Traveling Salesman Problem

Aulia Rahma Amin[1], Mukhamad Ikhsan[2], Lastiko Wibisono[3]

Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13009@students.if.itb.ac.id[1],
if13033@students.if.itb.ac.id[2], if13051@students.if.itb.ac.id[3]

Abstrak

Permasalahan TSP *(Traveling Salesman Problem*) adalah permasalahan dimana seorang *salesman* harus mengunjungi semua kota dimana tiap kota hanya dikunjungi sekali, dan dia harus mulai dari dan kembali ke kota asal. Tujuannya adalah menentukan rute dengan jarak total atau biaya yang paling minimum. Permasalahan TSP merupakan permasalahan yang memang mudah untuk diselesaikan dengan algoritma *Brute Force*, tetapi hal itu hanya dapat dilakukan dengan jumlah kota atau simpul yang tidak banyak. Kompleksitas algoritma untuk permasalahan TSP dengan algoritma *Brute Force* adalah O(n!) dengan catatan n adalah jumlah kota atau simpul dan setiap kota atau simpul terhubung dengan semua kota atau simpul lainnya. Dengan jumlah sebanyak 20 kota, maka banyak sirkuit Hamilton yang mungkin adalah sebanyak 6 x 10^{16}.

***Kata kunci**: traveling salesman problem, metaheuristics*

1. Pendahuluan

Selain masalah sarana transportasi, efisiensi pengiriman surat atau barang ditentukan pula oleh lintasan yang diambil untuk mengirimkan surat atau barang tersebut. Oleh karena itu solusi optimal dari permasalahan TSP ini, akan sangat membantu perusahaan pegiriman surat atau barang untuk mengefisienkan proses pengiriman barang, baik dari segi waktu maupun dana.

Hingga kini kompleksitas algoritma permasalahan TSP masih tidak dapat diketahui pasti, bahkan setelah 50 tahun lebih pencarian. Hal tersebut menjadikan TSP menjadi salah satu permasalahan yang hingga kini belum terselesaikan dalam banyak permasalahan optimasi matematis.

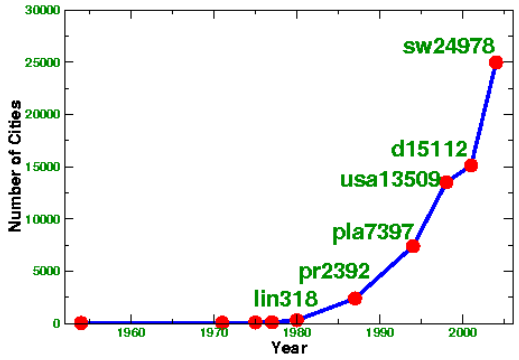
2. Sejarah Permasalahan TSP

Permasalahan matematika tentang *Traveling Salesman Problem* dikemukakan pada tahun 1800 oleh matematikawan Irlandia William Rowan Hamilton[1] dan matematikawan Inggris Thomas Penyngton[2]. Gambar dibawah ini adalah foto dari permainan Icosian Hamilton yang membutuhkan pemain untuk menyelesaikan perjalanan dari 20 titik menggunakan hanya jalur-jalur tertentu.

Diskusi mengenai awal studi dari Hamilton dan Kirkman dapat ditemukan di *Graph Theory 1736-1936*[3] oleh N. L. Biggs, E. K. LLoyd, dan R. J. Wilson, Clarendon Press, Oxford, 1976.

Bentuk umum dari TSP pertama dipelajari oleh para matematikawan mulai tahun 1930. Diawali oleh Karl Menger[4] di Vienna dan Harvard. Setelah itu permasalahan TSP dipublikasikan oleh Hassler Whitney[5] dan Merrill Flood[6] di Princeton. Penelitian secara detail dari hubungan antara Menger dan Whitney, dan perkembangan TSP sebagai sebuah topik studi dapat ditemukan di makalah Alexander Schrijver's[7] *"On the history of combinatorial optimization (till 1960)"*[8]

3. Perkembangan Pemecahan TSP



Gambar 1. Grafik Pemecahan permasalahan TSP dengan n-kota terhadap tahun diselesaikannya

Untuk menilai apakah kita melakukan perkembangan dalam menyelesaikan permasalahan TSP, kita pasti menilai dari jumlah waktu yang makin berkurang. Misal kita memiliki metode baru A yang lebih cepat menyelesaikan permasalahan TSP dibanding metode B, kita akan menilai bahwa kita telah menemukan solusi lebih baik. Tetapi masalah perangkingan untuk metode ini akan sangat sulit dilakukan, karena metode-metode yang sangat berkaitan erat satu sama lain tidak dapat dinilai hanya melalui perbandingan yang sederhana (permasalahan-permasalahan TSP yang ringan).

