan TSP dengan enumerasi secara Brute Force. Tujuannya adalah agar memberikan perbandingan mengenai dua metode tersebut. Dengan begitu kita akan mengetahui bahwa metode Branch And Bound merupakan metode yang lebih baik.

Kata Kunci*: Branch, Bound, Tree, Search Tree, Branch And Bound Method, TSP, Pedagang Keliling, Graf, Graf Lengkap.*

1. PENDAHULUAN

Persoalan Pedagang Keliling (Travelling Salesman Person) merupakan salah satu masalah yang paling banyak muncul jika kita membahas mengenai teori graf. Persoalan ini diilhami oleh seorang pedagang keliling yang berkeliling kota untuk menjual dagangannya. Penjelasan dari masalahnya adalah sebagai berikut: “seorang pedagang keliling ingin menjual dagangannya ke seluruh kota, ia mulai dari kota satu dan berharap untuk mengunjungi setiap n-1 kota yang lain hanya sekali dan kembali lagi ke kota semula. Masalahnya adalah tentukan sirkuit terpendek yang harus dilalui oleh seorang pedagang tersebut dari berbagai kemungkinan sirkuit yang ada” [2].

Kota dapat dinyatakan sebagai simpul dari graf, sedangkan sisi menyatakan jalan yang menghubungkan antar dua buah kota. Bobot pada sisi menyatakan jarak lintasan antara dua bua kota. Persoalan pedagang keliling tidak lain merupakan menentukan sebuah sirkuit Hamilton dari graf yang mempunyai bobot minimum.



Gambar 1.1 : TSP

Persoalan pedagang keliling (TSP) tidak hanya diterapkan oleh seorang pedagang keliling yang berkeliling kota untuk menjual dagangannya. Banyak sekali masalah dalam kehidupan sehari – hari yang merupakan penerapan dari TSP, antara lain :

1. Misalkan sebuah mobil pos ditugaskan untuk mengambil surat dari kotak pos yang tersebar pada n buah lokasi di berbagai sudut kota. Persoaalan ini dapat disajikan dengan graf yang mempunyai n+1 simpul. Satu simpul menyatakan kantor pos tempat mobil pos mulai berangkat. Sisi (i, j) diberi bobot yang sama dengan jarak dari kotak pos i ke kotak pos j. Rute yang dilalui mobil pos adalah sebuah perjalanan yang mengnjungi setiap kotak pos hanya satu kali dan kembali lagi ke kotak pos awal. Kita harus menentukan rute perjalanan

yang mempunyai total jalan terpendek.

2. Dalam lingkungan produksi terdapat beberapa komoditi yang dihasilakan oleh sekumpulan mesin. Proses fabrikasi merupakan sebuah siklus. Tiap – tiap siklus produksi menghasilkan n komoditi berbeda. Bila pekerjaan mesin diubah dari produksi komoditas i ke produksi komoditas j, perubahan tersebut mendatangkan biaya sebesar Cij. Diinginkan untuk menemukan runtunan produksi yang menghasilkan n komoditas ini. Runtunan tersebut harus meminimumkan total biaya akibat perubahan urutan produksi. Karena komoditas diproses secara siklus, adalah perlu memasukan biaya untuk memulai siklus berikutnya. Ini adalah biaya akibat perubahan produksi komoditi terakhir ke komoditi pertama. Masalah ini dapat dianggap sebagai TSP pada graf n simpul dengan sisi bobotnya Cij.

Oleh karena itu, Persoalan Pedagang Keliling (TSP) ini merupakan persoalan yang penting dalam kehidupan kita bahkan beberapa industri. Hal ini akan menimbulkan pertanyaan tentang bagaimana menyelesaikan persoalan TSP ini? Ada beberapa metode yang digunakan untuk memecahkan persoalan ini. Salah satunya adalah dengan algoritma TSP enumerasi secara *Brute Force*. Namun, algoritma ini merupakan algoritma yang kurang baik karena mempunyai kompleksitas waktu yang lama. Oleh karena itu, persoalan ini membutuhkan pemecahan yang lebih baik. Saya memperkenalkan metode *Branch And Bound* untuk memecahkan masalah ini. Metode ini merupakan metode yang cukup baik untuk memecahkan masalah TSP. metode ini menggunakan pohon untuk menyimpan berbagai kemungkinan solusi untuk kemudian diolah dan didapat hasilnya. Saya hanya akan membahas pemecahan TSP menggunakan metode ini pada graf lengkap saja.

