Penerapan *Dynamic Programming* pada *Route Planning*

Vincent Tanjaya 13518133
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518133@std.stei.itb.ac.id

Abstract*—Pada saat ini, *Route Planning* adalah salah satu hal yang paling menjadi perhatian oleh suatu maskapai penerbangan, karena hal tersebut dapat mempengaruhi berbagai konteks seperti keuntungan, bahan bakar, dan keselamatan. Dari pemecahan persoalan ini dengan menggunakan *Dynamic Programming* maka diharapkan solusi yang didapatkan optimal. Contoh persoalan *Route Planning* lainnya adalah *Travelling Salesman Problem* atau disingkat sebagai TSP, dan masih banyak persoalan lain yang dapat dioptimalisasi oleh *Dynamic Programming.

***Keywords—Dynamic Programming,* optimal, *Flight Planning, Travelling Salesman Problem, Brute Force, Greedy*.**

I. Pendahuluan

Route Planning adalah suatu perencanaan rute terbaik, yang dimana memiliki parameter-parameter tertentu. Contoh persoalan *Route Planning* diantaranya *Flight Planning* dan *Travelling Salesman Problem.* Pada *Flight Planning* memiliki aspek parameter seperti kegunaan bahan bakar, dan memiliki fungsi pembatas keselamatan. Arti fungsi pembatas keselamatan adalah jika rute tersebut sudah optimal tetapi tidak aman, maka rute tersebut tidak diambil. Pada *Travelling Salesman Problem* yang menjadi aspek parameter biasanya adalah jarak tempuh tiap kota, dan memiliki fungsi pembatas yaitu rute yang dilalui harus merupakan suatu sirkuit *Hamilton*. Arti fungsi pembatas pada *Travelling Salesman Problem* adalah dimana suatu kota hanya boleh tepat sekali dilewati, dan kota akhir yang dituju harus merupakan kota awal keberangkatan. Persoalan-persoalan di atas dapat diselesaikan menggunakan *Dynamic Programming*. *Dynamic Programming* adalah suatu algoritma yang menyelesaikan suatu persoalan dengan mempartisi suatu persoalan menjadi banyak sub persoalan dimana sub persoalan yang sudah dikerjakan tidak akan diulangi.

II. Landasan Teori

A. *Dynamic Programming*

Dynamic Programming adalah suatu algoritma yang menyelesaikan suatu persoalan dengan mempartisi suatu persoalan menjadi banyak sub persoalan dimana sub persoalan yang sudah dikerjakan sudah tidak akan dikerjakan kembali. Contoh persoalan *Dynamic Programming* adalah *Integer Knapsack*, *Travelling Salesman Problem, Flight Planning, Capital Budgeting Problem,* Penyelesaian Fibonacci ke n dalam waktu *O(n),* dan masih banyak lagi persoalan yang dapat diselesaikan dengan *Dynamic Programming.*

Cara kerja *Dynamic Programming* pada penyelesaian optimasi adalah mengubah suatu permasalahan menjadi sub masalah dan kemudian sub masalah itu dihitung nilai optimalnya dan kemudian digabungkan sehingga mendapat hasil yang optimal dimana dengan sub masalah yang sudah dikerjakan tidak akan dikerjakan kembali.

Karakteristik penyelesaian persoalan dengan *Dynamic Programming* adalah :

1. Terdapat banyak pilihan yang mungkin,
2. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya, jika belum ada solusi sebelumnya maka akan dibangun solusi dengan cara biasa,
3. Menggunakan persyaratan optimasi dan kendala untuk membatasi jumlah pilihan yang harus dipertambangkan pada suatu tahap.

Perbedaan *Dynamic Programming* dengan *Greedy* adalah *Greedy* hanya satu rangkaian yang dihasilkan sedang *Dynamic Programming* mempertimbangkan lebih dari satu rangkaian yang berarti solusi dari *Greedy* tidak selalu optimal, sedangkan hasil yang didapatkan dari *Dynamic Programming* selalu optimal dengan kompleksitas yang lebih mangkus dari *Brute Force.*

Pada *Dynamic Programming,* rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas. 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 sehingga persoalan pada tahap k tidak diulang.

Karakteristik persoalan *Dynamic Programming* :

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.