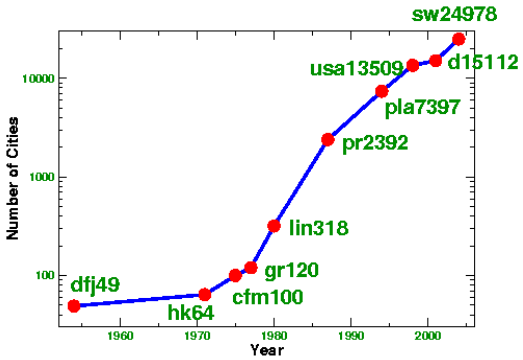
Oleh karena itu, untuk memutuskan apakah metode A lebih baik dibandingkan metode B, Kita harus mengesampingkan hasil dari contoh-contoh kasus yang sederhana dan dapat diselesaikan oleh hampir semua metode pemecahan permasalahan TSP. Saat ini kita seharusnya berkonsentrasi pada permasalahan-permasalahan yang benar-benar sulit yang sangat sulit terpecahkan sampai saat ini. Dari permasalahan-permasalahan yang sangat sulit ini, kita gunakan metode A dan metode B untuk menyelesaikannya. Setelah itu kita dapat memutuskan metode A lebih baik dari metode B jika untuk setiap permasalahan n-kota dengan n bilangan yang besar, metode A lebih cepat dalam menyelesaikan permasalahan dibanding metode B.

Agar perbandingan metode-metode TSP ini dapat diaplikasikan, kita dapat menganalisis solusi-solusi dari sebuah metode dan menjamin bahwa setiap n akan memakan sejumlah waktu f(n). Jadi f(n) fungsi waktu dari sebuah metode terhadap jumlah kota (n). Untuk membandingkan dua buah metode, kita cukup membandingkan fungsi f(n) dari masing-masing metode. Hal ini tentu saja dapat menghasilkan perhitungan yang salah ketika ada metode yang baik tapi dianalisis dengan buruk sehingga menghasilkan f(n) yang buruk. Dalam banyak permasalahan komputasional, studi tentang *algoritma+fungsi* telah menghasilkan solusi matematika yang sangat bagus, yang penting dalam pengembangan untuk penyelesaian permasalahan praktis. Hal ini telah menjadi subjek studi yang utama dalam sains komputer.

Untuk setiap permasalahan TSP akan sangat mudah kita katakan bahwa fungsi f(n)-nya adalah *(n-1)*!=*n-1* x *n-2* x *n-3* x ...x 3 x 2 x 1 dan jumlah jalur yang mungkin terjadi adalah *(n-1)*!/2. Hasil yang lebih baik ditemukan pada tahun 1962 oleh Michael Held dan Richard Karp, yang menemukan algoritma dan fungsi yang mempunyai proporsi $n^2 2^n$, yaitu n x n x2 x 2 x … x 2, dimana ada sebanyak n perkalian dua. Untuk setiap n bernilai besar, fungsi f(n) Held-Karp akan selalu lebih kecil dibandingkan *(n-1)*!.

Untuk setiap orang yang tertarik untuk menyelesaikan permasalahan TSP dalam jumlah besar, ada sebuah berita buruk bahwa selama 43 tahun sejak Held dan Karp menemukan fungsi f(n)= $n^2 2^n$ ternyata tidak ditemukan fungsi f(n) yang lebih baik lagi[9]. Hal ini tentu saja sangat mengecewakan karena dengan *n*=30 fungsi f(n) Held-Karp menghasilkan nilai yang sangat besar, dan untuk *n*=100 merupakan sesuatu yang mustahil untuk diselesaikan oleh kemampuan komputer saat ini.