2. PEMECAHAN TSP DENGAN METODE EXHAUSTIVE ENUMERATION

Sebagai permulaan, kita akan mencoba untuk memecahkan Persoalan Pedagang keliling dengan menggunakan algoritma yang biasa (*cheapest*). Algoritma ini akan mengenumerasikan semua kemungkinan solusi sirkuit hamilton, kemudian akan mencari sirkuit mana yang mempunyai bobot paling kecil (minimum).

Didalam sebuah graf lengkap dengan n buah simpul terdapat ½ (n - 1)! Sirkuit Hamilton. Oleh karena itu, algoritma ini akan mengenumerasikan sebanyak jumlah tersebut pada sebuah graf lengkap kemudian akan menyimpan nilai dari bobot sirkuit yang ditemukan dan membandingkan nilainya dengan nilai bobot sirkuit lain, hal ini akan terus dilakukan sampai ½ (n – 1)! Kali.

Jika kita perhatikan, algoritma yang digunakan merupakan algoritma pencarian nilai minimum yang kemudian dimodifikasi dengan menggunakan fungsi atau posedur tertentu.

Tabel 1.1. Algoritma TSP (*Exhaustive*)

```
Function TSP (input G : graf) -> Bobot;

Var
Bmin, Temp : Bobot;
n : Integer;

Begin
  N = JumlahSimpul (G)
  Bmin = AmbilBobot (AmbilSirkuit)
  For I := 2 to ½(n - 1)! Do
    Begin
      Temp = AmbilBobot (AmbilSirkuit)
      If (Bmin > Temp) then
        Begin
         Swap (Bmin,  Temp)
        End
    End
  Return Bmin;
End
```

Jika kita olah berdasarkan kompleksitas aimptotik algoritma, kita peroleh nilai O(n!) untuk algoritma ini. Hasil ini didapatkan dari

$= O (1/2(n – 1) !)$

$=O(1/2\ n !)$

$=O(n !)$

Jika kita perhatikan, jelas algoritma ini tidak mangkus untuk nilai n yang besar, misalkan saja untuk n = 20, dibutuhkan enumerasi sebanyak 6×10^{16} penyelesaian. Ini akan membutuhkan waktu proses yang lama sekali. Oleh karena itu, Persoalan Pedagang Keliling membutuhkan metode yang lebih mengkus daripada algoritma *Exhaustive Enumeration*. Metode *Branch And Bound* merupakan salah satu metode yang dapat digunakan untuk menggantikan algoritma tersebut.

3. METODE *BRANCH AND BOUND WITH SEARCH TREE*

3.1. Definisi metode *Branch And Bound*

Metode *Branch And Bound* diusulakan pertama kali oleh A.H.Land dan A.G.Doig pada tahun 1960. Sebenarnya metode ini dibuat untuk pemrograman linier (*linier programming* [5]), Namun kenyataanya metode ini mampu menyelesaikan permasalahan seperti TSP dan beberapa masalah lain. Metode ini

menggunakan pohon pencarian (*Search Tree*), setiap simpul di pohon merupakan representasi dari sejumlah kemungkinan solusi dari TSP. Metode ini hanya dapat digunakan untuk masalah optimasi saja (*optimazion problem*).

Algoritma dimulai dengan pengisian sebuah nilai ke akar dari pohon pencarian tersebut. Pencabangan dilakukan dengan memasang sebuah *pending node* ke *pending node* lain yang lebih rendah levelnya. Bobot juga dihitung pada setiap proses dan ditulis di simpul pohon. Jika sebuah simpul diketahui merupakan solusi yang tidak mungkin bagi persoalan yang dihadapai, simpul tersebut diisi dengan nilai tak terbatas (infinity). Algoritma berhenti ketika sudah tidak mungkin lagi untuk membentuk simpul baru di pohon atau hasil terakhir yang ditemukan merupakan hasil yang lebih rendah (minimum) dari isi simpul yang telah ada pada level yang lebih rendah [3].

Kita definisikan S = {Ej} merupakan kumpulan dari berbagai kemungkinan solusi yang ada, dengan kata lain, E1, E2, E3, …,En merupakan himpunan simpul – simpul yang membentuk sirkuit Hamilton dan dari sekian banyak kemungkinan sirkuit Hamilton tersebut terdapat sebuah solusi untuk TSP. Kita definisikan juga | S | sebagai kardinalitas dari S dan f merupakan fungsi yang didefinisikan pada elemen Ej. Kita akan mencari solusi E* $\in$ S yang merupakan solusi minimum dari fungsi f.

