Penggunaan Algoritma Branch & Bound dan Dynamic Programming pada Treasure Hunter

Marcelinus Henry M. 13512082
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512082@std.stei.itb.ac.id

***Abstrak* – Dalam berbagai persoalan mengenai jarak terdekat dan jarak efektif, algoritma *branch and bound* menjadi beberapa cara yang sangat tepat dan sangat populer untuk menyelesaikan persoalan-persoalan yang ada. Algoritma tersebut, tanpa kita sadari telah kita gunakan untuk memecahkan masalah sehari-hari. Makalah ini akan menjelaskan bagaimana algoritma tersebut digunakan untuk mendapatkan solusi jarak terdekat dalam permainan *treasure hunter*.**

***Index Terms* – jarak terdekat, cost, node hidup, solusi**

I. PENDAHULUAN

Pada makalah ini akan dibahas tentang mendapatkan solusi minimum dari suatu permasalahan *TSP* (*The Salesperson Problem*) yaitu permasalahan tentang mencari jalan terpendek untuk melewati semua node dan kembali ke node awal. Pada permasalahan ini, *TSP* yang dihadirkan memiliki beberapa pengecualian. Permasalahan yang dihadapi yaitu seorang perompak berangkat dari suatu pulau, katakana pulau 1. Perompak tersebut ingin pergi ke semua pulau yang memiliki harta karun dan mendapatkannya, kemudian kembali ke pulau awal. Namun tidak semua pulau terhubung dengan suatu jalur langsung dan tidak setiap pulau memilki harta karun. Bagaimana jalur yang harus diambil perompak agar jalurnya minimum ? Permasalahan seperti ini tidak bisa diselesaikan dengan algoritma biasa, tetapi harus dengan beberapa tahap proses. Pada makalah ini juga menggunakan dua buah algoritma yaitu Branch & Bound dan Dynamic Programming dimana dua algoritma ini bisa menyelesaikan permasalahan baik jarak terdekat ataupun *TSP*.

II. TEORI

Algoritma *Branch & Bound*

Algoritma Branch & Bound adalah algoritma seperti BFS namun simpul yang akan diekspansi berikutnya ditentukan berdasarkan nilai cost yang ditentukan. $\hat{c}(i)$ = nilai taksiran lintasan minimal dari simpul status i ke simpul status tujuan.

Langkah-langkah Branch & Bound

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi, maka solusi telah ditemukan. Berhenti.
2. Jika Q kosong, tidak ada solusi. Berhenti.
3. Jika Q tidak kosong, pilih dari antrian Q simpul I yang mempunyai cost paling kecil. Jika terdapat beberapa simpul I yang memenuhi, pilih yang mempunyai kedalaman lebih dalam.
4. Jika simpul I adalah simpul solusi, berarti solusi sudah ditemukan, berhenti. Jika simpul I bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
6. Kembali ke langkah 2.

Algoritma Dynamic Programming

Dynamic Programming adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan.

Karakteristik penyelesaian persoalan dengan Program Dinamis:

1. terdapat sejumlah berhingga pilihan yang mungkin,
2. solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. kita menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas. Prinsip Optimalitas: jika solusi total optimal, maka bagian solusi sampai tahap ke-*k* juga optimal. Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap *k* ke tahap *k* + 1, kita dapat menggunakan hasil optimal dari tahap *k* tanpa harus kembali ke tahap awal. Ongkos pada tahap *k* +1 = (ongkos yang dihasilkan pada tahap *k*) + (ongkos dari tahap *k* ke tahap *k* + 1)

Karateristik persoalan program dinamis

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status

(*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut.

3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap $k + 1$.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Misalkan x_1, x_2, …, x_n menyatakan peubah (*variable*) keputusan yang harus dibuat masing-masing untuk tahap 1, 2, …, n. Maka,

1. Program dinamis maju. Program dinamis bergerak mulai dari tahap 1, terus maju ke tahap 2, 3, dan seterusnya sampai tahap n. Runtunan peubah keputusan adalah $x_1, x_2, \ldots, x_n$.
2. Program dinamis mundur. Program dinamis bergerak mulai dari tahap n, terus mundur ke tahap $n - 1$, $n - 2$, dan seterusnya sampai tahap 1. Runtunan peubah keputusan adalah $x_n, x_{n-1}, \ldots, x_1$.