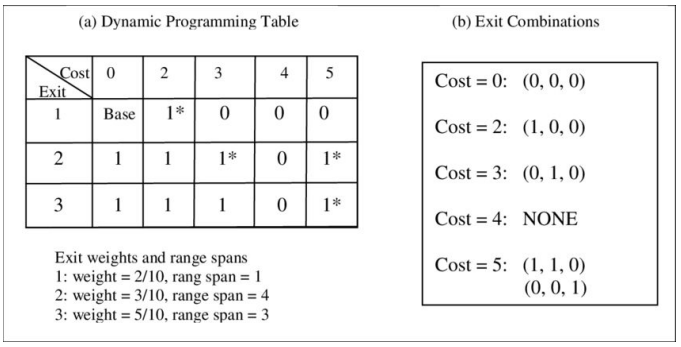
Dengan menggunakan *Dynamic Programming*, persoalan diselesaikan secara bertahap. Persoalan dengan *Dynamic Programming* dapat diselesaikan secara maju, yaitu mulai dari tahap 1, kemudian dilanjutkan ke tahap 2, 3, dan seterusnya hingga tahap n. Selain secara maju, persoalan dapat juga diselesaikan secara mundur, yaitu bahwa *Dynamic Programming* bergerak mulai dari tahap n, dilanjutkan mundur ke tahap n-1, n-2, dan seterusnya hingga tahap 1. Persoalan *Dynamic Programming* dapat diselesaikan secara mundur, yaitu bergerak mulai dari tahap n, terus mundur ke tahap n – 1, n – 2, dan seterusnya sampai tahap 1. Runtunan peubah keputusan adalah xn , xn-1 , …, x1 .

Prinsip optimalitas pada *Dynamic Programming* maju: ongkos pada tahap k +1 = (ongkos yang dihasilkan pada tahap k) + (ongkos dari tahap k ke tahap k + 1);k = 1, 2, …, n – 1.

Prinsip optimalitas pada *Dynamic Programming* mundur: ongkos pada tahap k = (ongkos yang dihasilkan pada tahap k + 1) + (ongkos dari tahap k + 1 ke tahap k);k = n, n – 1, …, 1.

Langkah-langkah pengembangan algoritma *Dynamic Programming* :

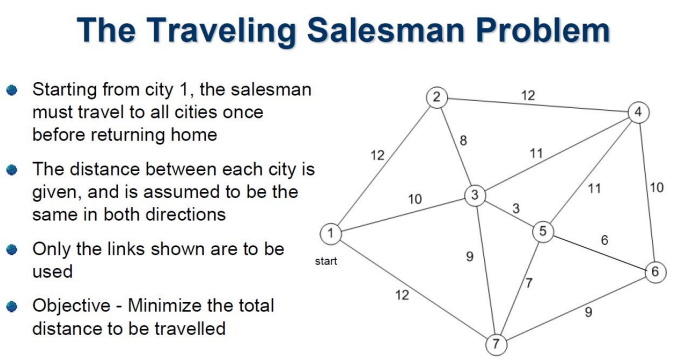
1. Karakteristikkan struktur solusi optimal.
2. Definisikan secara rekursif nilai solusi optimal.
3. Hitung nilai solusi optimal secara maju atau mundur.
4. Konstruksi solusi optimal. (Opsional)



Gambar 1. Contoh persoalan *Dynamic Programming.*

B. *Travelling Salesman Problem*

Persoalan *Travelling Salesman Problem* pertama diformulasikan pada tahun 1930 dan menjadi salah satu persoalan yang paling sering dipelajari dalam permasalahan optimasi. Persoalan *Traveling Salesman Problem* berbunyi seperti ini "Dengan serangkaian kota dan jarak antara setiap pasangan kota, masalahnya adalah menemukan rute terpendek yang mengunjungi setiap kota sekali dan kembali ke titik awal". *Travelling Salesman Problem* adalah persoalan dimana setiap kota yang dikunjungi harus berupa suatu sirkuit *Hamilton* dimana kota yang dikunjungi hanya boleh tepat sekali dan kota terakhir yang dituju adalah kota awal. Pada persoalan *Travelling Salesman Problem* mencari sirkuit *Hamilton* yang paling optimal dimana biasanya yang menjadi parameter optimal adalah jarak tempuh.