Misalkan persoalan mempunyai properti yang memungkinkan melakukan partisi ($\prod$) dari himpunan bagian $S_{0lm..p}$ dari S.

$$\prod = \{ S_{0lm..p1}, S_{0lm..p2}, \ldots, S_{0lm..pn} \}, \quad n > 1 \quad (1)$$

Secara rekursif :

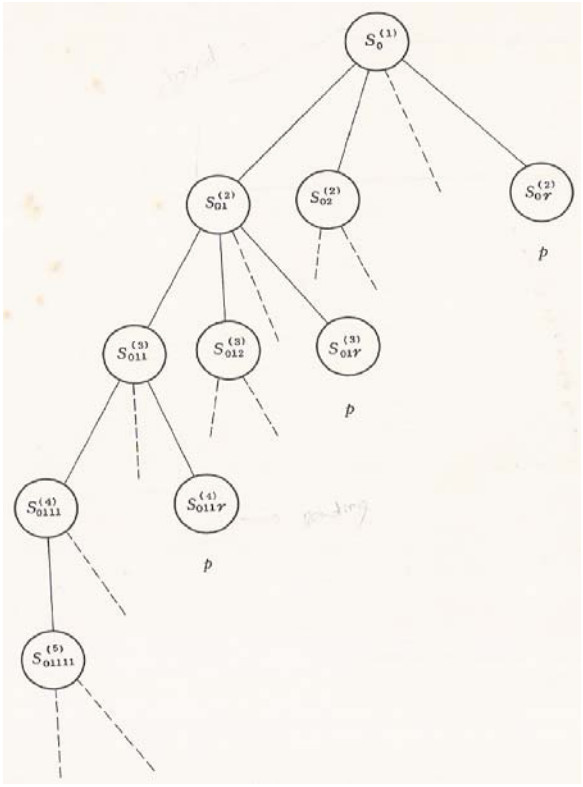
$$S_{0lm..pk} \neq \varnothing, \quad k = 1, 2, \ldots, q \quad (2)$$

$$S_{0lm..pk} \cap S_{0lm..pj} = \varnothing, \quad k,j = 1, 2, \ldots, q, \quad k \neq j \quad (3)$$

Dengan kondisi awal :

$$S_0^{(1)} = S$$

Representasi gambar dari persoalan ini dilukis pada gambar 2.1, di sini kita melihat sebuah pohon pencarian (search tree). Setiap simpul merupakan sebuah himpunan bagian $S_{0lm..pk}$ dari Solusi. *Superscript* i menyatakan level simpul di pohon pencarian. *0lm..pk* menyatakan *path* atau jalan dari akar pohon.



Gambar 2.1 : *Solution Search Tree*

Closed node pada pohon merupakan simpul yang sudah tidak diproses lagi (tidak bisa dipartisi lagi). Sedangakan *Pending node* merupakan simpul yang belum ditutup artinya masih bisa diproses lagi atau di partisi. Sebagai contoh, S_0, S_{01} merupakan closed node dan S_{011} merupakan pending node.

Selama simpul pohon masih bisa dipartisi menjadi dua atau lebih, kardinalitas dari himpunan bagian menurun sejalan dengan bertambahnya level.

$$| S_{011} | < | S_{01} | < \ldots < | S_0 | = | S |$$

Kita akan mencapai sebuah simpul (*Pending node*) yang hanya mengandung satu elemen dari S, yang juga mengandung solusi. Simpul ini disebut *Terminal node* [1].

Tujuan dari metode ini adalah untuk menemukan solusi dengan mengenumerasikan sedikit mungkin simpul pohon. Dalam melakukan pencabangan harus dipilih sebuah *pending node* yang mempunyai bobot paling kecil. Pencarian solusi dengan pencabangan berhenti ketika kita telah menemukan solusi yang mungkin dan merupakan bobot terkecil dari semua bobot *pending node* [1].

3.2. Implementasi dalam pemecahan TSP pada graf lengkap

Metode Branch And Bound sebenarnya bukan merupakan metode yang mutlak untuk penyelesaian TSP, metode ini merupakan kumpulan dari berbagai cara penyelesaian masalah (*class of solving problem*),

hanya saja persamaan karakteristik cara – cara tersebut yang membuat mereka disebut *Branch And Bound*. Saya menyajikan salah satu metode dari sekian banyak metode yang digunakan didalam *Branch And Bound*.