Prinsip optimalitas pada PD maju: ongkos pada tahap k +1 = (ongkos yang dihasilkan pada tahap k) + (ongkos dari tahap k ke tahap $k + 1$) $k = 1, 2, \ldots, n - 1$

Prinsip optimalitas pada PD mundur:ongkos pada tahap k = (ongkos yang dihasilkan pada tahap $k + 1$) + (ongkos dari tahap $k + 1$ ke tahap k) $k = n, n - 1, \ldots, 1$

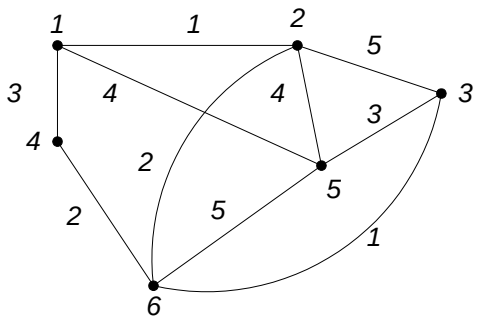
Langkah-langkah pengembangan

1. Karakteristikkan struktur solusi optimal.
2. Definisikan secara rekursif nilai solusi optimal.
3. Hitung nilai solusi optimal secara maju atau mundur. Konstruksi solusi optimal.

III. MENENTUKAN JARAK TERDEKAT

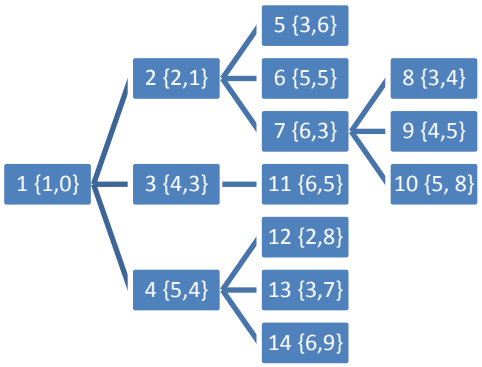
A. Menentukan jarak terdekat sederhana

Dalam menentukan jarak terdekat, dua algoritma yang telah disebutkan dapat digunakan, tentu saja dengan cara yang berbeda. Disini penulis akan menggunakan algoritma Branch & Bound.



Gambar 3.1

Seperti yang dilihat pada gambar 1, kita akan menentukan jarak terdekat dari 1 ke 3. Dengan algoritma Branch & Bound, kita tentukan cost dari setiap node adalah panjang jarak yang dilalui dari akar ke daun melewati node-node yang telah dibangkitkan, seperti pada gambar berikut.



Gambar 3.2

Gambar 3.2 menunjukkan pembangkitan pohon dengan algoritma Branch & Bound. Penulisan pada node yaitu urutan pembangkitan daun, {node tujuan, cost untuk sampai ke node tersebut}. Urutan pembangkitan yaitu dari akar (node 1) kemudian diekspansi ke node 2, 3, dan 4. Node 1 tidak diekspansi ke node 5 ataupun 6 karena tidak ada jalur yang langsung menghubungkan node-node tersebut dan dalam kasus ini dianggap memiliki cost tak hingga.

Kemudian dari daun-daun hidup (node hidup) yang telah dibangkitkan, dipilih daun yang memiliki cost minimal, yaitu daun 2. Daun ini kemudian diekpansi dengan cara yang sama seperti kita mengekspansi akar. Didapatlah daun 5, 6, dan 7. Ditemukan satu solusi yaitu daun 5, namun masih terdapat daun yang lain yang memiliki cost lebih kecil dari solusi yang ditemukan yang bisa saja menjadi solusi efektif. Maka dari itu algoritma berlanjut dengan mengekspansi daun hidup dengan cost terkecil yaitu daun 3. Hal ini terjadi terus hingga didapat solusi yaitu daun 9, dimana cost bernilai paling kecil yaitu 4. Jadi solusi jarak terdekat dari 1 ke 3 yaitu 1-2-6-3.

B. Menentukan solusi TSP sederhana

Dalam menentukan solusi TSP (The Salesperson Problem) penulis akan menggunakan algoritma Dynamic Programming.

$$\begin{bmatrix} \infty & 10 & 15 & 20 \\ 5 & \infty & 9 & 10 \\ 6 & 13 & \infty & 12 \\ 8 & 8 & 9 & \infty \end{bmatrix}$$

Matriks diatas adalah contoh matriks ketetanggaan yang akan kita cari solusi TSP-nya. Dapat kita lihat bahwa matriks diatas memiliki $n = 4$. Maka solusi untuk TSP dengan Dynamic Programming akan membutuhkan 4 tahap.