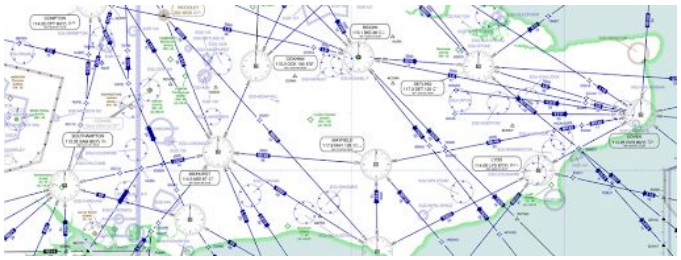


Gambar 2. Contoh Persoalan *Travelling Salesman Problem*

Travelling Salesman Problem dapat diselesaikan dengan *Dynamic Programming* dengan kompleksitas algoritma dalam notasi Big-O $O(n^2 2^n)$ dimana *Dynamic Programming* memberikan algoritma yang lebih mangkus daripada *Brute Force (O(n!))* meskipun hanya sedikit.

C. *Flight Planning*

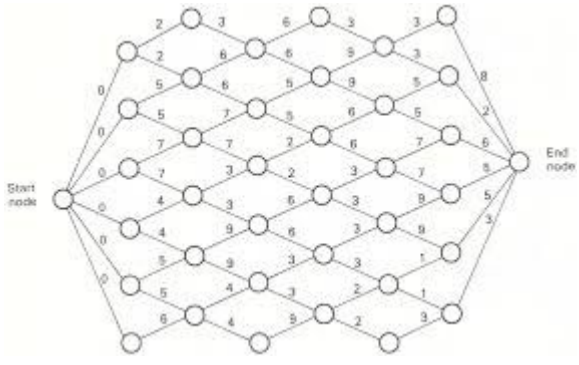
Flight Planning adalah suatu perencanaan suatu rute pesawat dalam penerbangan antar wilayah. Persoalan ini dapat diselesaikan dengan *Dynamic Programming.* Parameter untuk optimasi dalam *Flight Planning* adalah faktor keselamatan, bahan bakar, biaya dan masih banyak faktor yang lain.



Gambar 3. Contoh *Flight Plan*

Komponen-komponen *Flight Plan* :

1. jenis flight plan, yang menunjukkan aturan yang digunakan dalam menyusun flight plan; dapat berupa instrumen atau visual
2. identifikasi armada pesawat, yang menunjukkan nomor registrasi pesawat yang melakukan penerbangan (unik untuk setiap pesawat)
3. jenis pesawat dan peralatan khusus jika ada, misalnya perlengkapan tambahan seperti transponder
4. rencana kecepatan, dalam satuan knot
5. titik keberangkatan, yaitu bandar udara asal
6. waktu keberangkatan, sesuai jadwal dan waktu yang sebenarnya
7. ketinggian (altitude) terbang; dalam dunia penerbangan, ada bentuk diskrit per satuan ratus kaki menjadi flight level, disingkat FL
8. rute (jalur penerbangan) yang dilalui
9. destinasi, yaitu tujuan penerbangan
10. perkiraan waktu perjalanan
11. keterangan, untuk diperhatikan oleh kendali lalu lintas udara, misalnya tidak mau menerima aturan lepas landas SID (Standard Instrument Departure) atau aturan mendarat STAR (Standard Arrival Terminal Route)
12. jumlah bahan bakar yang dibawa
13. bandar udara alternatif terdekat
14. informasi pilot
15. jumlah orang dalam pesawat
16. warna pesawat, berguna dalam identifikasi ketika terjadi musibah
17. kontak darat di tujuan



Gambar 4. Graf ilustrasi penentuan suatu *Route Planning*

III. Implementasi Dan Pembahasan

A. *Travelling Salesman Problem*

Contoh persoalan *Travelling Salesman Problem* untuk jumlah kota 4.

$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

Tahap 1:

$f(i,null) = c_{i1}$, $2 <= i <= n$;

Diperoleh :

$f(2,null) = c_{21} = 5$;

$f(3,null) = c_{31} = 6$;

$f(4,null) = c_{41} = 8$;

Tahap 2:

$f(i,s) = \min\{c_{ij} + f(j, S - \{j\})$ untuk $|S| = 1$

Diperoleh:

$f(2, \{3\}) = \min\{c23 + f(3, null)\} = \min\{9 + 6\} = \min\{15\} = 15$

$f(2, \{4\}) = \min\{c24 + f(4, null)\} = \min\{10 + 8\} = \min\{18\} = 18$

$f(3, \{2\}) = \min\{c32 + f(2, null)\} = \min\{13 + 5\} = \min\{18\} = 18$

$f(3, \{4\}) = \min\{c34 + f(4, null)\} = \min\{12 + 8\} = \min\{20\} = 20$

$f(4, \{2\}) = \min\{c42 + f(2, null)\} = \min\{8 + 5\} = \min\{13\} = 13$

$f(4, \{3\}) = \min\{c43 + f(3, null)\} = \min\{9 + 6\} = \min\{15\} = 15$

Tahap 3 :