Graf lengkap mempunyai jumlah derajat yang sama pada setiap simpul. Solusi TSP pada graf ini merupakan sirkuit Hamilton yang mempunyai bobot paling kecil (minimum). Metode *Branch And Bound* dapat digunakan untuk memecahkan masalah ini pada graf Lengkap. Selama pencarian solusi, *current node* merupakan simpul pohon yang akan diproses selanjutnya atau yang mendapat perhatian. Sirkuit dari *current node* inilah yang nanti akan diproses. Sisi yang yang tidak boleh dimasukkan kedalam solusi disebut sisi yang terlarang (*forbidden*) dan sebaliknya disebut bebas (*free*) [1]. Disetiap simpul pohon, kita merekam bobot dari sirkuit yang diperoleh sebagai kemungkinan solusi. Jika sirkuit tersebut mengandung sisi yang terlarang (yang menyebabkan sirkuit terseut terlarang nantinya) , sisi tersebut akan terlarang sampai anak – anak pohon di bawahnya. L0 merupakan batas atas dari solusi setiap waktu pencarian. Jadi sirkuit yang melebihinya jelas bukan merupakan solusi dari TSP. Berikut disajikan algoritma dari metode Branch And Bound pada pemecahan TSP di graf lengkap Kn.

Algoritma :

Langkah 1. inisialisasi. Ambil salah satu sirkuit (bebas), biasanya sirkuit yang merupakan lintasan terluar, hitung bobotnya dan berinama L0. L0 merupakan batas atas awal dari masalah ini. E0 merupakan himpunan simpul dari sirkuit ini. Akar dari search tree diisi dengan bobot dari L0. kemudian simpul ini dipersiapkan untuk dipartisi menjadi 2 simpul pohon (*current node*).

Langkah 2. Menghitung batas bawah yang baru. Hitung batas bawah yang baru yaitu *l* untuk current node dengan mangambil *n* simpul yang mempunyai bobot terkecil. Simpan himpunan simpul ini sebagai E.

Jika E bukan sirkuit, langsung ke langkah 3, yang lain, ke langkah 2.1

Langkah 2.1. jika *l* = L0 atau *l* < batas terbawah yang ada pada pending node yang bersesuaian, hentikan proses; E adalah solusi TSP-nya.

Langkah 2.2. jika *l* < L0, *l* diisi dengan L0 dan E diisi dengan E0, lanjut ke langkah 3.

Langkah 3. Bersiap untuk mempartisi simpul pohon. Cari *pending node* yang mempunyai bobot paling kecil (calon simpul yang akan dipartisi). Jika simpul yang sudah dipilih tersebut merupakan sebuah sirkuit, hentikan pencarian. Simpul tersebut merupakan solusinya. Solusinya adalah sirkuit graf lengkap yang berkorespondensi dengan simpul pohon tersebut.

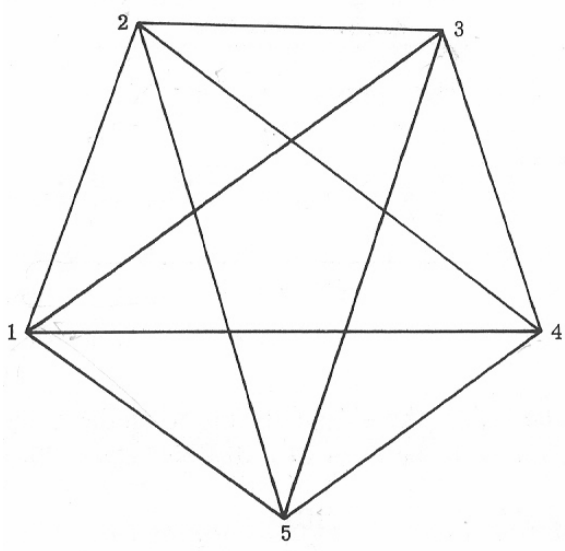
Langkah 3.1. jika E mengandung simpul dari Kn yang mempunyai derajat lebih dari 2. Pilih salah satu sisi yaitu *e* (yang belum diproses sebelumnya) yang bersisian dengan simpul tersebut. Maju ke langkah 4.

Langkah 3.2. jika tidak ada simpul yang lebih besar dari 2 tetapi mengandung *upasirkuit*, maka *e* diisi dengan salah satu simpul yang membangun *upasirkuit* tersebut.