Tahap 1

$f(i,\phi) = c \quad , 2 \le i \le n$

$f(2,\phi) = c_{21} = 5$

$f(3,\phi) = c_{31} = 6$

$f(4,\phi) = c_{41} = 8$

Tahap 2

$f(i,S) = \min\{c_{ij} + f(j, S - \{j\})\}$

$f(2,\{3\}) = \min\{c_{23} + f(3,\phi)\} = \min\{9+6\} = 15$

$f(2,\{4\}) = \min\{c_{24} + f(4,\phi)\} = \min\{10+8\} = 18$

$f(3,\{2\}) = \min\{c_{32} + f(2,\phi)\} = \min\{13+5\} = 18$

$f(3,\{4\}) = \min\{c_{34} + f(4,\phi)\} = \min\{12+8\} = 20$

$f(4,\{2\}) = \min\{c_{42} + f(2,\phi)\} = \min\{8+5\} = 13$

$f(4,\{3\}) = \min\{c_{43} + f(3,\phi)\} = \min\{9+6\} = 15$

Tahap 3

$$\begin{aligned} f(2,\{3,4\}) &= \min\{c_{23} + f(3,\{4\}), c_{24} + f(4,\{3\})\} \\ &= \min\{9+20, 10+15\} \\ &= 25 \end{aligned}$$

$$\begin{aligned} f(3,\{2,4\}) &= \min\{c_{32} + f(2,\{4\}), c_{34} + f(4,\{2\})\} \\ &= \min\{13+18, 12+13\} \\ &= 25 \end{aligned}$$

$$\begin{aligned} f(4,\{2,3\}) &= \min\{c_{42} + f(2,\{3\}), c_{43} + f(3,\{2\})\} \\ &= \min\{8+15, 9+18\} \\ &= 23 \end{aligned}$$

Tahap 4

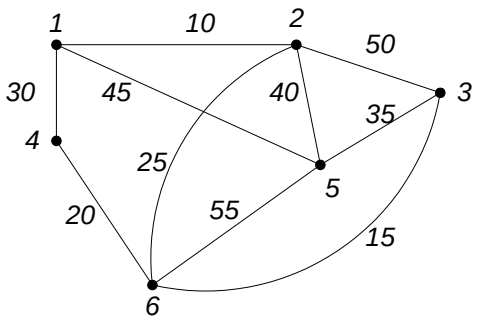
$$\begin{aligned} f(1,\{2,3,4\}) &= \min\{c_{12} + f(2,\{3,4\}), c_{13} + f(3,\{2,4\}), \\ &\qquad c_{14} + f(4,\{2,3\})\} \\ &= \min\{10+25, 15+25, 20+23\} \\ &= 35 \end{aligned}$$

Setelah menemukan jarak terdekat yaitu 35, yang harus dilakukan adalah backtrack untuk mendapatkan solusi minimum. Backtrack dapat dilakukan seperti membangkitkan pohon.

Solusi minimum di atas didapat dari backtrack tahap-tahap yang telah dilakukan dalam Dynamic Programming. Dari tahap 4 diketahui bahwa yang menghasilkan jarak minimum yaitu dari 1 ke 2, kemudian dari tahap 3 diketahui yang menghasilkan jarak minimum yaitu dari 2 ke 4 dan seterusnya hingga mendapat solusi yang lengkap yang ditunjukkan oleh gambar diatas.

C. Penyelesaian Treasure Hunter dengan algoritma Branch & Bound dan Dynamic Programming

Tidak seperti dua persoalan diatas, persoalan Treasure Hunter memiliki beberapa kondisi dimana kedua algoritma diatas menjadi tidak efektif. Pertama, semua node tidak harus dikunjungi karena tidak di setiap node terdapat harta karun, sehingga membuat solusi menjadi tidak efektif. Kedua, tidak semua node dari persoalan ini terhubung, sehingga untuk sampai ke node tertentu dibutuhkan "jalan khusus" dan mungkin melewati sebuah node lebih dari sekali. Untuk itu, persoalan ini akan dibagi menjadi 2 buah persoalan. Persoalan pertama yaitu mengubah persoalan Treasure Hunter menjadi persoalan TSP murni, dengan mendefinisikan jarak terdekat antar node dan persoalan kedua yaitu menyelesaikan persoalan TSP murni dengan algoritma yang sudah ada. Ilustrasi persoalan Treasure Hunter sebagai berikut.