$f(i,s) = \min\{c_{ij} + f(j, S - \{j\})$ untuk $|S| = 2$ dan $i!=1$, $1 != S$ dan $i!=S$;

Diperoleh:

$f(2, \{3, 4\}) = \min\{c23 + f(3, \{4\}), c24 + f(4, \{3\})\}$

$= \min\{9 + 20, 10 + 15\}$

= min{29, 25}

= 25

f(3, {2, 4}) = min{c32 + f(2, {4}), c34 + f(4, {2})}

= min{13 + 18, 12 + 13}

= min{31, 25}

= 25

f(4, {2, 3}) = min{c42 + f(2, {3}), c43 + f(3, {2})}

= min{8 + 15, 9 + 18}

= min{23, 27}

= 23

Dengan menggunakan persamaan(1) maka diperoleh :

f(1, {2, 3, 4}) = min{c12 + f(2, {3, 4}), c13 + f(3, {2, 4}), c14 + f(4, {2, 3})}

= min{10 + 25, 15 + 25, 20 + 23}

= min{35, 40, 43}

= 35

Jadi bobot tur hasil *Dynamic Programming* adalah 35.

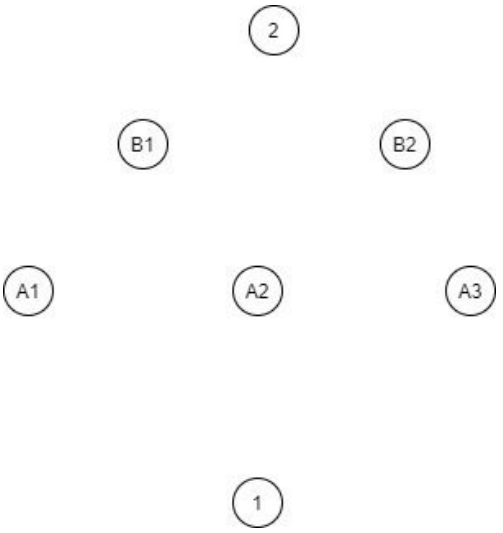
Menentukan lintasan yang dilalui untuk mendapatkan bobot tur 35, dengan rekonstruksi persoalan :

1. Tinjau pada setiap f(i, S) nilai j yang meminimumkan persamaan (2)
2. Misalkan J(i, S) adalah nilai yang dimaksudkan tersebut. Maka, J(1, {2, 3, 4}) = 2. Jadi, tur mulai dari simpul 1 selanjutnya ke simpul 2.
3. Simpul berikutnya dapat diperoleh dari f(2, {3, 4}), yang mana J(2, {3, 4}) = 4. Jadi, simpul berikutnya adalah simpul 4.
4. Simpul terakhir dapat diperoleh dari f(4, {3}), yang mana J(4, {3}) = 3. Jadi, tur yang optimal adalah 1, 2, 4, 3, 1 dengan bobot (panjang) = 35.

Dari hasil analisis diatas maka lintasan yang diperoleh adalah sebagai berikut :

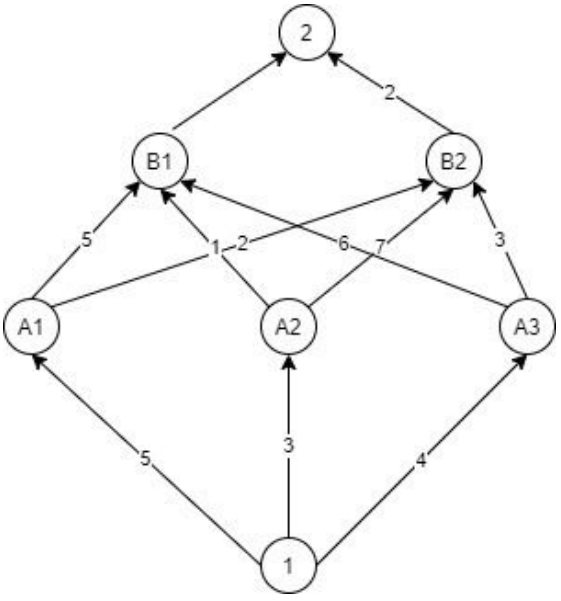
1->2->4->3->1 dengan bobot 35

B. Flight Planning



Gambar 5. Contoh Persoalan *Flight Planning*

Misalkan pada gambar diatas terdapat wilayah awal berupa 1 dan tujuan akhir berupa 2 dimana untuk mencapai tujuan akhir harus melewati salah satu tahap A dan salah satu tahap B, dimana semua tahap ini sudah diverifikasi dengan fungsi pembatas keselamatan. Gambar dibawah merupakan rincian dari persoalan diatas serta bobot (*cost*) pada setiap jalur