Langkah 4. Lakukan Partisi. $S_{0...kl}$ dipartisi menjadi 2 simpul. Satu simpul yaitu $S_{0...kl1}$ mempunyai E sebagai sebuah solusi. Satu simpul lagi yaitu $S_{0...kl2}$ menyimpan E sebagai sisi yang terlarang dan kemudian akan di tutup (*closed node*). Jika $S_{0...kl1}$ menyimpan sebuah sisi yang membentuk sebuah upasirkuit di graf lengkap atau membentuk derajat lebih dari 2 di sebuah simpul graf lengkap, isi dari simpul pohon tersebut diisi dengan lambing ∞ (infinity), artinya simpul pohon ini bukan sebuah solusi yang mungkin dan kemudian akan menjadi *closed node*. Setelah itu, maka simpan nilai dari simpul $S_{0...kl1}$ sebagai *l* yang baru. Kemudian, tutuplah simpul induknya yaitu $S_{0..kl}$. Sekarang, simpul pohon yang akan diproses adalah simpul $S_{0...kl2}$ (*current node*). Ulangi lagi ke langkah 2.

4. PENGUJIAN METODE BRANCH AND BOUND PADA GRAF LENGKAP K5

Kita mulai dengan mendefinisikan masalah seperti berikut.



Gambar4.1 : Graf K5

Diberikan sebuah graf lengkap dengan derajat 5 (K5), kita diminta untuk memecahkan TSP pada graf ini jika diberikan bobot – bobot sisi

Tabel 4.1 : Jarak Antar Sisi

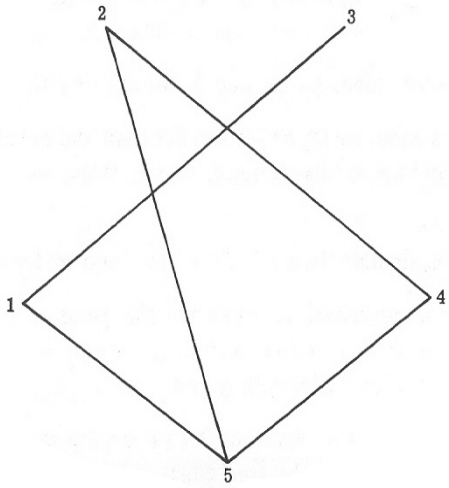
	2	3	4	5
1	13	3	19	4
2		25	3	5
3			12	10
4				8

Masalah ini dapat kita pecahkan dengan menggunakan metode *Branch And Bound*. Algoritmanya adalah sebagai berikut

Perulangan 1

Langkah 1. kita ambil E0 yaitu {a12, a23, a34, a45, a51}, semua sisinya bebas (*free*). Bobot dari sirkuit ini adalah L0 = 62 dan S = S_0.

Langkah 2. kita ambil 5 sisi dengan bobot terkecil. Yaitu E = {a13, a24, a25, a15, a45} dengan bobot *l* = 23 (gambar 4.2). Siman *l* di simpul pohon.



Gambar 4.2 : Upagraf E (1)

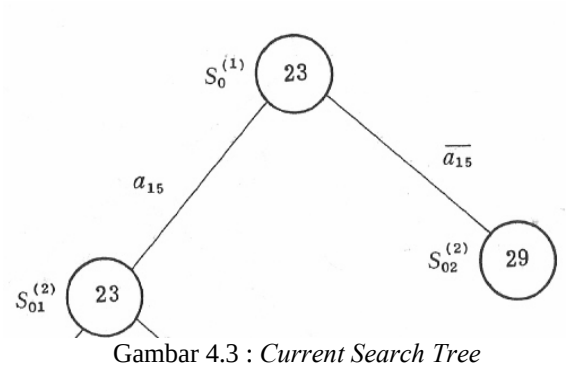
Langkah 3. Simpul 5 pada graf (gambar 4.2) mempunyai derajat > 2 yaitu 3. Misal, kita ambil sisi a15 untuk langkah selanjutnya.

Langkah 4. definisikan sebuah pernyataan untuk partisi simpul pohon.

S_{01} = {Ej ∈ S_0 | a15 ∈ Ej}

S_{02} = {Ej ∈ S_0 | a15 ∉ Ej}

S_0 ditutup. Sekarang *current node* adalah S_{02}. Batas bawah sementara adalah 23.



Gambar 4.3 : *Current Search Tree*

Perulangan 2

Langkah 2. *current node* adalah S_{02} ; dengan a15 terlarang. Jadi kita ganti sisi a15 dengan sebuah sisi, misalnya a35 (gambar 4.5).