Bila disederhanakan menjadi matriks ketetanggaan maka akan menjadi seperti berikut.

$$\begin{bmatrix} \infty & 10 & \infty & 30 & 45 & \infty \\ 10 & \infty & 50 & \infty & 40 & 25 \\ \infty & 50 & \infty & \infty & 35 & 15 \\ 30 & \infty & \infty & \infty & \infty & 20 \\ 45 & 40 & 35 & \infty & \infty & 55 \\ \infty & 25 & 15 & 20 & 55 & \infty \end{bmatrix}$$

Dengan treasure ada di 2, 3, 4, 5 dan node awal ada di 1.

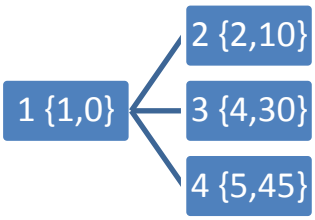
Untuk itu penulis akan menyelesaikan permasalahan ini dengan 2 tahap yang sudah dijelaskan. Tahap pertama akan diselesaikan menggunakan algoritma Branch &

Bound dan tahap kedua akan diselesaikan dengan algoritma Dynamic Programming.

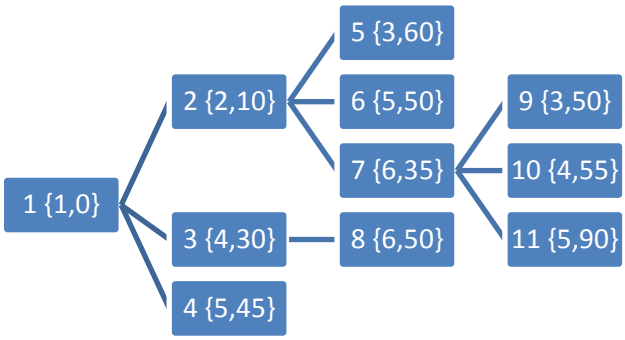
Tahap pertama, mencari jarak terdekat node terkait dengan node terkait lainnya. Node terkait disini dimaksudkan pada node-node yang membuat solusi menjadi efektif, yaitu node rumah (node 1), dan node yang berisikan harta karun (node 2, 3, 4, 5), lalu membuat matriks ketetanggaan dari kelima node terkait tersebut yang telah menjadi persoalan TSP murni untuk diselesaikan dengan algoritma Dynamic Programming.

Mencari jarak terdekat node 1 dengan node terkait lainnya.

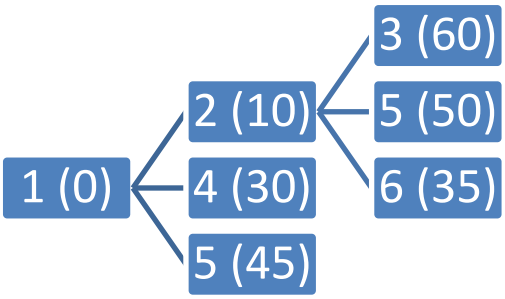
Jarak node 1 ke node 2



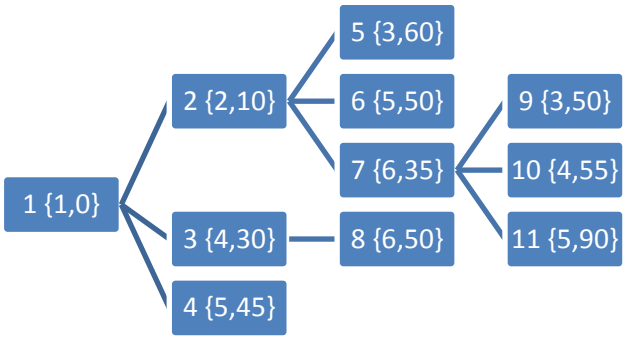
Jarak node 1 ke node 3



Jarak node 1 ke node 4



Jarak node 1 ke node 5



Dari algoritma Branch & Bound di atas, (hanya dari node 1 yang digambarkan), didapat hasil sebagai berikut.