Perkembangan fungsi f(n) dalam TSP yang sangat lambat ini mungkin memang tidak dapat kita hindari; dengan computer saat ini bisa jadi memang tidak ada metode yang menghasilkan f(n) yang mempunyai performansi yang baik, misal n^c dimana *c* adalah sebuah angka konstanta, oleh karena itu, n x n x n x … x n dimana *n* muncul sebanyak *c* kali. Diskusi mengenai teknis permasalahan ini dapat anda lihat dalam makalah Stephen Cook's[10] dan Institut Matematika Clay menawarkan hadiah sebesar satu juta dola US bagi siapa saja yang dapat menemukannya.



Gambar 2. perkembangan pemecahan TSP dengan skala logaritma terhadap jumlah kota

Dari gambar ini kita dapat melihat bahwa perkembangan pemecahan masalah TSP telah dicapai selama lebih dari 30 tahun kebelakang. Jika hal ini terus berlanjut, kita dapat memperkirakan bahwa untuk 30 tahun kedepan kita dapat

menyelesaikan permasalahan TSP untuk jumlah kota yang berjumlah jutaan. Riset untuk menyelesaikan permasalahan 1,9 juta kota dapat anda temukan di http://www.tsp.gatech.edu/world/index.html.

3.1 Perkembangan Penyelesaian TSP

a. 49Kota - *DANTZIG49*

DANTZIG49 adalah permasalahan yang diteliti oleh Dantzig, Fulkerson, dan Johnson yang dapat kita lihat dalam makalah mereka tahun 1954 tentang solusi dari permasalahan TSP yang terdiri dari satu kota tiap 48 negara bagian di Amerika Serikat ditambah kota Washington, D. C. Para penulis bekerja dengan 49 kota tersebut. Jalur optimal dari 49 kota ini, menggunakan jalan pintas yang terdapat pada 7 kota selain 49 kota tersebut. Jalur yang dibuat berdasarkan jarak pada setiap jalur dalam kota diantara 49 kota tersebut. Para penulis mengatakan bahwa mereka mendapat tabel jarak dari Bernice Brown karyawan Perusahaan Rand.

b. 120 Kota - *GR120*

GR120 telah menjadi pengujian standar bagi permasalahan TSP sejak tahun 1977. GR120 ini mempunyai 120 titik yang terdiri dari jarak tempuh antara 120 kota yang terdapat di sekitar Jerman. Daftar kota-kota ini terdapat pada Atlas Umum Negara Jerman tahun 1967/68.

c. 318 Kota - *LIN318*

LIN318 muncul pada tahun 1973 dalam makalah yang ditulis oleh S.Lin dan B.W. Kernighan. Data dari LIN318 terdiri dari 318 yang muncul dari hasil pengeboran, dimana bornya adalah sebuah sinar laser. Lin dan Kernighan menulis bahwa mereka mendapatkan datanya dari R.Haberman. Permasalahan ini pertama diselesaikan oleh H.Crowder dan M. W. Padberg pada tahun 1980.

d. 532 Kota - *ATT532*

ATT532 muncul pada tahun 1987 didalam makalah yang ditulis oleh M. Padberg dan G. Rinaldi. Dalam ATT532 terdapat data mengenai 532 kota yang berlokasi di Benua Amerika Serikat, dan Padberg serta Rinaldi menulis bahwa mereka mendapatkan permasalahannya dari Shen Lin karyawan dari laboratorium AT&T Bell.

e. 666 Kota - *GR666*

GR666 pertama kali diselesaikan oleh O. Holland dan M. Groetschel, yang muncul dalam Tesis PhD Olaf Holland's pada tahun 1987. Data terdiri dari 666 kota menarik yang tersebar di seluruh dunia.

f. 2392 Kota - *PR2392*

PR2392 memiliki data sebanyak 2,392 titik, hasil kontribusi dari M. Padberg dan G. Rinaldi. Layout dari titik-titik tersebut dibuat oleh Perusahaan Tektronics.

g. 7397 Kota - *PLA7397*

PLA7397 adalah sebuah *programmed logic array application*, yang terdiri dari 7,397 kota.

h. 15112 Kota - *D15112*

D15112.ini mempunyai data mengenai 15,112 kota-kota yang terdapat di Negara Jerman.

i. 24978 Kota – *Sweded24978*

24,978 kota di Swedia, didapat datanya dari National Imagery and Mapping Agency database nama-nama fitur geografi.