E = {a13, a24, a25, a35, a45}, *l* = 29

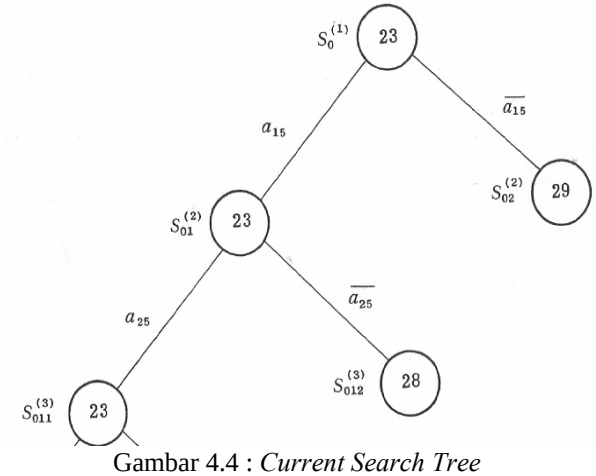
Langkah 3. simpul pada *pending node* yang memliki bobot terkecil adalah S_{01} dengan bobot 23 karena S_{02} mempunyai bobot 29. Oleh karena itu, S_{01} yang akan kita gunakan untuk pencabangan (*Branching*) selanjutnya. Sisi bebas pada simpul 5 tinggal a25 dan a45, karena a15 sudah diproses sebelumnya. Misal, kita ambil a25 untuk proses selanjutnya.

Langkah 4. definisikan

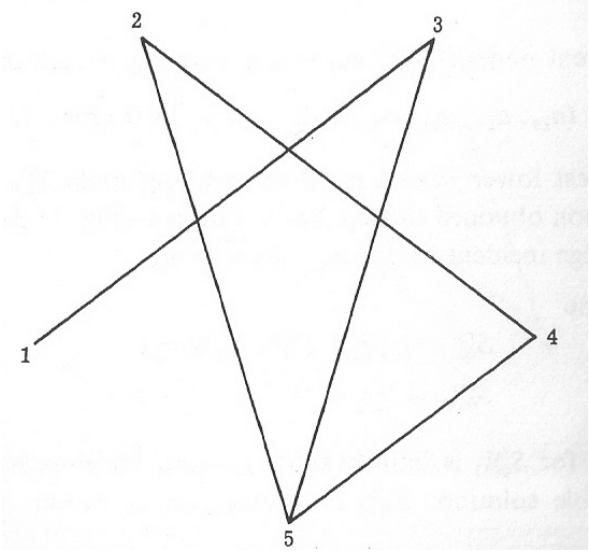
S_{011} = {Ej ∈ S_{01} | a25 ∈ Ej}

S_{012} = {Ej ∈ S_{01} | a25 ∉ Ej}

S_{01} ditutup. Sekarang *current node* adalah S_{012}. Batas bawah sementara adalah 23.



Gambar 4.4 : *Current Search Tree*



Gambar 4.5 : Upagraf E (2)

Perulangan 3

Langkah 2. *current node* adalah S_{012}, a15 sudah diproses dan a25 dilarang. a15 tidak dilarang karena simpul S_{012} bukan turunan dari simpul $S_{02.}$ Jadi, kita ambil sisi a45 (gambar 4.6).

$$E = \{a13, a15, a24, a35, a45\},\ l = 28$$

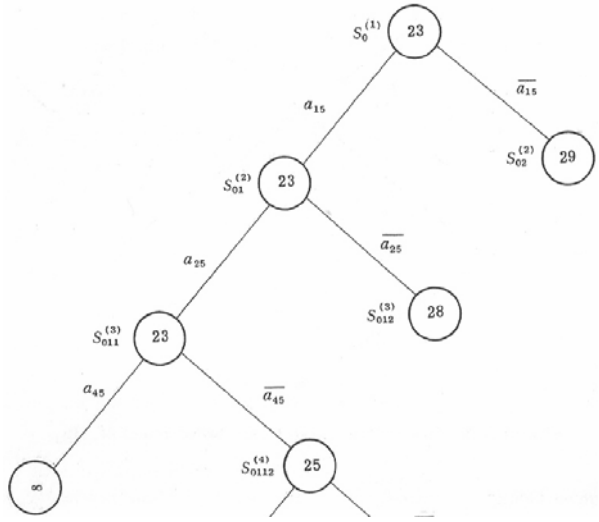
Langkah 3. simpul pada pending node yang memiliki bobot terkecil adalah dengan S_{011} bobot 23. Oleh karena itu, simpul inilah yang akan digunakan untuk *branching* selanjutnya. Sisi bebas pada simpul 5 yang mempunyai derajat 3 tinggal a45. Jadi, kita ambil a45.