Node asal	Node tujuan	Jalur	Cost
1	2	1-2	10
1	3	1-2-6-3	50
1	4	1-4	30
1	5	1-5	45
2	3	2-6-3	40
2	4	2-1-4	40
2	5	2-5	40
3	4	3-6-4	35
3	5	3-5	35
4	5	4-6-3-5	70

Setelah tahap 1 selesai, dimana kita mengetahui semua jarak terdekat antar node terkait, kita bisa membentuk matriks ketetanggaan baru sebagai berikut.

$$\begin{bmatrix} \infty & 10 & 50 & 30 & 45 \\ 10 & \infty & 40 & 40 & 40 \\ 50 & 40 & \infty & 35 & 35 \\ 30 & 40 & 35 & \infty & 70 \\ 45 & 40 & 35 & 70 & \infty \end{bmatrix}$$

Dari matriks ketetanggaan tersebut dapat kita lihat, semua pengecualian persoalan sudah dihilangkan. Semua node sekarang adalah node terkait (node penting) dan semua node "terhubung", walaupun sebenarnya terhubung tidak secara langsung (melalui node lain).

Tahap dua yaitu menyelesaikan persoalan TSP yang sudah disederhanakan dengan algoritma Dynamic Programming.

Tahap 1

$f(i,\phi) = c \quad , 2 \le i \le n$

$f(2,\phi) = c_{21} = 10$

$f(3,\phi) = c_{31} = 50$

$f(4,\phi) = c_{41} = 30$

$f(5,\phi) = c_{51} = 45$

Tahap 2

$f(i,S) = \min\{c_{ij} + f(j, S-\{j\})\}$

$$f(2,\{3\}) = \min\{c_{23} + f(3,\phi)\} = \min\{40+50\} = 90$$
$$f(2,\{4\}) = \min\{c_{24} + f(4,\phi)\} = \min\{40+30\} = 70$$
$$f(2,\{5\}) = \min\{c_{25} + f(5,\phi)\} = \min\{40+45\} = 85$$
$$f(3,\{2\}) = \min\{c_{32} + f(2,\phi)\} = \min\{40+10\} = 50$$
$$f(3,\{4\}) = \min\{c_{34} + f(4,\phi)\} = \min\{35+30\} = 65$$
$$f(3,\{5\}) = \min\{c_{35} + f(5,\phi)\} = \min\{35+45\} = 80$$
$$f(4,\{2\}) = \min\{c_{42} + f(2,\phi)\} = \min\{40+10\} = 50$$
$$f(4,\{3\}) = \min\{c_{43} + f(3,\phi)\} = \min\{35+50\} = 85$$
$$f(4,\{5\}) = \min\{c_{45} + f(5,\phi)\} = \min\{70+45\} = 115$$
$$f(5,\{2\}) = \min\{c_{52} + f(2,\phi)\} = \min\{40+10\} = 50$$
$$f(5,\{3\}) = \min\{c_{53} + f(3,\phi)\} = \min\{35+50\} = 85$$
$$f(5,\{4\}) = \min\{c_{54} + f(4,\phi)\} = \min\{70+30\} = 100$$

Tahap 3

$$\begin{aligned} f(2,\{3,4\}) &= \min\{c_{23} + f(3,\{4\}), c_{24} + f(4,\{3\})\} \\ &= \min\{40+65, 40+85\} \\ &= 105 \end{aligned}$$
$$\begin{aligned} f(2,\{3,5\}) &= \min\{c_{23} + f(3,\{5\}), c_{25} + f(5,\{3\})\} \\ &= \min\{40+80, 40+85\} \\ &= 120 \end{aligned}$$
$$\begin{aligned} f(2,\{4,5\}) &= \min\{c_{24} + f(4,\{5\}), c_{25} + f(5,\{4\})\} \\ &= \min\{40+115, 40+100\} \\ &= 140 \end{aligned}$$
$$\begin{aligned} f(3,\{2,4\}) &= \min\{c_{32} + f(2,\{4\}), c_{34} + f(4,\{2\})\} \\ &= \min\{40+70, 35+50\} \\ &= 85 \end{aligned}$$
$$\begin{aligned} f(3,\{2,5\}) &= \min\{c_{32} + f(2,\{5\}), c_{35} + f(5,\{2\})\} \\ &= \min\{40+85, 35+50\} \\ &= 85 \end{aligned}$$
$$\begin{aligned} f(3,\{4,5\}) &= \min\{c_{34} + f(4,\{5\}), c_{35} + f(5,\{4\})\} \\ &= \min\{35+115, 35+100\} \\ &= 135 \end{aligned}$$
$$\begin{aligned} f(4,\{2,3\}) &= \min\{c_{42} + f(2,\{3\}), c_{43} + f(3,\{2\})\} \\ &= \min\{40+90, 35+50\} \\ &= 85 \end{aligned}$$
$$\begin{aligned} f(4,\{2,5\}) &= \min\{c_{42} + f(2,\{5\}), c_{45} + f(5,\{2\})\} \\ &= \min\{40+85, 70+50\} \\ &= 120 \end{aligned}$$
$$\begin{aligned} f(4,\{3,5\}) &= \min\{c_{43} + f(3,\{5\}), c_{45} + f(5,\{3\})\} \\ &= \min\{35+80, 70+85\} \\ &= 115 \end{aligned}$$
$$\begin{aligned} f(5,\{2,3\}) &= \min\{c_{52} + f(2,\{3\}), c_{53} + f(3,\{2\})\} \\ &= \min\{40+90, 35+50\} \\ &= 85 \end{aligned}$$
$$\begin{aligned} f(5,\{2,4\}) &= \min\{c_{52} + f(2,\{4\}), c_{54} + f(4,\{2\})\} \\ &= \min\{40+70, 70+50\} \\ &= 110 \end{aligned}$$
$$\begin{aligned} f(5,\{3,4\}) &= \min\{c_{53} + f(3,\{4\}), c_{54} + f(4,\{3\})\} \\ &= \min\{35+65, 70+85\} \\ &= 100 \end{aligned}$$