4. Prosedur Sederhana Pemecahan TSP

Dalam penyelesaian masalah TSP kita dapat membagi kedalam 2 metode, yaitu metode optimal dan metode aproksimasi. Metode optimal akan menghasilkan hasil yang optimal (minimum) sedangkan metode aproksimasi akan menghasilkan hasil yang mendekati optimal.

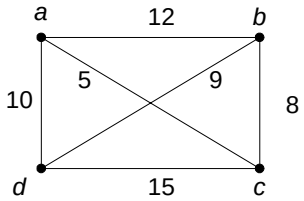
4.1 Metode Optimal

Sejak permasalahan TSP ditemukan pada tahun 1800 oleh matematikawan Irlandia Sir William Rowan Hamilton dan matematikawan Inggris Thomas Penyngton Kirkman, pusat perhatian studi ini adalah menemukan secara pasti nilai minimum dari persoalan TSP dengan konsekuensi dibutuhkan waktu yang cukup lama untuk menyelesaikannya.

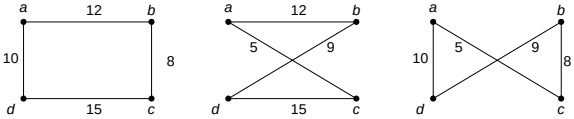
4.1.1 Complete Enumeration

Metode ini akan mengenumerasi setiap kemungkinan yang terdapat dalam graf, setelah itu algoritma ini akan membandingkan lintasan mana yang paling minimum

Misal untuk kasus berikut ini :



jumlah titik yang terdapat adalah empat buah, dan banyak kemungkinan lintasannya adalah 3 buah. Yaitu :



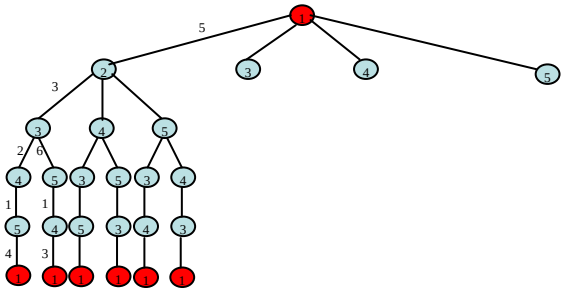
Lintasan pertama = (*a*, *b*, *c*, *d*, *a*) atau (*a*, *d*, *c*, *b*, *a*)
Mempunyai panjang = 10 + 12 + 8 + 15 = 45

Lintasan Kedua = (*a*, *c*, *d*, *b*, *a*) atau (*a*, *b*, *d*, *c*, *a*)
Mempunyai panjang = 12 + 5 + 9 + 15 = 41

Lintasan Ketiga = (*a*, *c*, *b*, *d*, a) atau (*a*, *d*, *b*, *c*, *a*)
Mempunyi panjang = 10 + 5 + 9 + 8 = 32

Dari hasil enumerasi ini didapat hasil minimum yaitu 32. Tetapi jumlah enumerasi dari algoritma ini adalah *(n – 1)*! yang tidak akan efisien jika jumlah *n* bernilai sangat besar.

4.1.2 Branch and Bound



Sama dengan complete enumeration, pada algoritma Branch and Boundpun ternyata memiliki kompleksitas algoritma (n-1)!, dimana n adalam jumlah kota.

Untuk conotoh kasus diatas hasil yang dicapai adalah 15.

4.1.3 Dynamic Programming

Misalkan $G = (V, E)$ adalah graf lengkap berarah dengan sisi-sisi yang diberi harga $c_{ij} > 0$ untuk setiap i dan j adalah simpul-simpul di dalam V. Misalkan $|V| = n$ dan $n > 1$. Setiap simpul diberi nomor 1, 2, …, n.

Asumsikan perjalanan (tur) dimulai dan berakhir pada simpul 1.

Setiap tur pasti terdiri dari sisi (1, k) untuk beberapa $k \in V - \{1\}$ dan sebuah lintasan dari simpul k ke simpul 1.

Lintasan dari simpul k ke simpul 1 tersebut melalui setiap simpul di dalam $V - \{1, k\}$ tepat hanya sekali.

Prinsip Optimalitas: jika tur tersebut optimal maka lintasan dari simpul k ke simpul 1 juga menjadi lintasan k ke 1 terpendek yang melalui simpul-simpul di dalam $V - \{1, k\}$.