Langkah 4. definisikan

$S_{0111} = \{Ej \in S_{011} \mid a45 \in Ej\}$

$S_{0112} = \{Ej \in S_{011} \mid a45 \notin Ej\}$

Batas untuk S_{0111} adalah ∞ karena ketiga sisi yang membentuk derajat 3 semuanya sudah diproses yaitu a15, a25, a45 (membantuk simpul derajat 3). Jadi S_{0111} harus ditutup. S_{011} ditutup. Sekarang *current node* adalah S_{0112}. Batas bawah sementara adalah 28.



Gambar 4.6 : *Current Search Tree*

Perulangan 4

Langkah 2. *current node* adalah S_{0112}, a15, a25 sudah diproses dan a45 dilarang. a15, a25 tidak dilarang karena simpul S_{012} bukan turunan dari simpul $S_{02,}$ $S_{012.}$ Jadi, kita ambil sisi a25 (gambar 4.7).

$$E = \{a13, a15, a24, a35, a25\},\ l = 25$$

Langkah 3. Batas bawah adalah 25 pada pending node yaitu oleh simpul S_{0112} . Simpul 5 pada graf lengkap masih mempunyai derajat 3, dan satu-satunya sisi yang belum diproses adalah a35, jadi kita ambil sisi a35 untuk proses selanjutnya.

Langkah 4. definisikan

$S_{01121} = \{Ej \in S_{0112} \mid a35 \in Ej\}$

$S_{01122} = \{Ej \in S_{0112} \mid a35 \notin Ej\}$

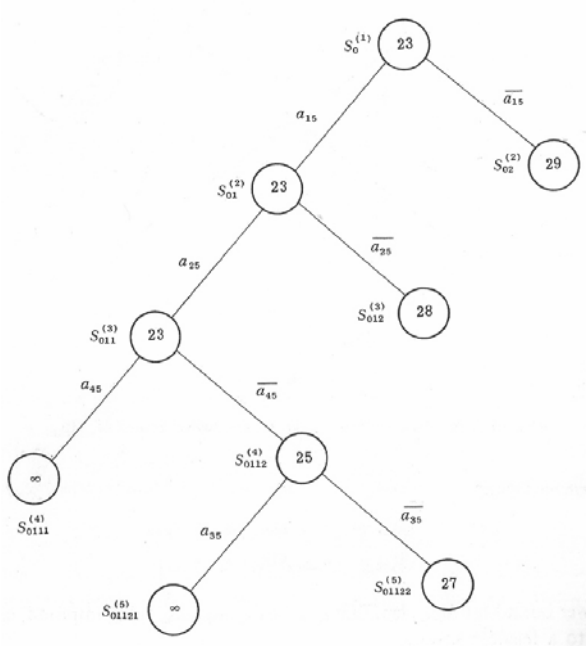
Batas bawah untuk S_{01121} adalah ∞ karena ketiga sisi yang membentuk derajat 3 semuanya sudah diproses yaitu a15, a25, a35 (membentuk simpul derajat tiga). Sekarang *current node* adalah $S_{01122,}$ S_{01121} dan S_{0112} ditutup. Batas bawah sementara adalah 25.

Perulangan 5

Langkah 2. *current node* adalah S_{01122}, a15 dan a25 sudah diproses, a35 dan a45 dilarang. jadi kita ambil a34.

$$E = \{a13, a15, a24, a25, a34\},\ l = 27$$

E adalah sebuah sirkuit, jadi proses pencarian dihentikan karena kita sudah menemukan solusinya yaitu E. 27 lebih kecil dari 28 dan 29 (gambar 4.7).



Gambar 4.7 : *Final Search Tree*

Dari pengujian terlihat bahwa untuk memecahkan TSP pada graf K5, metode ini hanya melakukan enumerasi sebanyak 9 simpul. Hal ini jauh lebih baik dibandingkan mengenumerasikan semua sirkuit sebanyak ½(5 -2)! = 12 cara.

5. PENGUJIAN DAN PEMBAHASAN

Metode *Exhaustive Enumeration* akan melakukan perulangan sebanyak jumlah sirkuit Hamilton yang ada pada graf Kn yaitu sebanyak ½(n – 1)! Kali. Kita sudah mengetahui bahwa hal ini akan mengakibatkan lonjakan waktu penyelesaian yang begitu besar seiring dengan bertambahnya jumlah masukan. Hal ini disebabkan kompleksitas asimptotik metode merupakan fungsi factorial atau T(n) = O(n!).