Tahap 4

$$\begin{aligned} f(2,\{3,4,5\}) &= \min\{c_{23} + f(3,\{4,5\}), c_{24} + f(4,\{3,5\}), \\ &\qquad c_{25} + f(5,\{3,4\})\} \\ &= \min\{40+135, 40+115, 40+100\} \\ &= 140 \end{aligned}$$
$$\begin{aligned} f(3,\{2,4,5\}) &= \min\{c_{32} + f(2,\{4,5\}), c_{34} + f(4,\{2,5\}), \\ &\qquad c_{35} + f(5,\{2,4\})\} \\ &= \min\{40+140, 35+120, 35+110\} \\ &= 145 \end{aligned}$$
$$\begin{aligned} f(4,\{2,3,5\}) &= \min\{c_{42} + f(2,\{3,5\}), c_{43} + f(3,\{2,5\}), \\ &\qquad c_{45} + f(5,\{2,3\})\} \\ &= \min\{40+120, 35+85, 70+85\} \\ &= 120 \end{aligned}$$
$$\begin{aligned} f(5,\{2,3,4\}) &= \min\{c_{52} + f(2,\{3,4\}), c_{53} + f(3,\{2,4\}), \\ &\qquad c_{54} + f(4,\{2,3\})\} \\ &= \min\{40+105, 35+85, 70+85\} \\ &= 120 \end{aligned}$$

Tahap 5

$$\begin{aligned} f(1,\{2,3,4,5\}) &= \min\{c_{12} + f(2,\{3,4,5\}), c_{13} + f(3,\{2,4,5\}), \\ &\qquad c_{14} + f(4,\{2,3,5\}), c_{15} + f(5,\{2,3,4\})\} \\ &= \min\{10+140, 50+145, 30+120, 45+120\} \\ &= 150 \end{aligned}$$

Jadi didapat jarak minimum yaitu 150. Solusi bisa didapatkan melalui backtrack ke proses yang sudah dilakukan sebelumnya.

IV. KESIMPULAN

Untuk menyelesaikan masalah jarak terpendek yang tidak sederhana, kita dapat memisahkan masalah tersebut menjadi beberapa masalah yang lebih sederhana sehingga dapat dikerjakan dengan cara yang sudah ada ataupun algoritma-algoritma yang udah ada, kemudian solusinya bisa didapat dengan menyatukan solusi-solusi yang didapat dari setiap masalah atau tahap. Pada masalah Treasure Hunter ini sudah ditunjukkan permasalahan TSP (The Salesperson Problem) yang tidak sederhana dapat diselesaikan dengan 2 tahap yang sederhana.

V. REFERENSI

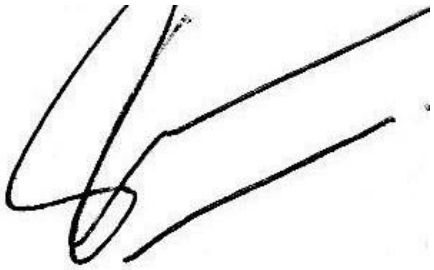
[1] Munir, Rinaldi, Diktat Kuliah IF 2211 Strategi Algoritma.Program Studi Teknik Informatika ITB, 2013

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2014

ttd



Marcelinus Henry M. 13512082