Misalkan $f(i, S)$ adalah bobot lintasan terpendek yang berawal pada simpul i, yang melalui semua simpul di dalam S dan berakhir pada simpul 1.

Nilai $f(1, V - \{1\})$ adalah bobot tur terpendek. Berdasarkan prinsip optimalitas tersebut, diperoleh hubungan rekursif sebagai berikut:

$$f(1, V - \{1\}) = \min_{2 \le k \le n} \{c_{1k} + f(k, V - \{1, k\})\} \quad (1)$$

Dengan merampatkan persamaan (1), diperoleh

$$f(i, \varnothing) = c_{i,1}, \quad 2 \le i \le n$$

(basis)

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

(rekurens) (2)

Persamaan (1) dapat dipecahkan untuk memperoleh $\{1\}$) jika kita mengetahui $f(k, V - \{1, k\})$ untuk semua pilihan nilai k. Nilai f tersebut dapat diperoleh dengan menggunakan persamaan (2).

Kita menggunakan persamaan(2) untuk memperoleh $f(i, S)$ untuk $|S| = 1$, kemudian kita dapat memperoleh $f(i, S)$ untuk $|S| = 2$, dan seterusnya. Bila $|S| = n - 1$, nilai i dan S ini diperlukan sedemikian sehingga $i \neq 1$, $1 \notin S$ dan $i \notin S$.

Tinjau persoalan TSP untuk $n = 4$:

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

Ini adalah matriks ketetanggan dari suatu graf yang akan kita cari lintasan terpendeknya.

Tahap 1:

$$f(i, \varnothing) = c_{i,1}, \quad 2 \le i \le n$$

Diperoleh:

$f(2, \varnothing) = c_{21} = 5;$
$f(3, \varnothing) = c_{31} = 6;$
$f(4, \varnothing) = c_{41} = 8;$

terdapat tiga kemungkinan yang dapat dijadikan lintasan pertama.

Tahap 2:

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

untuk $|S| = 1$

Diperoleh:

$f(2, \{3\}) = \min\{c_{23} + f(3, \varnothing)\} = \min\{9 + 6\} = \min\{15\} = 15$
$f(3, \{2\}) = \min\{c_{32} + f(2, \varnothing)\} = \min\{13 + 5\} = \min\{18\} = 18$
$f(4, \{2\}) = \min\{c_{42} + f(2, \varnothing)\} = \min\{8 + 5\} = \min\{13\} = 13$
$f(2, \{4\}) = \min\{c_{24} + f(4, \varnothing)\} = \min\{10 + 8\} = \min\{18\} = 18$
$f(3, \{4\}) = \min\{c_{34} + f(4, \varnothing)\} = \min\{12 + 8\} = \min\{20\} = 0$
$f(4, \{3\}) = \min\{c_{43} + f(3, \varnothing)\} = \min\{9 + 6\} = \min\{15\} = 15$

Tahap 3:

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

untuk $|S| = 2$ dan $i \neq 1$, $1 \notin S$ dan $i \notin S$.

Diperoleh:

$$f(2, \{3, 4\}) = \min\{c_{23} + f(3, \{4\}), c_{24} + f(4, \{3\})\}$$
$$= \min\{9 + 20, 10 + 15\}$$
$$= \min\{29, 25\} \quad = 25$$

$$f(3, \{2, 4\}) = \min\{c_{32} + f(2, \{4\}), c_{34} + f(4, \{2\})\}$$
$$= \min\{13 + 18, 12 + 13\}$$
$$= \min\{31, 25\} \quad = 25$$

$$f(4, \{2, 3\}) = \min\{c_{42} + f(2, \{3\}), c_{43} + f(3, \{2\})\}$$
$$= \min\{8 + 15, 9 + 18\}$$
$$= \min\{23, 27\} = 23$$

Dengan menggunakan persamaan (1) diperoleh:

$$f(1, \{2, 3, 4\}) = \min\{c_{12} + f(2, \{3, 4\}), c_{13} + f(3, \{2, 4\}), c_{14} + f(4, \{2, 3\})\}$$
$$= \min\{10 + 25, 15 + 25, 20 + 23\}$$
$$= \min\{35, 40, 43\} = 35$$

Jadi, bobot tur yang berawal dan berakhir di simpul 1 adalah 35.