Berikut data jumlah perulangan yang harus dilakukan seiring bertambahnya masukan dengan metode *exhaustive enumeration*

Tabel 4.1. Langkah sebagai fungsi dari n

Jumlah Masukan (n)	Langkah
4	3
6	60
10	181440
20	6×10^{16}

Dari tabel 4.1 jelas terlihat bahwa jumlah langkah akan melonjak dengan drastis walaupun jumlah masukan hanya bertambah 5 atau 10 masukan. Bagaimana jika kita membutuhkan penyelesaian TSP dengan graf yang mempunyai simpul 30 atau lebih. Oleh karena itu, kita membutuhkan metode yang jauh lebih baik lagi.

Metode *Branch And Bound* menggunakan menggunakan pohon pencarian dengan mempartisi beberapa solusi. Metode ini juga menggunakan *lower bound* (batas bawah) dari sebuah *current node* untuk melanjutkan pencabangan, akibatnya simpul yang yang bukan *lower bound* akan berhenti pencabangannya. Hal – hal yang dilakukan dalam metode ini jelas memberikan gambaran bahwa tidak perlu memeriksa semua kemungkinan solusi untuk mendapatkan solusi yang sebenarnya. Dengan begitu, kemangkusan dari metode jelas lebih hebat dibanding metode *Exhaustive Enumeration.*

Efisiensi dari metode ini bergantung pada proses *splitting* simpul pohon [5]. Dengan kata lain, metode ini mempunyai Kompleksitas rata – rata. Jika dibandingkan dengan metode sebelumnya (Kompleksitas kasus baik sama dengan Kompleksitas kasus buruk), metode ini jelas lebih baik. Efisiensi dari metode ini juga dipengaruhi oleh algoritma pencabangan dan pencarian batas, jika algoritma yang digunakan harus melakukan perulangan, waktu penelesaian akan lama. Akan tetapi, jika algoritma pencabangan dan pencarian batas hebat, metode ini mempunyai kemampuan yang sangat luar biasa.

Sebuah sumber menyebutkan bahwa metode *Branch And Bound* mampu menyelesaikan Persoalan Pedagang Keliling dengan 40 – 60 simpul dalam waktu yang singkat. Pada saat ini, modifikasi dari metode ini seperti metode *cutting plane* mampu menyelesaikan sebanyak 15112 simpul. Pada bulan April 2006, TSP dengan jumlah simpul sebanyak 85900 mampu diselesaikan dengan *CONCORDE* memanfaatkan modifikasi dari metode *Branch And Bound* [4].

6. KESIMPULAN

Pemecahan Persoalan Pedagang Keliling dengan menggunakan Metode Exhaustive Enumeration memang mempunyai algoritma yang mudah dimengerti dan sederhana, namun algoritma ini mempunyai masalah didalam penanganan jumlah masukan, hal ini disebabkan karena algoritma ini mempunyai kompleksitas algoritma O(n!), jadi jika n bernilai besar, algoritma ini akan memakan waktu yang lama sekali.

Untuk mengatasi hal ini, Metode *Branch And Bound* merupakan salah satu alternatif yang dapat kita gunakan untuk mengatasi hal ini. Metode ini menggunakan pohon pencarian sebagai pegerjaannya, setiap simpul pada pohon pencarian merupakan kemungkinan solusi dari Persoalan Pedagang Keliling. Oleh karena itu, walaupun metode ini merupakan metode *heuristic* (pendekatan), metode ini mempunyai kemangkusan yang jauh lebih baik dibandingkan

algoritma *Exhaustive Enumeration*.

Kita juga dapat memodifikasi metode *Branch And Bound* ini untuk mendapatkan metode turunan yang mempunyai kemampuan yang lebih hebat.

DAFTAR REFERENSI

[1] E.J. Henley, R.A. Wiliams, "Graph Theory in Modern Engineering",*Academic Press*, 1973, hal.190 - 199.

[2] Rinaldi Munir, "Matematika Diskrit", Informatika Bandung, 2005, hal.421 – 424.

[3] M. Gondran, M. Minoux, "Graphs And Algorithms", *John Wiley & Sons*, 1984, hal.496 – 514.

[4] Travelling Salesman Problem. (2007). *http://wikipedia.org/wiki/Travelling_salesman_person*.
Tanggal akses : 29 Desember 2007 pukul 07:00

[5] Branch And Bound. (2007). *http://wikipedia.org/wiki/Branch_And_bound*.
Tanggal akses : 29 Desember 2007 pukul 07.1