Gambar 6. Graf keterhubungan antar tahap

Persoalan diatas akan diselesaikan menggunakan *Dynamic Programming*. *Dynamic Programming* yang akan digunakan adalah *Dynamic Programming* maju. Pada persoalan diatas terdapat tiga tahap. Nilai keputusan terbaik dapat dinyatakan sebagai relasi rekurens sebagai berikut:

$$f_1(s) = c_{x1s} \text{ (basis)}$$

$$f_k(s) = min\{c_{xkS} + f_{k-1}(xk)\}\ k = 2,3; \text{ (Rekurens)}$$

Penyelesaian persoalan :

Tahap 1:

$$f_1(s) = c_{x1s} \text{ (basis)}$$

S	Solusi Optimal	
	$f_1(s)$	x_1
A1	5	D
A2	3	D
A3	4	D

Tahap 2 :

$$f_2(s) = min\{c_{x2S} + f_1(x2)\}$$

S	Solusi Optimal	
	$f_2(s)$	x_2
B1	4	A2
B2	7	A1,A3

Tahap 3 :

$$f_3(s) = min\{c_{x3S} + f_2(x3)\}$$

S	Solusi Optimal	
	$f_3(s)$	x_3
2	5	B1

Maka dapat terlihat bahwa cost terbaik yang diperlukan untuk menempuh dari 1 ke 2 adalah 5.

Rekonstruksi penyelesaian, terlihat pada solusi optimal node yang sebelumnya dilalui adalah B1, dan pada B1 node yang sebelumnya dilalui adalah A2 dan node sebelumnya lagi adalah D. Maka solusi optimalnya adalah:

1->A2->B1->2 dengan cost 5

IV. Kesimpulan

Dynamic Programming adalah salah satu algoritma yang dapat menyelesaikan suatu persoalan *Route Planning* seperti *Travelling Salesman Problem, Flight Planning,* dan persoalan-persoalan lainnya seperti *Integer Knapsack,* Penyelesaian Fibonacci ke n dalam waktu linear. *Dynamic Programming* biasanya mengubah suatu persoalan menjadi sub persoalan yang dimana sub persoalan tersebut hanya diulang sekali yang dimana jelas dapat memiliki kompleksitas yang lebih mangkus dibandingkan dengan *Brute Force* karena pada *Brute Force* mengulangi sub persoalan yang telah dikerjakan sebelumnya.

Dynamic Programming dapat diselesaikan dengan cara *Dynamic Programming* maju dan *Dynamic Programming* mundur. *Dynamic Programming* juga menggunakan prinsip optimalitas. 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 sehingga persoalan pada tahap k tidak diulang.

Video Link at Youtube

https://youtu.be/BuHDZjeDeMw

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena hanya atas berkat-nya makalah berjudul "Penerapan *Dynamic Programming* pada *Route Planning*" dapat selesai. Penulis juga berterima kasih kepada dosen mata kuliah IF2220 Strategi Algoritma, pada Semester II Tahun 2019/2020 atas ilmu-ilmu yang telah diberikan kepada penulis.

Referensi

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2019-2020/Makalah2019/13518133.pdf diakses pada 3 Maret 2020 pukul 11:00

[2] https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/ diakses pada 3 Maret 2020 pukul 12:30

[3] https://www.researchgate.net/publication/306347541_FLIGHT_PLANNING diakses pada 3 Maret 2020 pukul 12:45

[4] https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial/ diakses pada 3 Maret 2020 pukul 13:00

[5] https://blog.usejournal.com/top-50-dynamic-programming-practice-problems-4208fed71aa3 diakses pada 3 Maret 2020 pukul 15:00

[6] http://www.math.uwaterloo.ca/tsp/ diakses pada pukul 19:00

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, 29 April 2020

Vincent Tanjaya, 13518133