Lintasan yang dilalui di dalam tur tersebut dapat direkonstruksi jika kita menyimpan pada setiap $f(i, S)$ nilai j yang meminimumkan ruas kanan persamaan (2).

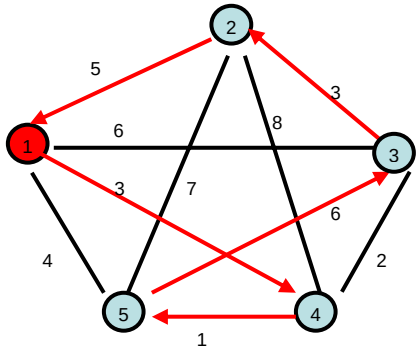
Misalkan $J(i, S)$ adalah nilai yang dimaksudkan tersebut. Maka, $J(1, \{2, 3, 4\}) = 2$. Jadi, tur mulai dari simpul 1 selanjutnya ke simpul 2.

Simpul berikutnya dapat diperoleh dari $f(2, \{3, 4\})$, yang mana $J(2, \{3, 4\}) = 4$. Jadi, simpul berikutnya adalah simpul 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.

4.2 Metode Aproksimasi

4.2.1 Greedy Heuristic



Pada algoritma ini, pemilihan lintasan akan dimulai pada lintasan yang memiliki nilai paling minimum, setiap mencapai suatu kota, algoritma ini akan memilih kota selanjutnya yang belum dikunjungi dan memiliki jarak yang paling minimum. Algoritma ini disebut juga *Nearest Neighbour*.

Kompleksitas algoritma ini memang sangat mengagumkan yaitu O(n), tetapi hasil yang kita dapat bisa sangat jauh dari hasil yang optimal, semakin banyak kota semakin besar pula perbedaan hasil yang dicapai. Misalnya untuk contoh kasus yang sama dengan algoritma *Branch and Bound* sebelumnya yang menghasilkan nilai 15, maka algoritma ini menghasilkan nilai 18 berbeda sebesar 20% dari hasil sebelumnya padahal jumlah kota hanya 5 buah.

4.3 Algoritma-algoritma lainnya :

4.3.1 Heuristics

Teknik ini digunakan untuk mencari jawaban dari masalah kombinatorial dengan secepat mungkin. Algoritma tradisional akan gagal ketika menghadapi permasalahan yang sangat rumit, seperti permasalahan TSP dengan jumlah kota *(n)* yang sangat besar.

Metode Heuristic memberikan pendekatan untuk menyelesaikan permasalahan optimasi kombinatorial. *Combinatorial search* akan memberi kita hasil yang mungkin dan mencari yang hasil yang mendekati optimal dari hasil-hasil tersebut. Tetapi mungkin memang tidak ada metode Heuristik yang menghasilkan solusi yang merupakan solusi optimal. Metode Heuristik seperti simulated annealing, genetic algoritm, dan neural network mengusahakan suatu cara untuk mencari hasil yang baik tapi bukan yang terbaik.

4.3.1.1 Genetic Algoritm

Algoritma Genetic mendapatkan inspirasi dari proses evolusi dan seleksi alam. Melalui proses seleksi alam, organisme atau mahluk hidup beradaptasi untuk mengoptimalkan kesempatan mereka untuk tetap bias hidup didalam sebuah lingkungan. Mutasi secara random menghasilkan kode genetik dari mahluk hidup, yang kemudian diturunkan kepada anaknya. Jika mutasi meningkatkan kualitas genetik, maka secara otomatis sang anak akan terbantu untuk selamat.

4.3.1.2 Simulated Annealing

Inspirasi tentang metode ini datang dari proses fisika mengenai pendinginan lelehan material yang berubah menjadi padat. Ketika lelehan besi didinginkan terlalu cepat, maka retakan dan gelembung udara akan muncul, menghancurkan permukaan dan integritas strukturnya. Oleh karena itu untuk menghasilkan besi yang baik, besi harus

didinginkan secara pelan-pelan dan diberi jeda. *Annealing* adalah teknik metalurgi yang meggunakan ilmu penjadwalan proses pendinginan untuk menghasilkan efisiensi dalam menggunakan energi dan menghasilkan besi yang optimal.

4.3.1.3 Neural Network

Neural Network adalah paradigma komputasional yang terinspirasi dari arsitektur otak manusia. Ketika otak memiliki intuisi yang sangat baik dalam memecahkan persoalan, maka mesinpun seharusnya dibangun seperti itu.

5. Aplikasi Permasalahan TSP

5.1 Gnome Sequencing

Pusat penelitian di the National Institute of Health menggunakan *Concorde's TSP solver* untuk membangun peta radiasi hybrid sebagai bagian dari dari proyek *genome sequencing*. TSP akan mengusahakan jalan untuk menyatukan peta local kedalam sebuah peta radiasi hybrid untuk sebuah genome. Laporan mengenai proyek ini dapat dilihat dari makalah *"A Fast and Scalable Radiation Hybrid Map Construction and Integration Strategy"* yang ditulis oleh R. Agarwala, D.L. Applegate, D. Maglott, G.D. Schuler, and A.A. Schaffler.

Aplikasi TSP ini telah diadaptasikan oleh orang-orang Perancis dalam menuliskan Peta genome dari tikus. Proyek ini dideskripsikan pada *"A Radiation Hybrid Transcript May of the Mouse Genome"*, Nature Genetics 29 (2001), pages 194--200

5.2 Starlight Interferometer Program

Tim insinyur dari Hernandez Engineering di Houston dan dari Universitas Brigham Young, telah melakukan sebuah eksperimen menggunakan *Chained Lin-Kerninghan* untuk mengoptimasi penggambaran urutan dari objek-objek luar angkasa. Hal ini dapat kita lihat dalam proposal *NASA Starlight space interferometer program*. Tujuan dari studi ini adalah meminimalisasikan penggunaan penggunaan bahan baker dalam pertargetan dan penggambaran dari maneuver sepasang satelit yang diterjunkan dalam sebuah misi (kota-kota dalam TSP digambarkan sebagai objek-objek luar angkasa, dan biaya perjalana antara satu kota denga lainnya dianalogikan dengan kebutuhan bahan baker yang dibutuhkan satelit). Laporan mengenai proyek ini dapat and abaca dalam makalah "Fuel Saving Strategies for Separated Spacecraft Interferometry".

5.3 Scan Chain Optimization

Sebuah manufaktur dalam semi-konduktor menggunakan *Concorde TSP Solver* untuk diimplementasikan dalam *Chained Lin-Kernighan heuristic* dalam eksperimen untuk mengoptimasi peng-scanan rantai dari IC (Integrated Circuit). *Scan chains* adalah sebuah perutean yang bias digunakan untuk tujuan pengetesan sebuah chip dan juga berguna untuk meminimalisasi baik dari segi waktu maupun tenaga.

6. Kesimpulan

TSP merupakan permasalahan yang klasik, yang muncul sejak tahun 1800, tetapi permasalahan ini akan terus menjadi objek studi yang menarik. Jika kita dapat menyelesaikan masalah-masalah TSP yang mempunyai *n* jutaan, maka proses-proses routing, pengiriman barang, dan permasalahan lainnya akan semakin efisien untuk dilakukan.

Daftar Pustaka

[1] http://www-groups.dcs.st-andrews.ac.uk/~history/Mathematicians/Hamilton.html

[2] http://www-groups.dcs.st-andrews.ac.uk/~history/Mathematicians/Kirkman.html

[3] http://www.oup.co.uk/isbn/0-19-853916-9

[4] http://www-groups.dcs.st-andrews.ac.uk/~history/Mathematicians/Menger.html

[5] http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Whitney.html

[6] http://infoshare1.princeton.edu:2003/libraries/firestone/rbsc/finding_aids/mathoral/pmc11.htm

[7] http://www.cwi.nl/~lex/

[8] http://www.cwi.nl/~lex/files/histco.ps

[9] http://www.win.tue.nl/~gwoegi/papers/exact.pdf

[10] http://www.claymath.org/millennium/P_vs_NP/Official_Problem_Description.pd

[11] http://www.tsp.gatech.edu

[12] Munir, Rinaldi. *Strategi Algoritmik bab Dynamic Programming*. Bandung.

[13] Munir, Rinaldi. *Matematka Diskrit bab Graf*. Bandung. Informatika ITB.